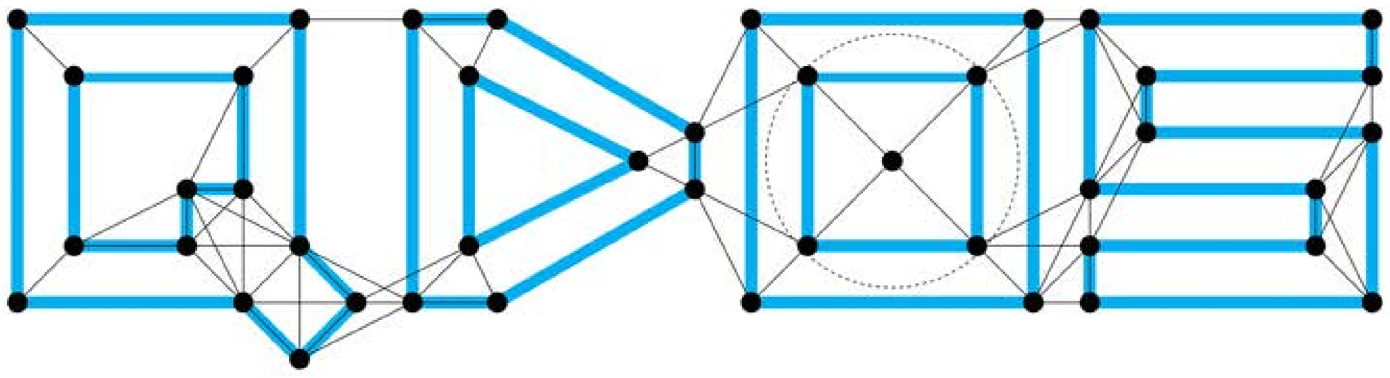
Quantized Distributed Optimization Schemes

A monotone operator approach



J.A.G. (Jake) Jonkman

TU Delft

CAS-MS-2017-05

Quantized Distributed Optimization Schemes

A monotone operator approach

by

J.A.G. (Jake) Jonkman

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday September 15th, 2017 at 13:30.

Student number:	4149831	
Faculty:	EEMCS	
Master Program:	Electrical Engineering	
Track:	Signals and Systems	
Project duration:	November 14th, 2016 – September 15th, 2017	
Thesis committee:	Dr. ir. R. Heusdens,	TU Delft, supervisor, Circuits And Systems Group
	Dr. ir. R.C. Hendriks,	TU Delft, Circuits And Systems Group
	Dr. ir. J.H. Weber,	TU Delft, Applied Mathematics Group

CAS-MS-2017-05

Circuits and Systems Group

The logo on the cover, QDOS, is an abbreviation of the title "Quantized Distributed Optimization Schemes". The image consists of a fully connected distributed network (black nodes), connected via a wireless connection (black edges) and wired connections (blue edges). The transmission range of one of the nodes is indicated using a dashed circle, based on the range of the node inside the O. The image is created using Matlab.

TU Delft

Abstract

Recently, the effects of quantization on the Primal-Dual Method of Multipliers were studied. In this thesis, we have used this method as an example to further investigate the effects of quantization on distributed optimization schemes in a much broader sense. Using monotone operator theory, the effect of quantization on all distributed optimization algorithms that can be cast as a monotone operator was researched for two different problem subclasses. The averaging problem was used as an example of a quadratic problem, while the Gaussian channel capacity problem was an example of the non-linear problem subclass. A fixed bit rate quantizer was used in combination with a dynamic cell width, to analyse the robustness of distributed optimization schemes against quantization effects. In particular, we have shown that for practical implementations it is possible to incorporate fixed bit rate quantization with dynamic cell width in a distributed optimization algorithm without loss of performance for both problem classes.

Preface

What you have just commenced reading is the Master Thesis “Quantized Distributed Optimization Schemes; a monotone operator approach”, in which the effects of quantization on distributed algorithms, such as the Primal-Dual Method of Multipliers, are studied. This work follows from the initial study “Quantization effects in PDMM: a first study for synchronous distributed averaging”, by Ir. Daan H.M. Schellekens.

This work has been written to obtain the degree of Master of Science at the Delft University of Technology (TU Delft). Over the period November 2016 - September 2017, I have researched, studied, coded, and composed this thesis. Under the guidance of my supervisor Dr. ir. Richard Heusdens, the basis and goals of my work were established, and with his help and that of my daily supervisor Thom Sherson, I achieved these goals in the best way possible.

I want to express my gratitude to both of my supervisors, who aided me through the rough patches, helped me get new ideas and were always prepared to answer my questions. I also want to thank my colleagues, in particular Niels van Wijngaarden, B.Sc., for the sparring, brainstorming, nice atmosphere in the room and fun we had during our projects. A special thank you goes out to my parents, who supported me when I needed it, provided distraction when necessary but also kept me focussed on my work.

I hope you enjoy reading my work.

J.A.G. (Jake) Jonkman
Delft, September 2017

Contents

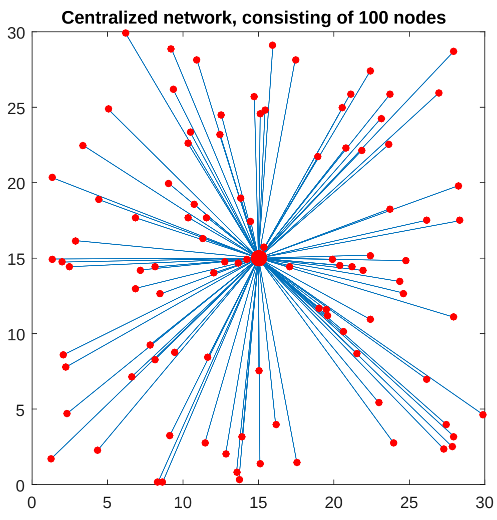
List of Figures

1 Introduction

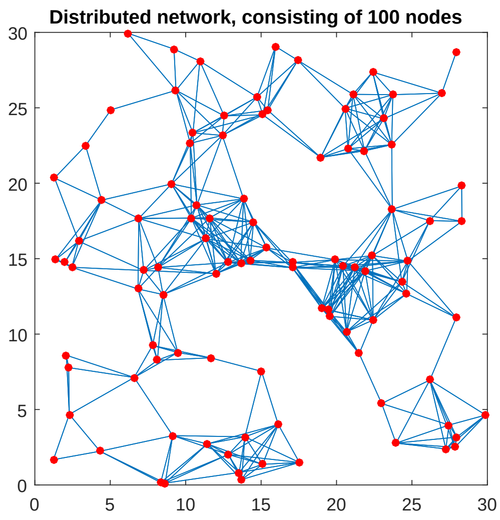
Due to new technology, such as faster and cheaper processing power and larger overall systems, the interest in the field of distributed signal processing has taken flight in the last few decades [1]. Examples are Big Data [2], a field where the amount of data is too large to be processed by a single computer, but also the developments in Smart Grids [3]. Following these changes to a node based network, the benefits of distributed optimization schemes became evident. The technological basis that supports distributed signal processing is that modern systems almost all consist of networks with a large number of nodes, which all contain one or more sensors, a processing unit, and communication abilities. This gives way to a new type of computation: distributed algorithms. Contrary to central processing, distributed processing will use the nodes in the grid to perform computations locally and only send required and processed information through the network, without the use of a central processor.

To indicate the difference between the traditional centralized network and the proposed distributed network, a graphical interpretation of both topologies is included in Fig. 1.1. It is clear from Fig. 1.1a that the amount of connections in the centralized case are equal to the amount of nodes minus one $(n-1)$ and the distances can be rather large. Therefore, the amount of edges is fixed, regardless of the node locations, and the distances depend on the layout of the network. Most decentralized algorithms require neighbour-based communication only, as indicated in Fig. 1.1b. This eases the constraints on transmission power, as the distances are smaller, and thus increases efficiency, although the number of connections depends on the topology. In general, a node will have more than one connection, which intuitively means the local information can be distributed by broadcasting to neighbouring nodes. As the connections are neighbour-based, it is easier to include or exclude nodes in the distributed case. For all those reasons, distributed schemes are automatically more scalable than conventional centralized systems and are also more robust against transmission errors [4]. Due to these advantages, these schemes are rising in popularity, and while for some methods the effects of quantization have been studied [5–7], the exact properties such as convergence rates and robustness are yet to be discovered for many other distributed methods. Therefore, these distributed optimization schemes will form the basis of this work.

Throughout the literature, distributed optimization problems are defined over a graph $G = (V, E)$, consisting of the nodes V and connecting edges E. In Fig. 1.1b, the nodes are indicated as red dots and the edges are blue lines. Most distributed optimization schemes can be categorized into two main classes: probabilistic inference and convex optimization. The former contains the well-known sum-product and max-sum algorithms, which solve the problem using belief propagation [8–10]. This method of solving works best for quadratic problems in combination with acyclic graphs. However, many practical problems cannot be implemented in a tree structure, or only in a very inefficient manner. In other words, the implemented graphs will often contain loops, resulting in convergence problems. The other category, convex optimization, does not have any constraints on the graph other than it being a connected graph (there has to be a path between any two nodes in the graph) [4]. In this thesis, the convex optimization approach will be discussed thoroughly.



(a) Centralized network topology with a central processing centre. All connections are between nodes and the processing centre.



(b) Distributed network topology without central processing centre. All connections are between neighbours within a specified range.

Figure 1.1: Different network topologies.

1.1. Distributed optimization problem

The general form of an optimization problem, defined over a graph $G = (V, E)$, is the following:

$$\text{minimize} \sum_{i\in V} f_i(\mathbf{x}_i) + \sum_{(i,j)\in E} f_{ij}(\mathbf{x}_i, \mathbf{x}_j) \tag{1.1}$$

For the convex optimization case, the assumption is made that $\{f_i\}$ and $\{f_{ij}\}$ are all convex functions. This is a valid assumption, as many optimization problems either are convex or can be relaxed to a convex form. One of the most frequently used problems is averaging. This averaging problem is an example of a distributable problem, for which many algorithms such as the gossip algorithm [11, 12], the *alternating direction method of multipliers* (ADMM) [13, 14] and the *primal-dual method of multipliers* (PDMM) [15–17] exist. Numerous other problems can be solved in a distributed manner as well.

The first step to decentralize a problem is to formulate the original problem as an optimization problem over a certain graph G, as in Eq. (1.1). However, the original problem is not always distributable, as the original functions may contain global variables, for example resulting from constraints. To solve this issue, the optimization problem can be recast as a decentralized problem by adding local variables and constraints, making it a distributed optimization problem. By choosing the best-suited distributed solver, the process is complete. An example of this process will be included in Chapter 2.

One of the interesting aspects of these distributed algorithms is the convergence rate. Conditions for convergence often exist, but convergence rates are not always specified. For example, the rates of ADMM and PDMM have been studied [14, 17–20], in the case of PDMM even on systems including quantization [21]. Unfortunately, there is still much unknown about quantization effects in distributed signal processing. As it is such a vast area of research, the work in [21] focussed on the averaging problem and cannot easily be generalized to different classes of problems. Therefore, further research on this topic was called for.

1.2. Research question and document outline

In this thesis, quantization effects in distributed algorithms will be further explored and the convergence rates will be analysed for two classes of problems. The central research question in this thesis is:

What are the effects of fixed rate quantization on the convergence of distributed optimization schemes in synchronous operation? Is the convergence rate of such an optimization scheme robust against the errors introduced by fixed rate quantization?

The structure of this thesis is as follows: Chapter 2 contains the nomenclature and the principles behind two optimization problems: the averaging problem and the Gaussian channel capacity problem. Chapter 3 is a preliminary chapter, which informs about the details of solving a decentralized problem using the Primal-Dual Method of Multipliers, the mathematical theory of monotone operators and the notion of operator splitting. In Chapter 4, the afore mentioned Primal-Dual Method of Multipliers and monotone operator theory are combined to yield the Monotone Primal-Dual Method of Multipliers. Up to Chapter 4, all chapters have assumed ideal transmissions, but this is not possible in practice. After the theory behind quantization is explained in preliminary Chapter 5, it is combined with the results from previous chapters in Chapter 6. The theoretical results are verified by simulations, which are discussed in Chapter 7. A final conclusion, along with a discussion and recommendations for future work are presented in Chapter 8.

2 Background

2.1. Nomenclature

In the course of this work, the following notation is used:

- Scalars will be indicated by a lower case letter (a).
- Vectors are displayed as a lower case, bold faced letter ($\mathbf{v}$). Elements are selected using indices (scalar and vector elements respectively v_i and $\mathbf{v}_i$).
- Matrices are indicated by an upper case letter (A). Elements are selected via indices (scalar, vector and submatrix elements respectively a_{ij}, $\mathbf{A}_i$ and A_{ij}).
- S is used to indicate a generic mathematical operator.
- The transpose of a matrix or vector is indicated by a superscript T (A^T or $\mathbf{v}^T$).
- The inverse, either of a matrix or of an operator, is indicated by a superscript -1 (A^{-1} or S^{-1}).
- The pseudo inverse (Moore-Penrose Matrix Inverse [22]) of a matrix is indicated by a superscript † ($A^{\dagger}$).
- In iterative processes, the iteration indicator (k) will be displayed brackets in subscript ($z_{(k)}$).
- The fixed point of an operator and the end point of an algorithm are denoted by a subscript * (z_*).
- Unless otherwise specified, $\|\bullet\|$ is used as the 2-norm ($\|\bullet\|_2$).
- The base-2 logarithm is implied when using log(), any other bases will be indicated by subscript.
- A graph consisting of the nodes V and edges E is noted as $G = (V, E)$.
- The number of nodes in a certain graph is denoted as N.
- The number of undirected edges in a certain graph is referred to as M.
- The neighbours of a node i (all other nodes directly connected to node i via an edge) are noted by $\mathcal{N}_i$.

2.2. Different optimization problems

As discussed in the introduction, the first step of solving an optimization problem in a distributed manner is the problem formulation. Suppose we start with a standard convex optimization problem with equality constraints[1]:

$$\min_{\mathbf{x}} f(\mathbf{x}) \quad \text{s.t.} \quad A\mathbf{x} = \mathbf{b} \tag{2.1}$$

[1]The equality constraints have to be affine for the problem to be a convex optimization problem.

An important observation is the notion of *consensus*: the distributed problems that are considered are formulated in such a way that all nodes try to achieve consensus about a global variable. For this purpose, each node contains a local estimate of this global variable and to ensure consensus, the constraints for Eq. (2.1) have to be consensus constraints: $x_i - x_j = 0$ if i and j are neighbours and x_i and x_j are the estimates at nodes i and j. The assumption is made that $f(\mathbf{x})$ is convex, but there are different possible subtypes of functions, all with different properties. In this work, two different functions will be studied: the averaging problem and the Gaussian channel capacity problem.

2.2.1. Averaging problem

For the averaging problem, the objective is to compute the average of a certain node-based parameter. Examples are the average light-intensity in an image or the average temperature in a room, the latter of which will be used in the remainder of this work. Allow T_i to be the temperature at node i and $x_{i,(k)}$ to be the estimate of the average at node i, iteration k. The objective is to make all x_i converge to the same x_* and to let x_* be the average of the vector $\mathbf{T}$. In other words: $x_* = \frac{1}{N}\sum_{i=1}^{N} T_i$. A function $f(\mathbf{x})$ that yields this x_* can be found. For the averaging case, Eq. (2.1) becomes:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \sum_{i=1}^{N} \frac{1}{2}(x_i - T_i)^2 \\ \text{s.t.} \quad & x_i - x_j = 0, \quad \forall (i,j) \in E \end{aligned} \tag{2.2}$$

Clearly, this is a quadratic problem, which has certain properties in relation to convergence. In order to test the effects on non-quadratic problems, a different starting problem has to be defined.

2.2.2. Gaussian channel capacity problem

Apart from the averaging problem, the *Gaussian channel capacity* (GCC) problem is another example of a distributable optimization scheme. This problem is often found in multi-channel information theory analysis. Suppose we have n channels, all with their own noise variance σ_i^2, and a message is to be send using these channels. The GCC problem consists of finding the optimal distribution of the total transmission power P_{tot} among the channels. The optimal way to use the channels is the so-called water filling solution, which can be computed algorithmically. For simplicity, it is assumed that the channels are ordered in such a way that $\sigma_i^2 \leq \sigma_j^2$ if $i < j$.

Start off by using the channel with the lowest noise variance, and add power to this channel. Keep adding power (P_i) until either you reach the total amount of power P_{tot}, or $\sigma_i^2 + P_i = \sigma_{i+1}^2$. If the latter is the case, include the next channel and add power to both channel i and $i+1$. Again, continue adding power until all power is used ($P_i + P_{i+1} = P_{tot}$) or $\sigma_{i+1}^2 + P_{i+1} = \sigma_{i+2}^2$. Continue in this way until all power is distributed. This process is summarized in Algorithm 1

Algorithm 1: Water filling algorithm

```
Data: $\boldsymbol{\sigma^2}, n, P_{tot}$
Result: $\mathbf{x}$
Sort $\boldsymbol{\sigma^2}$;
$P_{available} = P_{tot}$;
$k = 1$;
while $P_{Available} > 0$ do
    if $k<n$ then
        $P_{Next} = \min\left(\frac{P_{Available}}{k}, \sigma^2_{k+1} - \sigma^2_k\right)$;
        $x_{1,..,k} = x_{1,..,k} + P_{Next}$;
        $P_{Available} = P_{Available} - k * P_{Next}$;
        $k = k + 1$;
    else
        $P_{Next} = \frac{P_{Available}}{k}$;
        $x_{1,..,k} = x_{1,..,k} + P_{Next}$;
        $P_{Available} = 0$;
    end
end
```

In Fig. 2.1, an example is illustrated, with four different channels and total amount of power P_{tot} equal to 1. For simplicity, units have been omitted as they are not relevant for this example. The noise variances are 1, 1.5, 1.75 and 2. Therefore, 0.5 power is applied to the first channel. Then, with 0.5 power left, we include the second channel and distribute the 0.5 power left between those two channels. As a result, channel 1 gets 0.75 power and channel 2 gets 0.25 power. Channel 3 and 4 both get 0 power, as they have too much noise. If the total amount of power would be higher than 1 (and less or equal than 1.75), the remaining power would be distributed between channels 1, 2 and 3. Clearly, the distribution of the power behaves like water, filling from low to high, where the total amount of water is equal to the total amount of power. This behaviour was the inspiration behind the name “water filling”.

Apart from this intuitive and iterative approach, the problem can also be solved by an optimization approach. The GCC problem can be cast as an optimization problem of the following form [23]:

$$\begin{aligned} \min_{\mathbf{x}} \quad & -\sum_{i\in V} \log(\sigma_i^2 + x_i) \\ \text{s.t.} \quad & \mathbf{x} \geq \mathbf{0} \\ & \mathbf{1}^T\mathbf{x} = P_{tot} \end{aligned} \tag{2.3}$$

This is a non-linear problem in the form of Eq. (2.1) with additional inequality constraints. In Section 3.2, this problem is cast into the Primal-Dual Method of Multipliers framework, including these extra inequality constraints. The next step after defining the problems is translating them to a decentralized problem, which is discussed in the next chapter.

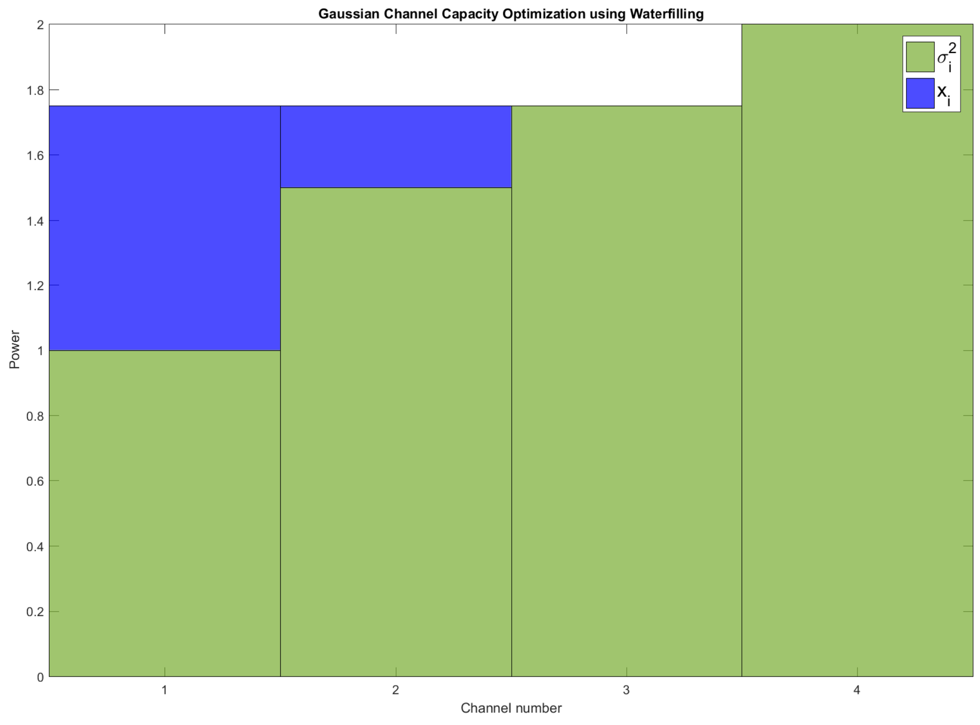


Figure 2.1: Water filling example with four channels

3
Preliminary: Mathematical basis

For both the quadratic problem (averaging problem) and the non-quadratic problem (GCC), there are many algorithms that solve the problem. Based on previous work, the *Primal-Dual Method of Multipliers* (PDMM) was chosen as solving algorithm.

3.1. Solving the decentralized averaging problem using the Primal-Dual Method of Multipliers

Assuming the standard form in Eq. (2.1), the next step is to separate the problem into a sum of node-based functions:

$$\min_{\mathbf{x}} \sum_{i\in V} f_i(x_i) \quad \text{s.t. } \ a_{ij}x_i + a_{ji}x_j = b_{ij}, \ \ \forall (i,j) \in E \tag{3.1}$$

This is possible for every problem, even if multiple f_i depend on the same variable $\mathbf{x}$ by introducing two different variables x_i and x_j including $x_i = x_j$ in the equality constraints. In Section 3.2, the proceedings with inequality constraints are clarified. For the averaging problem, the separation is implicit in the problem generation as it is already expressed as a sum of node-based functions. The next step is to transcribe Eq. (3.1) into an unconstrained problem, using the *primal Lagrangian*:

$$\mathscr{L}_p(\mathbf{x}, \boldsymbol{\nu}) = \sum_{i\in V} \left(f_i(x_i) + \sum_{j\in\mathcal{N}_i} \nu_{ij}(a_{ij}x_i + a_{ji}x_j - b_{ij}) \right)$$

The primal Lagrangian consists of a function of the original variable and a new *dual variable*. This dual variable is multiplied with the original equality constraint, so that the effect of the dual variable is non-existent if and only if the original constraints are fulfilled. As the Lagrangian is a function of two variables, optimization will have to occur in both variables to obtain the overall optimum. When the supremum of the Lagrangian is taken with respect to the dual variable, two options occur:

- the original constraints on the primal variable $\mathbf{x}$ are fulfilled, in which case the dual variable is multiplied by zero and is thus arbitrary,
- the original constraints are not fulfilled, in which case the supremum over the dual variable is infinity.

By then taking the infimum over the original variable $\mathbf{x}$, the optimum is found: find the $\mathbf{x}$ that minimizes the cost but is also contained in the feasible set. In fact, optimizing this Lagrangian yields the same optimum as the original constrained problem[1]. As previously stated, this optimum will be found by optimizing the Lagrangian in the following way:

$$\inf_{\mathbf{x}} \sup_{\boldsymbol{\nu}} \ \mathscr{L}_p(\mathbf{x}, \boldsymbol{\nu}) = \sup_{\boldsymbol{\nu}} \inf_{\mathbf{x}} \ \mathscr{L}_p(\mathbf{x}, \boldsymbol{\nu}) = \sup_{\boldsymbol{\nu}} \ g(\boldsymbol{\nu})$$

[1]For the optima to be equivalent, Slater's condition must hold: the problem has to be strictly feasible. In the case of equality constraint optimization, this is always the case.

$g(\boldsymbol{\nu})$ is called the *dual problem* and is defined as the infimum over $\mathbf{x}$ of the Lagrangian, which yields Eq. (3.2).

$$\begin{aligned} g(\boldsymbol{\nu}) &= \inf_{\mathbf{x}} \ \mathscr{L}_p(\mathbf{x},\boldsymbol{\nu}) \\ &= -\sup_{\mathbf{x}} \ -\mathscr{L}_p(\mathbf{x},\boldsymbol{\nu}) \\ &= -\sup_{\mathbf{x}} \ \sum_{i\in V}\left(-f_i(x_i) - \sum_{j\in\mathcal{N}_i} \nu_{ij}(a_{ij}x_i + a_{ji}x_j - b_{ij})\right) \end{aligned} \tag{3.2}$$

For the next step, the convex conjugate of the function f_i is used.

Definition 3.1. The convex conjugate of a function, $f^*(\mathbf{u})$, is defined as [24]:

$$f^*(\mathbf{u}) = \sup_{\mathbf{x}}\left(\mathbf{u}^T\mathbf{x} - f(\mathbf{x})\right)$$

Utilizing Definition 3.1, Eq. (3.2) can be transformed to Eq. (3.3):

$$g(\boldsymbol{\nu}) = \sum_{i\in V} -f_i^*\left(\sum_{j\in\mathcal{N}_i} a_{ij}\nu_{ij}\right) + \sum_{(i,j)\in E} \nu_{ij}b_{ij} \tag{3.3}$$

The problem is that this function is not separable, as ν_{ij} is a variable that corresponds to a certain edge (i,j) and is thus used by both nodes i and j. Therefore, it is necessary to define a new node-based variable $\lambda_{i|j}$, in such a way that $\lambda_{i|j} = \lambda_{j|i} = \nu_{ij}$. Applying this notion to Eq. (3.3) and optimizing this dual function results in:

$$\max_{\boldsymbol{\nu},\boldsymbol{\lambda}} \ \sum_{i\in V} -f_i^*\left(\sum_{j\in\mathcal{N}_i} a_{ij}\lambda_{i|j}\right) + \sum_{(i,j)\in E} \nu_{ij}b_{ij} \quad \text{s.t. } \lambda_{i|j} = \lambda_{j|i} = \nu_{ij}$$

To turn this new optimization problem into an unconstrained one, the Lagrangian is used again, this time yielding the *dual Lagrangian.*

$$\mathscr{L}_d(\boldsymbol{\nu},\boldsymbol{\lambda},\boldsymbol{\zeta}) = \sum_{i\in V}\left(-f_i^*\left(\sum_{j\in\mathcal{N}_i} a_{ij}\lambda_{i|j}\right) + \sum_{j\in\mathcal{N}_i} \zeta_{i|j}(\nu_{ij} - \lambda_{i|j})\right) + \sum_{(i,j)\in E} \nu_{ij}b_{ij} \tag{3.4}$$

$\zeta_{i|j}$ are the dual Lagrange multipliers, for which Fenchel's inequality must hold with equality at the saddle point [17]:

$$\zeta^*_{i|j} = a_{ji}x_j - b_{ij}$$

As a result, we constrict the form of the dual Lagrange multipliers $\zeta_{i|j}$ to be $\zeta_{i|j} = a_{ji}x_j - b_{ij}$. Substituting this equality in Eq. (3.4) yields:

$$\mathscr{L}_d(\boldsymbol{\nu},\boldsymbol{\lambda},\mathbf{x}) = \sum_{i\in V}\left(-f_i^*\left(\sum_{j\in\mathcal{N}_i} a_{ij}\lambda_{i|j}\right) - \sum_{j\in\mathcal{N}_i} \lambda_{j|i}(a_{ij}x_i - b_{ij})\right) - \sum_{(i,j)\in E} \nu_{ij}(b_{ij} - a_{ij}x_i - a_{ji}x_j)$$

This dual Lagrangian will be combined with the primal Lagrangian to result in the *primal-dual Lagrangian* in:.

$$\mathscr{L}_{pd}(\mathbf{x},\boldsymbol{\lambda}) = \mathscr{L}_p(\mathbf{x},\boldsymbol{\nu}) + \mathscr{L}_d(\boldsymbol{\nu},\boldsymbol{\lambda},\mathbf{x}) = \sum_{i\in V}\left(f_i(x_i) - f_i^*\left(\sum_{j\in\mathcal{N}_i} A_{ij}\lambda_{i|j}\right) - \sum_{j\in\mathcal{N}_i} \lambda_{j|i}(a_{ij}x_i - b_{ij})\right)$$

To ensure primal and dual feasibility, two penalty terms are included, $\mathscr{H}_p$ and $\mathscr{H}_d$.

$$\begin{aligned} \mathscr{H}_p(\mathbf{x}) &= \sum_{(i,j)\in E} \frac{\rho_p}{2}\|a_{ij}x_i + a_{ji}x_j - b_{ij}\|^2 \\ \mathscr{H}_d(\boldsymbol{\lambda}) &= \sum_{(i,j)\in E} \frac{\rho_d}{2}\|\lambda_{i|j} - \lambda_{j|i}\|^2 \end{aligned}$$

The final problem becomes $\mathscr{L}_{pdmm}(\mathbf{x},\boldsymbol{\lambda}) = \mathscr{L}_{pd}(\mathbf{x},\boldsymbol{\lambda}) + \mathscr{H}_p(\mathbf{x}) - \mathscr{H}_d(\boldsymbol{\lambda})$. This can be solved iteratively, by the following updates [17]:

$$x_{i,(k+1)} = \arg\min_{x_i}\left(f_i(x_i) - x_i\left(\sum_{j\in\mathscr{N}_i} a_{ij}\lambda_{j|i,(k)}\right) + \sum_{j\in\mathscr{N}_i}\frac{\rho_p}{2}\|a_{ij}x_i + a_{ji}x_{j,(k)} - b_{ij}\|^2\right) \tag{3.5a}$$

$$\lambda_{i,(k+1)} = \arg\min_{\lambda_i}\left(f_i^*\left(\sum_{j\in\mathscr{N}_i} a_{ij}\lambda_{i|j}\right) + \sum_{j\in\mathscr{N}_i}\lambda_{i|j}(a_{ji}x_{j,(k)} - b_{ij}) + \sum_{j\in\mathscr{N}_i}\frac{\rho_d}{2}\|\lambda_{i|j} - \lambda_{j|i,(k)}\|^2\right) \tag{3.5b}$$

After studying the latter update equation, it can be seen that the expanded form is equal to that of Eq. (3.6) if $\rho_d = \rho_p^{-1}$.

$$\lambda_{i|j,(k+1)} = \lambda_{j|i,(k)} + \rho_p(a_{ij}x_{i,(k+1)} + a_{ji}x_{j,(k)} - b_{ij}) \tag{3.6}$$

3.2. Solving the decentralized Gaussian channel capacity problem using the Primal-Dual Method of Multipliers

Recall the GCC optimization problem in Eq. (2.3), where $P_{tot} = 1$. This problem cannot be cast into the PDMM framework in the exact same way as the averaging problem, as there are both global inequality and equality constraints, and no consensus constraints. The first step will therefore be to find the equivalent consensus problem. For this purpose, the Lagrangian of this problem is constructed:

$$\begin{aligned}\mathscr{L}(\mathbf{x},\boldsymbol{\lambda},\mu) &= \sum_{i\in V}\left(-\log(\sigma_i^2 + x_i)\right) - \mu(\mathbf{1}^T\mathbf{x} - 1) - \boldsymbol{\lambda}^T\mathbf{x}\\ \text{s.t. } &\boldsymbol{\lambda} \geq \mathbf{0}\end{aligned} \tag{3.7}$$

Even with the inequality constraints in Eq. (2.3), Slater's condition still holds, so there is strong duality and the solution to the Lagrangian is the same as the solution to the original problem. The *Karush–Kuhn–Tucker conditions* (KKT conditions) of the Lagrangian in Eq. (3.7) are as follows [23]:

$$\begin{aligned}\mathbf{x}_* &\geq \mathbf{0}\\ \mathbf{1}^T\mathbf{x}_* &= 1\\ \boldsymbol{\lambda}_* &\geq \mathbf{0}\\ \lambda_{i,*}x_{i,*} &= 0 \ \ \forall i\in V\\ \frac{-1}{\sigma_i^2 + x_{i,*}} - \mu_* - \lambda_{i,*} &= 0 \ \ \forall i\in V\end{aligned}$$

Solving the last condition for $x_{i,*}$ leads to

$$x_{i,*} = \frac{-1}{\mu_* + \lambda_{i,*}} - \sigma_i^2 \tag{3.8}$$

Substituting Eq. (3.8) into Eq. (3.7) yields the dual function

$$g(\boldsymbol{\lambda},\mu) = \sum_{i\in V}\left(-\log(\frac{-1}{\mu+\lambda_i}) + 1 + \mu\sigma_i^2 + \lambda_i\sigma_i^2\right) + \mu$$

From this dual function, an implicit constraint on $\mu+\lambda_i$ arises: $\mu+\lambda_i < 0 \ \ \forall i\in V$. Although λ_i is a local node variable, μ is still global. To make this problem distributed, a new variable μ_i is defined at each node, with the constraint that $\mu_i = \mu_j = \mu \ \ \forall(i,j)\in E$. After substituting this into the dual function, the dual problem becomes:

$$\begin{aligned}\min_{\boldsymbol{\lambda},\boldsymbol{\mu}} \ &\sum_{i\in V}\left(\log(\frac{-1}{\mu_i+\lambda_i}) - \mu_i\sigma_i^2 - \lambda_i\sigma_i^2 - \frac{\mu_i}{N}\right)\\ \text{s.t. } &\mu_i = \mu_j, \ \ \forall(i,j)\in E\\ &0 > \mu_i + \lambda_i \forall i\in V\\ &0 \leq \lambda_i \ \ \forall i\in V\end{aligned} \tag{3.9}$$

This is again a separable optimization problem, this time including consensus constraints. It is important to realise that only the first constraint is a consensus constraint among different nodes, the latter two are local constraints at each node. From this point, the same derivation as in Section 3.1 can be followed. This leads to the form as in Eqs. (3.5a) and (3.6) where $x_i = \mu_i$ but the update on μ_i is now a constrained optimization problem. In Section 4.2, we will proceed with this problem, but first a background on monotone operator theory is presented.

3.3. Monotone operator theory

Thus far, only mathematical functions have been used in the analysis. From this point on, mathematical operators will be introduced as these operators will be used throughout this thesis. For this section, most definitions and fundamentals are inspired by [25]. These operators are defined as

$$S(\mathbf{x}) = \{\mathbf{y} \in \mathbb{R}^n | \exists \mathbf{x} \in \mathbb{R}^n : (\mathbf{x}, \mathbf{y}) \in S\}$$

It is important to realize that applying an operator results in a set, so the formulation $\mathbf{y} \in S(\mathbf{x})$ is used. However, in case $S(\mathbf{x})$ is a singleton for any $\mathbf{x}$, then S is a function for that $\mathbf{x}$ and the common notation $\mathbf{y} = S(\mathbf{x})$ is used, although strictly $\{\mathbf{y}\} = S(\mathbf{x})$ would be more correct. Contrary to inverse functions, inverse relation always exists (and as functions are a special type of relations, all functions will thus have an inverse relation although they may not have an inverse function). In general, the inverse relation is defined as

$$S^{-1}(\mathbf{x}) = \{(\mathbf{x}, \mathbf{y}) | (\mathbf{y}, \mathbf{x}) \in S\}$$

Although not all properties of functions hold for relations, some do. They can for example be subject to multiplication, so that

$$cS(\mathbf{x}) = \{(\mathbf{x}, c\mathbf{y}) | (\mathbf{x}, \mathbf{y}) \in S\}, c \in \mathbb{R}$$

Operators can also act upon operators, for which the notation $S_2 \circ S_1(\mathbf{x})$ is used. This means the operator S_1 first acts upon $\mathbf{x}$, the result is then the input of the operator S_2.

A fixed point of an operator is defined as any point where $\mathbf{x}_* \in S(\mathbf{x}_*)$. This means it is possible to stay in a certain point if the operator is applied again, and if the operator yields a singleton it is even an equilibrium point of the function. There are more definitions that are specific to operator theory:

Definition 3.2. The *resolvent* of an operator S is defined as

$$J_{cS}(\mathbf{x}) = (I + cS)^{-1}(\mathbf{x})$$

Definition 3.3. The *Cayley operator* of S is defined as

$$C_{cS}(\mathbf{x}) = (2J_{cS} - I)(\mathbf{x})$$

Definition 3.4. The *proximal point* is defined as the resolvent of the subdifferential of a function [25]:

$$J_{c\partial f}(\mathbf{x}) = \text{prox}_{cf}(\mathbf{x}) = \arg\min_{\mathbf{u}} \left(f(\mathbf{u}) + \frac{1}{2c} \|\mathbf{u} - \mathbf{x}\|_2^2 \right)$$

Similar to functions, relations can also be classified, e.g. as being monotone or Lipschitz continuous. A relation is called *monotone* if

$$(S(\mathbf{y}) - S(\mathbf{x}))^T (\mathbf{y} - \mathbf{x}) \geq 0 \ \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$$

In [26], an alternate condition for monotonicity is presented: an operator is monotone if and only if $\text{ran}(I+S) = \mathbb{R}^n$. Operators for which the following inequality holds are referred to as *strongly monotone* with parameter m.

$$(S(\mathbf{y}) - S(\mathbf{x}))^T (\mathbf{y} - \mathbf{x}) \geq m \|\mathbf{y} - \mathbf{x}\|_2^2$$

A monotone operator is *maximal* if it is not contained properly by any other monotone operator. A related and very powerful lemma is the following:

Lemma 3.1. If S is maximal monotone on $\mathbb{R}^n$ and $c > 0$:

$$0 \in S(\mathbf{x}) \Leftrightarrow J_{cS}(\mathbf{x}) = \mathbf{x}$$

Proof. Allow $\mathbf{x}$ to be a solution to the monotonic inclusion problem. Add $\mathbf{x}$ on both sides and apply basic operator theory:

$$0 \in S(\mathbf{x}) \Leftrightarrow \mathbf{x} \in (I + cS)(\mathbf{x}) \Leftrightarrow (I + cS)^{-1}(\mathbf{x}) = \mathbf{x} \Leftrightarrow \mathbf{x} = J_{cS}(\mathbf{x})$$

where equality holds as the resolvent of a monotone operator is single valued [25]. □

It is possible to make a simple derivation from this lemma:

Corollary 3.1. If S maximal monotone on $\mathbb{R}^n$ and $c > 0$:

$$0 \in S(\mathbf{x}) \Leftrightarrow C_{cS}(\mathbf{x}) = \mathbf{x}$$

Proof. Apply Definition 3.3 in combination with Lemma 3.1:

$$C_{cS}(\mathbf{x}) = 2J_{cS}(\mathbf{x}) - \mathbf{x} = \mathbf{x}$$

□

A last important notion is *Lipschitz continuity*. This yields an upper bound on the rate of change, as

$$\|f(\mathbf{y}) - f(\mathbf{x})\| \leq L\|\mathbf{y} - \mathbf{x}\|$$

Non-expansive relations are operators that are Lipschitz continuous with parameter $L \leq 1$. *Contractive* operators have a Lipschitz parameter $L < 1$.

After this background in operator theory, it is time to put this knowledge to use. The PDMM problem will be translated into a *monotonic inclusion problem* using this monotone operator theory. Operator splitting will then be applied to generate two sub-problems, which will be studied subsequently. Therefore, the notion of operator splitting must first be introduced.

3.4. Operator Splitting

For solving a maximal monotonic inclusion problem such as $0 \in S(\mathbf{x})$, Lemma 3.1 shows that this is equivalent to finding the fixed point of the resolvent, or even that of the Cayley operator by Corollary 3.1. However, the inversion step in the resolvent and Cayley operator can be challenging. To overcome this, the so-called operator splitting method can be used. This method decomposes the original operator S into two maximal monotone operators. Splitting results in $S = S_1 + S_2$ and is only beneficial if the resolvents of the two sub-operators are easily computed. An important notion following from this splitting is:

$$0 \in S_1(\mathbf{x}) + S_2(\mathbf{x}) \Leftrightarrow \mathbf{y} \in S_1(\mathbf{x}) \text{ and } -\mathbf{y} \in S_2(\mathbf{x}) \tag{3.10}$$

Many splitting algorithms exist, but two will be emphasised in this work: Peaceman-Rachford splitting and Douglas-Rachford splitting. Although splitting into more than two sub-operators is possible, those splitting algorithms are not covered in this thesis.

One of the methods of operator splitting is the following: As S_1 is maximal monotone, $\mathbf{x} = J_{cS_1}(\mathbf{x}+c\mathbf{y}),\ \ \mathbf{y} \in S_1(\mathbf{x})$ [27, Corollary 2.3]. Therefore:

$$\begin{aligned}\mathbf{x} - c\mathbf{y} &= J_{cS_1}(\mathbf{x} + c\mathbf{y}) + \mathbf{x} - (\mathbf{x} + c\mathbf{y})\\ &= (2J_{cS_1} - I)(\mathbf{x} + c\mathbf{y})\\ &= C_{cS_1}(\mathbf{x} + c\mathbf{y})\end{aligned}$$

Recalling Eq. (3.10), this implies $\mathbf{x} = J_{cS_2}(\mathbf{x} - c\mathbf{y}),\ \ -\mathbf{y} \in S_2(\mathbf{x})$ and therefore

$$\begin{aligned}\mathbf{x} + c\mathbf{y} &= J_{cS_2}(\mathbf{x} - c\mathbf{y}) + \mathbf{x} - (\mathbf{x} - c\mathbf{y})\\ &= (2J_{cS_2} - I)(\mathbf{x} - c\mathbf{y})\\ &= C_{cS_2}(\mathbf{x} - c\mathbf{y})\end{aligned}$$

Combining the two yields

$$\mathbf{x} + c\mathbf{y} = C_{cS_2} \circ C_{cS_1}(\mathbf{x} + c\mathbf{y}), \quad \mathbf{x} = J_{cS_1}(\mathbf{x} + c\mathbf{y})$$

By defining $\mathbf{z} = \mathbf{x} + c\mathbf{y}$, the Peaceman-Rachford splitting method [28, 29] is found.

Definition 3.5. The Peaceman-Rachford splitting method is defined as

$$\begin{aligned} 0 \in S_1(\mathbf{x}) + S_2(\mathbf{x}) \Leftrightarrow \mathbf{z}_{(k+1)} &= C_{cS_2} \circ C_{cS_1}(\mathbf{z}_{(k)}), \\ \mathbf{x}_{(k+1)} &= J_{cS_1}(\mathbf{z}_{(k+1)}) \end{aligned}$$

From this, a slight adjustment can be made to create an $\frac{1}{2}$-averaged iteration, known as Douglas-Rachford splitting [29, 30].

Definition 3.6. The Douglas-Rachford splitting method is defined as

$$\begin{aligned} \mathbf{z}_{(k+1)} &= \frac{1}{2}(I + C_{cS_2} \circ C_{cS_1})(\mathbf{z}_{(k)}), \\ \mathbf{x}_{(k+1)} &= J_{cS_1}(\mathbf{z}_{(k+1)}) \end{aligned} \tag{3.11}$$

4 Monotone Primal-Dual Method of Multipliers

After this background on operator splitting, the monotone interpretation of the Primal-Dual Method of Multipliers algorithm will be discussed. For the sake of clarity, this approach will be referred to as MPDMM.

4.1. Monotone Primal-Dual Method of Multipliers for the averaging problem

The first step to solving the averaging problem using MPDMM is defining the dual problem using the convex conjugate as defined in Definition 3.1. Recall the standard form of the separable optimization problem:

$$\min \sum_{i \in V} f_i(\mathbf{x}_i) \quad \text{s.t. } A_{ij}\mathbf{x}_i + A_{ji}\mathbf{x}_j = \mathbf{b}_{ij} \tag{4.1}$$

Recall the averaging problem in Eq. (2.2):

$$\begin{aligned} \min_{\mathbf{x}} \ & \sum_{i=1}^{N} \frac{1}{2}(x_i - T_i)^2 \\ \text{s.t. } & x_i - x_j = 0, \ \ \forall (i,j) \in E \end{aligned}$$

This problem is indeed of the same form as Eq. (4.1), where $f_i(\mathbf{x}_i) = \frac{1}{2}(x_i - T_i)^2$, A_{ij} are scalars such that $a_{ij} = -a_{ji} \ \ \forall (i,j) \in E$ and $b_{ij} = 0 \ \ \forall (i,j) \in E$. For simplicity, the following convention is used: $a_{ij} = 1$ and $a_{ji} = -1$ if $(i,j) \in E, \ \ i < j$. In the interest of generality, these simplifications are not included in the derivation of the MPDMM algorithm.

Applying Definition 3.1 and writing the dual problem in the more compact matrix form results in:

$$\min f^*(-A^T\boldsymbol{\nu}) + \mathbf{b}^T\boldsymbol{\nu} \tag{4.2}$$

Here, A is an RxN matrix for a graph with N nodes and R undirected edges, while $\boldsymbol{\nu}$ is an R-dimensional vector and $\mathbf{b}$ is also R-dimensional. The optimality condition is easily deduced by taking the derivative with respect to the dual variable $\boldsymbol{\nu}$: $0 \in \partial_{\boldsymbol{\nu}} f^*(-A\boldsymbol{\nu}) + \mathbf{b}$. In the non-matrix form, this corresponds to

$$0 \in \partial_{\boldsymbol{\nu}_{ij}} f_i^*(-\mathbf{a}_i^T\boldsymbol{\nu}) + \partial_{\boldsymbol{\nu}_{ij}} f_j^*(-\mathbf{a}_j^T\boldsymbol{\nu}) + \mathbf{b}_{ij}$$

Clearly, this problem is non-separable as both f_i and f_j depend on $\boldsymbol{\nu}_{ij}$. Following the same method as in the previous chapter, two new node-based variables are created, $\boldsymbol{\lambda}_{i|j}$ and $\boldsymbol{\lambda}_{j|i}$, in such a way that at convergence $\boldsymbol{\lambda}_{i|j} = \boldsymbol{\lambda}_{j|i} = \boldsymbol{\nu}_{ij}$. Stacking all $\boldsymbol{\lambda}_{i|j}$ yields the $\boldsymbol{\lambda}$ vector, which therefore has twice the dimension of $\boldsymbol{\nu}$: $\boldsymbol{\lambda} \in \mathbb{R}^{2R}$. Note that $2R = M$ as the number of directed edges is twice the number of undirected edges. A new and properly dimensioned matrix C is defined using the A matrix. To summarize, the following is known for a graph with N nodes and R undirected edges/M directed edges:

- $\boldsymbol{\lambda}(l)$ and $\boldsymbol{\lambda}(l+R)$ both correspond to the $\boldsymbol{\nu}_{ij}$ of the l^{th} edge, $l < R$.

- C consists of two $R \times N$ blocks, where the top block only consists of the part of A that contains $A_{ij},\ \ i<j$ and the lower block consists of the opposite: $A_{ji},\ \ i<j$. Hence, $C \in \mathbb{R}^{MxN}$.

Thus, $\boldsymbol{\lambda}$ is a repetitive vector, where the first half and second half are equal. In the averaging case, $a_{ij} = -a_{ji}$ with the convention that $a_{ij} = 1 \Leftrightarrow i<j,\ \ (i,j) \in E$. Therefore, the top part of C consists of all positive values in the A matrix (A_+), while the bottom part consists of the negative values (A_-): $C = \begin{bmatrix} A_+ \\ A_- \end{bmatrix}$. As $\boldsymbol{\lambda}$ and C are now properly dimensioned, only $\mathbf{b}$ does not have a compatible dimension. For this purpose, a new vector $\mathbf{d}$ is defined: $\mathbf{d} = \frac{1}{2}\begin{bmatrix} \mathbf{b} \\ \mathbf{b} \end{bmatrix}$ so that $\mathbf{d} \in \mathbb{R}^M$. This results in a different expression for Eq. (4.2):

$$\min f^*(-C^T\boldsymbol{\lambda}) + \mathbf{d}^T\boldsymbol{\lambda} + I_\Lambda(\boldsymbol{\lambda}) \tag{4.3}$$

where P is an MxM permutation matrix swapping the first and last R rows and $I_\Lambda(\boldsymbol{\lambda})$ is the indicator function for the set $\Lambda = \{\boldsymbol{\lambda} \in \mathbb{R}^M | \boldsymbol{\lambda} = P\boldsymbol{\lambda}\}$, as defined below.

$$I_\Lambda(\boldsymbol{\lambda}) = \begin{cases} 0, & \boldsymbol{\lambda} \in \Lambda \\ \infty, & \boldsymbol{\lambda} \notin \Lambda \end{cases}$$

Following the definition of C, it is clear that $C + PC = \begin{bmatrix} A \\ A \end{bmatrix}$. This leads to the following important notion:

$$(C+PC)\mathbf{x} = 2\mathbf{d} \Leftrightarrow A\mathbf{x} = \mathbf{b}$$

This proves the equivalence of Eqs. (4.2) and (4.3). Taking the derivative of Eq. (4.3) with respect to $\boldsymbol{\lambda}$ yields Eq. (4.4).

$$\mathbf{0} \in -C\partial f^*(-C^T\boldsymbol{\lambda}) + \mathbf{d} + N_\Lambda(\boldsymbol{\lambda}) \tag{4.4}$$

Here, the normal cone operator $N_\Lambda(\boldsymbol{\lambda})$ is used, which is the derivative of the indicator function and defined as

$$N_\Lambda(\boldsymbol{\lambda}) = \begin{cases} \emptyset & \boldsymbol{\lambda} \notin \Lambda \\ \{\mathbf{g} | \mathbf{g}^T(\mathbf{y} - \boldsymbol{\lambda}) \leq 0, \forall \mathbf{y} \in \Lambda\} & \boldsymbol{\lambda} \in \Lambda \end{cases}$$

Eq. (4.4) is a so-called *monotonic inclusion problem*, as all operators are monotone[1]. Peaceman-Rachford splitting will now be applied, where $S_1 = -C\partial f^*(-C^T\boldsymbol{\lambda}) + \mathbf{d}$ and $S_2 = N_\Lambda(\boldsymbol{\lambda})$. To start further analysis of this splitting method, the resolvents will be studied initially. An important insight about the resolvent is the following: $J_{cS_2}(\mathbf{y}) = J_{N_\Lambda}(\mathbf{y}) = \text{prox}_{cI_\Lambda}(\mathbf{y})$ and therefore Definition 3.4 can be applied. This leads to the derivation in Eq. (4.5), where the final result reads the projection of $\mathbf{y}$ onto Λ.

$$\begin{aligned} J_{cS_2} &= \arg\min_u \left(I_\Lambda(\mathbf{u}) + \frac{1}{2c}\|\mathbf{u} - \mathbf{y}\|_2^2 \right) \\ &= \arg\min_{u \in \Lambda} \|\mathbf{u} - \mathbf{y}\|_2^2 \\ &= \Pi_\Lambda(\mathbf{y}) \end{aligned} \tag{4.5}$$

The solution to the constrained optimization problem in the second line of Eq. (4.5) can be computed explicitly. Let us call this solution $\boldsymbol{u}^+$. The Lagrangian of the constrained optimization problem reads:

$$L(\mathbf{y}, \mathbf{u}, \boldsymbol{\kappa}) = \|\boldsymbol{u} - \mathbf{y}\|_2^2 + \boldsymbol{\kappa}^T(P\mathbf{u} - \mathbf{u})$$

Since $\mathbf{y}$ is known, the optimality condition can be analysed by computing the gradient with respect to the other variables:

$$\begin{aligned} \nabla_{\mathbf{u}} = 0 &\rightarrow 2(\boldsymbol{u}^+ - \mathbf{y}) + P^T\boldsymbol{\kappa} - \boldsymbol{\kappa}^T = 0 \\ \nabla_{\boldsymbol{\kappa}} = 0 &\rightarrow (P\boldsymbol{u}^+ - \boldsymbol{u}^+) = 0 \end{aligned}$$

[1] $\partial f^* = (\partial f)^{-1}$, $(\partial f)^{-1}$ is monotone as the inverse of a monotone operator is monotone. The subdifferential of the convex function f is monotone. As the normal cone operator is the subdifferential of a convex function, this is also a monotone operator.

Combining these equations yields $\mathbf{u}^+ = \frac{1}{2}(I+P)\mathbf{y}$ so that $J_{cS_2}(\mathbf{y}) = \frac{1}{2}(I+P)\mathbf{y}$. Returning to the original problem, we see that the Cayley operator $C_{cS_2}(\mathbf{y}) = (2J_{cS_2} - I)(\mathbf{y}) = P\mathbf{y}$, so this entire Cayley operator can be replaced by the permutation matrix P. This permutation matrix represents the transmission of data along the edges of the graph.

Now for the other resolvent: let $\boldsymbol{\lambda}^+ = J_{cS_1}(\mathbf{z})$ so that $\boldsymbol{\lambda}^+ = (I + cS_1)^{-1}(\mathbf{z}) \Leftrightarrow \boldsymbol{\lambda}^+ = \mathbf{z} - cS_1(\boldsymbol{\lambda}^+)$. Recall the definition for S_1: $-C\partial f^*(-C^T\boldsymbol{\lambda}) + \mathbf{d}$ so that

$$\boldsymbol{\lambda}^+ = \mathbf{z} - c\left(-C\partial f^*(-C^T\boldsymbol{\lambda}^+) + \mathbf{d}\right) \tag{4.6}$$

Let $\mathbf{x}^+$ be the subgradient of the convex conjugate:

$$\mathbf{x}^+ \in \partial f^*(-C^T\boldsymbol{\lambda}^+) \tag{4.7}$$

Substituting Eq. (4.7) into Eq. (4.6) yields

$$\boldsymbol{\lambda}^+ = \mathbf{z} + c\left(-C\mathbf{x}^+ + \mathbf{d}\right)$$

From Eq. (4.7) it can also be derived that $0 \in \partial f(\mathbf{x}^+) + C^T\boldsymbol{\lambda}^+$ which after substituting $\boldsymbol{\lambda}^+$ results in:

$$\mathbf{x}^+ = \arg\min_{\mathbf{x}} \left(f(\mathbf{x}) + \mathbf{z}^T(C\mathbf{x} - \mathbf{d}) + \frac{c}{2}\|C\mathbf{x} - \mathbf{d}\|_2^2\right)$$

Following the definition of the Cayley operator, $C_{cS_1} = 2J_{cS_1} - I$ becomes $\mathbf{y} = C_{cS_1}(\mathbf{z}) = 2\boldsymbol{\lambda}^+ - \mathbf{z}$. Thus, the steps will iterate in the following way:

$$\mathbf{x}_{(k+1)} = \arg\min_{\mathbf{x}} \left(f(\mathbf{x}) + \mathbf{z}_{(k)}^T(C\mathbf{x} - \mathbf{d}) + \frac{c}{2}\|C\mathbf{x} - \mathbf{d}\|_2^2\right) \tag{4.8a}$$
$$\boldsymbol{\lambda}_{(k+1)} = \mathbf{z}_{(k)} + c(C\mathbf{x}_{(k+1)} - \mathbf{d}) \tag{4.8b}$$
$$\mathbf{y}_{(k+1)} = 2\boldsymbol{\lambda}_{(k+1)} - \mathbf{z}_{(k)} \tag{4.8c}$$
$$\mathbf{z}_{(k+1)} = P\mathbf{y}_{(k+1)} \tag{4.8d}$$

For the sake of clarity, the dimensions of the variables and matrices are now specified:

- $\mathbf{x} \in \mathbb{R}^N$
- $C \in \mathbb{R}^{MxN}$
- $\mathbf{d} \in \mathbb{R}^M$
- $\boldsymbol{\lambda} \in \mathbb{R}^M$
- $\mathbf{y} \in \mathbb{R}^M$
- $\mathbf{z} \in \mathbb{R}^M$
- $P \in \mathbb{R}^{MxM}$

As discussed earlier, the permutation operator corresponds to exchange of variables. In Eq. (4.8d) it is clear that two neighbouring nodes exchange their variables z_{ij} and z_{ji} by this permutation operator. By first substituting Eq. (4.8c) into Eq. (4.8d) and then simplifying, an alternate expression for $\mathbf{z}_{(k+1)}$ can be found. This expression for $\mathbf{z}_{(k+1)}$ can be substituted into Eqs. (4.8a) and (4.8b), yielding

$$\mathbf{x}_{(k+1)} = \arg\min_{\mathbf{x}} \left(f(\mathbf{x}) + \mathbf{x}^T C^T P\boldsymbol{\lambda}_{(k)} + \frac{c}{2}\|C\mathbf{x} + PC\mathbf{x}^{(k)} - 2\mathbf{d}\|_2^2\right)$$
$$\boldsymbol{\lambda}_{(k+1)} = P\boldsymbol{\lambda}_{(k)} + c(C\mathbf{x}_{(k+1)} + PC\mathbf{x}_{(k)} - 2\mathbf{d})$$

These are exactly equal to the PDMM update equations in Eqs. (3.5a) and (3.6) under the assumption that $\rho_d = \rho_p^{-1}$, so the PDMM operation can indeed be cast as an monotonic inclusion problem with Peaceman-Rachford splitting. For the averaging problem, the definitions earlier in this section can be substituted into Eqs. (4.8a) to (4.8d) which results in:

$$\mathbf{x}_{(k+1)} = \arg\min_{\mathbf{x}} \left(\sum_{i\in V} \left(\frac{1}{2}(x_i - T_i)^2 \right) + \mathbf{z}_{(k)}^T C\mathbf{x} + \frac{c}{2}\|C\mathbf{x}\|_2^2 \right) \tag{4.9a}$$

$$\boldsymbol{\lambda}_{(k+1)} = \mathbf{z}_{(k)} + cC\mathbf{x}_{(k+1)} \tag{4.9b}$$

$$\mathbf{y}_{(k+1)} = 2\boldsymbol{\lambda}_{(k+1)} - \mathbf{z}_{(k)} \tag{4.9c}$$

$$\mathbf{z}_{(k+1)} = P\mathbf{y}_{(k+1)} \tag{4.9d}$$

It is important to observe that C is not full rank, and neither is PC. As $\mathbf{z}$ is only altered by these two matrices, there is an uncontrollable subspace of $\mathbf{z}$ which will be denoted by $\mathfrak{U} = \begin{bmatrix} C & PC \end{bmatrix}^{\perp}$. If $\mathbf{z}_{(0)}$ is initialized as a random vector, there is a large probability that a part of $\mathbf{z}_{(0)}$ is contained in this uncontrollable subspace $\mathfrak{U}$:

$$\prod_{\mathfrak{U}} \mathbf{z}_{(0)} \neq \mathbf{0}$$

Therefore, this part of $\mathbf{z}$ will never leave $\mathfrak{U}$ and only get permuted every iteration. At convergence, $\mathbf{z}$ will therefore not reach the actual fixed point but rather reach a limit cycle with two states $\bar{\mathbf{z}}_1$ and $\bar{\mathbf{z}}_2$, caused by the permutation of the uncontrollable subspace. These states have a fixed distance to the actual fixed point $\mathbf{z}_*$, where this distance is exactly equal to the norm of the part of $\mathbf{z}$ in $\mathfrak{U}$. As the controllable part of $\mathbf{z}$ still converges to $\mathbf{z}_*$, the derived $\mathbf{x}$ still converges to the fixed point $\mathbf{x}_*$, which is equal to the average of $\mathbf{x}$. As only the controllable part of $\mathbf{z}$ influences the solution $\mathbf{x}$, its behaviour is most relevant. The uncontrollable part of $\mathbf{z}$ might mask certain properties of the convergence of $\mathbf{z}$, which shows the necessity to nullify the uncontrollable part. An important counterargument will rise in the following chapter, where the transmission of data will be discussed. In fact, the variable that will be transmitted is $\mathbf{z}$, or a difference vector derived from $\mathbf{z}$, so the uncontrollable part might still have influence there. For these reasons, multiple scenarios were studied:

- Initialize $\mathbf{z}_{(0)} = \mathbf{0}$, so $\prod_{\mathfrak{U}} \mathbf{z}_{(0)} = \mathbf{0}$. This works perfect for MPDMM, but as will be discussed in the next chapter, quantization will add an error to $\mathbf{z}$. Therefore, it can occur that quantization injects an error into $\mathfrak{U}$ which causes the algorithm to converge to $\bar{\mathbf{z}}_1$ and $\bar{\mathbf{z}}_2$.
- Split $\mathbf{z}$ into its controllable and uncontrollable part by projecting onto $\mathfrak{U}$. This will show the behaviour of the controllable part while the error by the uncontrollable part can also be studied separately.
- Double iterations. By considering the difference between $\mathbf{z}_{(k)}$ and $\mathbf{z}_{(k+2)}$ as iterates, there will be convergence to a certain point $\bar{\mathbf{z}}$. This point is not the true fixed point $\mathbf{z}_*$, but it has a finite distance to $\mathbf{z}_*$. It can be proven that the double iterates $\mathbf{z}_{(k+2)}$ converge to this new fixed point. To do so, let $\mathbf{x}_*$ be the fixed point of $\mathbf{x}$. Combine Eqs. (4.9b) to (4.9d) at convergence to yield

$$\mathbf{z}_{(k+1)} = P\mathbf{z}_{(k)} + 2cP(C\mathbf{x}_* - \mathbf{d}) \tag{4.10}$$

Following this, the next iteration can be written as

$$\begin{aligned}\mathbf{z}_{(k+2)} &= PP\mathbf{z}_{(k)} + 2cPP(C\mathbf{x}_* - \mathbf{d}) + 2cP(C\mathbf{x}_* - \mathbf{d}) \\ &= \mathbf{z}_{(k)} + 2c\left((C\mathbf{x}_* - \mathbf{d}) + P(C\mathbf{x}_* - \mathbf{d})\right)\end{aligned} \tag{4.11}$$

Closer study of $(C\mathbf{x}_* - \mathbf{d}) + P(C\mathbf{x}_* - \mathbf{d})$ shows the following:

$$\begin{aligned}(C\mathbf{x}_* - \mathbf{d}) + P(C\mathbf{x}_* - \mathbf{d}) &= (C + PC)\mathbf{x}_* - 2\mathbf{d} \\ &= \begin{bmatrix} A \\ A \end{bmatrix}\mathbf{x}_* - \begin{bmatrix} \mathbf{b} \\ \mathbf{b} \end{bmatrix}\end{aligned}$$

These are exactly the primal feasibility constraints, so $\begin{bmatrix} A & A \end{bmatrix}^T \mathbf{x}_* - \begin{bmatrix} \mathbf{b} & \mathbf{b} \end{bmatrix}^T = \mathbf{0}$ for feasible $\mathbf{x}_*$. Applying this insight to Eq. (4.11) results in:

$$\begin{aligned}\mathbf{z}_{(k+2)} &= \mathbf{z}_{(k)} + 2c\left((C\mathbf{x}_* - \mathbf{d}) + P(C\mathbf{x}_* - \mathbf{d})\right) \\ &= \mathbf{z}_{(k)} \\ &= \bar{\mathbf{z}}\end{aligned}$$

Note that k can be chosen to be even or uneven, but the fixed point $\bar{\mathbf{z}}$ is different for the even and uneven iterates, resulting in $\bar{\mathbf{z}}_1$ and $\bar{\mathbf{z}}_2$.

To examine the behaviour of the algorithms, the norm $\|\mathbf{z}_{(k)} - \mathbf{z}_*\|$ plays a crucial role. To compute this, the actual $\mathbf{z}_*$ has to be known exactly. For the averaging problem, it is possible to compute the $\mathbf{z}_*$ explicitly, given $\mathbf{x}_*$. This can be done using the following method: Let $\mathbf{z}_{*,1}$ be a $\mathbf{z}$ with empty uncontrollable subspace after convergence, and let $\mathbf{z}_{*,2}$ be the $\mathbf{z}$ at the next iteration, following the iteration in Eq. (4.10) with $\mathbf{d} = \mathbf{0}$. As convergence is reached, the iteration after $\mathbf{z}_{*,2}$ will yield $\mathbf{z}_{*,1}$ again:

$$\mathbf{z}_{*,2} = P(\mathbf{z}_{*,1} + 2\rho C\mathbf{x}_*) \tag{4.12a}$$

$$\mathbf{z}_{*,1} = P(\mathbf{z}_{*,2} + 2\rho C\mathbf{x}_*) \tag{4.12b}$$

For these $\mathbf{z}$ updates, Eq. (4.8a) is used, again with $\mathbf{d} = \mathbf{0}$. This means the derivative of $f(\mathbf{x}_*) + \mathbf{z}_{(k)}^T C\mathbf{x} + \frac{c}{2}\|C\mathbf{x}_*\|_2^2$ with respect to $\mathbf{x}_*$ is zero:

$$\nabla f(\mathbf{x}_*) = -C^T\mathbf{z}_{*,1} - \rho C^T C\mathbf{x}_* \tag{4.13a}$$

$$\nabla f(\mathbf{x}_*) = -C^T\mathbf{z}_{*,2} - \rho C^T C\mathbf{x}_* \tag{4.13b}$$

Substituting Eq. (4.12a) into Eq. (4.13b) results in

$$\begin{aligned}\nabla f(\mathbf{x}_*) &= -C^T P\mathbf{z}_{*,1} - 2\rho C^T PC\mathbf{x}_* - \rho C^T C\mathbf{x}_* \\ &= -C^T P\mathbf{z}_{*,1} - \rho C^T PC\mathbf{x}_* - \rho C^T PC\mathbf{x}_* - \rho C^T C\mathbf{x}_* \\ &= -C^T P\mathbf{z}_{*,1} - \rho C^T PC\mathbf{x}_* - \rho C^T \underbrace{(PC\mathbf{x}_* + C\mathbf{x}_*)}_{=\mathbf{0}} \\ &= -C^T P\mathbf{z}_{*,1} - \rho C^T PC\mathbf{x}_* \end{aligned} \tag{4.14}$$

Rearranging Eq. (4.13a) and Eq. (4.14) yields the following set of equations:

$$-C^T\mathbf{z}_{*,1} - \underbrace{(\nabla f(\mathbf{x}_*) + \rho C^T C\mathbf{x}_*)}_{g_1} = 0$$

$$-C^T P\mathbf{z}_{*,1} - \underbrace{(\nabla f(\mathbf{x}_*) + \rho C^T PC\mathbf{x}_*)}_{g_2} = 0$$

So in more compact form:

$$-\begin{bmatrix} C^T \\ PC^T \end{bmatrix}\mathbf{z}_{*,1} = \begin{bmatrix} g_1 \\ g_2 \end{bmatrix}$$

$$\mathbf{z}_{*,1} = -\begin{bmatrix} C^T \\ PC^T \end{bmatrix}^{\dagger}\begin{bmatrix} g_1 \\ g_2 \end{bmatrix} \tag{4.15}$$

As only the controllable subspace of $\mathbf{z}$ is used in Eq. (4.15), there is no limit cycle and $\mathbf{z}_{*,1} = \mathbf{z}_{*,2} = \mathbf{z}_*$. This explicitly computed $\mathbf{z}_*$ can now be used to show the error $\|\mathbf{z}_{(k)} - \mathbf{z}_*\|$ at each iteration.

4.2. Monotone Primal-Dual Method of Multipliers for the Gaussian channel capacity problem

The final problem that was discussed in the PDMM preliminary was Eq. (3.9). This can be solved by MPDMM, using a similar method as in the averaging case, described in the previous section. Note that the optimization problem within PDMM is in this case a constrained optimization problem. The iterates are expressed in the same form as Eqs. (4.8a) to (4.8d):

$$\begin{aligned}\boldsymbol{\lambda}_{(k+1)}, \boldsymbol{\mu}_{(k+1)} &= \arg\min_{\boldsymbol{\lambda},\boldsymbol{\mu}}(q(\boldsymbol{\lambda},\boldsymbol{\mu})) = \arg\min_{\boldsymbol{\lambda},\boldsymbol{\mu}}\left(\sum_{i\in V}\left(\log(\frac{-1}{\mu_i+\lambda_i}) - \mu_i\sigma_i^2 - \lambda_i\sigma_i^2 - \frac{\mu_i}{N}\right) + \mathbf{z}_{(k)}^T C\boldsymbol{\mu} + \frac{\rho}{2}\|C\boldsymbol{\mu}\|_2^2\right) \\ &\quad \text{s.t. } \lambda_i \geq 0,\ \mu_i + \lambda_i < 0 \ \ \forall i \in V \end{aligned} \tag{4.16a}$$

$$\boldsymbol{\beta}_{(k+1)} = \mathbf{z}_{(k)} + \rho C\boldsymbol{\mu}_{(k+1)} \tag{4.16b}$$

$$\mathbf{y}_{(k+1)} = 2\boldsymbol{\beta}_{(k+1)} - \mathbf{z}_{(k)} \tag{4.16c}$$

$$\mathbf{z}_{(k+1)} = P\mathbf{y}_{(k+1)} \tag{4.16d}$$

As Eq. (4.16a) is now a constrained optimization problem, the proceedings are slightly different from the averaging case. Let C_i be the i-th column of C, so rewriting $q(\boldsymbol{\lambda},\boldsymbol{\mu})$ to fit all terms in the summation yields:

$$q(\boldsymbol{\lambda},\boldsymbol{\mu}) = \sum_{i\in V}\left(\log(\frac{-1}{\mu_i+\lambda_i}) - \mu_i\sigma_i^2 - \lambda_i\sigma_i^2 - \frac{\mu_i}{N} + \mathbf{z}_{(k)}^T C_i\mu_i + \frac{\rho}{2}C_i^T C_i\mu_i^2\right)$$

Taking the partial derivatives results in:

$$\partial_{\mu_i}(q(\boldsymbol{\lambda},\boldsymbol{\mu})) = \frac{-1}{\mu_i+\lambda_i} - \sigma_i^2 - \frac{1}{N} + C_i^T\mathbf{z} + \rho C_i^T C_i\mu_i = 0 \tag{4.17a}$$

$$\partial_{\lambda_i}(q(\boldsymbol{\lambda},\boldsymbol{\mu})) = \frac{-1}{\mu_i+\lambda_i} - \sigma_i^2 = 0 \tag{4.17b}$$

An expression for λ_i can be obtained by rewriting $\nabla_{\lambda_i}(q(\boldsymbol{\lambda},\boldsymbol{\mu}))$:

$$\lambda_i = -\mu_i - \frac{1}{\sigma_i^2}$$

Substituting this into the expression for $\nabla_{\mu_i}(q(\boldsymbol{\lambda},\boldsymbol{\mu}))$ yields:

$$-\frac{1}{N} + C_i^T\mathbf{z} + \rho C_i^T C_i\mu_i = 0$$

$$\implies \mu_i = \frac{\frac{1}{N} - C_i^T\mathbf{z}}{\rho C_i^T C_i} \tag{4.18}$$

The constraints on $\boldsymbol{\lambda}$ still have to be checked. If $\lambda_i < 0$ and the constraint is thus active, the optimal λ_i will be 0. In this case, Eq. (4.18) is no longer correct, but $\lambda_i = 0$ has to be substituted into Eq. (4.17a). This yields:

$$\frac{-1}{\mu_i} - \sigma_i^2 - \frac{1}{N} + C_i^T\mathbf{z} + \rho C_i^T C_i\mu_i = 0$$

$$\underbrace{-1}_{a_0} + \underbrace{\left(-\sigma_i^2 - \frac{1}{N} + C_i^T\mathbf{z}\right)}_{a_1}\mu_i + \underbrace{\rho C_i^T C_i}_{a_2}\mu_i^2 = 0$$

$$\mu_i = \frac{-a_1 \pm \sqrt{a_1^2 - 4a_0a_2}}{2a_2} \tag{4.19}$$

Recall the constraint on the pair μ_i,λ_i: $\mu_i + \lambda_i < 0$. As it is known that $\lambda_i = 0$, this reduces to $\mu_i < 0$. Further examination of Eq. (4.19) shows that there will always be exactly one solution for μ_i that is negative.

Proof. As the network is connected and $C_i^T C_i$ is exactly the number of neighbours of node i, $C_i^T C_i$ is strictly larger than zero. Therefore:

$$a_2 > 0$$
$$-4a_0a_2 = 4a_2 > 0$$
$$a_1^2 - 4a_0a_2 > a_1^2$$
$$\sqrt{a_1^2 - 4a_0a_2} > |a_1|$$

□

In Algorithm 2, the MPDMM algorithm for the GCC problem is summarized.

Algorithm 2: MPDMM algorithm for the GCC problem.

```
Data: $\sigma^2, n, rho, MaxIterations$
Result: $\mathbf{x}, \mathbf{z}$
$\boldsymbol{\mu} = 1$;
$\boldsymbol{\lambda} = 1$;
$\boldsymbol{\mu}_{Temp} = 1$;
$\boldsymbol{\lambda}_{Temp} = 1$;
$err\_x = \infty$;
$k = 0$;
while $k <$ MaxIterations && $err\_x > err\_bound$ do
    for $i$=1:$n$ do
        $\boldsymbol{\mu}_{Temp} = \frac{\frac{1}{N} - C_i^T \mathbf{z}}{\rho C_i^T C_i}$;
        $\boldsymbol{\lambda}_{Temp} = -\boldsymbol{\mu}_{Temp} - \frac{1}{\boldsymbol{\sigma}^2}$;
        if $\boldsymbol{\lambda}_{Temp} > 0$ then
            $\boldsymbol{\lambda} = \boldsymbol{\lambda}_{Temp}$;
            $\boldsymbol{\mu} = \boldsymbol{\mu}_{Temp}$;
        else
            $\boldsymbol{\lambda} = 0$;
            $a0 = -1$;
            $a1 = -\sigma_i^2 - \frac{1}{N} + C_i^T \mathbf{z}$;
            $a2 = \rho C_i^T C_i$;
            $\boldsymbol{\mu} = \frac{-a_1 - \sqrt{a_1^2 - 4a_0 a_2}}{2a_2}$;
        end
    end
end
```

As in the averaging case, there is an uncontrollable part of $\mathbf{z}$ which causes convergence to a limit cycle with two limit points instead of the actual single fixed point. Similar methods as in the averaging case can be used: projection onto the controllable and uncontrollable subspaces of $\mathbf{z}$, zero initialisation and two-step differences. Another important issue is that $\mathbf{z}_*$ cannot be computed explicitly in the same manner as in the averaging case. The easiest way to see this is to allow the constraints on $\boldsymbol{\lambda}$ and μ to be a barrier function, the subgradient does not yield a singleton at the edges of the feasible set. Therefore, the $\mathbf{z}_*$ will be estimated by the projection of $\bar{\mathbf{z}}$ on the controllable subspace of $\mathbf{z}$. This $\bar{\mathbf{z}}$ is only a valid limit point if $\mathbf{x}$ converges. But, as $\mathbf{x}_*$ can be computed explicitly, the convergence of $\mathbf{x}$ can be checked, which validates the $\bar{\mathbf{z}}$ and thus $\mathbf{z}_*$.

4.3. Inexact Krasnosel'skiĭ-Mann iterations

As will become clear in Chapters 5 and 6, the effect of quantisation on the iterations is one of the key aspects of this thesis. Therefore, the inexact Krasnosel'skiĭ-Mann iteration in Definition 4.1 [31] is used as a first study on the effect of these quantisation errors.

Definition 4.1. The inexact Krasnosel'skiĭ-Mann iteration is defined as

$$\begin{aligned} \mathbf{z}_{(k+1)} &= \mathbf{z}_{(k)} + \alpha_{(k)}(S(\mathbf{z}_{(k)}) + \boldsymbol{\epsilon}_{(k)} - \mathbf{z}_{(k)}) \\ &= \left((1-\alpha_{(k)})I + \alpha_{(k)}S\right)(\mathbf{z}_{(k)}) + \alpha_{(k)}\boldsymbol{\epsilon}_{(k)} \end{aligned} \tag{4.20}$$

where S is a non-expansive operator, and fix $S = \{x : x = S(x)\} \neq \emptyset$. $\boldsymbol{\epsilon}_{(k)}$ is the error of approximating $S(\mathbf{z}_{(k)})$. The error after each iteration $\mathbf{e}$, and an extra parameter τ can be defined.

$$\begin{aligned} \mathbf{e}_{(k)} &\triangleq (I-S)(\mathbf{z}_{(k)}) = \frac{(\mathbf{z}_{(k)} - \mathbf{z}_{(k+1)})}{\alpha_{(k)}} + \boldsymbol{\epsilon}_{(k)} \\ \tau_{(k)} &\triangleq \alpha_{(k)}(1-\alpha_{(k)}) \end{aligned}$$

It is clear to see the similarities between Eqs. (3.11) and (4.20). In fact, if $\alpha_{(k)} = \frac{1}{2}\ \forall k$ and $S = C_{cS_2} \circ C_{cS_1}$, Eq. (4.20) becomes:

$$\mathbf{z}_{(k+1)} = \frac{1}{2}(I + C_{cS_2} \circ C_{cS_1})(\mathbf{z}_{(k)}) + \frac{1}{2}\boldsymbol{\epsilon}_{(k)}$$

Hence, the Douglas-Rachford splitting iterations can be cast as inexact Krasnosel'skiĭ-Mann iterations, where $\frac{1}{2}\boldsymbol{\epsilon}_{(k)}$ describes an error, for example due to quantisation.

In [31, Proposition 1(iii)], it is proven that if $(\tau_{(k)})_{k\in\mathbb{N}} \notin l_+^1$ and $\left(\alpha_{(k)}\|\boldsymbol{\epsilon}_{(k)}\|_2^2\right) \in l_+^1$, the sequence of errors $(\mathbf{e}_{(k)})_{k\in\mathbb{N}}$ converges strongly to zero. Indeed, it is proven that the sequence $(\mathbf{z}_{(k)})_{k\in\mathbb{N}}$ converges weakly to a solution $\mathbf{z}_* \in \operatorname{fix} S$. Moreover, suppose that $C_2 = \sum_{k\in\mathbb{N}} \alpha_{(k)}\|\boldsymbol{\epsilon}_{(k)}\| < +\infty$, $M_{(k)} = \sum_{j=0}^{k} \alpha_j$ and $\bar{\boldsymbol{e}}_k = \frac{1}{M_{(k)}}\sum_{j=0}^{k} \mathbf{e}_j$ then $\|\bar{\boldsymbol{e}}_{(k)}\| \leq \frac{2(d_0+C_2)}{M_{(k)}}$ with $d_0 = d(\mathbf{z}_{(0)}, \mathbf{z}_*)$. This result confirms the initial hypothesis that convergence can still occur even if quantization errors are included. Further analysis can now be done, starting with a background on quantization.

5 Preliminary: Quantization

Thus far, we have used the notion of distributed algorithms to indicate problems that are solved throughout a graph. However, this requires data to be transmitted along the edges of this graph. To implement distributed algorithms, data compression is necessary and plays an important role, with both benefits and disadvantages. As discussed in the previous chapter, MPDMM requires the transmission of data. This transmission was cast in the permutation operator P. As $\mathbf{z} \in \mathbb{R}^n$ in general, a problem arises: the exact values can never be transmitted in a limited bandwidth transmission. Therefore, lossy encoding, although it inevitably introduces an error, will be required.

Quantization is the act of transcribing an input to a limited set of symbols, which can then be transmitted [32]. Within signal processing, quantization is used to map signals to a finite set of values, known as a *codebook*. This means multiple input values will be mapped to one specific code, so quantization itself can be seen as a mathematical operator. The range of the input values that correspond to a specific code is called the *cell*, where different codes can have different cell sizes and structures. Although many quantization schemes exist, the simplest and most intuitive one will be explained: the uniform scalar quantizer. For this quantizer, all cells except the outermost two have identical cell widths Δ, only the code corresponding to each cell is different. The code corresponding to each cell is $\frac{(2a+1)\Delta}{2}$, where $a \in \mathbb{N}, |a| \leq \frac{n}{2}$ and n the amount of cells/codes. The input-output relation of the uniform scalar quantizer with 4 levels is indicated in Fig. 5.1, the range is indicated in red. The codebook and corresponding cells are described in Table 5.1. Clearly, quantization using this scheme is lossy; there is no point to point mapping and information will therefore be lost. This is modelled by *quantization noise*, n_q. For the non-outermost cells, the quantization noise is limited to $\frac{\Delta}{2}$, while the outermost cells can have unlimited quantization noise. This unlimited quantization noise is referred to as an *out of range error*, as the input is outside the range of the quantizer ($[-2\Delta, 2\Delta]$ for the quantizer in Fig. 5.1). Effectively, this means the input of the quantizer is not only approximated by a nearby quantization level, but also limited in the norm. Not only the quantization error n_q but also out of range errors will turn out to have a significant role in the convergence of distributed algorithms.

5.1. Entropy

Earlier, the relation between quantization noise and the amount of cells was introduced. Clearly, allowing more cells results in less quantization noise. Two different situations occur: there can either be a limited

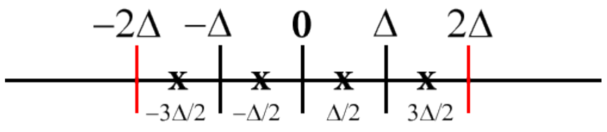


Figure 5.1: The input-output relation of a 2 bit uniform scalar quantizer with cellwidth Δ.

Table 5.1: Codebook and cell definition for a 2 bit uniform scalar quantizer with cellwidth Δ.

Codebook	Cell minimum	Cell maximum
$\Delta/2$	0	Δ
$3\Delta/2$	Δ	∞
$-\Delta/2$	Δ	0
$-3\Delta/2$	$-\infty$	$-\Delta$

number of possible input messages or an unlimited amount. Examples of limited possibilities are discrete systems and status updates, whereas the transmission of real or complex numbers is an example of a system with an unlimited amount of messages. In the case of a finite number of feasible messages, lossless quantization will occur if there are equally many cells, each containing one message. In the case of unlimited possibilities, there will always be a quantization error as infinitely many, infinitely small cells are required for lossless quantization. This is impossible in practice, although it can be emulated by using a relatively high bit rate. To determine the amount of bits required for transmission, the first step is to determine the entropy of the message. Two different cases are defined: the discrete and continuous entropy. For the discrete messages, so a finite number of feasible messages, the entropy is calculated using the following equation [33]:

$$H(X) = \sum_{i=1}^{a} p_i \log p_i$$

where there are a different messages, each with their own probability p_i. In case there are infinitely many possible messages, the continuous entropy is used, defined in Definition 5.1.

Definition 5.1. The differential entropy is defined as [34]:

$$h(X) = -\int_{\mathcal{X}} f_X(x)\log(f_X(x))dx$$

In this work, the messages consist of continuous mathematical variables and therefore, differential entropy is used. The problem is that $f_X(x)$, the distribution of the transmitted variables, is unknown. As a result, the exact bit rate is hard to determine. However, for scalar quantization the bit rate can be upper-bounded by the differential entropy of a Gaussian distribution, as long as the variances are equal [35, Theorem 3.1], resulting in Eq. (5.1).

$$h(X) \leq \frac{1}{2}\log(2\pi e\sigma^2) \tag{5.1}$$

Assuming this, differential entropy leads to the worst case entropy resulting in an upper bound on the entropy. As previously stated, this entropy is a measure for the amount of bits the quantizer uses. However, there is one more factor that plays an important role: distortion. Quantization of a continuous variable using a finite number of bits always introduces an error, or *distortion*. Intuitively, allowing a large distortion will require fewer bits because the result is allowed to have a larger error. Indeed, for a uniform quantizer, the bit rate can be approximated as in Eq. (5.2) [36].

$$H(X) \approx h(X) - \frac{1}{2}\log\left(12D_{\Delta_{(k)}}\right) \tag{5.2}$$

Here, $D_{\Delta_{(k)}}$ is the distortion caused by the cell width Δ. For this distortion, the following holds: $D_{\Delta_{(k)}} = \frac{\Delta_{2(k)}}{12}$ Combining Eqs. (5.1) and (5.2), and the definition of the distortion yields

$$\begin{aligned} H(X) &\leq \frac{1}{2}\log\left(2\pi e\sigma_X^2\right) - \frac{1}{2}\log\left(12D_{\Delta_{(k)}}\right) \\ &\leq \frac{1}{2}\log\left(\frac{2\pi e\sigma_X^2}{\Delta_{(k)}^2}\right) \end{aligned} \tag{5.3}$$

5.2. Dynamic Quantizer

For different scenarios, different optimal quantizers exist. In case of a steady entropy, the simplest quantizer is a fixed bit rate, fixed cell quantizer using entropy encoding. This means the entropy is determined and the bit rate and cell width are chosen subsequently. On the other hand, the entropy of the input variable might change over time, usually decreasing due to redundant information in the variable. To cope with this dynamic entropy, there are two options:

- Using an adaptive bit rate, the bit rate of the quantizer changes over time to adjust for the difference in entropy. The cells remain unchanged.
- Using a dynamic cell width, the cells are altered while the bit rate remains constant to accommodate the changing entropy.

Both options follow directly from Eq. (5.3), as either $\Delta_{(k)}$ is constant which results in a different $H(X)$ or vice versa. In this work, a dynamic cell width quantizer is chosen, while the bit rate is kept constant.

6 Quantization effects in distributed optimization schemes

After the background on quantization, the consequences of quantizing messages in MPDMM can be discussed. Recall from Chapter 4 that MPDMM can be written in the Peaceman-Rachford and Douglas-Rachford updates. Different problems will generate different operators S_1 and thus C_{cS_1}, although in general C_{cS_2} will be the permutation operator in MPDMM. The convergence of these operators can now be discussed, including the effects of quantization.

The convergence depends on two different error terms. As MPDMM does not converge instantly, there is a certain error at every iteration. This error is squared to yield the squared error $\|\mathbf{z}_{(k)} - \mathbf{z}_*\|^2$. Note that this error is present in both quantized and non-quantized MPDMM, as in many other distributed algorithms such as ADMM. The second error term is the quantization error $\|\mathbf{n}_{q,(k)}\|$. This extra error is only present while quantizing; its effect is to be determined. It is proven in [21] that for quadratic problems, the quantization noise will have little influence on the convergence rate if it is dominated by the MPDMM error. In other words, the quantization noise has to be smaller than the accuracy of the MPDMM iteration: $\|\mathbf{z}_{(k)} - \mathbf{z}_*\|^2 >> \|\mathbf{n}_{q,(k)}\|^2$.

6.1. Geometric convergence of the Monotone Primal-Dual Method of Multipliers variable z

For the PDMM algorithm, it is empirically shown that the primal variable has a geometric convergence bound. In MPDMM, not only the primal variable but also the controllable part of the dual variable converges geometrically. A theoretical proof for this bound is not yet available for all classes of problems, but can be derived for certain classes. To determine the convergence of MPDMM, we first examine $\|\mathbf{z}_{(k)} - \mathbf{z}_*\|^2$. By definition, $C_{cS_2} = P$ is Lipschitz and non-expansive. Suppose C_{cS_1} is also Lipschitz with parameter $L \in [0,1]$, so that they are both non-expansive. Then $C_{cS_2} \circ C_{cS_1}$ is also non-expansive. For the averaged operator $\frac{1}{2}(I + C_{cS_2} \circ C_{cS_1})$, the same holds. Due to this insight, S' will be used as a shortcut notation for either $C_{cS_2} \circ C_{cS_1}$ or its $\frac{1}{2}$-averaged operator. The quantization step is in practice incorporated in the transmission step, so in the permutation operator. However, they can be modeled as two separate operations. In Fig. 6.1, a block diagram of the non-averaged operator is displayed. A new variable is defined, $\tilde{\mathbf{z}}_{(k+1)} = S'(\hat{\mathbf{z}}_{(k)})$. The quantized version of $\tilde{\mathbf{z}}$ is now represented by $\hat{\mathbf{z}}$:

Definition 6.1. $\hat{\mathbf{z}}_{(k)} \triangleq \tilde{\mathbf{z}}_{(k)} + \mathbf{n}_{q,\tilde{z}_{(k)}}$ is the quantized version of $\tilde{\mathbf{z}}_{(k)}$

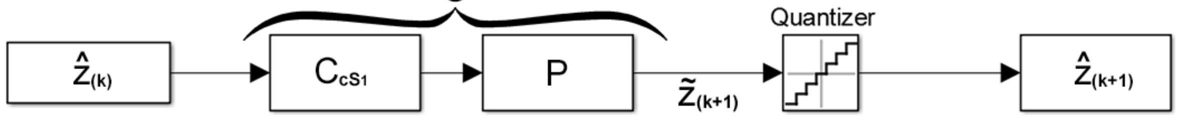


Figure 6.1: Block diagram of the non-averaged operator.

Combining this insight with Definition 6.1, the link between $\tilde{\mathbf{z}}^{(k+1)}$ and $\hat{\mathbf{z}}^{(k)}$ is the following:

$$\begin{aligned}\tilde{\mathbf{z}}_{(k+1)} &= S'(\hat{\mathbf{z}}_{(k)}) \\ &= S'(\tilde{\mathbf{z}}_{(k)} + \mathbf{n}_{q,\tilde{z}_{(k)}})\end{aligned}$$

Suppose $\mathbf{z}_*$ is a fixed point of S' such that $S'(\mathbf{z}_*) = \mathbf{z}_*$. As S' is non-expansive:

$$\begin{aligned}\|\tilde{\mathbf{z}}_{(k+1)} - \mathbf{z}_*\| &= \|S'(\tilde{\mathbf{z}}_{(k)} + \mathbf{n}_{q,\tilde{z}^{(k)}}) - S'(\mathbf{z}_*)\| \\ &\leq L\|\tilde{\mathbf{z}}_{(k)} - \mathbf{z}_* + \mathbf{n}_{q,\tilde{z}_{(k)}}\| \\ &\leq L\|\tilde{\mathbf{z}}_{(k)} - \mathbf{z}_*\| + L\|\mathbf{n}_{q,\tilde{z}_{(k)}}\| \qquad (6.1)\end{aligned}$$

For geometric convergence of $\tilde{\mathbf{z}}$ with parameter p: $\|\tilde{\mathbf{z}}_{(k+1)} - \mathbf{z}_*\| \leq p\|\tilde{\mathbf{z}}_{(k)} - \mathbf{z}_*\|$ so

$$\begin{aligned}L\|\tilde{\mathbf{z}}_{(k)} - \mathbf{z}_*\| + L\|\mathbf{n}_{q,\tilde{z}_{(k)}}\| &\leq p\|\tilde{\mathbf{z}}_{(k)} - \mathbf{z}_*\| \\ L\|\mathbf{n}_{q,\tilde{z}_{(k)}}\| &\leq (p-L)\|\tilde{\mathbf{z}}_{(k)} - \mathbf{z}_*\| \\ \|\mathbf{n}_{q,\tilde{z}_{(k)}}\| &\leq \frac{p-L}{L}\|\tilde{\mathbf{z}}_{(k)} - \mathbf{z}_*\|\end{aligned}$$

This leads to an important notion: $\frac{p-L}{L}$ has to be non-negative so $p \geq L$ and thus we conclude that the convergence bound is slower if quantization noise is added. However, Eq. (6.1) shows that if the quantization error is significantly smaller than the MPDMM error, the effect of quantization on the convergence is negligible. This follows from Eq. (6.1):

$$L\|\tilde{\mathbf{z}}_{(k)} - \mathbf{z}_*\| + L\|\mathbf{n}_{q,\tilde{z}_{(k)}}\| \approx L\|\tilde{\mathbf{z}}_{(k)} - \mathbf{z}_*\|$$

Recalling Eq. (6.1), we can consider the previous iteration k:

$$\|\tilde{\mathbf{z}}_{(k)} - \mathbf{z}_*\| \leq L\|\tilde{\mathbf{z}}_{(k-1)} - \mathbf{z}_*\| + L\|\mathbf{n}_{q,\tilde{z}_{(k-1)}}\|$$

such that combining this with Eq. (6.1) yields

$$\|\tilde{\mathbf{z}}_{(k+1)} - \mathbf{z}_*\| \leq L^2\|\tilde{\mathbf{z}}_{(k-1)} - \mathbf{z}_*\| + L\|\mathbf{n}_{q,\tilde{z}_{(k)}}\| + L^2\|\mathbf{n}_{q,\tilde{z}_{(k-1)}}\|$$

This gives rise to

$$\|\tilde{\mathbf{z}}_{(k+1)} - \mathbf{z}_*\| \leq L^{k+1}\|\tilde{\mathbf{z}}_{(0)} - \mathbf{z}_*\| + \sum_{i=0}^{k} L^{1+i}\|\mathbf{n}_{q,\tilde{z}_{(k-i)}}\|$$

As shown by the Krasnosel'skiĭ-Mann iterations, the sum of errors should be finite for MPDMM to converge.

Recall from the end of Chapter 4 that there is an uncontrollable subspace of $\mathbf{z}$ that causes convergence to two limit points $\bar{\mathbf{z}}_1$ and $\bar{\mathbf{z}}_2$, which both have a finite but non-zero distance from $\mathbf{z}_*$. In other words:

$$r \leq \|\mathbf{z}_{(k+1)} - \mathbf{z}_*\| \leq \|\mathbf{z}_{(k)} - \mathbf{z}_*\|$$

One of the solutions to this problem was to consider two step differences. This means a new operator S'' can be defined as two steps of the MPDMM algorithm:

$$S''(\mathbf{z}) = (S' \circ S')(\mathbf{z})$$

which has a fixed point

$$\bar{\mathbf{z}} \in \mathrm{fix}(S' \circ S') \supseteq \mathrm{fix}(S')$$

Here, $\mathrm{fix}(S' \circ S') \supseteq \mathrm{fix}(S')$ as the original $\mathbf{z}_*$ is always contained within $\mathrm{fix}(S' \circ S')$, but the alternating limit points of $\mathrm{fix}(S' \circ S')$ are not in $\mathrm{fix}(S')$. Studying the behaviour of two step iterations is therefore nothing different than studying this new operator with fixed point $\bar{\mathbf{z}}$:

$$\mathbf{z}_{(k+2)} = (S' \circ S')(\mathbf{z}_{(k)}) \Leftrightarrow \mathbf{z}'_{(k+1)} = S''(\mathbf{z}'_{(k)})$$

Moreover, the following holds:

$$0 \leq \|\mathbf{z}_{(k+2)} - \bar{\mathbf{z}}\| \leq \|\mathbf{z}_{(k)} - \bar{\mathbf{z}}\|$$

Therefore, all previous derivations hold for both one step and two step iterations.

6.2. Message definition: the difference vector

As in [21], the difference between iterations will be quantized and send instead of the actual variables. This difference will converge to zero if $\mathbf{z}$ converges to a fixed point. This difference will be referred to as $\mathbf{v}_{(k+1)} \triangleq \hat{\mathbf{z}}_{(k)} - \tilde{\mathbf{z}}_{(k+1)}$. As this difference is transferred, $\mathbf{v}_{(k+1)}$ is the input of the quantizer, yielding the quantized version $\hat{\mathbf{v}}_{(k+1)} \triangleq \mathbf{v}_{(k+1)} + \mathbf{n}_{q,v_{(k+1)}}$. The $\hat{\mathbf{z}}_{(k+1)}$ can then be reconstructed as

$$\begin{aligned}
\hat{\mathbf{z}}_{(k+1)} &\triangleq \hat{\mathbf{z}}_{(k)} - \hat{\mathbf{v}}_{(\mathbf{k+1})} \\
&= \hat{\mathbf{z}}_{(k)} - (\hat{\mathbf{z}}_{(k)} - \tilde{\mathbf{z}}_{(k+1)} + \mathbf{n}_{q,v_{(k+1)}}) \\
&= \tilde{\mathbf{z}}_{(k+1)} - \mathbf{n}_{q,v_{(k+1)}}
\end{aligned}$$

This is again linear in $\mathbf{n}_{q,v_{(k+1)}}$ so all previous derivations and results still hold. In fact, this is similar to Definition 6.1 if $\mathbf{n}_{q,\tilde{z}_{(k)}} = -\mathbf{n}_{q,v_{(k)}}$. A block diagram indicating the process can be seen in Fig. 6.2. Combining the

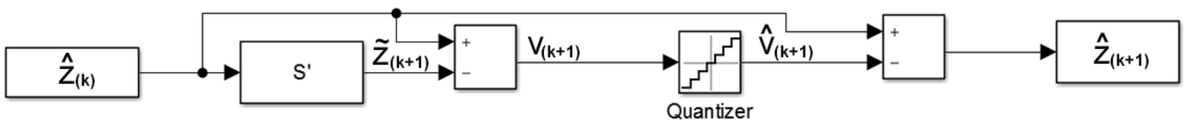


Figure 6.2: Block diagram of the full system, including quantization

definitions of $\mathbf{v}_{(k+1)}$ and $\hat{\mathbf{z}}_{(k)}$ yields an alternate definition for $\mathbf{v}_{(k+1)}$:

Definition 6.2. $\mathbf{v}_{(k+1)} = \hat{\mathbf{z}}_{(k)} - \tilde{\mathbf{z}}_{(k+1)} = \tilde{\mathbf{z}}_{(k)} - \mathbf{n}_{q,v_{(k)}} - \tilde{\mathbf{z}}_{(k+1)}$

Using Definition 6.2, an upper bound of $\mathbf{v}_{(k+1)}$ can be examined.

$$\begin{aligned}
\|\mathbf{v}_{(k+2)}\| &= \|\tilde{\mathbf{z}}_{(k+1)} - \mathbf{n}_{q,v_{(k+1)}} - \tilde{\mathbf{z}}_{(k+2)}\| \\
&\leq \|\tilde{\mathbf{z}}_{(k+1)} - \tilde{\mathbf{z}}_{(k+2)}\| + \|\mathbf{n}_{q,v_{(k+1)}}\| \\
&= \|S'(\tilde{\mathbf{z}}_{(k)} + \mathbf{n}_{q,\tilde{z}_{(k)}}) - S'(\tilde{\mathbf{z}}_{(k+1)} + \mathbf{n}_{q,\tilde{z}_{(k+1)}})\| + \|\mathbf{n}_{q,v_{(k+1)}}\| \\
&\leq L\|(\tilde{\mathbf{z}}_{(k)} + \mathbf{n}_{q,\tilde{z}_{(k)}}) - (\tilde{\mathbf{z}}_{(k+1)} + \mathbf{n}_{q,\tilde{z}_{(k+1)}})\| + \|\mathbf{n}_{q,v_{(k+1)}}\| \\
&= L\|\tilde{\mathbf{z}}_{(k)} + \mathbf{n}_{q,\tilde{z}_{(k)}} - \tilde{\mathbf{z}}_{(k+1)} - \mathbf{n}_{q,\tilde{z}_{(k+1)}}\| + \|\mathbf{n}_{q,v_{(k+1)}}\| \\
&= L\|(\mathbf{v}_{(k+1)}) - \mathbf{n}_{q,\tilde{z}_{(k+1)}}\| + \|\mathbf{n}_{q,v_{(k+1)}}\| \\
&\leq L\|\mathbf{v}_{(k+1)}\| + L\|\mathbf{n}_{q,\tilde{z}_{(k+1)}}\| + \|\mathbf{n}_{q,v_{(k+1)}}\| \\
&\overset{a}{=} L\|\mathbf{v}_{(k+1)}\| + (1+L)\|\mathbf{n}_{q,v_{(k+1)}}\|
\end{aligned} \tag{6.2}$$

In (a), the equality $\|\mathbf{n}_{q,\tilde{z}_{(k+1)}}\| = \|\mathbf{n}_{q,v_{(k+1)}}\|$ is used. Yet again, it follows that the quantization noise has influence, but it might be negligible compared to the variable $\mathbf{v}$ itself.

6.3. Two step differences

As discussed in Chapter 4, a solution to the problem concerning the uncontrollable part of $\mathbf{z}$ was to study double iterations. A variant on this is redefining $\mathbf{v}$ to send two-step differences as opposed to one-step differences:

Definition 6.3. The two step difference is defined as:

$$\mathbf{v}_{2,(k+1)} \triangleq \hat{\mathbf{z}}_{(k-1)} - \tilde{\mathbf{z}}_{(k+1)}$$

This leads to a new definition for $\hat{\mathbf{z}}_{(k+1)}$ as well:

$$\begin{aligned}
\hat{\mathbf{z}}_{(k+1)} &= \hat{\mathbf{z}}_{(k-1)} - \hat{\mathbf{v}}_{(k+1)} \\
&= \hat{\mathbf{z}}_{(k-1)} - (\hat{\mathbf{z}}_{(k-1)} - \tilde{\mathbf{z}}_{(k+1)} + \mathbf{n}_{q,v_{2,(k+1)}}) \\
&= \tilde{\mathbf{z}}_{(k+1)} - \mathbf{n}_{q,v_{2,(k+1)}}
\end{aligned} \tag{6.3}$$

Following Definition 6.2, $\mathbf{v}_{2,(k+1)}$ can be redefined as well:

Definition 6.4. $\mathbf{v}_{2,(k+1)} = \tilde{\mathbf{z}}_{(k-1)} - \mathbf{n}_{q,v_{2,(k-1)}} - \tilde{\mathbf{z}}_{(k+1)}$

Following the same steps as in Eq. (6.2) reveals that the same upper bound holds for $\mathbf{v}_{2,(k+1)}$ as well.

For both $\mathbf{v}_{(k+1)}$ and $\mathbf{v}_{2,(k+1)}$, the convergence properties can be determined using previous results. It is trivial that $\mathbf{v}_* = \mathbf{0}$ if $\mathbf{z}_{(k)}$ converges to a single fixed point $\mathbf{z}_*$. On the contrary, $\mathbf{v}_* \neq \mathbf{0}$ if $\mathbf{z}_{(k)}$ converges to two different states $\bar{\mathbf{z}}_1$ and $\bar{\mathbf{z}}_2$. However, $\mathbf{v}_{2,*} = 0$ in all cases, even if $\mathbf{z}$ converges to both $\bar{\mathbf{z}}_1$ and $\bar{\mathbf{z}}_2$. Therefore, $\|\mathbf{v}_{2,(k)}\|$ converges geometrically with parameter m if

$$\begin{aligned} \|\mathbf{v}_{2,(k+1)} - \mathbf{v}_*\| &\leq m\|\mathbf{v}_{2,(k)} - \mathbf{v}_*\| \\ \|\mathbf{v}_{2,(k+1)}\| &\leq m\|\mathbf{v}_{2,(k)}\| \end{aligned} \tag{6.4}$$

Combining Eqs. (6.2) and (6.4) yields

$$\begin{aligned} L\|\mathbf{v}_{(k)}\| + (1+L)\|\mathbf{n}_{q,v_{(k)}}\| &\leq m\|\mathbf{v}_{(k)}\| \\ \|\mathbf{n}_{q,v_{(k)}}\| &\leq \frac{m-L}{1+L}\|\mathbf{v}_{(k)}\| \end{aligned}$$

The covariance matrix of the transmitted variables (in this case $\mathbf{v}_{2,(k+1)}$) is of importance as it is used to determine the entropy of the message. This entropy can be computed by using the diagonal elements from the covariance matrix as variance σ^2 in Eq. (5.1).

Definition 6.5. The covariance matrix of vector $\mathbf{v}_{2,(k)}$) is defined as

$$\operatorname{cov}(\mathbf{v}_{2,(k)}) = \mathbb{E}[(\mathbf{v}_{2,(k)} - \mathbb{E}[\mathbf{v}_{2,(k)}])(\mathbf{v}_{2,(k)} - \mathbb{E}[\mathbf{v}_{2,(k)}])^H]$$

The definition of the covariance matrix can be rewritten to

$$\begin{aligned} \operatorname{cov}(\mathbf{v}_{2,(k)}) &= \mathbb{E}[\mathbf{v}_{2,(k)}\mathbf{v}_{2,(k)}^H] - \mathbb{E}[\mathbf{v}_{2,(k)}]\mathbb{E}[\mathbf{v}_{2,(k)}^H] \\ &\overset{(a)}{\leq} \mathbb{E}[\mathbf{v}_{2,(k)}\mathbf{v}_{2,(k)}^H] \\ &\leq \operatorname{trace}(\mathbb{E}[\mathbf{v}_{2,(k)}\mathbf{v}_{2,(k)}^H])I \\ &= \mathbb{E}[\operatorname{trace}(\mathbf{v}_{2,(k)}\mathbf{v}_{2,(k)}^H)]I \\ &= \mathbb{E}[\|\mathbf{v}_{2,(k)}\|^2]I \end{aligned}$$

where the positive semi-definiteness of $\mathbb{E}[\mathbf{v}_{2,(k)}]\mathbb{E}[\mathbf{v}_{2,(k)}^H]$ is used in (a). Note that this proof does not have any requirements on the expectation of $\mathbf{v}_{2,(k)}$. To summarize, the variance of $\mathbf{v}_{2,(k)}$ is upper bounded by $\mathbb{E}[\|\mathbf{v}_{2,(k)}\|^2]I$ and $\mathbf{v}_{2,(k)}$ converges geometrically with r for some $\mathbf{n}_{q,v_{2,(k)}}$, thus the variance decreases with at least $2r$.
Recall that the entropy depends on the variance:

$$H(X) \leq \frac{1}{2}\log\left(\frac{2\pi e\sigma_X^2}{\Delta_{(k)}^2}\right)$$

Therefore, if the decrease in variance is known, the same decrease in cell width can be implemented to maintain a fixed bit rate for the quantization, while decreasing the quantization noise.

Furthermore, for the one step difference $\mathbf{v}_{(k)}$, an interesting observation can be made. Note that $\mathbf{v}_{(k)}$, the difference between two consecutive iterations, will contain the uncontrollable part of $\mathbf{z}$. Although it is known that $\mathbf{v}_* \neq \mathbf{0}$, we still transmit the one step differences $\mathbf{v}_{(k)}$ using a quantizer with decreasing cell width and a constant amount of cells. As discussed in Chapter 5, quantization can be seen as a mathematical operator where out of range errors can occur if the input range of the quantizer is not sufficiently large. This property of the quantization operator can be used to alter the uncontrollable part of $\mathbf{z}$. The quantization operator limits the range of $\mathbf{v}_{(k)}$, which can be translated into a limitation on $\mathbf{z}_{(k)}$. As $\mathbf{v}_{(k)}$ is the difference between iteration k and $k+1$, during which the uncontrollable subspace gets permuted, the uncontrollable subspace has an extra limitation: it has to be repetitive. This means that the upper and lower half of the projection of $\mathbf{z}_{(k)}$ onto the uncontrollable subspace $\mathfrak{U}$ have to be equal:

$$P\prod_{\mathfrak{U}}\mathbf{z}_{(k)} = \prod_{\mathfrak{U}}\mathbf{z}_{(k)}$$

Therefore, out of range errors will suppress the two limit point alternating behaviour, resulting in convergence to a single limit point. However, this has no influence on the convergence of the controllable part of $\mathbf{z}_{(k)}$, nor on the convergence of $\mathbf{x}_{(k)}$. In the two step difference case, there is no direct limit on the difference between $\mathbf{z}_{(k)}$ and $\mathbf{z}_{(k+1)}$, so the uncontrollable part of $\mathbf{z}_{(k)}$ does not have to be identical after permutation. As a result, the flip state is not suppressed and two different $\bar{\mathbf{z}}$ can be found.

7 Results

To validate the theory in the previous chapters, simulations in MATLAB are discussed in this chapter. Two different cases are discussed: the averaging problem and the GCC problem. For each problem, a different topology was selected. For the averaging case, a large number of nodes were randomly distributed in a square area, while the GCC problem uses a significantly smaller amount of nodes. The difference in amount of nodes is due to the different nature of the two problems. For each problem, two different penalty parameters ρ are used: a near optimal one, empirically determined, and a non-optimal one. In Table 7.1, the parameters used in each problem are summarized.

Table 7.1: Parameters for the different problems

	Averaging Problem	GCC Problem
Number of nodes	100	10
Amount of edges	402	30
Optimal ρ	0.89	0.033
Non-optimal ρ	2.40	0.89

7.1. Averaging problem

The topology of the grid used for the averaging problem can be seen in Fig. 7.1. First of all, the MPDMM algorithm itself is verified. In Fig. 7.2a, the behaviour of $\mathbf{z}_{(k)}$ is visualized. Clearly, the controllable part of $\mathbf{z}_{(k)}$ converges to $\mathbf{z}_*$ while the uncontrollable part does not converge to $\mathbf{z}_*$ but stops converging at a finite distance. The controllable subspace of $\mathbf{z}_{(k)}$ will be referred to as $\mathbf{z}_{p,(k)}$ as only this subspace has an influence

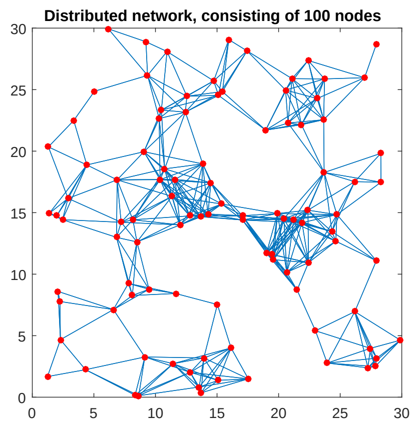
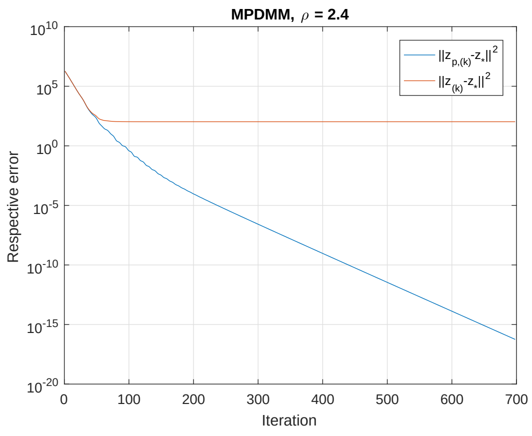
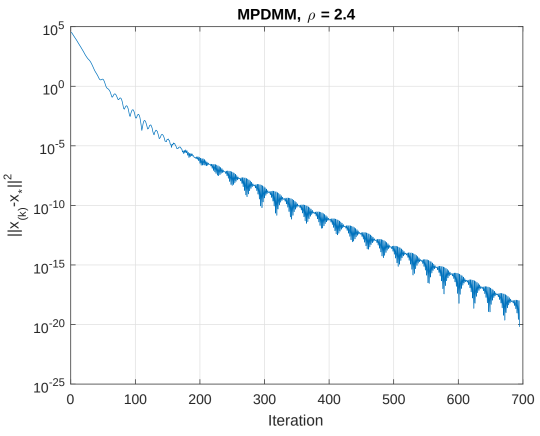


Figure 7.1: Randomly distributed network topology with 100 nodes.

on $\mathbf{x}_{(k)}$. Figure 7.2b shows the behaviour of $\mathbf{x}_{(k)}$, which converges to $\mathbf{x}_*$ despite the uncontrollable part of $\mathbf{z}_{(k)}$ being non-zero. For the sake of brevity, further plots of $\mathbf{x}_{(k)}$ will not be included as only $\mathbf{z}_{(k)}$ is (indirectly) transmitted along the edges.



(a) Convergence of z for MPDMM on the averaging problem



(b) Convergence of x for MPDMM on the averaging problem

Figure 7.2: Results of the primal and dual variables for the monotone interpretation of the PDMM algorithm on the averaging problem with non-optimal ρ.

7.1.1. Fixed cell width quantization

The next step is to include quantization, with a fixed bit rate and fixed cell width. The specifications of the fixed rate quantizer are:

- 10 bits,
- $\Delta = 0.01$ and fixed.

The effect of quantization can be clearly seen in Fig. 7.3. The norm of the error follows that of the regular MPDMM algorithm, until the quantization noise becomes dominant. By using a dynamic cell width quantizer, a fixed rate can be attained while constantly decreasing the quantization noise floor, so the convergence of MPDMM should not be affected.

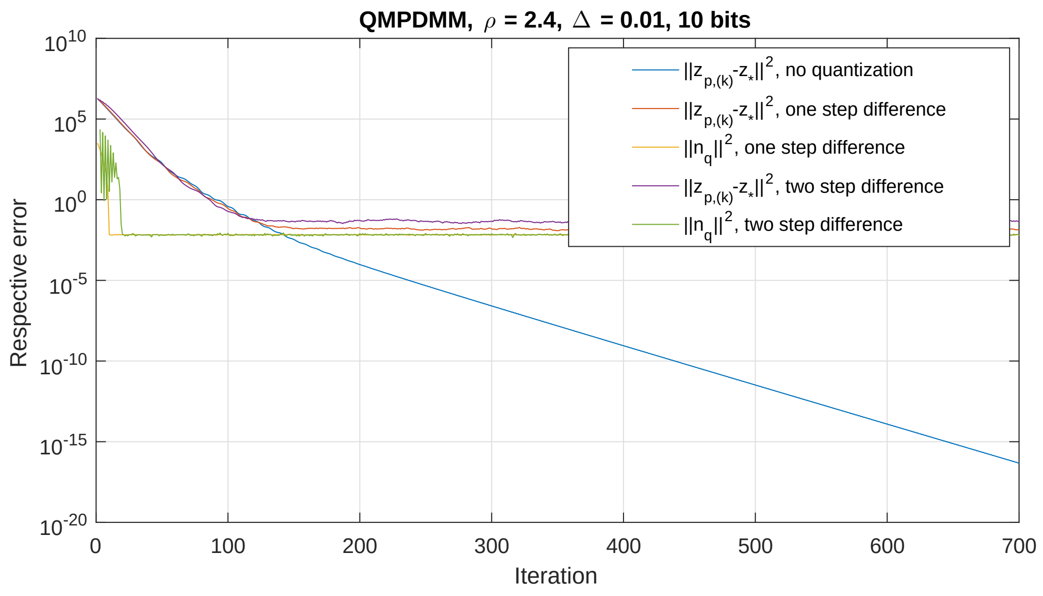


Figure 7.3: QMPDMM results of the averaging problem for non-optimal ρ using a 10 bit, fixed cell width quantizer.

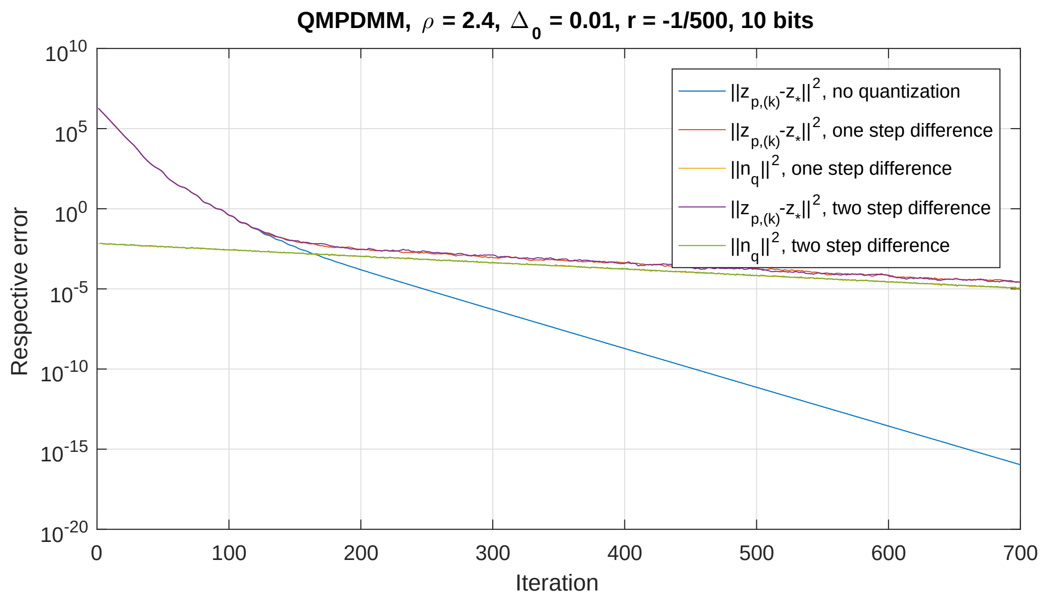


Figure 7.4: QMPDMM results of the averaging problem for non-optimal ρ using a 10 bit, dynamic cell width quantizer with cell width rate -1/500.

7.1.2. Dynamic cell width quantization

To decrease the quantization noise floor over time, a dynamic cell width quantizer is used. A geometric rate will be used as dynamics for the cell width. The rate of cell width change is determined empirically, based on the convergence rate of the MPDMM algorithm. As stated in Section 6.3, the change in entropy, the convergence of **v** and the convergence of **z** are all related. If the rate of the cell width is too high, all the messages will be outside the range of the quantizer so out of range errors will occur. If the rate of the cell width is too low, the quantization error will dominate over the MPDMM error and the overall convergence rate is affected. It turns out the convergence rate of the MPDMM algorithm is a good estimate for the rate of the cell width decrease.

The results for an initial cell width of 0.01 with rate $\frac{-1}{500}$ and a ten bit quantizer are included in Fig. 7.4. Clearly, the error in **z** is lower bounded by the maximum of the MPDMM error and the quantization noise N_q. It attains the normal MPDMM convergence until the quantization noise becomes dominant.

As theorized in the previous chapters, convergence will not be affected if the MPDMM error is dominant over the quantization noise. As the MPDMM error itself cannot be changed, the quantization noise has to be lowered, which is achieved by decreasing the cell width faster. This yields Fig. 7.5, which uses a rate of $\frac{-1}{50}$. Clearly, the convergence is not slower due to quantization, it is even faster than in the non-quantized case. This is an interesting observation, as less information is transmitted but the algorithm converges faster. An explanation for this behaviour has not yet been proven, although it is theorized that the out of range errors play an important role in the process.

Before studying this hypothesis, a different subject has to be discussed: the parameters of the quantizer. The rate of the quantizer cell width is not the only parameter that influences the convergence of the algorithm, but there are two more parameters: the initial cell width and the bit rate. There is a dependency between these three parameters as the bit rate and cell width determine the range of the quantizer. Therefore, the initial range of the quantizer is determined by the initial cell width and bit rate. To keep the initial range constant, the initial cell width is increased to ten while the bit rate is decreased to a single bit. The rate of the quantizer cell width is kept constant at $\frac{-1}{50}$. The simulation results can be seen in Fig. 7.6, where again the algorithm including quantization outperforms the non-quantized version. Using a single bit quantizer effectively means transmitting the sign, with a single reproduction level (either positive or negative), which makes the practical implementation very straightforward.

As previously stated, the quantized version of the algorithm outperforms the non-quantized algorithm in some cases. One of the main factors that could influence the performance is the range limitation by quantiza-

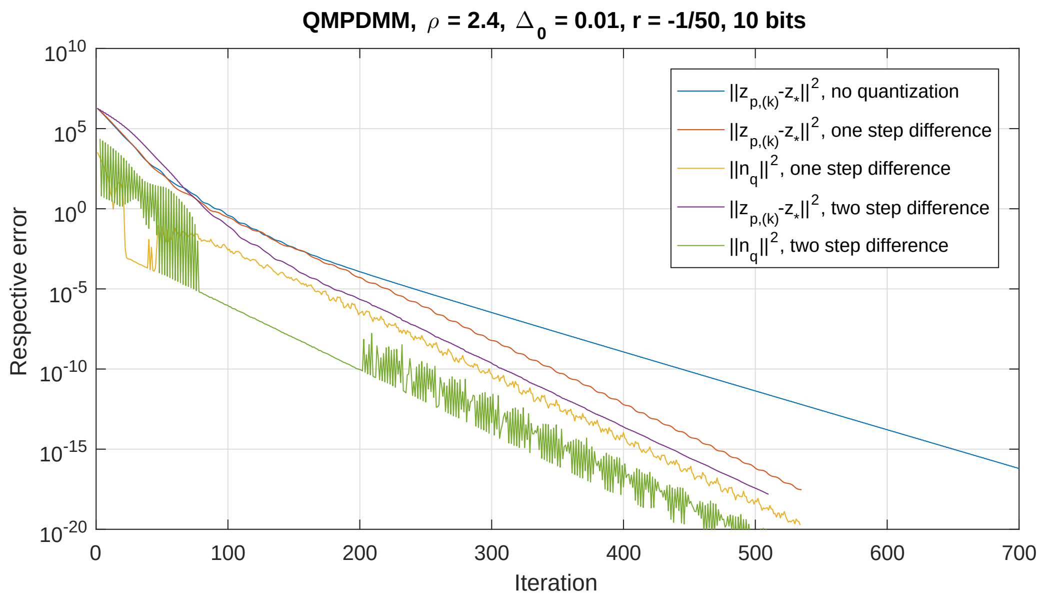


Figure 7.5: QMPDMM results of the averaging problem for non-optimal ρ using a 10 bit, dynamic cell width quantizer with cell width rate -1/50.

tion. The quantizer not only has a decreasing cell width, but as a result also a decreasing range. The resulting out of range errors, which lead to a non-uniform error distribution, might be key in this behaviour. The out of range errors in the simulation from Fig. 7.6 were studied and in Fig. 7.7, the percentage of out of range errors is displayed for each iteration. Clearly, this percentage is relatively high and constant. To investigate the effects of these out of range errors, the same simulation as in Fig. 7.6 is repeated, with the difference that infinitely many cells are used. As a result, there are no out of range errors, while the quantization error of the data that was in range in the original simulation remains unchanged. Clearly, the rate of convergence is significantly slower in the case with infinitely many cells and no out of range errors.

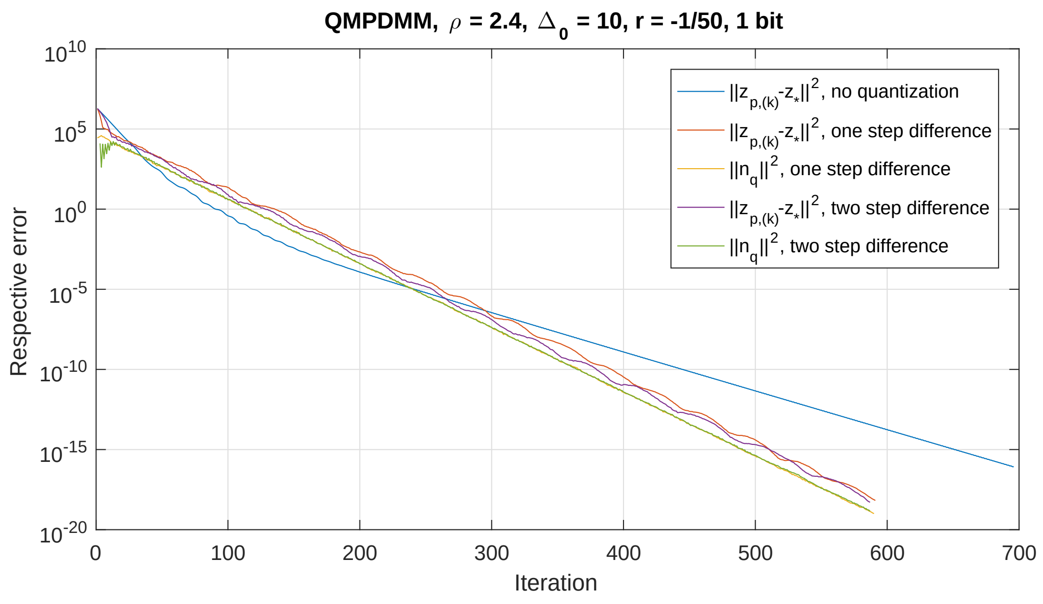


Figure 7.6: QMPDMM results of the averaging problem for non-optimal ρ using a single bit, dynamic cell width quantizer with cell width rate -1/50.

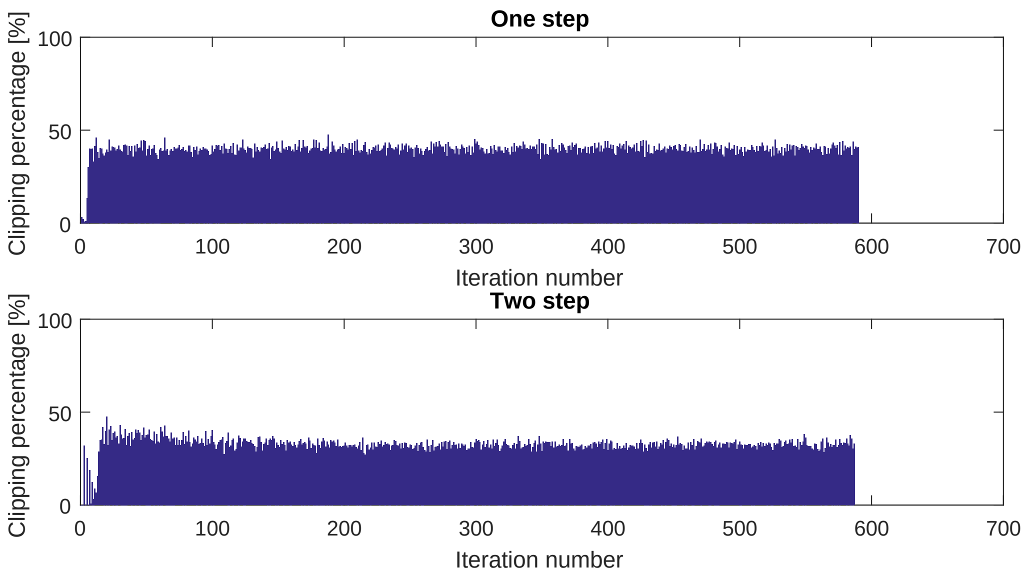


Figure 7.7: The percentage of out of range errors for the simulation in Fig. 7.6.

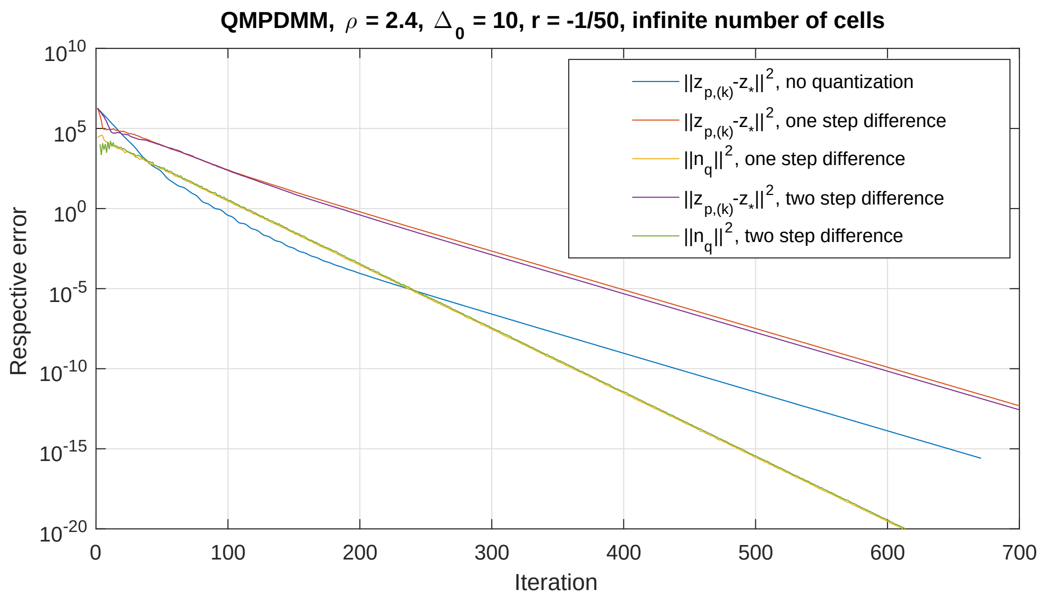


Figure 7.8: QMPDMM results of the averaging problem for non-optimal ρ using infinitely many dynamic cells with cell width rate -1/50.

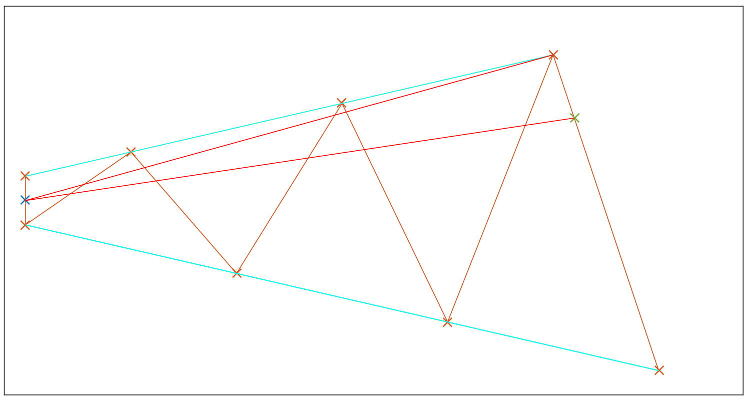


Figure 7.9: A visualization of the effect of out of range errors on the course of z.

An explanation for this behaviour, although not mathematically verified, is that the out of range errors limit the step size of the algorithm and will therefore help attain the lower bound set by the quantization noise. In Fig. 7.9, an example indication of the course of **z** is plotted. Because of the non-empty uncontrollable subspace, the **z** will alternate between two states, which converge to two different end points (orange x) with a fixed distance from $\mathbf{z}_*$ (blue x). However, if such a step is limited in range by quantization, the step will be shorter, as indicated by the green x. The distance between $\mathbf{z}_*$ and this new point can be smaller than distance between $\mathbf{z}_*$ and the original new point, as indicated with red lines. Therefore, the convergence for this iteration seems faster than without quantization. After this iteration, the next step is unpredictable as there is a new starting point, the green x. However, the assumption is made that the algorithm will converge in a similar fashion towards the two end points. In the simulation results, we see indeed in Fig. 7.8 a much larger gap between the quantization noise and the error in **z** than Fig. 7.6.

For the previous results, a non-optimal ρ was used. This means the algorithm can still be tuned to be faster, which might have an influence on the quantization effects. Indeed, the results for the optimal ρ are slightly different than those for a non-optimal ρ, which can be seen in Fig. 7.10. The MPDMM rate is attained, although for this ρ the quantized variant is not faster than the non-quantized algorithm.

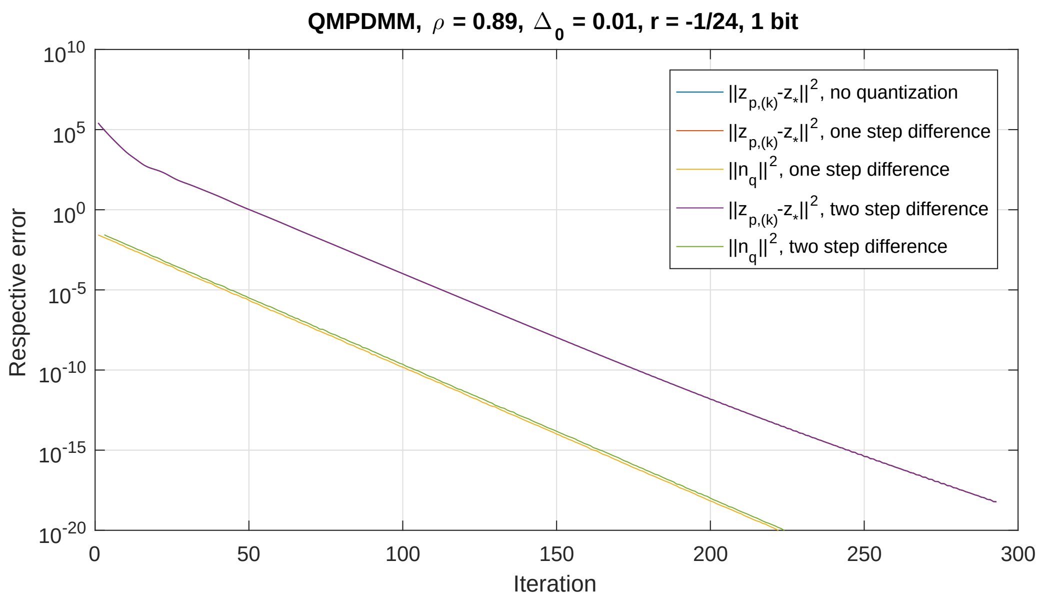


Figure 7.10: QMPDMM results of the averaging problem for near-optimal ρ using a single bit, dynamic cell width quantizer with cell width rate -1/24.

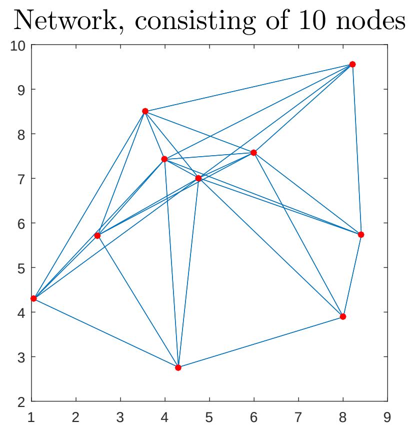


Figure 7.11: Randomly distributed network topology with 10 nodes.

7.2. Gaussian channel capacity problem

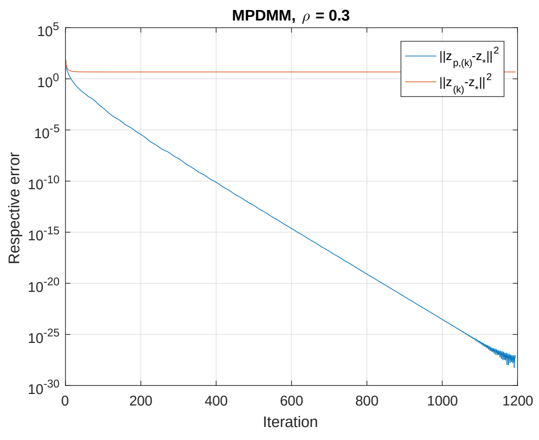
The topology of the grid used for the GCC problem can be seen in Fig. 7.11. For this problem too, the first step is to verify the MPDMM algorithm. First of all, recall that there is no exact solution for $\mathbf{z}_*$, so an estimation has to be done:

$$\mathbf{z}_* \approx \prod_{\mathfrak{U}^\perp} \mathbf{z}_{(final)} = \mathbf{z}_{(final)} - \prod_{\mathfrak{U}} \mathbf{z}_{(final)}$$

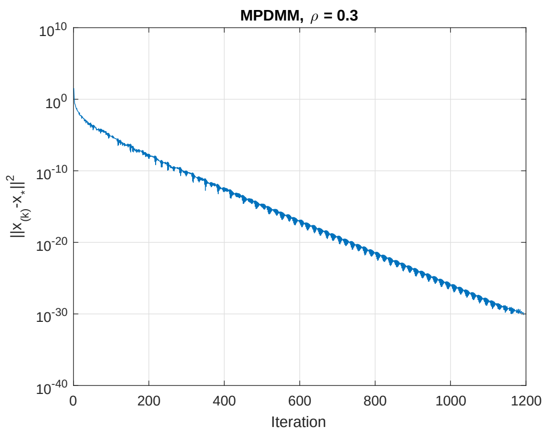
Note also that the solution of MPDMM is the combination of μ and $\boldsymbol{\lambda}$. To confirm the validity of the estimated $\mathbf{z}_*$, the convergence of $\mathbf{x}_{(k)}$ has to be confirmed. $\mathbf{x}_*$ can be determined analytically, from which μ_* and $\boldsymbol{\lambda}_*$ can be derived and vice versa. Therefore, the estimation of $\mathbf{z}_*$ can be verified by the computed $\mathbf{x}_{(final)}$, which should be equal to $\mathbf{x}_*$. In Fig. 7.12b, it is clear that $\mathbf{x}_{(k)}$ indeed converges to $\mathbf{x}_*$, which validates the estimation of $\mathbf{z}_*$. Figure 7.12a shows the convergence of the controllable subspace of $\mathbf{z}_{(k)}$ and $\mathbf{z}_{(k)}$ as a whole.

As the algorithm is now verified, the same steps as in the averaging case can be taken. Figure 7.13 shows that the quantized algorithm using a one bit quantizer is again a little faster than the original algorithm, while still converging to the same $\mathbf{z}_*$. This $\mathbf{z}_*$ was verified to be correct by the method described earlier.

For the optimal ρ, it was not possible to use a single bit quantizer without loss of performance. The best result, which can be seen in Fig. 7.14, was found using an eight bit quantizer, where again the convergence is slightly faster after quantization.



(a) Convergence of z for MPDMM on the GCC problem



(b) Convergence of x for MPDMM on the GCC problem

Figure 7.12: Results of the primal and dual variables for the monotone interpretatation of the PDMM algorithm on the GCC problem with non-optimal ρ.

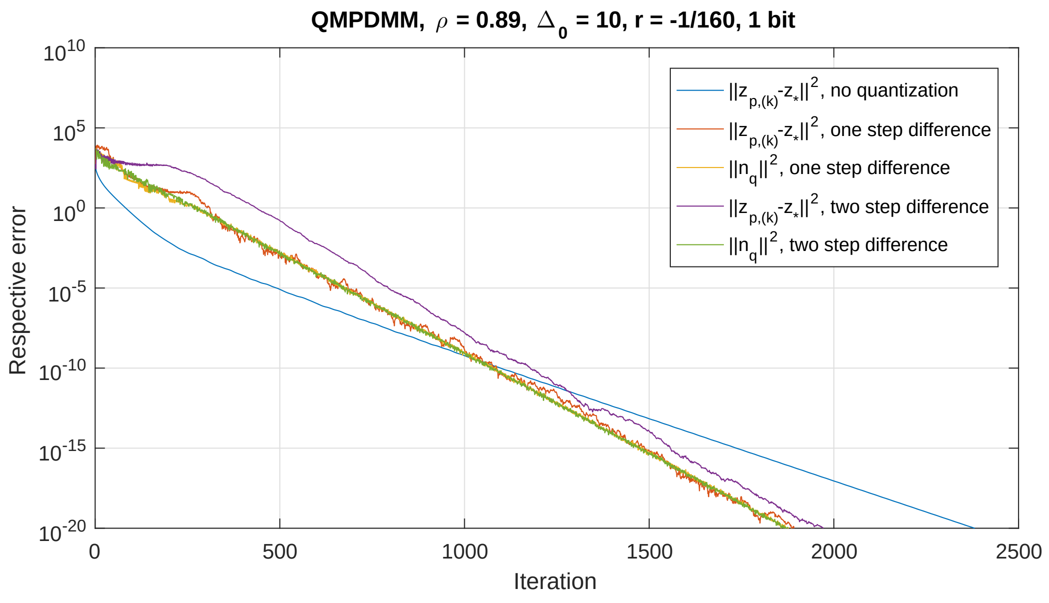


Figure 7.13: QMPDMM results of the GCC problem for non-optimal ρ using a one bit, dynamic cell width quantizer.

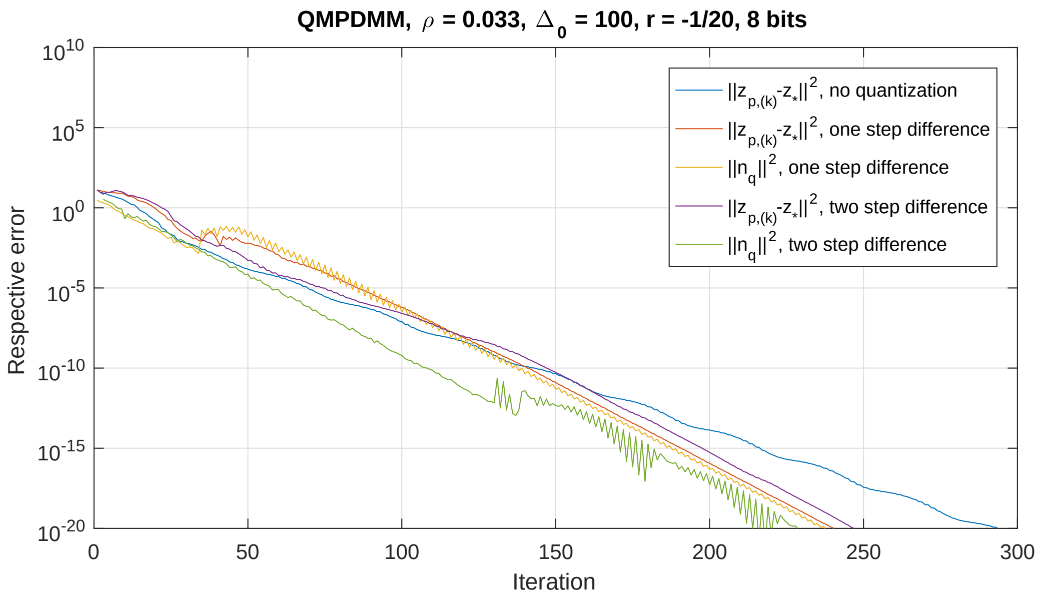


Figure 7.14: QMPDMM results of the GCC problem for near-optimal ρ using an eight bit, dynamic cell width quantizer.

8 Conclusion, discussion and future work

In this thesis, the effects of quantization on distributed optimization schemes were studied. Theoretical results as well as simulation results have shown that fixed rate quantization can be integrated in a distributed optimization scheme. The conclusions for each of the discussed classes of problems will now be discussed.

8.1. Conclusions for the averaging problem

The monotone Primal-Dual Method of Multipliers algorithm was verified, after which the effects of quantization were investigated. For the initial simulation, a fixed cell width quantizer was used. This confirmed that as long as the natural PDMM error dominates the quantization noise, quantization has no significant effect on the rate of convergence. However, as the PDMM error decreases over iterations, the quantization noise becomes more and more dominant, until they reach the same order. As soon as the quantization noise is too large, convergence is slowed to the rate of the quantization noise. In the case of a constant cell width, this means there is no rate and thus the algorithm stops converging.

By using the dynamic cell width quantizer, the quantization noise has a constant rate and the convergence of the PDMM algorithm is indeed bounded by that of the natural algorithm itself and the quantization noise. By estimating the rate of the PDMM algorithm and using that rate to decrease the cell width, it turned out there was no significant loss in performance, and in some cases even a gain in convergence speed. Even with a single bit quantizer, it was possible to converge as fast or even faster than the non-quantized PDMM algorithm for some problems. If the ρ parameter was tuned optimally, there was no longer an increase in convergence rate, but the original convergence rate was attained. The reasons for this gain in speed are still to be determined, although it is expected that the out of range errors introduced by the quantizer play a key role in this behaviour. The performance of the two step difference message-passing implementation was slightly better than that of the one using one step difference messages.

8.2. Conclusions for the Gaussian channel capacity problem

Before any analysis was done, the implementation of the MPDMM algorithm for the GCC problem was verified. Then, the quantization effects were studied. For the GCC problem, the results of the fixed cell width quantizer are similar to that of the averaging case. The algorithm converges until the quantization error is in the same order as the error of the PDMM algorithm itself, then it follows the rate of the quantization noise. For a ten bit quantizer, this rate was below the natural PDMM rate and convergence was thus not negatively affected, and again, a decrease of bits to a single bit quantizer was possible without sacrificing performance. For the optimal ρ, the lowest rate was an eight bit quantizer, which had similar performance to that of the ten bit quantization scheme. The performance of the two step message-passing implementation and the one step message-passing implementation were not significantly different, although the one step messages seemed to be slightly faster for this problem.

8.3. Final conclusion

After the analysis of both the theoretical and empirical results, the research questions can now be answered:

It is possible to apply fixed rate quantization in distributed optimization schemes in synchronous operation without losing performance. The two types of quantization errors - quantization noise and out of range errors - have different effects on the convergence of said optimization schemes, but given certain constraints on the errors, the convergence rate of the optimization schemes is unaffected. The robustness of the optimization schemes against the quantization errors holds for applications where finite precision is the aim; infinite precision whilst maintaining performance is not possible for a non zero quantization error.

8.4. Discussion

In this work, some assumptions were made, which will now be revisited and discussed.

In Chapter 5, the comparison is made between the quantized version and the unquantized version of a variable. In the simulations, all variables are represented by a finite number of bits and thus quantized. However, the quantization that occurs by representation a variable as a 64-bit double according to the IEEE Standard 754 yields such a small quantization error that it is neglected throughout this work.

In Chapter 6, the assumption is made that the PDMM and MPDMM algorithms converge geometrically. Although there is no strict proof yet for all types of problems, this is proven for the averaging case and there is strong empiric evidence that convergence rate is geometric for many more problems, such as the GCC problem. Therefore, this assumption was made and verified by simulation for each problem before the effects of quantization were studied.

Chapter 7 states the parameters for each simulation, which were often tuned empirically. There are three important tuning parameters: the bit rate, the initial cell width and the rate of cell width decrease. It might therefore be possible to achieve similar performance with fewer bits, by using a larger initial cell width and different rate of cell width decrease. The exact dependence of the convergence speed on these three parameters is not known exactly, although it is suspected that they are not independent of each other. The exact relation between these parameters was not in the scope of this thesis, but may provide valuable information for the practical implementation of distributed optimization schemes.

8.5. Future work

Apart from the assumptions, there are also some elements that were not studied or just briefly mentioned and not further studied. These aspects will now be discussed.

8.5.1. Asynchronous operation

In this work, the PDMM algorithm was configured in synchronous operation. However, this has a couple of disadvantages in practice, such as the need for a synchronous clock signal at all nodes, and the transmit and receive phases being simultaneous. Therefore, a study on the effects in asynchronous operation, where the nodes transmit alternately at a random interval, would be recommended.

8.5.2. Piecewise affine and fully dynamic rate

Thus far, we have seen that the algorithm including quantization has two possible outcomes:

- The algorithm converges until machine precision.
- The algorithm stops converging before the machine precision level is achieved.

The latter is caused by the quantization noise level being too high compared to the PDMM error itself. In the dynamic cell width scenario, this is typically caused by a too large rate of cell width decrease. This inspired an idea for a different rate for the quantizer cell width: start off with a steep rate and lower this rate over time to create a piecewise affine rate. A disadvantage is that when the algorithm stops converging due to the cell width being too small, there is no way of continuing. To deal with this issue, the following suggestion is made:

- Start with an overestimate of the convergence rate of the PDMM algorithm as rate of change for the dynamic cell width.
- Iterate until the algorithm stops converging.

- Increase the cell width slightly and lower the rate of change.
- Repeat this iterative process.

This method will maximize the rate and correct for a too fast change in cell width. For the practical implementation, the combination of such dynamics and asynchronous operation might be very powerful, and therefore the suggestion is made to investigate this in future work.

8.5.3. Packet loss and dynamic graphs

In this thesis, the assumption is made that there is a fixed graph with nodes that all contain local data. However, one of the benefits of distributed optimization schemes is that they allow for a dynamic graph, where nodes enter and leave the network. The nature of these networks is often wireless, which implies lossy transmission. This lossy transmission is not only in the form of quantization, which was the centre of this work, but also the possibility of corrupt transmissions and packet losses. The effect of packet losses in combination with quantization was not researched in this work, but will contain valuable information for the practical implementation of the quantized distributed optimization schemes.

Bibliography

[1] F. Zhao and L. J. Guibas, *Wireless sensor networks: an information processing approach.* Morgan Kaufmann, 2004.

[2] V. Mayer-Schönberger and K. Cukier, *Big data: A revolution that will transform how we live, work, and think.* Houghton Mifflin Harcourt, 2013.

[3] H. Farhangi, "The path of the smart grid," *IEEE power and energy magazine*, vol. 8, no. 1, 2010.

[4] M. Zhu and S. Martínez, "On distributed convex optimization under inequality and equality constraints," *IEEE Transactions on Automatic Control*, vol. 57, no. 1, pp. 151–164, 2012.

[5] T. C. Aysal, M. J. Coates, and M. G. Rabbat, "Distributed average consensus with dithered quantization," *IEEE Transactions on Signal Processing*, vol. 56, no. 10, pp. 4905–4918, 2008.

[6] R. Carli, F. Fagnani, P. Frasca, and S. Zampieri, "Gossip consensus algorithms via quantized communication," *Automatica*, vol. 46, no. 1, pp. 70–80, 2010.

[7] T. Li, M. Fu, L. Xie, and J.-F. Zhang, "Distributed consensus with limited communication data rate," *IEEE Transactions on Automatic Control*, vol. 56, no. 2, pp. 279–292, 2011.

[8] J. Pearl, "Fusion, propagation, and structuring in belief networks," *Artificial intelligence*, vol. 29, no. 3, pp. 241–288, 1986.

[9] M. J. Wainwright, M. I. Jordan *et al.*, "Graphical models, exponential families, and variational inference," *Foundations and Trends® in Machine Learning*, vol. 1, no. 1–2, pp. 1–305, 2008.

[10] C. C. Moallemi and B. Van Roy, "Convergence of min-sum message passing for quadratic optimization," *IEEE Transactions on Information Theory*, vol. 55, no. 5, pp. 2413–2423, 2009.

[11] 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.

[12] R. Carli, G. Como, P. Frasca, and F. Garin, "Distributed averaging on digital erasure networks," *Automatica*, vol. 47, no. 1, pp. 115–121, 2011.

[13] S. Boyd, "Alternating direction method of multipliers," in *Talk at NIPS Workshop on Optimization and Machine Learning*, 2011.

[14] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, "On the linear convergence of the admm in decentralized consensus optimization." *IEEE Trans. Signal Processing*, vol. 62, no. 7, pp. 1750–1761, 2014.

[15] G. Zhang and R. Heusdens, "Bi-alternating direction method of multipliers over graphs," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on.* IEEE, 2015, pp. 3571–3575.

[16] ——, "Bi-alternating direction method of multipliers," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on.* IEEE, 2013, pp. 3317–3321.

[17] ——, "Distributed optimization using the primal-dual method of multipliers," *IEEE Transactions on Signal and Information Processing over Networks*, 2017.

[18] E. Wei and A. Ozdaglar, "On the o (1= k) convergence of asynchronous distributed alternating direction method of multipliers," in *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE.* IEEE, 2013, pp. 551–554.

[19] M. Hong and Z.-Q. Luo, “On the linear convergence of the alternating direction method of multipliers,” *Mathematical Programming*, pp. 1–35, 2012.

[20] G. Zhang, R. Heusdens, and W. B. Kleijn, “On the convergence rate of the bi-alternating direction method of multipliers,” in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 3869–3873.

[21] D. H. Schellekens, “Quantization effects in pdmm,” Master’s thesis, TU Delft, June 2016, http://resolver.tudelft.nl/uuid:96cb75a3-7cf9-41a4-bce9-bc6a6ff5eb7e.

[22] R. Penrose, “A generalized inverse for matrices,” in *Mathematical proceedings of the Cambridge philosophical society*, vol. 51, no. 3. Cambridge University Press, 1955, pp. 406–413.

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

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

[25] E. K. Ryu and S. Boyd, “Primer on monotone operator methods,” *Appl. Comput. Math*, vol. 15, no. 1, pp. 3–43, 2016.

[26] F. Bénézit, A. G. Dimakis, P. Thiran, and M. Vetterli, “Order-optimal consensus through randomized path averaging,” *IEEE Transactions on Information Theory*, vol. 56, no. 10, pp. 5150–5167, 2010.

[27] J. Eckstein and D. P. Bertsekas, “On the douglas—rachford splitting method and the proximal point algorithm for maximal monotone operators,” *Mathematical Programming*, vol. 55, no. 1, pp. 293–318, 1992.

[28] D. W. Peaceman and H. H. Rachford, Jr, “The numerical solution of parabolic and elliptic differential equations,” *Journal of the Society for Industrial and Applied Mathematics*, vol. 3, no. 1, pp. 28–41, 1955.

[29] P.-L. Lions and B. Mercier, “Splitting algorithms for the sum of two nonlinear operators,” *SIAM Journal on Numerical Analysis*, vol. 16, no. 6, pp. 964–979, 1979.

[30] J. Douglas and H. H. Rachford, “On the numerical solution of heat conduction problems in two and three space variables,” *Transactions of the American mathematical Society*, vol. 82, no. 2, pp. 421–439, 1956.

[31] J. Liang, J. Fadili, and G. Peyré, “Convergence rates with inexact non-expansive operators,” *Mathematical Programming*, vol. 159, no. 1-2, pp. 403–434, 2016.

[32] K. Sayood, *Introduction to data compression*. Newnes, 2012.

[33] D. R. Anderson, *Information Theory and Entropy*. Springer, 2008.

[34] J. A. Thomas and T. M. Cover, *Elements of information theory*. John Wiley & Sons, 2006.

[35] C. Marsh, “Introduction to continuous entropy,” 2013.

[36] J. Østergaard, “Multiple-description lattice vector quantization,” Ph.D. dissertation, Delft University of Technology, 2007.