Distributed Optimisation Using Stochastic PDMM

Convergence, transmission losses and privacy

Master Thesis

Sebastian Jordan

Delft University of Technology

TUDelft

Distributed Optimisation Using Stochastic PDMM

Convergence, transmission losses and privacy

by

Sebastian Jordan

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended on Thursday 7 July, 2022 at 14:00.

Student number:	4590813	
Faculty:	EEMCS	
Master program:	Electrical Engineering	
Specialisation:	Signals and Systems	
Project Duration:	November, 2021 - July, 2022	
Thesis committee:	Prof. dr. ir. R. Heusdens,	TU Delft, supervisor
	Dr. ir. R. C. Hendriks,	TU Delft
	Prof. dr. ir. G. J. T. Leus,	TU Delft
	Dr. ir. J. H. Weber,	TU Delft

Cover Image: Edges of a random geometric graph, generated with Matlab.

TU Delft

Abstract

In recent years, the large increase in connected devices and the data that is collected by these devices has caused a heightened interest in distributed processing. Many practical distributed networks are of heterogeneous nature. Because of this, algorithms operating within these networks need to be simple, robust against network dynamics and resource efficient. Additionally, if privacy preservation methods are properly implemented, they add to the power of distributed processing by making it possible to leverage the data of many different users, without infringing the privacy of the individuals involved.

In this study we focus on the primal-dual method of multipliers (PDMM), which is a promising distributed optimisation algorithm that seems to be suitable for distributed optimisation in heterogeneous networks. Most theoretical work that can be found in existing literature focuses on synchronous versions of PDMM. However, in heterogeneous networks, asynchronous algorithms are favourable over synchronous algorithms. So far, simulation results have indicated that asynchronous PDMM converges and can even converge in the presence of transmission losses.

In this work we analyse the properties of stochastic PDMM, which is a general framework that can model variations of PDMM such as asynchronous PDMM and PDMM with transmission losses. We build upon previous empirical results of PDMM and formulate theoretical proofs to substantiate these results. After defining stochastic PDMM and proving its convergence, we compare a number of PDMM variations that have been mentioned throughout the literature. Lastly, we derive a lower bound for the variance of the auxiliary variable in the context of stochastic PDMM, assuming uniform updating probabilities. This lower bound indicates that subspace based privacy preservation is applicable to certain instances of stochastic PDMM, like asynchronous PDMM.

The main result of this work is a theoretical proof that shows that stochastic PDMM converges almost surely if the updating probabilities of each auxiliary variable are nonzero. Two important conclusions that follow from this proof are the almost sure convergence of asynchronous PDMM and unicast PDMM with transmission losses. Another useful result is the fact that subspace based privacy preservation is effective when using asynchronous PDMM.

Preface

After multiple years of studying in Delft, I am now nearing the end of my studies. Before you lies my thesis report that dives into various aspects of distributed optimisation using stochastic PDMM. This report was written as partial fulfilment of the degree Master of Science in Electrical Engineering at the Delft University of Technology, in the period November 2021-July 2022. During the research for this thesis I have studied existing literature, performed various numerical simulations and formulated theoretical proofs regarding various stochastic implementations of PDMM. On top of that, I have had the opportunity to present some of my results at the WIC Symposium on Information Theory and Signal Processing in the Benelux, which was a great experience. It has been a tremendously fruitful period in which I have developed my academic research capabilities and have managed to find some interesting results along the way.

In general, communication and collaboration have a positive influence on achievable results, this is also applicable to this thesis. For this reason, I would like to thank some people that helped me throughout my graduation period and positively influenced the end result.

Firstly, I would like to extend my sincere gratitude to my thesis supervisor Richard Heusdens. Throughout the entire process Richard has been extremely enthusiastic and supportive. At the beginning of the process Richard helped me get to grips with the rather involved mathematical background theory. During the latter stages, when I was diving into more and more sidepaths, he would help me keep focused on the main goals of this thesis.

I would also like to thank Richard Hendriks, Jos Weber and Geert Leus for being members of my thesis committee and showing interest in my research.

Furthermore, I would like to thank all people at CAS for providing a good atmosphere at the faculty. A special thanks goes out to the other MSc students at CAS for keeping things fun in the office and sparring about our research.

Lastly, I would like to thank my family and friends for being supportive throughout my studies, for having much appreciated coffee breaks and for help with my thesis, like proofreading and brainstorming.

I hope you enjoy reading this thesis!

Sebastian Jordan
Delft, June 2022

Contents

List of Figures

List of Tables

1 Introduction

The world around us is becoming increasingly connected. More and more electronic devices are being used and these devices are producing more and more data. Many devices have the ability to (wirelessly) connect to other devices and thus form large networks. Throughout evolution, the human species has greatly benefited from their ability to collaborate with others. The achievements that can be reached by many people combined surpass the sum of the individual achievements. This same philosophy can be used to leverage the computing power and data from a network of connected devices, by means of distributed signal processing.

Traditionally networks are structured in a centralised manner. Think of large servers that have many different devices connected to them to access information, see Figure 1.1a for an example. There are a number of downsides to centralised networks. The reliability is highly dependent on a single (central) device in the network. Any error or external influence that affects this single point of failure has large consequences for the entire network. Moreover, centralised networks are hard to scale. When the number of devices in the network grows, the required computing power of the central device increases because it will need to process more data. Another aspect that becomes harder when scaling up a centralised network is the communication between the central device and all other devices. The central device will require a high communication bandwidth to be able to connect to all devices and the distances between the central device and other devices can become large, resulting in high communication power requirements.

In recent years, the large increase in connected devices and the data that is collected by these devices has caused a heightened interest in distributed processing. Many practical networks are structured following a distributed network topology, see Figure 1.1b for an example. Distributed processing could leverage the full potential of large scale distributed networks. An important aspect of these types of networks is that they are often of heterogeneous nature, with connected devices having different computation, communication and power specifications. Because of the heterogeneity of modern networks, algorithms operating within these networks need to be:

- **simple**, due to the fact that many distributed algorithms are iterative and require many steps until convergence;
- **robust against network dynamics**, because of the varying and heterogeneous nature of large scale distributed networks;
- **resource efficient**, because it is common for some devices in the network to have limited computation and power resources.

The aspects mentioned above are not achievable with centralised networks and instead require distributed solutions. When distributed networks increase in size more devices are added to the network, but these devices also bring more (potential) computing power to the network. Because of this, distributed topologies are more scalable than centralised topologies. On top of that, distributed networks do not have a single point of failure and communication is only needed between nearby neighbours, thereby reducing the communication costs.

With regards to distributed algorithms, a lot of research has been done in the context of distributed averaging consensus, where the average of noisy measurements is calculated over the entire network.

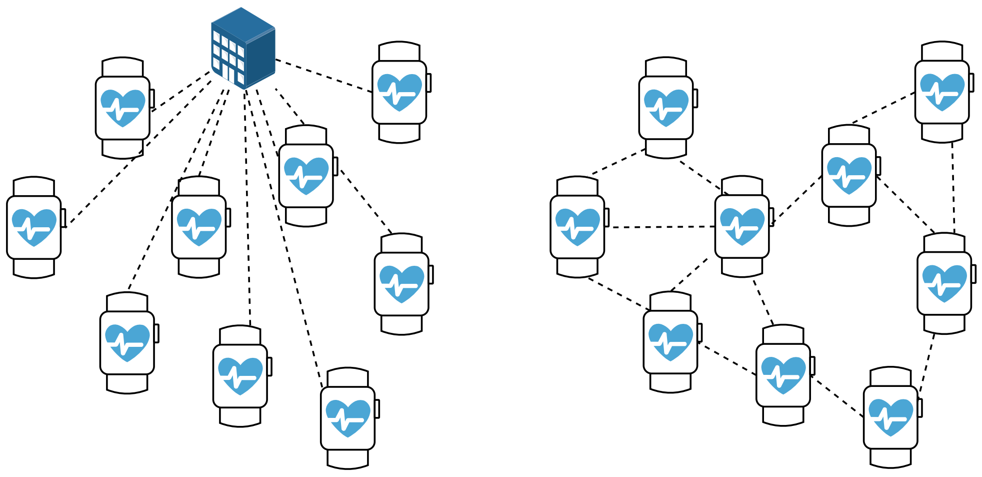

Figure 1.1: Example indicating the difference between a centralised network (a) and a distributed network (b), consisting of a number of smart watches collecting personal health data.

As presented in [1] gossip based algorithms are relatively simple and can be performed asynchronously. Randomised gossip is one of these algorithms, but has relatively slow convergence rates and is not robust against transmission losses or the removal of devices in the network. In [2] network knowledge is used to speed up convergence rates, by combining distributed averaging with geometric routing. However, this type of method is sensitive to network topology changes and transmission losses. Weighted gossip algorithms, like the one discussed in [3], use weights to add robustness against transmission loss. Although consensus based algorithms can in theory be applied for other distributed optimisation problems than just averaging, most of the existing analysis is done based on distributed averaging consensus. Furthermore, the algorithm in [3] is based on a second order Newton-Raphson scheme and thus requires the second derivatives of the cost functions.

A more general type of distributed algorithms is the class of convex optimisation based algorithms. One of these algorithms is the primal-dual method of multipliers (PDMM), introduced in [4]. Its convergence has been proved for synchronous implementations of the algorithm in [5] and simulations show it also converges if it is implemented asynchronously. Asynchronous PDMM has the advantage that there is no need for clock synchronisation between the nodes, which is extremely useful in heterogeneous distributed networks. Asynchronous algorithms also do not require additional communication overhead to keep the clocks synchronised and are not slowed down by the slowest device in the network. Furthermore, subspace based privacy preservation, as proposed in [6] and [7], is applicable to synchronous PDMM. In [5] it is shown that PDMM is closely related to the more commonly used ADMM algorithm (see [8]), but can achieve faster convergence rates. Additionally, by slightly rewriting the update equations, a broadcast implementation of PDMM was derived, instead of the usual unicast implementation. This broadcast implementation comes with a number of benefits. For example, it is a lot simpler to implement and requires only one transmitted message per iteration instead of one transmitted message per neighbour of the selected node each iteration. In simulations it is seen that, contrary to asynchronous unicast PDMM, broadcast PDMM does not converge in a situation with transmission losses. For many real world implementations, the wireless link between nodes will not be ideal, so robustness against transmission losses is an important aspect of distributed algorithms.

Privacy is another important aspect of distributed optimisation. Because of the nature of distributed networks, many different users can be connected and exchange information. Depending on the application, adequate measures may need to be taken in order to preserve the privacy of user data in such networks. If privacy preservation methods are properly implemented, they add to the power of distributed processing by making it possible to leverage the data of many different users, without infringing the privacy of the individuals involved. This property is particularly interesting with data relating to matters like health [9], financial transactions [10] and smart metering [11].

A common type of privacy preservation method is secure multiparty computation (SMPC), which is based on cryptographic techniques. These cryptographic techniques offer privacy preservation at the cost of high computational complexity. An example of such a technique, using fully homomorphic encryption, is presented in [12]. In some methods, like [13], a technique called secret sharing is used to lower the computational costs. With secret sharing, the private data is shared over different parties in the network, so that it is only possible to retrieve the private data if a sufficient number of devices combine their shares of the secret. However, this is usually paired with an increase in communication cost, which is undesirable in large scale distributed networks. Additionally, these methods are not fully decentralised because of the fact that a trusted third party is required during the necessary preprocessing phase.

Another commonly used privacy preservation method, introduced in [14], is differential privacy. This method is based on noise insertion on the messages that are shared between devices in the network. The computational costs are lower than full encryption and comparable to secret sharing based SMPC. In [15], differential privacy is used in the context of distributed constrained optimisation. The downside of differential privacy is the fact that there is always a trade-off between privacy and accuracy. Other techniques, like [16], use a differential privacy method that uses a trusted third party, thus not being fully decentralised.

Recently a promising type of privacy preservation has been proposed that addresses some of the downsides of SMPC and differential privacy. Subspace based privacy preservation, first proposed in [6], uses the random initialisation in a network dependent subspace as privacy preserving noise. The subspace in which the noise is inserted does not affect the optimisation variable. Because of this, there is no trade-off between privacy and accuracy. Furthermore, secure encryption is only needed in the initialisation phase of the algorithm. This means computational costs and communication costs are a lot lower when compared to SMPC.

In this work we analyse the properties of stochastic PDMM, which is a general framework that model variations of PDMM such as asynchronous PDMM and PDMM with transmission losses. We build upon previous empirical results and aim to find theoretical proofs to substantiate these results. This report revolves around the following research question:

- **Main:** What influence do stochastic updates and transmission losses have on the favourable properties of PDMM?

To address this question we focus on the following three subquestions:

- Can a proof be formulated for the convergence of stochastic PDMM?
- What is the difference between unicast and broadcast implementations and how does this effect convergence of stochastic PDMM?
- Is subspace based privacy preservation effective in combination with stochastic PDMM implementations?

1.1. Related Work

This work builds upon the existing research regarding PDMM. The algorithm was first introduced in [4], after which the link with monotone operator theory was made in [5]. In [5] the convergence of standard synchronous PDMM is proved for strongly convex and differential cost functions. If the cost function also has a Lipschitz continuous gradient a linear convergence rate is derived. It is also shown that the averaged version of synchronous PDMM converges for arbitrary convex closed and proper cost functions.

More recently a subspace based privacy method, that is compatible with PDMM, was presented in [6]. This method is further analysed in [17] and [7]. So far proofs relating to subspace based privacy preservation are based on synchronous versions of PDMM.

The background theory used for the proof of stochastic PDMM is mainly related to monotone operator theory [18] and probability theory [19]. Some convergence proofs for related ADMM based stochastic algorithms can be found in the literature. For instance the convergence of asynchronous ADMM (1/2-averaged operator) is proved in [20] and this proof is later generalised to work with a general θ-averaged ADMM adaptation in [21]. An alternative approach to the convergence proof of this class of mathematical problems is given in [22]. In [23] robustness against transmission loss is explicitly mentioned in the context of asynchronous θ-averaged ADMM.

Standard PDMM is a nonexpansive operator, which means in general it shows faster convergence than averaged operators like ADMM. To the best of our knowledge, there does not yet exist a proof of convergence for stochastic versions of nonexpansive operators, like PDMM, without the need for an additional operator averaging step.

1.2. Main contribution

The main contribution of this work is the convergence proof of stochastic PDMM, which is a general framework that includes asynchronous PDMM and PDMM with transmission losses. This convergence proof is a good step towards real world applications of PDMM, which will most likely be implemented asynchronously. The theoretical nature of this proof gives us more insight into the convergence properties of stochastic PDMM than the empirical results from the past and is a good step in the direction of the reliable use of stochastic PDMM in critical real world applications. Additionally, to the best of the author's knowledge, this is the first work that analyses and compares the properties of PDMM variations like unicast, broadcast and differential implementations. Lastly, this work provides the first analysis of subspace based privacy preservation applied to a stochastic distributed optimisation algorithm.

1.3. Organisation of the report

The report is organised as follows:

- In Chapter 2 the general nomenclature used throughout the work is defined.
- In Chapter 3 the most relevant background information is summarised and the PDMM algorithm is formally introduced.
- In Chapter 4 a formal proof is given for the convergence of stochastic PDMM.
- In Chapter 5 a number of PDMM variations are analysed and compared, focusing on the differences between unicast and broadcast PDMM in the presence of transmission losses.
- In Chapter 6 subspace based privacy is analysed for stochastic PDMM.
- In Chapter 7 numerical results are given to validate the statements made throughout the report.
- In Chapter 8 the conclusions of this work are summarised and recommendations are given regarding future work.

2 Nomenclature

The following notational conventions are used throughout this work:

- Lower case letters x are used to denote scalars, bold lowercase letters $\mathbf{x}$ are used to denote vectors and bold uppercase letters $\mathbf{X}$ are used to denote matrices.
- Calligraphic letters $\mathcal{X}$ are used to denote sets.
- The powerset, $2^{\mathcal{S}}$, is the set containing all subsets of set $\mathcal{S}$, which is defined as $2^{\mathcal{S}} = \{\mathcal{A}|\mathcal{A} \subseteq \mathcal{S}\}$.
- Bold uppercase letters are also used to denote operators. Where an operator $\mathbf{T}$ on $\mathbb{R}^n$ is a subset of $\mathbb{R}^n \times \mathbb{R}^n$ defined as $\mathbf{T} = \{(\mathbf{x}, \mathbf{y})|\mathbf{x}, \mathbf{y} \in \mathbb{R}^n\}$. With slight abuse of notation, by $\mathbf{T}(\mathbf{x})$ and $\mathbf{Tx}$ we mean the set$\{\mathbf{y}|(\mathbf{x}, \mathbf{y}) \in \mathbf{T}\}$. The difference between a matrix and an operator will become clear from the context in which bold uppercase letters are used.
- $\mathbf{I}$ is the identity matrix of appropriate dimensions
- $\mathbf{1}$ and $\mathbf{0}$ are the all ones and all zeros vector respectively, each of appropriate dimensions.
- Uppercase letters X are used to denote a scalar random variable. Vectors and matrices consisting of random variables have no explicit notation. It will be mentioned in the accompanying text when vectors or matrices are made up of random variables.
- $\mathbb{P}\{\omega\}$ is used to denote the probability that the event ω occurs and $\mathbb{E}[\bullet]$ is used to denote the statistical expectation.
- An instance of a sequence, x, at iteration number, k, is indicated as $x^{(k)}$.
- The notation f^* is used to denote the conjugate of function f, defined as $f^*(\mathbf{y}) = \sup_{\mathbf{x}}(\mathbf{y}^T\mathbf{x} - f(\mathbf{x}))$.
- To facilitate notation in this report, norms are assumed to be the two-norm, so $||x||$ is used to indicate $||x||_2$. Any other p-norm is denoted as $||x||_p$.
- We define $||\mathbf{a}||^2_{\mathbf{Q}} = \langle \mathbf{a}, \mathbf{Qa} \rangle$, where $\langle \bullet, \bullet \rangle$ is used to denote the inner product.
- Given a matrix $\mathbf{A}$, the range space of $\mathbf{A}$ is denoted by ran($\mathbf{A}$), where $\forall \mathbf{y} \in$ ran($\mathbf{A}$), $\exists \mathbf{u} \mid \mathbf{Au} = \mathbf{y}$. The kernel space of $\mathbf{A}$ is denoted by ker($\mathbf{A}$), where $\forall \mathbf{y} \in$ ker($\mathbf{A}$), $\mathbf{Ay} = \mathbf{0}$.

3 Background

In this chapter, some relevant background theory relating to distributed optimisation is given. First, in Section 3.1 we will state a number of useful properties from monotone operator theory that are used throughout the rest of the report. Next, in Section 3.2 some results from probability theory are introduced, which will be used in the stochastic convergence proof given in this report. After this, a summary of some general aspects of distributed optimisation is given in Section 3.3. This is followed by the derivation of PDMM and the definition of synchronous PDMM in Section 3.4. Finally, in Section 3.5 a subspace based privacy preservation method is presented that can be applied to PDMM.

3.1. Monotone operator theory

Monotone operator theory is very useful in the analysis and comparison of different optimisation algorithms. Using this framework, many seemingly different optimisation algorithms can be derived and analysed in a unified way. Furthermore, the most insightful derivation of PDMM, which is stated in [5] and summarised in Section 3.4, is based on monotone operator theory.

It is assumed that the reader of this report has basic knowledge of monotone operator theory, so in this section we will only state the results of monotone operator theory that are most relevant for the rest of this report. We would like to refer to Appendix A for a more complete overview of useful definitions and properties regarding monotone operator theory. For the interested reader, we recommend [18] and [24] for a more exhaustive treatment of monotone operators.

Definition 3.1. *If a sequence $\left(\mathbf{x}^{(k)}\right)_{k\in\mathbb{N}}$ in $\mathbb{R}^{(n)}$ possesses a subsequence that converges to a point $\mathbf{x}^* \in \mathbb{R}^{(n)}$, then $\mathbf{x}$ is a **sequential cluster point** of $\left(\mathbf{x}^{(k)}\right)_{k\in\mathbb{N}}$.*

Definition 3.2. *An operator $\mathbf{T}$ on $\mathbb{R}^n$ is **nonexpansive** if, for all $\mathbf{x},\mathbf{y} \in \mathbb{R}^n$, we have $||\mathbf{T}(\mathbf{y}) - \mathbf{T}(\mathbf{x})|| \leq ||\mathbf{y} - \mathbf{x}||$.*

Definition 3.3. *Let $\mathbf{T}$ be a nonexpansive operator and let $\theta \in (0,1)$. Then $\mathbf{T}$ is **averaged with constant θ**, or **θ-averaged**, if there exists a nonexpansive operator $\mathbf{R}$ such that $\mathbf{T} = (1-\theta)\mathbf{I} + \theta\mathbf{R}$.*

Definition 3.4. *Let $\mathbf{T}$ be an operator on $\mathbb{R}^n$, then, given $\mathbf{x}^{(0)} \in \mathbb{R}^n$, a sequence of **Banach-Picard iterations** is defined as $\mathbf{x}^{(k+1)} = \mathbf{T}\mathbf{x}^{(k)} \quad \forall k \in \mathbb{N}$.*

Theorem 3.1. *[18, Prop. 5.15-5.16] **Krasnosel'skii-Mann iterations:** Let $\mathbf{T}$ be an operator such that fix$(\mathbf{T}) \neq \varnothing$. The sequence*

$$x^{(k+1)} = x^{(k)} + \lambda^{(k)}\left(\mathbf{T}\left(x^{(k)}\right) - x^{(k)}\right) \quad \forall k \in \mathbb{N},$$

converges (weakly) to a point in fix$(\mathbf{T})$ in two cases:

1. *$\mathbf{T}$ is nonexpansive and $\left(\lambda^{(k)}\right)_{k\in\mathbb{N}}$ is a sequence in $[0,1]$ such that $\sum_{k\in\mathbb{N}} \lambda^{(k)}(1-\lambda^{(k)}) = +\infty$,*
2. *$\mathbf{T}$ is θ-averaged and $\left(\lambda^{(k)}\right)_{k\in\mathbb{N}}$ is a sequence in $[0,1/\theta]$ such that $\sum_{k\in\mathbb{N}} \lambda^{(k)}(1-\theta\lambda^{(k)}) = +\infty$.*

Theorem 3.2. *[18, Lem. 2.47] Let $(\mathbf{x}^{(k)})_{k\in\mathbb{N}}$ be a sequence in $\mathbb{R}^n$ and let $\mathcal{C}$ be a nonempty subset of $\mathbb{R}^n$. Suppose that, for every $\mathbf{x} \in \mathcal{C}$, $(||\mathbf{x}^{(k)} - \mathbf{x}||)_{k\in\mathbb{N}}$ converges and that every weak sequential cluster point of $(\mathbf{x}^{(k)})_{k\in\mathbb{N}}$ belongs to $\mathcal{C}$. Then $(\mathbf{x}^{(k)})_{k\in\mathbb{N}}$ converges weakly to a point in $\mathcal{C}$.*

3.2. Probability

Because asynchronous updates and transmission losses are both random processes, some results from probability theory will be required in the related convergence proofs in this report. In this section a few relevant results from probability theory are given. This includes the definition of a martingale and an important related convergence theorem. We will not cover the fine details of probability theory and martingales and for the interested reader, we recommend [19] for a more exhaustive explanation of probability theory.

Consider a probability space $(\Omega, \mathcal{F}, P)$, where Ω denotes the set of all possible outcomes, $\mathcal{F}$ denotes the family of all possible events and P denotes the family of probabilities for all events in $\mathcal{F}$. Let $(\mathcal{F}^{(k)})_{k\geq 0}$ be an increasing sequence of σ-algebras, having the property that $\mathcal{F}^{(k)} \subset \mathcal{F}^{(k+1)} \subset \mathcal{F}$, for all $k \geq 0$.

Definition 3.5. *[19, Def. 23.3] Consider a random vector $Y : \Omega \to \mathbb{R}^n$ and define $\mathcal{B}(\mathbb{R}^n)$ as the Borel σ-algebra of $\mathbb{R}^n$ (see [19, Thm. 2.1]). The **σ-algebra generated by** Y is*

$$\sigma(Y) = \left\{A \subset \Omega \, : \, Y^{-1}(B) = A, \text{for some } B \in \mathcal{B}(\mathbb{R}^n)\right\}.$$

Definition 3.6. *[19, Def. 24.1] A sequence of random variables $(X^{(k)})_{k\geq 0}$ is called **a martingale (a supermartingale** respectively), if*

1. *$\mathbb{E}\left[|X^{(k)}|\right] < \infty$, for each k;*
2. *$X^{(k)}$ is $\mathcal{F}^{(k)}$-measurable, for each k;*
3. *$\mathbb{E}\left[X^{(k+1)}|\mathcal{F}^{(k)}\right] = X^{(k)}$ a.s. ($\mathbb{E}\left[X^{(k+1)}|\mathcal{F}^{(k)}\right] \leq X^{(k)}$ a.s. respectively), for each k.*

Theorem 3.3. *[19, Cor. 27.1] If $X^{(k)}$ is a nonnegative supermartingale, then $\lim_{k\to\infty} X^{(k)} = X$ exists a.s., and is finite a.s.*

Definition 3.7. *Let $A^{(k)}$ be a sequence of events (in $\mathcal{A}$), we define $\limsup_{k\to\infty} A^{(k)} = \cap_{k=1}^{\infty}\left(\cup_{l\geq k} A^{(l)}\right)$.*

The event that is described in Definition 3.7 can be interpreted probabilistically as "$A^{(k)}$ **occurs infinitely often"**, which can be abbreviated as "**i.o.**", so $\limsup_{k\to\infty} A^{(k)} = \{A^{(k)} \quad \text{i.o.}\}$.

Definition 3.8. *[19, Def. 23.5] Let $Y \in L^1(\Omega, \mathcal{F}, P)$ and let $\mathcal{A}$ be a sub σ-algebra of $\mathcal{F}$, then the **conditional expectation** of Y given $\mathcal{A}$ is the unique element $\mathbb{E}[Y|\mathcal{A}]$ such that*

$$\mathbb{E}[YZ] = \mathbb{E}[\mathbb{E}[Y|\mathcal{A}]Z],$$

for all $Z \in L^2(\Omega, \mathcal{A}, P)$.

Theorem 3.4 (Jensen's Inequality). *[19, Thm. 23.9] Let $\varphi : \mathbb{R} \to \mathbb{R}$ be a convex function and let X and $\phi(X)$ be integrable random variables, for any sub σ-algebra $\mathcal{A}$,*

$$\varphi\left(\mathbb{E}[X|\mathcal{A}]\right) \leq \mathbb{E}\left[\varphi(X)|\mathcal{A}\right].$$

3.3. Problem statement

In this section we will mathematically define the problem statement of distributed optimisation. First, we define a distributed network and after this we formulate an optimisation problem over this network. This optimisation problem can be solved using different types of algorithms, one of which is PDMM, which will be introduced in Section 3.4.

3.3.1. Network definition

Consider a general undirected network consisting of n nodes, see Figure 3.1 for a simple example of such a network. The corresponding graph is described by $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, ..., n\}$ denotes the set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes the set of m undirected edges, so that $(i, j) \in \mathcal{E}$ if nodes i and j share a physical connection[1]. Additionally we define the set of directed edges $\mathcal{E}_{\text{dir}}$ as the set that has an entry $(i|j)$ and $(j|i)$ for each $(i, j) \in \mathcal{E}$ and thus contains $2m$ directed edges.

[1]We assume that the tuples (i, j) are always ordered such that $i < j$

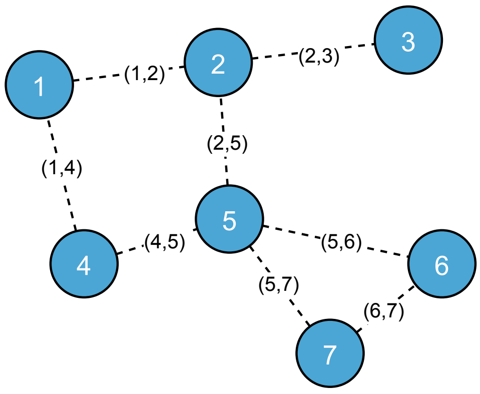

Figure 3.1: Example of a graph $\mathcal{G}(\mathcal{V},\mathcal{E})$ with $\mathcal{V}=\{1,...,5\}$ and $\mathcal{E}=\{(1,2),(1,4),...,(5,7),(6,7)\}$.

Furthermore, we use the following notational conventions:

- A variable x_i is related to node i;
- A variable $x_{i|j}$ is related to directed edge $(i|j)$;
- The neighbourhood of a node i is defined as $\mathcal{N}_i = \big\{\{j\in\mathcal{V}\,|\,(i,j)\in\mathcal{E}\}\cup\{j\in\mathcal{V}|(j,i)\in\mathcal{E}\}\big\}$.
- Each node i is equipped with a function $f_i\in\Gamma_0(\mathcal{H}^{n_i})$, where each function is dependent on a local variable $\mathbf{x}_i\in\mathbb{R}^{n_i}$ and Γ_0 denotes the set of all convex, closed and proper (CCP) functions.

3.3.2. Optimisation problem

We would like to solve the following optimisation problem:

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

which is defined over the graph $\mathcal{G}$ with constraints between entries defined by $\mathbf{A}_{i|j}\in\mathbb{R}^{m_{i|j}\times n_i}$ and $\mathbf{b}_{i|j}\in\mathbb{R}^{m_{i|j}}$. Let $n_{\mathcal{V}}=\sum_{i\in\mathcal{V}} n_i$ and $m_{\mathcal{E}}=\sum_{(i,j)\in\mathcal{E}} m_{i|j}$. This optimisation problem is separable and with suitable algorithms it can be solved in a distributed manner. Various techniques can be used to solve the optimisation problem stated in (3.1). In this work we use the distributed algorithm PDMM.

For many practical use cases distributed consensus is desired. This means that for all $(i,j)\in\mathcal{E}$ the constraints in (3.1) are defined using $\mathbf{A}_{i|j}=\mathbf{I}$, $\mathbf{A}_{j|i}=-\mathbf{I}$ and $\mathbf{b}_{i,j}=\mathbf{0}$. Because consensus problems are very common, throughout this report we will assume the network constraints correspond to these consensus constraints. The results can also be extended to more general problems where this is not the case.

3.4. PDMM

In this section we will summarise the derivation of PDMM. After this, we will state the definition of synchronous PDMM, which is the most analysed version of PDMM so far. Finally, we briefly introduce PDMM variations that will be discussed in detail in Chapter 5.

3.4.1. Derivation

Consider a distributed optimisation problem as defined in (3.1). The Lagrangian dual of this problem is given by

$$\min_{\boldsymbol{\nu}_{i|j},\forall(i,j)\in\mathcal{E}} \quad \sum_{i\in\mathcal{V}} f_i^*\left(-\sum_{j\in\mathcal{N}_i}\mathbf{A}_{i|j}^T\boldsymbol{\nu}_{i|j}\right)+\mathbf{b}^T\boldsymbol{\nu}.$$

We can define $\boldsymbol{\lambda} \in \mathbb{R}^{2m_{\mathcal{E}}}$, $\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_n] \in \mathbb{R}^{2m_{\mathcal{E}} \times n\nu}$ and $\mathbf{b} \in \mathbb{R}^{2m_{\mathcal{E}}}$, with the following entries for each edge $e_l = (i, j) \in \mathcal{E}$:

- $\mathbf{c}_i(l) = \mathbf{A}_{i|j}$ and $\mathbf{c}_j(l + m_{\mathcal{E}}) = \mathbf{A}_{j|i}$,
- $\boldsymbol{\lambda}(l) = \boldsymbol{\lambda}_{i|j}$ and $\boldsymbol{\lambda}(l + m_{\mathcal{E}}) = \boldsymbol{\lambda}_{j|i}$
- $\mathbf{b}(l) = \mathbf{b}(l + m_{\mathcal{E}}) = \frac{1}{2}\mathbf{b}_{i,j}$.

Using the definitions above and a symmetric permutation matrix, $\mathbf{P} \in \mathbb{R}^{2m_{\mathcal{E}} \times 2m_{\mathcal{E}}}$, that permutes the top $m_{\mathcal{E}}$ rows with the bottom $m_{\mathcal{E}}$ rows, we can reformulate the Lagrangian dual problem as

$$\begin{aligned} \min_{\boldsymbol{\lambda}} \quad & f^*(-\mathbf{C}^T\boldsymbol{\lambda}) + \mathbf{d}^T\boldsymbol{\lambda} \\ \text{s.t.} \quad & (\mathbf{I} - \mathbf{P})\boldsymbol{\lambda} = \mathbf{0}. \end{aligned} \tag{3.2}$$

This is called the extended dual problem, see [5]. The constraint can be expressed by using the indicator function of $\ker(\mathbf{I} - \mathbf{P})$, denoted as $\iota_{\ker(\mathbf{I}-\mathbf{P})}$. A minimiser, say $\boldsymbol{\lambda}^*$, to the problem stated in (3.2) is found when

$$\mathbf{0} \in \mathbf{C}\partial f^*(\mathbf{C}^T\boldsymbol{\lambda}^*) - \mathbf{d} + \partial\iota_{\ker(\mathbf{I}-\mathbf{P})}(\boldsymbol{\lambda}^*). \tag{3.3}$$

As shown in [5], by rephrasing (3.3) as a fixed point condition using Peaceman-Rachford splitting (see Def. A.20), the PDMM operator is defined as

$$\mathbf{T}_{\mathrm{P},\rho} = \mathbf{C}_{\rho\mathbf{T}_2} \circ \mathbf{C}_{\rho\mathbf{T}_1}, \tag{3.4}$$

where $\mathbf{C}_{\rho\mathbf{T}}$ denotes the Cayley operator of $\mathbf{T}$ and $\mathbf{T}_1 = \mathbf{C}\partial f^*(\mathbf{C}^T\boldsymbol{\lambda}) - \mathbf{d}$ and $\mathbf{T}_2 = \partial\iota_{ker(\mathbf{I}-\mathbf{P})}$. Both $\mathbf{T}_1$ and $\mathbf{T}_2$ are maximal monotone operators, so their resolvent operators are both firmly nonexpansive. It follows that the Cayley operators are both nonexpansive, which makes $\mathbf{T}_{\mathrm{P},\rho}$ nonexpansive.

To guarantee convergence for arbitrary CCP cost functions, we can apply operator averaging. This results in the θ-averaged PDMM operator, which is defined as

$$\mathbf{T}_{\theta\mathrm{P},\rho} = (1-\theta)\mathbf{I} + \theta\mathbf{C}_{\rho\mathbf{T}_2} \circ \mathbf{C}_{\rho\mathbf{T}_1} \tag{3.5}$$

with $\theta \in (0, 1)$. Standard PDMM is equivalent to (3.5) with $\theta = 1$. Note that standard PDMM does not follow the definition of an averaged operator and, therefore, is at best nonexpansive.

As derived in [5], the update equations for PDMM can be made explicit by evaluating the Cayley operators in (3.5). This leads to four update equations that can be used to implement PDMM:

$$\mathbf{x}^{(k+1)} = \arg\min_{\mathbf{x}} \left(f(\mathbf{x}) + \left\langle \mathbf{C}^T\mathbf{z}^{(k)}, \mathbf{x} \right\rangle + \frac{\rho}{2}||\mathbf{C}\mathbf{x} - \mathbf{d}||^2 \right), \tag{3.6}$$

$$\boldsymbol{\lambda}^{(k+1)} = \mathbf{z}^{(k)} + \rho\left(\mathbf{C}\mathbf{x}^{(k+1)} - \mathbf{d}\right), \tag{3.7}$$

$$\mathbf{y}^{(k+1)} = 2\boldsymbol{\lambda}^{(k+1)} - \mathbf{z}^{(k)}, \tag{3.8}$$

$$\mathbf{z}^{(k+1)} = (1-\theta)\mathbf{z}^{(k)} + \theta\mathbf{P}\mathbf{y}^{(k+1)}. \tag{3.9}$$

Throughout this report, we will refer to $\mathbf{x}$ as the *primal variable*, $\boldsymbol{\lambda}$ as the *dual variable* and $\mathbf{y}, \mathbf{z}$ as the *auxiliary variables*.

3.4.2. Synchronous PDMM

The update equations that were derived in the previous section are separable across the nodes. This makes it possible to derive individual update equations, that can be performed in a distributed setting. Furthermore, as mentioned in Section 3.3.2, we assume consensus constraints. These assumptions hold for all numerical examples that will be included in this report. Therefore, in the remainder of the report we will use the following properties:

- $\mathbf{d} = \mathbf{0}$,
- $\mathbf{A}_{i|j} = \mathbf{I}$ and $\mathbf{A}_{j|i} = -\mathbf{I}$, $\forall(i, j) \in \mathcal{E}$,
- $||\mathbf{C}\mathbf{x}||^2 = \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}||\mathbf{A}_{i|j}\mathbf{x}_i||^2 = \sum_{i\in\mathcal{V}} d_i||\mathbf{x}_i||^2$, where d_i denotes the degree of node i,
- $\mathbf{z}^T\mathbf{C}\mathbf{x} = \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\mathbf{z}_{i|j}^T\mathbf{A}_{i|j}\mathbf{x}_i$.

Algorithm 1 Synchronous θ-averaged PDMM (unicast).

```
 1: Initialise:  $\mathbf{z}^{(0)} \in \mathbb{R}^{2m\mathcal{E}}$                                    ▷ Initialisation
 2: for $k = 0, ...,$ do
 3:     for all $i \in \mathcal{V}$ do                                                  ▷ Primal update
 4:         $\mathbf{x}_i^{(k+1)} = \arg\min_{\mathbf{x}_i} \left[ f_i(\mathbf{x}_i) + \sum_{j \in \mathcal{N}_i} \left( (\mathbf{z}_{i|j}^{(k)})^T \mathbf{A}_{i|j}\mathbf{x}_i + \frac{\rho}{2}||\mathbf{A}_{i|j}\mathbf{x}_i||_2^2 \right) \right]$
 5:         for all $j \in \mathcal{N}_i$ do                                             ▷ Dual update
 6:             $\boldsymbol{\lambda}_{i|j}^{(k+1)} = \mathbf{z}_{i|j}^{(k)} + \rho \mathbf{A}_{i|j}\mathbf{x}_i^{(k+1)}$
 7:             $\mathbf{y}_{i|j}^{(k+1)} = 2\boldsymbol{\lambda}_{i|j}^{(k+1)} - \mathbf{z}_{i|j}^{(k)}$
 8:         end for
 9:     end for

10:     for all $i \in \mathcal{V}, j \in \mathcal{N}_i$ do                    ▷ Transmit updated variables (unicast)
11:         Node$_j$ $\leftarrow$ Node$_i$($\mathbf{y}_{i|j}^{(k+1)}$)
12:     end for

13:     for all $i \in \mathcal{V}, j \in \mathcal{N}_i$ do                                  ▷ Auxiliary update
14:         $\mathbf{z}_{j|i}^{(k+1)} = (1-\theta)\mathbf{z}_{j|i}^{(k)} + \theta \mathbf{y}_{i|j}^{(k+1)}$
15:     end for
16: end for
```

With these properties it is possible to write out individual update equations that can be used for distributed implementations of PDMM. Algorithm 1 incorporates these individual updating equations into the pseudocode for synchronous PDMM. In the synchronous implementation of PDMM, at every iteration, all nodes perform a primal update followed by a dual and an auxiliary update. After this, the updated auxiliary variables are transmitted to the neighbours, which can then perform their update of $\mathbf{z}$. In [5], the convergence of synchronous standard PDMM is proved for strongly convex, differential cost functions and the convergence of synchronous θ-averaged PDMM is proved for arbitrary CCP cost functions.

3.4.3. PDMM variations

There are a number of different implementations of PDMM that have been introduced throughout the years. In this section we introduce the main aspects in which these variations can differ. The main update equations of all of PDMM variations are the same as the update equations derived in this section. The differences in these implementations arise from the distribution of the different parts of the algorithm and at which point in the updating equations a message is transmitted to neighbouring nodes. So far no in depth review of these PDMM variations has been performed and for some of these implementations the convergence has not yet been proved. In Chapter 5 we will analyse a variety of PDMM implementations and where possible link these algorithms to the convergence proof of stochastic PDMM, that is given in Chapter 4.

Synchronous and asynchronous algorithms

A distinction can be made between *synchronous* and *asynchronous* implementations of distributed algorithms. Synchronous implementations require a global clock that all nodes in the network are synced to. At each time slot of this global clock every node contacts its neighbours. The fact that clock synchronisation is needed adds a lot of communication overhead. Furthermore, for correct synchronisation the execution times of the computations and communication need to be known a priori and the duration of an iteration is governed by the slowest device in the network. In large scale distributed networks this is often hard to determine and network properties are often dynamic.

Asynchronous algorithms, on the other hand, do not require synchronisation with a global clock. This makes them a lot more suitable for many real world distributed applications. An asynchronous clock model is based on local clocks at each node. According to these local clocks a random node (or a random subset of nodes) is activated and contacts its neighbours according to the algorithm in question. Practical distributed networks are often heterogeneous and thus consist of different types of devices. In situations with heterogeneous networks asynchronous algorithms are highly desirable, because these algorithms do not require extra communication for synchronisation and the iteration speed is not limited by the slowest device in the network.

Unicast and broadcast
Another aspect in which distributed algorithms can vary is the way in which data is transmitted from node to node. In some algorithms a specific message is sent to each neighbouring node individually at every iteration, this is called *unicast*. In other cases the broadcast nature of wireless communications can be exploited. In such cases *broadcast* algorithms can be used, which only require the transmission of a single universal message to all neighbouring nodes at every iteration. Because less transmissions are required, broadcast algorithms are favourable in applications where communication costs must be kept low.

3.5. Subspace based privacy preservation

In this section we will explain the working of subspace based privacy preservation. First we will define the adversary model under which the method works, next we will define the privacy metric and the concept of subspace perturbation. Finally, we will explain the working of privacy preserving PDMM.

The main idea behind subspace based privacy preservation is to use random initialisation of the dual or auxiliary variable in a particular subspace as privacy preserving noise. This subspace does not affect the primal variable. The method was first proposed in [6] applied to broadcast synchronous PDMM. It is also compared to other types of privacy preservation methods in [17] and more recently a communication efficient synchronous version is presented in [7], which uses combines differential quantisation with subspace based privacy preservation. In this section we will summarise the main results of these papers, some of which will be used in the analysis of subspace based privacy preservation with stochastic PDMM (see Chapter 6). For simplicity we will assume that at each node the local variable and constraint variables are one dimensional. This means that $n_i = 1\ \forall i \in \mathcal{V}$ and $m_{i|j} = 1$ $\forall (i,j) \in \mathcal{E}$, so $n_{\mathcal{V}} = n$ and $m_{\mathcal{E}} = m$. The results can be straightforwardly generalised to higher dimensional variables.

3.5.1. Adversary model

Subspace based privacy preservation works in the presence of two types of adversaries. We assume that the two types of adversaries are both present and can cooperate by sharing their gathered information. With this information the adversaries attempt to infer the private data of honest nodes in the network.

The first adversary type consists of corrupted nodes in the network that collude and share information together. These corrupted nodes, known as passive adversaries, are assumed to follow the algorithm instructions, so they do not actively sabotage the optimisation. We will use $\mathcal{N}_h$ and $\mathcal{N}_c$, to denote honest and corrupt nodes respectively. The honest neighbourhood of node i is defined as $\mathcal{N}_{i,h} = \mathcal{N}_i \cap \mathcal{N}_h$ and the set of corrupt edges is defined as $\mathcal{E}_c = \{(i,j) : (i,j) \in \mathcal{E}, (i,j) \notin \mathcal{N}_h \times \mathcal{N}_h\}$.

The other type of adversary, known as an eavesdropping adversary, can eavesdrop all messages transmitted along edges in the network. With securely encrypted communication channels, this type of adversary has no effect, but in the case of large scale distributed optimisation such encryption is not feasible due to the large number of iterations needed to reach convergence.

3.5.2. Privacy metric

For this method privacy is defined using the mutual information between the private data and the data that is observed by the adversaries. Let S_i be the private continuous random variable, that relates to node i and have variance σ_i^2. Let $Z_i = S_i + N_i$ be what the adversaries can observe related to S_i, where N_i is a random variable that is statistically independent of S_i. N_i acts as noise masking the private data S_i. To achieve a certain privacy level δ, the mutual information between S_i and Z_i should be below this value. Using $I(X;Y)$ to denote the mutual information between random variables X and Y, we can define δ-level privacy as $I(S_i;Z_i) \leq \delta$. As derived in [6], under the assumption that the random variables have Gaussian distributions, the condition for reaching δ-level privacy at node i is

$$\sigma_{N_i}^2 \geq \frac{\sigma_{S_i}^2}{2^{2\delta} - 1}.$$

This condition shows that an arbitrarily high level of privacy (arbitrarily small mutual information) can be achieved by making the noise variance arbitrarily large.

3.5.3. Subspace perturbation

Subspace based privacy exploits the fact that there is a subspace that does not affect the primal variable. As discussed in [5], every two synchronous PDMM updates the auxiliary variables are only affected in the subspace

$$\Psi = \text{ran}(\mathbf{C}) + \text{ran}(\mathbf{PC}).$$

The orthogonal subspace of Ψ is defined as

$$\Psi^{\perp} = \ker(\mathbf{C}^T) \cap \ker((\mathbf{PC})^T).$$

The auxiliary variable can be split up into $\mathbf{z}_{\Psi}^{(k)} = \mathbf{\Pi}_{\Psi}\mathbf{z}^{(k)}$ and $\mathbf{z}_{\Psi^{\perp}}^{(k)} = (\mathbf{I} - \mathbf{\Pi}_{\Psi})\mathbf{z}^{(k)}$, where $\mathbf{\Pi}_{\Psi}$ is used to denote the orthogonal projection onto the subspace Ψ[2]. By definition of the subspace $\Psi^{\perp}$, we can see that $\mathbf{C}^T\mathbf{z}_{\Psi^{\perp}}^{(k)} = \mathbf{0}$ and thus the components of $\mathbf{z}^{(k)}$ that are in the subspace $\Psi^{\perp}$ do not play a role in updating the primal variable (see (3.6)). Because of this property, arbitrarily large noise in the subspace $\Psi^{\perp}$ will not affect primary convergence, making it possible to use noise in this subspace to mask private data. The matrix $[\mathbf{C}\ \mathbf{PC}] \in \mathbb{R}^{2m\times 2n}$ can be seen as an incidence matrix of a bipartite graph with $2n$ nodes and $2m$ edges. As mentioned in [6], $\text{dim}(\Psi) = 2n - 1$ as long as $m \geq n$, so that $\Psi^{\perp}$ is nonempty.

3.5.4. Privacy preserving synchronous PDMM

In this section we will analyse subspace based privacy preserving synchronous PDMM. The general idea is to randomly initialise $\mathbf{z}$ (or $\boldsymbol{\lambda}$), so that $\mathbf{z}_{\Psi^{\perp}}$ (or $\boldsymbol{\lambda}_{\Psi^{\perp}}$) is also nonzero. The two assumptions that are required to preserve privacy are:

1. The communication channels in the network are securely encrypted when transmitting the initialised values $\mathbf{z}^{(0)}$ (or $\boldsymbol{\lambda}^{(0)}$);
2. Each honest node has at least one honest neighbour, so for an honest node i, $\mathcal{N}_{i,h} \neq \varnothing$.

Under these assumptions the private data is obscured by the noise of $\mathbf{z}_{\Psi^{\perp}}$ (or $\boldsymbol{\lambda}_{\Psi^{\perp}}$) and thus the adversaries never have full knowledge of the private data.

To analyse what information the adversaries can deduce, we can express the primal update equation (3.6) as

$$0 \in \partial f_i(x_i^{(k+1)}) + \sum_{j\in\mathcal{N}_i} \mathbf{A}_{i|j} z^{(k)} + \rho d_i x_i^{(k+1)}. \tag{3.10}$$

The private data of a node is present in the local objective function f_i, so the only term in (3.10) that contains private data is $\partial f_i(x_i^{(k+1)})$. Through knowledge of the PDMM algorithm, adversaries can use the inclusion stated above to deduce information about the term $\partial f_i(x_i^{(k+1)})$. The data that is known to the adversaries depends on the implementation of PDMM. So far, two synchronous implementations have been analysed. In [6] broadcast λ-update PDMM is analysed and in [7] adaptive quantised differential PDMM is analysed. We will highlight some important results from these two papers.

In the case that broadcast synchronous λ-update PDMM is used, it is shown in [6] that, after deducing the known terms from (3.10), the adversaries can observe

$$\partial f_i\left(x_i^{(k+1)}\right) + \sum_{j\in\mathcal{N}_{i,h}} \mathbf{A}_{i|j}\lambda_{j|i}^{(k)}.$$

As the iterations proceed, $\boldsymbol{\lambda}_{\Psi}^{(k)}$ converges and thus its variance decreases to zero, making it unusable for privacy preservation. The only part that can continuously preserve privacy by having a nonzero variance is $\boldsymbol{\lambda}_{\Psi^{\perp}}^{(k)}$. In the case of synchronous PDMM $\boldsymbol{\lambda}_{\Psi^{\perp}}^{(k)} = \mathbf{P}^k(\mathbf{I} - \mathbf{\Pi}_{\Psi})\boldsymbol{\lambda}^{(0)}$. So under the assumption that $\mathcal{N}_{i,h} \neq \varnothing$, a sufficient condition to achieve a mutual information that is smaller than δ is

$$\exists j \in \mathcal{N}_{i,h}\ :\ \text{var}\left(\left[\mathbf{\Pi}_{\Psi^{\perp}}\boldsymbol{\lambda}^{(0)}\right]_{j|i}\right) \geq \frac{\text{var}\left(\partial f_i(x_i^{(k+1)})\right)}{2^{2\delta}-1},$$

[2]The same split can be applied to $\boldsymbol{\lambda}$.

assuming that the random variables have Gaussian distributions. In [7], a similar analysis can be found for adaptive quantised differential synchronous PDMM, where the quantised difference of the auxiliary variable $\mathbf{z}$ is sent to neighbouring nodes, see [25]. It is also shown that $\mathbf{z}_{\Psi^\perp}^{(k)} = 1/2(\mathbf{I}+\mathbf{P}) + 1/2(2\theta-1)^k(\mathbf{I}-\mathbf{P})\mathbf{z}_{\Psi^\perp}^{(0)}$, which is used to derive a closed form expression for $\mathbb{E}\left[\mathbf{z}_{\Psi^\perp}^{(k)}\mathbf{z}_{\Psi^\perp}^{(k),T}\right]$. If the auxiliary variable is randomly initialised in such a way that $\mathbb{E}\left[\mathbf{z}^{(0)}\mathbf{z}^{(0),T}\right] = \sigma^2\mathbf{I}$, the covariance matrix of $\mathbf{z}_{\Psi^\perp}^{(k)}$ can be expressed as

$$\mathbb{E}\left[\mathbf{z}_{\Psi^\perp}^{(k)}\mathbf{z}_{\Psi^\perp}^{(k),T}\right] = \mathbf{\Pi}_{\Psi^\perp}\left(\frac{\sigma^2}{2}\left((\mathbf{I}+\mathbf{P}) + |2\theta-1|^{2k}(\mathbf{I}-\mathbf{P})\right)\right).$$

This expression indicates that the variance of $\mathbf{z}_{\Psi^\perp}$ has a nonzero nondecreasing component that depends on the initialisation variance.

4 Stochastic PDMM Convergence

In this chapter, we first define a general stochastic version of PDMM in Section 4.1. After this, we formulate the convergence proof for stochastic θ-averaged PDMM and stochastic standard PDMM in Sections 4.2 and 4.3 respectively. The main approach of the proof in this chapter was originally presented in [21], but we include specific results for non-averaged PDMM. To the best of our knowledge, this is the first convergence proof for stochastic Banach-Picard type iterations with a nonexpansive operator, without the necessity of operator averaging. Furthermore, an earlier attempt at this proof was made in [26] (unpublished). In this chapter we follow the main steps of this proof and add steps to complete it. The convergence results of this chapter will be used in Chapter 5 to prove the convergence of asynchronous PDMM and PDMM with transmission loss.

We would like to note that some of the lemmas and theorems from [18] that are used in the proof mention *weak* convergence. In the case of PDMM $\mathbf{z}^{(k)} \in \mathbb{R}^{2m_\mathcal{E}}$ is finite dimensional and thus by [18, Lemma 2.51] weak convergence implies strong convergence. Thus, throughout this chapter we only consider strong convergence.

4.1. Stochastic PDMM definition

First, we will define a stochastic Banach-Picard iteration, which forms the update equation for stochastic PDMM when applied to the PDMM operator $\mathbf{T}_{\mathrm{P},\rho}$. Once this is defined, we state two assumptions, which are then used to derive an expression that forms the basis of the proofs in Section 4.2 and Section 4.3.

Stochastic updates can be defined by assuming that each auxiliary variable $\mathbf{z}_{i|j}$ can be updated based on a Bernoulli random variable $U_{i|j} \in \{0,1\}$, with mean $\mu_{i|j} = \mathbb{P}\{U_{i|j} = 1\} = \mathbb{E}[U_{i|j}] \in (0,1)$. We define the random matrix $\mathbf{U}^{(k)} \in \mathbb{R}^{2m_\mathcal{E} \times 2m_\mathcal{E}}$ as a block diagonal matrix with entries

$$\mathbf{u}_{i|j} = U_{i|j}\mathbf{I}_{m_{i,j}} \quad \forall i \in \mathcal{V}, j \in \mathcal{N}_i,$$

ordered similarly as the entries of $\mathbf{z}$.

Definition 4.1. *For an operator* $\mathbf{T}$ *and a sequence of realisations of random updating matrices* $\left(\mathbf{U}^{(k)}\right) = \mathit{diag}(\mathbf{u}^{(k)})$*, we define the **stochastic Banach-Picard iteration** as:*

$$\mathbf{z}^{(k+1)} = \left(\mathbf{I} - \mathbf{U}^{(k+1)}\right)\mathbf{z}^{(k)} + \mathbf{U}^{(k+1)}\mathbf{T}\mathbf{z}^{(k)}. \tag{4.1}$$

To summarise, $\mathbf{U}^{(k)}$ is a random diagonal updating matrix that has zeros and ones on its diagonal. This matrix determines which entries of $\mathbf{z}$ are updated in a particular iteration:

- $U_{i|j} = 1$ corresponds to updating entry $\mathbf{z}_{i|j}$ using a Banach-Picard iteration of operator $\mathbf{T}$,
- $U_{i|j} = 0$ corresponds to not updating entry $\mathbf{z}_{i|j}$.

Consider a stochastic Banach-Picard sequence of operator $\mathbf{T}$, see Definition 4.1, with fix$(\mathbf{T}) \neq \varnothing$, where the following assumptions hold true:

Assumption 4.1. $\left(\mathbf{U}^{(k)}\right)_{k\in\mathbb{N}}$ *is a random i.i.d. sequence*[1].

Assumption 4.2. $\mu_{i|j} > 0, \forall i \in \mathcal{V}, j \in \mathcal{N}_i$, *so* $\mathbb{E}[\mathbf{U}] = \bar{\mathbf{U}} \succ 0.$

Now, take $\mathbf{z}^* \in$ fix$(\mathbf{T})$ and a known initialisation $\mathbf{z}^{(0)}$.[2] Using Assumptions 4.1 and 4.2 and the conditional expectation with respect to the sigma-algebra $\mathcal{F}^{(k)} = \sigma\left(\mathbf{z}^{(0)}, ..., \mathbf{z}^{(k)}\right)$, we can derive

$$\begin{aligned}
\mathbb{E}\left[||\mathbf{z}^{(k+1)} - \mathbf{z}^*||^2_{\bar{\mathbf{U}}^{-1}}|\mathcal{F}^{(k)}\right] &= \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\frac{1}{\mu_{i|j}}\mathbb{E}\left[\left|\left|\mathbf{z}^{(k)}_{i|j} - U^{(k+1)}_{i|j}\mathbf{z}^{(k)}_{i|j} + U^{(k+1)}_{i|j}\left[\mathbf{T}\mathbf{z}^{(k)}\right]_{i|j} - \mathbf{z}^*_{i|j}\right|\right|^2|\mathcal{F}^{(k)}\right]\\
&= \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\frac{1}{\mu_{i|j}}\mathbb{E}\Bigg[||\mathbf{z}^{(k)}_{i|j} - \mathbf{z}^*_{i|j}||^2 + \left(U^{(k+1)}_{i|j}\right)^2\left|\left|\left[\mathbf{T}\mathbf{z}^{(k)}\right]_{i|j} - \mathbf{z}^{(k)}_{i|j}\right|\right|^2\\
&\qquad + 2U^{(k+1)}_{i|j}\left\langle\mathbf{z}^{(k)}_{i|j} - \mathbf{z}^*_{i|j}, \left[\mathbf{T}\mathbf{z}^{(k)}\right]_{i|j} - \mathbf{z}^{(k)}_{i|j}\right\rangle|\mathcal{F}^{(k)}\Bigg]\\
&= \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\frac{1}{\mu_{i|j}}\Bigg(||\mathbf{z}^{(k)}_{i|j} - \mathbf{z}^*_{i|j}||^2\\
&\qquad + \mu_{i|j}\left[\left[\mathbf{T}\mathbf{z}^{(k)}\right]^2_{i|j} - \left(\mathbf{z}^{(k)}_{i|j}\right)^2 - 2\mathbf{z}^*_{i|j}\left[\mathbf{T}\mathbf{z}^{(k)}\right]_{i|j} + 2\mathbf{z}^*_{i|j}\mathbf{z}^{(k)}_{i|j}\right]\Bigg)\\
&= \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\frac{1}{\mu_{i|j}}\left(||\mathbf{z}^{(k)}_{i|j} - \mathbf{z}^*_{i|j}||^2 + \mu_{i|j}\left[\left|\left|\left[\mathbf{T}\mathbf{z}^{(k)}\right]_{i|j} - \mathbf{z}^*_{i|j}\right|\right|^2 - ||\mathbf{z}^{(k)}_{i|j} - \mathbf{z}^*_{i|j}||^2\right]\right)\\
&= \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\frac{1-\mu_{i|j}}{\mu_{i|j}}||\mathbf{z}^{(k)}_{i|j} - \mathbf{z}^*_{i|j}||^2 + \left|\left|\left[\mathbf{T}\mathbf{z}^{(k)}\right]_{i|j} - \mathbf{z}^*_{i|j}\right|\right|^2\\
&= ||\mathbf{z}^{(k)} - \mathbf{z}^*||^2_{\bar{\mathbf{U}}^{-1}} - ||\mathbf{z}^{(k)} - \mathbf{z}^*||^2 + ||\mathbf{T}\mathbf{z}^{(k)} - \mathbf{z}^*||^2,
\end{aligned} \tag{4.2}$$

where the third line results from the conditioning on $\mathcal{F}^{(k)}$ and the fact that $U^2_{i|j} = U_{i|j}, \forall i \in \mathcal{V}, j \in \mathcal{N}_i$. It is worth noting that the outcome of (4.2) is still a random variable and thus the related expressions include an implicit "almost surely" qualifier. In Sections 4.2 and 4.3 we will use (4.2) to show that stochastic averaged PDMM and stochastic standard PDMM converge almost surely.

4.2. Stochastic averaged PDMM convergence proof

In this section, we provide a convergence proof for stochastic θ-averaged PDMM. This is a general framework that can be used to model PDMM variations such as asynchronous PDMM and PDMM with transmission losses. First, almost sure auxiliary convergence is proved in Theorem 4.1, for arbitrary CCP cost functions. This proof is based on Lemmas 4.1-4.3 and Theorem 3.2. Next, it is shown that auxiliary convergence leads to a primal optimal solution in Theorem 4.2.

Assume a sequence of stochastic Banach-Picard iterations of the θ-averaged PDMM operator $\mathbf{T}_{\theta \mathrm{P},\rho}$, as stated in (3.5). For arbitrary CCP cost functions $\mathbf{T}_{\mathrm{P},\rho}$ is nonexpansive and thus $\mathbf{T}_{\theta\mathrm{P},\rho}$ is θ-averaged. Consequently, we can use [18, Proposition 4.35] and (4.2) to show that

$$\mathbb{E}\left[||\mathbf{z}^{k+1} - \mathbf{z}^*||^2_{\bar{\mathbf{U}}^{-1}}|\mathcal{F}^{(k)}\right] \leq ||\mathbf{z}^{(k)} - \mathbf{z}^*||^2_{\bar{\mathbf{U}}^{-1}} - \frac{1-\theta}{\theta}||(\mathbf{I} - \mathbf{T}_{\mathrm{P},\rho})\mathbf{z}^{(k)}||^2. \tag{4.3}$$

This inequality will be used for both Lemma 4.1 and 4.3. Firstly, (4.3) shows that the sequence of random variables $||\mathbf{z}^{(k)} - \mathbf{z}^*||^2_{\bar{\mathbf{U}}^{-1}}$ forms a nonnegative supermartingale, see Definition 3.6. Because the square root function is concave (on $\mathbb{R}^+$), Jensen's inequality [19, Thm. 23.9] can be applied to find

$$\sqrt{\mathbb{E}\left[||\mathbf{z}^{k+1} - \mathbf{z}^*||^2_{\bar{\mathbf{U}}^{-1}}|\mathcal{F}^{(k)}\right]} \geq \mathbb{E}\left[||\mathbf{z}^{k+1} - \mathbf{z}^*||_{\bar{\mathbf{U}}^{-1}}|\mathcal{F}^{(k)}\right]. \tag{4.4}$$

[1]Note that no assumption is made on the dependence between the entries $U_{i|j}$ of $\mathbf{U}^{(k)}$.

[2]Note that $\mathbf{z}^{(0)}$ and $\mathbf{z}^*$ are deterministic and all other $\mathbf{z}^{(k)}$'s are random vectors.

From (4.3) we know that $\mathbb{E}\left[||\mathbf{z}^{k+1}-\mathbf{z}^*||^2_{\bar{\mathbf{U}}^{-1}}|\mathcal{F}^{(k)}\right] \leq ||\mathbf{z}^{(k)}-\mathbf{z}^*||^2_{\bar{\mathbf{U}}^{-1}}$. Using the fact that the square root function is nondecreasing on $\mathbb{R}^+$ and using (4.4) we find

$$\mathbb{E}\left[||\mathbf{z}^{k+1}-\mathbf{z}^*||_{\bar{\mathbf{U}}^{-1}}|\mathcal{F}^{(k)}\right] \leq \sqrt{\mathbb{E}\left[||\mathbf{z}^{k+1}-\mathbf{z}^*||^2_{\bar{\mathbf{U}}^{-1}}|\mathcal{F}^{(k)}\right]} \leq ||\mathbf{z}^{(k)}-\mathbf{z}^*||_{\bar{\mathbf{U}}^{-1}}. \tag{4.5}$$

This shows that $||\mathbf{z}^{(k)}-\mathbf{z}^*||_{\bar{\mathbf{U}}^{-1}}$ is also a nonnegative supermartingale. We can use this property to prove the following lemma.

Lemma 4.1. *There is a probability one set on which $||\mathbf{z}^{(k)}-\mathbf{z}^*||_{\bar{\mathbf{U}}^{-1}}$ converges for every $\mathbf{z}^* \in$ fix($\mathbf{T}_{P,\rho}$).*

Proof. As shown in (4.5), for any $\mathbf{z}^* \in$ fix($\mathbf{T}_{P,\rho}$), $||\mathbf{z}^{(k)}-\mathbf{z}^*||_{\bar{\mathbf{U}}^{-1}}$ is a nonnegative supermartingale. Thus, Theorem 3.3 can be applied to show that it converges almost surely to a random variable $\mathbf{Z} \in [0,\infty)$. □

For the next part of the proof, we will use (4.3) to formulate a useful inequality involving $\mathbb{E}\left[||\mathbf{z}^{k+1}-\mathbf{z}^*||^2_{\bar{\mathbf{U}}^{-1}}\right]$. This inequality is derived in Lemma 4.2 below.

Lemma 4.2. *Assume that (4.3) holds, then*

$$\mathbb{E}\left[||\mathbf{z}^{k+1}-\mathbf{z}^*||^2_{\bar{\mathbf{U}}^{-1}}\right] \leq ||\mathbf{z}^{(0)}-\mathbf{z}^*||^2_{\bar{\mathbf{U}}^{-1}} - \frac{1-\theta}{\theta}\sum_{t=0}^{k}\mathbb{E}\left[||(\mathbf{I}-\mathbf{T}_{P,\rho})\mathbf{z}^{(t)}||^2\right].$$

Proof. Take the expected value on both sides of (4.3) to get

$$\mathbb{E}\left[||\mathbf{z}^{k+1}-\mathbf{z}^*||^2_{\bar{\mathbf{U}}^{-1}}\right] \leq \mathbb{E}\left[||\mathbf{z}^{(k)}-\mathbf{z}^*||^2_{\bar{\mathbf{U}}^{-1}}\right] - \frac{1-\theta}{\theta}\mathbb{E}\left[||(\mathbf{I}-\mathbf{T}_{P,\rho})\mathbf{z}^{(k)}||^2\right]. \tag{4.6}$$

First, we show that the for $k=0$

$$\begin{aligned}\mathbb{E}\left[||\mathbf{z}^{1}-\mathbf{z}^*||^2_{\bar{\mathbf{U}}^{-1}}\right] &\leq \mathbb{E}\left[||\mathbf{z}^{(0)}-\mathbf{z}^*||^2_{\bar{\mathbf{U}}^{-1}}\right] - \frac{1-\theta}{\theta}\mathbb{E}\left[||(\mathbf{I}-\mathbf{T}_{P,\rho})\mathbf{z}^{(0)}||^2\right] \\ &= ||\mathbf{z}^{(0)}-\mathbf{z}^*||^2_{\bar{\mathbf{U}}^{-1}} - \sum_{t=0}^{1}\frac{1-\theta}{\theta}\mathbb{E}\left[||(\mathbf{I}-\mathbf{T}_{P,\rho})\mathbf{z}^{(t)}||^2\right],\end{aligned}$$

where we use the fact that $\mathbf{z}^{(0)}$ and $\mathbf{z}^*$ are deterministic. This shows that for the base case the expression stated in the lemma holds. Next, assume that

$$\mathbb{E}\left[||\mathbf{z}^{k+1}-\mathbf{z}^*||^2_{\bar{\mathbf{U}}^{-1}}\right] \leq ||\mathbf{z}^{(0)}-\mathbf{z}^*||^2_{\bar{\mathbf{U}}^{-1}} - \frac{1-\theta}{\theta}\sum_{t=0}^{k}\mathbb{E}\left[||(\mathbf{I}-\mathbf{T}_{P,\rho})\mathbf{z}^{(t)}||^2\right],$$

then by evaluating (4.6) for iteration $k+2$ we get

$$\begin{aligned}\mathbb{E}\left[||\mathbf{z}^{k+2}-\mathbf{z}^*||^2_{\bar{\mathbf{U}}^{-1}}\right] &\leq \mathbb{E}\left[||\mathbf{z}^{k+1}-\mathbf{z}^*||^2_{\bar{\mathbf{U}}^{-1}}\right] - \frac{1-\theta}{\theta}\mathbb{E}\left[||(\mathbf{I}-\mathbf{T}_{P,\rho})\mathbf{z}^{(k)}||^2\right] \\ &= ||\mathbf{z}^{(0)}-\mathbf{z}^*||^2_{\bar{\mathbf{U}}^{-1}} - \frac{1-\theta}{\theta}\sum_{t=0}^{k+1}\mathbb{E}\left[||(\mathbf{I}-\mathbf{T}_{P,\rho})\mathbf{z}^{(t)}||^2\right],\end{aligned}$$

which shows that the induction step is also valid. Now, by mathematical induction, we arrive at the desired result. □

Because $0 \leq \mathbb{E}\left[||\mathbf{z}^{k+1}-\mathbf{z}^*||^2_{\bar{\mathbf{U}}^{-1}}\right]$, we can reformulate the expression in Lemma 4.6 as

$$\sum_{t=0}^{k}\mathbb{E}\left[||(\mathbf{I}-\mathbf{T}_{P,\rho})\mathbf{z}^{(t)}||^2\right] \leq \frac{\theta}{1-\theta}||\mathbf{z}^{(0)}-\mathbf{z}^*||^2_{\bar{\mathbf{U}}^{-1}}.$$

By taking the limit of $t \to \infty$ we get

$$\sum_{t=0}^{\infty}\mathbb{E}\left[||(\mathbf{I}-\mathbf{T}_{P,\rho})\mathbf{z}^{(t)}||^2\right] \leq \frac{\theta}{1-\theta}||\mathbf{z}^{(0)}-\mathbf{z}^*||^2_{\bar{\mathbf{U}}^{-1}} < \infty, \tag{4.7}$$

where the last inequality arises because $||\mathbf{z}^{(0)}-\mathbf{z}^*||^2_{\bar{\mathbf{U}}^{-1}}$ is known and finite. We will use the result from (4.7) to prove another lemma.

Lemma 4.3. *There is a probability one set on which all cluster points of $\left(\mathbf{z}^{(k)}\right)_{k\in\mathbb{N}}$ are in fix $\left(\mathbf{T}_{\mathsf{P},\rho}\right)$*

Proof. Using Markov's inequality [19, Cor. 5.1] and (4.7), we can write

$$\sum_{k=0}^{\infty}\mathbb{P}\left\{||(\mathbf{I}-\mathbf{T}_{\mathsf{P},\rho})\mathbf{z}^{(k)}||^2\geq\epsilon\right\}\leq\frac{1}{\epsilon}\sum_{k=0}^{\infty}\mathbb{E}\left[||(\mathbf{I}-\mathbf{T}_{\mathsf{P},\rho})\mathbf{z}^{(k)}||^2\right]<\infty\quad\forall\epsilon>0.$$

Now we can apply Borel Cantelli's lemma [19, Thm. 10.5] and Definition 3.7 to conclude that

$$\mathbb{P}\left\{||(\mathbf{I}-\mathbf{T}_{\mathsf{P},\rho})\mathbf{z}^{(k)}||^2\geq\epsilon\quad\text{i.o.}\right\}=0.$$

That is, there is a set of events with probability one, such that

$$\forall\epsilon\;\exists k_0\;:\;||(\mathbf{I}-\mathbf{T}_{\mathsf{P},\rho})\mathbf{z}^{(k)}||^2<\epsilon\quad\forall k\geq k_0.$$

Hence, $(\mathbf{I}-\mathbf{T}_{\mathsf{P},\rho})\mathbf{z}^{(k)}\to 0$ almost surely.

Consider an event in the set in which $(\mathbf{I}-\mathbf{T}_{\mathsf{P},\rho})\mathbf{z}^{(k)}\to 0$ and take any sequential cluster point of $(\mathbf{z}^{(k)})_{k\in\mathbb{N}}$, which we denote as $\mathbf{z}^*$. Then, by definition of a cluster point, there exists a subsequence $f(k)$ ($f:\mathbb{N}\to\mathbb{N}$) such that $\mathbf{z}^{(f(k))}\to\mathbf{z}^*$. Because $\mathbf{T}_{\mathsf{P},\rho}$ is nonexpansive, it is Lipschitz continuous and thus continuous so that

$$0=\lim_{k\to\infty}\left(\mathbf{z}^{(f(k))}-\mathbf{T}_{\mathsf{P},\rho}\left(\mathbf{z}^{(f(k))}\right)\right)=\lim_{k\to\infty}\mathbf{z}^{(f(k))}-\lim_{k\to\infty}\mathbf{T}_{\mathsf{P},\rho}\left(\mathbf{z}^{(f(k))}\right)=\mathbf{z}^*-\mathbf{T}_{\mathsf{P},\rho}(\mathbf{z}^*),$$

Hence, $\mathbf{z}^*\in\text{fix}(\mathbf{T}_{\mathsf{P},\rho})$. □

Next, we combine Lemma 4.1 and Lemma 4.3 to prove almost sure auxiliary convergence in the following theorem.

Theorem 4.1. *The sequence $\mathbf{z}^{(k+1)}=\left(\mathbf{I}-\mathbf{U}^{(k+1)}\right)\mathbf{z}^{(k)}+\mathbf{U}^{(k+1)}\mathbf{T}_{\theta P,\rho}\left(\mathbf{z}^{(k)}\right)$ converges almost surely to a random variable $\mathbf{z}^*$ that is supported by fix($\mathbf{T}_{\mathsf{P},\rho}$). This will be referred to as auxiliary convergence.*

Proof. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be the sets of events in which Lemma 4.1 and 4.3 hold, respectively. If we consider an event $\alpha\in\mathcal{A}_1\cap\mathcal{A}_2$ we can apply Theorem 3.2 to prove that $(\mathbf{z}^{(k)})_{n\in\mathbb{N}}$ converges to a point in fix($\mathbf{T}_{\mathsf{P},\rho}$). Because $\mathbb{P}\{\mathcal{A}_1\cap\mathcal{A}_2\}=1$, we arrive at the conclusion. □

Now that we have proved the auxiliary convergence of stochastic θ-averaged PDMM, all that remains is to show that auxiliary convergence also implies primal convergence. This is done in the theorem that follows.

Theorem 4.2. *Stochastic θ-averaged PDMM converges almost surely to a primal optimal point $\mathbf{x}^*$.*

Proof. Given any realisation of the sequence $\left(\mathbf{z}^{(k)}\right)_{k\in\mathbb{N}}$, we know from Theorem 4.1 that $\mathbf{z}^{(k)}$ converges almost surely to some $\mathbf{z}^*\in\text{fix}(\mathbf{T}_{\mathsf{P},\rho})$. As derived in [5], $\mathbf{z}^*$ can be used to calculate a primal optimal solution of (3.1) using $\mathbf{x}^*=\arg\min_{\mathbf{x}}\left(f(\mathbf{x})+\left\langle\mathbf{C}^T\mathbf{z}^*,\mathbf{x}\right\rangle+\frac{\rho}{2}||\mathbf{C}\mathbf{x}||^2\right)$. □

4.3. Stochastic standard PDMM convergence proof

Because the standard PDMM operator $\mathbf{T}_{\mathsf{P},\rho}$ is nonexpansive for CCP cost functions, we can use (4.2) to derive

$$\begin{aligned}\mathbb{E}\left[||\mathbf{z}^{(k+1)}-\mathbf{z}^*||^2_{\bar{\mathbf{U}}^{-1}}|\mathcal{F}^{(k)}\right]&=||\mathbf{z}^{(k)}-\mathbf{z}^*||^2_{\bar{\mathbf{U}}^{-1}}-||\mathbf{z}^{(k)}-\mathbf{z}^*||^2+||\mathbf{T}\mathbf{z}^{(k)}-\mathbf{z}^*||^2\\&=||\mathbf{z}^{(k)}-\mathbf{z}^*||^2_{\bar{\mathbf{U}}^{-1}}-||\mathbf{z}^{(k)}-\mathbf{z}^*||^2+||\mathbf{T}\mathbf{z}^{(k)}-\mathbf{T}\mathbf{z}^*||^2\\&\leq||\mathbf{z}^{(k)}-\mathbf{z}^*||^2_{\bar{\mathbf{U}}^{-1}}.\end{aligned}$$

From this inequality we can conclude that the sequence $(||\mathbf{z}^{(k)}-\mathbf{z}^*||^2_{\bar{\mathbf{U}}^{-1}})_{k\in\mathbb{N}}$ forms a nonnegative supermartingale and thus by Theorem 3.3 converges almost surely to a bounded random variable. This alone is not enough to prove convergence of stochastic standard PDMM. Contrary to θ-averaged PDMM, in this case (4.3) does not hold. Thus, to prove the convergence of stochastic standard PDMM, we use further assumptions on the objective f, which are stated in Lemma 4.4.

Since $\mathbf{x}^{(k+1)}$ minimises $f(\mathbf{x}) - \mathbf{z}^{(k)T}\mathbf{C}\mathbf{x} + \frac{c}{2}||\mathbf{C}\mathbf{x} - \mathbf{d}||^2$, we have that $\mathbf{0} \in \partial f(\mathbf{x}^{(k+1)}) - \mathbf{C}^T\mathbf{z}^{(k)} + \rho\mathbf{C}^T(\mathbf{C}\mathbf{x}^{(k+1)} - \mathbf{d})$. By using (3.7), this can be rewritten as

$$\mathbf{0} \in \partial f(\mathbf{x}^{(k+1)}) - \mathbf{C}^T\boldsymbol{\lambda}^{(k+1)}. \tag{4.8}$$

For the remaining part of this proof we will take $\mathbf{z}^* \in \text{fix}\,(\mathbf{T}_{\mathrm{P},\rho})$ and define

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \left(f(\mathbf{x}) + \langle \mathbf{C}^T\mathbf{z}^*, \mathbf{x}\rangle + \frac{\rho}{2}||\mathbf{C}\mathbf{x}||^2 \right), \tag{4.9}$$

$$\boldsymbol{\lambda}^* = \mathbf{J}_{\rho\mathbf{T}_1}(\mathbf{z}^*) = \mathbf{z}^* + \rho\mathbf{C}\mathbf{x}^*.$$

We would like to point out that $\mathbf{z}^*$, $\mathbf{z}^{(0)}$, $\mathbf{x}^*$ and $\boldsymbol{\lambda}^*$ are deterministic and all other $\mathbf{z}^{(k)}$'s, $\mathbf{x}^{(k)}$'s and $\boldsymbol{\lambda}^{(k)}$'s are random vectors.

Lemma 4.4. *If we assume:*

1. *f is differentiable,*
2. *f is β-strongly convex,*

the following inequality holds

$$\left\langle \boldsymbol{\lambda}^{(k)} - \boldsymbol{\lambda}^*, \mathbf{C}\mathbf{x}^{(k)} - \mathbf{C}\mathbf{x}^* \right\rangle = \left\langle \nabla f(\mathbf{x}^{(k)}) - \nabla f(\mathbf{x}^*), \mathbf{x}^{(k)} - \mathbf{x}^* \right\rangle \geq \beta||\mathbf{x}^{(k)} - \mathbf{x}^*||^2.$$

Proof. The equality holds because of the differentiability of f in combination with (4.8). The inequality results from the fact that f is β-strongly convex, which implies that ∇f is β-strongly monotone. □

By using (3.7)-(3.9) and the polarisation identity [18, Lemma 2.12], we have

$$\begin{aligned}
||\mathbf{T}_{\mathrm{P},\rho}(\mathbf{z}^{(k)}) - \mathbf{z}^*||^2 &= ||\mathbf{z}^{(k+1)} - \mathbf{z}^*||^2 \\
&= ||\mathbf{P}\left(2\boldsymbol{\lambda}^{(k+1)} - \mathbf{z}^{(k)} - (2\boldsymbol{\lambda}^* - \mathbf{z}^*)\right)||^2 \\
&= ||2\boldsymbol{\lambda}^{(k+1)} - \mathbf{z}^{(k)} - (2\boldsymbol{\lambda}^* - \mathbf{z}^*)||^2 \\
&= ||\mathbf{z}^{(k)} - \mathbf{z}^*||^2 - 4\left\langle \boldsymbol{\lambda}^{(k+1)} - \boldsymbol{\lambda}^*, \mathbf{z}^{(k)} - \boldsymbol{\lambda}^{(k+1)} - (\mathbf{z}^* - \boldsymbol{\lambda}^*)\right\rangle \\
&= ||\mathbf{z}^{(k)} - \mathbf{z}^*||^2 - 4\rho\left\langle \boldsymbol{\lambda}^{(k+1)} - \boldsymbol{\lambda}^*, \mathbf{C}\mathbf{x}^{(k+1)} - \mathbf{C}\mathbf{x}^*\right\rangle.
\end{aligned} \tag{4.10}$$

Next, utilising Lemma 4.4 and (4.10), the inequality in (4.2) can be reformulated as

$$\begin{aligned}
0 \leq \mathbb{E}\left[||\mathbf{z}^{(k+1)} - \mathbf{z}^*||^2_{\mathbf{U}^{-1}} | \mathcal{F}^{(k)}\right] &= ||\mathbf{z}^{(k)} - \mathbf{z}^*||^2_{\mathbf{U}^{-1}} - ||\mathbf{z}^{(k)} - \mathbf{z}^*||^2 + ||\mathbf{T}_{\mathrm{P},\rho}\mathbf{z}^{(k)} - \mathbf{z}^*||^2 \\
&= ||\mathbf{z}^{(k)} - \mathbf{z}^*||^2_{\mathbf{U}^{-1}} - 4\rho\left\langle \nabla f(\mathbf{x}^{(k)}) - \nabla f(\mathbf{x}^*), \mathbf{x}^{(k)} - \mathbf{x}^*\right\rangle \\
&\leq ||\mathbf{z}^{(k)} - \mathbf{z}^*||^2_{\mathbf{U}^{-1}} - 4\rho\beta||\mathbf{x}^{(k)} - \mathbf{x}^*||^2.
\end{aligned} \tag{4.11}$$

We will utilise (4.11) in the following theorem to prove the almost sure primal convergence of stochastic standard PDMM under the assumptions stated in Lemma 4.4.

Theorem 4.3. *Stochastic standard PDMM converges almost surely to a primal optimal point* $\mathbf{x}^*$.

Proof. By taking the expectation on both sides of (4.11) and taking the same steps that are used in the proof of Lemma 4.1, we arrive at

$$\sum_{k=0}^{\infty} \mathbb{E}\left[||\mathbf{x}^{(k)} - \mathbf{x}^*||^2\right] \leq \frac{1}{4\rho\beta}||\mathbf{z}^{(0)} - \mathbf{z}^*||^2_{\mathbf{U}^{-1}} < \infty,$$

which shows that the sum of the expected value of the primal error is bounded. Using Markov's inequality [19, Cor. 5.1] and the equation above we get

$$\sum_{k=0}^{\infty} \mathbb{P}\left\{||\mathbf{x}^{(k)} - \mathbf{x}^*||^2 \geq \epsilon\right\} \leq \frac{1}{\epsilon}\sum_{k=0}^{\infty} \mathbb{E}\left[||\mathbf{x}^{(k)} - \mathbf{x}^*||^2\right] < \infty \quad \forall \epsilon > 0.$$

Now we can apply Borel Cantelli's lemma [19, Theorem 10.5] to conclude that

$$\mathbb{P}\left\{||\mathbf{x}^{(k)} - \mathbf{x}^*||^2 \geq \epsilon \quad \text{i.o.}\right\} = 0.$$

So there is a set of events with probability 1 where $\forall\epsilon\ \exists k_0$ such that $||\mathbf{x}^{(k)} - \mathbf{x}^*||^2 < \epsilon, \forall k \geq k_0$. Hence $||\mathbf{x}^{(k)} - \mathbf{x}^*||^2 \to 0$ almost surely. Because $\mathbf{z}^* \in \text{fix}\,(\mathbf{T}_{\text{P},\rho})$, $\mathbf{x}^*$ is a primal optimal solution by construction, see (4.9) and [5]. □

4.4. Summary

In Section 4.1 a formal definition is given of stochastic PDMM. This is a general framework which can be used to describe various different versions of PDMM, as will be done in Chapter 5. Sections 4.2 and 4.3 state two convergence proofs related to stochastic PDMM. For both proofs it is assumed that each edge is updated based on a Bernoulli random variable that has a nonzero mean, thus each edge is updated with a nonzero probability. The main results of this chapter are:

- Stochastic θ-averaged PDMM converges almost surely for arbitrary CCP cost functions, see Theorem 4.2;
- Stochastic standard PDMM converges almost surely for strongly convex and differentiable cost functions, see Theorem 4.3.

5 Asynchronous PDMM Algorithms

In this chapter we will define and analyse various asynchronous PDMM algorithms. We will start by giving some general definitions related to asynchronous algorithms in Section 5.1. In Section 5.2, we discuss the link between stochastic PDMM and asynchronous PDMM in the presence of transmission loss. After this, we will analyse the pseudocode and the convergence of a number of different asynchronous PDMM algorithms in Sections 5.3-5.5. Subsequently, we further analyse some of the problems that occur with broadcast PDMM in Section 5.6. Finally, we compare the various algorithms discussed throughout this chapter in Section 5.7.

5.1. Definitions

In this section we will give some definitions related to various asynchronous PDMM algorithms that are discussed in this chapter. These definitions are summarised in Table 5.1. As mentioned in Section 3.4.3, a distinction can be made between a so called unicast or broadcast implementation. For asynchronous implementations of PDMM, instead of updating all nodes in the network at each iteration k, only one node (or a subset of nodes), $\mathcal{V}_a^{(k+1)} \in 2^{\mathcal{V}}$, is activated. In addition to the active nodes, we also define a subset of active directive edges as $\mathcal{E}_a^{(k+1)} \in 2^{\mathcal{E}_{\text{dir}}}$, which can be used to model transmission losses between nodes.

The various implementations of stochastic PDMM can all be divided into the following four main components that happen at each iteration:

1. **Activation:** A random subset of nodes, $\mathcal{V}_a$, and a random subset of edges, $\mathcal{E}_a$, are made active. Note that if $\mathcal{V}_a = \mathcal{V}$ and $\mathcal{E}_a = \mathcal{E}$, all nodes will perform complete updates at each iteration and the algorithm is equivalent to the regular synchronous updating scheme.
2. **Active node updates:** The activated nodes perform a primal update followed by auxiliary and/or dual updates for each of their neighbours.
3. **Transmit updated variables:** The updated variables are transmitted to the neighbouring nodes. This could consist of the unicast transmission of auxiliary/dual variables or the broadcast transmission of the updated primal variables.
4. **Secondary node updates:** Each secondary node, defined as a neighbouring node that is connected to an active node by an active edge, receives an updated variable from the active node. These secondary nodes can then execute local updates with the newly received values.

The set of active nodes, $\mathcal{E}_a$, is used as an entry requirement for performing secondary node updates. A node that is a neighbour of an active node, but has an inactive edge linking itself with the active node, will not receive the updated variables due to transmission failure. Because of this, the local update can not be performed at the neighbouring node and thus it will not be considered a secondary node.

To clearly visualise the effect of transmission losses on an iteration, we will give a simple example. Consider a graph, $\mathcal{G}$, with five nodes and five edges. At a particular iteration both node one and node five are active, thus $\mathcal{V}_a = \{1, 5\}$. Figure 5.1 depicts this network in two scenarios. One where communication among all edges is good and one which contains two lossy connections that form inactive

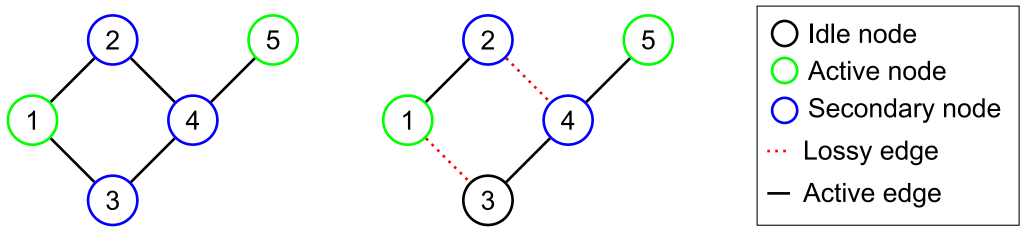

Figure 5.1: Left: Lossless asynchronous distributed network, $\mathcal{V}_a = \{1, 5\}$ and $\mathcal{E}_a = \mathcal{E}$; Middle: Lossy synchronous distributed network, $\mathcal{V}_a = \{1, 5\}$ and $\mathcal{E}_a = \{(1, 2), (3, 4), (4, 5)\}$; Right: Legend.

Table 5.1: A list of definitions related to the PDMM algorithms.

Unicast	A unique message is transmitted to each neighbour of an active node.
Broadcast	One general message transmitted to all neighbours of an active node.
k	Iteration number.
Active nodes	All nodes that perform a primal update at the current iteration.
Secondary nodes	All nodes that receive updated variables from an active node at the current iteration.
Active edge	A directive edge connecting two nodes that supports the transmission of messages in both directions.
Inactive edge	An edge connecting two nodes that does not support any transmission of messages. Also referred to as *lossy edge*.
$\mathcal{V}_a^{(k+1)}$	Set of active nodes at iteration k.
$\mathcal{E}_a^{(k+1)}$	Set of active edges over which communication is possible at iteration k.

edges. When comparing the two scenarios, it can be seen that node three becomes a secondary node in the lossless case. In the lossy case the variables at node three, relating to active node one, will not be updated, because the updated variables from node one are never received.

5.2. Convergence

As proved in Chapter 4, stochastic PDMM converges almost surely. Asynchronous PDMM can be seen as a specific case of stochastic PDMM where the entries of $\mathbf{U}^{(k)}$ are defined as Bernoulli random variables, $V_{i|j}$, with the following mean

$$v_{i|j} = \mathbb{P}\left[\left\{j \in \mathcal{V}_a^{(k)}\right\}\right].$$

In some cases transmission loss can also be seen as a specific case of stochastic PDMM where a Bernoulli random variable $E_{i|j}$, with mean

$$e_{i|j} = \mathbb{P}\left[\left\{(i, j) \in \mathcal{E}_a^{(k)}\right\}\right],$$

determines which directed edges are active. A combination of asynchronous updating and transmission loss can be modelled with random variable

$$U_{i|j} = V_{i|j} E_{i|j},$$

where we assume $V_{i|j}$ and $E_{i|j}$ are independent and thus $\mathbb{E}[U_{i|j}] = \mu_{i|j} = v_{i|j} e_{i|j}$.

In applied PDMM algorithms each node has certain variables stored locally and the PDMM iteration is divided up into multiple updating steps. The general definition of stochastic PDMM that is given in Chapter 4, only models a complete iteration of the auxiliary variable $\mathbf{z}$. So this definition is limited in the PDMM variations it covers. For instance it does not take inconsistencies between values for the same variable, that are stored at different nodes, into account. In the following sections we will analyse a number of different PDMM variations to indicate which of these implementations follow the stochastic PDMM definition and thus converge according to the proof given in Chapter 4.

Algorithm 2 Asynchronous θ-averaged PDMM (unicast).

```
 1: Initialise:  $\mathbf{z}^{(0)} \in \mathbb{R}^{2m_\mathcal{E}}$                                  ▷ Initialisation
 2: for $k = 0, ...,$ do
 3:     Select a random subset of active nodes: $\mathcal{V}_a^{(k+1)} \in 2^{\mathcal{V}}$
 4:     Select a random subset of active edges: $\mathcal{E}_a^{(k+1)} \in 2^{\mathcal{E}}$
 5:     for $i \in \mathcal{V}_a^{(k+1)}$ do                                          ▷ Active node updates
 6:         $\mathbf{x}_i^{(k+1)} = \arg\min_{\mathbf{x}_i} \left[f_i(\mathbf{x}_i) + \sum_{j\in\mathcal{N}_i}\left((\mathbf{z}_{i|j}^{(k)})^T\mathbf{A}_{i|j}\mathbf{x}_i + \frac{\rho}{2}||\mathbf{A}_{i|j}\mathbf{x}_i||_2^2\right)\right]$
 7:         for all $j \in \mathcal{N}_i$ do
 8:             $\boldsymbol{\lambda}_{i|j}^{(k+1)} = \mathbf{z}_{i|j}^{(k)} + \rho\mathbf{A}_{i|j}\mathbf{x}_i^{(k+1)}$
 9:             $\mathbf{y}_{i|j}^{(k+1)} = 2\boldsymbol{\lambda}_{i|j}^{(k+1)} - \mathbf{z}_{i|j}^{(k)}$
10:         end for
11:     end for

12:     for $i \in \mathcal{V}_a^{(k+1)}, j \in \mathcal{N}_i$ do                    ▷ Transmit updated variables (unicast)
13:         Node$_j$ $\leftarrow$ Node$_i$($\mathbf{y}_{i|j}^{(k+1)}$)
14:     end for

15:     for $i \in \mathcal{V}_a^{(k+1)}, j \in \mathcal{N}_i \,:\, (i|j) \in \mathcal{E}_a^{(k+1)}$ do      ▷ Secondary node updates
16:         $\mathbf{z}_{j|i}^{(k+1)} = (1-\theta)\mathbf{z}_{j|i}^{(k)} + \theta\mathbf{y}_{i|j}^{(k+1)}$
17:     end for
18: end for
```

5.3. z-update PDMM

When deriving PDMM using monotone operator theory, the most natural PDMM implementation is based around the update of the auxiliary variable $\mathbf{z}$. Each iteration of PDMM is completely characterised by the vector $\mathbf{z}^{(k)}$ and therefore only $\mathbf{z}^{(0)}$ needs to be initialised. In this section we will analyse a unicast and broadcast implementation of z-update PDMM.

5.3.1. Unicast

Algorithm 2 contains the pseudocode for asynchronous unicast PDMM. For the unicast implementation, the auxiliary variable $\mathbf{y}_{i|j}^{(k+1)}$ is calculated at the active node i, for each of its neighbours $j \in \mathcal{N}_i$, and transmitted to these neighbours in separate unicast messages. The secondary nodes can then use their received value to calculate $\mathbf{z}_{j|i}^{(k+1)}$.

In a lossless case Algorithm 2 can be seen as an instance of stochastic PDMM and thus converges. In the case of a transmission loss between active node i and secondary node j, the active node updates are performed at node i, but the secondary node updates are not performed at node j. Because the active node updates at node i are completely characterised by $\mathbf{z}_{i|j}, \forall j \in \mathcal{N}_i$, the unnecessarily updated variables will simply be overwritten in another iteration that node i is activated. Because of this property and the fact that each $\mathbf{z}_{i|j}$ is only required at node i, unicast z-update PDMM can be seen as a form of stochastic PDMM and thus converges.

5.3.2. Broadcast

It is also possible to implement asynchronous PDMM using broadcast communication. The pseudocode for this implementation is given in Algorithm 3. The same message is broadcast from an active node i to all of its neighbours $j \in \mathcal{N}_i$. Upon receiving this message, the secondary nodes can perform their necessary local updates. The common updated variable, that all neighbours $j \in \mathcal{N}_i$ can use for their respective updates, is $\mathbf{x}_i^{(k+1)}$. This value will be broadcast to all neighbours of an active node i.

In a lossless case Algorithm 3 can be seen as an instance of stochastic PDMM and thus converges. The convergence in the presence of transmission losses is slightly more complicated. As can be seen from lines 17 and 19 in Algorithm 3, both $\mathbf{z}_{i|j}$ and $\mathbf{z}_{j|i}$ are required at node i for each $j \in \mathcal{N}_i$. This means that in the broadcast implementation of PDMM each variable, $\mathbf{z}_{i|j}$, is calculated at both node i and node j using locally available variables. By inspection of (3.6)-(3.9), it can be seen that the global formulation

Algorithm 3 Asynchronous θ-averaged PDMM (broadcast).

```
1: Initialise: $\mathbf{z}^{(0)} \in \mathbb{R}^{2m\mathcal{E}}$                                   ▷ Initialisation
2: for $k = 0, ...,$ do
3:     Select a random subset of active nodes: $\mathcal{V}_a^{(k+1)} \in 2^{\mathcal{V}}$
4:     Select a random subset of active edges: $\mathcal{E}_a^{(k+1)} \in 2^{\mathcal{E}}$
5:     for $i \in \mathcal{V}_a^{(k+1)}$ do                                                  ▷ Active node updates
6:         $\mathbf{x}_i^{(k+1)} = \arg\min_{\mathbf{x}_i} \left[f_i(\mathbf{x}_i) + \sum_{j\in\mathcal{N}_i} \left((\mathbf{z}_{i|j}^{(k)})^T \mathbf{A}_{i|j}\mathbf{x}_i + \frac{\rho}{2}||\mathbf{A}_{i|j}\mathbf{x}_i||_2^2\right)\right]$
7:         for all $j \in \mathcal{N}_i$ do
8:             $\boldsymbol{\lambda}_{i|j}^{(k+1)} = \mathbf{z}_{i|j}^{(k)} + \rho\mathbf{A}_{i|j}\mathbf{x}_i^{(k+1)}$
9:             $\mathbf{y}_{i|j}^{(k+1)} = 2\boldsymbol{\lambda}_{i|j}^{(k+1)} - \mathbf{z}_{i|j}^{(k)}$
10:            $\mathbf{z}_{j|i}^{(k+1)} = (1-\theta)\mathbf{z}_{j|i}^{(k)} + \theta\mathbf{y}_{i|j}^{(k+1)}$
11:        end for
12:    end for

13:    for all $i \in \mathcal{V}_a^{(k+1)}, j \in \mathcal{N}_i$ do                    ▷ Transmit updated variables (broadcast)
14:        $\text{Node}_j \leftarrow \text{Node}_i(\mathbf{x}_i^{(k+1)})$
15:    end for

16:    for $i \in \mathcal{V}_a^{(k+1)}, j \in \mathcal{N}_i : (i|j) \in \mathcal{E}_a^{(k+1)}$ do          ▷ Secondary node updates
17:        $\boldsymbol{\lambda}_{i|j}^{(k+1)} = \mathbf{z}_{i|j}^{(k)} - \rho\mathbf{A}_{j|i}\mathbf{x}_i^{(k+1)}$
18:        $\mathbf{y}_{i|j}^{(k+1)} = 2\boldsymbol{\lambda}_{i|j}^{(k+1)} - \mathbf{z}_{i|j}^{(k)}$
19:        $\mathbf{z}_{j|i}^{(k+1)} = (1-\theta)\mathbf{z}_{j|i}^{(k)} + \theta\mathbf{y}_{i|j}^{(k+1)}$
20:    end for
21: end for
```

of PDMM only allows for one value for each **z** entry. After a transmission loss between active node i and secondary node j, a mismatch occurs in the stored values for variable $\mathbf{z}_{j|i}$. This prevents broadcast z-update PDMM form converging. We will give a numerical example of this behaviour in Section 5.6.

5.4. λ-update PDMM

The non averaged version of PDMM can be easily rewritten to a form containing updates of the dual variable $\boldsymbol{\lambda}$ and primal variable **x** only. Because the λ-update variant of PDMM is used in earlier work, like [4] and [6], and because it could be insightful for the analysis of the dual variables, we include this version in our analysis.

As shown in [5], the reformulation is done by using the fact that

$$\mathbf{z}^{(k+1)} = \mathbf{P}\left(\boldsymbol{\lambda}^{(k+1)} + \rho\mathbf{C}\mathbf{x}^{(k+1)}\right).$$

By using the equation above, the auxiliary variables **y** and **z** are not necessarily needed and we arrive at the following two updating equations:

$$\mathbf{x}^{(k+1)} = \arg\min_{\mathbf{x}} \left[f(\mathbf{x}) + \langle \mathbf{C}^T\mathbf{P}\boldsymbol{\lambda}^{(k)}, \mathbf{x}\rangle + \frac{\rho}{2}||\mathbf{C}\mathbf{x} + \mathbf{P}\mathbf{C}\mathbf{x}^{(k)}||^2\right],$$

$$\boldsymbol{\lambda}^{(k+1)} = \mathbf{P}\boldsymbol{\lambda}^{(k)} + \rho(\mathbf{C}\mathbf{x}^{(k+1)} + \mathbf{P}\mathbf{C}\mathbf{x}^{(k)}).$$

The averaging step in averaged PDMM is done at the z-update step. For the use of averaging in the λ-update version, some rewriting is needed in order to get the correct updating equations. We will use the following property, derived from (3.7):

$$\mathbf{z}^{(k)} = \boldsymbol{\lambda}^{(k+1)} - \rho\mathbf{C}\mathbf{x}^{(k+1)}. \tag{5.1}$$

Algorithm 4 Asynchronous λ-update PDMM (unicast).

```
1:  Initialise: $\boldsymbol{\lambda}^{(0)} \in \mathbb{R}^{2m_{\mathcal{E}}}, \mathbf{x}^{(0)} \in \mathbb{R}^{n_{\mathcal{V}}}$        ▷ Initialisation
2:  for $k = 0, ...,$ do
3:      Select a random subset of active nodes: $\mathcal{V}_a^{(k+1)} \in 2^{\mathcal{V}}$
4:      Select a random subset of active edges: $\mathcal{E}_a^{(k+1)} \in 2^{\mathcal{E}}$
5:      for $i \in \mathcal{V}_a^{(k+1)}$ do        ▷ Active node updates
6:          $\mathbf{x}_i^{(k+1)} = \arg\min_{\mathbf{x}_i} \left[ f_i(\mathbf{x}_i) + \sum_{j\in\mathcal{N}_i} \left( (\boldsymbol{\lambda}_{j|i}^{(k)})^T \mathbf{A}_{i|j}\mathbf{x}_i + \frac{\rho}{2}||\mathbf{A}_{i|j}\mathbf{x}_i + \mathbf{A}_{j|i}\mathbf{x}_j^{(k)}||_2^2 \right) \right]$
7:          for all $j \in \mathcal{N}_i$ do
8:              $\boldsymbol{\lambda}_{i|j}^{(k+1)} = \boldsymbol{\lambda}_{j|i}^{(k)} + \rho \mathbf{A}_{i|j}(\mathbf{x}_i^{(k+1)} - \mathbf{x}_j^{(k)})$
9:          end for
10:     end for

11:     for $i \in \mathcal{V}_a^{(k+1)}, j \in \mathcal{N}_i \ : \ (i|j) \in \mathcal{E}_a^{(k+1)}$ do        ▷ Transmit updated variables (unicast)
12:         Node$_j \leftarrow$ Node$_i(\mathbf{x}_i^{(k+1)})$
13:         Node$_j \leftarrow$ Node$_i(\boldsymbol{\lambda}_{i|j}^{(k+1)})$
14:     end for
15: end for
```

Using (5.1), the z-update equation can be rewritten as

$$\begin{aligned}
\mathbf{z}^{(k+1)} &= (1-\theta)\mathbf{z}^{(k)} + \theta\left(\mathbf{P}\mathbf{z}^{(k)} + 2\rho\mathbf{P}\mathbf{C}\mathbf{x}^{(k+1)}\right) \\
&= (1-\theta)\left(\boldsymbol{\lambda}^{(k+1)} - \rho\mathbf{C}\mathbf{x}^{(k+1)}\right) + \theta\left(\mathbf{P}(\boldsymbol{\lambda}^{(k+1)} - \rho\mathbf{C}\mathbf{x}^{(k+1)}) + 2\rho\mathbf{P}\mathbf{C}\mathbf{x}^{(k+1)}\right) \\
&= (1-\theta)\left(\boldsymbol{\lambda}^{(k+1)} - \rho\mathbf{C}\mathbf{x}^{(k+1)}\right) + \theta\left(\mathbf{P}\boldsymbol{\lambda}^{(k+1)} + \rho\mathbf{P}\mathbf{C}\mathbf{x}^{(k+1)}\right).
\end{aligned} \tag{5.2}$$

Now the λ-update equation can be written as

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

where (5.1) was used, followed by (5.2).

Apart from this altered λ-update equation, the x-update equation also needs to be changed to work with θ-averaging. This leads to long expressions for both the x and λ-update, which are not very insightful. For this reason the algorithms for λ-update PDMM are only given for non-averaged PDMM.

Following the same reasoning as in Section 5.3, the pseudocode for asynchronous λ-update PDMM can be derived for both the unicast and the broadcast scheme. The resulting implementations are given in Algorithms 4 and 5 respectively. These algorithms also include the possibility to model transmission losses by defining the set of active edges $\mathcal{E}_a$.

As can be seen in Algorithms 4 and 5, the primal update equation at node i requires the previous values of the primal values of all neighbouring nodes. This means that the values of $\mathbf{x}$ also need initialising and the value of each $\mathbf{x}_i$ is not only needed at node i itself, but also at the neighbours of node i. Furthermore, the initialisation should be shared before taking any update steps. If this is not done, multiple different values will be stored for each $\mathbf{x}_i$ entry. This would not match the definition of PDMM as stated in (3.6)-(3.9) and could lead to convergence issues. Furthermore, in the case of unicast λ-update PDMM, in addition to the unicast transmission of the dual variables, a broadcast transmission of the primal variable is also required for λ-update PDMM. This results in more communication overhead than z-update PDMM.

In a lossless case Algorithms 4 and 5 can be seen as an instances of stochastic PDMM and thus converge. Following the same lines of reasoning as in Section 5.3, we conclude that unicast asynchronous λ-update PDMM corresponds to the definition of stochastic PDMM and thus also converges in the presence of transmission loss. Broadcast λ-update PDMM, however, has the same problem as

Algorithm 5 Asynchronous λ-update PDMM (broadcast).

```
1:  Initialise: $\boldsymbol{\lambda}^{(0)} \in \mathbb{R}^{2m\mathcal{E}}, \mathbf{x}^{(0)} \in \mathbb{R}^{n\mathcal{V}}$    ▷ Initialisation
2:  for $k = 0, ...,$ do
3:      Select a random subset of active nodes: $\mathcal{V}_a^{(k+1)} \in 2^{\mathcal{V}}$
4:      Select a random subset of active edges: $\mathcal{E}_a^{(k+1)} \in 2^{\mathcal{E}}$
5:      for $i \in \mathcal{V}_a^{(k+1)}$ do    ▷ Active node updates
6:          $\mathbf{x}_i^{(k+1)} = \arg\min_{\mathbf{x}_i} \Big[ f_i(\mathbf{x}_i) + \sum_{j\in\mathcal{N}_i} \Big( (\boldsymbol{\lambda}_{j|i}^{(k)})^T \mathbf{A}_{i|j}\mathbf{x}_i + \frac{\rho}{2}||\mathbf{A}_{i|j}\mathbf{x}_i + \mathbf{A}_{j|i}\mathbf{x}_j^{(k)}||_2^2 \Big) \Big]$
7:          for all $j \in \mathcal{N}_i$ do
8:              $\boldsymbol{\lambda}_{i|j}^{(k+1)} = \boldsymbol{\lambda}_{j|i}^{(k)} + \rho\mathbf{A}_{i|j}(\mathbf{x}_i^{(k+1)} - \mathbf{x}_j^{(k)})$
9:          end for
10:     end for

11:     for $i \in \mathcal{V}_a^{(k+1)}, j \in \mathcal{N}_i$ do    ▷ Transmit updated variables (broadcast)
12:         Node$_j$ $\leftarrow$ Node$_i(\mathbf{x}_i^{(k+1)})$
13:     end for

14:     for $i \in \mathcal{V}_a^{(k+1)}, j \in \mathcal{N}_i \ : \ (i|j) \in \mathcal{E}_a^{(k+1)}$ do    ▷ Secondary node updates
15:         $\boldsymbol{\lambda}_{i|j}^{(k+1)} = \boldsymbol{\lambda}_{j|i}^{(k)} - \rho\mathbf{A}_{j|i}(\mathbf{x}_i^{(k+1)} - \mathbf{x}_j^{(k)})$
16:     end for
17: end for
```

broadcast z-update PDMM. As can be seen in line 15 of Algorithm 5, for all $i \in \mathcal{V}$ and $j \in \mathcal{N}_i$ both $\boldsymbol{\lambda}_{i|j}$ and $\boldsymbol{\lambda}_{j|i}$ are required at node i. After a transmission loss from active node i to secondary node j, a mismatch occurs in the stored values for variable $\boldsymbol{\lambda}_{i|j}$, preventing convergence. With perfect transmissions, asynchronous broadcast λ-update PDMM does not have this mismatch problem and thus it has the same convergence properties as asynchronous unicast λ-update PDMM.

5.5. Differential PDMM

In [25] an adaptive differentially quantised version of PDMM is proposed. This PDMM version reduces transmission costs when compared to regular PDMM and is also shown to be suitable for subspace based privacy preservation in [7]. In this section we will analyse differential PDMM. The adaptive quantisation part as discussed in [25] is a specific extension of differential PDMM. The adaption rate of the quantiser needs to be tuned according to the convergence rate of the problem at hand. However, the privacy preservation capabilities presented in [7] do not rely on the quantisation. So, to keep the analysis simple we will use non-quantised differential PDMM. For this reason we will only analyse non-quantised differential PDMM.

Differential PDMM works by sending the difference between the updated auxiliary variable $\mathbf{y}^{(k+1)}$ and its previous value $\mathbf{y}^{(k)}$. As long as the initial values $\mathbf{y}^{(0)}$ are transmitted via secure encryption, subspace privacy preservation works is effective. The pseudocode for differential PDMM is given in Algorithm 6. In a lossless case Algorithm 6 can be seen as an instance of stochastic PDMM and thus converges. In the presence of transmission loss, despite being a unicast type implementation, differential PDMM has a similar mismatch issue as the previously discussed broadcast implementations. As can be seen in lines 9 and 10 in Algorithm 6, for all $i \in \mathcal{V}$ and $j \in \mathcal{N}_i$, both $\mathbf{z}_{i|j}$ and $\mathbf{z}_{j|i}$ are needed at i. Again this is not taken into account in the definition of stochastic PDMM and the loss of a transmission between active node i and secondary node j, will result in a mismatch between these values and thus prevent convergence.

5.6. Broadcast analysis

As mentioned in Section 5.3 and Section 5.4, broadcast PDMM does not seem to be robust against transmission loss due to the resulting inconsistency between different instances of the same auxiliary or dual variable. Interestingly, asynchronous unicast PDMM does seem to be robust against transmission losses. In this section we give a numerical example indicating the difference between unicast and

Algorithm 6 Asynchronous θ-averaged differential PDMM (unicast).

```
1:  Initialise:  $\mathbf{z}^{(0)} \in \mathbb{R}^{2m_{\mathcal{E}}}$                                  ▷ Initialisation
2:  for $k = 1, ...,$ do
3:      Select a random subset of active nodes: $\mathcal{V}_a^{(k+1)} \in 2^{\mathcal{V}}$
4:      Select a random subset of active edges: $\mathcal{E}_a^{(k+1)} \in 2^{\mathcal{E}}$
5:      for $i \in \mathcal{V}_a^{(k+1)}$ do                                                   ▷ Active node updates
6:          $\mathbf{x}_i^{(k+1)} = \arg\min_{\mathbf{x}_i} \left[ f_i(\mathbf{x}_i) + \sum_{j \in \mathcal{N}_i} \left( (\mathbf{z}_{i|j}^{(k)})^T \mathbf{A}_{i|j}\mathbf{x}_i + \frac{\rho}{2}||\mathbf{A}_{i|j}\mathbf{x}_i||_2^2 \right) \right]$
7:          for all $j \in \mathcal{N}_i$ do
8:              $\boldsymbol{\lambda}_{i|j}^{(k+1)} = \mathbf{z}_{i|j}^{(k)} + \rho \mathbf{A}_{i|j}\mathbf{x}_i^{(k+1)}$
9:              $\mathbf{y}_{i|j}^{(k+1)} = 2\boldsymbol{\lambda}_{i|j}^{(k+1)} - \mathbf{z}_{i|j}^{(k)}$
10:             $\mathbf{z}_{j|i}^{(k+1)} = (1-\theta)\mathbf{z}_{j|i}^{(k)} + \theta \mathbf{y}_{i|j}^{(k+1)}$
11:             $\mathbf{v}_{j|i}^{(k+1)} = \mathbf{z}_{j|i}^{(k+1)} - \mathbf{z}_{j|i}^{(k)}$
12:         end for
13:     end for

14:     for $i \in \mathcal{V}_a^{(k+1)}, j \in \mathcal{N}_i$ do                    ▷ Transmit updated variables (unicast)
15:         Node$_j$ ← Node$_i$($\mathbf{v}_{j|i}^{(k+1)}$)
16:     end for

17:     for $i \in \mathcal{V}_a^{(k+1)}, j \in \mathcal{N}_i \ : \ (i|j) \in \mathcal{E}_a^{(k+1)}$ do          ▷ Secondary node updates
18:         $\mathbf{z}_{j|i}^{(k+1)} = \mathbf{z}_{j|i}^{(k)} + \mathbf{v}_{j|i}^{(k+1)}$
19:     end for
20: end for
```

broadcast PDMM in the presence of transmission losses. Additionally, we discuss the initialisation of broadcast PDMM and propose an adaption to make broadcast PDMM slightly more robust.

In Algorithm 3 it can be seen that the value of $\mathbf{z}_{i|j}$ is used at both node i and node j, when using a broadcast implementation. Because it is likely that a transmission loss will result in two different stored values for the same variable, we use a notational convention to indicate the different versions of each variable. We define $\mathbf{z}_{i|j,l}$ as *local* copies of $\mathbf{z}_{i|j}$, because they are stored locally at node i and similarly we define $\mathbf{z}_{i|j,r}$, which are *remote* copies of $\mathbf{z}_{i|j}$ stored remotely at node j.

We only analyse the asynchronous z-update implementation of PDMM, but a similar analysis can also be done for λ-update PDMM, differential PDMM and synchronous versions of all discussed PDMM implementations.

5.6.1. Transmission loss example

In this section, the problem that occurs when implementing broadcast PDMM in combination with transmission losses is illustrated with a simple numerical example.

Consider a network with two nodes that are connected by one edge. In this case

$$\mathbf{C} = \begin{bmatrix} A_{1|2} & 0 \\ 0 & A_{2|1} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix},$$
$$\mathbf{D} = \mathbf{C}^T\mathbf{C} = \mathbf{I}.$$

If we consider a simple distributed averaging problem where the measurement vector is $\mathbf{a} = \mathbf{1}$ and $\rho = 0.4$, we can derive the following update equations by using (7.1):

$$x_1^{(k+1)} = \frac{a_1 - z_{1|2,l}^{(k)}}{1+\rho} = \frac{1 - z_{1|2,l}^{(k)}}{1.4},$$

$$x_2^{(k+1)} = \frac{a_2 + z_{2|1,l}^{(k)}}{1+\rho} = \frac{1 + z_{2|1,l}^{(k)}}{1.4},$$

Table 5.2: Values of asynchronous unicast PDMM variables at node one and two, with a transmission failure between node one and two at $k = 1$. Bold values are updated in the respective iteration.

Node	Variable	$k=0$	$k=1$	$k=2$	$k=3$	...	$k \to \infty$
1	x_1	-	**0.7143**	0.7143	**1.224**		1
	$z_{1\|2,l}$	0	0	$-$**0.5714**	-0.5714		-0.4
	$z_{2\|1,r}$	0	**0.5714**	0.5714	**0.3265**		0.4
2	x_2	-	0	**0.7143**	0.7143		1
	$z_{1\|2,r}$	0	0	$-$**0.5714**	-0.5714		-0.4
	$z_{2\|1,l}$	0	0	0	**0.3265**		0.4

Table 5.3: Values of asynchronous broadcast PDMM variables at node one and two, with a transmission failure between node one and two at $k = 1$. Bold values have been updated in the respective iteration.

Node	Variable	$k=0$	$k=1$	$k=2$	$k=3$	...	$k \to \infty$
1	x_1	-	**0.7143**	0.7143	**0.7143**		0.7143
	$z_{1\|2,l}$	0	0	**0**	0		0
	$z_{2\|1,r}$	0	**0.5714**	0.5714	**0.5714**		0.5714
2	x_2	-	0	**0.7143**	0.7143		0.7143
	$z_{1\|2,r}$	0	0	$-$**0.5714**	-0.5714		-0.5714
	$z_{2\|1,l}$	0	0	0	**0**		0

In the case of unicast, the z-update equations are the same for local and remote $z_{i|j}$'s, because the value is only calculated at the active node after the primary update. The value of $z_{i|j}$ is then sent to the corresponding neighbour. The z-update equations for unicast are as follows:

$$z_{1|2,r}^{(k+1)} = z_{1|2,l}^{(k+1)} = z_{2|1,l}^{(k)} + 2\rho A_{2|1} x_2^{(k+1)} = z_{2|1,l}^{(k)} - 0.8 x_2^{(k+1)},$$

$$z_{2|1,r}^{(k+1)} = z_{2|1,l}^{(k+1)} = z_{1|2,l}^{(k)} + 2\rho A_{1|2} x_1^{(k+1)} = z_{1|2,l}^{(k)} + 0.8 x_1^{(k+1)}.$$

In the case of broadcast, the z-update equations are different for local and remote $z_{i|j}$'s. This is because they are calculated with the values that are available at the respective nodes. This leads to the following z-update equations:

$$z_{1|2,l}^{(k+1)} = z_{2|1,r}^{(k)} + 2\rho A_{2|1} x_2^{(k+1)} = z_{2|1,r}^{(k)} - 0.8 x_2^{(k+1)},$$

$$z_{2|1,l}^{(k+1)} = z_{1|2,r}^{(k)} + 2\rho A_{1|2} x_1^{(k+1)} = z_{1|2,r}^{(k)} + 0.8 x_1^{(k+1)},$$

$$z_{1|2,r}^{(k+1)} = z_{2|1,l}^{(k)} + 2\rho A_{2|1} x_2^{(k+1)} = z_{2|1,l}^{(k)} - 0.8 x_2^{(k+1)},$$

$$z_{2|1,r}^{(k+1)} = z_{1|2,l}^{(k)} + 2\rho A_{1|2} x_1^{(k+1)} = z_{1|2,l}^{(k)} + 0.8 x_1^{(k+1)}.$$

Using these update equations, the PDMM variables are calculated for the first three iterates. As a simple case we consider an asynchronous updating scheme where node one and node two are activated one by one. Furthermore, the transmission from node one to node two after the first x-update is assumed to fail. Tables 5.2 and 5.3 show the resulting values for the first three iterates.

In Table 5.2 the results for unicast PDMM are stated. We can see that the transmission failure at $k = 1$ causes a mismatch in values stored for the variable $z_{2|1}$. From $k = 3$ this mismatch is rectified. Assuming there are no more transmission failures, the convergence to the optimal $\mathbf{x}^*$ will continue as normal. This is possible because the value of $z_{2|1,r}$, which is stored at node 1, is not needed for any update equations. In general the remote copies $\mathbf{z}_{i|j,r}$ are not used, which is also why it is not described in Algorithm 2.

In Table 5.3 the results for broadcast PDMM are stated. We can see that within three iterations, broadcast PDMM, in combination with a transmission loss, gets stuck in a limit cycle with $\mathbf{x}^{(k)} = 0.7143 \cdot \mathbf{1} \neq \mathbf{1}$. This shows that it reaches a fixed point of broadcast PDMM that does not coincide with the primal optimal $\mathbf{x}^*$.

5.6.2. Inconsistent initialisation

From [5], it is known that PDMM converges for arbitrary initialisation of $\mathbf{z}$. For broadcast PDMM, random initialisation of $\mathbf{z}$ is not trivial. As described throughout this section, there are two instances of each $\mathbf{z}_{i|j}$ that are stored at node i and j respectively. Random initialisation for broadcast PDMM is possible, but extra care needs to be taken to ensure that this initialisation is consistent. If this is not the case, the algorithm will start with a mismatch between $\mathbf{z}_l$ and $\mathbf{z}_r$ and, as shown in the previous example, this can prevent the algorithm from converging to a primal optimal point.

Throughout the rest of this report a consistent initialisation of $\mathbf{z}_{i|j,l}$ and $\mathbf{z}_{i|j,r}$ is assumed when using broadcast PDMM. In a practical implementation this would require each node i to share its initialised $\mathbf{z}_{i|j,l}$ with neighbour j to initialise $\mathbf{z}_{i|j,r}$, through a reliable communication channel. In an application where the network topology could vary during the optimisation process, this would require the reliable exchange of dual variables after every network change, which is undesirable.

5.6.3. Broadcast PDMM adaption

A proposed adaption to broadcast PDMM is to use acknowledge messages in combination with the possibility to revert an updated auxiliary variable at an active node. We will call this adaption ”robust broadcast PDMM”. For this implementation, each secondary node that receives an updated x value from an active node, should send an acknowledge statement to the active node in question. By checking for these acknowledge messages, an active node i can assume the updated $\mathbf{x}_i$ value never arrived at a neighbour j if it has not received an acknowledge statement from this neighbour. So if, after a predetermined acknowledge timeout, an acknowledge from neighbour j has not been received by the active node, this active node can revert the updated auxiliary variable corresponding to this neighbour, $\mathbf{z}_{j|i,r}$. By doing this, $\mathbf{z}_{j|i,l}^{(k+1)}$ and $\mathbf{z}_{j|i,r}^{(k+1)}$ will still have same value after a transmission loss.

As long as the initialisation of $\mathbf{z}^{(0)}$ is consistent across nodes, meaning that $\mathbf{z}_l^{(0)} = \mathbf{z}_r^{(0)}$, robust broadcast PDMM is equivalent to unicast PDMM with regards to convergence in the presence of transmission loss.

Although this adaption may seem like a good solution, it also has a number of downsides. The acknowledge messages obviously add extra communication cost to the algorithm. Furthermore, if an updated variable does arrive at a secondary node j, but its acknowledge message does not arrive at the active node i, a similar mismatch situation will occur as without the adaption. In a practical implementation the chance of an acknowledge message transmission failure would likely be smaller than the chance of a regular transmission failure, due to the fact that the size of acknowledge message is probably much smaller. Nevertheless, this adaption only shifts the problem and does not provide a completely robust solution in a practical scenario. A robust solution would most likely require the use of a reliable communication link, for instance TCP [27].

5.7. Comparison

In this chapter a number of different PDMM implementations have been discussed. In this section these implementations will be compared. An overview of the comparison is given in Table 5.4.

Apart from having a direct link to the dual variables of the original optimisation problem, there are no benefits to the λ-update algorithm. There are two disadvantages to this algorithm. Firstly, more variables need to be stored at each node and secondly the x-update step also needs changing when implementing operator averaging.

The z-update variant has a direct interpretation in monotone operator theory, which is useful for convergence analysis. Because of this and the previously mentioned disadvantages of the λ-update variant, in the remainder of the report only the z-update variant will be used.

When looking at the difference between unicast and broadcast, we can see that broadcast requires twice the amount of stored variables. This does not only increase the amount of required storage at a node, but also links the initialisation of different nodes. With a unicast implementation, each node can arbitrarily initialise $\mathbf{z}_{i|j}$ for each of its neighbours. When this is done in the case of a broadcast implementation, each $\mathbf{z}_{i|j}$ is stored at two nodes. Because of this, the initialised values will need to be shared reliably among neighbours to ensure there is consistency between the values that are stored at different nodes for the same variable.

Differential PDMM has similar properties to broadcast z-update PDMM. It is not robust against transmission loss, but it does lend itself to subspace based privacy preservation. If implemented correctly,

Table 5.4: Comparison between different versions of asynchronous PDMM: z-update unicast, broadcast and differential, and λ-update unicast and broadcast.

	Unicast	Broadcast
z-update		
Stored variables at node i	$\mathbf{z}_{i\|j} \quad \forall j \in \mathcal{N}_i$	$\mathbf{z}_{i\|j}, \mathbf{z}_{j\|i} \quad \forall j \in \mathcal{N}_i$
Transmitted variables ($j \leftarrow i$)	$\mathbf{y}_{i\|j}$	$\mathbf{x}_i$
Transmissions per iteration	d_i	1
Operator averaging	Only affects z-update step	Only affects z-update step
Transmission loss	Converges	Does not converge
Subspace based privacy	Not possible	Possible
λ-update		
Stored variables at node i	$\mathbf{x}_j, \boldsymbol{\lambda}_{j\|i} \quad \forall j \in \mathcal{N}_i$	$\mathbf{x}_j, \boldsymbol{\lambda}_{i\|j}, \boldsymbol{\lambda}_{j\|i} \quad \forall j \in \mathcal{N}_i$
Transmitted variables ($j \leftarrow i$)	$\boldsymbol{\lambda}_{i\|j}, \mathbf{x}_i$	$\mathbf{x}_i$
Transmissions per iteration	$d_i + 1$	1
Operator averaging	Affects both λ-update and x-update step	Affects both λ-update and x-update step
Transmission loss	Converges	Does not converge
Subspace based privacy	Not possible	Possible
Differential z-update		
Stored variables at node i	$\mathbf{z}_{i\|j}, \mathbf{z}_{j\|i} \quad \forall j \in \mathcal{N}_i$	-
Transmitted variables ($j \leftarrow i$)	$\mathbf{v}_{j\|i}$	-
Transmissions per iteration	d_i	-
Operator averaging	Only affects z-update step	-
Transmission loss	Does not converge	-
Subspace based privacy	Possible	-

adaptive quantised differential PDMM will have the lowest communication cost.

6 Privacy Preservation with Stochastic PDMM

In this chapter we analyse subspace based privacy preservation in combination with stochastic PDMM. We refer to Section 3.5 for background information and definitions relating to subspace based privacy preservation. In previous work only synchronous versions of PDMM have been considered and the steps used to prove the privacy preservation properties do not immediately hold in a stochastic setting. Thus, we will prove the working subspace based privacy preservation with stochastic PDMM using two steps. First, we derive the condition for privacy preservation for broadcast z-update PDMM in Section 6.1. After this, we derive a lower bound on the variance of $\mathbf{z}$ in the subspace $\Psi^\perp$ in Section 6.2. This is done under the assumption that the stochastic updating probabilities are uniform.

6.1. Broadcast z-update privacy analysis

In this section the privacy preservation properties of broadcast z-update PDMM, Algorithm 3, will be analysed, following a similar approach as in [7] and [6]. We will derive a sufficient condition to reach δ-level privacy, based on the variance of $\mathbf{z}^{(k)}_{\Psi^\perp}$.

By rewriting (3.6), for each $\mathbf{x}_i$ the following holds:

$$\mathbf{0} \in \partial f_i(\mathbf{x}_i^{(k+1)}) + \left[\mathbf{C}^T\mathbf{z}^{(k)}\right]_i + \rho d_i \mathbf{x}_i^{(k+1)} = \partial f_i(\mathbf{x}_i^{(k+1)}) + \sum_{j\in\mathcal{N}_i} \mathbf{A}_{i|j}\mathbf{z}^{(k)}_{i|j} + \rho d_i \mathbf{x}_i^{(k+1)}, \tag{6.1}$$

where d_i is the degree of node i. Because we are considering a broadcast implementation of PDMM, the variables that are transmitted are $\mathbf{x}_i^{(k)}$, for each active node at iteration k. If there is a combination of passive and eavesdropping adversaries present in the network, together they will have knowledge

$$\{\mathbf{z}^{(k)}_{i|j}\}_{(i,j)\in\mathcal{E}_c} \cup \{\mathbf{x}_i^{(k)}\}_{i\in\mathcal{V}}.$$

After deducting the known variables from (6.1), what is left is

$$\partial f_i(\mathbf{x}_i^{(k+1)}) + \sum_{j\in\mathcal{N}_{i,h}} \mathbf{A}_{i|j}\mathbf{z}^{(k)}_{i|j} = \partial f_i(\mathbf{x}_i^{(k+1)}) + \sum_{j\in\mathcal{N}_{i,h}} \mathbf{A}_{i|j}\left[\mathbf{\Pi}_\Psi \mathbf{z}^{(k)}\right]_{i|j} + \sum_{j\in\mathcal{N}_{i,h}} \mathbf{A}_{i|j}\left[\mathbf{\Pi}_{\Psi^\perp}\mathbf{z}^{(k)}\right]_{i|j}. \tag{6.2}$$

As is shown in [5], $\mathbf{\Pi}_\Psi\mathbf{z}^{(k)} \to \mathbf{z}^*$ for any $\mathbf{z}^{(0)}$. Because of this the corresponding term in (6.2) will eventually always have the same value and thus have zero variance[1]. Taking this into account, a sufficient condition for δ-level privacy is

$$\exists j \in \mathcal{N}_{i,h} \;:\; \text{var}\left(\left[\mathbf{\Pi}_{\Psi^\perp}\mathbf{z}^{(k)}\right]_{i|j}\right) \geq \frac{\text{var}(\partial f_i(\mathbf{x}_i^{(k+1)}))}{2^{2\delta}-1},$$

[1]Note that if the problem has non-unique primal optimisers there can be a small variance after convergence in the asynchronous case. However, the magnitude of this variance is relatively small and is not controllable. Thus, it is not useful for privacy preservation.

assuming that the random variables have Gaussian distributions and each honest node i has at least one honest neighbour and thus $\mathcal{N}_{i,h} \neq \varnothing$. Thus, the variance of $\mathbf{z}_{\Psi^\perp}^{(k)}$ should be sufficiently large in order to achieve δ-level privacy.

6.2. Lower bound on variance

Deriving a bound for the variance of the auxiliary variable $\mathbf{z}$ in the case of stochastic PDMM is more involved than in the synchronous case. In this section we will utilise the following three lemmas and some properties of the conditional expectation to derive a lower bound on the variance of stochastic PDMM.

Lemma 6.1. *Let $\mathbf{\Pi}_{\Psi^\perp}$ denote the projection onto the subspace $\Psi^\perp$ and $\mathbf{P}$ denote the PDMM permutation matrix. Then*

$$\mathbf{\Pi}_{\Psi^\perp}\mathbf{P} = \mathbf{P}\mathbf{\Pi}_{\Psi^\perp}.$$

Proof. See [7, Lem. 5.2]. □

Lemma 6.2. *Let $\mathbf{x} \in \mathbb{R}^{2m_\mathcal{E}}$ be a random vector and let $\mathbf{\Pi}_{\Psi^\perp}$ denote the projection onto the subspace $\Psi^\perp$. Then*

$$\mathbf{\Pi}_{\Psi^\perp}\mathbb{E}[\mathbf{x}] = \mathbb{E}[\mathbf{\Pi}_{\Psi^\perp}\mathbf{x}].$$

Proof. By linearity of the expectation operator for any deterministic matrix $\mathbf{M}$ (of appropriate dimensions)

$$\mathbf{M}\mathbb{E}[\mathbf{x}] = \mathbb{E}[\mathbf{M}\mathbf{x}].$$

Applying this property with $\mathbf{M} = \mathbf{\Pi}_{\Psi^\perp} \in \mathbb{R}^{2m_\mathcal{E}\times 2m_\mathcal{E}}$ gives the desired result. □

Lemma 6.3. *Let $\mathbf{\Pi}_{\Psi^\perp}$ denote the projection onto the subspace $\Psi^\perp$. Then*

$$\mathbf{\Pi}_{\Psi^\perp}\left(\mathbf{P}\mathbf{C}\mathbf{x}^{(k+1)}\right) = \mathbf{0}.$$

Proof. Since $\Psi = \text{ran}(\mathbf{C}) + \text{ran}(\mathbf{P}\mathbf{C})$, $\mathbf{P}\mathbf{C}\mathbf{x}^{(k+1)} \in \Psi$. Because $\Psi^\perp$ is orthogonal to Ψ, $\mathbf{\Pi}_{\Psi^\perp}\left(\mathbf{P}\mathbf{C}\mathbf{x}^{(k+1)}\right) = \mathbf{0}$. □

When using subspace based privacy preservation the variance of $\mathbf{z}_{\Psi^\perp}^{(k)}$ is used to preserve privacy. In the case of synchrous PDMM a useful closed form expression for $\mathbf{z}_{\Psi^\perp}^{(k)}$ can be derived, as shown in [7]. In the case of stochastic PDMM deriving such a closed form expression is harder as we will show.

By substitution of (3.7)-(3.9) in (4.1), a stochastic PDMM iteration can be expressed as

$$\mathbf{z}^{(k+1)} = \left(\mathbf{I} - \theta\mathbf{U}^{(k+1)}\right)\mathbf{z}^{(k)} + \theta\mathbf{U}^{(k+1)}\left(\mathbf{P}\mathbf{z}^{(k)} + 2\rho\mathbf{P}\mathbf{C}\mathbf{x}^{(k+1)}\right). \tag{6.3}$$

When looking at the part of $\mathbf{z}$ in the subspace $\Psi^\perp$, we can write

$$\mathbf{z}_{\Psi^\perp}^{(k+1)} = \mathbf{\Pi}_{\Psi^\perp}\left(\mathbf{I} - \theta\mathbf{U}^{(k+1)}\right)\mathbf{z}^{(k)} + \theta\mathbf{\Pi}_{\Psi^\perp}\mathbf{U}^{(k+1)}\left(\mathbf{P}\mathbf{z}^{(k)} + 2\rho\mathbf{P}\mathbf{C}\mathbf{x}^{(k+1)}\right),$$

which can not be further simplified because in general $\mathbf{\Pi}_{\Psi^\perp}\mathbf{U}^{(k+1)} \neq \mathbf{U}^{(k+1)}\mathbf{\Pi}_{\Psi^\perp}$. Moreover, because $\mathbf{U}^{(k+1)}$ can have a nonzero component in $\Psi^\perp$,

$$\mathbf{\Pi}_{\Psi^\perp}\left(\mathbf{U}^{(k+1)}\mathbf{P}\mathbf{C}\mathbf{x}^{(k+1)}\right) \neq \mathbf{0}.$$

As a result, unlike with sycnhronous PDMM, with stochastic PDMM $\mathbf{z}_{\Psi^\perp}^{(k)}$ is dependent on $\mathbf{x}^{(k+1)}$ and thus depends on the chosen cost function. This makes it harder to evaluate the variance of $\mathbf{z}_{\Psi^\perp}$ for arbitrary cost functions. Thus, to simplify the analysis, we make use of the observation that in expected value stochastic PDMM with uniform updating probabilities is equivalent to μ-averaged PDMM. We assume uniform updating probabilities, meaning that $\mu_{i|j} = \mu$, $\forall i \in \mathcal{V}, j \in \mathcal{N}_i$. This assumption holds in the case of asynchronous PDMM where one node is selected uniformly at random and updated each iteration, thus $\mu_{i|j} = 1/n$, $\forall i \in \mathcal{V}, j \in \mathcal{N}_i$. The assumption also holds in the case of transmission losses as long as the probability of an edge being (in)active is equal for all edges.

By taking the conditional expectation of (6.3) with respect to a sub-σ-algebra $\mathcal{H} \subset \mathcal{F}$ and using the fact that $\mathbf{U}^{(k+1)}$ is independent of $\mathbf{z}^{(k)}$ and $\mathbf{x}^{(k+1)}$, we get

$$\begin{aligned}\mathbb{E}\left[\mathbf{z}^{(k+1)}|\mathcal{H}\right] &= \mathbb{E}\left[\mathbf{z}^{(k)}|\mathcal{H}\right] - \theta\mathbb{E}\left[\mathbf{U}^{(k+1)}\right]\mathbb{E}\left[\mathbf{z}^{(k)}|\mathcal{H}\right] + \theta\mathbb{E}\left[\mathbf{U}^{(k+1)}\right]\left(\mathbf{P}\mathbb{E}\left[\mathbf{z}^{(k)}|\mathcal{H}\right] + 2\rho\mathbf{P}\mathbf{C}\mathbb{E}\left[\mathbf{x}^{(k+1)}|\mathcal{H}\right]\right)\\ &= \left(\mathbf{I} - \theta\bar{\mathbf{U}}\right)\mathbb{E}\left[\mathbf{z}^{(k)}|\mathcal{H}\right] + \theta\bar{\mathbf{U}}\left(\mathbf{P}\mathbb{E}\left[\mathbf{z}^{(k)}|\mathcal{H}\right] + 2\rho\mathbf{P}\mathbf{C}\mathbb{E}\left[\mathbf{x}^{(k+1)}|\mathcal{H}\right]\right)\\ &= \left(1 - \theta\mu\right)\mathbb{E}\left[\mathbf{z}^{(k)}|\mathcal{H}\right] + \theta\mu\left(\mathbf{P}\mathbb{E}\left[\mathbf{z}^{(k)}|\mathcal{H}\right] + 2\rho\mathbf{P}\mathbf{C}\mathbb{E}\left[\mathbf{z}^{(k)}|\mathcal{H}\right]\right),\end{aligned} \tag{6.4}$$

where for the last step we assume equal updating probabilities.

As discussed in Section 6.1, the part of $\mathbf{z}$ in the subspace $\Psi^{\perp}$ is relevant for privacy preservation. For a given initialisation of $\mathbf{z}^{(0)}$ and sigma-algebra $\mathcal{F}^{(0)} = \sigma(\mathbf{z}^{(0)})$, we can use (6.4) and Lemmas 6.1-6.3 to write

$$\begin{aligned}\mathbb{E}\left[\mathbf{z}_{\Psi^{\perp}}^{(k+1)}|\mathcal{F}^{(0)}\right] &= \left(1-\theta\mu\right)\mathbf{\Pi}_{\Psi^{\perp}}\mathbb{E}\left[\mathbf{z}^{(k)}|\mathcal{F}^{(0)}\right] + \theta\mu\mathbf{\Pi}_{\Psi^{\perp}}\left(\mathbf{P}\mathbb{E}\left[\mathbf{z}^{(k)}|\mathcal{F}^{(0)}\right] + 2\rho\mathbf{P}\mathbf{C}\mathbb{E}\left[\mathbf{x}^{(k+1)}|\mathcal{F}^{(0)}\right]\right)\\ &= \left(\left(1-\theta\mu\right)\mathbf{I} + \theta\mu\mathbf{P}\right)\mathbb{E}\left[\mathbf{z}_{\Psi^{\perp}}^{(k)}|\mathcal{F}^{(0)}\right]\\ &= \left(\left(1-\theta\mu\right)\mathbf{I} + \theta\mu\mathbf{P}\right)^{k}\mathbf{z}_{\Psi^{\perp}}^{(0)}\\ &= \frac{1}{2}\left(\left(\mathbf{I}+\mathbf{P}\right) + \left(2(1-\theta\mu)-1\right)^{k}\left(\mathbf{I}-\mathbf{P}\right)\right)\mathbf{z}_{\Psi^{\perp}}^{(0)},\end{aligned} \tag{6.5}$$

where the last equality is proved in [7] using an eigenvalue decomposition. We will use the final expression from (6.5) in the derivation of a lower bound on the variance.

Let $\mathbf{z}^{(0)}$ be initialised randomly with zero mean noise having covariance matrix $\mathbb{E}\left[\mathbf{z}^{(0)}\mathbf{z}^{(0),T}\right] = \sigma^2\mathbf{I}$. Now we can use the definition of conditional expectation (see Def. 3.8) and apply Jensen's inequality (see Thm. 3.4) to the convex function $\phi(X) = X^2$, to arrive at

$$\mathbb{E}\left[\mathbf{z}_{\Psi^{\perp}}^{(k)}\mathbf{z}_{\Psi^{\perp}}^{(k),T}\right]_{(i|j),(i|j)} = \mathbb{E}\left[([\mathbf{z}_{\Psi^{\perp}}^{(k)}]_{i|j})^2\right] = \mathbb{E}\left[\mathbb{E}\left[(z_{i,\Psi^{\perp}}^{(k)})^2\middle|\mathcal{F}^{(0)}\right]\right] \geq \mathbb{E}\left[\left(\mathbb{E}\left[z_{i,\Psi^{\perp}}^{(k)}\middle|\mathcal{F}^{(0)}\right]\right)^2\right] \quad \forall i \in \mathcal{V}, j \in \mathcal{N}_i. \tag{6.6}$$

From (6.6) we can conclude that the variance of $\mathbf{z}_{\Psi^{\perp}}^{(k)}$, corresponding to the diagonal elements of the covariance matrix $\mathbb{E}\left[\mathbf{z}_{\Psi^{\perp}}^{(k)}\mathbf{z}_{\Psi^{\perp}}^{(k),T}\right]$, is bounded below by an expression involving the entries of (6.5). We can reformulate this as a vector inequality relating to the diagonal of the covariance matrix as follows:

$$\begin{aligned}\operatorname{diag}\left(\mathbb{E}\left[\mathbf{z}_{\Psi^{\perp}}^{(k)}\mathbf{z}_{\Psi^{\perp}}^{(k),T}\right]\right) &\geq \operatorname{diag}\left(\mathbb{E}\left[\mathbb{E}\left[\mathbf{z}_{\Psi^{\perp}}^{(k)}\middle|\mathcal{F}^{(0)}\right]\mathbb{E}\left[\mathbf{z}_{\Psi^{\perp}}^{(k)}\middle|\mathcal{F}^{(0)}\right]^{T}\right]\right)\\ &= \operatorname{diag}\Bigg(\mathbb{E}\Bigg[\frac{1}{4}\left(\left(\mathbf{I}+\mathbf{P}\right) + \left(2(1-\theta\mu)-1\right)^{k}\left(\mathbf{I}-\mathbf{P}\right)\right)\mathbf{z}_{\Psi^{\perp}}^{(0)}\\ &\qquad \times \mathbf{z}_{\Psi^{\perp}}^{(0),T}\left(\left(\mathbf{I}+\mathbf{P}\right) + \left(2(1-\theta\mu)-1\right)^{k}\left(\mathbf{I}-\mathbf{P}\right)\right)^{T}\Bigg]\Bigg)\\ &= \operatorname{diag}\left(\frac{\sigma^2}{2}\mathbf{\Pi}_{\Psi^{\perp}}\left(\left(\mathbf{I}+\mathbf{P}\right) + \left(2(1-\theta\mu)-1\right)^{2k}\left(\mathbf{I}-\mathbf{P}\right)\right)\right),\end{aligned} \tag{6.7}$$

where the equality in the second line is found by filling in (6.5) and the final equality is proved in [7]. This inequality provides a lower bound for the variance of the auxiliary variable that is based on the initialisation variance σ^2. Because we are considering a broadcast implementation, the initial value of each $\mathbf{z}_{i|j}^{(0)}$ should be known at both node i and node j. To prevent the adversaries from gaining all knowledge after the initialisation round, the transmission of the initialised auxiliary values should be done over a securely encrypted communication channel. Once the initialised auxiliary variables are securely shared, only transmissions of the primal variable are needed. These transmissions do not need to be encrypted, because the bound on the auxiliary variance prevents the adversaries from gaining knowledge about the private data.

Summarising, we conclude that the privacy of node i can be guaranteed when using broadcast z-update stochastic PDMM under the following assumptions:

1. The values of $\{\mathbf{z}_{i|j}^{(0)}\}_{j\in\mathcal{N}_i}$ are initialised with a sufficiently large variance.
2. The communication channels are securely encrypted when transmitting the initial values $\mathbf{z}^{(0)}$.
3. Each honest node i has at least one honest neighbour.
4. The stochastic updating probabilities are uniform with probability μ, so that $\mathbb{E}[\mathbf{U}^{(k)}] = \mu\mathbf{I}$.

In the case of adaptive differential PDMM, instead of broadcast PDMM, the privacy condition can be derived using the same steps as in [7]. The variance bound given in (6.7) is also applicable to this implementation. Thus, subspace based privacy preservation also works with stochastic adaptive differential PDMM, under the same assumptions as stated above.

7 Numerical Experiments

In this chapter the theoretical results from this report regarding stochastic PDMM are validated through numerical experiments. First, we will discuss the method and parameters used for the simulations in Section 7.1. Next, we will use stochastic PDMM to solve the following three distributed optimisation problems: distributed averaging consensus, distributed least squares (LS) minimisation and distributed ℓ_1-norm minimisation in Sections 7.2, 7.4 and 7.5, respectively. In Section 7.3, we present simulation results showing the difference between unicast and broadcast PDMM in the presence of transmission loss. The cost function used for these simulations is distributed averaging. In Section 7.6, we present an empirical result indicating the equivalence between the expected value of stochastic PDMM and averaged synchronous PDMM. Subsequently, in Section 7.7 we provide simulation results to indicate that the variance of $\mathbf{z}^{(k)}$ is bounded below and subspace based privacy is possible with stochastic PDMM. Finally, the main results are summarised in Section 7.8.

7.1. Simulation setup

All simulations in this report were performed using Matlab. First, all experiments were performed with an object-oriented, node based simulation. In this simulation, each node in the network is defined as its own object, that has its own locally available variables and a list of neighbours. Each node object is also equipped with a number of functions that can perform updates to its stored variables and transmit variables to neighbouring nodes. By implementing PDMM in this an object-oriented way, it is straightforward to define algorithm variations such as asynchronous updating schemes and transmission losses, whilst keeping a close resemblance to a real world distributed network. Furthermore, the equations used for the updating functions in the node objects are equivalent to those defined in the algorithms in Chapter 5.

Because the convergence proofs for stochastic PDMM are stated using the global matrix equations relating to the entire network, simulations were also performed using these matrix based updating equations. As mentioned in Section 5.6, the standard matrix equations for PDMM do not allow for broadcast implementations with transmission loss or inconsistent initialisation, which can both happen in real world deployments of PDMM. In the case where there are no inconsistencies between different versions of the same variable stored at separate nodes, the object-oriented and matrix based simulations are equivalent. This also indicates that the proof of convergence for stochastic PDMM holds for practical implementations of PDMM. We will not give separate results for each simulation method, because in most cases the results are identical.

For the simulations that are discussed in this section a random geometric graph (see [28]) was generated, consisting of $n = 30$ nodes, randomly placed within an area of 100 by 100 meters. An edge is created whenever the distance between two nodes is within the communication range, which is set at $r = 50$m. The resulting network is connected with high probability. This follows from [1, Lem. 9], where it is shown that $\mathbb{P}\{$"Graph connected"$\} \geq 1 - 1/n^2 = 99.9$%, if $nr^d \geq 2\,\text{ln}(n)$[1], which holds with the chosen values for our simulations. The resulting network is depicted in Figure 7.1.

[1]This equation holds for a unit cube of dimension d, so the value of r must be normalised if the area in question does a unit cube.

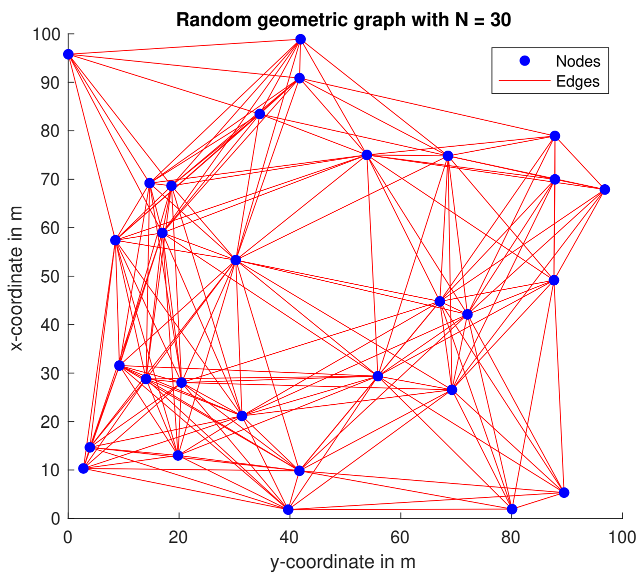

Figure 7.1: Random geometric graph that is used for simulations, $N = 30, r = 50$m.

Table 7.1: Simulation parameters.

Area	100×100m^2
Number of nodes, n	30
Wireless range of nodes, r	50m
PDMM parameter, ρ	0.4

Although the choice of PDMM parameter ρ does influence convergence speed, it was chosen to keep this value fixed at $\rho = 0.4$ for the simulations in this chapter. The optimum ρ is dependent on network topology and the problem at hand and it is not something that is further investigated in this chapter. Unless explicitly mentioned, we use standard PDMM without operator averaging, so $\theta = 1$. Furthermore, we assume consensus constraints as mentioned in Section 3.4.3 and we use $\mathbf{D} = \mathbf{C}^T\mathbf{C}$ to denote the degree matrix with diagonal entries d_i, corresponding to the degree of each node i. The common parameters that are used for all simulations are summarised in Table 7.1.

7.2. Averaging

Distributed averaging is a straightforward problem, where the nodes in the network each aim to calculate the average value of all nodes. The related cost function is:

$$f(\mathbf{x}) = \sum_{i\in\mathcal{V}} \left(\frac{1}{2}||\mathbf{x}_i - \mathbf{a}_i||^2\right).$$

By substituting this cost function into (3.6) and solving the minimisation we arrive at the following closed form solution

$$\mathbf{x}_i = \frac{a_i - \sum_{j\in\mathcal{N}_i} \mathbf{A}_{i|j}^T \mathbf{z}_{i|j}^{(k)}}{1 + \rho d_i}. \tag{7.1}$$

We perform distributed averaging using unicast PDMM. The values in the measurement vector $\mathbf{a} = [a_1, a_2, ..., a_n]^T$ are randomly assigned wit a variance $\sigma^2 = 400$ and a mean of 20. The maximum number of iterations is set to 5000. In Figure 7.2 it can be seen that the random initialisation of $\mathbf{z}^{(0)}$ does not effect the overall rate of convergence, except for a larger initial error. For this reason, in further simulations the z-vector is always initialised as $\mathbf{z}^{(0)} = \mathbf{0}$. In Figure 7.3 a comparison is made between different methods of node activation.

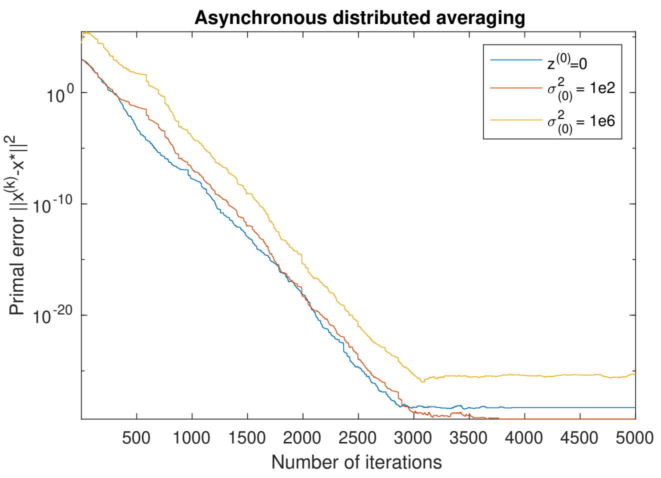

Figure 7.2: Convergence of distributed averaging with different levels initialisation variance using asynchronous PDMM.

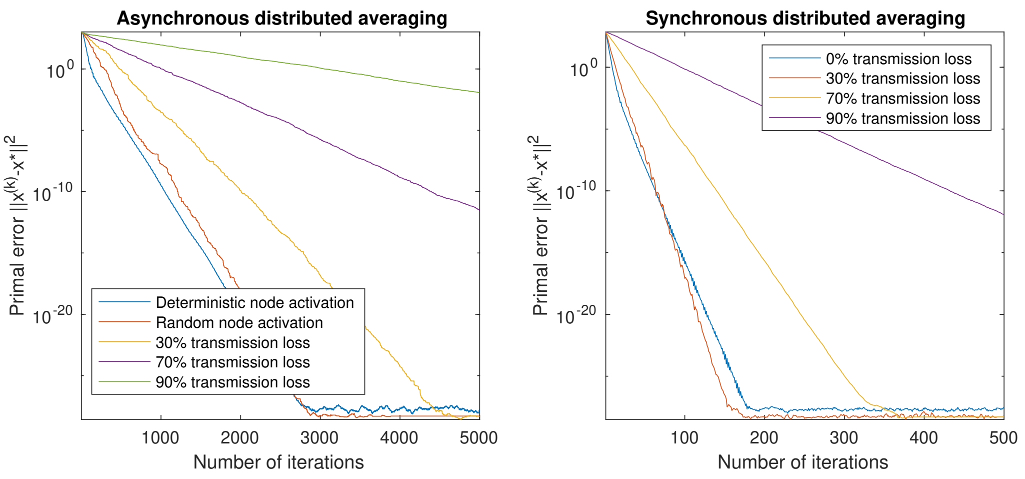

Figure 7.3: Convergence of distributed averaging with different levels of transmission loss. Left: asynchronous PDMM, Right: synchronous PDMM.

The following types of node activation are considered:

- Asynchronous deterministic node activation: $\mathcal{V}_a^{(K)} = \text{mod}(k-1, n) + 1$,
- Asynchronous random node activation: $\mathcal{V}_a^{(k)} = \text{randi}(n)$ (also referred to as *asynchronous*),
- Synchronous node activation: $\mathcal{V}_a^{(k)} = \mathcal{V}$.

Furthermore, for the asynchronous deterministic node activation and for the synchronous node activation, a number of different transmission error probabilities were simulated using unicast PDMM.

From Figure 7.3, we can conclude that synchronous PDMM converges within the least amount of iterations. Additionally, we can see that transmission loss does not prevent the convergence of unicast PDMM and it only slightly effects the convergence rate in both the asynchronous and synchronous case. Because asynchronous PDMM is more likely to be used in practical implementations and for ease of comparison, for the rest of the simulations we will use asynchronous node activation.

7.3. Broadcast with transmission losses

In Section 5.6, we indicated the problem that occurs with broadcast PDMM in the presence of transmission loss using a simple example. We also proposed an adaption based on acknowledge messages. Figure 7.4 shows the results for the 2-node averaging example mentioned in Section 5.6. From Figure 7.4 we can conclude that broadcast PDMM is not robust against transmission loss and the proposed

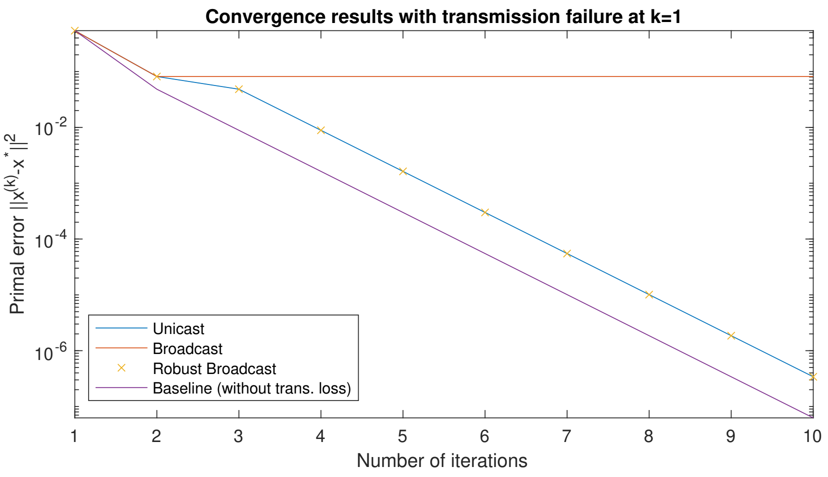

Figure 7.4: Convergence of 2-node distributed averaging, with a transmission loss at $k = 1$.

adaption is effective. It produces the same results as unicast PDMM in the presence of transmission losses. As mentioned in Section 5.6.1, this adaption is only effective if it can be guaranteed that all acknowledge messages are received properly.

7.4. Least squares

Distributed LS is a common optimisation problem, where the nodes in the network aim to calculate the least squares solution of an overdetermined linear system of equations based on measured data. Each node only has partial information of the system, thus collaboration is needed between the nodes in the network. The related cost function is:

$$f(\mathbf{x}) = \sum_{i\in\mathcal{V}} \left(\frac{1}{2}||\mathbf{y}_i - \mathbf{Q}_i\mathbf{x}_i||^2\right), \tag{7.2}$$

where $\mathbf{y}_i \in \mathbb{R}^{p_i}$ are decision vectors, $\mathbf{Q}_i \in \mathbb{R}^{p_i\times u}$ contains the input observations of node i, with $p_i > u$ because we assume an overdetermined system. After substituting (7.2) into (3.6), the minimisation has the following closed form solution

$$\mathbf{x}_i = (\mathbf{Q}_i^T\mathbf{Q}_i + \rho d_i\mathbf{I})^\dagger \left(\mathbf{Q}_i^T\mathbf{y}_i - \sum_{j\in\mathcal{N}_i} \mathbf{A}_{i|j}^T\mathbf{z}_{i|j}^{(k)}\right).$$

For our simulations we use $p_i = 5, \forall i \in \mathcal{V}$ and $u = 3$. The primal variable $\mathbf{x}_i \in \mathbb{R}^3$ is determined by $\mathbf{z}^{(0)}$ so it does not need to be initialised. Both $\mathbf{y}_i \in \mathbb{R}^5$ and $\mathbf{Q}_i \in \mathbb{R}^{5\times3}$ are randomly initialised following a normal distribution with zero-mean and unit variance. As mentioned in the previous section, the auxiliary variable is also initialised with all zeros. Figure 7.5 shows the convergence results for distributed LS with different transmission loss probabilities. From Figure 7.5 we can conclude that asynchronous unicast PDMM converges in the presence of transmission loss. The rate of convergence is slightly affected by the transmission loss probability.

7.5. ℓ_1 norm

In this section we investigate the effect of operator averaging in combination with stochastic PDMM. Using synchronous standard PDMM, there is no convergence guarantee for nondifferentiable cost functions and the convergence proof for stochastic standard PDMM also uses the additional assumptions of differentiability and strong convexity.

We consider the simple nondifferentiable objective

$$f(\mathbf{x}) = ||\mathbf{x} - \mathbf{a}||_1, \tag{7.3}$$

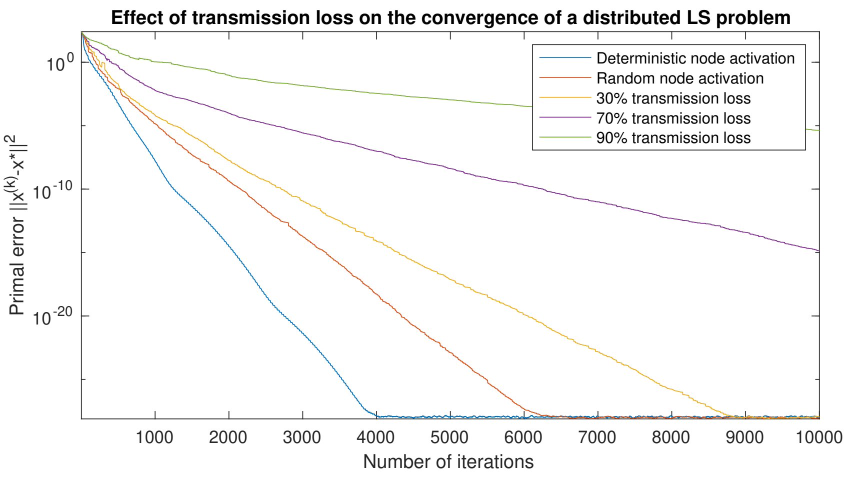

Figure 7.5: Convergence of distributed least squares with different levels of transmission loss using asynchronous PDMM.

where $\mathbf{x}$ is the vector of primal optimisation variables and $\mathbf{a}$ contains random measurements for each node. By substituting (7.3) into (3.6) and setting the derivative to $\mathbf{x}$ equal to zero, we find

$$\begin{aligned}
\text{sign}\left(\mathbf{x}^{(k+1)}-\mathbf{a}\right)+\mathbf{C}^T\mathbf{z}^{(k)}+\rho\mathbf{D}\mathbf{x}^{(k+1)} &= \mathbf{0}\\
\rho^{-1}\text{sign}\left(\mathbf{x}^{(k+1)}-\mathbf{a}\right)+\mathbf{D}\left(\mathbf{x}^{(k+1)}-\mathbf{a}\right) &= -\rho^{-1}\mathbf{C}^T\mathbf{z}^{(k)}-\mathbf{D}\mathbf{a}\\
\left(\rho^{-1}\text{sign}+\mathbf{I}\right)\left(\mathbf{D}(\mathbf{x}^{(k+1)}-\mathbf{a})\right) &= -\rho^{-1}\mathbf{C}^T\mathbf{z}^{(k)}-\mathbf{D}\mathbf{a}\\
\mathbf{x}^{(k+1)} &= \mathbf{D}^{-1}\mathcal{S}_{\rho^{-1}}\left(-\rho^{-1}\mathbf{C}^T\mathbf{z}^{(k)}-\mathbf{D}\mathbf{a}\right)+\mathbf{a},
\end{aligned}$$

where $\mathcal{S}_{\rho^{-1}}$ is the element-wise soft-thresholding operator defined as

$$(\mathcal{S}_c(\mathbf{x}))_i = \begin{cases} x_i - c, & x_i > c\\ 0, & |x_i| < c\\ x_i + c, & x_i < c.\end{cases}$$

The third line above follows from the fact that $\mathbf{D}$ is diagonal and positive definite, which is why $\text{sign}(\mathbf{x}) = \text{sign}(\mathbf{D}\mathbf{x})$ holds. For a single node the x-update can be calculated as

$$\mathbf{x}_i^{(k+1)} = d_i^{-1}\mathcal{S}_{\rho^{-1}}\left(-\rho^{-1}\left(\sum_{j\in\mathcal{N}_i}\mathbf{A}_{i|j}\mathbf{z}_{i|j}^{(k)}\right)-d_ia_i\right)+a_i.$$

Because the ℓ_1-norm is not strictly convex, the solution to this problem is in general not unique. Hence, the primal mean squared error is not a practical measure to visualise convergence. Instead the objective error $||f(\mathbf{x}^{(k)})-f(\mathbf{x}^*)||$ is used to analyse the convergence. Like in the previous simulations, the measurement vector $\mathbf{a}$ is randomly initialised and $\mathbf{z}^{(0)}$ is initialised with all zeros.

In Figure 7.6 we can see that without operator averaging, $\theta = 1$, asynchronous PDMM still seems to converge. This is an interesting result, because in the synchronous case PDMM without operator averaging does not converge. For all θ-averaging values between zero and one, the asynchronous algorithm also converges, but with different rates depending on the value of θ. As can be seen in Figure 7.6, lower θ-values lead to slower convergence rates.

To further examine the behaviour of stochastic standard PDMM for nondifferential cost functions, we performed Monte Carlo simulations with different random node activations. To decrease the simulation time we used a network of $n = 12$ nodes with $r = 40m$. Figure 7.7 shows the results of 100 Monte Carlo runs using random asynchronous PDMM. In Figure 7.7 it can be seen that all of the 100 Monte Carlo runs converge, indicating that it was probably not a coincidental result depending on the specific node selection realisation in the simulation of Figure 7.6. The red line in Figure 7.7 indicates the convergence trajectory of the squared error of the mean primal variable, where the mean is taken over the Monte Carlo runs.

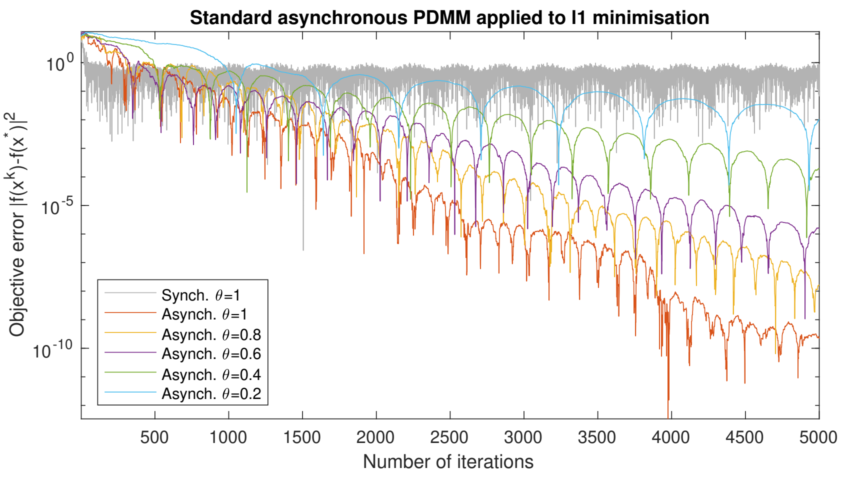

Figure 7.6: Convergence of ℓ_1 norm minimisation for different θ-averaging values.

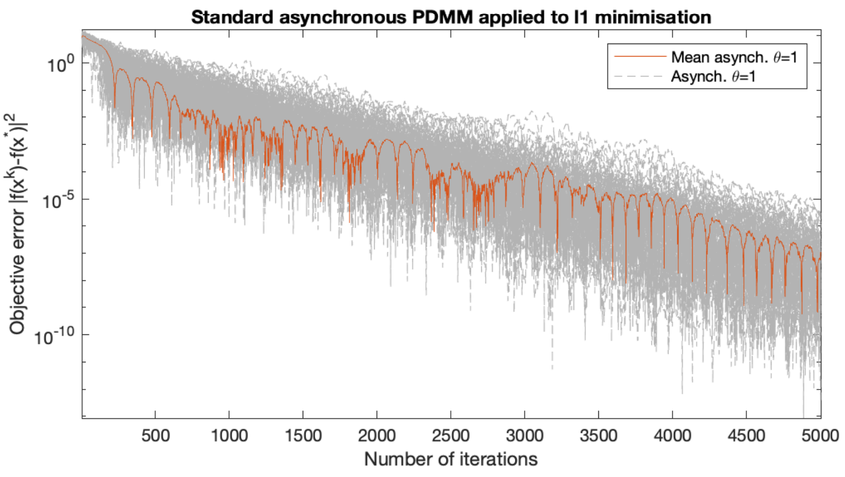

Figure 7.7: Convergence of ℓ_1 norm minimisation for 100 Monte Carlo runs using non-averaged asynchronous PDMM.

7.6. Operator averaging and stochastic PDMM

In this section we want to highlight an interesting interpretation of the stochastic Banach-Picard iteration. If we assume uniform updating probabilities, $\forall i \in \mathcal{V}, j \in \mathcal{N}_i,\ \mu_{i|j} = \mu$, the expected value of $\mathbf{z}^{(k+1)}$ is

$$\begin{aligned}\mathbb{E}\left[\mathbf{z}^{(k+1)}\right] &= \mathbb{E}\left[\left(\mathbf{I} - \mathbf{U}^{(k+1)}\right)\mathbf{z}^{(k)} + \mathbf{U}^{(k+1)}\mathbf{T}\mathbf{z}^{(k)}\right] \\ &= \left(\mathbf{I} - \bar{\mathbf{U}}\right)\mathbb{E}\left[\mathbf{z}^{(k)}\right] + \bar{\mathbf{U}}\mathbf{T}\left(\mathbb{E}\left[\mathbf{z}^{(k)}\right]\right) \\ &= (1-\mu)\,\mathbb{E}\left[\mathbf{z}^{(k)}\right] + \mu\mathbf{T}\left(\mathbb{E}\left[\mathbf{z}^{(k)}\right]\right)\end{aligned} \tag{7.4}$$

From this equation we can conclude that in the mean, a stochastic Banach-Picard iteration is equivalent to a Krasnosel'skii-Mann iteration (see Thm. 3.1) with averaging parameter μ.

This could partly support the interesting convergence results for non-averaged asynchronous PDMM that can be seen in Figures 7.6 and 7.7. In these simulations, asynchronous random node activation was used, with one active node per iteration. This corresponds to an updating probability of $\mu = 1/n = 1/12$ for each entry of $\mathbf{z}$. In Figure 7.8 a comparison is made between $1/12$-averaged synchronous PDMM and the mean of non-averaged asynchronous PDMM, using the same simulation parameters as in Section 7.5. The rates of convergence of these two cases are comparable. As the iterations increase the similarity between the mean stochastic convergence and the averaged synchronous convergence becomes slightly smaller. This could be due to the fact that the higher the number of iterations become, the higher the number of stochastic outcomes are possible. So for more

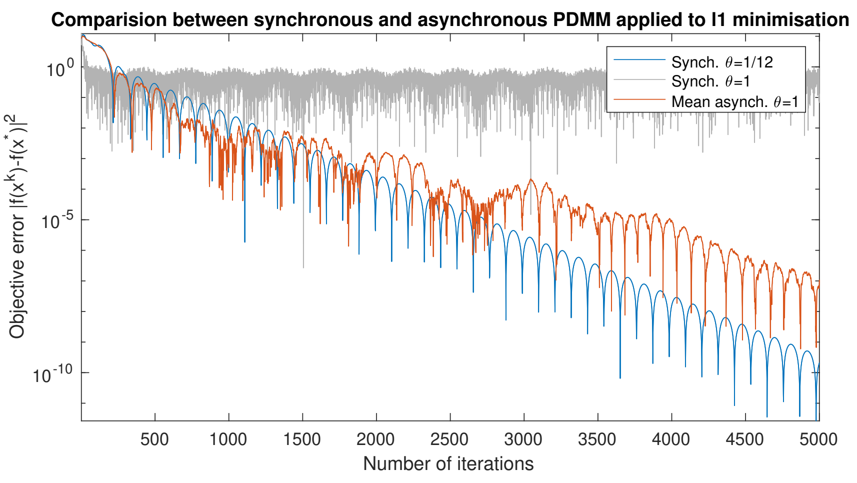

Figure 7.8: Comparison between the convergence of ℓ_1 norm minimisation using synchronous and asynchronous PDMM. The mean asynchronous data is calculated using 100 Monte Carlo runs.

statistical accuracy at higher iteration numbers, more Monte Carlo runs could be beneficial. Additionally, as mentioned in Section 7.5, the ℓ_1 minimisation problem does not have a unique minimum. Due to the randomness of the stochastic PDMM iterates, different primal optimal solutions are reached at each realisation and the mean of all these optimal solutions is not necessarily the solution found by using synchronous PDMM.

One should note that the results in this section are purely empirical and meaningful expressions for the variance of stochastic PDMM need to be derived to justify these results from a general theoretical perspective.

7.7. Privacy

To validate the variance bound that is derived in Chapter 6, we perform distributed averaging simulations using asynchronous broadcast PDMM. The asynchronous node activation is performed randomly using $\mathcal{V}_a^{(k)} = \text{randi}(n)$, which results one node being activated each iteration with a probability of $\mu = 1/N$ and $\mathbb{E}[\mathbf{U}^{(k)}] = \mu\mathbf{I}$. The values in the measurement vector $\mathbf{a} = [a_1, a_2, ..., a_n]^T$ are randomly assigned with a variance $\sigma_a^2 = 400$ and a mean of 20. The entries of $\mathbf{z}^{(0)}$ are randomly initialised with zero mean and variance $\sigma_{z^{(0)}}^2 = 100$. To analyse the variance of $\mathbf{z}^{(k)}$ we perform 50 Monte Carlo runs with different initialisations of $\mathbf{z}^{(0)}$, for consistency the sequence of selected nodes is kept constant for each Monte Carlo run. The maximum number of iterations in each run is set to 3000.

The left plot in Figure 7.9 shows that the variance of $\mathbf{z}_{\Psi}^{(k)}$ goes to zero as expected. The right plot in Figure 7.9 shows that the mean variance[2] of $\mathbf{z}_{\Psi^\perp}^{(k)}$ does not go to zero and stays above the derived bound. Thus subspace based privacy preservation is effective with asynchronous broadcast PDMM. The left plot in Figure 7.10 is added to give a slightly more detailed view of the mean variance compared to the derived bound. The fact that the mean variance does not go to zero does not necessarily ensure that the variance of each entry of $\mathbf{z}^{(k)}$ is nonzero. The right plot in Figure 7.10 shows that the variance of each entry of $\mathbf{z}^{(k)}$ is in fact nonzero after convergence.

7.8. Summary

Summarising, in this section we have provided numerical results to support the theoretical results from Chapters 4-6. Simulations were performed on a random geometric graph using variations PDMM variations. None of the numerical results contradict the theoretical results and as expected both asynchronous PDMM and PDMM with transmission losses converge if implemented using unicast transmissions. Interestingly, the simulation results for the simple ℓ_1 minimisation problem indicate that stochastic PDMM can converge when using nondifferentiable non-strongly convex cost functions, without the need for operator averaging. An interpretation of this result is given by looking at the expected value of

[2]Mean taken over the entries of the vector.

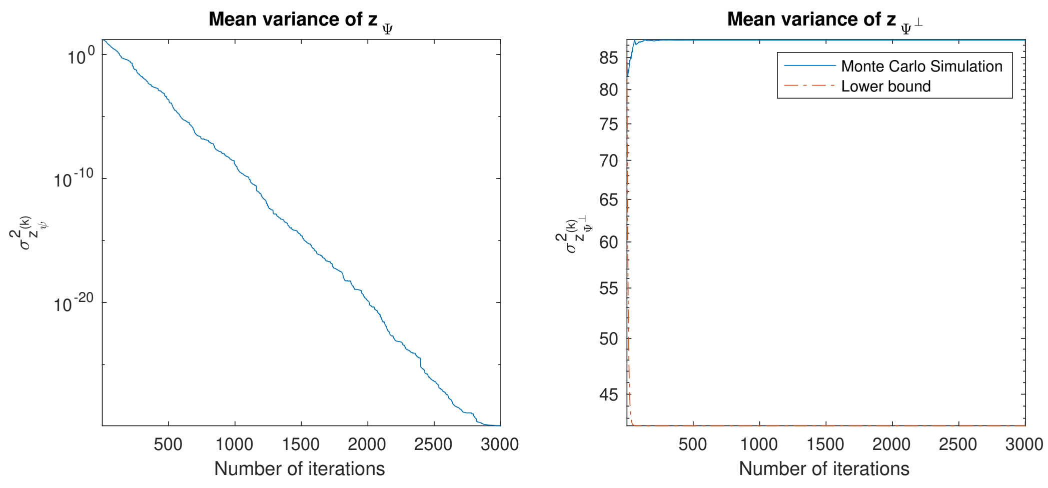

Figure 7.9: Variance results for asynchronous distributed averaging with subspace based privacy preservation. Left: Mean variance in subspace Ψ, Right: Mean variance in subspace $\Psi^{\perp}$.

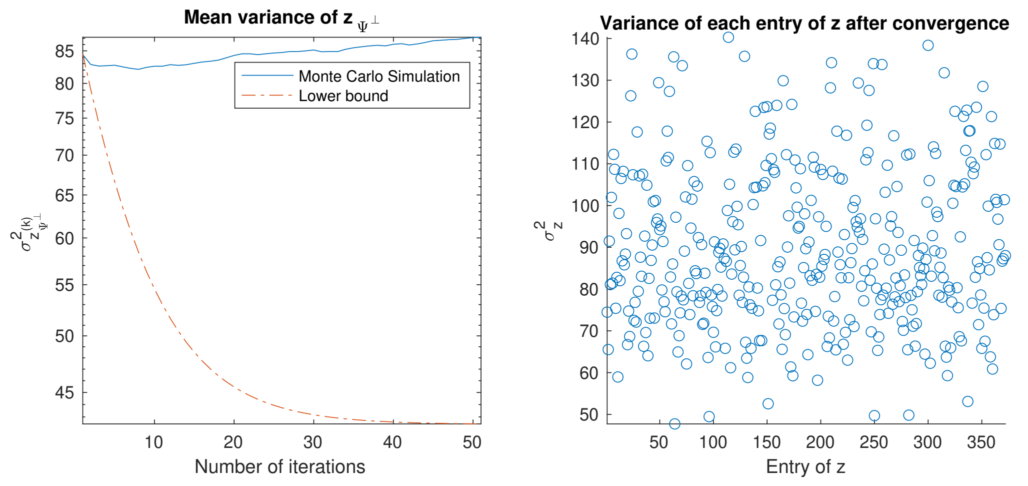

Figure 7.10: Variance results for asynchronous distributed averaging with subspace based privacy preservation. Left: Mean variance in subspace $\Psi^{\perp}$, Right: Variance of individual entries of **z** after convergence.

asynchronous PDMM and comparing this to $1/n$-averaged synchronous PDMM. Lastly, the variance of asynchronous PDMM is analysed by means of a distributed averaging simulation. As expected, the variance of $\mathbf{z}_{\Psi^{\perp}}$ always stays above the lower bound that was derived in Chapter 6.

8 Conclusions and Future Work

In this report the properties of stochastic PDMM are analysed. This is a general framework that can be used to model PDMM variations like asynchronous updating and PDMM with transmission losses. The conclusions of this work are summarised in the context of the research questions formulated in Chapter 1.

Can a proof be formulated for the convergence of stochastic PDMM?
As proved in Chapter 4, stochastic PDMM converges almost surely to a primal optimal solution. The only assumption required is that the updating probability of each entry of the auxiliary variable $\mathbf{z}$ is nonzero. The convergence proof for stochastic θ-averaged PDMM holds for arbitrary CCP cost functions and the convergence proof for stochastic standard PDMM holds for differentiable and strongly convex cost functions.

What is the difference between unicast and broadcast implementations and how does this effect convergence of stochastic PDMM?
In distributed PDMM implementations two locally stored versions of each variable $\mathbf{z}_{i|j}$ are present for all $i \in \mathcal{V}, j \in \mathcal{N}_i$. If these variables are initialised consistently and remain consistent throughout the iterations, unicast and broadcast PDMM are identical and can be described by the definition of stochastic PDMM. This is the case with asynchronous PDMM with consistent initialisation and perfect communication channels, where the convergence proof from Chapter 4 thus holds.

In the presence of transmission losses or inconsistent initialisation, only unicast PDMM can be described by the definition of stochastic PDMM. This is because unicast PDMM is inherently robust, due to sending the updated variables to neighbouring nodes at a convenient step in the algorithm. Because of this, any inconsistencies are simply overwritten at a future iteration and thus do not have a lasting effect on convergence. Thus, unicast PDMM converges in the presence of transmission loss.

Broadcast PDMM is not inherently robust and an inconsistency in the auxiliary variables affects the convergence of the algorithm. The definition of stochastic PDMM does not allow for two different values to be stored for an auxiliary variable and thus the convergence proof from Chapter 4 does not apply to this case. When inconsistently initialised or in the presence of transmission losses broadcast PDMM does not converge to a primal optimal solution. This same conclusion holds for differential PDMM.

Is subspace based privacy preservation effective in combination with stochastic PDMM implementations?
The privacy preservation proof given in [6] can not applied to stochastic PDMM because of the fact that the subspace $\mathbf{z}_{\Psi^\perp}$ is randomly affected by stochastic updating matrix $\mathbf{U}$. Under the assumption that the stochastic updating probabilities are uniform with probability μ, a lower bound for the variance of $\mathbf{z}_{\Psi^\perp}^{(k)}$ is derived in Chapter 6. This lower bound confirms that subspace based privacy preservation is usable for stochastic PDMM.

Main: What influence do stochastic updates and transmission losses have on the favourable properties of PDMM?
All in all, many of the properties that have previously been proved for synchronous PDMM are now also proved for stochastic PDMM. The most important theoretical results are:

- Stochastic PDMM converges almost surely if the updating probability of each entry of $\mathbf{z}$ is nonzero. The applicable cost functions are the same as in the synchronous case. Some consequences of this result are:
  - All consistent asynchronous PDMM algorithms converge for the same cost functions as synchronous PDMM.
  - Unicast PDMM converges in the presence of transmission losses. This result holds for synchronous and asynchronous updating schemes.
- Broadcast PDMM does not converge in the presence of transmission losses due to inconsistencies in stored values for the same auxiliary variable.
- Despite the fact that the privacy preserving subspace $\Psi^\perp$ is affected by random updating matrix $\mathbf{U}$, subspace based privacy preservation still works with stochastic PDMM, assuming uniform updating probabilities.
- In the mean, stochastic PDMM with uniform updating probability μ is equivalent to μ-averaged synchronous PDMM.

Lastly, we would like to highlight an important empirical result. The simulations performed during this research seem to indicate that some stochastic forms of PDMM, for example asynchronous PDMM, converge almost surely for arbitrary CCP functions. This holds without the need for extra operator averaging.

8.1. Future work

In addition to the results from this report that are summarised in the section above, we list a number of suggestions for potential further research. These suggestions are based on empirical observations and questions that arose during the work on this thesis.

Stochastic standard PDMM for arbitrary CCP cost functions

In Section 7.6 we presented an empirical result that suggests that stochastic standard PDMM converges almost surely for arbitrary CCP cost functions. To justify these results from a general theoretical perspective, meaningful expressions for the variance of stochastic PDMM need to be derived. It seems that some statistical independence is needed between the random variables $U_{i|j}$ for this result to hold true. Intuitively this makes sense, because in the extreme case that all $U_{i|j}$ are equal to the same random variable U, each stochastic update will either correspond to a synchronous update step or to an iteration in which nothing happens. The current convergence proofs in Chapter 4 do not make any assumptions regarding the independence of different $U_{i|j}$ variables, so a different approach may need to taken.

Privacy preservation

In Section 6.2 we proved that subspace based privacy preservation is still effective when using stochastic PDMM. We derived a lower bound for the variance of $\mathbf{z}^{(k)}$ under the assumption that the updating probabilities are uniform, so $\mathbb{E}[\mathbf{U}] = \mu\mathbf{I}$. This assumption limits the applicable cases of stochastic PDMM, so it would be useful to extend this result to be applicable to non-uniform updating probabilities.

Empirically it was found that for certain network topologies there can be zeros on the diagonal of the projection matrix $\mathbf{\Pi}_{\Psi^\perp}$. Let $\mathcal{E}_0$ denote the set of edges for which this happens, then $\sigma^2_{i|j} = 0, \forall(i,j) \in \mathcal{E}_0$, because the bound for the variance of $\mathbf{z}_{\Psi^\perp}$ contains a multiplication with $\mathbf{\Pi}_{\Psi^\perp}$, as can be seen in (6.7). As a result, the edges in $\mathcal{E}_0$ will not contribute to the privacy preservation. Some preliminary simulations seem to indicate that this problem is network dependent and nodes with a single neighbour cause this issue. Additionally, other edges forming linear chains with a single node stub also have this problem. It would be useful to derive the exact network conditions for which this problem occurs, so that it can be determined which nodes are kept private if the network topology is known.

It was also found that when randomly determining the averaging parameter θ at each node at every iteration, similar convergence results are attained as stochastic averaged PDMM. This could potentially be exploited by randomly selecting a $\theta_{i|j}$ at each node at each iteration and using this randomness to preserve privacy, without the need for broadcast or differential messaging. This would lead to an asynchronous, private, unicast version of PDMM, making it robust against transmission loss.

Dynamic network topologies

It would be interesting to analyse the working of PDMM in combination with dynamic network topologies. A possible method to analyse this is to define stochastic PDMM with an extended z-vector containing two entries per theoretically possible network edge and use the stochastic updating parameter $U_{i|j}$ to determine the network topology at each iteration. In [29] a sufficient condition for convergence of a distributed averaging consensus algorithm with a varying network topology is given. The sufficient condition is that the network must be connected in the mean. This includes cases where the network is not connected at each iteration. A convergence proof along these lines that applies to PDMM would generalise our results and include asynchronous PDMM, PDMM with transmission loss, PDMM over varying networks and PDMM over directed graphs.

This kind of general proof could also lead to a potential solution to the problems caused by transmission losses in combination with broadcast PDMM (see Chapter 5). Because not all edges are needed at each iteration, edges could be dynamically enabled/disabled depending on their respective signal strength. As long as the network remains connected in the mean, this algorithm should still converge. If implemented and functioning correctly, this method could hopefully disable an edge with low signal strength before a potential transmission loss occurs, thus preventing a mismatch in data between local and remote versions of the same variable. Of course, this method also adds some overhead, because of the signal strength monitoring. It also requires knowledge of the network to determine how many edges can be disabled whilst keeping the mean network connected.

PDMM extensions

The future work recommendations that have been mentioned so far are mainly based on results and observations related to this thesis. In this section, we list two potential extensions to PDMM that are not closely related to this work.

A useful extension to PDMM would be a PDMM based algorithm that works with time varying objective functions. In literature these kinds of algorithms are called online algorithms. Online PDMM could be useful in situations where a varying quantity needs to be tracked using a sensor network.

As mentioned in [30], inertial and relaxation methods for Krasnoselski-Mann iterations could be used to speed up convergence. It would be interesting to apply these kinds of methods to PDMM so PDMM can be used in situations where a high convergence speed is needed.

References

[1] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, “Randomized gossip algorithms,” en, *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2508–2530, Jun. 2006, ISSN: 0018-9448. DOI: 10.1109/TIT.2006.874516.

[2] A. D. G. Dimakis, A. D. Sarwate, and M. J. Wainwright, “Geographic Gossip: Efficient Averaging for Sensor Networks,” en, *IEEE Transactions on Signal Processing*, vol. 56, no. 3, pp. 1205–1216, 2008, ISSN: 1053-587X. DOI: 10.1109/TSP.2007.908946.

[3] N. Bof, R. Carli, G. Notarstefano, L. Schenato, and D. Varagnolo, “Multiagent Newton–Raphson Optimization Over Lossy Networks,” *IEEE Transactions on Automatic Control*, vol. 64, no. 7, pp. 2983–2990, Jul. 2019, ISSN: 0018-9286, 1558-2523, 2334-3303. DOI: 10.1109/TAC.2018.2874748.

[4] G. Zhang and R. Heusdens, “Distributed Optimization Using the Primal-Dual Method of Multipliers,” en, *IEEE Transactions on Signal and Information Processing over Networks*, vol. 4, no. 1, pp. 173–187, Mar. 2018, ISSN: 2373-776X, 2373-7778. DOI: 10.1109/TSIPN.2017.2672403.

[5] T. W. Sherson, R. Heusdens, and W. B. Kleijn, “Derivation and Analysis of the Primal-Dual Method of Multipliers Based on Monotone Operator Theory,” *IEEE Transactions on Signal and Information Processing over Networks*, vol. 5, no. 2, pp. 334–347, Jun. 2019, ISSN: 2373-776X, 2373-7778. DOI: 10.1109/TSIPN.2018.2876754.

[6] Q. Li, R. Heusdens, and M. G. Christensen, “Privacy-Preserving Distributed Optimization via Subspace Perturbation: A General Framework,” en, *IEEE Transactions on Signal Processing*, vol. 68, pp. 5983–5996, 2020, ISSN: 1053-587X, 1941-0476. DOI: 10.1109/TSP.2020.3029887.

[7] Q. Li, R. Heusdens, and M. G. Christensen, “Communication efficient privacy-preserving distributed optimization using adaptive differential quantization,” en, *Signal Processing*, vol. 194, p. 108 456, May 2022, ISSN: 01651684. DOI: 10.1016/j.sigpro.2022.108456.

[8] S. Boyd, “Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers,” en, *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2010, ISSN: 1935-8237, 1935-8245. DOI: 10.1561/2200000016.

[9] N. A. Trayanova, D. M. Popescu, and J. K. Shade, “Machine Learning in Arrhythmia and Electrophysiology,” en, *Circulation Research*, vol. 128, no. 4, pp. 544–566, Feb. 2021, ISSN: 0009-7330, 1524-4571. DOI: 10.1161/CIRCRESAHA.120.317872.

[10] V. Niranjani, V. Akshaya, V. Harish, and S. Abhishek, “Block Chaining in Finance and Accounting,” en, in *2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Coimbatore, India: IEEE, Mar. 2021, pp. 2034–2037, ISBN: 978-1-66540-520-1 978-1-66540-521-8. DOI: 10.1109/ICACCS51430.2021.9442064.

[11] G. Giaconi, D. Gunduz, and H. V. Poor, “Privacy-Aware Smart Metering: Progress and Challenges,” en, *IEEE Signal Processing Magazine*, vol. 35, no. 6, pp. 59–78, Nov. 2018, ISSN: 1053-5888, 1558-0792. DOI: 10.1109/MSP.2018.2841410.

[12] C. Gentry, “Fully homomorphic encryption using ideal lattices,” en, in *Proceedings of the 41st annual ACM symposium on Symposium on theory of computing - STOC ’09*, Bethesda, MD, USA: ACM Press, 2009, p. 169, ISBN: 978-1-60558-506-2. DOI: 10.1145/1536414.1536440.

[13] I. Damgård, V. Pastro, N. Smart, and S. Zakarias, “Multiparty Computation from Somewhat Homomorphic Encryption,” en, in *Advances in Cryptology – CRYPTO 2012*, R. Safavi-Naini and R. Canetti, Eds., vol. 7417, Series Title: Lecture Notes in Computer Science, Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 643–662, ISBN: 978-3-642-32008-8 978-3-642-32009-5. DOI: 10.1007/978-3-642-32009-5_38.

[14] C. Dwork, “Differential Privacy,” in *Automata, Languages and Programming : 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, ser. Lecture Notes in Computer Science 1611-3349, Berlin, Heidelberg : Springer Berlin Heidelberg, 2006, pp. 1–12, ISBN: 978-3-540-35907-4. DOI: 10.1007/11787006_1.

[15] S. Han, U. Topcu, and G. J. Pappas, “Differentially Private Distributed Constrained Optimization,” en, *IEEE Transactions on Automatic Control*, vol. 62, no. 1, pp. 50–64, Jan. 2017, ISSN: 0018-9286, 1558-2523. DOI: 10.1109/TAC.2016.2541298.

[16] M. T. Hale and M. Egerstedt, “Cloud-Enabled Differentially Private Multiagent Optimization With Constraints,” en, *IEEE Transactions on Control of Network Systems*, vol. 5, no. 4, pp. 1693–1706, Dec. 2018, ISSN: 2325-5870, 2372-2533. DOI: 10.1109/TCNS.2017.2751458.

[17] Q. Li, J. S. Gundersen, R. Heusdens, and M. G. Christensen, “Privacy-Preserving Distributed Processing: Metrics, Bounds and Algorithms,” en, *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 2090–2103, 2021, ISSN: 1556-6013, 1556-6021. DOI: 10.1109/TIFS.2021.3050064.

[18] H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*, en, ser. CMS Books in Mathematics. Cham: Springer International Publishing, 2017, ISBN: 978-3-319-48311-5. DOI: 10.1007/978-3-319-48311-5.

[19] J. Jacod and P. E. Protter, *Probability essentials*, English. Berlin; New York: Springer, 2004, OCLC: 780176454, ISBN: 978-3-642-55682-1.

[20] F. Iutzeler, P. Bianchi, P. Ciblat, and W. Hachem, “Asynchronous distributed optimization using a randomized alternating direction method of multipliers,” en, in *52nd IEEE Conference on Decision and Control*, Firenze: IEEE, Dec. 2013, pp. 3671–3676, ISBN: 978-1-4673-5717-3. DOI: 10.1109/CDC.2013.6760448.

[21] P. Bianchi, W. Hachem, and F. Iutzeler, “A Coordinate Descent Primal-Dual Algorithm and Application to Distributed Asynchronous Optimization,” en, *IEEE Transactions on Automatic Control*, vol. 61, no. 10, pp. 2947–2957, Oct. 2016, ISSN: 0018-9286, 1558-2523. DOI: 10.1109/TAC.2015.2512043.

[22] P. L. Combettes and J.-C. Pesquet, “Stochastic Quasi-Fejér Block-Coordinate Fixed Point Iterations with Random Sweeping,” en, *SIAM Journal on Optimization*, vol. 25, no. 2, pp. 1221–1248, Jan. 2015, ISSN: 1052-6234, 1095-7189. DOI: 10.1137/140971233.

[23] N. Bastianello, R. Carli, L. Schenato, and M. Todescato, “Asynchronous Distributed Optimization Over Lossy Networks via Relaxed ADMM: Stability and Linear Convergence,” en, *IEEE Transactions on Automatic Control*, vol. 66, no. 6, pp. 2620–2635, Jun. 2021, ISSN: 0018-9286, 1558-2523, 2334-3303. DOI: 10.1109/TAC.2020.3011358.

[24] E. K. Ryu and S. Boyd, “A PRIMER ON MONOTONE OPERATOR METHODS,” en, *APPL. COMPUT. MATH.*, p. 41, 2016. [Online]. Available: https://web.stanford.edu/~boyd/papers/pdf/monotone_primer.pdf.

[25] J. A. Jonkman, T. Sherson, and R. Heusdens, “Quantisation Effects in Distributed Optimisation,” en, in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Calgary, AB: IEEE, Apr. 2018, pp. 3649–3653, ISBN: 978-1-5386-4658-8. DOI: 10.1109/ICASSP.2018.8461782.

[26] T. Sherson, R. Heusdens, and W. B. Kleijn, “Derivation and Analysis of the Primal-Dual Method of Multipliers Based on Monotone Operator Theory,” en, *arXiv:1706.02654 [math]*, Nov. 2017, arXiv: 1706.02654. [Online]. Available: http://arxiv.org/abs/1706.02654.

[27] V. Cerf and R. Kahn, “A Protocol for Packet Network Intercommunication,” en, *IEEE Transactions on Communications*, vol. 22, no. 5, pp. 637–648, May 1974, ISSN: 0096-2244. DOI: 10.1109/TCOM.1974.1092259.

[28] J. Dall and M. Christensen, “Random geometric graphs,” en, *Physical Review E*, vol. 66, no. 1, p. 016 121, Jul. 2002, ISSN: 1063-651X, 1095-3787. DOI: 10.1103/PhysRevE.66.016121.

[29] S. Kar and J. Moura, “Distributed Consensus Algorithms in Sensor Networks: Quantized Data and Random Link Failures,” en, *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1383–1400, Mar. 2010, ISSN: 1053-587X, 1941-0476. DOI: 10.1109/TSP.2009.2036046.

[30] Q.-L. Dong, Y. J. Cho, S. He, P. M. Pardalos, and T. M. Rassias, *The Krasnosel'skiĭ-Mann Iterative Method: Recent Progress and Applications*, en, ser. SpringerBriefs in Optimization. Cham: Springer International Publishing, 2022, ISBN: 978-3-030-91653-4 978-3-030-91654-1. DOI: 10.1007/978-3-030-91654-1.

A Monotone operator theory

In this appendix a list of useful definitions and properties regarding monotone operator theory are given. We would like to refer to [18] and [24] for a more exhaustive explanation of monotone operators.

Definition A.1. *A **relation, mapping or operator*** $\mathbf{T}$ *on* $\mathbb{R}^n$ *is a subset of* $\mathbb{R}^n \times \mathbb{R}^n$*, defined as*

$$\mathbf{T} = \{(\mathbf{x}, \mathbf{y}) \,|\, \mathbf{x}, \mathbf{y} \in \mathbb{R}^n\}\,.$$

Definition A.2. *We have* $\mathbf{T}(\mathbf{x}) = \{\mathbf{y} \in \mathbb{R}^n \,|\, (\mathbf{x}, \mathbf{y}) \in \mathbf{T}\}$ *and we denote the **range** of* $\mathbf{T}$ *as* $\operatorname{ran}\mathbf{T} = \{\mathbf{y} \in \mathbb{R}^n \,|\, \exists \mathbf{x} \in \mathbb{R}^n : (\mathbf{x}, \mathbf{y}) \in \mathbf{T}\}$*, and the **domain** of* $\mathbf{T}$ *as* $\operatorname{dom}\mathbf{T} = \{\mathbf{x} \in \mathbb{R}^n \,|\, \mathbf{T}(\mathbf{x}) \neq \varnothing\}\,.$

Definition A.3. *The **identity relation** is defined as* $\mathbf{I} = \{(\mathbf{x}, \mathbf{x}) \,|\, \mathbf{x} \in \mathbb{R}^n\}$.

Definition A.4. *The **subdifferential relation** is defined as*

$$\partial f = \left\{(\mathbf{x}, \mathbf{g}) \,|\, \mathbf{x} \in \mathbb{R}^n, \forall \mathbf{y} \in \mathbb{R}^n \; f(\mathbf{y}) \geq f(\mathbf{x}) + \mathbf{g}^T(\mathbf{y} - \mathbf{x})\right\}. \tag{A.1}$$

Definition A.5. *If* $\mathbf{T}(\mathbf{x})$ *is a singleton or empty for any* $\mathbf{x}$*, then* $\mathbf{T}$ *is called a **function** or **single-valued**.*

Definition A.6. *The **sum** of two operators,* $\mathbf{A}$ *and* $\mathbf{B}$*, is defined as* $\mathbf{A}+\mathbf{B} = \{(\mathbf{x}, \mathbf{y}+\mathbf{z}) \,|\, (\mathbf{x}, \mathbf{y}) \in \mathbf{A}, (\mathbf{x}, \mathbf{z}) \in \mathbf{B}\}$.

Definition A.7. *The **composition** of two operators,* $\mathbf{A}$ *and* $\mathbf{B}$*, is defined as*

$$\mathbf{A} \circ \mathbf{B} = \{(\mathbf{x}, \mathbf{z}) \,|\, \exists \mathbf{y} \in \mathbb{R}^n : (\mathbf{x}, \mathbf{y}) \in \mathbf{B}, (\mathbf{y}, \mathbf{z}) \in \mathbf{A}\}\,. \tag{A.2}$$

Definition A.8. *The **inverse** of an operator* $\mathbf{T}$*, is defined as* $\mathbf{T}^{-1} = \{(\mathbf{x}, \mathbf{y}) \,|\, (\mathbf{y}, \mathbf{x}) \in \mathbf{T}\}$.

Definition A.9. *An operator* $\mathbf{T}$ *is **Lipschitz continuous** with constant* $L \geq 0$ *if we have* $||\mathbf{T}(\mathbf{y}) - \mathbf{T}(\mathbf{x})|| \leq L||\mathbf{y} - \mathbf{x}||$*, for all* $\mathbf{x}, \mathbf{y} \in \operatorname{dom}\mathbf{T}$.

- If $L \leq 1$, $\mathbf{T}$ is called *nonexpansive*.
- If $L < 1$, $\mathbf{T}$ is called a *contraction*.

Definition A.10. *The **fixed point set** of operator* $\mathbf{T}$ *is given by* $\operatorname{fix}(\mathbf{T}) = \{\mathbf{x} \in \mathbb{R}^n \,|\, \mathbf{x} = \mathbf{T}(\mathbf{x})\}$.

Definition A.11. *The **Banach-Picard iteration** is an algorithm that can be used to find a point in* $\operatorname{fix}(\mathbf{T})$*, defined as* $\mathbf{x}^{(k+1)} = \mathbf{T}\left(\mathbf{x}^{(k)}\right)$*, where* $\mathbf{x}^{(0)} \in \mathbb{R}^n$ *is some starting point.*

Definition A.12. *Let* $\mathbf{T}$ *be a nonexpansive operator and let* $\theta \in (0, 1)$*. Then* $\mathbf{T}$ *is **averaged with constant*** θ*, or **θ-averaged**, if there exists a nonexpansive operator* $\mathbf{R}$ *such that* $\mathbf{T} = (1-\theta)\mathbf{I} + \theta\mathbf{R}$.

Definition A.13. *An operator* $\mathbf{T}$ *is called **monotone** if,* $(\mathbf{u} - \mathbf{v})^T(\mathbf{x} - \mathbf{y}) \geq 0$*, for all* $(\mathbf{x}, \mathbf{u}), (\mathbf{y}, \mathbf{v}) \in \mathbf{T}$.

Definition A.14. *An operator* $\mathbf{T}$ *is called **strongly monotone** or coercive with parameter* $m > 0$ *if,* $(\mathbf{T}(\mathbf{x}) - \mathbf{T}(\mathbf{y}))^T(\mathbf{x} - \mathbf{y}) \geq m||\mathbf{x} - \mathbf{y}||^2$*, for all* $\mathbf{x}, \mathbf{y} \in \operatorname{dom}\mathbf{T}$.

Definition A.15. *An operator* $\mathbf{T}$ *is called* ***β-cocoercive*** *with parameter* $\beta > 0$ *if* $(\mathbf{T}(\mathbf{x}) - \mathbf{T}(\mathbf{y}))^T(\mathbf{x} - \mathbf{y}) \geq \beta||\mathbf{T}(\mathbf{x}) - \mathbf{T}(\mathbf{y})||^2$*, for all* $\mathbf{x}, \mathbf{y} \in$ *dom* $\mathbf{T}$.

Definition A.16. *When* $\beta = 1$*, cocoercivity implies that* $\left(\mathbf{T}(\mathbf{x}) - \mathbf{T}(\mathbf{y})\right)^T(\mathbf{y} - \mathbf{x}) \geq ||\mathbf{T}(\mathbf{y}) - \mathbf{T}(\mathbf{x})||^2$*, for all* $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$*, and the operator is called* ***firmly nonexpansive****.*

Definition A.17. *The* ***resolvent*** *of an operator* $\mathbf{T}$ *is defined as* $\mathbf{J}_{\rho\mathbf{T}} = (\mathbf{I} + \rho\mathbf{T})^{-1}$*, where* $\rho \in \mathbb{R}$*.*

Definition A.18. *The* ***Cayley operator****, reflection operator, or reflected resolvent of an operator* $\mathbf{T}$ *is defined as* $\mathbf{C}_{\rho\mathbf{T}} = 2\mathbf{J}_{\rho\mathbf{T}} - \mathbf{I}$*.*

Definition A.19. *An function* f *is* ***strongly convex*** *with parameter* m *if* $f(\mathbf{x}) - m||\mathbf{x}||^2$ *is convex, or equivalently if* ∂f *is* m*-strongly monotone.*

Definition A.20. ***Peaceman-Rachford splitting*** *is an operator splitting method that aims to find* $\mathbf{x}^*$ *such that* $\mathbf{0} \in \mathbf{T}_1(\mathbf{x}^*) + \mathbf{T}_2(\mathbf{x}^*)$*, by finding a fixed point in the following iteration sequence*

$$\mathbf{z}^{(k+1)} = \mathbf{C}_{\rho\mathbf{T}_2} \circ \mathbf{C}_{\rho\mathbf{T}_1}(\mathbf{z}^{(k)}), \mathbf{x}^{(k)} = \mathbf{J}_{\rho\mathbf{T}_1}(\mathbf{z}^{(k)}). \tag{A.3}$$

(see [18, Chapter 26] for more information)

B
SITB Poster Submission

The poster included on the following page was presented at the 42nd WIC Symposium on Information Theory and Signal Processing in the Benelux (SITB 2022). This symposium took place in Louvain-la-Neuve on 1-2 June. The poster includes some preliminary results that were obtained during the work on this thesis.

Convergence of Stochastic PDMM

Sebastian Jordan[*], Richard Heusdens[*†]

[*]TU Delft, Circuits and Systems Group, [†]Netherlands Defence Academy

Email: `s.o.jordan@student.tudelft.nl`

Introduction

In this work, we analyse a stochastic version of the primal-dual method of multipliers (PDMM), which is a promising algorithm in the field of distributed optimisation. So far, its convergence has been proven for synchronous implementations of the algorithm [1],[2]. We prove the convergence of stochastic PDMM. This is a general framework that can model many PDMM variations including asynchronous updating schemes and transmission loss. Furthermore, we present an interesting simulation observation regarding the need for operator averaging.

The Primal-Dual Method of Multipliers

- We consider a network consisting of a set of nodes, $\mathcal{V}$, connected by a set of edges, $\mathcal{E}$.
- Problem: minimise a convex closed and proper (CCP) separable cost function $f(\mathbf{x})$ subject to edge constraints:

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

- As derived in [1], the update equations for PDMM are

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

- A complete PDMM update can be seen as the application of an operator $\mathbf{T}_{\text{PDMM}}$ to $\mathbf{z}$.

Stochastic PDMM Convergence Proof

- $\mathbf{U}^{(k)} \in \mathbb{R}^{M_\mathcal{E} \times M_\mathcal{E}}$ is a random diagonal selection matrix with entries based on $U_{i|j} \in \{0,1\}$. It determines which $\mathbf{z}_{i|j}$ entries are updated. Assumption: $\mathbb{E}[\mathbf{U}] \succ \mathbf{0}$.
- Stochastic PDMM is defined as:

$$\mathbf{z}^{(k+1)} = \left(\mathbf{I} - \mathbf{U}^{(k+1)} \right) \mathbf{z}^{(k)} + \mathbf{U}^{(k+1)} \mathbf{T}_{\text{PDMM}}\left(\mathbf{z}^{(k)} \right).$$

- Asynchronous PDMM and PDMM with transmission loss are both specific versions of stochastic PDMM.
- Proof follows similar steps to the ones taken in [3] and builds upon a previous unfinished proof from [4].
- Stochastic sequence $\mathbb{E}\left[||\mathbf{z}^{k+1} - \mathbf{z}^*||^2_{\bar{\mathbf{U}}^{-1}} \middle| \mathcal{F}^{(k)} \right]$ is a nonnegative supermartingale.
- Almost sure convergence to a value supported by $\text{fix}\left(\mathbf{T}_{\text{PDMM}}\right)$. For stochastic averaged PDMM or stochastic non-averaged PDMM with strongly convex and differentiable cost functions.

Simulation results

- Non-differentiable cost function $f(\mathbf{x}) = ||\mathbf{x} - \mathbf{a}||_1$.
- Synchronous non-averaged PDMM does not converge, see Figure 1.
- Asynchronous non-averaged PDMM and synchronous $1/N$-averaged PDMM both converge with a similar convergence rate, see Figure 1.

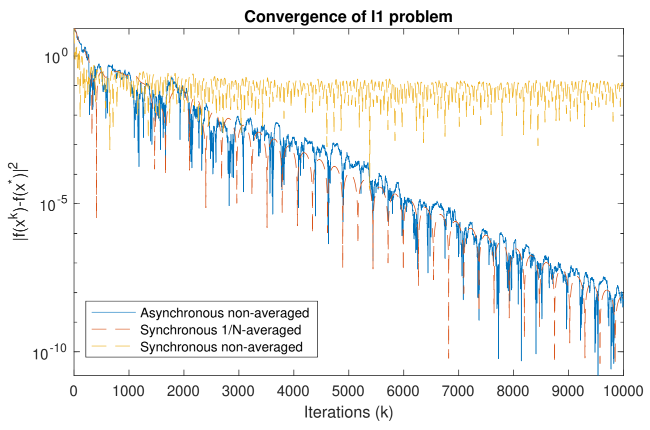

Figure 1: Convergence results for simple $\ell1$-problem in a random geometric network with 12 nodes.

Conclusions

- Stochastic θ-averaged PDMM converges almost surely for arbitrary CCP cost functions.
- Stochastic non-averaged PDMM converges almost surely for differential and strongly convex CCP cost functions.
- It is observed in simulations that asynchronous PDMM, contrary to synchronous PDMM, does not seem to require additional operator averaging to converge for arbitrary CCP cost functions.

References

[1] T. W. Sherson, R. Heusdens, and W. B. Kleijn, "Derivation and Analysis of the Primal-Dual Method of Multipliers Based on Monotone Operator Theory," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 5, no. 2, pp. 334–347, Jun. 2019.

[2] G. Zhang and R. Heusdens, "Distributed Optimization Using the Primal-Dual Method of Multipliers," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 4, no. 1, pp. 173–187, Mar. 2018.

[3] P. Bianchi, W. Hachem, and F. Iutzeler, "A Coordinate Descent Primal-Dual Algorithm and Application to Distributed Asynchronous Optimization," *IEEE Transactions on Automatic Control*, vol. 61, no. 10, pp. 2947–2957, Oct. 2016.

[4] T. Sherson, R. Heusdens, and W. B. Kleijn, "Derivation and Analysis of the Primal-Dual Method of Multipliers Based on Monotone Operator Theory," *arXiv:1706.02654 [math]*, Nov. 2017, arXiv: 1706.02654.