University of Groningen
Faculty of Mathematics and Natural Sciences

Master's Thesis in Applied Mathematics

Distributed Steepest Descent: A Method for Distributed Linear Least Squares Problems

Author:
K. van Geffen

Supervisor:
prof. dr. M.K. Camlibel

Second assessor:
dr. ir. F.W. Wubs

Abstract

The need for distributed processing arises naturally in for example wireless sensor networks, smart grids and some filter applications. All the computations are done by autonomous agents, where the communication among agents is restricted by the network topology. In this context, local measurements define, together, a global linear least square problem. Some of the existing methods are characterized by a two-step update of (i) a local minimization and (ii) a communication step. These methods are preferred for the computationally cheap updates, the scalability in network size and the minimal amount of data exchange. In this thesis we focus on two such methods, namely the decentralized gradient descent (`DGD`) method, which is a gradient descent based method, and the kernel projection (`KP`) method, which is based on local projection. We establish a link between the two widely different methods and propose two intermediate methods, namely an α- and β-variant of the distributed steepest descent (`DSD`) method. To this end, we introduce a steepest descent method for underdetermined system (`SDud`) and establish linear convergence results. When executed until convergence, we show that `SDud` essentially solves the local projection problem for `KP`. Additionally, we show that an application of the conjugate gradient method, `CGud`, computes this projection more efficiently. Finally, we consider numerical experiments and observe that `DSD` can be competitive with `DGD` and `KP`.

Acknowledgements

I would like to thank my supervisor, prof. dr. M.K. Camlibel, for his guidance in this project, for encouraging me to explore new ideas and finally, for providing me with new insights along the way. I enjoyed collaborating with my supervisor and I am looking forward to do so in the continuation of my academic career.

Contents

1 Introduction

In this thesis we consider multi-agent networks with $M > 1$ agents in the context of distributed linear least squares problems $Ax = b$. Each agent is assumed to know one or more rows of the partitioned matrix $[A\ b]$, denoted by $[A_i\ b_i]$. This problem can be formulated as a distributed convex consensus optimization problem, i.e. the problem we consider is that of M agents collaboratively solving the consensus optimization problem as follows:

$$\text{minimize} \quad f(x) = \sum_{i=1}^{M} f_i(x),$$

where f_i is a local convex function only known to some agent i. The distributed least squares problem is observed to be a special case of this problem by defining the local residual functions $f_i(x) = \frac{1}{2}\|b_i - A_i x\|_2^2$.

The communication among agents is restricted by the network topology, this is characterized by a graph $\mathcal{G}$ with M vertices and a set of edges characterizing the neighbor relations of the network. For example, for undirected networks the edge (i, j) is in the edge set just in case that agent i and agents j are able to communicate with each other. The data exchange is thus restricted by the network topology.

The methods that we consider involve two-step updates, with (i) a local minimization step and (ii) a communication step. Typically, each agent holds a state $x^i_{(t)}$ as a current approximation to the global minimizer x^* of the consensus optimization problem, at each iteration t. Then the local minimization steps (i) are for example based on the traditional gradient descent iteration as follows:

$$y^i_{(t)} = x^i_{(t)} - \alpha^i_{(t)} \nabla f_i(x^i_{(t)}),$$

where $\alpha^i_{(t)} \in \mathbb{R}$ is a suitable constant. Next, the communication step (ii) is typically characterized by an exchange of intermediate states $y^i_{(t)}$ among agents, followed by a weighted averaging of the intermediate states:

$$x_{(t+1)} = \sum_{j \in \mathcal{N}_i^*} w_{ij} y^i_{(t)},$$

where $\mathcal{N}_i^*$ is the index set of agents neighboring agent i, containing the agent i itself as well. By definition of weighted averaging, we have $w_{ij} \geq 0$ and $\sum_{j \in \mathcal{N}_i^*} w_{ij} = 1$.

The methods that we consider have the advantage of computationally cheap iterations, good scalability in the network size and a reasonable amount of data

exchange. If necessary, this approach ensures that the local data $[A_i \ b_i]$ remains protected, since it is not shared with other agents. Alternatively, we could allow agents to synchronize a common state $x_{(t)}$ and share the gradients $\nabla f_i(x_{(t)})$ to obtain the global gradient $\nabla f(x_{(t)}) = \sum_{i=1}^{M} \nabla f_i(x_{(y)})$. Then each agent can perform, in principle, the same global gradient descent update. However, the process of obtaining $\nabla f(x_{(t)})$ is another iterative process and requires that each agents knows either the exact value of M or at least an upper bound. This process clearly requires a lot of data exchange at each iteration. However, the two-step approach is likely to require significantly more iterations and we can not guarantee that the amount of data exchange is less for this approach. Nonetheless, this approach has the advantage of robustness for individual agent failures, which is commonly a preferred property in practical applications.

Unfortunately these methods suffer from an inherent problem, namely that the local minimization step (i) never takes global information into account. Since local and global minimizers generally do not coincide, we can see that the global minimizer x^* is generally not an equilibrium point of such two-step updates. Due to this, the convergence results that are obtained for such methods is generally a trade-off between speed of convergence and desired accuracy. Hence, such methods are intended for problems that do not require a very high accuracy, in contrast to linear least squares problems that are generally considered in the field of numerical mathematics.

1.1 Background

According to [1, (p. 3)], the study on truly distributed optimization can be traced back to early work in the 1980s. This is the type of distributed optimization that operates without a fusion center, in which data could be collected, combined and distributed. We say that the optimization is decentralized. In contrast, the parallel computing community generally considers centralized distributed problems with such a fusion center.

The need for truly distributed processing arises naturally from applications, such as in sensor networking [2, 3, 4, 5], some filter applications [6], smart grids [7] and cognitive radio networks [8, 9]. In these applications, the computations are typically executed by processors on board of sensors, which are physically separated from each other. The challenges in these applications are numerous. For example, the sensor network can be large scale and dynamically changing. Furthermore, the sensors typically have a limited communication reach, limited energy resources, limited computational capacity and can sometimes be prone to failure due to hostile environments [4]. These challenges ask for methods that take into account the following: Scalability to the network size, robustness to dynamical network topologies and the need for computationally cheap updates with a modest amount of data exchange (which is considered to be the most energy consuming operation in decentralized distributed processing). Additionally, sometimes privacy preservation is required in data-sensitive applications, as is described in [1].

Especially sensor networks have received significant attention in the recent years because of their huge potential in applications [2]. The challenges they present in the processing of signals is illustrated with an example [1, (p. 3-4)]. The example we consider is a problem in cognitive radio networks, called

spectrum sensing. This problem aims at detecting unused spectrum bands, such that they can be used by cognitive radios. In this context, the vector $x \in \mathbb{R}^n$ contains elements that correspond to the magnitude of the corresponding channel. Then each cognitive radio does a time-domain measurement of x satisfying $b_i = F^{-1}G_i x + \eta_i$, where b_i is measured and F^{-1} and G_i are problem specific matrices [1]. The term η_i is a zero-mean stochastic variable that models the measurement noise. If all cognitive radios reach consensus about x for their local measurements, then they can pick the unused spectrum from this consensus variable x. Now, in a more general context, we imagine that we consider local measurements of the form:

$$b_i = A_i x + \eta_i, \tag{1.1}$$

for some problem specific matrix A_i. This is also the traditional model for measurements in distributed estimation [10]. This example illustrates how sensor networks may have to deal with distributed linear least square problems of the form $Ax = b$, where the partitioned matrix $[A\ b]$ is distributed row-wise by the subset of rows $[A_i\ b_i]$. It is precisely in this context that we consider distributed linear least square problems.

1.2 Notation and conventions

- Vectors and matrices are denoted by respectively lower- and upper case symbols, e.g. $x \in \mathbb{R}^n$ and $A \in \mathbb{R}^{n\times n}$. Depending on the context, we may use either lower- and upper case symbols to denote constants, e.g. $n, M \in \mathbb{R}$. Matrices can also be represented as the collection of its entries, for which we will use the shorthand notation $A = [a_{ij}]$. Similarly, a vector $x \in \mathbb{R}^n$ may be a collection of M stacked vectors $x^i \in \mathbb{R}^{n_i}$ for $i = 1, \ldots, M$, with $\sum_{i=1}^{M} n_i = n$. For this we will use the shorthand notation $x = [x^i]$; it should be clear that we mean by this that:
$$x = \begin{bmatrix} x^1 \\ \vdots \\ x^M \end{bmatrix}.$$

- In a multi-agent network, we generally denote by M the number of agents in the network; naturally agents are numbered as $1, \ldots, M$. Vectors such as $x^i \in \mathbb{R}^{n_i}$ may come with a superscript i, to denote that it is a state belonging to agent i. Individual states are allowed to change in discrete time $t = 0, 1, 2, \ldots$ due to e.g. an iterative process. Hence, vectors such as $x^i_{(t)} \in \mathbb{R}^{n_i}$ may come with an additional subscript (t), to denote that it is the state of agent i at iteration t. In contrast, an individual function $f_i : \mathbb{R}^{n_i} \to \mathbb{R}$ belonging to some agent i is denoted with a subscript i; this way we may represent function specific constants such as $L_{f_i} \in \mathbb{R}$ more conveniently.

- Corresponding to some function $f : \mathbb{R}^n \to \mathbb{R}$, there may be a set of optimal solutions $\mathcal{X}^* \subseteq \mathbb{R}^n$, such as for example a set of minimizers. We mention that subscripts i may be used accordingly in this context. Assuming that $\mathcal{X}^* \neq \emptyset$ we denote elements in this set by $x^* \in \mathcal{X}^*$. Given any vector $x \in \mathbb{R}^n$, we denote by $\operatorname{dist}(x, \mathcal{X}^*)$ the Euclidean distance of x from the set $\mathcal{X}^*$, i.e. we have:
$$\operatorname{dist}(x, \mathcal{X}^*) = \inf\{\|x - x^*\|_2 \mid x^* \in \mathcal{X}^*\}.$$

If the minimum is attained for some $x_P^* \in \mathcal{X}^*$, then we say that x_P^* is the *projection* of x onto $\mathcal{X}^*$. Hence, x_P^* is a projection of x onto $\mathcal{X}^*$ if and only if $\text{dist}(x, \mathcal{X}^*) = \|x - x_P^*\|_2$.

- Throughout this thesis we will use the bar-notation for two purposes. For lower case symbols, for example, we denote by $\bar{x}$ the average of the M stacked vectors of $x = [x^i]$. Hence, assuming that the vectors x^i have a common size, we have:

$$\bar{x} = \frac{1}{M} \sum_{i=1}^{M} x^i.$$

Furthermore, we may use the bar-notation whenever we take the *Kronecker product* $\otimes$ of some matrix $W \in \mathbb{R}^{m \times n}$ with the identity matrix $I \in \mathbb{R}^{p \times q}$, both of possibly arbitrary size. We use the bar-notation as follows:

$$\bar{W} = W \otimes I.$$

Both applications of the bar-notation are somewhat related by:

$$\bar{x} = \frac{1}{M} \sum_{i=1}^{M} x^i = \frac{1}{M} (\mathbb{1}^T \otimes I)x = \frac{1}{M} \bar{\mathbb{1}}^T x,$$

where we introduce the notation $\mathbb{1} \in \mathbb{R}^M$ for the vectors of ones.

- We denote by $\rho(A)$ the spectral radius of the matrix A, i.e. its largest eigenvalue in absolute value.
- We say that a term $q(\alpha)$, as a function of α, is of order $\mathcal{O}(\alpha)$ if there exists a constant $c > 0$ such that $q(\alpha) \leq c\alpha$ for any α on the domain of q.
- Given a set $\mathcal{V} \subset \mathcal{S}$, we denote the complement by $\mathcal{V}^C := \mathcal{S} \backslash \mathcal{V}$.

1.3 Outline

The outline of the remainder of this thesis will be as follows. In Chapter 2 we review some basic theory about convex optimization. Then in Chapter 3 we consider convex optimization in the context of multi-agent networks. Here we discuss some basic theory and conventions about multi-agent networks, graph theory and stochastic matrices. We also discuss the inherent problem for two-step algorithms. Next, in Chapter 4, we review some of the existing methods for distributed optimization. We review methods in both the context of general convex optimization and distributed linear systems. The methods of greatest importance for this thesis are the decentralized gradient descent (`DGD`) method of Section 4.1.2, and the kernel projection (`KP`) method of Section 4.2.2. The latter is specifically designed for consistent linear systems, and we can only expect reasonable results for linear least squares problems when the optimal residual $r^* = b - Ax^*$ is reasonably small.

In Chapter 5 we propose another method under the name of `DSD`, a method based on the steepest descent method of Section 2.3.2. We first propose a variant of the steepest descent method for underdetermined systems, such as $A_i x = b_i$, useful for local minimization. We additionally show in Section 5.1.2 that an

application of the conjugate gradient method efficiently solves the local projection problems of the KP method, which is a useful results for implementations. Then in Section 5.2 we propose two variants of the DSD method, namely the α-DSD and the β-DSD variants. The α-variant is based on local residual minimization, whereas the β-variant is based on local error minimization, or equivalently, minimization of the distance from the individual solution spaces $\mathcal{X}_i^*$ for the system $A_i x = b_i$. The convergence analysis turned out to be complex for these method due to the nonlinear terms in the local steepest descent steps. We did manage to show a strict decrease of the error for the β-DSD when applied to consistent linear systems, in Section 5.2.1. Nonetheless, we establish a link for both variants of the DSD methods with both the DGD and KP method. We conclude that we can interpret these variants as intermediate approaches of the KP and DGD methods. This link is confirmed with numerical experiments in Section 5.3. Finally, we discuss the results of this thesis in Chapter 6.

2 Convex optimization

In this chapter we consider convex optimization problems as a special class of mathematical optimization problems of the form:

$$\begin{aligned} \text{minimize} \quad & f_0(x) \\ \text{subjected to} \quad & f_i(x) \leq d_i, \qquad i = 1, \ldots, k \\ & x \in \mathbb{R}^n. \end{aligned} \tag{2.1}$$

Then we will consider a more specific class of convex problems called the unconstrained minimization; these are optimizations problems of the form (2.1), but with the absence of the constraints $f_i(x) \leq d_i$. Next we consider quadratic minimization problems, i.e. unconstrained minimization problems where $f_0(x)$ is a quadratic function; a well-studied minimization problem which is known to be directly related to the linear least squares problem. Finally, in the last section, we consider some algorithms for solving unconstrained minimization problems. In particular we study the steepest descent method and the conjugate gradient method. These methods are extensively studied by both the research fields of mathematical optimization and numerical mathematics, since they are not only straightforwardly applicable to general convex, unconstrained minimization problems, but they also have some useful properties in the context of quadratic minimization. Due to these properties we are able to efficiently solve linear least squares problems and derive insightful convergence theorems.

2.1 Convex optimization problems

Mathematical optimization problems of the form (2.1) arise in many applications. Usually the interpretation of these problems, is to consider the objective function $f_0(x) : \mathbb{R}^n \to \mathbb{R}$ as some cost-function corresponding to the choice of the vector $x \in \mathbb{R}^n$, and the constraint functions $f_i(x) : \mathbb{R}^n \to \mathbb{R}$ as a confinement to the allowed choices for $x \in \mathbb{R}^n$. Now the choice of x is called feasible when we indeed have that $f_i(x) \leq d_i$ for all i and we can now define the feasible set $\mathcal{X} \subseteq \mathbb{R}^n$ as:

$$\mathcal{X} := \{x \in \mathbb{R}^n \mid f_i(x) \leq d_i \text{ for all } i\}. \tag{2.2}$$

Minimization of the objective function is confined to the feasible set $\mathcal{X}$. With this in mind we define the optimal solution to the mathematical optimization problem as follows.

Definition 2.1 (optimal solution)**.** *The vector $x^* \in \mathbb{R}^n$ is called the optimal solution for the mathematical optimization problem (2.1) if $x^* \in \mathcal{X}$ and if $f_0(x^*) \leq f_0(x)$ for all $x \in \mathcal{X}$.*

Mathematical optimization problems of the form (2.1) are a broad class of optimization problems and its properties are mainly depending on the properties of the functions $f_0, f_1, \ldots, f_k$. Therefore the problem of finding an optimal solution is generally a non-trivial and difficult task. Fortunately many of the optimization problems that originate from practical applications have useful properties that can be exploited when we are searching for an optimal solution. For example, one important and well-known property is convexity of both the objective and constraint functions. We will first give a formal definition of convex sets and convex functions.

Definition 2.2 (convex set)**.** *A set $C \subseteq \mathbb{R}^n$ is convex if for every $x_1, x_2 \in C$ and for any θ with $0 < \theta < 1$ we have that $\theta x_1 + (1 - \theta)x_2 \in C$.*

Intuitively a convex set is a subset C of $\mathbb{R}^n$ that contains all the points on the line segments between any two points in the set C. Observe that $C = \mathbb{R}^n$ is a convex set itself. Now a convex function is a function f whose domain is a convex set and is furthermore formally defined as follows.

Definition 2.3 (convex function)**.** *The function $f : C \to \mathbb{R}$ is convex if its domain $C \subseteq \mathbb{R}^n$ is a convex set and if for any $x_1, x_2 \in C$ and any θ with $0 < \theta < 1$ we have that:*

$$f(\theta x_1 + (1 - \theta)x_2) \leq \theta f(x_1) + (1 - \theta)f(x_2). \tag{2.3}$$

Similarly, we say that a function is *strictly convex* when we have a strict inequality in equation (2.3).

In this thesis we mainly consider continuous convex functions whose domain is the entire space $\mathbb{R}^n$, i.e. we have that $C = \mathbb{R}^n$. In case C is a proper subset of $\mathbb{R}^n$, then it is described in [11, Chapter 3 (p. 68)] how we can extend the domain of f to $\mathbb{R}^n$ while maintaining convexity. However, this extension does not ensure continuity on the boundary of C.

When we have to deal with complex expressions for f, it can be difficult to recognize convexity by definition 2.3. This is typically the case when one deals with optimization problems that arise from practical applications. Also when we are dealing with complex convex functions, then it is in our best interest to show that f is indeed convex because these functions have useful properties; properties that can be exploited in the development of solving algorithms (or *solvers*) for convex optimization problems. Solvers for convex optimization problems have indeed been developed for quite some time and we can appreciate that the recognition of convex functions is an important step for solving any optimization problem. Otherwise it can be worth the effort to try to reformulate your problem such that f becomes convex. Anyway, convexity of functions is an attractive property and we are interested in both the properties and conditions for this property.

In order to show that a given function f is convex, we could try a direct approach by showing convexity of f by definition 2.3. That is, we should show that the inequality (2.3) holds for any $x, y \in C$. Another approach would be

to use the well-known first- and second-order conditions of convexity; assuming that f is at least twice differentiable, these are properties of f based on its first- and second-order derivatives, respectively $\nabla f(x)$ and $\nabla^2 f(x)$. The first-order conditions is for example described in [11, Section 3.1.3 (p. 69)] or in [12, Proposition 1.2.5 (p. 31)]. We state the property in the following lemma, which is provided without proof.

Lemma 2.4 (first-order convexity condition)**.** *Let $f : C \to \mathbb{R}$ be a differentiable function on the convex domain C, then f is convex if and only if:*

$$f(x_2) \geq f(x_1) + \nabla f(x_1)^T (x_2 - x_1), \tag{2.4}$$

for any $x_1, x_2 \in C$.

Now again when f is strictly convex, then we have a strict inequality here [11, Equation (3.3) (p. 70)]. Observe that the right-hand side of the inequality is the expression for the tangent space of f at $x_1 \in C$, hence the geometrical interpretation of this property is that the tangent space at any point $x \in C$ is always below the function f on the domain C. The tangent space, which is also the first-order Taylor approximation of f, is considered to be a global under-estimator of f.

The second-order condition of convexity identifies a convex function by its *Hessian*, i.e. its second-order derivative $\nabla^2 f$. Simply put, the condition relates positive semi-definiteness of the Hessian, i.e. $\nabla^2 f \geq 0$, to convexity of f. Or equivalently: In order for f to be convex, the inner product of the gradient ∇f with any positive direction v (a direction in which none of the components of x decrease) must be nondecreasing in this direction v. This condition is formulated in for example [11, Section 3.1.4 (p. 71)], but it is formulated and proved more extensively in [12, Proposition 1.2.6 (p. 34)]. In the latter, positive definiteness of the Hessian is related to strict convexity of f and it is shown how the converse relation of convexity towards positive semi-definiteness is guaranteed. We formulate this lemma without proof.

Lemma 2.5 (second-order convexity condition)**.** *Let $f : C \to \mathbb{R}$ be a twice differentiable function on the convex domain C.*

(i) If $\nabla^2 f$ is positive semi-definite, i.e. if $\nabla^2 f(x) \geq 0$, for all $x \in C$, then f is convex over C.

(ii) If $\nabla^2 f$ is positive definite, i.e. if $\nabla^2 f(x) > 0$, for all $x \in C$, then f is strictly convex over C.

(iii) If C is open and f is convex over C, then $\nabla^2 f$ is positive semi-definite for all $x \in C$.

Notice that strict convexity does not necessarily guarantee positive definiteness of the Hessian, even if C is open. The most straightforward example for this is the scalar, strict convex function $f(x) = x^4$ on the open interval $\mathbb{R}$; we observe that the Hessian is zero in the point $x = 0$. In this case we can merely guarantee positive semi-definiteness by statement (iii) in the above lemma.

Both the first- and the second-order condition for convexity can be used to identify convex functions, however they can also be used to formulate and prove

optimality criteria for solutions of convex optimization problems of the general form as in equation(2.1); here we require that $f_0, f_1, \ldots, f_k$ are differentiable convex functions. The optimality criterion that we consider here is described in [11, Section 4.2.3 (p. 139)]. We state the result in the following lemma and provide a short proof for it.

Lemma 2.6 (first-order optimality criterion)**.** *Let $f_0 : C \to \mathbb{R}$ be the convex objective function corresponding to the optimization problem (2.1). Furthermore, let $\mathcal{X}$ be defined as in (2.2), then $x^* \in \mathcal{X}$ is optimal if and only if:*

$$\nabla f_0(x^*)^T(x - x^*) \geq 0, \tag{2.5}$$

for all $x \in \mathcal{X}$.

Proof. ($\Leftarrow$) Let $x^* \in \mathcal{X}$ satisfy equation (2.5). Combining this property of x^* with the first-order convexity condition, it is straightforward to derive that $f_0(x) \geq f_0(x^*)$ for any $x \in \mathcal{X}$. Hence x^* is optimal by definition 2.1.

($\Rightarrow$) Now let x^* be optimal and assume for contradiction that there exists some $x \in \mathcal{X}$ with the property that:

$$\nabla f_0(x^*)^T(x - x^*) < 0.$$

We consider any point $y(t)$ on the line segment between x and x^*, i.e. we have $y(t) = tx + (1-t)x^*$ for some $t \in [0, 1]$. Since the feasible set $\mathcal{X}$ is actually convex for convex optimization problems (this is explained in [11, Section 4.2.1 (p. 137)]) we know that $y(t)$ belongs to the feasible set $\mathcal{X}$ for any $t \in [0, 1]$. Now observe that:

$$\frac{d}{dt} f_0(z(t))\bigg|_{t=0} = \nabla f_0(x)^T(x - x^*) < 0.$$

Hence there must exist some small positive $t > 0$ such that $f_0(y(t)) < f_0(x^*)$. This contradicts the assumptions that x^* is optimal and we conclude that can not exist an $x \in \mathcal{X}$ with the property that $\nabla f_0(x^*)^T(x - x^*) < 0$. □

In the next section we consider a special class of convex optimization problems, namely the class of unconstrained minimization problems. In this case we are able to simplify the optimality criterion of the previous lemma. Before we continue we consider an example that illustrates how the theory of this section applies to general quadratic functions.

Example 2.1 (quadratic functions)
Quadratic functions have the main focus in this thesis and in this example we consider them as a special case. A quadratic function $f : \mathbb{R}^n \to \mathbb{R}$ is a polynomial function in the components of $x \in \mathbb{R}^n$, with maximum degree 2. Hence they can be written in the following form:

$$f(x) = \frac{1}{2} x^T Q x - q^T x + r, \tag{2.6}$$

here we have $Q \in \mathbb{R}^{n \times n}$, $q \in \mathbb{R}^n$ and $r \in \mathbb{R}$. We may assume without loss of generality that Q is symmetric. If this is not the case we can simply redefine

$\hat{Q} := \frac{1}{2}(Q + Q^T)$ and use the equality $x^TQx = \frac{1}{2}x^T(Q + Q^T)x = x^T\hat{Q}x$ to redefine f such that Q is indeed symmetric. In order to apply either the first- or second-order conditions of convexity, we give the gradient and the Hessian of f:

$$\nabla f(x) = Qx - q, \tag{2.7}$$
$$\nabla^2 f(x) = Q. \tag{2.8}$$

We immediately see, by lemma 2.5, that convexity of f is ensured by $Q \geq 0$ (and strict convexity by $Q > 0$). In this case conditions for convexity are obtained relatively easy. We may wonder what conditions we find on f when we would use either (a) the first-order conditions of convexity from lemma 2.4 or (b) definition 2.3 of convex functions directly.

(a) When we substitute the expressions for f and ∇f in equation (2.4) and re-order terms conveniently, we obtain the following equivalent first-order condition of convexity:

$$\frac{1}{2}x^TQx - x^TQy + \frac{1}{2}y^TQy \geq 0,$$

for any $x, y \in \mathbb{R}^n$. We recall that Q is symmetric such that we have $x^TQy = y^TQx$ and we observe that an equivalent inequality is given by:

$$\frac{1}{2}(x - y)^TQ(x - y) \geq 0,$$

for any $x, y \in \mathbb{R}^n$. Now let $y = 0$ and $x \in \mathbb{R}^n$ be arbitrary, then we observe that the above equality is equivalent to $Q \geq 0$, i.e. the first-order and second-order conditions are equivalent.

(b) Now we substitute the expression for f in equation (2.3) and re-order the terms conveniently to obtain that a convex quadratic function satisfies by definition that:

$$0 \leq \theta(1 - \theta)x^TQx - 2\theta(1 - \theta)x^TQy + \theta(1 - \theta)y^TQy, \tag{2.9}$$

for any $x, y \in \mathbb{R}^n$. Similarly to (a), we obtain an equivalent inequality:

$$\theta(1 - \theta)(x - y)^TQ(x - y) \geq 0,$$

for any $x, y \in \mathbb{R}^n$. Here we observe that $\theta(1 - \theta) > 0$ for any $\theta \in [0, 1]$ such that the above condition on convexity is again equivalent to $Q \geq 0$.

In conclusion, we see that convexity of quadratic functions solely depends on the properties of the matrix Q, whereas the terms q and r are redundant in this context. ■

2.2 Unconstrained minimization

In this section we consider the simplest class of optimization problems, namely the class of unconstrained minimization problems. Naturally, an unconstrained

minimization problem is an optimization problem without constraints; hence it is of the following form:

$$\text{minimize} \quad f(x) \tag{2.10}$$

We assume that $f : C \to \mathbb{R}$ is a convex and twice differentiable function. The feasible set $\mathcal{X}$ (2.2) is now the convex domain C of f and in this thesis this is usually the entire space $\mathbb{R}^n$, as it is also the case for quadratic functions. For now we just require C to be a open convex set, such that differentiability of f holds on C.

Observe that the convex optimization problem (2.1) is a more general problem than the unconstrained minimization problem above, hence the optimality criterion of lemma 2.6 remains valid. However, assuming that f is differentiable on the open convex domain C we are able to specify this result. We follow the reasoning in [11, Section 4.2.3 (p. 140)] and state and prove the following lemma.

Lemma 2.7 (optimality criterion unconstrained minimization)**.** *Let $f : C \to \mathbb{R}$ be the convex objective function corresponding to the optimization problem (2.10). Then $x^* \in C$ is optimal if and only if:*

$$\nabla f(x^*) = 0. \tag{2.11}$$

Proof. ($\Rightarrow$) Let x^* be the optimal solution to (2.10). By lemma 2.6, and since $\mathcal{X} = C$, we know that $\nabla f(x^*)^T(x - x^*) \geq 0$ for any $x \in C$. Now let $x(t) = x^* - t\nabla f(x^*)$. Since C is open we know that there exist a sufficiently small $t > 0$ such that $x(t) \in C$. For this t we find that:

$$\nabla f(x^*)^T(x(t) - x^*) = -t\|\nabla f(x^*)\|_2^2 \geq 0.$$

This can only hold if $\nabla f(x^*) = 0$.

($\Leftarrow$) Trivial (see the proof of lemma 2.6). □

The result of the above lemma reveals that there are several possibilities. If equation (2.11) is uniquely solved for some $x^* \in C$, then the optimal solution x^* is unique. If there are multiple solutions, then there exists some open set $\mathcal{X}^* \subseteq \mathcal{X} = C$ over which ∇f is zero. Then we observe from equation (2.4) that for any $x_1, x_2 \in \mathcal{X}^*$ we have both $f(x_2) \geq f(x_1)$ and $f(x_1) \geq f(x_2)$. Hence, f is constant over $\mathcal{X}^*$ and we conclude that $\mathcal{X}^*$ is a set of optimal solutions. However, it is also possible that there are no solutions for equation (2.11) and generally the minimum is not attained in this case. That can either mean that the minimum is attained at the boundary of C or it can mean that the cost function f is unbounded over C. We will illustrate the possibilities in the following example.

Example 2.2 (optimality of quadratic functions)
In this example we are going to consider three quadratic functions given by:

$$f_i(x) = \frac{1}{2}x^T Q_i x - q_i^T x + r_i, \quad i = 1, 2, 3,$$

for some $Q_i \in \mathbb{R}^{2\times 2}$, $q_i \in \mathbb{R}^2$ and $r_i \in \mathbb{R}$. We require f_i to be convex, i.e. we require that $Q \geq 0$. Then in each case the optimality criterion of lemma 2.7 is given by:

$$\nabla f_i(x) = Q_i x - q_i = 0,$$

i.e. the optimal solution (if it exists) satisfies $Q_i x = q_i$. The three cases we consider have the following properties.

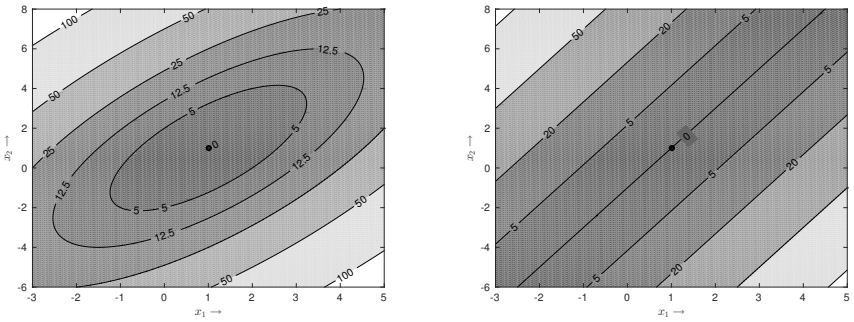
$i = 1$: We require Q_1 to be symmetric positive definite (SPD) such that the system $Q_1 x = q_1$ is uniquely solvable for any choice of q_1. Since $Q_1 > 0$ we know by lemma 2.5 that f is strictly convex. In this case we will observe that the optimal solution x^* is unique.

$i = 2$: We require Q_2 to be symmetric positive semi-definite and we require that $q_2 \in R(Q_2)$. In this case the system $Q_2 x = q_2$ has infinitely many solutions and hence there are infinitely many optimal solutions. The optimal set $\mathcal{X}^*$ is given by:

$$\mathcal{X}^* = \{x^* + tu \mid t \in \mathbb{R}\},$$

here x^* is a particular solution to $Q_2 x = q_2$ and u is a vector that spans the one-dimensional kernel of Q_2.

$i = 3$: We take $Q_3 = Q_2$, but now we require that $q_3 \notin R(Q_3)$. In this case the system $Q_3 x = q_3$ is inconsistent and an optimal solution does not exist. In this case we will observe that the minimization problem is unbounded: By moving in the right direction of the one-dimensional kernel of Q_3, we are able to make the value of f arbitrary small.



(a) Contour plot of f_1.

(b) Contour plot of f_2.

Figure 2.1: Contour plots of the quadratic surfaces corresponding to $f_1(x)$ (left) and f_2 (right) as a function of $x = [x_1\, x_2]^T$. The graph of f_1 is a paraboloid with elliptical contour lines at positive levels $f_1(x) = c > 0$, whereas the graph of f_2 is a horizontal parabolic cylinder with two parallel, linear contour lines at positive levels $f_2(x) = c > 0$.

We consider each case individually. For the first case ($i = 1$), let:

$$Q_1 = \begin{bmatrix} 4 & -2 \\ -2 & 2 \end{bmatrix}, \quad q_1 = \begin{bmatrix} 2 \\ 0 \end{bmatrix} \quad \text{and} \quad r_1 = 1.$$

One can verify that Q_1 has two positive eigenvalues, such that Q_1 is indeed SPD. Let $\mathbb{1}$ denotes the vector of ones, then we observe that $q_1 = Q_1 \mathbb{1}$ and we conclude that the unique optimal solution is $x^* = \mathbb{1}$. The choice for r_1 is such that $f_1(\mathbb{1}) = 0$. In this case the graph of $f_1(x)$ represents an elliptic paraboloid with a minimum point at $x^* = \mathbb{1}$; this is illustrated with a contour plot of f_1 in figure 2.1a.

For the second case ($i = 2$), let:

$$Q_2 = \begin{bmatrix} 4 & -2 \\ -2 & 1 \end{bmatrix}, \quad q_2 = \begin{bmatrix} 2 \\ -1 \end{bmatrix} \quad \text{and} \quad r_2 = \frac{1}{2}.$$

Observe that Q_2 can be written as $Q_2 = A_2^T A_2$ where $A_2 = \begin{bmatrix} 2 & -1 \end{bmatrix}$. Hence we have $\text{rank}(Q_2) = 1$ and $x^T Q_2 x = x^T A_2^T A_2 x = \|A_2 x\|_2^2 \geq 0$ for any $x \in \mathbb{R}^2$, which shows that Q_2 is symmetric positive semi-definite. Also observe that we again have that $q_2 = Q_2 \mathbb{1}$ and that r_2 is chosen such that $f_2(\mathbb{1}) = 0$. Furthermore we have:

$$\ker(Q_2) = \text{span}\{u\} := \text{span}\{\begin{bmatrix} 1 \\ 2 \end{bmatrix}\},$$

and we observe that $q_2 = Q_2(\mathbb{1} + tu)$ for any $t \in \mathbb{R}$. Similarly, we have $f_2(\mathbb{1} + tu) = 0$ for any $t \in \mathbb{R}$ and hence the line $\mathbb{1} + tu$ represents the set of all optimal solutions $\mathcal{X}^*$. In this case the graph of $f_2(x)$ represents a parabolic cylinder that is centered around the line $\mathbb{1} + tu$, which is illustrated with a contour plot of f_2 in figure 2.1b.

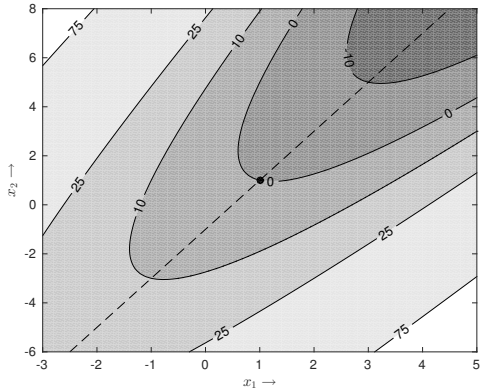


Figure 2.2: Contour plot of the quadratic surface corresponding to $f_3(x)$ as a function of $x = [x_1\ x_2]^T$. The dotted line corresponds to the line $\mathbb{1} + tu$; the graph of f_3 is a parabolic cylinder that is tilted downwards in the positive direction of the line $\mathbb{1} + tu$ and has parabolic contour lines at any level $f_3(x) = c$.

For the last case ($i = 3$), we let:

$$Q_3 = Q_2 = \begin{bmatrix} 4 & -2 \\ -2 & 1 \end{bmatrix}, \quad q_3 = \begin{bmatrix} 3 \\ 1 \end{bmatrix} \quad \text{and} \quad r_3 = \frac{7}{2}.$$

here we have chosen r_3 such that it remains true that $f_3(\mathbb{1}) = 0$. Comparing this case with the previous, we observe that $q_3 = q_2 + u$. Clearly we have $q_3 \notin \mathbb{R}(Q_3)$ such that the system $Q_3 x = q_3$ is inconsistent. Now, using the fundamental property that $R(Q_3) \perp \ker(Q_3)$, it is straightforward to show that $f_3(\mathbb{1} + tu) = -tu^T u = -t\|u\|_2^2$ and we find that:

$$\lim_{t \to \infty} f_3(\mathbb{1} + tu) = -\infty.$$

This shows that the unconstrained minimization problem with f_3 is unbounded over $\mathbb{R}^2$. If we confine f_3 to the open convex domain $[x_1\ x_2]^T \in (-3, 5) \times (-6, 8)$

then a minimum is attained on the boundary of the domain, at $x = [4.75\ 8]^T$ with $f(x) = -17.625$. The unboundedness of f_3 is explained by considering its graph over $\mathbb{R}^2$: In this case the graph of f_3 represents a tilted version of the parabolic cylinder of the previous case, which is illustrated with a contour plot of f_3 in figure 2.2. ■

2.2.1 Quadratic minimization and least squares

Quadratic minimization is the type of unconstrained minimization where f is a convex quadratic function (2.6). In the previous section we have seen an example with three types of convex quadratic minimization problems. We have also seen that the minimization of a general convex quadratic function is equivalent to solving a linear system $Qx = q$. Ideally we have a SPD matrix Q such that the system is consistent for any q and is furthermore uniquely solvable; in this case the optimal solution is given by $x^* = Q^{-1}q$.

However, not all linear systems $Ax = b$ can be related to a convex quadratic minimization problem. At least not directly since the matrix A is in general not SPD. Nonetheless linear systems are well-known to be related to convex quadratic minimization problems when we are considering its least squares solution. To this end, let $A \in \mathbb{R}^{m\times n}$ where $m > n$ and $\text{rank}(A) = n$ and let $b \in \mathbb{R}^m$ be any vector. Generally we have that $b \notin R(A)$ and the linear system is inconsistent. From the quadratic minimization point of view, the best thing we can try to do now is to minimize the following quadratic function:

$$f(x) := \frac{1}{2}\|b - Ax\|_2^2 = \frac{1}{2}x^T A^T Ax - (A^T b)^T x + b^T b. \tag{2.12}$$

We will show that $A^T A > 0$ under the assumptions that $m > n$ and $\text{rank}(A) = n$. Trivially we observe that $A^T A$ is symmetric. Now consider any eigenvalue λ of $A^T A$ and assume for contradiction that is of negative value. Let v be the corresponding eigenvector, then we find:

$$\|Av\|_2^2 = (Av)^T Av = v^T A^T Av = \lambda v^T v = \lambda \|v\|_2^2 < 0.$$

The contradiction shows that $\lambda \geq 0$ for any eigenvalue of $A^T A$. Similarly, if $\lambda = 0$ is a eigenvalue of $A^T A$ and u the corresponding eigenvector, then we find that $\|Au\|_2^2 = 0$. By the properties of a norm, this can hold if and only if $Au = 0$. This contradicts the assumption that $\text{rank}(A)$ and we may conclude that $A^T A$ is indeed SPD. Given that $A^T A > 0$ we know that the unconstrained minimization of $f(x) = \frac{1}{2}\|Ax - b\|_2^2$ satisfies the optimality criterion of lemma 2.7. Comparing equation (2.12) and (2.7) we conclude that the optimal solution satisfies:

$$A^T Ax = A^T b. \tag{2.13}$$

That is, it is the *least squares* solution of the system $Ax = b$ which is given by $x^* = (A^T A)^{-1} A^T b$.

In the next section we will see how the property of symmetric positive definiteness can be exploited when we are dealing with either a linear system or a quadratic minimization problem.

2.3 Descent methods

In this section we are going to consider solvers for unconstrained minimization and for now we assume that they are always strictly convex. In this case we know that the problem is uniquely solved by solving $\nabla f(x) = 0$, however, this is generally not done directly. It might either be too complex to develop a direct method for this purpose, but it is also possible that such a direct method has a non-feasible time-complexity: For example in quadratic minimization, solving the system $Qx = q$ directly is non-feasible when Q is a very large matrix. For this reason the methods of interest are generally iterative, i.e. methods that update an estimate $x_{(t)}$ of x^* in discrete time $t = 1, 2, 3, \ldots$, initiated by some initial guess $x_{(0)}$. For general descent methods this looks like:

$$x_{(t+1)} = x_{(t)} + \alpha_{(t)} d_{(t)}, \tag{2.14}$$

for some $\alpha_{(t)} \in \mathbb{R}_+$ and some $d_{(t)} \in \mathbb{R}^n$. The vector $d_{(t)}$ is called the *search direction* and in descent methods we require that $f(x_{(t)})$ decreases in the direction of $d_{(t)}$, i.e. the directional derivative of f in the direction of $d_{(t)}$ must be negative:

$$\partial_d f(x) := \nabla f(x)^T d < 0,$$

where we omitted the dependency on t for clarity. We will sometimes refer to $d_{(t)}$ by a *descent direction*. Since $d_{(t)}$ is a descent direction, we know by differentiability of f that there exists some $\alpha_{(t)} \in \mathbb{R}_+$ such that $f\left(x_{(t+1)}\right) < f\left(x_{(t)}\right)$. We call the choice for $\alpha_{(t)}$ the *step size* and preferably we always have:

$$\alpha_{(t)} = \underset{\alpha > 0}{\operatorname{argmin}} \left\{ f\left(x_{(t)} + \alpha d_{(t)}\right) \right\}. \tag{2.15}$$

This is what we call an *exact line search* for obvious reasons. The above problem is actually a convex unconstrained minimization problem itself. To see this, we define $f_d(\alpha) := f(x + \alpha d)$ with $\alpha > 0$, $x \in C$ and d some arbitrary descent direction. Then we observe that, since C is open, there must exist some greatest value $a > 0$ such that $x + ad \notin C$ and $x + \alpha d \in C$ for any α on the open convex interval $(0, a)$. Finally we observe that f_d is convex by the following derivation:

$$\begin{aligned} f_d(\theta\alpha_1 + (1-\theta)\alpha_2) &= f(x + (\theta\alpha_1 + (1-\theta)\alpha_2)d) \\ &= f\big(\theta(x + \alpha_1 d) + (1-\theta)(x + \alpha_2 d)\big) \\ &\leq \theta f(x + \alpha_1 d) + (1-\theta) f(x + \alpha_2 d) \\ &= \theta f_d(\alpha_1) + (1-\theta) f_d(\alpha_2), \end{aligned}$$

for any $\alpha_1, \alpha_2 \in (0, a)$ and any $\theta \in [0, 1]$. Hence, the exact line search (2.15) is equivalently solved by the problem $f_d'(\alpha) = 0$ by lemma 2.7. When such an exact line search is not possible due to the complexity of f, we can use a backtrack line search as is described in [11, Section 9.2 (p. 464)].

A descent method is characterized by the choice of $d_{(t)}$ and the step size $\alpha_{(t)}$. The gradient descent method is the best known method in this category and we consider it in the next section.

2.3.1 Gradient descent method

The gradient descent method is a classical method for solving unconstrained minimization problems. This is a descent method that uses the negative gradient $-\nabla f$ as the descent direction in each iteration. We will explain why the gradient is a natural choice. There are many descent directions that we could choose, but a straightforward approach is to search for the steepest descent (sd) direction which is defined as follows:

$$d_{sd} := \underset{\|v\|=1}{\operatorname{argmin}} \left\{ \nabla f(x)^T v \right\}.$$

Note that this expression depends on the choice of the norm $\|\cdot\|$. If we use the Euclidean norm $\|\cdot\|_2$, then we know from basic calculus that the maximum value of the directional derivative in Euclidean space $\partial_d f$ is indeed attained at the gradient ∇f [13, Section 14.6 (p. 916)]. The positive direction ∇f corresponds to a direction of maximal increase, whereas the negative direction $-\nabla f$ corresponds to a direction of maximal decrease. When we are using the Euclidean norm, the gradient- and steepest descent method are just the same methods. If we would however use for example the Q-norm $\|\cdot\|_Q$ (in case of quadratic minimization), then it is shown in [11, Section 9.4.1 (p. 476)] that the direction of steepest descent is given by $d_{sd,(t)} = -Q^{-1}\nabla f\left(x_{(t)}\right)$. This is of course an infeasible approach.

The update scheme of the gradient method is given by:

$$x_{(t+1)} = x_{(t)} - \alpha_{(t)} \nabla f\left(x_{(t)}\right), \tag{2.16}$$

where $\alpha_{(t)}$ is defined by (2.15) with $d_{sd,(t)} = \nabla f\left(x_{(t)}\right)$. A proof of linear convergence is provided in [11, Section 9.3.1, (p. 466)]. We omit the complete proof and in stead provide the essential properties of strict convex functions on which the proof is based upon. The first fundamental property of essence is the lower bound on the Hessian of a strict convex function, namely that:

$$\nabla^2 f \leq KI, \tag{2.17}$$

for some $K \in \mathbb{R}_+$. Note that in case of a quadratic function we can take $K = \lambda_{\max}(Q)$. Using a second-order Taylor expansion in combination with this upper bound, we obtain the following inequality:

$$f(x_2) \leq f(x_1) + \nabla f(x_1)^T (x_2 - x_1) + \frac{K}{2}\|x_2 - x_1\|_2^2, \tag{2.18}$$

for any $x_1, x_2 \in C$. This is an important property in the convergence proof of the gradient descent method. Now if we define by f^* the optimal value $f(x^*)$, then the previous upper bound can be used to derive the following inequality:

$$f^* \geq f(x) - \frac{1}{2K}\|\nabla f(x)\|_2^2, \tag{2.19}$$

for any $x \in C$.

The second fundamental property of essence is the lower bound on the Hessian of a strict convex function, namely that:

$$\nabla^2 f \geq kI, \tag{2.20}$$

for some $k \in \mathbb{R}_+$ ($k \leq K$). Note that in case of a quadratic function we would have $k = \lambda_{\min}(Q)$. This property can be used to derive a better lower bound than the one that is provided in the first-order condition of convexity of lemma 2.4, the result is as follows:

$$f(x_2) \geq f(x_1) + \nabla f(x_1)^T(x_2 - x_1) + \frac{k}{2}\|x_2 - x_1\|_2^2, \tag{2.21}$$

for any $x_1, x_2 \in C$. This gives us the following upper bound on the optimal value f^*:

$$f^* \geq f(x) - \frac{1}{2k}\|\nabla f(x)\|_2^2, \tag{2.22}$$

for any $x \in C$.

The convergence proof of the gradient method is based mainly on the the above results. For the gradient method we have linear convergence towards the optimal value, for which the explicit result is as follows:

$$f\left(x_{(t+1)}\right) \leq (1 - \frac{1}{\kappa}) f\left(x_{(t)}\right), \quad \text{where } \kappa := \frac{K}{k} \in (0, 1). \tag{2.23}$$

Note that $(1 - \frac{1}{\kappa}) < 1$. Here the constant κ is referred to by the condition number of f and in case of quadratic functions it is given by $\kappa = \frac{\lambda_{\max}}{\lambda_{\min}} = \kappa(Q)$, which is actually the 2-norm condition number for the SPD matrices.

2.3.2 Steepest descent method

In this section we consider the gradient descent method applied specifically to quadratic minimization problems. Sometimes the gradient descent method is also referred to by the *steepest descent (SD) method*, which is indeed justified when we are using the 2-norm. In this thesis we make the distinction that the steepest descent method is actually the gradient descent method specifically applied to quadratic minimization problems. When we are minimizing a strictly convex, quadratic function of the form (2.6), we observe that the constant part is negligible. That is, we can assume that we are minimizing a quadratic function $f(x) = \frac{1}{2}x^T Ax - b^T x$ for some $A > 0$ and some $b \in \mathbb{R}^n$. Equivalently, we are solving the system $Ax = b$ with a iterative method. We follow the conventions that are generally used in linear system solving and define the residual r, given an estimate x of the solution x^* to the system $Ax = b$, as follows:

$$r := b - Ax.$$

Not coincidentally the residual equals the negative gradient of f, i.e. we have $r = -\nabla f(x) = d_{sd}$. Now we derive an explicit expression for the exact line search by solving $f_r'(\alpha) = 0$. Using the the chain rule we find that:

$$\begin{aligned} f_r'(\alpha) &= \frac{d}{d\alpha} f(x + \alpha r) \\ &= \nabla f(x + \alpha r)^T \left(\frac{d}{d\alpha}(x + \alpha r) \right) \\ &= \big(A(x + \alpha r) - b\big)^T r \\ &= \big(\alpha Ar - r\big)^T r \\ &= \alpha r^T Ar - r^T r, \end{aligned}$$

such that the exact line search is clearly given by:

$$\alpha_{(t)} = \frac{r_{(t)}^T r_{(t)}}{r_{(t)}^T A r_{(t)}} = \frac{\|r_{(t)}\|_2^2}{\|r_{(t)}\|_A^2}, \tag{2.24}$$

where we used the well-defined A-norm for SPD matrices. The update scheme of the steepest descent method now looks as follows:

$$\begin{aligned} r_{(t)} &= b - Ax_{(t)}, \\ x_{(t+1)} &= x_{(t)} + \alpha_{(t)} r_{(t)}, \end{aligned} \tag{2.25}$$

where $\alpha_{(t)}$ is defined by equation (2.24). For practical reasons we mention that the the residual is equivalently updated by:

$$r_{(t+1)} = r_{(t)} - \alpha_{(t)} A r_{(t)}.$$

With respect to cost-efficiency of the method this is the favored update, since the term $Ar_{(t)}$ has to be computed for the step size $\alpha_{(t)}$ anyway.

The third equality in the above derivation of $\alpha_{(t)}$ shows that the exact line search is such that at any iteration $t+1$ we have:

$$r_{(t+1)}^T r_{(t)} = 0. \tag{2.26}$$

In other words, we observe that the new search direction is always orthogonal to the previous search direction. When the problem is of size n, then this means after at most $n+1$ we are searching in a direction in which we have searched before. This is illustrated with an example.

Example 2.3 (steepest descent)
We follow up on example 2.2 for the case $i = 1$; neglecting the constant part, we consider the function $f(x) = \frac{1}{2}x^T Ax - b^T x$ with:

$$A = \begin{bmatrix} 4 & -2 \\ -2 & 2 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} 2 \\ 0 \end{bmatrix}.$$

We recall that the optimal solution is given by $x^* = \mathbb{1}$. The `SD` method is typically initiated with a zero initial guess $x_{(0)} = 0$ and with this choice we have computed the first 12 iterates according to the update scheme (2.25), this is depicted in figure 2.3. Due to the earlier observation, we conclude and observe that at each odd iteration $t \geq 3$ the search direction is exactly the same as in any previous odd iteration. Naturally a similar result can be formulated for each even iteration.

Recall that each ellipse in figure 2.3 corresponds to a level set $f(x) = c$ for some $c \in \mathbb{R}$, i.e. the ellipse is a level surface, and from basic calculus we know that the gradient $f(x)$ at any point x on the level surface is perpendicular to this surface [13, Section 14.6 (p. 917)]. The geometrical properties of an ellipse are such that if we were to start in any point x_0 on the ellipse that is crossed by one of its axes, then $-\nabla f(x_0)$ points directly towards the center x^*. In this particular case the `SD` method converges in one step. We now argue that the axes are actually the eigenvectors of A. To see this, let x_0 be such a starting point,

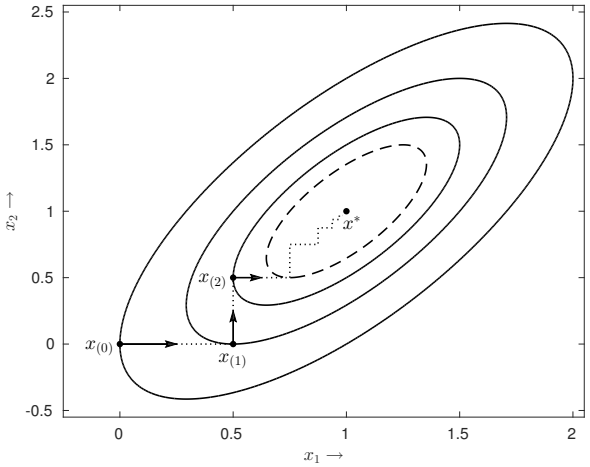


Figure 2.3: The first twelve iterations of the `SD` method applied to $f(x)$ as a function of $x = [x_1\ x_2]^T$, with an initial guess $x_{(0)} = 0$. The contour lines correspond to level surfaces $f(x) = f(x_{(i)})$ for $i = 0, 1, 2, 3$.

then by our claim we have that $-\nabla f(x_0) = r_0 = b - Ax_0$ is an eigenvector of Q with some corresponding eigenvalue $\lambda > 0$. We do a single `SD` update according equation (2.25) and pre-multiply with A on both sides to find that x_1 satisfies:

$$\begin{aligned} Ax_1 &= A(x_0 + \alpha_0 r_0) \\ &= A\left(x_0 + \frac{r_0^T r_0}{r_0^T A r_0} r_0\right) \\ &= A\left(x_0 + \frac{1}{\lambda} r_0\right) \\ &= Ax_0 + r_0 \\ &= b, \end{aligned}$$

i.e. we find that $x_1 = x^*$. The geometrical interpretation of this problems learns us that the eigenvectors of A correspond indeed to the axes of the ellipse $f(x) = c$ for any $c > f(x^*)$. ■

The `SD` method is just a specific case of the gradient descent method with an exact line search and therefore we already know that the method converges linearly. However, there is also an alternative proof provided in [14, Section 8.6, (p. 235)]. It is observed that f is equivalently defined (except for a constant term) as follows:

$$f(x) = \frac{1}{2}(x^* - x)^T A(x^* - x) = \frac{1}{2}\|x^* - x\|_A^2. \tag{2.27}$$

Observe that f can be interpreted as the squared A-norm of the error $x^* - x$. We end this section with the convergence theorem that is provided in [14]. We omit the proof here, but we will reconsider it in Section 5.1 where we show that the steepest descent method convergence linearly for particular positive semi-definite matrices as well. For SPD A the convergence result is as follows.

Theorem 2.8 (steepest descent)**.** *For any initial guess $x_0 \in \mathbb{R}$ the steepest descent method (2.25) converges linearly to the unique optimal solution x^* of f, equation (2.27). Furthermore, at any iteration $t+1$ we have:*

$$\|x^* - x_{(t+1)}\|_A^2 \leq \left(\frac{\kappa - 1}{\kappa + 1}\right)^2 \|x^* - x_{(t)}\|_A^2, \quad \textit{where} \quad \kappa = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}. \tag{2.28}$$

In the next section we consider an improvement of the SD method, where it is prevented that at some point in discrete time t we are searching in a direction in which we have searched before.

2.3.3 Conjugate gradient method

The conjugate gradient (CG) methods uses the geometrical properties of a quadratic minimization problem in a way that overcomes the shortcomings of the SD method. The fundamental problem with the SD method is that the convergence can be extremely slow when $\kappa(A)$ is large. A geometrical interpretation of this problem, similar to the one in example 2.3, is given in [15, (Section 10.2.1, (p. 521)]: "*Geometrically this means that the level curves of f are very elongated hyperellipsoids and minimization corresponds to finding the lowest point in a relatively flat, steep-sided valley. In steepest descent, we are forced to traverse back and forth across the valley rather than down the valley.*"

The CG method is not only a popular method in the field of optimization, but perhaps even more so in the field of numerical mathematics. Here it is classified as a method belonging to the broad class of the so called *Krylov subspace* methods.

The CG method will be explained according the approach of both [14, Chapter 9] and [15, Chapter 10]. The CG method avoids the back an forth traversing by choosing linear independent descent directions. Now suppose we had n linear independent descent directions $d_{(0)}, d_{(1)}, \ldots, d_{(n-1)}$ beforehand, and that at any iteration $t+1$ we have that:

$$x_{(t+1)} = \underset{x \in x_{(0)} + \text{span}\{d_{(0)}, \ldots, d_{(t)}\}}{\text{argmin}} \|x^* - x\|_A, \tag{2.29}$$

then clearly the exact solution is found in at most n iterations. The CG method uses the geometrical properties of f to ensure that this is a viable approach for an iterative method, i.e. the descent directions will only be known implicitly beforehand and the above minimization will be done iteratively with relatively cheap updates. To this end we first require that the set of descent directions is A-orthogonal, or *conjugate with respect to A*. This means that $d_{(j)}^T A d_{(i)} = 0$ for any $i \neq j$, such that we can write that:

$$x^* = \alpha_0 d_{(0)} + \ldots + \alpha_{n-1} d_{(n-1)},$$

where we find that $\alpha_i = \frac{d_{(i)}^T b}{d_{(i)}^T A d_{(i)}^T}$ by pre-multiplying with $d_{(i)}^T A$ on both sides. Observe that if we simply used Euclidean orthogonality here, then we needed the optimal solution x^* to determine α_i. With respect to A-orthogonality we make the following observation. Denote by $y = Sx$ the coordinate transformation with the transition matrix:

$$S := A^{\frac{1}{2}}. \tag{2.30}$$

Since SPD matrices are diagonalizable, we can write $A = U^T DU$ for some positive diagonal D and orthonormal U. In this way we observe that $A^{\frac{1}{2}} = U^T D^{\frac{1}{2}} U$ is well-defined. Now observe that the A-norm in the x coordinate system is transformed to the 2-norm in the y coordinate system; $\|y\|_2 = \|A^{\frac{1}{2}} x\|_2 = \|x\|_A$. Similarly, A-orthogonality in the x coordinate system transforms to actual orthogonality in the y coordinate system; $y_j^T y_i = x_j^T A x_i = 0$. From this point of view we argue that the minimization (2.29), or equivalently the minimization of $f(x)$, with the A-orthogonal set $\{d_{(i)}\}$ is the same as the minimization of $\|y^* - y\|_2 := \|Sx^* - Sx\|_2$ with the orthogonal set $\{Sd_{(i)}\}$. Considering equation (2.29) as an iterative process for $t = 1, 2, \ldots, n$, we can now see that the new direction $Sd_{(t)}$ is independent of the previous directions. In the y coordinate system we can simply do a projection of the current estimate $Sx_{(t)}$ on the new direction. In the original coordinate system, this means that we can simply do an exact line search with $d_{(t)}$. We may now follow the derivation in Section 2.3.2 to conclude that the exact line search is given by:

$$\alpha_{(t)} = \frac{r_{(t)}^T d_{(t)}}{d_{(t)}^T A d_{(t)}}. \tag{2.31}$$

Another justification of this exact line search is given in the so called *Expanding subspace theorem* [14, (p. 266)].

As mentioned before, the `CG` method does not require the A-orthogonal set beforehand, instead the `CG` method computes this sequentially. Recall that $-\nabla f$ is a direction of steepest descent and in the `CG` method we aim to diverge as less as possible from this direction. This is initiated with $d_{(0)} = r_{(0)} = -\nabla f(x_{(0)})$ and then we require at any iteration $t + 1$ that:

$$d_{(t+1)} = \underset{d \in \text{span}\{Ad_{(0)}, \ldots, Ad_{(t)}\}^\perp}{\text{argmin}} \|d - r_{(t+1)}\|_2.$$

Note that it is the closest, in Euclidean norm, A-orthogonal vector to the set of all previous A-orthogonal vectors. In principle $d_{(t+1)}$ is obtained with a conjugate Gram-Schmidt process with $r_{(t+1)}$ to all the previous vectors $d_{(0)}, \ldots, d_{(t)}$. The `CG` method however, has the advantage of shorter recurrences due to its special structure; it can be derived that $r_{(t+1)}$ is already A-orthogonal to $d_{(0)}, \ldots, d_{(t-1)}$, such that the new residual only needs be to A-orthogonalized with respect to the previous search direction $d_{(t)}$ [14, 15]. Then the following update scheme for the `CG` method suffices:

$$\begin{aligned} \alpha_{(t)} &= \frac{r_{(t)}^T d_{(t)}}{d_{(t)}^T A d_{(t)}} \\ x_{(t+1)} &= x_{(t)} + \alpha_{(t)} d_{(t)} \\ \beta_{(t)} &= \frac{r_{(t+1)}^T A d_{(t)}}{d_{(t)}^T A d_{(t)}} \\ d_{(t+1)} &= r_{(t+1)} + \beta_{(t)} d_{(t)}. \end{aligned} \tag{2.32}$$

For the `CG` method it is observed that $r_{(t+1)}$ is already A-orthogonal to the set $d_{(0)}, \ldots, d_{(t-1)}$, such that we only have to do the A-orthogonalization with respect to $d_{(t)}$. This is also stated in the following lemma, which is formulated and proved in [14, (p. 267, 270)].

Lemma 2.9 (conjugate gradient properties)**.** *If the CG method (2.32) does not terminate at iteration $t = k$, then the following properties hold:*

(i) $span\{r_{(0)}, r_{(1)}, \ldots, r_{(k)}\} = span\{r_{(0)}, Ar_{(0)}, \ldots, A^k r_{(0)}\}$;

(ii) $span\{d_{(0)}, d_{(1)}, \ldots, d_{(k)}\} = span\{r_{(0)}, Ar_{(0)}, \ldots, A^k r_{(0)}\}$;

(iii) $r_{(k)}^T d_{(i)} = 0 \quad for\ any \quad i < k$;

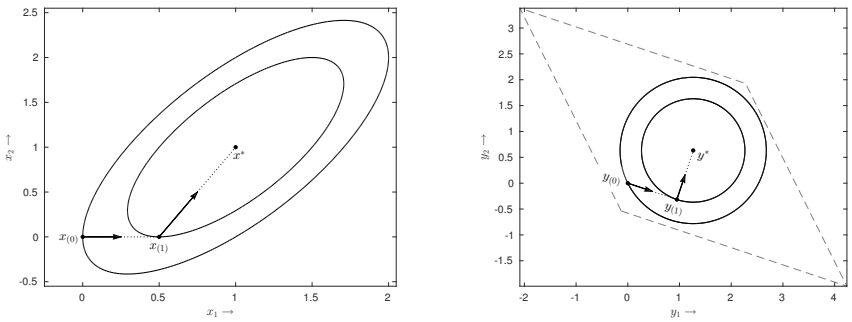
(iv) $d_{(k)}^T A d_{(i)} = 0 \quad for\ any \quad i < k$.

The first two properties show that the space spanned by either the residuals or the descent directions, are both equal to the space:

$$\mathcal{K}_{k+1}(A, r_{(0)}) := \text{span}\{r_{(0)}, Ar_{(0)}, \ldots, A^k r_{(0)}\}. \tag{2.33}$$

This is known as a *Krylov subspace* of dimension $k + 1$, which is a well-known concept in the field of numerical mathematics. The third property shows that $r_{(t+1)}$ is independent of all previous descent directions, which shows that the method can not stagnate for any $r_{(t+1)} \neq 0$. Finally, the last property verifies that the search directions are conjugate with respect to A. Furthermore we conclude by the linear independence of the residuals (iii) , that the method converges after at most n iterations. We follow up on example 2.3 to illustrate the convergence of the CG method.

Example 2.4 (conjugate gradient)
We now solve the same problem with the CG method. Since the problem is of size $n = 2$, the method converges in just two iterations. This is illustrated graphically in figure 2.4a. We mention that the two search directions are A-orthogonal. Now we consider the same problem in the transformed coordinate system $y = Sx$. We recall that the two search directions are actually orthogonal in the transformed coordinate systems, which is also observed in figure 2.4b. Observe that the



(a) Original coordinate system. **(b)** Transformed coordinate system.

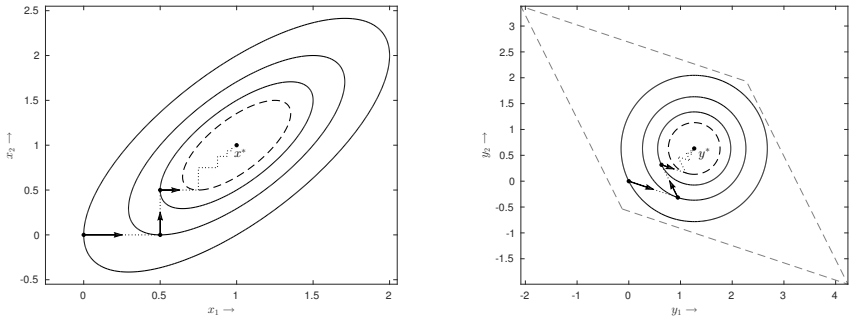
Figure 2.4: The CG method applied to the quadratic minimization of f with a zero initial guess $x_{(0)} = 0$. The convergence is shown in the original system (left) and in the transformed coordinate system $y = Sx$ (right). The dashed gray boundary on the right represents the transformation of the boundary from the original domain on the left.

elliptical level surfaces transformed to circular ones, which we can explain by considering f as a function of y, by substitution of $x = A^{-\frac{1}{2}}y$. We find that:

$$f(y) = \frac{1}{2}(y^* - y)^T(y^* - y),$$

and we observe that the levels surfaces of f indeed correspond to circles.

We are going to reconsider the convergence of the SD method. In figure 2.5a we revisit the SD convergence behavior in the original coordinate system, whereas in figure 2.5b we consider this in the transformed coordinate system. The coordinate transformation emphasizes the back and forth traversing that is described in [15, (Section 10.2.1, (p. 521)]. It is an intuitive visualization of the shortcomings of the SD method.



(a) Original coordinate system. **(b)** Transformed coordinate system.

Figure 2.5: The SD method applied to the quadratic minimization of f with a zero initial guess $x_{(0)} = 0$ considered in the original system (left) and in the transformed coordinate system $y = Sx$ (right).

■

We have analyzed the CG method as a method that converges in at most n iterations, in this regard we may interpret CG as a direct method. However, for large matrices A we hope to converge to a reasonable estimate in far less than n iterations and in this regard we like to consider CG as an iterative method. From lemma 2.9 we know that at any iteration t we have that $\text{span}\{d_{(0)}, d_{(1)}, \ldots, d_{(t)}\} = \text{span}\{r_{(0)}, r_{(1)}, \ldots, r_{(t)}\}$ and hence, from the minimization (2.29), we know that CG can not perform worse than SD. A more specific result of linear convergence of CG is derived in [14, Section 9.8 (p. 282)].

Theorem 2.10 (conjugate gradient)**.** *For any initial guess $x_{(0)} \in \mathbb{R}^n$ the conjugate gradient method (2.32) terminates in at most n iterations. Before termination, the conjugate gradient method converges linearly to the unique optimal solution x^* of f. Furthermore, at any iteration $t+1$ we have:*

$$\|x^* - x_{(t+1)}\|_A^2 \leq 4\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^2 \|x^* - x_{(t)}\|_A^2, \quad \textit{where} \quad \kappa = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}. \quad (2.34)$$

3 Distributed convex optimization

In this chapter we consider unconstrained optimization problems that are, in some sense, distributed. This means that the total of information that defines the minimization problem (2.10) is distributed over multiple entities. Depending on the context, such entities can for example be some type of sensors or just some computer processors. From a theoretical point of view it is sufficient to simply refer to this entities by *agents* and we generally assume that there are $M > 1$, although strictly speaking this number can also be time-varying. The information may be distributed, since for some problems the convex function $f(x)$ is written as the sum of some M other convex functions, i.e. we may have:

$$f(x) = \sum_{i=1}^{M} f_i(x), \tag{3.1}$$

for some M convex functions $f_1, \ldots, f_M$. Naturally, in a distributed environment the *local function* $f_i(x)$ is only known to some agent i, assuming that the agents are numbered as $1, \ldots, M$. We say that such a distributed optimization problem is *centralized* when there is some global entity that has excess to the *global function* $f(x)$. This is for example the case with parallel computing, where the aim is to solve very large minimization problems effectively by doing the computations in parallel over multiple processors. To this end the information of equation (3.1) can be distributed over M processors and then each processors is able to communicate its computations with the global entity, which in turn is able to combine all the computed sub-results in a suitable manner. In this case the communication of the agents is centralized. However, in this thesis we are specifically interested in optimization problems in which no such global entity exists. Such problems have a truly distributed nature and we say that the communication is *decentralized*; in this context the agents can only communicate with neighboring agents, i.e. other agents that are in some sense nearby. We say that availability of information is *local*. In some cases these problems arise naturally such as for example in sensor networking [2, 3, 4, 5].

In many ways a truly distributed environment is non-favorable, e.g. in terms of solving the optimization problem *efficiently*. From this point of view the problem is inherently deficient since (i) none of the agents have excess to the global information and (ii) the (precise) communication structure among agents is generally unknown to individual agents. This means that a large amount

of communication may be required. Also with regard to solving the problem *effectively*, the truly distributed environment is non-favorable. Generally, we consider agents to be entities with small computational capacity and limited energy resources, therefore we allow agents to do only small computations and aim to reduce the required amount of communication as much as possible; here we mention that communication is considered to be the most energy consuming operation in distributed computing. In contrast, communication is considered to be the most time consuming operation in parallel computing. This may not be the main motivation in distributed computing, though it is yet another motivation that we will take into account. Still, the truly distributed environment may be favorable over centralized environments in some cases. The main reason for this is the flexibility, robustness and its natural occurrence in practical applications.

The organization of this chapter will be as follows. First in Section 3.1 we introduce multi-agent networks, the related notational conventions and basic algorithms for distributed averaging. Then in Section 3.2 we consider distributed unconstrained minimization problems in the context of multi-agent networks. Here we consider some of the general approaches, as well as some of the well-known issues. Then, as a special case, we will consider how this applies to distributed quadratic minimization problems in Section 3.2.1.

3.1 Multi-agent networks

In a truly distributed environment all the computations and communication are performed by $M > 1$ autonomous agents. These agents are part of a *multi-agent network*. In addition, the structure of the multi-agent network is defined by the underlying communication topology. This is typically characterized by a graph $\mathcal{G}$, either directed or undirected and either time-dependent or time-independent. For time-dependent graphs we should use the notation $\mathcal{G}_{(t)}$, but in this thesis we mainly consider the time-independent case. The graph $\mathcal{G}$ has M vertices corresponding to the M agents; similarly to the agents, these are numbered as $1, \dots, M$. Furthermore, the graph $\mathcal{G}$ has a set of *arcs* that characterize the neighbor relations of the network. In directed networks, we say that j is a neighbor of i whenever agent j is able to share its information with agent i. This does not necessarily mean that i can share its information with j, i.e. i is not necessarily a neighbor of j; this is characterized with a directed arc $(j, i) \in \mathcal{G}$. In undirected networks, the communication among neighbors always goes both ways and the use of directed arcs becomes redundant. Then we can simply characterize a neighboring relation between i and j by an undirected edge (i, j), or equivalently by the edge (j, i). We mention that the use of either edges or arcs is not of great importance in this thesis, in stead we are mainly interested in each agent's neighbor sets $\mathcal{N}_i \subseteq \{1, \dots, M\}$. For either directed and undirected graphs, we may define it for each agent i as follows:

$$\mathcal{N}_i := \{j \mid (j, i) \in \mathcal{G}\}. \tag{3.2}$$

We will often include the agent itself to its neighbor set and hence we introduce a shorthand notation for this by $\mathcal{N}_i^* = \mathcal{N}_i \cup \{i\}$. Next we introduce some standard conventions, definitions and theory that are common to graph theory. First of all we denote by d_i the *degree* (or *indegree* in case of directed graphs) of some

vertex i; in both cases it represents the amount of neighbors of agent i and it is equal to the cardinality of the set $\mathcal{N}_i$. With this definition we define the *degree matrix* $D := \text{diag}\{d_1, \ldots, d_M\}$ of size $M \times M$. Furthermore, in graph theory, graphs are typically represented by the so called *adjacency matrix* $N = [n_{ij}]$. This matrix N contains all the neighbor relations and is uniquely defined by:

$$n_{ij} := \begin{cases} 1 & \text{if } (i,j) \in \mathcal{G}, \\ 0 & \text{otherwise.} \end{cases} \tag{3.3}$$

We mention that the definition of N applies to both undirected and directed graphs. Notice that $n_{ji} = 1$ just in case that j is a neighbor of i, such that the neighbor relations are actually contained in N^T. For undirected graphs this is all the same, since in this case N is symmetric.

For undirected graphs, the degree and adjacency matrix uniquely defines the *Laplacian matrix* L as follows:

$$L := D - N. \tag{3.4}$$

This matrix may be used to identify certain properties of a graph, such as for example *connectedness*, a property that applies to both undirected- and directed graphs. For this we first need to define the notion of *paths*.

Definition 3.1 (paths)**.** *Given a graph $\mathcal{G}$, either directed or undirected, a set of vertices $i =: i_0, i_1, \ldots, i_{k-1}, i_k := j$ defines a path from i to j, either directed or undirected, if and only if for all $l = 0, \ldots, k-1$ we have that (i_l, i_{l+1}) is respectively an arc or an edge in $\mathcal{G}$.*

Then connectedness is defined as follows.

Definition 3.2 (connected graphs)**.** *A graph $\mathcal{G}$, either directed or undirected, is called connected when there exists a path, either directed or undirected respectively, between every pair of vertices i, j.*

For directed graphs we mention that this form of connectedness is usually referred to by *strongly connectedness*. In Section 4.1 we describe a form of connectedness for a sequence of time-dependent graphs $\mathcal{G}_{(t)}$, which is considered to be a minimal assumption on connectivity in practical applications. This property is referred to by *repeatedly jointly strongly connectedness*, which is usually a more realistic form of connectedness in a dynamic environment.

We elaborate on undirected graphs. For these graphs, there exists another useful matrix called the *incidence matrix* H of size $M \times K$, where K denotes the number of edges of $\mathcal{G}$. This matrix, which is not necessarily uniquely defined, is obtained by first orienting the graph $\mathcal{G}$ as follows: Convert any edge (i, j) to a directed arc at random, either from i to j or vice versa, then define its entries as follows:

$$h_{ik} = \begin{cases} -1 & \text{if } i \text{ is the tail of the } k\text{th edge}, \\ 1 & \text{if } i \text{ is the head of the } k\text{th edge}, \\ 0 & \text{otherwise.} \end{cases} \tag{3.5}$$

For this matrix H it can be derived that $HH^T = D - N = L$, see e.g. [16]. Furthermore, it can be derived for connected graphs that $\text{rank}(H) = M - 1$ and $\ker(H^T) = \text{span}\{\mathbb{1}\}$. Subsequently, we may obtain the result of the following lemma.

Lemma 3.3. *Let $L = HH^T = D - N$ be the Laplacian matrix corresponding to the undirected graph $\mathcal{G}$, then we have rank(L) = rank$(HH^T) = M - 1$ if and only if $\mathcal{G}$ is connected.*

From this lemma it follows that $\ker(L) = \ker(H^T) = \text{span}\{\mathbb{1}\}$, where it should be obvious that $\mathbb{1} \in \ker(L)$ from $L = D - N$.

3.1.1 General consensus algorithms

The previous section covers some basic theory of graph theory. Yet another important concept in networks is that of the *stochastic matrix*; a type of matrix that arises naturally from weighted averaging in a multi-agent network. For example, whenever agents share information among their neighbors, then it is common that each agents take a weighted average of the information it received from its neighbors. This idea is also fundamental for *consensus algorithms for distributed averaging*. In its most simple and fundamental form, we can imagine a network of M agents, each agent knowing a scalar s^i, where the agents need to reach consensus about the average value $\bar{s} = \frac{1}{M}\sum_{i=1}^{M} s^i$. This problem is well-known and the literature that describes it is vast, e.g. see [17]. For example, a consensus can be reached iteratively by initiating $u^i_{(0)} = s^i$ for all i and executing the following update in discrete time:

$$u^i_{(t+1)} = \frac{1}{d_i + 1} \sum_{j \in \mathcal{N}_i^*} u^j_{(t)}, \quad \text{for } i = 1, \dots, M. \tag{3.6}$$

The only necessary and sufficient condition for uniform linear convergence towards some consensus variable $\hat{s}$, i.e. such that $u^i_{(t)} \to \hat{s}$ for all i, is connectedness of the underlying graph $\mathcal{G}$, which may be directed or undirected. In this case each agent does a straightforward averaging with equal weights $\frac{1}{d_i+1}$ for each neighbor. Now, unless the weights satisfy a particular property, the consensus variable $\hat{s}$ does not correspond to the actual average $\bar{s}$, but rather to some weighted average $\hat{s} = \sum_{i=1}^{M} q_i s^i$, with $q_i > 0$ and $\sum_{i=1}^{M} q_i = 1$. Before we consider this property, we consider a more general form of weighted averaging.

First of all, we generalize (3.6) by allowing arbitrary weights $w_{ij} \in (0, 1)$. Furthermore we assume that each agent has a state $x^i \in \mathbb{R}^n$ in stead of just a scalar s^i. Then the generalized update scheme, initiated by $y^i_{(0)} = x^i$, may be of the following form:

$$y^i_{(t+1)} = \sum_{j \in \mathcal{N}_i^*} w_{ij} y^j_{(t)}, \quad \text{for } i = 1, \dots, M, \tag{3.7}$$

where we require $\sum_{j \in \mathcal{N}_i^*} w_{ij} = 1$. By letting $w_{ij} = 0$ whenever $j \notin \mathcal{N}_i^*$, we define a matrix $W = [w_{ij}]$ with the property that $W\mathbb{1} = \mathbb{1}$. This matrix of weight factors is said to be *right-stochastic*. In fact, this property is a necessary and sufficient condition for any nonnegative matrix W to be right-stochastic, as is stated in the following lemma.

Lemma 3.4. *Let $W = [w_{ij}]$ be a nonnegative matrix, i.e. with each entry $w_{ij} \geq 0$. Then W is right-stochastic if and only if $W\mathbb{1} = \mathbb{1}$.*

By definition of W, we are able to give a compact formulation of the generalized consensus update scheme (3.7); we denote by $y_{(t)} = [y^i_{(t)}]$ the vectors of all states

$y^i_{(t)}$, such that we have:

$$y_{(t+1)} = (W \otimes I) y_{(t)}, \tag{3.8}$$

here $\otimes$ denotes the *Kronecker product*. The generalized consensus update still convergences uniformly and linearly towards some weighted average $\hat{x} = \sum_{i=1}^{M} q_i x^i$ if and only if the underlying graph $\mathcal{G}$ is connected, for any choice of W that satisfies the aforementioned. We will explain this some more by elaborating on stochastic matrices. Here we will also consider the particular property of W that guarantees convergence to the actual average $\bar{x} = \frac{1}{M} \sum_{i=1}^{M} x^i$, a result that is given in lemma 3.5.

3.1.2 Stochastic matrices

Stochastic matrices are important as they have the so-called *mixing property*, e.g. a weighted averaging (3.8) with a right-stochastic matrix W brings the new states $y^i_{(t+1)}$ closer together. More specifically, they bring all states closer to some global weighted average $\hat{x} = \sum_{i=1}^{M} q_i x^i$. Even more specific, for connected networks the matrix W has the property that $\rho(W) = 1$, where $\lambda = 1$ is an eigenvalue with algebraic multiplicity 1, corresponding to the eigenvector $\mathbb{1}$. Throughout this thesis, it is assumed that the eigenvalues are ordered as $1 = \lambda_1 > \lambda_2 \geq \cdots \geq \lambda_M > -1$. Now, it is straightforward to derive that:

$$W^t \to \mathbb{1} q^T \quad \text{as} \quad t \to \infty, \tag{3.9}$$

for some $q \in \mathbb{R}^M$ containing the weight factors q_i that define the weighted average $\hat{x}$; observe that the limit is a right-stochastic matrix as well, since $(\mathbb{1}q^T)\mathbb{1} = \mathbb{1}(q^T\mathbb{1}) = \mathbb{1}$. We mention that the linear rate of convergence is equivalent to that of $\lambda^t \to 0$ with $\lambda := \max\{|\lambda_2|, |\lambda_M|\}$, i.e. the convergence of the second largest eigenvalue, in absolute value, towards zero. However, we may be interested in the actual average $\bar{x}$, rather than some weighted average $\hat{x}$. A necessary and sufficient condition for the update rule (3.8) to converge to the actual average $\bar{x}$, is that W must be both left- and right-stochastic, or *doubly-stochastic* for short. That is, we require that W satisfies both $W\mathbb{1} = \mathbb{1}$ and $\mathbb{1}^T W = \mathbb{1}^T$. Considering equation (3.9), we conclude that the limit must be doubly-stochastic as well and we find that $\frac{1}{M}\mathbb{1}^T = \left(\frac{1}{M}\mathbb{1}^T\right)\left(\mathbb{1}q^T\right) = \left(\frac{1}{M}\mathbb{1}^T\mathbb{1}\right) q^T = q^T$. In this case the update (3.7) converges indeed to the average $\bar{x} = \frac{1}{M} \sum_{i=1}^{M} x^i$.

Lemma 3.5. *Consider the general consensus update (3.8). We have global convergence towards the actual average $\bar{x}$, i.e. $y_{(t)} \to (\bar{x} \otimes \mathbb{1})$ as $t \to \infty$, if and only if $\mathcal{G}$ is connected and W is doubly-stochastic. The rate of convergence is equivalent to that of $\lambda^t \to 0$, with $\lambda = \max\{|\lambda_2|, |\lambda_M|\}$.*

Since all agents are autonomous, they should be able to compute weight factors from local information. For a good mixing of the states, we prefer to have $|\lambda|$ as small as possible. As a general rule of thumbs, all agents should give their neighbors significant weights.[1] If they do not, then W gets closer to I and $|\lambda_2|$ gets closer to 1. In the remainder of this section, we consider some of the commonly used choices.

[1] In Example 4.2, Section 4.1.2, we observe that this rule of thumb does not necessarily apply to, e.g., complete bipartite graphs, since then the negative eigenvalue λ_M becomes arbitrary close to -1 when we allow the size of the graph to grow arbitrarily.

Uniform weights A straightforward approach is to let each agent define uniform weights, based on its number of neighbors, i.e. we let:

$$w_{ij} = \begin{cases} \frac{1}{d_i+1} & \text{if } j \in \mathcal{N}_i^*, \\ 0 & \text{otherwise.} \end{cases} \tag{3.10}$$

Notice that this choice of weights corresponds to the consensus update of equation (3.6). This approach is described in for example [18] and has generally relatively good properties, that is, with respect to the mixing properties that can be obtained from just local information. Generally, the uniform weights give a right-stochastic W.

Laplacian weights Another approach is to let the agents agree about the maximum degree $d_{\max} = \max\{d_1, \ldots, d_M\}$. Then let:

$$w_{ij} = \begin{cases} \frac{1}{d_{\max}} & \text{if } j \in \mathcal{N}_i, \\ 1 - \frac{d_i}{d_{\max}} & \text{if } j = i, \\ 0 & \text{otherwise.} \end{cases} \tag{3.11}$$

This approach is described in e.g. [19], but it has worse mixing properties than the uniform weights. A very similar approach is described in [20], called the *maximum degree* weights, where d_{max} is simply replaced by the upper-bound M for the maximum degree. This indeed does not benefit the mixing properties as well. In both cases this approach gives a right-stochastic W when the network is directed. However, W becomes symmetric doubly-stochastic when the network is undirected.

Metropolis-Hasting weights In complex directed networks it is a nontrivial task to obtain a doubly-stochastic W, since agents can not necessarily communicate back to their neighbors. Hence, column-wise summation to 1 can not be guaranteed without a very complex communication strategy among agents. It is generally accepted that it is only possible to achieve this in undirected networks. For example, in [21, (p. 56)] it is described how a doubly-stochastic matrix W can be obtained from any right-stochastic matrix R corresponding to an undirected network with significant weights (i.e. with $r_{ij} > 0$ whenever $j \in \mathcal{N}_i^*$). It is obtained by:

$$w_{ij} = \begin{cases} \min\{r_{ij}, r_{ji}\} & \text{if } j \in \mathcal{N}_i, \\ 1 - \sum_{k \in \mathcal{N}_i} w_{ik} & \text{if } j = i, \\ 0 & \text{otherwise.} \end{cases} \tag{3.12}$$

Observe that W is symmetric as well. Now suppose that R was the right-stochastic matrix obtained from the uniform weights, then we obtain a special case to this procedure that gives us the so-called *Metropolis-Hasting weights*. This approach is described in e.g. [19] and the weights are obtained directly (and locally) by:

$$w_{ij} = \begin{cases} \frac{1}{\max\{d_i, d_j\}+1} & \text{if } j \in \mathcal{N}_i, \\ 1 - \sum_{k \in \mathcal{N}_i} w_{ik} & \text{if } j = i, \\ 0 & \text{otherwise.} \end{cases} \tag{3.13}$$

Nonsingular weight matrices For completeness we mention that it always possible to obtain a nonsingular stochastic matrix $\tilde{W}$, with a positive spectrum, from any other stochastic matrix W, which is either right-, left- or doubly-stochastic [1, (p. 7)]. This is done by defining $\tilde{W} = \frac{1}{2}(W + I)$. This shift gives us eigenvalues $\tilde{\lambda}_i$ of $\tilde{W}$ satisfying $\tilde{\lambda}_i = \frac{1}{2}(\lambda_i + 1)$. Hence, we have $\tilde{\lambda}_1 = 1$ and $0 < \tilde{\lambda}_i < 1$ for any other i, i.e. W is indeed nonsingular with a positive spectrum. It is also readily checked that either right-, left- or doubly-stochastic properties are preserved by this transformation. We mention that this transformation is obtained directly and locally by:

$$\tilde{w}_{ij} = \begin{cases} \frac{1}{2}w_{ij} & \text{if } j \in \mathcal{N}_i \\ \frac{1}{2}(w_{ii} + 1) & \text{if } j = i \\ 0 & \text{otherwise.} \end{cases} \tag{3.14}$$

This transformation brings $\tilde{W}$ closer to I by a factor a 2 with respect to any norm $\|\cdot\|$, as we observe that $\|\tilde{W} - I\| = \|\frac{1}{2}(W + I) - I\| = \|\frac{1}{2}(W - I)\| = \frac{1}{2}\|W - I\|$. As mentioned before, this does not benefit the mixing properties.

In the next section we consider how the basic concepts of multi-agent networks may be used for solving distributed unconstrained minimization problems.

3.2 Distributed unconstrained minimization

In this section we consider the distributed version of unconstrained minimization problems, Section 2.2. We assume that f is written as the sum of some $M > 1$ convex functions $f_1, \ldots, f_M$, cf. (3.1). For simplicity we also assume that $f(x)$ is defined on the entire space $x \in \mathbb{R}^n$. Since f is the finite sum of convex functions, f is a convex functions itself. Now in a truly distributed environment, each agent i has in principle only access to the local information f_i. This imposes some fundamental problems. For example, if we consider the set $\mathcal{X}_i^* \subseteq \mathbb{R}^n$ of *local minimizers* for f_i in comparison to the set of *global minizers* $\mathcal{X}^* \subseteq \mathbb{R}^n$ for f, then it may well be that $\mathcal{X}_i^* \cap \mathcal{X}^* = \emptyset$. This is because the functions $f_1, \ldots, f_M$ are, in principle at least, independent. This shows us that individual minimization does not necessarily provides agents with useful information, since it can lead to an arbitrary bad approximation of the global minimizer.

There are various general approaches that we can follow, and many of such approaches have already been studied (one more extensively than the other). For example, we can try to iteratively exchange data among agents to obtain an approximate average of all the data, i.e. of all functions $f_1, \ldots, f_M$. This approach is for example followed in [2]. Clearly this average is, at least approximately, just a scalar multiple of f; at this point a local minimization corresponds to global minimization. For this approach, the exchange of data can be huge and it may not be a feasible approach. Furthermore, in some cases it might not even be allowed to exchange data [1].

In the former approach the simple consensus update of equation (3.8) is used as a tool for computing averaged data. This is in some sense the most straightforward approach, but it is not necessarily the most sensible one. Nonetheless the consensus update remains an important tool for many other distributed

algorithms. For example, if we want to reduce the amount of communication and keep the individual computations at a lower level, then perhaps a better approach is to do a two-step algorithm as follows:

$$\begin{aligned} y^i_{(t)} &= \phi^i(x^i_{(t)}) \\ x^i_{(t+1)} &= \sum_{j\in\mathcal{N}_i^*} w_{ij} y^j_{(t)}, \quad \text{for } i = 1,\dots,M. \end{aligned} \tag{3.15}$$

Here ϕ^i represents some individual update that gives, to some extent, a better approximation for a local minimizer of f_i in comparison to the previous iterate $x^i_{(t)}$. Typically this is done with some gradient descent like update. Then in the next step we allow the agents to communicate so that they can compute a weighted average of the of updated states of their neighbors. This approach is a trade-off between (i) iteratively distributing global information for computing average data and (ii) using only local information for local minimization. This approach is already extensively studied in many ways, some of these will be studied in Chapter 4. We mention that it is sometimes convenient to reverse the steps of this two-step algorithm:

$$\begin{aligned} y^i_{(t+1)} &= \sum_{j\in\mathcal{N}_i^*} w_{ij} x^j_{(t)} \\ x^i_{(t+1)} &= \phi^i(y^i_{(t)}), \qquad \text{for } i = 1,\dots,M. \end{aligned} \tag{3.16}$$

The two variants of the two-steps algorithm are very similar, especially in a fixed network topology. Hence, we may generally expect very similar convergence for both variants. In contrast, when the data is stochastic, it is shown that reversion of these steps can have significant impact on the convergence behavior [22, 23].

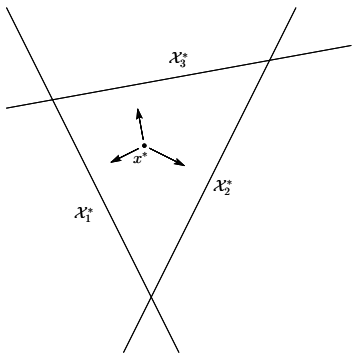


Figure 3.1: Schematic representation of a distributed environment with three agents with corresponding optimal solution spaces $\mathcal{X}_i^*$ for $i = 1, 2, 3$, which are illustrated schematically by straight lines. The set of global minimizer is represented by a unique point x^*, i.e. $\mathcal{X}^* = \{x^*\}$. The arrows represent the divergence from the set $\mathcal{X}^*$ by the individual agents i towards their local optimal solution space $\mathcal{X}_i^*$, as it occurs for the update schemes of equations (3.15), (3.16)

Many of these approaches seem to work reasonably well and they are sometimes preferred for their relatively cheap iterations, both in terms of computational and

communicational effort. However, we mention that this approach is inherently problematic for one important reason: If $\mathcal{X}_i \cap \mathcal{X}^* = \emptyset$ for all i, which may be realistic in many practical problems, then even if we would initiate the procedure with $x^i_{(0)} = x^*$ for all i, with $x^* \in \mathcal{X}^*$, we would diverge from the set $\mathcal{X}^*$ in the first iteration. This is illustrated schematically in figure 3.1. We observe that the global minimizer x^* generally does not correspond to a consensus result. In the best case, we may hope to find an appropriate choice for ϕ^i such that we converge to a reasonably close environment of the global minimizer.

The update schemes of equations (3.15), (3.16) is the fundamental form that is of interest in this thesis. This general update scheme has been studied extensively and we consider some of the most important ones in Chapter 4.

3.2.1 Distributed quadratic minimization

As a special case we consider distributed quadratic minimization problems, which are of main interest in this thesis. Even more specifically, we consider such problems in the context of linear least squares problems for overdetermined system $Ax = b$, where we have $A \in \mathbb{R}^{m\times n}$ with $m > n$ and $\operatorname{rank}(A) = n$. We recall from Section 2.2.1 that the corresponding quadratic function is given by $f(x) = \frac{1}{2}\|b - Ax\|_2^2$. Now suppose that the matrix A and the right-hand side b are partitioned row-wise over M components, i.e. we have matrices $A_i \in \mathbb{R}^{m_i\times n}$ and left-hand sides $b_i \in \mathbb{R}^{m_i}$ with $m_i \leq n$ and $\sum_{i=1}^M m_i = m$, such that:

$$A = \begin{bmatrix} A_1 \\ \vdots \\ A_M \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} b_1 \\ \vdots \\ b_M \end{bmatrix}. \tag{3.17}$$

Here we will always assume, for simplicity, that $\operatorname{rank}(A_i) = m_i$ for all i. It is straightforward to derive that:

$$\begin{aligned} f(x) &= \frac{1}{2}\|b - Ax\|_2^2 \\ &= \sum_{i=1}^M \frac{1}{2}\|b_i - A_i x\|_2^2 \\ &=: \sum_{i=1}^M f_i(x) \end{aligned} \tag{3.18}$$

Here f is a strict convex function by lemma 2.5 since $A^TA > 0$. Similarly, each f_i is a convex function since $A_i^TA_i \geq 0$. In this way the distributed linear least squares problem constitutes a distributed unconstrained minimization problem. These problems, where each of M agents knows only the part $[A_i \;\; b_i]$ of the partitioned matrix $[A \;\; b]$, arise naturally in for example sensor networking. Here the partitioned matrix may correspond to a observation measurements of the form $b_i = A_i x + \eta_i$, where η_i is a zero mean term that models the measurement noise [22, 24]. The general occurrence that $\mathcal{X}_i^* \cap \mathcal{X}^* = \emptyset$ remains an inherent issue in the context of distributed quadratic minimization problems. This is illustrated in the following example.

Example 3.1
In this example we consider a case with $A \in \mathbb{R}^{3\times 2}$, $b \in \mathbb{R}^3$ and $M = 3$, i.e. each agents i knows a row $A_i = a_i$ and a scalar b_i. Here we have:

$$A = \begin{bmatrix} 10 & 5 \\ -10 & 5 \\ -2 & 11 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} 10 \\ -11 \\ 14 \end{bmatrix}.$$

Notice that $a_i x = b_i$ has infinitely many solutions, whereas $Ax = b$ is inconsistent since $b \notin R(A)$. Clearly we have $\mathcal{X}^* = \{x^*\}$ with $x^* = (A^T A)^{-1} A^T b$; the example is constructed such that $x^* = \mathbb{1}$. For each agent the set of local minimizers are $\mathcal{X}_i = \{x \in \mathbb{R}^2 \mid a_i x = b_i\}$, which are straight lines in the two-dimensional plane. The inherent problem that was described in the previous section follows, since we have $\mathcal{X}_i^* \cap \mathcal{X}^* = \emptyset$ for all i. The schematic representation in figure 3.1

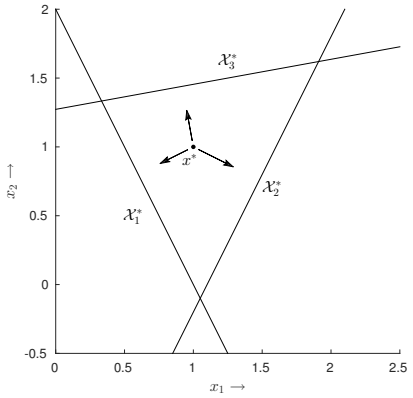


Figure 3.2: Distributed environment with three agent where each agents has information $[A_i \;\; b_i]$. Points in the two-dimensional plane are denoted by $x = [x_1 \;\; x_2]^T$. The local optimal solution spaces $\mathcal{X}_i^*$ for $i = 1, 2, 3$ are straight lines, whereas the global minimizer x^* lies in the region enclosed by these three lines.

actually corresponds precisely to this least squares problem, which is depicted for clarity in figure 3.2 above. ■

4 Distributed algorithms

In this chapter, we will review some of the existing methods for distributed convex optimization. The literature on such methods is vast, hence we have made a selection of methods that are relevant for this thesis. The outline of this chapter will be as follows. First, in Section 4.1, we consider two methods that are suitable for general distributed unconstrained minimization problems such as described in Section 3.2. We will observe that the method of Section 4.1.2 is actually very suitable for the distributed least squares problems as described in Section 3.2.1. Next, in Section 4.2, we will consider some methods that are designed specifically for either distributed least squares problems or distributed consistent linear systems.

4.1 General distributed unconstrained optimization

In this section we consider two algorithms for general unconstrained optimization problems. Before we continue, we review some conventions that are common to many distributed algorithms, and which are important throughout this chapter. One convention deals with asynchronous operations and another deals with connectedness of time-dependent networks.

In a truly distributed environment, it is generally very difficult to synchronize the communication. This is indeed typical for many practical applications. For this reason, it is often assumed that each agents operates at individual discrete times t_k, in stead of common discrete times $t = 0, 1, 2, \ldots$ (or $t_k = k$). The iterates are then simply denoted by the counter k, indicating the kth individual discrete time-interval $[t_{k-1}, t_k]$. Agent i then uses at iteration k the information that is available at time t_k.

In the context of time-dependent networks, it makes sense to assume some form of connectedness for the sequence of neighbor graphs $\mathcal{G}_{(t_k)}$; each individual graphs $\mathcal{G}_{(t_k)}$ may not be necessarily connected, however, the sequence do need to be such that all agents eventually receive information from all other agents, either directly or indirectly. Obviously, this needs to happen infinitely often. This idea is fundamental in the literature and it is considered to be a minimal assumption on connectivity [21]. This type of connectivity is sometimes called *repeatedly jointly strongly connectedness*, e.g. [24]. Formally this can be described as follows, for both directed and undirected graphs. We denote by $E_{(k)}$ the set of edges of the graph $\mathcal{G}_{(t_k)}$ and define the set E_∞ of edges that represent the agent pairs j, i that communicate directly infinitely many times, i.e.

$$E_\infty = \{(j, i) \mid (j, i) \in E_{(k)} \text{ for infinitely many indices } k\}. \tag{4.1}$$

For repeatedly jointly strongly connectedness, E_∞ should of course be (strongly) connected, however, these relations contained in E_∞ must occur repeatedly as well. That is, all edges $(j,i) \in E_\infty$ should exists over infinitely many bounded intervals. This is formulated in the following definition.

Definition 4.1 (repeatedly jointly strongly connected)**.** *The sequence of graphs $\mathcal{G}_{(t_k)}$, either directed or undirected, is repeatedly jointly strongly connected if the following relations hold.*

(i) the edge set E_∞ is (strongly) connected;

(ii) there exists a smallest integer $B \geq 1$ such that:

$$(j,i) \in E_{(k)} \cup E_{(k+1)} \cup \cdots \cup E_{(k+B-1)},$$

for all $(j,i) \in E_\infty$ and $k \geq 0$.

In this way we ensure that each agent receives, either directly or indirectly, information from all agents at least once every B consecutive time-intervals; after all, the graph with the edge set $E_{(k)} \cup E_{(k+1)} \cup \cdots \cup E_{(k+B-1)} \supseteq E_\infty$ is (strongly) connected for any $k \geq 0$. It is said that the topology has a *bounded intercommunication interval* B.

4.1.1 Distributed subgradient

We consider the *distributed subgradient method* described by A. Nedić and A. Ozdaglar in [21]. This paper provides an approach for the general update scheme (3.15) that is applicable to a very broad range of distributed unconstrained minimization problems. The method is designed for the distributed minimization of $f(x) = \sum_{i=1}^{M} f_i(x)$, where each agent has a locally known, different, convex and possibly non-smooth function $f_i(x)$. Furthermore, the paper claims to allow for asynchronous communication among neighbors, i.e. each agent is allowed to have its "own" discrete time sequence $t_0, t_1, t_2, \ldots$ at which it operates, according to a changing network topology $\mathcal{G}_{(t_k)}$. However, it appears from the analysis that each agent actually operates at a common time sequence and we choose to simplify this; we let the agents operate at the same discrete time sequence $t_k = k$.

The term *subgradient*, that partially describes this method, emphasizes that it allows for non-smooth functions, i.e. functions for which the gradient may not be well-defined. The gradient $\nabla f(x)$ of a smooth convex functions has the desirable property given by the first-order convexity condition in lemma 2.4; for non-smooth convex functions f we may define a subgradient $s_f(x)$ as a vector that has this same property. That is, $s_f(x)$ is a subgradient if:

$$f(x) + s_f(x)^T(x' - x) \leq f(x'), \tag{4.2}$$

for all $x' \in C$, where C is the convex domain of f. The method then uses any such subgradient $s_f(x^i_{(t)})$ as a descent direction $d^i_{(t)}$ for each agent i, at time t_k; in this way it allows for non-smooth functions. In the most general form, the paper proposes that each agent i iteratively does the following update:

$$x^i_{(t+1)} = \sum_{j=1}^{M} w_{ij,(t)} x^j_{(t)} - \alpha^i_{(t)} d^i_{(t)}. \tag{4.3}$$

Observe that the two steps of the general update (3.15) are combined in a single step update. The term $w_{ij,(t)}$ indicates that the underlying structure is that of a time-varying topology; here it is assumed that the time-varying topology is a repeatedly jointly connected one. Implicitly the update (4.3) defines a sequence of time-dependent right-stochastic matrices $W_{(t)}$. Here it is assumed that the stochastic weights are significant for neighboring agents, i.e. that $w_{ij,(t)} \geq \eta > 0$ for all $j \in \mathcal{N}^*_{i,(t)}$, and 0 otherwise. For a sequence of right-stochastic matrices it is shown that:

$$W_{(t)}W_{(t-1)} \cdots W_{(0)} \to \mathbb{1}q^T,$$

where $q \in \mathbb{R}^n$ is a stochastic vector (i.e. $q^T\mathbb{1} = 1$). The convergence is linear with a convergence factor $(1 - \eta^{B(M-1)}) < 1$, see [21, Lemma 4]. Then it is shown that same linear convergence towards the uniform steady state matrix $\frac{1}{M}\mathbb{1}\mathbb{1}^T$ is obtained for a sequence of doubly-stochastic matrices, which is the main motivation for the choice of doubly-stochastic matrices. Note that this requires an undirected network topology, as is discussed in Section 3.1.2.

In the remainder of this section we simplify the results from [21] by assuming a time-independent connected network topology, with a fixed doubly-stochastic matrix W. Observe that this is a special case of repeatedly jointly connectedness with $B = 1$. Furthermore, the update (4.3) is simplified in the paper itself by assuming a common stepsize α among neighbors. The update we consider is thus of the following form:

$$x^i_{(t+1)} = \sum_{j=1}^{M} w_{ij}x^j_{(t)} - \alpha d^i_{(t)}. \tag{4.4}$$

We introduce the notation w^t_{ij} for the elements of the matrix W^t. With this notation it is observed that the states $x^i_{(t+1)}$ satisfy the following relation:

$$x^i_{(t+1)} = \sum_{j=1}^{M} w^t_{ij}x^j_{(0)} - \alpha \sum_{s=0}^{t-1} \left(\sum_{j=1}^{M} w^{t-s}_{ij} d^j_{(s)} \right) - \alpha d^i_{(t)}.$$

This expression gives rise to an explicit expression for the average of all states, $\bar{x}_{(t+1)} = \frac{1}{M}\sum_{j=1}^{M} x^j_{(t+1)}$, in terms of the initial guesses $x^i_{(0)}$ and all subgradients $d^i_{(s)}$ that are computed up to iteration $t + 1$:

$$\bar{x}_{(t+1)} = \frac{1}{M}\sum_{j=1}^{M} x^j_{(0)} - \frac{\alpha}{M}\sum_{s=0}^{t} \left(\sum_{j=1}^{M} d^j_{(s)} \right).$$

This expression satisfies the convenient and insightful recurrence relation that follows:

$$\bar{x}_{(t+1)} = \bar{x}_{(t)} - \frac{\alpha}{M}\sum_{j=1}^{M} d^j_{(t)}. \tag{4.5}$$

This expression is insightful since it can be interpreted as a global update scheme for the entire network, where $\sum_{j=1}^{M} d^j_{(t)}$ approximates the gradient $\nabla f(\bar{x}_{(t)}) = \sum_{j=1}^{M} \nabla f_i(\bar{x}_{(t)})$. In other words, the global update scheme is approximately that of a gradient descent method.

The convergence of the distributed subgradient method is proven under the additional assumptions that $\mathcal{X}^*$ is nonempty and that the considered subgradients $d^j_{(t)}$ are bounded, i.e. that $\|d^j_{(t)}\|_2 \leq D$ for some $D > 0$. Furthermore, there are two quantities important in the convergence proof; these are averaged vectors over the t iterations as follows:

$$\hat{x}_{(t)} = \frac{1}{t}\sum_{s=0}^{t-1} \bar{x}_{(s)} \quad \text{and} \quad \hat{x}^i_{(t)} = \frac{1}{t}\sum_{s=0}^{t-1} x^i_{(s)}.$$

By introducing the notation $f^* := f(x^*)$ for any $x^* \in \mathcal{X}^*$, the convergence theorem (for time-independent, undirected connected networks) is formulated as follows [21, Proposition 3], where we recall the definition of the disctance function from Section 1.2.

Theorem 4.2 (Distributed subgradient method)**.** *Let the distributed subgradient method (4.4) be executed by a time-independent, undirected and connected network, with a corresponding fixed doubly-stochastic matrix W containing significant weights (i.e. W is such that $w_{ij} \geq \eta$ whenever $j \in \mathcal{N}^*_i$, and 0 otherwise). Let $\mathcal{X}^*$ be nonempty and subgradients be bounded by D. Furthermore, let the choice of α be such that it satisfies the following for the initial guesses $x^j_{(0)}$:*

$$\max \|x^j_{(0)}\|_2 \leq \alpha D. \tag{4.6}$$

Then we have, with $\bar{x}_{(t)}$ generated by (4.5), the following.

(i) For every agent's state $x^i_{(t)}$, a uniform upper bound from the difference with the average state $\bar{x}_{(t)}$ is given by:

$$\|\bar{x}_{(t)} - x^i_{(t)}\|_2 \leq 2\alpha D C_0,$$

for all $t \geq 0$, with:

$$C_0 = 1 + \frac{m}{1 - \left(1 - \eta^{(M-1)}\right)^{1/(M-1)}} \frac{1 + \eta^{-(M-1)}}{1 - \eta^{(M-1)}}.$$

(ii) For the averaged vectors $\hat{x}_{(t)}$ and $\hat{x}^i_{(t)}$, we have the following upper bounds for the values attained by the objective function f:

$$f(\hat{x}_{(t)}) \leq f^* + \frac{M\, dist^2\left(\bar{x}_{(0)}, \mathcal{X}^*\right)}{2\alpha t} + \frac{\alpha D^2 C}{2},$$

and:

$$f(\hat{x}^i_{(t)}) \leq f^* + \frac{M\, dist^2\left(\bar{x}_{(0)}, \mathcal{X}^*\right)}{2\alpha t} + \alpha D^2 \left(\frac{C}{2} + 2MC_0\right),$$

where $C = 1 + 8MC_0$.

Part (i) of the theorem shows that the difference between any agent's state $x^i_{(t)}$ and the global average state $\bar{x}_{(t)}$ is bounded above by a constant that is proportional to the stepsize α. Observe that both C and C_0 are constants related to the matrix W and D is a constant that bounds the subgradients. Part (ii)

provides upper bounds for the difference from the objective function values $f(\hat{x}_{(t)})$, $f(\hat{x}^i_{(t)})$ and the optimal value f^*. Here, the first term in the difference is inversely proportional to α and converges to 0 at rate $1/t$ and the second term is a non-diminishing constant proportional to α. We mention that a condition on α for the convergence results are given by equation (4.6).

The convergence theorem shows that exact convergence is not possible with a fixed α; in stead there is a trade-off between the accuracy of the approximate solution and the computational work load that is required to generate such a solution. Roughly speaking, if we reduce α by a positive factor $c < 1$, then the workload required to generate a approximate solution, that is more accurate by a factor c, increases with a factor $1/c > 1$. We mention that the convergence is with rate $\mathcal{O}(1/t)$ towards an $\mathcal{O}(\alpha)$-environment of the optimal solution x^*.

4.1.2 Decentralized gradient descent

Next we consider the *decentralized gradient descent* method described K. Yuan, Q. Ling and W. Yin in [1]. This method is quite similar to the distributed subgradient method, however, the method assumes smooth convex functions $f_i(x)$ for all i. Nonetheless it is interesting to consider this method, since it is shown under additional assumptions that the method converges linearly to an $\mathcal{O}(\alpha)$-environment of the optimal solution x^*, which is more favorable then the convergence rate $\mathcal{O}(1/t)$ of the distributed subgradient method. This is especially interesting for the purpose of this thesis, because the additional assumptions apply to the case of distributed linear least squares minimization.

The decentralized gradient descent method operates synchronously under a fixed network topology with a symmetric doubly stochastic matrix W. The method assumes smooth convex functions f_i such that the negative gradient $-\nabla f_i$ can be used as a descent direction, rather than some unknown subgradient. Otherwise the update scheme is the same as that from the distributed subgradient method with fixed stepsize α:

$$x^i_{(t+1)} = \sum_{j=1}^{M} w_{ij} x^j_{(t)} - \alpha \nabla f_{i,(t)}, \tag{4.7}$$

here we introduced the notation $\nabla f_{i,(t)} := \nabla f_i(x^i_{(t)})$ for the sake of brevity. In this part we explicitly assume that the initial guesses are zero, i.e. that $x^i_{(0)} = 0$ for all i.

First of all, [1] shows convergence to an $\mathcal{O}(\alpha)$-environment of some $x^* \in \mathcal{X}^*$ with a convergence rate $\mathcal{O}(1/t)$, similar to the distributed subgradient method. Then, linear convergence to such an environment is proven under additional assumptions, under which the optimal solution x^* is unique. In both cases we require bounded gradients, again similar to the distributed subgradient method. Instead of assuming that the gradients are bounded over the domain of f_i, it is shown that the gradients remain bounded for small enough stepsizes α, under the assumptions that each f_i has a Lipschitz continuous gradient ∇f_i with finite constant L_{f_i}, i.e. we require:

$$\|\nabla f_i(x_1) - \nabla f_i(x_2)\|_2 \leq L_{f_i} \|x_1 - x_2\|_2, \tag{4.8}$$

for any x_1, x_2 on the domain of f_i. In example 4.1, we observe that this assumption is more convenient in the context of linear least squares minimization. We mention that it is straightforward to derive that ∇f is Lipschitz continuous with finite constant $L_f = \sum_{i=1}^{M} L_{f_i}$. Now, recall that we assume that the eigenvalues of W are numbered according to a non-increasing ordering, i.e. we have $1 = \lambda_1(W) > \lambda_2(W) \geq \cdots \geq \lambda_M(W) > -1$. Furthermore, we define $g_{(t)} := [\nabla f_{i,(t)}]$ and $L_g := \max_i\{L_{f_i}\}$ and require that the stepsize α is restricted by:

$$\alpha \leq \frac{1 + \lambda_M(W)}{L_g}. \tag{4.9}$$

At this point the paper introduces the auxiliary function that follows:

$$\xi_\alpha\left([x^i]\right) = -\frac{1}{2}\sum_{i,j=1}^{M} w_{ij}(x^i)^T x^j + \sum_{i=1}^{M}\left(\frac{1}{2}\|x^i\|_2^2 + \alpha f_i(x^i)\right), \tag{4.10}$$

where it is observed that the part $\frac{1}{2}\left(\sum_{ij} w_{ij}(x^i)^T x^j - \sum_i \frac{1}{2}\|x^i\|_2^2\right)$ is convex and nonnegative due to the spectral properties of W. Additionally, it is Lipschitz continuous with constant $L_{\xi_\alpha} = (1 - \lambda_M) + \alpha L_g$. An important observation with the auxiliary function, is that the update scheme (4.7) can be written as:

$$x^i_{(t+1)} = x^i_{(t)} - \nabla_i \xi_\alpha\left([x^i_{(t)}]\right).$$

With this relation, it can be shown with zero initial guesses and α restricted by (4.9), that the gradients are always bounded by [1, Theorem 1]:

$$\|g_{(t)}\|_2 \leq \sqrt{2L_g \sum_{i=1}^{M}\left(f^i_{(0)} - f^*\right)} := D, \tag{4.11}$$

here we assume that $\mathcal{X}^*$ is nonempty.

We omit the very similar convergence result with rate $\mathcal{O}(1/t)$ for general convex smooth functions, since we are specifically interested in the potential linear convergence rate of the method. This is achieved for global functions f that are *strongly convex* with modulus $\mu_f > 0$, i.e. for f such that:

$$\left(\nabla f(x_1) - \nabla f(x_2)\right)^T (x_1 - x_2) \geq \mu_f \|x_1 - x_2\|_2^2, \tag{4.12}$$

for any x_1, x_2 in the domain of f. If for example all f_i are strongly convex with modulus μ_{f_i}, then it is straightforward to derive that f is strongly convex with modulus $\mu_f = \sum_{i=1}^{M} \mu_{f_i}$. The strongly convex property implies strict convexity of the global function f, such that the optimal solution x^* is observed to be unique. However, for more general f with $|\mathcal{X}^*| > 1$ the paper provides a specific case for which linear convergence is still achieved, namely for the case where f is *restricted strongly convex* with modulus $\nu_f > 0$. These are functions such that:

$$\left(\nabla f(x) - \nabla f(x^*_P)\right)^T (x - x^*_P) \geq \nu_f \|x - x^*_P\|_2^2, \tag{4.13}$$

for any x in the domain of f. Here $x^*_P \in \mathcal{X}^*$ is the projection of x onto $\mathcal{X}^*$, i.e. it is the vector such that $\text{dist}(x, \mathcal{X}^*) = \|x - x^*_P\|_2$. Observe that $\nabla f(x^*_P) = 0$

by the result of lemma 2.7. In this case, if each f_i is restricted strongly convex, then f is not necessarily restricted strongly convex as well. This follows since $\mathcal{X}_i^* \cap \mathcal{X}^* = \emptyset$ may be true for any i.

In the following example we will consider the main assumptions for the decentralized gradient descent method in the context of distributed linear least squares minimization.

Example 4.1 (properties of distributed linear least squares problems) Recall that in the context of distributed linear least squares minimization, we have $f(x) = \frac{1}{2}\|b - Ax\|_2^2$ and $f_i(x) = \frac{1}{2}\|b_i - A_i x\|_2^2$ (cf. equation (3.18)). Here $[A_i\ b_i]$ are the distributed row-wise parts of $[A\ b]$ as indicated by equation (3.17). We recall that $A \in \mathbb{R}^{m\times n}$ with $m > n$ and $\text{rank}(A) = n$, and that $A_i \in \mathbb{R}^{m_i \times n}$ with $m_i < n$ and $\text{rank}(A_i) = m_i$.

We will first show *Lipschitz continuity* for the gradients $\nabla f, \nabla f_i$. From equation (2.12) it is straightforward to derive that:

$$\nabla f(x) = A^T Ax - A^T b \quad \text{and} \quad \nabla f_i(x) = A_i^T A_i x - A_i^T b_i. \tag{4.14}$$

Here we recall that $A^T A > 0$ and observe that $A_i^T A_i \geq 0$, such that the eigenvalues of $A^T A$ and $A_i^T A_i$ are respectively positive and nonnegative; for $A^T A$ we assume m distinct ordered eigenvalues denoted by $\lambda_1(A^T A) > \cdots > \lambda_m(A^T A) > 0$, similarly for $A_i^T A_i$ we assume m_i nonzero and distinct ordered eigenvalues denoted by $\lambda_1(A_i^T A_i) > \cdots > \lambda_m(A_i^T A_i) > 0$. Then we have:

$$v^T (A^T A)^k v \leq \lambda_1^k (A^T A)\, v^T v \quad \text{and} \quad v^T (A_i^T A_i)^k v \leq \lambda_1^k (A_i^T A_i)\, v^T v,$$

for any $v \in \mathbb{R}^n$ and any integer $k \geq 1$. We then observe that:

$$\begin{aligned} \|\nabla f_i(x_1) - \nabla f_i(x_2)\|_2^2 &= \|A_i^T A_i (x_1 - x_2)\|_2^2 \\ &= (x_1 - x_2)^T \left(A_i^T A_i\right)^2 (x_1 - x_2) \\ &\leq \lambda_1^2 (A_i^T A_i)\, (x_1 - x_2)^T (x_1 - x_2) \\ &= \lambda_1^2 (A_i^T A_i)\, \|x_1 - x_2\|_2^2. \end{aligned}$$

Hence, we observe that each ∇f_i is Lipschitz continuous with finite constant $L_{f_i} = \lambda_1(A_i^T A_i)$. Similarly, f is Lipschitz continuous with $L_f = \lambda_1(A^T A)$. Alternatively, we may conclude from an earlier observation that f is Lipschitz continuous with $L_f = \sum_{i=1}^M \lambda_1(A_i^T A_i)$. This could also been observed from a property of the sum of Hermitian matrices, see [25, (p.175)], which in the case of $A^T A = \sum_{i=1}^M A_i^T A_i$ states that:

$$\lambda_1(A^T A) \leq \sum_{i=1}^M \lambda_1(A_i^T A_i).$$

By definition of $L_f = \lambda_1(A^T A)$ we obtain a tight bound for the property of Lipschitz continuity, in contrast to the definition $L_f = \sum_{i=1}^M \lambda_1(A_i^T A_i)$. However, the latter definition is more suitable for distributed computations.

A similar derivation shows that ∇f is *strongly convex* with modulus $\mu_f = \lambda_n(A^TA)$:

$$\begin{aligned}\big(\nabla f(x_1) - \nabla f(x_2)\big)^T(x_1 - x_2) &= \big(A^TA(x_1 - x_2)\big)^T(x_1 - x_2)\\ &= (x_1 - x_2)^TA^TA(x_1 - x_2)\\ &\geq \lambda_n(A^TA)\,(x_1 - x_2)^T(x_1 - x_2)\\ &= \lambda_n(A^TA)\,\|x_1 - x_2\|_2^2,\end{aligned}$$

for any $x_1, x_2 \in \mathbb{R}^n$. Here the inequality follows because A^TA has a full set of (nonzero) eigenvalues. The above derivation shows us that ∇f_i is not strongly convex, since there exists $x_1, x_2 \in \mathbb{R}^n$ such that $x_1 - x_2 \in \ker(A_i^TA_i) = \ker(A_i)$. However, we can show that f_i is *restricted strongly convex* with modulus $\nu_{f_i} = \lambda_{m_i}(A_i^TA_i)$. To this end we follow Section 5.1. Here we show that there exist a unique $z_i^* \in R(A_i^T)$ such that $A_iz_i^* = b_i$. Then we have $f_i(z_i^*) = 0$ such that $\mathcal{X}_i^* = \{z_i^* + u \mid u \in \ker(A_i)\}$. Furthermore we use that any $x \in R^n$ can be uniquely written as $x = z + u$ for some $z \in R(A_i^T)$ and $u \in \ker(A_i)$, where $z \perp u$. From this it follows that $x_P^* = z_i^* + u$ for any $x = z + u \in \mathbb{R}^n$, and hence that $x - x_P^* = z - z_i^* \in R(A_i^T)$ for any $x = z + u \in \mathbb{R}^n$. Now, by a very similar derivation we may see that f_i is indeed *restricted strongly convex* with modulus $\nu_{f_i} = \lambda_{m_i}(A_i^TA_i)$:

$$\begin{aligned}\big(\nabla f_i(x) - \nabla f_i(x_P^*)\big)^T(x - x_P^*) &= \big(A_i^TA_i(x - x_P^*)\big)^T(x - x_P^*)\\ &= (z - z_i^*)^TA_i^TA_i(z - z_i^*)\\ &\geq \lambda_{m_i}(A_i^TA_i)\,(z - z_i^*)^T(z - z_i^*)\\ &= \lambda_{m_i}(A_i^TA_i)\,\|x - x_P^*)\|_2^2.\end{aligned}$$

The inequality follows since the eigenvectors of $A_i^TA_i$, corresponding to its nonzero eigenvalues, are all in $R(A_i^T)$. This is explained in more detail in Section 5.1. We finally mention that this example shows that a finite sum of restricted strongly convex functions can actually be strongly convex. ■

We continue with the linear convergence result of the decentralized gradient descent method. We recall the definition $\bar{x}_{(t)} = \frac{1}{M}\sum_{j=1}^M x_{(t)}^j$ and define the average function $\bar{f}$ in a similar manner:

$$\bar{f}(x) = \frac{1}{M}\sum_{i=1}^M f_i(x), \tag{4.15}$$

which simply satisfies $f = M\bar{f}$, such that both functions share the solution space $\mathcal{X}^*$. Clearly $\bar{f}$ is Lipschitz continuous with finite constant $L_{\bar{f}} = \frac{1}{M}L_f = \frac{1}{M}\sum_{i=1}^M L_{f_i}$. We can now formulate the result of [1, Lemma 1].

Lemma 4.3. *Suppose that $\nabla\bar{f}$ is Lipschitz continuous with constant $L_{\bar{f}}$. We then have:*

$$(x - x_P^*)^T\left(\nabla\bar{f}(x) - \bar{f}(x_P^*)\right) \geq c_1\|\nabla\bar{f}(x) - \bar{f}(x_P^*)\|_2^2 + c_2\|x - x_P^*\|,$$

(where $x_P^* \in \mathcal{X}^*$ is the projection of x on $\mathcal{X}^*$) for the following two cases with corresponding definitions for c_1, c_2:

(i) $\bar{f}$ *is strongly convex with modulus* $\mu_{\bar{f}}$*; then* $c_1 = \frac{1}{\mu_{\bar{f}}+L_{\bar{f}}}$ *and* $c_2 = \frac{\mu_{\bar{f}}L_{\bar{f}}}{\mu_{\bar{f}}+L_{\bar{f}}}$.

(ii) $\bar{f}$ *is restricted strongly convex with modulus* $\nu_{\bar{f}}$*; then* $c_1 = \frac{\theta}{L_{\bar{f}}}$ *and* $c_2 = (1-\theta)\nu_{\bar{f}}$ *for any* $\theta \in [0,1]$.

This lemma is used to determine the linear convergence rates for both the unrestricted- and the restricted case. To this end we define yet another quantity by $x^*_{(t)} \in \mathcal{X}^*$ as the projection of $\bar{x}_{(t)}$ onto $\mathcal{X}^*$, for any t. Note that this quantity can be used to formulate convergence results for both the unrestricted and restricted strongly convex case, where it simplifies to the unique minimizer x^* in the former case. Then the linear convergence result is formulated as follows [1, Theorem 4].

Theorem 4.4 (Decentralized gradient descent method)**.** *Let the decentralized gradient descent method (4.7) be executed by a time-independent, undirected and connected network, with a corresponding fixed symmetric doubly-stochastic matrix* W *containing significant weights (i.e.* W *is such that* $w_{ij} \neq 0$ *whenever* $j \in \mathcal{N}_i^*$*, and* 0 *otherwise). Let* f *be either strongly convex with modulus* μ_f *or restricted strongly convex with modulus* ν_f*. Let* f_i *be Lipschitz continuous with constant* L_{f_i}*. Furthermore, let the stepsize* α *be restricted by:*

$$\alpha \leq \min\left\{\frac{1+\lambda_M(W)}{L_g}\,,\, c_1\right\},$$

then we have:

(i) global linear convergence with:

$$\|x^*_{(t+1)} - \bar{x}_{(t+1)}\|_2^2 \leq c_3^2\|x^*_{(t)} - \bar{x}_{(t)}\|_2^2 + c_4^2;$$

(ii) uniform linear convergence with:

$$\|x^*_{(t+1)} - x^i_{(t+1)}\|_2 \leq c_3^t\|x^*_{(0)}\|_2 + \frac{c_4}{\sqrt{1-c_3^2}} + \frac{\alpha D}{1-\beta},$$

here c_1, c_2 *are given by lemma 4.3 with* $\mu_{\bar{f}} = \mu_f/M$ *and* $\nu_{\bar{f}} = \nu_f/M$*,* D *is given by equation (4.11) and* $\beta = \max\{|\lambda_2(W)|, |\lambda_M(w)|\} < 1$*. Furthermore we have:*

$$c_3^2 = 1 - \alpha c_2 + \alpha\delta - \alpha^2\delta c_2 \quad \textit{and} \quad c_4^2 = \alpha^3(\alpha + \delta^{-1})\frac{L_g^2 D^2}{(1-\beta)^2},$$

where $L_g = \max_i\{L_{f_i}\}$ *and* $\delta > 0$ *is any positive constant. In particular, if we set* $\delta = \frac{c_2}{2(1-\alpha c_2)}$*, then we have:*

$$c_3 = \sqrt{1 - \frac{\alpha c_2}{2}} \in (0,1),$$

such that global convergence can be characterized by:

$$\|x^*_{(t+1)} - \bar{x}_{(t+1)}\|_2 \leq c_3^t\|x^*_{(0)}\|_2 + \mathcal{O}\left(\frac{\alpha}{1-\beta}\right).$$

As a result, the method converges linearly to an $\mathcal{O}(\frac{\alpha}{1-\beta})$-environment of either the unique solution x^* or the optimal solution set $\mathcal{X}^*$, depending on whether f is either unrestricted or restricted strongly convex. With regard to the stepsize condition we make the following observation for the strongly convex case. This is in our interest since we observed that $f(x) = \frac{1}{2}\|b - Ax\|_2^2$ is strongly convex. For the strongly convex case we require (cf. lemma 4.3 and theorem 4.4):

$$\alpha \leq \min\left\{ \frac{1 + \lambda_M(W)}{L_g} \, , \, \frac{1}{\mu_{\bar{f}} + L_{\bar{f}}} \right\}. \tag{4.16}$$

We additionally mention in Section 3.1 that we can design a nonsingular W with a positive spectrum, such that we can neglect the term $\lambda_M(W)$ to simplify the condition on α.

Observe that it is not straightforward to determine the smallest term in a truly distributed environment. We resolve this issue by deriving the reasonable lower bound $\frac{1}{2L_g}$ for this minimum; here we mention that is relatively easy to compute the term $L_g = \max_i\{L_{f_i}\}$ in a distributive manner, assuming that each agent is able to determine its own Lipschitz constant L_{f_i}; for example, if the agents know the number M of agents in the network, this can be obtained with an iterative process of neighbors communicating the maximum of their received Lipschitz constants (this process requires at most $M - 1$ iterations).

Now in order to derive the lower bound for the minimum of equation (4.16), we only need to deal with the terms $\mu_{\bar{f}}$ and $L_{\bar{f}}$ that appear in the second term of the minimum. To start, we present the following lemma.

Lemma 4.5. *Suppose that f is a differentiable and strongly convex function with modulus μ_f. Furthermore, let ∇f be Lipschitz continuous with finite constant L_f. Then it holds that $\mu_f \leq L_f$.*

Proof. To establish the property $\mu_f \leq L_f$, we need the fundamental property $x^T y = \|x\|_2\|y\|_2 \cos(\theta) \leq \|x\|_2\|y\|_2$ for any $x, y \in \mathbb{R}^n$. We combine this with the properties of strong convexity and Lipschitz continuity, equations (4.12) and (4.8) respectively, to derive that:

$$\begin{aligned} \mu_f\|x_1 - x_2\|_2^2 &\leq \big(\nabla f(x_1) - \nabla f(x_2)\big)^T(x_1 - x_2) \\ &\leq \|\nabla f(x_1) - \nabla f(x_2)\|_2\|x_1 - x_2\|_2 \\ &\leq L_f\|x_1 - x_2\|_2^2, \end{aligned}$$

for any x_1, x_2 in the domain of f. □

The result of the lemma motivates us to replace the term $\mu_{\bar{f}}$ by its upper bound $L_{\bar{f}}$. Following, we show that both the term $L_{\bar{f}}$ and $\mu_{\bar{f}}$ are upper bounded by L_g:

$$\mu_{\bar{f}} \leq L_{\bar{f}} = \frac{1}{M}\sum_{i=1}^{M} L_{f_i} \leq \frac{1}{M}\sum_{i=1}^{M} L_g = L_g.$$

Hence, when we design W such that it has a positive spectrum, we may conclude that the following choice for α is a feasible one when f is strongly convex with modulus μ_f:

$$\alpha = \frac{1}{2L_g} \leq \min\left\{ \frac{1}{L_g} \, , \, \frac{1}{\mu_{\bar{f}} + L_{\bar{f}}} \right\} \leq \min\left\{ \frac{1 + \lambda_M(W)}{L_g} \, , \, \frac{1}{\mu_{\bar{f}} + L_{\bar{f}}} \right\}. \tag{4.17}$$

Example 4.2 (Metropolis-Hasting matrix)
We observed in Section 3.1 that the Metropolis-Hasting matrix provides a convenient way to construct a doubly stochastic matrix W in a truly distributed environment. If we were able to show that $\lambda_M(W) \geq -\frac{1}{2}$ for any such matrix, then the choice for α, equation (4.17), remains feasible. For the purpose of this thesis, we have considered many randomly produced connected graphs with $M = 100$ vertices, and observed that this seemed to be the case for all the considered graphs. Unfortunately, this is not necessarily the case for more structured graphs; first of all, a straightforward application of the Gershgorin circle theorem tells us that a lower bound for $\lambda_M(W)$, for any graph, is given by:

$$\lambda_M(W) \geq -\frac{d_{\max} - 1}{d_{\max} + 1}.$$

In this example we provide a graph for which the lower bound is tight. Consider a *complete bipartite graph*, i.e. a graph consisting of two sets of vertices with all possible edges between these two sets. Take both sets of equal size d, such that $M = 2d$ and $d_i = d_{\max} = d_{\min} = d$ for all i. We number the vertices in the first set by $1, \ldots, d$ and the vertices in the second by $d+1, \ldots, 2d$. Then the corresponding Metropolis-Hasting matrix (3.13) is given by:

$$W = \frac{1}{d+1} \begin{bmatrix} I_d & J_d \\ J_d & I_d \end{bmatrix},$$

with $I_d, J_d \in \mathbb{R}^{d \times d}$; I_d is the identity matrix and $J_d := \mathbb{1}_d \mathbb{1}_d^T$ is the matrix of ones. We claim that $\lambda_M = -\frac{d-1}{d+1}$ is an eigenvalue; first we consider:

$$W - \lambda_M I = \frac{1}{d+1} \begin{bmatrix} I_d & J_d \\ J_d & I_d \end{bmatrix} + \frac{d-1}{d+1} \begin{bmatrix} I_d & 0_d \\ 0_d & I_d \end{bmatrix} = \frac{1}{d+1} \begin{bmatrix} dI_d & J_d \\ J_d & dI_d \end{bmatrix},$$

and then we conclude that an eigenvector is clearly given by:

$$v_M = \begin{bmatrix} \mathbb{1}_d \\ -\mathbb{1}_d \end{bmatrix}.$$

This shows that the smallest eigenvalue λ_M may be arbitrary close to -1. We mentioned that this potential problem can be avoided by constructing a shifted Metropolis-Hasting matrix $\tilde{W} = \frac{1}{2}(W + I)$, which has a positive spectrum. For our choice of α, equation (4.17), this is too drastic since we do not require a positive spectrum; we only require that $\lambda_M(W) \geq -\frac{1}{2}$. Consequently, we are reducing the mixing properties of W more than is needed. For example, in this case a shifted Metropolis-Hasting matrix $\tilde{W} = \frac{1}{4}(3W + I)$ would satisfy, as we can see that this shift gives us $\lambda_M(\tilde{W}) \geq -\frac{1}{2}$; notice that a shifted matrix $\tilde{W}$ is essentially a convex combination of W and I. In a more general context, assume that our (feasible) choice for α, as a function of γ, is:

$$\alpha(\gamma) = \frac{1}{\gamma L_g}, \tag{4.18}$$

for some $\gamma \geq 2$. Then we require (c.f. theorem 4.4) that $\lambda_M \geq -\frac{\gamma-1}{\gamma}$, since then:

$$\frac{1 + \lambda_M}{L_g} \geq \frac{1 - \left(\frac{\gamma-1}{\gamma}\right)}{L_g} = \frac{1}{\gamma L_g}.$$

This requirement is always assured by shifting the Metropolis-Hasting matrix by a convex combination of W and I, given by $\frac{1}{2\gamma}\big((2\gamma-1)W+I\big)$. We end this example with an explicit expression for the entries $w_{ij}(\gamma)$ of a shifted Metropolis-Hasting matrix $W(\gamma)$, which is suitable for any choice of $\alpha(\gamma)$ with $\gamma \geq 2$ (c.f. equation 3.13):

$$w_{ij}(\gamma) = \begin{cases} \frac{2\gamma-1}{2\gamma}\left(\frac{1}{\max\{d_i,d_j\}+1}\right) & \text{if } j \in \mathcal{N}_i, \\ 1 - \frac{2\gamma-1}{2\gamma}\sum_{k\in\mathcal{N}_i}\left(\frac{1}{\max\{d_i,d_k\}+1}\right) & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{4.19}$$

■

4.2 Distributed linear systems

In this section we consider distributed algorithms that are directly applicable to either distributed least squares problems or distributed consistent linear systems. In Sections 4.2.1 and 4.2.2 we consider two methods that are relevant for the purpose of this thesis. Then in Sections 4.2.3, we mention, although briefly, the existence of two other related methods. However, these methods have no further relevance for the purpose of this thesis.

4.2.1 Diffusion least-mean squares

We review the *diffusion least-mean squares (LMS)* method, described by C. G. Lopes and A. H. Sayed in [22]. This method is closely related to the *decentralized gradient descent* method that we described in Section 4.1.2. A more general approach for these type of methods is described in a later paper by F. S. Cattivelli and A. H. Sayed in [23], for which the diffusion LMS method turns out to be a special case. The context of this method is fundamentally different, as it considers distributed stochastic processes rather than distributed linear systems. Nonetheless it is interesting to consider this method. Namely, a necessary condition for convergence of the described method is its stability in the mean, i.e. its *unbiasedness*. In this case we are essentially applying the method to a linear system $Ax = b$ with $b \in R(A)$. In this case the decentralized gradient descent is very similar and we may expect good results when we apply it to consistent linear systems.

The method under consideration is designed for the problem of *distributed estimation*. It is assumed that each agents does a scalar measurement $b^i_{(t)} \in \mathbb{R}$ of some independent random process $\mathbf{b^i}$, at each iteration. To this scalar measurement, a realization $a^i_{(t)} \in \mathbb{R}^{1\times n}$ of another independent random process $\mathbf{a^i}$ corresponds, which are related by a linear model:

$$b^i_{(t)} = a^i_{(t)}x^* - \eta^i_{(t)}, \tag{4.20}$$

which is a traditional model for measurements in distributed estimation [10]. Here $\eta^i_{(t)}$ is a zero-mean random variable for modeling the measurement noise. Notice that there exists a common solution x^* for the linear model, which is the

minimizer of [23, Equation (1)]:

$$\sum_{i=1}^{M} \mathbb{E} \left\| b_{(t)}^i - a_{(t)}^i x^* \right\|_2,$$

which is also the solution of interest for each agent. Here $\mathbb{E}$ denotes the expectation operator. A formal definition of the optimal solution x^* is given by [23, Equation (2)]:

$$x^* = \left(\sum_{i=1}^{M} \mathbb{E} \left(a_{(t)}^i \right)^T a_{(t)}^i \right)^{-1} \left(\sum_{i=1}^{M} \mathbb{E} \left(a_{(t)}^i \right)^T b_{(t)}^i \right).$$

As mentioned before, a necessary condition for the diffusion LMS method is its stability in the mean. For clarity we will follow the notation of Section 4.1.2. Furthermore, in this context a^i and b^i correspond to the mean of respectively the random processes $\mathbf{a^i}$ and $\mathbf{b^i}$; clearly they satisfy $b^i = a^i x^*$ for all i, cf. equation (4.20). Then, the mean stability of the method is analyzed by considering its expectation $\mathbb{E}$, in which case the method simplifies to (cf. equation (4.7)):

$$\begin{aligned} y_{(t)}^i &= \sum_{j \in \mathcal{N}_i} w_{ij} x_{(t)}^j \\ x_{(t+1)}^i &= y_{(t)}^i - \alpha_k \nabla f_{i,(t)}, \end{aligned} \tag{4.21}$$

with $f(x) = \frac{1}{2} \| b_i - a_i x \|_2^2$ and $\nabla f_{i,(t)} := \nabla f_i(y_{(t)}^i)$. Notice that we consider ∇f_i in the intermediate step $y_{(t)}^i$, rather than in the previous iterate $x_{(t)}^i$ as we would do in the decentralized gradient descent method (4.7). This actually also makes more sense from an intuitive perspective. It is allowed that $\alpha_i \neq \alpha_j$, but we will simplify this for the purpose of this thesis by assuming a common choice α. This particular method indeed corresponds to the diffusion LMS algorithm of [22], however, it is considered to be a special case of the method proposed in [23]. Here it is referred to by the Combine-then-Adapt (CTA) diffusion LMS algorithm.

In the mean, the traditional model (4.20) satisfies $a^i x^* = b^i$ such that x^* is a solution to $Ax = b$. We assume that x^* is unique, which means that $\text{rank}(A) = n$ and $b \in R(A)$. Then, we may uniquely define the error quantities $e_{(t)}^i = x^* - x_{(t)}^i$ and $\tilde{e}_{(t)}^i = x^* - y_{(t)}^i$. Furthermore, we define the block diagonal matrix $\mathcal{A} = \text{diag}\{a_1, \ldots, a_M\}$ and observe that $g_{(t)} = -\mathcal{A}^T \mathcal{A} \tilde{e}_{(t)}$; Recall that we defined $g_{(t)} = [\nabla f_{i,(t)}] = [A_i^T A_i (y_{(t)}^i - x^*)]$. By substitution in (4.21), it can be obtained that a fundamental iteration for the error quantity $e_{(t)}$ is given by:

$$e_{(t+1)} = (\bar{I} - \alpha \mathcal{A}^T \mathcal{A}) \bar{W} e_{(t)}, \tag{4.22}$$

where $\bar{I} = I \otimes I$ and $\bar{W} = W \otimes I$. We mention that connectedness of the underlying graph $\mathcal{G}$ is implicitly assumed, since in this case $e_{(t)}$ is the only error quantity that satisfies $\bar{W} e_{(t)} = e_{(t)}$. In contrast, if $\mathcal{G}$ consisted of connected components, then we could define other error quantities by using different solutions $x_1^*, \ldots, x_k^*$ for each of the k connected components and obtain a similar fundamental iteration as above. The mean stability depends on the spectral properties of the matrix $X\bar{W}$, where we defined $X = (\bar{I} - \alpha \mathcal{A}^T \mathcal{A})$. This in turn depends on a result of

[23, Lemma 1], however, we mention that the proof of this result is not entirely correct. The lemma assumes that W is right-stochastic, however, the result of this lemma can only be assured when W is doubly stochastic. Hence, we will assume that the underlying graph in undirected, such that W can also be chosen to be doubly stochastic. Then the mean stability follows by [23, Theorem 1].

Theorem 4.6 (unbiasedness of the diffusion LMS method)**.** *Let expectation of the diffusion LMS method (4.21) be executed by a time-independent and undirected connected network. Assume that the solution x^* to $Ax = b$ is unique. Furthermore, let W be connected and let α satisfy:*

$$\alpha < \frac{2}{\max_i\{\lambda_1(a_i^T a_i)\}}. \tag{4.23}$$

Then we have for the fundamental iteration of the error $e_{(t)}$ (4.22):

$$\rho(X\bar{W}) < 1,$$

such that $x^i_{(t)} \to x^*$ as fast as $\rho^t \to 0$, implicating unbiasedness of the diffusion LMS method.

We mention that this part of the analysis would also work if we would assume that each agents knows multiple rows A_i, instead of just a single row a_i. Notice the similarity of the conditions for α given by equations (4.23) and (4.18). The similarity follows by the observation in example 4.1 that $L_{f_i} = \lambda_1(A_i^T A_i)$, such that the condition (4.23) is equivalent to $\alpha < \frac{2}{L_g}$, where we recall that $L_g = \max_i\{L_{f_i}\}$.

4.2.2 Distributed kernel projection

In this section we consider the *distributed kernel projection* method, described by S. Mou, J. Liu and A.S. Morse in [24]. This method is not named as such in the article itself, but we conveniently choose to do so for the purpose of this thesis. The method applies to distributed linear systems $Ax = b$ for which it is assumed that there exist at least one solution. The method also works for linear systems with infinitely many solutions, but this result is redundant for our purpose. Hence, we will only consider the case where the solution x^* is unique; this means that we assume that the system $Ax = b$ is either square and nonsingular or overdetermined and consistent. The latter is not a very useful case for practical applications, since then the system is usually overdetermined and inconsistent. The paper proposes a solution for this problem, [24, Section IX (p. 2876)], but it is mentioned that this approach requires an elaborate communication step in the initialization of the method. Furthermore, this approach does not scale well with the number of agents M in the network. For this reason we consider this approach to be infeasible.

Nonetheless we take the effort to consider this method, since (i) the paper captures some interesting ideas and (ii) we will observe that this method is closely related to the method that we introduce in Section 5.2; in some way, we could consider this method as an intermediate step between the distributed kernel projection method and the decentralized gradient descent method of Section 4.1.2. From this point of view, the distributed kernel projection method

may provide useful information in the numerical experiments that will follow in Section 5.3.

The distributed kernel projection method allows for a time-dependent network topology, as long as the sequence of graphs is repeatedly jointly strongly connected. This is a necessary and sufficient condition for convergence. Later in the paper, [24, Section VIII (p. 2875)], it is shown that the algorithm also works with asynchronous communication; we will, however, omit this part for simplicity.

In an earlier paper [26], Mou and Morse studied the same method under the milder conditions of a fixed network topology and synchronous communication. Also, it is assumed that each agent knows only a single row $A_i = a_i$ of A, that there are $M = n$ agents and that $A \in \mathbb{R}^{n\times n}$ is nonsingular[1]. The analysis in this paper is widely different, but the fundamental approach of the method and the convergence result are very similar. We first follow this paper, since the analysis is far less technical.

The distributed kernel projection method differs fundamentally from gradient descent like approaches that we considered before, as it maintains solutions $x^i_{(t)}$ for the local systems $A_i x = b_i$, by projection onto the kernel of A_i. Now, assume that $x^i_{(t)} \in \mathcal{X}^*_i$ is such a solution at some iteration t. Then, in the next iteration, the aim is to find a new solution in $\mathcal{X}^*_i$ that minimizes, in the least square sense, the difference from the new solution with the consensus update $\frac{1}{d_i+1}\sum_{j\in\mathcal{N}^*_i} x^j_{(t)}$. This is done by: First projecting the difference $x^i_{(t)} - \frac{1}{d_i+1}\sum_{j\in\mathcal{N}^*_i} x^j_{(t)}$ onto the kernel of A_i, and then, adding the projection to the current solution $x^i_{(t)}$, i.e. the update scheme is given as follows:

$$x^i_{(t+1)} = x^i_{(t)} - P_i\left(x^i_{(t)} - \frac{1}{d_i+1}\sum_{j\in\mathcal{N}^*_i} x^j_{(t)}\right). \tag{4.24}$$

This scheme is the same as the one proposed in [24]. Here P_i is the orthogonal projection matrix onto the kernel of A_i; if we assume that $A_i \in \mathbb{R}^{m_i}$ with $\operatorname{rank}(A_i) = m_i$, we may define the projection matrices by:

$$P_i = I - A_i^T(A_iA_i^T)^{-1}A_i. \tag{4.25}$$

The projection matrices are symmetric, idempotent, i.e. $P_i^2 = P_i$, and satisfies $\lambda_{\max}(P_i) = \|P_i\|_2 \leq 1$. At first sight, the update scheme (4.24) looks fundamentally different from the general update scheme that we introduced in Section 3.2. However, we will see that this method can be written in the general form of equation (3.16). To this end we observe that $x^i_{(t)} - P_i x^i_{(t)} = z^*_i$, where z^*_i is the unique solution to $A_i x = b_i$ in $R(A_i^T)$, such that we can rewrite (4.24) as:

$$x^i_{(t+1)} = z^*_i + P_i\left(\frac{1}{d_i+1}\sum_{j\in\mathcal{N}^*_i} x^j_{(t)}\right).$$

[1]The assumption that each agents knows only a single row a_i is only explicitly used in the proof of [26, Lemma 3]. Here, the case that each agent knows only a single row is a special case and the result is generally not true otherwise. Unfortunately, this result is crucial for the arguments in the proof of [26, Proposition 1]. Nonetheless, the correctness of the method remains valid by the results in [24], but the proof then requires a fundamentally different approach.

From this we observe that $x^i_{(t+1)}$ is the projection of $\frac{1}{d_i+1}\sum_{j\in\mathcal{N}_i^*} x^j_{(t)}$ onto the solution space of $\mathcal{X}_i^*$. That is, we can rewrite (4.24) to a general form update (3.16) as:

$$\begin{aligned} y^i_{(t+1)} &= \frac{1}{d_i+1}\sum_{j\in\mathcal{N}_i^*} x^j_{(t)} \\ x^i_{(t+1)} &= y^i_{(t),P} \qquad\qquad \text{for } i = 1,\dots,M. \end{aligned} \tag{4.26}$$

Here $y^i_{(t),P}$ is the projection of $y^i_{(t)}$ onto $\mathcal{X}_i^*$. We proceed with the form of equation (4.24) and reformulate it in a more compact way. First of all, let $x_{(t)} = [x^i_{(t)}]$ be the vector of states and define the vector of differences by:

$$\delta_{(t)} = \bar{H}^T x_{(t)} \in \mathbb{R}^{nK},$$

where $\bar{H} = H \otimes I$, with the incidence matrix H (3.5), and K the number of edges. Observe that $\delta = [\delta^k]$, with $\delta^k = x^i - x^j$ for the kth oriented edge (i,j). Now let $\bar{D} = D \otimes I$ and $P = \text{diag}\{P_1,\dots,P_m\}$, then according to [26, Equation (3)] a compact notation for (4.24) is given by[2]:

$$x_{(t+1)} = x_{(t)} - P\bar{D}\bar{H}\delta_{(t)}. \tag{4.28}$$

The term $\delta_{(t)}$ is crucial in the proof of the convergence behavior [26] in the sense that $\delta_{(t)} \to 0$ is a necessary condition for convergence. We pre-multiply with $\bar{H}^T$ on both sides to obtain:

$$\begin{aligned} \delta_{(t+1)} &= \delta_{(t)} - \bar{H}^T P\bar{D}\bar{H}\delta_{(t)} \\ &= (I - Q)\delta_{(t)}, \end{aligned} \tag{4.29}$$

where we defined $Q := \bar{H}^T P\bar{D}\bar{H}$. For this matrix Q we formulate the result of [26, Proposition 1].

Lemma 4.7. *Let $Q := \bar{H}^T P\bar{D}\bar{H}$ be the matrix corresponding to the fundamental iteration for the differences $\delta_{(t)}$, as in equation (4.29), then:*

(i) Q is symmetric positive semi-definite with $0 \le \rho(Q) < 2$;

(ii) $\delta_{(t)} \to 0$ as fast as $\big(1-\rho(Q)\big)^t \to 0$, with $|1-\rho(Q)| < 1$.

Then, the property that $\ker(H^T) = \mathbb{1}$, or equivalently that:

$$\ker(\bar{H}^T) = \{[x] \in \mathbb{R}^{nM} \mid x \in \mathbb{R}^n\}, \tag{4.30}$$

is used to show that $\delta_{(t)}$ linearly converges to a vector of common states that correspond to the unique solution $x^* = A^{-1}b$, [26, Lemma 1].

[2] In this paper, it is mentioned that agent i is always taken to be a neighbor of itself, but the paper is unclear in its notation regarding this statement. We mention that this statement contradicts the compact form notation (4.28). This follows, as the compact form notation corresponds to (cf. equation (4.24)):

$$x^i_{(t+1)} = x^i_{(t)} - P_i\left(x^i_{(t)} - \frac{1}{d_i}\sum_{j\in\mathcal{N}_i} x^j_{(t)}\right). \tag{4.27}$$

The convergence results remain valid, but the update scheme to which it corresponds is slightly different.

Lemma 4.8. *If $\delta_{(t)} \to 0$ as fast as $\rho \to 0$, for some $\rho < 1$, then $x^i_{(t)} \to A^{-1}b$, for all i, as fast as $\rho \to 0$.*

The results of the previous two lemmas give the correctness of the kernel projection method, assuming that we have a fixed communication graph and that each agent knows a single row.

We now briefly consider the more general case of [24], where agents know an arbitrary amount of rows and where the network topology is time-dependent. The solution is assumed to be unique; an algebraic condition for this assumption is given by [24, Equation (9)]:

$$\bigcap_{i=1}^{M} \mathcal{P}_i = 0, \tag{4.31}$$

where $\mathcal{P}_i = R(P_i)$. This means that $Ax = b$ is allowed to be overdetermined, as long as the system is consistent. We do not, however, allow redundant agents, i.e. agents that are not necessarily needed to solve the equation $Ax = b$. We give a formal definition of redundancy.

Definition 4.9 (redundancy of agents)**.** *A set of agents with labels in $\mathcal{V} \subseteq \{1, \dots, M\}$ are called redundant, if a solution to $A_i x = b_i$ for all $i \in \mathcal{V}^C$ is also a solution to $Ax = b$.*

Recall that $\mathcal{V}^C$ is the complement of $\mathcal{V} \subseteq \{1, \dots, M\}$. We shall assume that no redundant set $\mathcal{V}$ exists. Now, for a time-dependent network topology, the update scheme (4.24) looks like:

$$x^i_{(t+1)} = x^i_{(t)} - P_i \left(x^i_{(t)} - \frac{1}{d_{i,(t)} + 1} \sum_{j \in \mathcal{N}^*_{i,(t)}} x^j_{(t)} \right). \tag{4.32}$$

We define the error quantity by $e_{(t)} = [e^i_{(t)}]$ with $e^i_{(t)} = x^* - x^i_{(t)}$. Since both $x^*, x^i_{(t)} \in \mathcal{X}^*_i$, we have that $e^i_{(t)} \in \ker(A_i)$ such that $P_i e^i_{(t)} = e^i_{(t)}$. From this it is observed that [24, Equation (7)]:

$$e^i_{(t+1)} = \frac{1}{d_{i,(t)} + 1} P_i \sum_{j \in \mathcal{N}_i} P_j e^j_{(t)}.$$

The kernel projection method does an averaging according to the uniform weights (3.10). The corresponding right-stochastic matrix is given by $W_{(t)} = (D_{(t)} + I)^{-1}(N_{(t)} + I)$, where $N_{(t)}$ is the time-dependent adjacency matrix (3.3). By definition of $W_{(t)}$ we are able to give a compact notation of the fundamental update for the error quantity[3]. It follows that [24, Equation (8)]:

$$e_{(t+1)} = P(W_{(t)} \otimes I) P e_{(t)}.$$

Or equivalently:

$$e_{(t+1)} = P(W_{(t)} \otimes I) P(W_{(t-1)} \otimes I) \cdots P(W_{(0)} \otimes I) P e_{(0)}, \tag{4.33}$$

[3]If each agent is not taken to be a neighbor of itself, cf. equation (4.27), then the stochastic matrix is given by $W_{(t)} = D^{-1}_{(t)} N_{(t)}$. In both cases the matrix $W_{(t)}$ is right-stochastic, such that the following convergence results apply to both cases.

where we used $P^2 = P$. Now, the linear convergence of the kernel projecting method follows by the linear convergence of the matrix product $P(W_{(t)} \otimes I)P(W_{(t-1)} \otimes I) \cdots P(W_{(0)} \otimes I)P$ to zero as t tends to infinity, with respect to the so called *mixed matrix norm* $\|\cdot\|_M$ for matrices $Q \in \mathbb{R}^{nM \times nM}$. This norm is shown to be well-defined and sub-multiplicative [24, Equation (10)]. The convergence towards zero of this matrix product is considered to be one of the main technical results of this paper [24, Theorem 3].

Theorem 4.10 (Kernel projection method)**.** *Let the kernel projection method (4.32) be executed by a time-dependent and possible directed network. Also, let the sequence of graphs $\mathcal{G}_{(t)}$ be repeatedly jointly strongly connected with a bounded intercommunication interval $B \geq 1$. Suppose that each agent knows some rows $[A_i \ b_i]$ of the partitioned matrix $[A \ b]$, that the solution x^* to $Ax = b$ is unique and that no redundant set $\mathcal{V} \subseteq \{1, \ldots, M\}$ exists. Then there exists some positive $\rho < 1$ for which we have, at any iteration t, that:*

$$\|P(W_{(t)} \otimes I)P(W_{(t-1)} \otimes I) \cdots P(W_{(0)} \otimes I)P\|_M \leq \rho^{(t-Bq)},$$

where $q := (M-1)^2$. Equivalently, for any agent i we have that:

$$x^i_{(t)} \to x^* \quad \text{as fast as} \quad \rho^t \to 0.$$

We mention that an expression for ρ exists [24, Theorem 3], however, the expression only suits theoretical purposes in the sense that it is very difficult to compute the actual value of ρ for explicit examples; it is only shown by theoretical derivation that we indeed have that $\rho < 1$.

Remark: The fact that we require that each agent must be able to project vectors onto the kernel of A_i, poses some issues for the practical implementation of this method. Indeed, we could perform the projection of some vector y onto $\ker(A_i)$ explicitly, by computing the matrix-vector product with P_i given by equation (4.25). However, this poses some issues on itself.

In the first place, this approach requires that we need to consider matrix-vector products with the matrix $(A_i A_i^T)^{-1}$, cf. equation (4.25). This will inevitably be the main cause for numerical errors, especially if A_i is close to being rank deficient. Due to this numerical errors, we can not guarantee that the projection is exactly (at least up to the machine precision) in the kernel of A_i. Consequently, by adding this approximated projection to the current iterate $x^i_{(t)}$ of agent i, we can not guarantee that we maintain an exact solution (up to machine precision) to $A_i x = b_i$.

This issue may potentially have a devastating effect on the convergence behavior of the kernel projection method. Hence, we propose an alternative solution. The general form formulation of the kernel projection method, equation (4.26), revealed that we may alternatively obtain the new iterate by a projection of $y^i_{(t+1)} = \frac{1}{d_i+1} \sum_{j \in \mathcal{N}_i^*} x^j_{(t)}$ onto the solution space $\mathcal{X}_i^*$. Now, from Section 5.1, we conclude that if we initiate the steepest descent method for underdetermined systems with $y^i_{(t+1)}$, for the particular system $A_i x = b_i$, then we converge precisely to its projection onto $\mathcal{X}_i^*$, cf. theorem 5.6. Or alternatively, with the conjugate gradient method for underdetermined systems, Section 5.1.2, we converge to this projection within m_i iterations (in exact arithmetic), cf. theorem 5.12. Either of these two approaches have two fundamental advantages.

There is no need to consider the term $(A_i A_i^T)^{-1}$, such that significant numerical errors are avoided.

Even though numerical errors also occur for either the steepest descent- or conjugate gradient method, they can be executed until convergence, i.e. until we obtain an exact solution up to machine precision.

We finally mention that the first step of the steepest descent- and conjugate gradient method are identical. Hence, if each agent knows only one row, then both approaches are identical since the conjugate gradient method converges in $m = 1$ step in this case. In fact, in this case we can even see that the explicit projection with the matrix P_i is an identical approach as well. This is the only case for which the kernel projection method and the distributed steepest descent method, which we introduce in Section 5.2, are identical.

4.2.3 Other methods

We briefly consider two fundamentally different approach for the sake of completeness.

Consensus-based distributed least-mean squares

A variant on the diffusion LMS method, Section 4.2.1, is described by I. D. Schizas, G. Mateos and G. B. Giannakis in [27, 28]. This method is a consensus-based diffusion LMS method, where additional variables are introduced to provoke consensus throughout the algorithm. These variables can be interpreted as Lagrange multipliers. Although this method seems to outperform many other methods, e.g. see the comparative survey on truly distributed solvers [29], we consider this method to be beyond the scope of this thesis.

Push-sum distributed least squares solver

Another method by K. E. Prikopa, H. Straková and W. N. Gansterer [30, 29] is the push-sum distributed least squares solver. This method aims to approximately compute the QR factorization for A in a truly distributed manner, followed by a distributed step of solving the system $QRx = b$, or equivalently $Rx = Q^T b$. This method appears to perform well in the same survey on truly distributed solvers [29]. However, the method requires a great amount of data exchange, which is something that we aim to avoid in this thesis.

5 Distributed steepest descent

In the previous chapter we have considered several distributed methods that aim to distributively solve a system $Ax = b$, either consistent or inconsistent. In this chapter we introduce yet another method. Inspired by the decentralized gradient descent (`DGD`) method, Section 4.1.2, and the kernel projection (`KP`) method, Section 4.2.2, we propose a method that uses properties of both the `DGD` and `KP` method. This method, the distributed steepest descent (`DSD`) method, does a two-step update of the form (3.15). The first step corresponds to a single step of a local steepest descent update, see also Section 2.3.2, and the second step is simply a communication update.

The outline of this chapter will be as follows. In Section 5.1 we show that the steepest descent method of Section 2.3.2 even converges for systems $A^TAx = A^Tb$, where $A \in \mathbb{R}^{m\times n}$ is an underdetermined and full-rank matrix. This somewhat justifies our choice for the `DSD` method. Here we observe that the method converges to the projection of the initial guess $x_{(0)}$, onto the solution space $\mathcal{X}^*$, hence justifying the relation of the `DSD` method with the `KP` method. Additionally we show that the full projection of the initial guess $x_{(0)}$ may be obtained more efficiently by the conjugate gradient method for underdetermined system, Section 5.1.2. Next we introduce two variants of the `DSD` method in Section 5.2. We will briefly consider the convergence behavior in Section 5.2.1 and consider the existence of equilibrium points in Section 5.2.2. We finally end this section with a comparative study on the three methods `DGD`, `KP` and `DSD` in Section 5.3.

5.1 Descent methods for local minimization

We introduce two descent methods that can be used for local minimization. The first is a variant of the steepest descent method of Section 2.3.2 and the second is a variant of the conjugate gradient method of Section 2.3.3. For both methods we establish a proof of linear convergence.

5.1.1 Steepest descent for underdetermined systems

In this section we will show how the steepest descent (`SD`) method of Section 2.3.2 can be applied to underdetermined systems $Ax = b$. Recall that the `SD` method is specifically designed for to SPD matrices $Q > 0$, such that we

observe from Section 2.2.1 that this method can also be used to solve the least squares systems $A^TAx = A^Tb$, for some full rank overdetermined system $Ax = b$. However, the correctness of the SD method applied to such systems with underdetermined matrices is less straightforward justified. In this section we assume that $A \in \mathbb{R}^{m\times n}$ with $m < n$. Furthermore we assume that the matrix is full rank, i.e. we have $\text{rank}(A) = m$. Under these assumptions will we show the correctness of the SD scheme when applied to positive semi-definite system of the form:

$$A^TAx = A^Tb, \tag{5.1}$$

where we observe that $A^TA \geq 0$ with $\text{rank}(A^TA) = m$. Upon proving the convergence of SD, applied to the above system, we have to deal with two issues.

1. Convergence of the steepest descent method has been proven for SPD matrices $Q > 0$, where we conveniently used the notion of the well-defined Q-norm $\|\cdot\|_Q$. For the system of equation (5.1), the norm $\|\cdot\|_{A^TA}$ is not well-defined and we need to avoid the use of it.

2. Since we assume that A is a full rank underdetermined matrix, we have $R(A) = \mathbb{R}^m$ and $\dim(\ker(A)) \geq 1$, and we observe that the system $Ax = b$ has infinitely many solutions. Notice, by $\ker(A^T) = \{0\}$, that the expanded system $A^TAx = A^Tb$ has the same solution set. Now since the original proof of the steepest descent method uses the existence of a unique solution, we will also need to show to which specific solution the method converges to. As it turns out, this depends on the initial guess $x_{(0)}$.

We denote by p the residual corresponding to the system $A^TAx = A^Tb$; given any estimate $x \in \mathbb{R}^n$ of x^*, we have $p = A^Tb - A^TAx$; it is related to the actual residual r by $p = A^Tr$. We mention that the actual residual is not of our interest in the remaining part of this section and give the update scheme of the steepest descent method, applied to (5.1).

$$\begin{aligned} p_{(t)} &= A^Tb - A^TAx_{(t)}, \\ \alpha_{(t)} &= \frac{p_{(t)}^Tp_{(t)}}{p_{(t)}^TA^TAp_{(t)}} = \frac{\|p_{(t)}\|_2^2}{\|Ap_{(t)}\|_2^2}, \\ x_{(t+1)} &= x_{(t)} + \alpha_{(t)}p_{(t)}. \end{aligned} \tag{5.2}$$

In this formulation we avoided the use of the ill-defined A^TA-norm by observing that it holds that $\|x\|_{A^TA} = \|Ax\|_2$ such that we can use the properties of the well-defined 2-norm. For completeness we mention that residual $p_{(t+1)}$ is updated by:

$$p_{(t+1)} = p_{(t)} - \alpha_{(t)}A^TAp_{(t)}. \tag{5.3}$$

We will often refer to this by method by SDud for short, emphasizing that it is the Steepest Descent method applied to underdetermined systems. Now, with regard to the update scheme (5.2) we address the following issue. Since the kernel of A is nontrivial for underdetermined systems, it might happen that $Ap_{(t)} = 0$ while $p_{(t)} \neq 0$. In this case the method would stagnate. We will show by the result of the following lemma that this never occurs

Lemma 5.1. *Let the SDud method of equation (5.2) be initiated by some arbitrary $x_{(0)} \in \mathbb{R}^n$ and let $p_{(t)}$ be the residual corresponding to the extended system*

$A^TAx = A^Tb$ *at any iteration* t*, then we have that* $Ap_{(t)} = 0$ *if and only if* $p_{(t)} = 0$*.*

Proof. Consider the case $Ap_{(t)} = 0$ and recall that $p_{(t)} = A^Tr_{(t)}$, where $r_{(t)}$ is the residual corresponding to the system $Ax = b$. We observe that $Ap_{(t)} = AA^Tr_{(t)} = 0$. Since AA^T is nonsingular, this equality holds if and only if $r_{(t)} = 0$, which in turn holds if and only if $p_{(t)} = 0$. The latter is concluded from the fact that $\ker(A^T) = \{0\}$. □

Now that we know that the `SD` method (5.2) never stagnates, we will determine to which specific solution it converges. To this end we are going to define the solution space $\mathcal{X}^*$, corresponding to the system $Ax = b$, in a particular way. We first need to show that there exist a unique particular solution $z^* \in R(A^T)$; by using the fact that the matrix AA^T is nonsingular we may find z^* in two steps: Solve the system $AA^Ty = b$ uniquely for y and then compute $z^* := A^Ty$. It follows from $\ker(A^T) = \{0\}$ that this solution is unique in $R(A^T)$. Now we define the solution space in the particular way as follows:

$$\mathcal{X}^* = \{z^* + u \mid u \in \ker(A)\}. \tag{5.4}$$

This definition is conveniently defined such that we can express to which specific solution `SDud` converges to, based on the initial guess $x_{(0)}$. Now let $x_{(0)}$ be an arbitrary initial guess, then we know by the *rank-nullity* theorem, combined with the fundamental property $R(A^T) \perp \ker(A)$, that we can write $x_{(0)}$ uniquely as:

$$x_{(0)} = z_{(0)} + u_{(0)}, \quad \text{where } z_{(0)} \in R(A^T) \text{ and } u_{(0)} \in \ker(A). \tag{5.5}$$

We can now state and prove the following lemma.

Lemma 5.2. *Let* $x_{(0)}$*, written in the form (5.5), be the initial guess for the* `SDud` *method of equation (5.2). Then for any iteration* t *we have that:*

$$x_{(t)} = z_{(t)} + u_{(0)}, \tag{5.6}$$

for some $z_{(t)} \in \ker(A^T)$*.*

Proof. The proof is by induction. The case $t = 0$ is trivial; now assume that the equality is true for any t and consider the case $t + 1$:

$$\begin{aligned} x_{(t+1)} &= x_{(t)} + \alpha_{(t)}p_{(t)} \\ &= x_{(t)} + \alpha_{(t)}A^Tr_{(t)} \\ &= \left(z_{(t)} + \alpha_{(t)}A^Tr_{(t)}\right) + u_{(0)}, \end{aligned}$$

where we conclude that $\left(z_{(t)} + \alpha_{(t)}A^Tr_{(t)}\right) \in R(A^T)$, since we have $z(t) \in R(A^T)$ by the induction hypothesis. □

This lemma shows that the component $u_{(0)} \in \ker(A)$ in the initial guess of $x_{(0)}$ is preserved throughout the algorithm. Hence, if the algorithm converges to a solution $x \in \mathcal{X}^*$, then this can only be to the specific solution $x^* = z^* + u_{(0)} \in \mathcal{X}$. Observe that that $x^* = x_{(0),P} := x^*_P$, i.e. the projection of $x_{(0)}$ onto $\mathcal{X}^*$. Now, the derivation in the proof of lemma 5.2 reveals that the `SD` method implicitly does an update of the form $z_{(t+1)} = z_{(t)} + \alpha_{(t)}p_{(t)}$ and we need to show that

$z_{(t)} \to z^*$ as $t \to \infty$. With this insight we can follow the proof as it is formulated in [14, Chapter 8]. We start by defining the error $e_{(t)}$ at any iteration t as the deviation of $x_{(t)}$ from the projected solution $x_P^* = z^* + u_{(0)} \in \mathcal{X}^*$. We observe that it satisfies:

$$\begin{aligned} e_{(t)} &= \left(z^* + u_{(0)}\right) - x_{(t)} \\ &= \left(z^* + u_{(0)}\right) - z_{(t)} + u_{(0)} \\ &= z^* - z_{(t)}. \end{aligned} \tag{5.7}$$

We observe that $e_{(t)} \in R(A^T)$ for any t; this is a key observation that we need later on. Now since the matrix $A^T A$ does not have a complete set of nonzero eigenvalues and since its inverse is not well-defined, we need to slightly refine the arguments of the original proof in [14, Chapter 8]. We proceed by defining its set of eigenvalues. Since $A^T A \geq 0$ and $\operatorname{rank}(A^T A) = m$, we know that $A^T A$ has m nonzero, positive eigenvalues $\lambda_1, \ldots, \lambda_m$. We assume that these eigenvalues are distinct and furthermore we assume without loss of generality that they are ordered such that:

$$\lambda_1 > \lambda_2 > \ldots > \lambda_m > 0.$$

We let $v_1, \ldots, v_m$ be the corresponding set of eigenvalues and assume, again without loss of generality, that they are normalized, i.e. that $v_i^T v_i = 1$ for all i. Since $A^T A$ is symmetric we also conclude that $v_j^T v_i = 0$ for any $i \neq j$. Furthermore we conclude from $\operatorname{rank}(A^T A) = m$ and $R(A) = \mathbb{R}^m$ that:

$$R(A^T) = R(A^T A) = \operatorname{span}\{v_1, \ldots, v_m\}, \tag{5.8}$$

i.e. with the above equation we have found an explicit expression for the range of A^T. Now recall that we observed from equation (5.7) that $e_{(t)} \in R(A^T)$, such that for any iteration t we can write the following:

$$e_{(t)} = \sum_{i=1}^{m} c_i v_i, \tag{5.9}$$

for some appropriate choice of constants $c_i \in \mathbb{R}$. Notice that the constants c_i actually require an argument t, however, since we are going to consider each iteration individually we omit the argument for simplicity. We now formulate and prove some useful properties for any vector $v \in R(A^T)$ that is written in the form of (5.9).

Lemma 5.3. *Let $v \in R(A^T)$ be uniquely written as $v = \sum_{i=1}^m c_i v_i$ for some appropriate constants $c_i \in \mathbb{R}$, then we have the following identities:*

$$\begin{aligned} w &:= A^T A v &&= \textstyle\sum_{i=1}^m c_i \lambda_i v_i; \\ \|v\|_2^2 &= v^T v &&= \textstyle\sum_{i=1}^m c_i^2; \\ \|Av\|_2^2 &= v^T A^T A v &&= \textstyle\sum_{i=1}^m c_i^2 \lambda_i; \\ \|w\|_2^2 &= w^T w &&= \textstyle\sum_{i=1}^m c_i^2 \lambda_i^2; \\ \|Aw\|_2^2 &= w^T A^T A w &&= \textstyle\sum_{i=1}^m c_i^2 \lambda_i^3. \end{aligned}$$

Proof. The identities follow by direct computations. The first identity is derived by substitution of $v = \sum_{i=1}^m c_i v_i$ and then applying the identity $A^T A v_i = \lambda_i v_i$. The second identity is also obtained by substitution of v and by subsequently observing that cross terms cancel due to orthogonality of the set $v_1, \ldots, v_m$. The remaining identities are derived in a similar manner. □

Before we proceed with the next step of the proof of convergence, similar to the next step in [14, Chapter 8], we make an important observation. The convergence of the original SD method is based on the convergence of the SPD Q-norm of the error towards zero, i.e. $\|e_{(t)}\|_Q \to 0$; this is also formulated in theorem 2.8 in Section 2.3.2. In our case, considering the previous work in this section, a natural approach is to show convergence of the 2-norm of the residual term $Ae_{(t)} = r_{(t)}$ towards zero, i.e. $\|Ae_{(t)}\|_2 = \|r_{(t)}\|_2 \to 0$. In this way we avoid the use of the ambiguous A^TA-norm. We can see that this is a sufficient approach, since $Ae_{(t)} = r_{(t)} = 0$ if and only if $e_{(t)} = 0$. This is true because $e_{(t)} \in R(A^T)$ and $R(A^T) \perp \ker(A)$. We proceed with the next step and state and prove the following lemma.

Lemma 5.4. *The update scheme of* `SDud`*, equation 5.2, satisfies:*

$$\|Ae_{(t+1)}\|_2^2 = \left(1 - \frac{(p_{(t)}^T p_{(t)})^2}{\big(p_{(t)}^T A^T A p_{(t)}\big)\big(e_{(t)} A^T A e_{(t)}\big)}\right) \|Ae_{(t)}\|_2^2. \tag{5.10}$$

Proof. The proof is direct; we observe that the error $e_{(t)}$ is updated according to $e_{(t+1)} = e_{(t)} - \alpha_{(t)} p_{(t)}$. For the sake of clarity, we omit the dependency on t f and temporarily define $\tilde{e} := e_{(t+1)}$, satisfying $\tilde{e} = e - \alpha p$. Then we we find that:

$$\begin{aligned} \|A\tilde{e}\|_2^2 &= \tilde{e}^T A^T A \tilde{e} \\ &= \big(e^T - \alpha p^T\big) A^T A \big(e - \alpha p\big) \\ &= e^T A^T A e - 2\alpha p^T A^T A e + \alpha^2 p^T A^T A p. \end{aligned}$$

The last equality is obtained by symmetry of A^TA. By substitution of the definition of $\alpha_{(t)}$, given by equation (5.2), it is straightforwardly computed that:

$$-2\alpha p^T A^T A e + \alpha^2 p^T A^T A p = -\frac{(p^T p)^2}{p^T A^T A p}. \tag{5.11}$$

The expression is well defined by lemma 5.1, assuming that $p \neq 0$. We continue with the previous derivation and find that:

$$\begin{aligned} \|A\tilde{e}\|_2^2 &= e^T A^T A e - \frac{(p^T p)^2}{p^T A^T A p} \\ &= \left(1 - \frac{(p^T p)^2}{\big(p^T A^T A p\big)\big(eA^T A e\big)}\right) \|Ae\|_2^2. \end{aligned}$$

□

The next step of the original SD convergence proof is based on the *Kantorovich inequality* for SPD matrices [14, (p. 237)]. The inequality is, among other things, obtained by using the fact that Q is nonsingular. Nonetheless we obtain a very similar inequality for underdetermined systems, where A^TA is singular.

Lemma 5.5 (Kantorovich inequality for underdetermined systems)**.** *For any $A \in \mathbb{R}^{m\times n}$ with $m \leq n$ and rank$(A) = m$, we let $v \in R(A^T)$ be arbitrary and define $w := A^TAv$. Any such vector satisfies the following inequality:*

$$\frac{(w^T w)^2}{\big(w^T A^T A w\big)\big(v^T A^T A v\big)} \geq \frac{4\lambda_1\lambda_m}{(\lambda_1 + \lambda_m)^2}. \tag{5.12}$$

Proof. Since $v \in R(A^T)$ and $A \in \mathbb{R}^{m\times n}$ with $m < n$ and $\text{rank}(A) = m$, we know that we can write v uniquely as $v = \sum_{i=1}^{m} c_i v_i$, where $c_i \in \mathbb{R}$ are some appropriate constants and $v_1, \ldots, v_m$ are the m normalized eigenvectors of A. Then, by the identities of lemma 5.3, we obtain that:

$$\begin{aligned}
\frac{(w^Tw)^2}{\left(w^TA^TAw\right)\left(v^TA^TAv\right)} &= \frac{\left(\sum_{i=1}^m c_i^2\lambda_i^2\right)^2}{\left(\sum_{i=1}^m c_i^2\lambda_i^3\right)\left(\sum_{i=1}^m c_i^2\lambda_i\right)} \\
&= \frac{\left(\sum_{i=1}^m c_i^2\lambda_i^2\right)/\left(\sum_{i=1}^m c_i^2\lambda_i^3\right)}{\left(\sum_{i=1}^m c_i^2\lambda_i\right)/\left(\sum_{i=1}^m c_i^2\lambda_i^2\right)} \\
&= \frac{1/\left(\sum_{i=1}^m \eta_i\lambda_i\right)}{\sum_{i=1}^m (\eta_i/\lambda_i)} \\
&=: \frac{\phi(\eta)}{\psi(\eta)},
\end{aligned}$$

where we used the transformation $\eta_i = c_i^2\lambda_i^2 / \sum_{i=1}^m (c_i^2\lambda_i^2)$ in the third equality. At this point, the derived expression is equivalent to the one in the proof of the original Kantorovich inequality and we can now follow the proof directly from [14, (p. 237)]. We notice that $\sum_{i=1}^m \eta_i = 1$ and $0 \leq \eta_i \leq 1$ and with this insight we can see that $\phi(\eta)$ involves a convex combination of the terms λ_i, whereas $\psi(\eta)$ involves a convex combination of the terms $1/\lambda_i$. For a graphical interpretation of the ratio $\phi(\eta)/\psi(\eta)$, two curves on the interval $\lambda \in [\lambda_m, \lambda_1]$ are of interest: the convex function $1/\lambda$ and the straight line that connects the outer points of the former curve, $(\lambda_m, 1/\lambda_m)$ and $(\lambda_1, 1/\lambda_1)$ respectively. Here the first curve lies below the latter by convexity of $1/\lambda$. With regard to these curves, we can consider both $\phi(\eta)$ and $\psi(\eta)$ as points on the vertical line $\bar{\lambda} := (\sum_{i=1}^m \eta_i\lambda_i) \in [\lambda_m, \lambda_1]$.

- $\phi(\eta) = 1/\bar{\lambda}$, hence $\phi(\eta)$ is clearly a point on the curve $1/\lambda$.
- $\psi(\eta)$ is a convex combination of the terms $1/\lambda_i$. By a generalization of the definition of a convex function, called *Jensen's inequality*, it is straightforward to derive that $\psi(\eta)$ is a point somewhere between the curve $1/\lambda$ and the straight line that connects the outer points.

We are interested in an upper-bound for the ratio $\phi(\eta)/\psi(\eta)$, i.e. we are interested in the minimization of this ratio. Now on the vertical line $\bar{\lambda}$ the value of $\phi(\eta)$ is fixed; on the other hand, $\psi(\eta)$ can be at most the value that is attained on the straight line. This would minimize the ratio $\phi(\eta)/\psi(\eta)$ on the vertical line $\bar{\lambda}$. Indeed such a choice for η always exists: Simply take a suitable convex combination of just the outer points of the interval $[\lambda_m, \lambda_1]$. In this way we could actually consider any vertical line on the interval $[\lambda_m, \lambda_1]$. Hence, upon minimizing the ratio $\phi(\eta)/\psi(\eta)$, the only convex combinations that are of interest are those corresponding to $\tilde{\eta}$ with $\tilde{\eta}_2 = \cdots = \tilde{\eta}_{m-1} = 0$ and $\tilde{\eta}_1 + \tilde{\eta}_m = 1$. Naturally we require $\tilde{\eta}_1, \tilde{\eta}_m \in [0, 1]$. For this particular vector $\tilde{\eta}$ we define $\lambda = \tilde{\eta}_1\lambda_1 + \tilde{\eta}_m\lambda_m$ and observe that $\tilde{\eta}_1/\lambda_1 + \tilde{\eta}_m/\lambda_m = (\lambda_1 + \lambda_m - \tilde{\eta}_1\lambda_1 - \tilde{\eta}_m\lambda_m)/\lambda_1\lambda_m$. Now we are able to give the following lower bound for any other convex combination η:

$$\frac{\phi(\eta)}{\psi(\eta)} \geq \min_{\lambda\in[\lambda_m,\lambda_1]} \frac{1/\lambda}{(\lambda_1 + \lambda_m - \lambda)/\lambda_1\lambda_m}.$$

One can show that the minimum is attained at $\lambda = (\lambda_1 + \lambda_m)/2$, such that we derive the following explicit lower bound:

$$\frac{\phi(\eta)}{\psi(\eta)} \geq \frac{4\lambda_1\lambda_m}{(\lambda_1+\lambda_m)^2}.$$

□

With the Kantorovich inequality for underdetermined systems we are now able to formulate and prove the convergence of the SD method applied to the system $A^TAx = A^Tb$.

Theorem 5.6 (SDud)**.** *Given any $x_{(0)} = z_{(0)} + u_{(0)}$, with $z_{(0)} \in R(A^T)$ and $u_{(0)} \in \ker(A)$, the **SDud** method of equation (5.2) converges linearly to the projected solution $x_P^* = z^* + u_{(0)} \in \mathcal{X}^*$. Furthermore, we have at every iteration $t+1$ the following inequality:*

$$\|Ae_{(t+1)}\|_2^2 \leq \left(\frac{\kappa-1}{\kappa+1}\right)^2 \|Ae_{(t)}\|_2^2, \quad \textit{with } \kappa = \frac{\lambda_1}{\lambda_m}. \tag{5.13}$$

Here we have for any t that $\|Ae_{(t)}\|_2 = 0$ if and only if $e(t) = 0$.

Proof. We have already observed the correctness of the last statement, namely that $\|Ae_{(t+1)}\|_2 = 0$ if and only if $e(t) = 0$. In order to prove the inequality (5.13), we define $v = e_{(t)} \in R(A^T)$ and $w = p_{(t)} = A^TAe_{(t)} = A^TAv$. Then we combine the results of lemma 5.4 and 5.5 to observe that:

$$\begin{aligned}
\|Ae_{(t+1)}\|_2^2 &\leq \left(1 - \frac{4\lambda_1\lambda_m}{(\lambda_1+\lambda_m)^2}\right)\|Ae_{(t)}\|_2^2 \\
&= \left(\frac{\lambda_1-\lambda_m}{\lambda_1+\lambda_m}\right)^2 \|Ae_{(t)}\|_2^2 \\
&= \left(\frac{\kappa-1}{\kappa+1}\right)^2 \|Ae_{(t)}\|_2^2,
\end{aligned}$$

where we defined $\kappa = \frac{\lambda_1}{\lambda_m}$. □

We end this section with a pseudo code for an efficient implementation of the SDud method. We mention that the explicit computation of A^TA is always prevented, since this is an expensive operation both in terms of computational effort and memory storage. Whenever a matrix vector product $z = A^TAx$ is required, this is done in two steps: First we compute $y = Ax$ and then $z = A^Ty$. The SDud update scheme requires only one such a matrix vector product for the update of the residual at each iteration, see equations (5.2) and (5.3). The additional benefit of the two step computation of this matrix vector product, is that we only need to do one computation with A and one with A^T at each iteration. We may obtain the efficient implementation in algorithm 1, given below. Notice that this implementation minimizes the storage of vectors terms as well.

input : $A \in \mathbb{R}^{m\times n}(m < n),\ b \in \mathbb{R}^m,\ x_{(0)} \in \mathbb{R}^n$.

% initialization
$x = x_{(0)}$
$r = b - Ax$
$p = A^T r$

% start SDud
for $t = 1, 2 \ldots, t_{\max}$ **do**
  % compute $\alpha_{(t)}$
  $q = Ap$
  $\rho = p^T p,\ \sigma = q^T q,\ \alpha = \rho/\sigma$
  % check for convergence
  if $\rho \leq$ `tol` **then** `break`
  % compute new iterate $x_{(t+1)}$
  $x \leftarrow x + \alpha p$
  % update residual $p_{(t+1)}$
  $w = A^T q$
  $p \leftarrow p - \alpha w$
end

Algorithm 1: SDud$(A, b, x_{(0)})$

5.1.2 Conjugate gradient for underdetermined systems

In the previous section we observed that the SDud method for underdetermined systems, $Ax = b$, searches for a unique solution $x_P^* = z^* + u_0$ in the affine space $u_0 + R(A^T)$, given some initial guess $x_0 = z_0 + u_0$ with $z_0 \in R(A^T)$ and $u_0 \in \ker(A)$. This poses the question whether this method can be optimized, similarly to the way that the conjugate gradient method optimizes the steepest descent method for SPD systems $Qx = q$. We follow up on Section 2.3.3 and show by a very similar reasoning that this is indeed possible. For this we will need the notion of A^TA-orthogonal vectors in the space $R(A^T)$, i.e. vectors $v, w \in R(A^T)$ that satisfy $v^TA^TAw = 0$. Analogously to the original CG method, it is possible to sequentially generate a linearly independent set of A^TA-orthogonal vectors that spans the space $R(A^T)$. It is actually done with an equivalent form of update scheme (2.32), all we need to do is to replace A by A^TA and $r_{(t)}$ by $p_{(t)}$. In this way we obtain the CG method for underdetermined systems:

$$\begin{aligned} \alpha_{(t)} &= \frac{p_{(t)}^T d_{(t)}}{d_{(t)}^T A^T A d_{(t)}} \\ x_{(t+1)} &= x_{(t)} + \alpha_{(t)} d_{(t)} \\ \beta_{(t)} &= \frac{p_{(t+1)}^T A d_{(t)}}{d_{(t)}^T A^T A d_{(t)}} \\ d_{(t+1)} &= p_{(t+1)} + \beta_{(t)} d_{(t)}. \end{aligned} \tag{5.14}$$

Similarly to the `SDud` method, we refer to this method by `CGud` for short. We mention that the scheme is initialized with $d_{(0)} = r_{(0)}$. In the remainder of this section we will prove the correctness of this update scheme and show that it converges within m iterations; the same number as the dimension of the affine space $u_0 + R(A^T)$. We are able to obtain this result by refining the arguments in [14, Chapter 9].

Suppose that we know beforehand a linearly independent set $d_{(0)}, \ldots, d_{(m-1)}$ of A^TA-orthogonal vectors, spanning the m-dimensional space $R(A^T)$. If these are the search directions we use, then we know by lemma 5.2 that $x_{(t)} = z_{(t)} + u_{(0)}$ for any $t \leq m$, for some $z_{(t)} \in R(A^T)$. This means, again, that we are implicitly doing an update of the form $z_{(t+1)} = z_{(t)} + \alpha_{(t)} d_{(t)}$ at each iteration. Now suppose that at any iteration $t+1$ we ensure that:

$$z_{(t+1)} = \underset{z \in z_{(0)} + \text{span}\{d_{(0)}, \ldots, d_{(t)}\}}{\text{argmin}} \|A(z^* - z)\|_2, \tag{5.15}$$

then clearly the method converges within m iterations. The first thing we should ask ourself is whether a linearly independent set $d_{(0)}, \ldots, d_{(m-1)}$ of A^TA-orthogonal vectors, spanning the space $R(A^T)$, exists. This is question is readily answered, simply consider the set $v_1, \ldots, v_m$ of m orthogonal eigenvectors of A^TA, spanning the space $R(A^T)$, and observe that this set has the property that $v_j^T A^T A v_i = \lambda_i v_j^T v_i = 0$ for any $i \neq j$. In the next lemma we show that any such set is always linearly independent, such that any other such set also spans the space $R(A^T)$.

Lemma 5.7. *Any set $d_{(0)}, \ldots, d_{(m-1)}$, with $d_{(i)} \in R(A^T)$, of A^TA-orthogonal vectors is linearly independent.*

Proof. We consider any linear combination of such a set and see under what conditions such a combination is zero:

$$\alpha_0 d_{(0)} + \cdots + \alpha_{m-1} d_{(m-1)} = 0.$$

Pre-multiplying with $d_{(i)}^T A^T A$ on both sides we find that for any i it should hold that $\alpha_i d_{(i)}^T A^T A d_{(i)} = \alpha_i \|A d_{(i)}\|_2^2 = 0$ for all i. This clearly means that $\alpha_i = 0$ for all i and we conclude that the set is indeed linearly independent. □

Now that we know of the existence of such a set, we will show that an exact line search with this set, at any iteration, suffices for the minimization of equation (5.15). This is shown by a variant of the *Expanding subspace theorem*, which we mentioned before, although briefly, in Section 2.3.3. For this purpose we denote by $\mathcal{K}_k$ the space spanned by the vectors $d_{(0)}, \ldots, d_{(k-1)}$, for some $k \leq m$. Not coincidentally this is the same notation we use for a Krylov space, equation (2.33), because we will see later on that this equals the space $\mathcal{K}_k(A^TA, p_{(0)})$ for the `CGud` scheme of equation (5.14).

Theorem 5.8 (Expanding subspace theorem for underdetermined systems)**.** *Let $d_{(0)}, \ldots, d_{(m-1)}$ be a set of A^TA-orthogonal vectors that spans the space $R(A^T)$ and let the following iteration scheme be initiated by any $x_{(0)} \in \mathbb{R}^n$:*

$$\begin{aligned} x_{(t+1)} &= x_{(t)} + \alpha_{(t)} d_{(t)} \\ \alpha_{(t)} &= \frac{p_{(t)}^T d_{(t)}}{d_{(t)}^T A^T A d_{(t)}}. \end{aligned}$$

Then for any iteration $t = k$, $x_{(k)}$ minimizes $\|A(x^* - x)\|_2^2 = \|A(z^* - z)\|_2^2$ on the line $x_{(k-1)} + \alpha d_{(k-1)}$ as well as on the affine space $x_{(0)} + \mathcal{K}_k$.

Proof. For the sake of completeness, we observe that the theorem implicitly claims that $z_{(k)} = z_{(k-1)} + \alpha_{(k-1)} d_{(k-1)}$ solves the minimization problem (5.15) on the affine space $z_{(0)} + \mathcal{K}_k$. Now since $x_{(k-1)} + \alpha d_{(k-1)} \subseteq x_{(0)} + \mathcal{K}_k$, we only need to show that the iteration scheme minimizes $f(x) := \|A(x^* - x)\|_2^2$ on the affine space $x_{(0)} + \mathcal{K}_k$. Here we observe that f is a convex function with its domain on the convex set $x_{(0)} + \mathcal{K}_k$. Hence, with the first-order optimality criterion of lemma 2.6 in mind, Section 2.1, we argue that it is sufficient to show that $\nabla f(x_{(k+1)}) = p_{(k+1)}$ is orthogonal to the space $\mathcal{K}_k$. That is, we need to show that $p_{(k+1)}^T d_{(i)} = 0$ for all $i = 1, \ldots, k$. Here we mention that the update $p_{(k+1)} = p_{(k)} - \alpha_{(k)} A^T A d_{(k)}$ applies to the residual. From here on the proof is by induction. The case $k = 0$ is trivial since $\mathcal{K}_0$ is empty. Now we assume the induction hypothesis for $k - 1$; under this assumption, we need to show $p_{(k+1)} \perp \mathcal{K}_k$. We first consider $i = k$ and observe that:

$$p_{(k+1)}^T d_{(k)} = p_{(k)}^T d_{(k)} - \alpha_{(k)} d_{(k)}^T A^T A d_{(k)} = 0,$$

which follows by definition $\alpha_{(k)}$. For the case $i < k$ we require:

$$p_{(k+1)}^T d_{(i)} = p_{(k)}^T d_{(i)} - \alpha_{(k)} d_{(k)}^T A^T A d_{(i)} = 0.$$

This holds since the first term cancels due to the induction hypothesis, and the second term cancels due to $A^T A$-orthogonality of the set $d_{(0)}, \ldots, d_{(m-1)}$. □

A useful property that follows from the proof above, is formulated in the following corollary.

Corollary 5.9. *The iteration scheme of theorem 5.8 has the property that at any iteration $t = k \leq m$:*

$$p_{(k+1)}^T d_{(i)} = 0 \quad \textit{for any} \quad i \leq k. \tag{5.16}$$

With this result we emphasize that, as long as $p_{(t+1)} \neq 0$, it is always ensured that the residual is linearly independent of all previous search directions. Now, as a last step, we need to show that the `CG` method for underdetermined systems (5.14) produces $A^T A$-orthogonal directions from the current residual $p_{(t+1)} = -\nabla f(x_{t+1})$. Then, from lemma 5.7 and corollary 5.9, we know that the new search direction is nonzero and linearly independent from the previous search directions, such that we may conclude from theorem 5.8 that the update scheme converges indeed within m iterations. This result is stated in the following lemma, analogously to lemma 2.9.

Lemma 5.10 (`CGud` properties)**.** *If the `CGud` method (5.14) does not terminate at iteration $t = k$, then the following properties hold:*

(i) $\mathrm{span}\{p_{(0)}, p_{(1)}, \ldots, p_{(k)}\} = \mathrm{span}\{p_{(0)}, A^T A p_{(0)}, \ldots, (A^T A)^k p_{(0)}\}$;

(ii) $\mathrm{span}\{d_{(0)}, d_{(1)}, \ldots, d_{(k)}\} = \mathrm{span}\{p_{(0)}, A^T A p_{(0)}, \ldots, (A^T A)^k p_{(0)}\}$;

(iii) $d_{(k)}^T A^T A d_{(i)} = 0 \quad \textit{for any} \quad i < k.$

Proof. Follow the arguments in [14, (p. 270)] analogously, with the matrix A^TA and the residuals $p_{(t)}$. This gives a direct proof for the three statements. □

Another useful result that we obtain by analogously following the proof in [14, (p. 270)], is the existence of alternative expressions for $\alpha_{(t)}$ and $\beta_{(t)}$, these are given in the following lemma.

Lemma 5.11. *Equivalent expressions for $\alpha_{(t)}$ and $\beta_{(t)}$ for the `CGud` method (5.14) are given as follows:*

$$\begin{aligned} \alpha_{(t)} &= \frac{p_{(t)}^T p_{(t)}}{d_{(t)}^T A^T A d_{(t)}}; \\ \beta_{(t)} &= \frac{p_{(t+1)}^T p_{(t+1)}}{p_{(t)}^T p_{(t)}}. \end{aligned} \tag{5.17}$$

The alternative expressions that are given in the above lemma are much more convenient for computational purposes; considering that the terms $p_{(t)}^T p_{(t)}$ and $p_{(t+1)}^T p_{(t+1)}$ are required to determine $\alpha_{(t)}$ and $\alpha_{(t+1)}$ respectively, the additional computational effort of determining the constant $\beta_{(t)}$ has become negligible.

The fact that `CG` is usually considered to be an iterative method, rather than a method that is executed until completion, is something that we will briefly discuss now. Similar to the `CG` method for SPD systems, it can be derived that the `CGud` method converges linearly to the the unique solution $x^* = z^* + u_{(0)}$, given any initial guess $x_{(0)} = z_{(0)} + u_{(0)}$. Of course, that is until it terminates after at most m iterations. For the original `CG` method this is argued in [14, Section 9.4, Section 9.8 (p. 282)]. We omit the details and give only the key observations needed to show that these arguments still apply, in a very similar manner.

First of all, following [14, Section 9.4], we observe from property (ii) in lemma 5.10, that the iterate $x_{(k+1)}$ can always be written as:

$$x_{(k+1)} = x_{(0)} + P_k(A^TA)p_{(0)}.$$

Here P_k is some polynomial of degree k. In our case we may eliminate the component $u_{(0)}$ on both sides to equivalently obtain that:

$$z_{(k+1)} = z_{(0)} + P_k(A^TA)p_{(0)}.$$

With this notation it is straightforwardly derived that:

$$\begin{aligned} f\left(x_{(k+1)}\right) &:= \|A(x^* - x_{(k+1)})\|_2^2 \\ &= \|A(z^* - z_{(k+1)})\|_2^2 \\ &= \left(z^* - z_{(k+1)}\right) A^T A \left(z^* - z_{(k+1)}\right) \\ &= \left(z^* - z_{(0)}\right) A^T A \big(I + A^T A P_k(A^TA)\big)^2 \left(z^* - z_{(0)}\right). \end{aligned}$$

Here we mention that the `CGud` scheme (5.15) implicitly minimizes this term over all possible polynomials P_k, at each iteration.

input : $A \in \mathbb{R}^{m\times n}(m < n),\ b \in \mathbb{R}^m,\ x_{(0)} \in \mathbb{R}^n$.

% initialization
$x = x_{(0)}$
$r = b - Ax$
$p = A^T r$
$d = p$
$\rho = p^T p$

% start `CGud`
for $t = 1, 2 \ldots, t_{\max}$ **do**

% check for convergence
if $\rho \leq$ `tol` **then** `break`

% compute $\alpha_{(t)}$
$q = Ad$
$\sigma = q^T q,\ \alpha = \rho/\sigma$

% compute new iterate $x_{(t+1)}$
$x \leftarrow x + \alpha d$

% update residual $p_{(t+1)}$
$w = A^T q$
$p \leftarrow p - \alpha w$

% compute $\beta_{(t+1)}$
$\rho' = \rho$
$\rho = p^T p,\ \beta = \rho/\rho'$

% compute new search direction $d_{(t+1)}$
$d \leftarrow p + \beta d$

end

Algorithm 2: `CGud`$(A, b, x_{(0)})$

Next we observe that $z^* - z_{(0)} \in R(A^T) = R(A^T A)$, such that we can write this term as a linear combination of the eigenvectors of $A^T A$, similar to equation (5.9). This is actually the key observation that justifies the claim that the reasoning of [14, Section 9.4] still applies. We recall the notation $e_{(t)} = z^* - z_{(t)}$, then it is concluded from [14, theorem 2 (p. 273)] that we have the following fundamental inequality:

$$\|Ae_{(k+1)}\|_2^2 = \max_{\lambda_i} \left\{ \left(1 + \lambda_i P_k(\lambda_i)\right)^2 \|Ae_{(0)}\|_2^2 \right\},$$

for any polynomial P_k of degree k. Note that the maximum is taken over all eigenvalues $\lambda_1, \ldots, \lambda_m$. From this inequality we may obtain a convergence theorem in the same way the `CG` convergence theorem 2.10 is derived in [14, Section 9.8 (p. 282)].

Theorem 5.12 (CGud)**.** *For any initial guess $x_{(0)} = z_{(0)}+u_{(0)}$, with $z_{(0)} \in R(A^T)$ and $u_{(0)} \in \ker(A)$, the CGud method of equation (2.32) terminates in at most m iterations. Before termination, the conjugate gradient method converges linearly to the projected solution $x_P^* = z^* + u_{(0)}$ in the affine space $u_{(0)} + R(A^T)$. Furthermore, at any iteration $t+1$ we have:*

$$\|Ae_{(t+1)}\|_2^2 \leq 4\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^2 \|Ae_{(t)}\|_2^2, \quad \textit{with} \quad \kappa = \frac{\lambda_1}{\lambda_m}. \tag{5.18}$$

We additional end this section with an efficient implementation of the CGud as well. Similarly to the SDud method, we require only one computation with A and one with A^T at each iteration. This remains possible due to the alternative expression for $\alpha(t)$ and $\beta_{(t)}$ of lemma 5.11. The pseudo code is given in algorithm 2. Notice that at each iteration t we need to store the term $p_{(t)}^T p_{(t)}$ for the computation of $\beta_{(t+1)}$ in the next iteration.

5.2 Distributed steepest descent

In this section we introduce the *distributed steepest descent* (DSD) method. We propose two variants for this method, which we refer to by respectively the α- and β-variant of DSD. For short we denote them by respectively α-DSD and β-DSD.

The α-variant of DSD

By recalling what is known about the decentralized gradient descent (DGD) method, Section 4.1.2, and the kernel projection (KP) method, Section 4.2.2, combined with the findings for the SDud and CGud methods of the previous section, we argue that an intermediate approach is indeed given by an update of the form (3.15), where the first step is a local single step of the SDud method. That is, we propose an update scheme as follows:

$$\begin{aligned} y_{(t)}^i &= x_{(t)}^i + \alpha_{(t)}^i p_{(t)}^i \\ x_{(t+1)}^i &= \sum_{j \in \mathcal{N}_i} w_{ij} y_{(t)}^j, \end{aligned} \tag{5.19}$$

where we have:

$$\alpha_{(t)}^i = \begin{cases} 0 & \text{if } p_{(t)}^i = 0, \\ \frac{\|p_{(t)}^i\|_2^2}{\|A_i p_{(t)}^i\|_2^2} & \text{otherwise.} \end{cases} \tag{5.20}$$

Note that $\alpha_{(t)}^i$ is well-defined by the result of lemma 5.1. For reasons that become clear in the next section, we assume that the underlying graph is undirected and that the matrix W is doubly stochastic. For practical purposes we may even assume that W is the Metropolis-Hasting matrix.

By defining $p_{(t)} = [p_{(t)}^i]$ and $D_\alpha = \text{diag}\{\alpha_{(t)}^1, \ldots, \alpha_{(t)}^M\}$, we obtain a compact form notation for (5.19) as follows:

$$x_{(t+1)} = \bar{W} x_{(t)} - \bar{W}\bar{D}_\alpha p_{(t)}. \tag{5.21}$$

We recall that we defined $\mathcal{A} = \text{diag}\{A_1, \ldots, A_M\}$ in Section 4.2.1. In case that we apply the DSD method to a consistent linear system, we observe in a very

similar manner that it holds that $p_{(t)} = \mathcal{A}^T\mathcal{A}e_{(t)}$, where we have the error quantity $e_{(t)} = [e^i_{(t)}]$ with $e^i_{(t)} = x^* - x^i_{(t)}$. Then, the fundamental iteration for the error is given by:

$$\begin{aligned} e_{(t+1)} &= \bar{W}e_{(t)} - \bar{W}\bar{D}_\alpha\mathcal{A}^T\mathcal{A}e_{(t)} \\ &= \bar{W}(\bar{I} - \bar{D}_\alpha\mathcal{A}^T\mathcal{A})e_{(t)}. \end{aligned} \tag{5.22}$$

The β-variant of DSD

The choice of $\alpha^i_{(t)}$ is such that it minimizes the residual based function $f(x) = \frac{1}{2}\|b_i - A_i x\|_2^2$ on the line $x^i_{(t)} + \alpha p^i_{(t)}$. In Appendix A we show that we may replace $\alpha^i_{(t)}$ by another term $\beta^i_{(t)}$ that minimizes the distance based function $f(x) = \frac{1}{2}\text{dist}(x, \mathcal{X}_i^*)^2 = \frac{1}{2}\|z_i^* - z_i\|$ on the line $x^i_{(t)} + \alpha p^i_{(t)}$. We also show that the convergence behavior of the SDud method remains very similar. The term $\beta^i_{(t)}$ is given by:

$$\beta^i_{(t)} = \begin{cases} 0 & \text{if } r^i_{(t)} = 0, \\ \frac{\|r^i_{(t)}\|_2^2}{\|p^i_{(t)}\|_2^2} & \text{otherwise.} \end{cases} \tag{5.23}$$

Hence, the update scheme for β-DSD is given by:

$$\begin{aligned} y^i_{(t)} &= x^i_{(t)} + \beta^i_{(t)}p^i_{(t)} \\ x^i_{(t+1)} &= \sum_{j\in\mathcal{N}_i} w_{ij}y^j_{(t)}. \end{aligned} \tag{5.24}$$

The compact form notation is given by:

$$x_{(t+1)} = \bar{W}x_{(t)} - \bar{W}\bar{D}_\beta p_{(t)}, \tag{5.25}$$

and the fundamental iteration for the error $e_{(t)}$, in compact notation, is given by (cf. (5.22)):

$$e_{(t+1)} = \bar{W}(\bar{I} - \bar{D}_\beta\mathcal{A}^T\mathcal{A})e_{(t)}. \tag{5.26}$$

5.2.1 Convergence analysis

In this section we do a brief analysis for the DSD methods, when applied to consistent linear systems $Ax = b$. The nonlinear terms contained in both D_α and D_β adds to the complexity of the analysis for both α-DSD and β-DSD respectively. In particular for the α-variant we observe that it is difficult to interpret the fundamental iteration for the error $e_{(t)}$ (5.22), since the single step of the α-variant of the local SDud method is based on the minimization of residual norm $\|A_i e^i_{(t)}\|_2$.

We restrict ourself to the β-DSD method, since the single step of β-variant of the local SDud method is based on the minimization of actual error $\|e^i_{(t)}\|_2$. In this case, it is more straightforward to interpret the fundamental iteration for error $e_{(t)}$ (5.26). From here on we denote the α- and β-variant of the SDud method by α-SDud and β-SDud respectively. Now to start, we present the following lemma.

Lemma 5.13. *For the β-DSD method (5.24), we have that the single step of β-SDud is such that:*

$$\|\tilde{e}^i_{(t)}\|_2 < \|e^i_{(t)}\|_2,$$

where $\tilde{e}^i_{(t)} = x^* - y^i_{(t)}$ *and* x^* *is the unique solution of* $Ax = b$.

Proof. First observe that $x^* \in \mathcal{X}_i^*$ for any i. Then by the orthogonality property $x^* - y^i_{(t),P} \perp y^i_{(t),P} - y^i_{(t)}$ and the properties of the 2-norm, we may derive that:

$$\begin{aligned} \|x^* - y^i_{(t)}\|_2^2 &= \|x^* - y^i_{(t),P} + y^i_{(t),P} - y^i_{(t)}\|_2^2 \\ &= \|x^* - y^i_{(t),P}\|_2^2 + \|y^i_{(t),P} - y^i_{(t)}\|_2^2. \end{aligned}$$

By the convergence properties of the β-`SDud` method, cf. theorem A.1, we then find that:

$$\begin{aligned} \|x^* - y^i_{(t)}\|_2^2 &= \|x^* - y^i_{(t),P}\|_2^2 + \|y^i_{(t),P} - y^i_{(t)}\|_2^2 \\ &\leq \|x^* - x^i_{(t),P}\|_2^2 + \left(\frac{\kappa_i - 1}{\kappa_i + 1}\right)^2 \|x^i_{(t),P} - x^i_{(t)}\|_2^2 \\ &< \|x^* - x^i_{(t),P}\|_2^2 + \|x^i_{(t),P} - x^i_{(t)}\|_2^2 \\ &= \|x^* - x^i_{(t)}\|_2^2, \end{aligned}$$

where we used that $y^i_{(t),P} = x^i_{(t),P}$ and defined $\kappa_i = \lambda_1(A_i^T A_i)/\lambda_M(A_i^T A_i)$. □

We were not able to establish a convergence proof for the β-`DSD` method, however, the result of the following lemma shows that the error $e_{(t)}$ strictly decreases at each step.

Lemma 5.14. *For the* β-`DSD` *method (5.24), we have at any iteration* $t+1$ *that:*

$$\|e_{(t+1)}\|_2^2 < \|e_{(t)}\|_2^2.$$

Proof. The proof is direct. First observe that $\tilde{e}_{(t)} = (\bar{I} - \bar{D}_\beta \mathcal{A}^T \mathcal{A}) e_{(t)}$, where $\tilde{e}_{(t)} = [\tilde{e}^i_{(t)}]$. Recall that we assumed W to be doubly stochastic; a consequence of the Birkhoff-von Neuman theorem, e.g. see [31], is that we have $\|W\|_2 = 1$ for any doubly stochastic matrix. Furthermore, observe that we generally have $\|x\|_2^2 = \sum_{i=1}^m \|x^i\|_2^2$ with $x = [x^i]$. Now we are able to show that:

$$\begin{aligned} \|e_{(t+1)}\|_2^2 &= \|\bar{W}\tilde{e}_{(t)}\|_2^2 \\ &\leq \|\bar{W}\|_2^2 \|\tilde{e}_{(t)}\|_2^2 \\ &= \|\tilde{e}_{(t)}\|_2^2 \\ &= \sum_{i=1}^M \|\tilde{e}^i_{(t)}\|_2^2 \\ &< \sum_{i=1}^M \|e^i_{(t)}\|_2^2 \\ &= \|e_{(t)}\|_2^2, \end{aligned}$$

where the strict inequality follows by the result of lemma 5.13. □

Note that the result of the previous lemma does not give a proof convergence. In the next section we will see that in case of nonsingular linear systems $Ax = b$, we have that $x = [x^*]$ is the only possible equilibrium point of both the α-`DSD` method (5.19) and the β-`DSD` method (5.24). Since these methods are intermediates of the `DGD` and `KP` methods, we still expect good convergence results.

5.2.2 Equilibrium points

In this section we consider the existence of equilibrium points for both of the variants of the DSD methods. We start by establishing a necessary condition for equilibrium points. To this end, let $x = [x^i]$ be a candidate for an equilibrium point. Then by definition of equilibrium points, x is an equilibrium point of α-DSD (5.25) if and only if:

$$x = \bar{W}x - \bar{W}\bar{D}_\alpha p, \tag{5.27}$$

where $p = [p^i]$ is the corresponding vector of least squares residuals. With this in mind, the following is a necessary condition on equilibrium points

Lemma 5.15. *If the point x is an equilibrium point to the α-DSD method, then:*

$$\sum_{i=1}^{M} \alpha_i p_i = 0, \tag{5.28}$$

Proof. Observe that a doubly stochastic matrix W is such that $\bar{\mathbb{1}}^T\bar{W} = \bar{\mathbb{1}}^T$, where $\bar{\mathbb{1}} = \mathbb{1} \otimes I$. By pre-multiplying (5.27) on both sides with $\bar{\mathbb{1}}^T$, we obtain that x is such that:

$$\begin{aligned} 0 &= \bar{\mathbb{1}}^T\bar{D}_\alpha p \\ &= \sum_{i=1}^{M} \alpha_i p_i = 0. \end{aligned}$$

□

The necessary condition for the β-DSD method is very similar, we state the result without proof.

Lemma 5.16. *If the point x is an equilibrium point to the β-DSD method, then:*

$$\sum_{i=1}^{M} \beta_i p_i = 0, \tag{5.29}$$

The necessary condition for the α-DSD method, lemma 5.15, can be simplified by the result of the following lemma.

Lemma 5.17. *The problem of finding nonzero least squares residuals $p_1, \ldots, p_M$ such that:*

$$\sum_{i=1}^{M} \alpha_i p_i = 0,$$

is equivalent to the problem of finding $q_1, \ldots, q_M$, $q_i \in R(A_i^T A_i)$, such that:

$$\sum_{i=1}^{M} q_i = 0.$$

Proof. Assume without loss of generality that all least squares residuals p_i are nonzero. If there is a zero residual p_k, we have $\alpha_k = 0$ and it follows from the proof that we may link p_k to another zero vector $q_k = 0$.

To show that the problems are equivalent, we need to show the proof goes both ways. First assume that we have a set $p_1, \ldots, p_M$ such that $\sum_{i=1}^{M} \alpha_i p_i = 0$. Then if we define $q_i = \alpha_i p_i \in R(A_i^T A_i)$, we see that q_i is a least squares residual by lemma 5.19. Furthermore it satisfies $\sum_{i=1}^{M} q_i = 0$.

The other way around, we assume that $q_1, \ldots, q_M$ is a set of least squares residuals such that $\sum_{i=1}^{M} q_i = 0$. Let us define:

$$p_i = \gamma_i q_i, \quad \text{where} \quad \gamma_i = \frac{\|A_i q_i\|_2^2}{\|q_i\|_2^2}.$$

Observe that $q_i \in R(A_i^T A_i)$. Then we observe that γ_i is defined such that we have:

$$\frac{1}{\alpha_i} = \frac{\|A_i p_i\|_2^2}{\|p_i\|_2^2} = \frac{\|\gamma_i A_i q_i\|_2^2}{\|\gamma_i q_i\|_2^2} = \frac{\|A_i q_i\|_2^2}{\|q_i\|_2^2} = \gamma_i,$$

which is sufficient to proof the equivalence relation. □

A very similar proof would give a similar equivalent result for the β-DSD method, but for the sake of brevity we choose to focus on only the α-DSD. We will now relate the solution to the equivalent problem to the set of all vectors in $\ker(A^T)$.

Lemma 5.18. *The problem of of finding all sets $q_1, \ldots, q_M$, with $q_i \in R(A_i^T A_i)$, such that:*

$$\sum_{i=1}^{M} q_i = 0$$

is equivalent to the problem of finding all vectors $\xi \in \ker(A^T)$.

Proof. ($\Rightarrow$) Suppose we are given a suitable set $q_1, \ldots, q_M$. We know there exists a particular vector $\tilde{z} = [\tilde{z}_i]$ such that:

$$\begin{aligned} [A_1^T A_1, \ldots, A_M^T A_M]\tilde{z} &= A_1^T A_1 \tilde{z}_1 + \ldots + A_M^T A_M \tilde{z}_M \\ &= q_1 + \ldots + q_M \\ &= 0. \end{aligned}$$

Or similarly:

$$\begin{aligned} [A_1^T A_1, \ldots, A_M^T A_M]\tilde{z} &= A_1^T A_1 \tilde{z}_1 + \ldots + A_M^T A_M \tilde{z}_M \\ &= A_1^T \xi_1 + \ldots + A_M^T \xi_M \\ &= A^T \xi \\ &= 0. \end{aligned}$$

Here we defined $\xi_i := A_i \tilde{z}_1$ and the vector $\xi = [\xi_i]$. We conclude that any set of vectors $q_1, \ldots, q_M$ is directly related to a particular vector $\xi \in \ker(A^T)$.

($\Leftarrow$) Conversely, let $\xi \in \ker(A^T)$ be given, then we know that:

$$\begin{aligned} A^T \xi &= A_1^T \xi_1 + \ldots + A_M^T \xi_M \\ &= A_1^T A_1 \tilde{z}_1 + \ldots + A_M^T A_M \tilde{z}_M \\ &= 0, \end{aligned}$$

here we introduced the linear system $A_i\tilde{z}_i := \xi$, which is always consistent since A_i is assumed to be full rank. □

We will summarize the results for finding all possible candidates for equilibrium points. For this we need the following useful lemma.

Lemma 5.19. *Any vector $p \in R(A_i^T A_i)$ can be written as a least squares residual, i.e. there always exists an x such that $p = A_i^T b_i - A_i^T A_i x$.*

Proof. Let $p \in R(A_i^T A_i)$, then there exists some z such that $p = A_i^T A_i z$. Since A_i is underdetermined and full rank, we know that there exists a unique $z_i^* \in R(A_i^T)$ such that $b_i = A_i z_i^*$. More generally, we have $b_i = A_i(z_i^* + u)$ for any $u \in \ker(A_i)$. Now let us define $x := z_i^* + u - z$, for any $u \in \ker(A_i)$, then we observe that $p = A_i^T b_i - A_i^T A_i x$. □

We summarize the results by giving the protocol that one should follow in order to find all points x that satisfy the necessary condition of 5.15.

1. initially:
   (a) compute the set $\ker(A^T)$;
   (b) compute the sets $\ker(A_i)$ for all i;
   (c) compute particular solution $z_i^* \in R(A_i^T)$ to the systems $A_i y_i = b_i$, for all i.
2. for any $\xi \in \ker(A^T)$, find the particular solutions $\tilde{z}_i^* \in R(A_i^T)$ to the systems $A_i z = \xi_i$, for all i.
3. to the particular solutions $\tilde{z}_i^*$, corresponding to some $\xi \in \ker(A^T)$, determine:
$$\gamma_i = \frac{\|A_i q_i\|_2^2}{\|q_i\|_2^2}, \qquad \text{where} \qquad q_i = A_i^T A_i \tilde{z}_i = A_i^T \xi_i,$$
   then compute $z_i = \gamma_i \tilde{z}_i$.
4. recall that $p_i = A_i^T A_i z_i$, corresponding to any $\xi \in \ker(A^T)$, defines a suitable set of least squares residuals, hence compute, according to lemma 5.19:
$$x_i = z_i^* + u_i - z_i,$$
   for all combinations of $u_i \in \ker(A_i)$, for all i.

Example 5.1 (candidates for equilibrium points and actual equilibrium points)
Suppose we have $M = 3$ agents and the partitioned matrix $[A \ b]$ with

$$A = \begin{bmatrix} 1 & -1 \\ 1 & 1 \\ -2 & 0 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} 2 \\ 4 \\ 0 \end{bmatrix},$$

such that each agents knows one row of the partitioned matrix. The example is constructed such that $x^* = \mathbb{1}$. We omit the calculations, but one could show that if we follow the protocol, then we find that all candidates are given by:

$$\begin{aligned} x_1 &= z_1^* + u_1 - z_1 = (c+1)\begin{bmatrix} 1 \\ -1 \end{bmatrix} + c_1 \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\ x_2 &= z_2^* + u_2 - z_2 = (c+2)\begin{bmatrix} 1 \\ 1 \end{bmatrix} + c_2 \begin{bmatrix} 1 \\ -1 \end{bmatrix} \\ x_3 &= z_3^* + u_3 - z_3 = c\begin{bmatrix} -2 \\ 0 \end{bmatrix} + c_3 \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \end{aligned}$$

for all $c, c_1, c_2, c_3 \in \mathbb{R}$. The actual equilibrium points, if any exist, is now determined by the matrix W. To this end, we consider the matrix:

$$W = \begin{bmatrix} \frac{2}{3} & \frac{1}{3} & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & \frac{1}{3} & \frac{2}{3} \end{bmatrix}.$$

If we carry on with the computations, we end up with a linear system in the coefficients c, c_1, c_2, c_3 as follows:

$$\begin{bmatrix} -3 & -1 & 1 & 0 \\ 3 & -1 & -1 & 0 \\ 6 & 0 & 1 & 0 \\ 0 & 0 & -1 & -1 \end{bmatrix} \begin{bmatrix} c \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} -1 \\ -3 \\ -2 \\ -2 \end{bmatrix}$$

In this particular case we have a consistent set of equations and there is a unique solution, namely $c = -\frac{1}{3}, c_1 = 2, c_2 = 0$ and $c_3 = 2$; the corresponding equilibrium point is given by:

$$x_1 = \frac{1}{3}\begin{bmatrix} 8 \\ 4 \end{bmatrix}, \quad x_2 = \frac{1}{3}\begin{bmatrix} 5 \\ 5 \end{bmatrix}, \quad x_3 = \frac{1}{3}\begin{bmatrix} 2 \\ 6 \end{bmatrix}.$$

which all differ significantly from the least squares solution $x^* = \mathbb{1}$. It is interesting to observe that this equilibrium seems to be asymptotically stable, since the algorithm gives the following numerical results after 50 iterations:

$$x_1 = \begin{bmatrix} 2.6666683 \\ 1.3333316 \end{bmatrix}, \quad x_2 = \begin{bmatrix} 1.6666683 \\ 1.6666602 \end{bmatrix}, \quad x_3 = \begin{bmatrix} 0.6666683 \\ 1.9999889 \end{bmatrix}.$$

■

The above example show that equilibrium points for linear least squares problems, if any exist, do generally not correspond to a consensus of the optimal solution x^*, and can differ significantly even for very small-sized problems. We finally mention that for nonsingular systems $Ax = b$, we have $\ker(A^T) = \{0\}$, such that $x = [x^*]$ is the only possible equilibrium. It is indeed a equilibrium point by lemma 5.28, since $p_1 = \ldots = p_M = 0$.

5.3 Numerical experiments

In this section we consider numerical experiments with the most important methods in this thesis. First we will confirm the convergence results for the α-DSD, β-SDud and CGud methods in Section 5.3.1. Then in the following two sections, we will consider numerical experiments with KP, DGD and both variants of DSD. First in Section 5.3.2 we consider the implementations of these methods. Next, in Section 5.3.3 we consider consistent linear systems and lastly, in Section 5.3.4 we consider linear least squares problems.

5.3.1 α-SDud, β-SDud and CGud

In this section we consider a small numerical experiment with α-SDud, β-SDud and CGud, such that we can compare the methods and confirm the obtained convergence results. We apply the methods to the underdetermined system $Ax = b$, with:

$$A = \begin{bmatrix} 1 & 1 & -2 & 0 \\ 0 & -2 & 3 & -1 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} 2 \\ -6 \end{bmatrix}. \tag{5.30}$$

We have chosen A such that we conveniently have that $\mathbb{1} \in \ker(A)$. It can be derived that the worst case convergence factor for both α-SDud and β-SDud, cf. theorem 5.6 and corollary A.1, is given by:

$$c := \left(\frac{\kappa - 1}{\kappa + 1}\right)^2 = 0.8, \tag{5.31}$$

where we recall that c is a worst case convergence factor for the residual norm $\|Ae_{(t)}\|_2^2$ for the α-SDudmethod, and for the error norm $\|e_{(t)}\|_2^2$ for the β-SDud method. Furthermore it can be derived that we have the particular solution $z^* \in R(A^T)$ given by:

$$z^* = A^T(AA^T)^{-1}b = \begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \end{bmatrix}. \tag{5.32}$$

We have initiated each method twice, where the two initial guesses were given by:

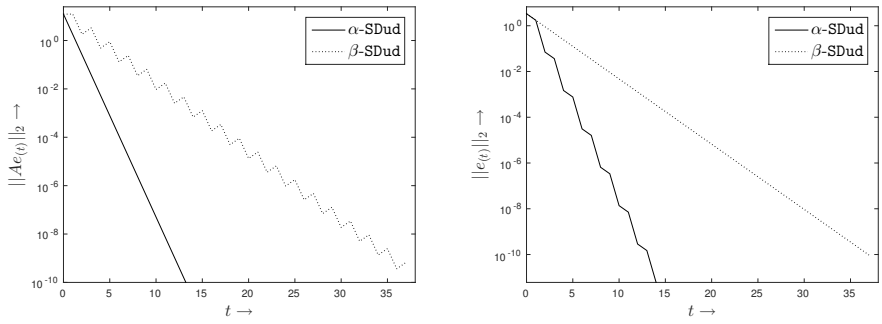
$$x_{(0)} = A^T y \quad \text{and} \quad x_{(0)} = A^T y + \mathbb{1}, \tag{5.33}$$

where y is a randomly chosen and fixed vector in $\mathbb{R}^2$. In these cases, we expect that each methods converges to respectively:

$$x_P = z^* = \begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \end{bmatrix} \quad \text{and} \quad x_P = z^* + \mathbb{1} = \begin{bmatrix} 0 \\ 2 \\ 0 \\ 2 \end{bmatrix}. \tag{5.34}$$

Before we consider the convergence result we make the following remark: Assuming that we have some initial guess $x_{(0)} = z_{(0)} + u_{(0)}$, the speed of convergence

can only depend on the part $z_{(0)} \in (A^T)$. After all, the part $u_{(0)}$ is preserved throughout the algorithm. That is, we expect the convergence behavior to be identical when we have some other initial guess $x_{(0)} = z_{(0)} + \tilde{u}_{(0)}$. This is precisely what we observed; the speed of convergence is identical for both initial guesses (5.33). Furthermore, the convergence behavior is indeed towards the projected solutions x_P that we hypothesized in equation (5.34). This is depicted in figure 5.1.



(a) Convergence residual norm.

(b) Convergence error norm.

Figure 5.1: For both initial guesses of equation (5.33), we have identical convergence behaviors in the residual norm (left) and the error norm (right), for both the α-`SDud` and β-`SDud` method. Here we have $e_{(t)} = x_P - x_{(t)}$, where x_P is the corresponding hypothesized projected solutions of equation (5.34) for either of the initial guesses.

Above all, we observe that α-`SDud` outperforms β-`SDud`; this has been observed for larger underdetermined systems $Ax = b$ as well. Apparently, the minimization of the residual norm is a better approach, which is perhaps also in concordance with the geometrical interpretation of the problem, see example 2.3. We also observe that residual norm minimization does not guarantee error norm minimization, as follows from figure 5.1a. The opposite statement is true as well, figure 5.1b. To confirm the convergence results of theorem 5.6 and corollary A.1, we observe the following. The α-`SDud` method convergences almost exactly linear (in the residual norm) with a convergence factor $c_\alpha \approx 0.0210 < c$, cf. equation (5.31). Similarly for the β-`SDud` method we have a linear convergence factor $c_\beta \approx 0.269 < c$ for the error norm.

With regard to the `CGud` method we finally mention that for both initial guesses (5.33), the method converged to the hypothesized projected solutions (5.34) in $t = 2$ iterations; that is convergence within machine precision. This is in agreement with the convergence result of theorem 5.12.

5.3.2 Implementations of DGD, KP and DSD

In the remaining sections, we will simulate distributed problems for the DGD method of Section 4.1.2, the KP method of Section 4.2.2 and both the α-DSD and β-DSD methods described in this chapter. In this section we will first provide the implementations of these simulations.

In the following two sections we will consider both consistent linear systems and linear least square problems, both of the form $Ax = b$. In all cases we will construct the right-hand side b such that x^* is known. For each method, we will consider the following relative error quantity $\varepsilon_{(t+1)}$ to analyze the convergence behavior:

$$\varepsilon_{(t+1)} := \frac{\|x^* - \bar{x}_{(t+1)}\|_2}{\|x^*\|_2}. \tag{5.35}$$

We mention that the average $\bar{x}_{(t+1)} = \frac{1}{M}\sum_{i=1}^{M} x^i_{(t+1)}$ can always be computed with a post-processing step according to the general consensus update (3.8); observe that we do require W to be doubly stochastic. We could initiate the post-processing step when a certain stopping criterion is met. Considering Section 5.2.2, we propose to define a stopping criterion based on reaching equilibrium. From a local point of view, this stopping criterion could for example be based on the quantity $x^i_{(t+1)} - x^i_{(t)}$. However, for the purpose of analyzing the simulation, we consider a global stopping criterion quantity $\Delta_{(t+1)}$ defined by:

$$\Delta_{(t+1)} := \frac{1}{M}\sum_{i=1}^{M} \frac{\|x^i_{(t+1)} - x^i_{(t)}\|_2}{\|x^i_{(t+1)}\|_2}. \tag{5.36}$$

Observe that the criterion $\Delta_{t+1} <$ tol indeed corresponds to a global reach of equilibrium, up to a user-defined precision tol.

For the sake of simplicity, we finally assume that each agents has a common amount of rows of the partitioned matrix $[A\ b]$, i.e. we have $m := m_i$ for all i. Hence, we have that $A \in \mathbb{R}^{mM \times n}$ with $n \leq mM$ and $A_i \in \mathbb{R}^{m \times n}$ with $n > m$. Additionally, we assume that the methods take as an input the adjacency matrix $N \in \mathbb{R}^{M \times M}$ corresponding to a undirected and connected graph $\mathcal{G}$. We mention that N contains the essential information about the degrees d_i, the neighbor sets $\mathcal{N}_i$ and consequently the information needed to compute doubly stochastic matrices W. Finally, we assume that $x^i_{(0)} = 0$ for all i and we will not consider this to be input.

Implementation of DGD

Recall from Section 4.1.2 that we have to determine:

$$L_g = \max_i\{L_{f_i}\} = \max_i\{\lambda_1(A_i^T A_i)\} \tag{5.37}$$

For this we will use build-in functions of MATLAB as well. We assume that the fixed step $\alpha(\gamma)$ then satisfies equation (4.18) for some $\gamma \geq 2$. Then we compute the corresponding shifted Metropolis-Hasting matrix $W(\gamma)$ by equation (4.19). We recall for the DCD update scheme (4.7) that $-\nabla f_{i,(t)} = p^i_{(t)} = A_i^T(b_i - A_i x^i_{(t)})$, and propose a pseudo code for simulations with the DCD method in algorithm 3. Here we omitted the computations with $\varepsilon_{(t)}$ for the sake of brevity.

input : $A_i \in \mathbb{R}^{m\times n}(m < n),\ b_i \in \mathbb{R}^m,\ N \in \mathbb{R}^{M\times M}, \gamma \geq 2.$

% initialization
compute L_g % eqn. (5.37)
$\alpha = \frac{1}{\gamma L_g}$ % eqn. (4.18)
$W = W(\gamma)$ % eqn. (4.19)

% start DGD
for $t = 1, 2 \ldots, t_{\max}$ **do**
  % DGD update
  for $i = 1, \ldots, M$ **do**
    $x^i_{prev} \leftarrow x^i$
    $p^i = A_i^T(b_i - A_i x)$
    $y^i = \sum_{j\in\mathcal{N}_i^*} w_{ij} x^i_{prev}$
    $x^i \leftarrow y^i + \alpha p^i$
  end
  % compute $\Delta_{(t)}$ of eqn. (5.36)
  $\Delta = 0$
  for $i = 1, \ldots, M$ **do**
    $\Delta \leftarrow \Delta + \frac{\|x^i - x^i_{prev}\|_2}{\|x^i\|_2}$
  end
  $\Delta \leftarrow \frac{1}{M}\Delta$
  % stopping criterion
  if $\Delta \leq$ `tol` **then** `break`
end

Algorithm 3: DGD$([A_i], [b_i], N, \gamma)$

Implementation of KP

Recall from Section 4.2.2 that the KP method is a two-step method, see equation (4.26). We also recall that $x^i_{(t+1)}$ may be obtained by a projection of the intermediate step $y^i_{(t)}$ onto $\mathcal{X}_i^*$. Here we proposed at the end of Section 4.2.2 to compute this projection, approximately, with the CGud method with initial guess $y^i_{(t)}$. We present a pseudo code for simulations with the KP method in algorithm 4. Here we denote the approximated projection by $\texttt{CGud}(A_i, b_i, y^i_{(t)}, t_{\max} = m, \texttt{tol})$, referring to the pseudo code of algorithm 2 and emphasizing that we allow at most $t_{\max} = m$ iterations according to the results of theorem 5.12. Also, the tolerance for the stopping criterion for CGud and KP are the same. We mention that computations with $\varepsilon_{(t)}$ are again omitted for the sake of brevity.

input : $A_i \in \mathbb{R}^{m\times n}(m < n),\ b_i \in \mathbb{R}^m,\ N \in \mathbb{R}^{M\times M}$.
% start KP
for $t = 1, 2 \ldots, t_{\max}$ **do**
  % KP update
  for $i = 1, \ldots, M$ **do**
    $x^i_{prev} = x^i$
    $y^i = \frac{1}{d_i+1} \sum_{j\in\mathcal{N}_i^*} x^i$
  end
  for $i = 1, \ldots, M$ **do**
    $x^i \leftarrow \texttt{CGud}(A, b, y^i, t_{\max} = m, \texttt{tol})$
  end
  % compute $\Delta_{(t)}$ of eqn. (5.36)
  $\Delta = 0$
  for $i = 1, \ldots, M$ **do**
    $\Delta \leftarrow \Delta + \frac{\|x^i - x^i_{prev}\|_2}{\|x^i\|_2}$
  end
  $\Delta \leftarrow \frac{1}{M}\Delta$
  % stopping criterion
  if $\Delta \leq$ tol **then** break
end

Algorithm 4: $\texttt{KP}([A_i], [b_i], N)$

Implementations of α-DSD and β-DSD

We present implementations of both the α-DSD and β-DSD methods. For the α-variant we could use for local updates, similar to the implementation of the KP method, the notation $\texttt{CGud}(A, b, x^i_{(t)}, t_{\max} = 1)$ (which is equivalent to $\texttt{SDud}(A, b, x^i_{(t)}, t_{\max} = 1)$ because of the identical first step). However, we prefer to give the actual computations according to a single step of the SDud scheme (5.2). Furthermore, for the mixing of the states we use the Metropolis-Hasting matrix W (3.13). In this way we present the pseudo code for simulations with the α-DSD method in algorithm 5.

input : $A_i \in \mathbb{R}^{m\times n}(m < n),\ b_i \in \mathbb{R}^m,\ N \in \mathbb{R}^{M\times M}$.

% initialization
compute W % eqn. (3.13)

% start α-DSD
for $t = 1, 2 \ldots, t_{\max}$ **do**

% α-DSD update
for $i = 1, \ldots, M$ **do**

$x^i_{prev} \leftarrow x^i$

% compute $\alpha^i_{(t)}$

$p^i = A_i^T(b_i - A_i x), \quad \rho^i = (p^i)^T p^i$

$q^i = A_i p^i, \quad \sigma^i = (q^i)^T q^i, \quad \alpha^i = \rho^i/\sigma^i$

% update

$y^i = x^i + \alpha^i p^i$

end

for $i = 1, \ldots, M$ **do**

$x^i \leftarrow \sum_{j\in\mathcal{N}_i^*} w_{ij} y^i$

end

% compute $\Delta_{(t)}$ of eqn. (5.36)

$\Delta = 0$

for $i = 1, \ldots, M$ **do**

$\Delta \leftarrow \Delta + \frac{\|x^i - x^i_{prev}\|_2}{\|x^i\|_2}$

end

$\Delta \leftarrow \frac{1}{M}\Delta$

% stopping criterion

if $\Delta \leq$ `tol` **then** `break`

end

Algorithm 5: α-DSD$([A_i], [b_i], N)$

The implementation β-DSD differs only in the local updates of the iterates $x^i_{(t)}$. For this we apply a single step of the β-SDud method (A.1). We present the pseudo code for local updates of the β-DSD method in algorithm 6. Observe that this method is computationally more efficient, since it saves a matrix-vector product with A_i at each iteration, cf. algorithm 5.

% β-DSD update
for $i = 1, \ldots, M$ **do**
  $x^i_{prev} \leftarrow x^i$
  % compute $\alpha^i_{(t)}$
  $r^i = b_i - A_i x, \qquad \rho^i = (r^i)^T r^i$
  $p^i = A_i^T r^i, \qquad \sigma^i = (p^i)^T p^i, \qquad \beta^i = \rho^i / \sigma^i$
  % update
  $y^i = x^i + \alpha^i p^i$
end
for $i = 1, \ldots, M$ **do**
  $x^i \leftarrow \sum_{j \in \mathcal{N}_i^*} w_{ij} y^i$
end

Algorithm 6: Local updates for β-DSD$([A_i], [b_i], N)$

5.3.3 DCD, KP and DSD with consistent linear systems

We will consider numerical experiments with DCD, KP, α-DSD and β-DSD. In this part we will consider consistent, square linear systems $Ax = b$. We have $A \in \mathbb{R}^{n \times n}$ and we need to choose m and M such that $mM = n$. We consider two type of matrices.

(i) Matrices $A = [a_{ij}]$, with a_{ij} randomly chosen from the standard uniform distribution on the open interval $(0, 1)$. For this we use the function `rand` in MATLAB.

(ii) Matrices $A = [a_{ij}]$, with a_{ij} randomly chosen from the standard uniform distribution on the open interval $(-1, 1)$.

In both cases A is completely dense and we mention that we ensure that A is well-conditioned by considering its 2-norm condition number with the MATLAB function `cond`. The right-hand side is constructed and defined as follows:

$$b = A\mathbb{1}, \tag{5.38}$$

such that the optimal solution $x^* = \mathbb{1}$ is known.

With regard to the network topology, we consider three types of graphs. To classify these graphs, we need the notion of *graph density* $\partial(\mathcal{G})$ for undirected graphs. First of all, a complete undirected graph with M vertices has the maximal amount of $\frac{1}{2}M(M-1)$ edges. Now, assuming that some undirected graph $\mathcal{G}$ has some $K \leq M$ edges, we define its graphs density as follows:

$$\partial(\mathcal{G}) = \frac{2K}{M(M-1)}. \tag{5.39}$$

Now the three types of graphs that we consider are as follows:

(i) Line graphs, i.e. graphs of the form $1 \leftrightarrow 2 \leftrightarrow 3 \ldots \leftrightarrow M$. Observe that:

$$\partial(\mathcal{G}) = \frac{2(M-1)}{M(M-1)} = \frac{2}{M}.$$

(ii) Circle graphs, i.e. graphs of the form $1 \leftrightarrow 2 \leftrightarrow 3 \ldots \leftrightarrow M \leftrightarrow 1$. Observe that:

$$\partial(\mathcal{G}) = \frac{2M}{M(M-1)} = \frac{2}{M-1}.$$

(iii) Random graphs with a pre-defined graph density $\partial(\mathcal{G})$. A strict upper-triangular matrix $T = [t_{ij}]$ is constructed with t_{ij} randomly chosen from the standard uniform distribution on the open interval $(0, 1)$. Then a threshold is chosen to drop small entries in T such that $\partial(\mathcal{G})$ is satisfied; the matrix N is constructed from the nonzero entries that remain in T.

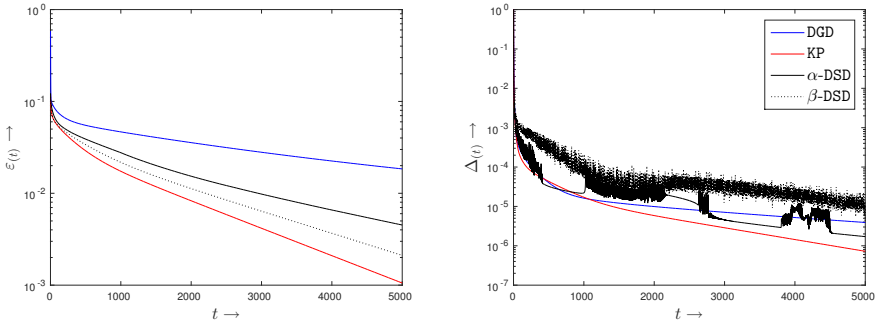
We motivate these choices as follows. The line graph is the most basic spanning tree, representing the most basic connected graph. The circle graph is very similar, but intuitively it should give a better flow of information over the network. Finally, the random graph represents a completely different type of connected graph, which is useful for comparison.

We finally mention that the user-defined variable γ for `DGD` is always set to 2, such that α is as large as possible. Then the guaranteed convergence, see theorem 4.4, is as fast a possible. Since x^* is unique, there is not necessarily the need to consider smaller choice for γ.

The following is a comparative study for all methods with different network topologies, for systems A of size 20×20.

Comparison of graph types

We have $A \in \mathbb{R}^{20\times 20}$ with A of the positive type (i), with condition number $\kappa(A) \approx 58.8$. We have $M = 5$ agents, each knowing $m = 4$ rows of the partitioned matrix $[A\ b]$. Furtermore, we have a tolerance of $\texttt{tol} = 10^{-8}$. We present the results with the line graph in figure 5.2. The convergence in the equilibrium $\Delta_{(t)}$

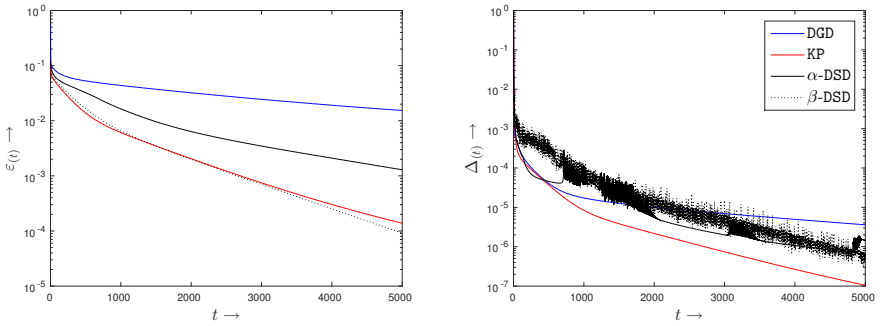


(a) Error convergence $\varepsilon_{(t)}$. **(b)** Equilibrium convergence $\Delta_{(t)}$.

Figure 5.2: Convergence results for the four methods with the line graph ($M = 5$), and $A \in \mathbb{R}^{20\times 20}$ of the positive type (i).

(5.36) is for both the α-`DSD` and β-`DSD` very unpredictable, while the convergence

in the error quantity $\varepsilon_{(t)}$ (5.35) is actually perfectly smooth. The KP method seems to outperform the other methods and the DGD performs the worst in this case. For comparison, we consider the convergence results with the circle graph, with $M = 5$ as well. We present the results in figure 5.3. We observe that the

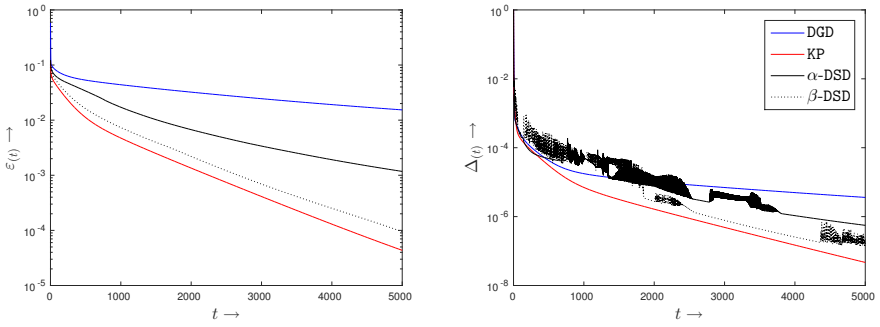


(a) Error convergence $\varepsilon_{(t)}$. **(b)** Equilibrium convergence $\Delta_{(t)}$.

Figure 5.3: Convergence results for the four methods with the circle graph ($M = 5$), and $A \in \mathbb{R}^{20\times 20}$ of the positive type (i).

overall convergence has improved, as we expected from the improved information flow for circle graphs. The convergence in the equilibrium remains non-smooth for α-DSD and β-DSD. However, with regard to convergence in the error quantity $\varepsilon_{(t)}$, we observe that β-DSD now slightly outperforms the KP method.

Notice that $\partial(\mathcal{G}) = \frac{2}{5} = 0.4$ for the line graph. For comparison, we considered a near complete, randomly produced graph with $\partial(cG) = 0.75$. The results are presented in figure 5.4, and we observe that the convergence behavior did not change significantly. We observe that the performance of $\beta-$DSD and KP are very similar.



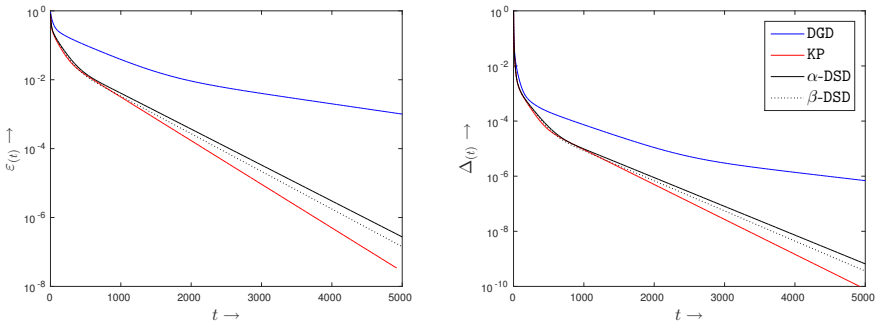
(a) Error convergence $\varepsilon_{(t)}$. **(b)** Equilibrium convergence $\Delta_{(t)}$.

Figure 5.4: Convergence results for the four methods with a random graph ($M = 5, \partial(\mathcal{G}) = 0.75$), and $A \in \mathbb{R}^{20\times 20}$ of the positive type (i).

For the KP method we mention that we observed that each agent executed on average, approximately $m = 4$ iterations of the CGud method for their local updates, at each global iteration t.

Other types of matrices

Following up on the previous work, we consider a matrix $A \in \mathbb{R}^{20\times 20}$ of the second type (ii), with condition number $\kappa(A) \approx 13.5$. For comparison, we present the results with the random graph ($M = 5, \partial(\mathcal{G}) = 0.75$), see figure 5.5.

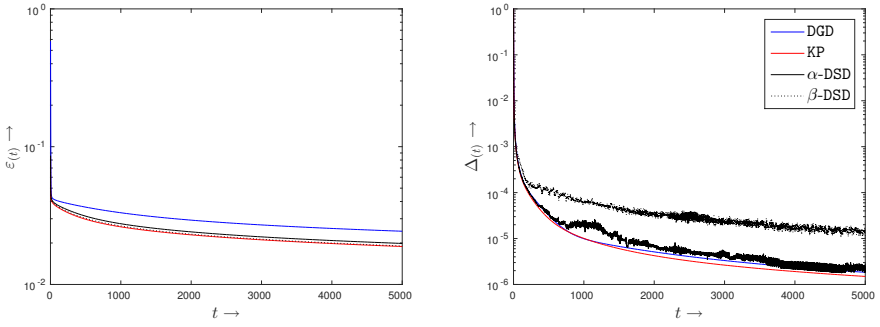


(a) Error convergence $\varepsilon_{(t)}$. **(b)** Equilibrium convergence $\Delta_{(t)}$.

Figure 5.5: Convergence results for the four methods with a random graph ($M = 5, \partial(\mathcal{G}) = 0.75$), and $A \in \mathbb{R}^{20\times 20}$ of the type (ii).

Scalability of the methods

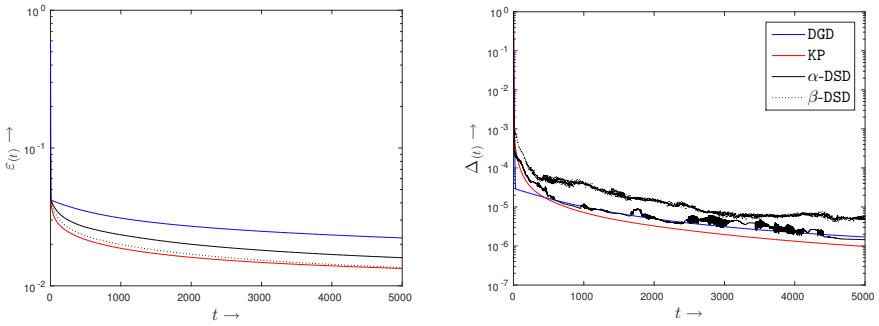
In this last part, we consider the scalability of the methods. To this end, we take a randomly computed matrix $A \in \mathbb{R}^{200\times 200}$ of type (i), with condition number $\kappa(A) \approx 1.58 \cdot 10^3$. Also, we let $M = 50$ such that we still have $m = 4$. All two-step methods of the form (3.15) or (3.16) are known to scale well with the network size, but for large dense network we typically have larger condition numbers, which is something that we need to take into account as well. We let $\texttt{tol} = 10^{-8}$ and consider the circle graph, with $\partial(\mathcal{G}) = \frac{2}{49} \approx 0.04$. The results are presented in figure 5.6.



(a) Error convergence $\varepsilon_{(t)}$. **(b)** Equilibrium convergence $\Delta_{(t)}$.

Figure 5.6: Convergence results for the four methods with the circle graph ($M = 50$), and $A \in \mathbb{R}^{200\times 200}$ of the positive type (i).

For comparison, we present the results with a random graph with density $\partial(\mathcal{G}) = 0.25$ in figure 5.7. We observe that a more dense graph leads to a



(a) Error convergence $\varepsilon_{(t)}$. **(b)** Equilibrium convergence $\Delta_{(t)}$.

Figure 5.7: Convergence results for the four methods with a random graph ($M = 50, \partial(\mathcal{G}) = 0.25$), and $A \in \mathbb{R}^{200\times 200}$ of the positive type (i).

marginal improvement of the convergence speed and it has not been observed to improve much by further increasing the graph density. Overall, we observe that scalability of the method is not very good in terms of convergence speed, which is probably due to the larger condition number of A.

Concluding remarks

With regard to the numerical experiments, we make some concluding remarks.

- When applied to consistent linear systems, `DGD` is outperformed by the other methods. Overall, the best performance was observed for `KP`.
- Overall, β-`DSD` performed slightly better than α-`DSD`. The performance of the β-`DSD` is generally very close to the performance of `KP`. However, we observed that `KP` generally required the full amount of m iterations for local `CGud` updates. From this point of view, we prefer β-`DSD` over `KP` for its computationally efficient iterations.
- Convergence towards equilibrium is generally much faster than convergence in the error. When all states are close to equilibrium, the convergence generally becomes extremely slow, especially for larger networks and larger dense systems.

5.3.4 `DCD`, `KP` and `DSD` with linear least squares problems

In this section we consider numerical experiments with linear least squars problems $Ax = b$. For this we have constructed randomly, similar to the previous section, full rank matrices $A \in \mathbb{R}^{mM\times n}$ with $mM > n$ and $m \leq n$. The right-hand side b is constructed as follows. First we define $\tilde{b} = A\mathbb{1}$, then we compute a random vector $u \in \ker(A^T)$ and define b as follows:

$$b = \tilde{b} + u. \tag{5.40}$$

Here we mention that a kernel matrix for A^T is readily obtained in MATLAB for modest size matrices. For the optimal solution we observe that $x^* = \mathbb{1}$ since we can derive that $A^T A\mathbb{1} = A^T b$. Also, observe that the optimal residual r^*

satisfies $r^* = b - A\mathbb{1} = u$, therefore we have chosen to scale u such that we always have:

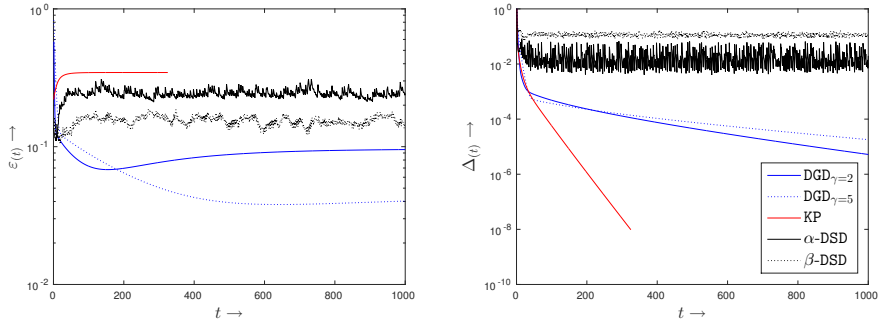
$$\frac{\|r^*\|_2}{\|\tilde{b}\|_2} = \frac{\|u\|_2}{\|\tilde{b}\|_2} = 0.1 \tag{5.41}$$

In this way we achieve two things, namely (i) we make sure that the results for different size matrices remain comparable and (ii) we have in some sense modeled a situation with a global relative measurement error of 10% (cf. equation (4.20)).

We mention that in the context of distruted least square problems, it makes sense to consider different values for γ, for the `DGD` method. To this end we consider two values, namely the minimal value $\gamma = 2$ and also $\gamma = 5$ for comparison. Following up on the previous section, we start with a comparative study for network topologies with a matrix $A \in \mathbb{R}^{20\times 10}$ and $M = 5$ agents.

Comparison of graph types

We have $A \in \mathbb{R}^{20\times 10}$ with A of the positive type (i), with condition number $\kappa(A) \approx 8.65$. We have $M = 5$ agents, each knowing $m = 4$ rows of the partitioned matrix $[A\ b]$. Furtermore, we have a tolerance of $\texttt{tol} = 10^{-8}$. We present the results with the line graph in figure 5.2. The convergence in the equilibrium



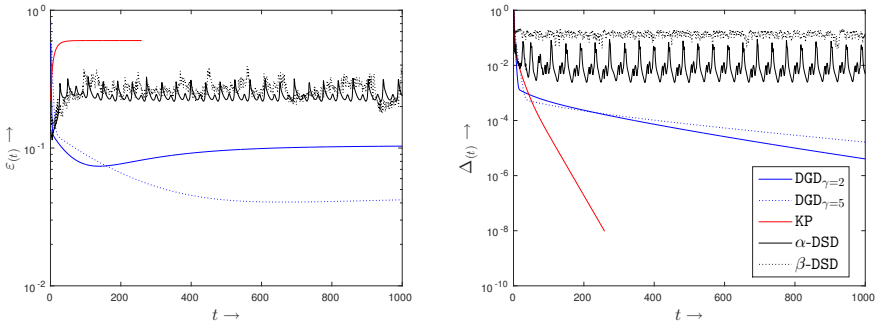
(a) Error convergence $\varepsilon_{(t)}$.

(b) Equilibrium convergence $\Delta_{(t)}$.

Figure 5.8: Convergence results for the four methods with the line graph ($M = 5$), and $A \in \mathbb{R}^{20\times 10}$ of the positive type (i).

$\Delta_{(t)}$ is relatively fast for the `KP` method, however, the equilibrium does not corresponds to the least accurate average approximation of the optimal solution x^*. The convergence in the equilibrium remains very unpredictable for the $\alpha-$`DSD` and β-`DSD` method, and we observe that the convergence in the error is only slightly better than the `KP` method. For the `DGD` method we observe that the convergence results are much smoother, and this method seems to outperform the other. As was to be expected from theorem 4.4, the convergence for $\gamma = 5$ is slower, but the obtained accuracy has improved.

For comparison we present the results obtained with the circle graph in figure 5.9. Overall, the convergence speed has not improved, and naturally the obtained accuracy has not improved as well. For the β-`DSD` we observe that there seems to be some equilibrium trajectory for the states of the agents, by periodicity of the equilibrium convergence behavior. We mention that the results did not improve by considering denser graphs.

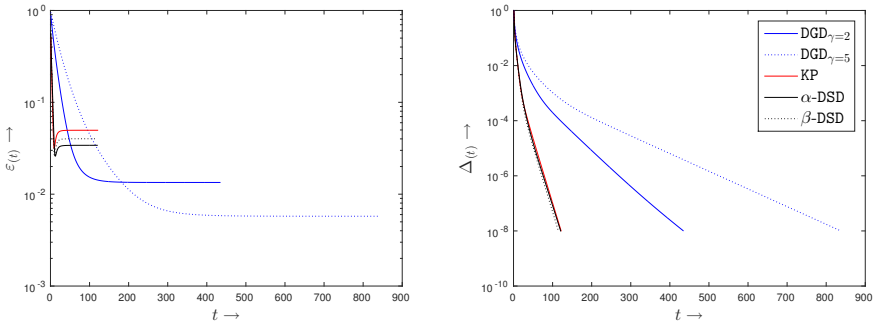


(a) Error convergence $\varepsilon_{(t)}$. **(b)** Equilibrium convergence $\Delta_{(t)}$.

Figure 5.9: Convergence results for the four methods with the circle graph ($M = 5$), and $A \in \mathbb{R}^{20\times 10}$ of the positive type (i).

Other types of matrices

Following up on the previous work, we consider a matrix $A \in \mathbb{R}^{20\times 10}$ of the second type (ii), with condition number $\kappa(A) \approx 13.5$. For comparison, we present the results with the circle graph, see figure 5.10.



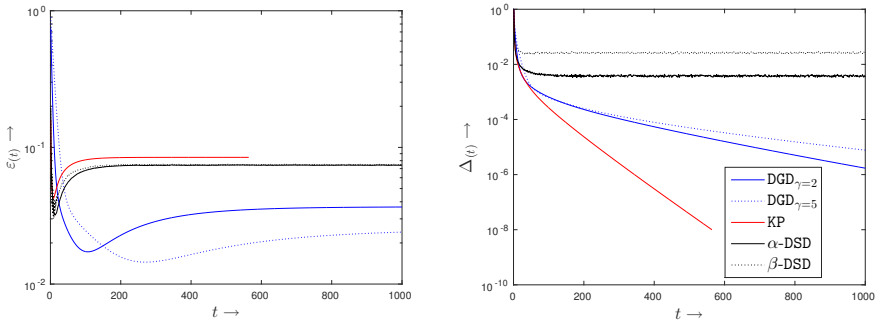
(a) Error convergence $\varepsilon_{(t)}$. **(b)** Equilibrium convergence $\Delta_{(t)}$.

Figure 5.10: Convergence results for the four methods with the circle graph ($M = 5$), and $A \in \mathbb{R}^{20\times 10}$ of type (ii).

Scalability of the methods

In this last part, we consider the scalability of the methods. To this end, we take a randomly computed matrix $A \in \mathbb{R}^{500\times 20}$ of type (i), with condition number $\kappa(A) \approx 15.21$. Also, we let $M = 100$ such that we have $m = 5$. The results are depicted in figure 5.11.

We observe that the qualitative behavior did not change significantly. Also, we obtain approximately the same accuracy as for the smaller sized matrices and networks, which is due to the fact that A is similarly well-conditioned. In contrast, we observed much slower convergence results for matrices with larger condition numbers. We mention that `DGD` still outperforms the other methods.



(a) Error convergence $\varepsilon_{(t)}$.

(b) Equilibrium convergence $\Delta_{(t)}$.

Figure 5.11: Convergence results for the four methods with the circle graph ($M = 5$), and $A \in \mathbb{R}^{500\times 20}$ of the positive type (i).

Concluding remarks

We finish this section with some concluding remarks.

- `DGD` outperforms the other methods, and in concordance with theorem 4.4, there is a trade-off between convergence speed and desired accuracy.
- `KP` has the worst performance, which is in line with our expectations; this method does the largest local minimization step.
- Unfortunately, both variants of `DSD` perform only slightly better than `KP`.

6 Discussion

In this thesis we did a comparative study on distributed methods for solving linear least squares problems. In this context, the DGD method played an important role. Although this method is introduced as a method for general distributed convex optimization, the main convergence results are derived for the strongly convex case [1]. Here it is recognized that the linear least squares problem is such a strongly convex case, and linear convergence is guaranteed when the common step-size α satisfies restriction (4.16). Our contribution to this method is that we derived a reasonable lower bound for this restriction, which is also relatively easy to determine in a truly distributed manner. In this way we defined feasible choices for α as a function of some $\gamma \geq 2$. Corresponding to the choice of the step-size $\alpha(\gamma)$, we proposed to construct a shifted Metropolis-Hasting matrix (4.19), as a function of γ. When only local information about the network is available, this is the minimal shift from the original Metropolis-Hasting matrix that guarantees convergence.

We also considered the KP method, which was designed specifically for consistent linear systems. We observed that this method is not easily generalized for linear least squares problems. For consistent linear systems, the method does not require user-defined constants and converges linearly in the framework of connected networks. Our contribution to this method is that we solved the implementation issues. We observed that the KP update scheme could be rewritten as a two-step method (4.26), and then observed that the local projection problems could be solved efficiently by an application of the CG method, namely by the CGud method for underdetermined systems.

The DGD and KP method are two vastly different methods, yet we established a link between the two by introducing two variants of the DSD method. These two variants are based on two variants of the SDud method, respectively the α-SDud and β-SDud methods. The α-variant is method based on residual norm minimization for underdetermined systems $A_i x = b_i$, whereas the β-variant is based on error norm minimization. For both local minimizers we established linear convergence to the projection of the initial guess, which essentially solves the local projection problems for the KP method as well.

The α-DSD method performs a single step of the α-SDud method for local minimization. This is very similar to the gradient descent steps of the DGD method, but it is also the first step of the CGud method. This shows the link between the two aforementioned methods. Additionally we proposed the β-DSD method, which performs a single step of the β-SDud method for local minimization. This method minimizes, locally, the distance from the solution space in the direction of the negative gradient $-\nabla f_i$. In this sense the single step of the β-SDud method

can be interpreted as an approximation of the local projections of the KP method. From this point of view the β-DSD is more related to the KP method, however, it is much more computationally efficient.

Unfortunately, the convergence analysis for both variants of the DSD method turned out to be too complex. However, due to its similarities with the KP, we were able to show strict decrease in the error norm for the β-DSD method, when applied to consistent linear systems. An analysis of the equilibrium points showed that x^* is the only possible equilibrium point for the consistent case. For linear least squares problems there sometimes exists equilibrium points as well, but these do generally not correspond to a consensus of the optimal solution x^*.

We end this discussion with the observations from the numerical experiments. For consistent linear systems we observed that the KP method outperformed the other methods. However, generally the method performed only slightly better than the more computationally efficient β-DSD method. Since we observed that the KP method generally required the full amount of m iterations for the local CGud step, we conclude that we prefer the β-DSD method in the considered experiments.

With regard to the numerical experiments with linear least squares problems, we mention that KP performed the worst. The α- and β-variants of DSD performed only slightly better, but not significantly. The DGD performed significantly better and we observed a trade-off between speed of convergence and the desired accuracy, which is in concordance with the convergence results for this method. In this sense the DGD is the preferred method. However, we mention that the overall convergence results were not impressive.

We conclude that we have provided a complete overview of the developments in the research area of distributed optimization, but we could unfortunately not provide any significant solutions to the challenges that still exist today. Nonetheless we solved some important implementation issues and considered some interesting and fundamentally different approaches than those described in the recent literature.

A β-variant of the SDud method

We introduce a variant of the SDud method of Section 5.1 and motivate its correctness by making the key observations that are needed to follow the convergence proof of the original SDud method. It follows that this variant has a very similar convergence results as the original method. Consider the SDud method (5.2). Recall that the choice of $\alpha_{(t)}$ is such that it minimizes the residual based function $f(x) = \frac{1}{2}\|b - Ax\|_2^2$ on the line $x_{(t)} + \alpha p_{(t)}$.

Alternatively, consider the minimization of the distance based function $f(x) = \frac{1}{2}\text{dist}(x, \mathcal{X}_i^*)^2 = \frac{1}{2}\|z^* - z\|$ on the line $x_{(t)} + \beta p_{(t)}$, which is indeed a convex function. Here we have $x = z + u$. Observe that we have $z_{(t)} = A^T(AA^T)^{-1}Ax_{(t)}$ and $z^* = A^T(AA^T)^{-1}b$, such that:

$$z^* - z_{(t)} = A^T(AA^T)^{-1}\big(b - Ax_{(t)}\big) = A^T(AA^T)^{-1}r_{(t)}.$$

For clarity we omit the dependency of t and then observe that:

$$\begin{aligned}
\text{dist}(x + \beta p, \mathcal{X}^*)^2 &= \|z^* - z - \beta p\|_2^2 \\
&= \|A^T(AA^T)^{-1}r - \beta p\|_2^2 \\
&= \big(r^T(AA^T)^{-1}A - \beta p^T\big)\big(A^T(AA^T)^{-1}r - \beta p\big) \\
&= r^T(AA^T)^{-1}r - 2\beta r^T r + \beta^2 p^T p
\end{aligned}$$

where we used for the last equality that $p = A^T r$. We then have that:

$$\frac{\partial}{\partial\beta}\text{dist}(x + \beta p, \mathcal{X}^*)^2 = -2r^T r + 2\beta p^T p,$$

and we observe that the exact line search, for the minimization of $f(x) = \frac{1}{2}\text{dist}(x, \mathcal{X}_i^*)^2$, is given by:

$$\beta_{(t)} = \frac{r_{(t)}^T r_{(t)}}{p_{(t)}^T p_{(t)}} = \frac{\|r_{(t)}\|_2^2}{\|p_{(t)}\|_2^2}.$$

For the sake of completeness, we present the update scheme for this variant of SDud, which we will refer to by the β-SDud method for obvious reasons. The

update scheme is as follows (cf. equation (5.2):

$$\begin{aligned} r_{(t)} &= b - Ax_{(t)}, \\ p_{(t)} &= A^T r_{(t)}, \\ \beta_{(t)} &= \frac{r_{(t)}^T r_{(t)}}{p_{(t)}^T p_{(t)}} = \frac{\|r_{(t)}\|_2^2}{\|p_{(t)}\|_2^2}, \\ x_{(t+1)} &= x_{(t)} + \beta_{(t)} p_{(t)}. \end{aligned} \tag{A.1}$$

We will make the key observations that are needed to prove the correctness β-SDud method. It follows that we have a very similar convergence result for this variant, namely that:

$$\|e_{(t)}\|_2^2 \leq \left(\frac{\kappa - 1}{\kappa + 1}\right)^2 \|e_{(t)}\|_2^2, \quad \text{with } \kappa = \frac{\lambda_1}{\lambda_m}.$$

The key observations are as follows.

- With regard to lemma 5.4, we mention that it is straightforward to show that we have:

  $$\|e_{(t+1)}\|_2^2 = \left(1 - \frac{(r_{(t)}^T r_{(t)})^2}{\big(p_{(t)}^T p_{(t)}\big)\big(e_{(t)} e_{(t)}\big)}\right) \|e_{(t)}\|_2^2.$$

  For this we need that the error is updated according to $e_{(t+1)} = e_{(t)} - \beta p_{(t)}$.

- With regard to the Kantorovich inequality for underdetermined systems, lemma 5.5, we first observe that the above can be rewritten as:

  $$\|e'\|_2^2 = \left(1 - \frac{(e^T A^T A e)^2}{\big(e^T (A^T A)^2 e\big)\big(e^T e\big)}\right) \|e\|_2^2,$$

  where we omitted the dependency on t for clarity and defined $e' := e_{(t+1)}$. Then, a very similar proof as the one for lemma 5.5 can be given to show that:

  $$\frac{(v^T A^T A v)^2}{\big(v^T (A^T A)^2 v\big)\big(v^T v\big)} \geq \frac{4\lambda_1 \lambda_m}{(\lambda_1 + \lambda_m)^2}, \tag{A.2}$$

  for any $v \in R(A^T)$. Note that we need the transformation

  $$\eta_i = c_i^2 \lambda_i / \sum_{i=1}^{m} (c_i^2 \lambda_i) \tag{A.3}$$

  in this proof.

With these two observation we can simply follow the arguments in Section 5.1 and we can now state the convergence result in the following corollary.

Corollary A.1 (β-SDud)**.** *Given any $x_{(0)} = z_{(0)} + u_{(0)}$, with $z_{(0)} \in R(A^T)$ and $u_{(0)} \in \ker(A)$, the β-SDud method converges linearly to the projected solution $x_P^* = z^* + u_{(0)} \in \mathcal{X}^*$. Furthermore, we have at every iteration $t+1$ the following inequality:*

$$\|e_{(t+1)}\|_2^2 \leq \left(\frac{\kappa - 1}{\kappa + 1}\right)^2 \|e_{(t)}\|_2^2, \quad \textit{with } \kappa = \frac{\lambda_1}{\lambda_m}. \tag{A.4}$$

Bibliography

[1] K. Yuan, Q. Ling, and W. Yin. On the convergence of decentralized gradient descent. *arXiv preprint arXiv:1310.7063*, 2013.

[2] L. Xiao, S. Boyd, and S. Lall. A scheme for robust distributed sensor fusion based on average consensus. In *IPSN 2005. Fourth International Symposium on Information Processing in Sensor Networks, 2005.*, pages 63–70. IEEE, 2005.

[3] H. Gharavi and S.P. Kumar. Special issue on sensor networks and applications, 2003.

[4] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 263–270. ACM, 1999.

[5] S. Kar, J.M.F. Moura, and K. Ramanan. Distributed parameter estimation in sensor networks: Nonlinear observation models and imperfect communication. *IEEE Transactions on Information Theory*, 58(6):3575–3605, 2012.

[6] U.A. Khan and J.M.F. Moura. Distributed Kalman filters in sensor networks: Bipartite fusion graphs. In *2007 IEEE/SP 14th Workshop on Statistical Signal Processing*, pages 700–704. IEEE, 2007.

[7] G.B. Giannakis, V. Kekatos, N. Gatsis, H. Kim, S.-J.and Zhu, and B.F. Wollenberg. Monitoring and optimization for power grids: A signal processing perspective. *IEEE Signal Processing Magazine*, 30(5):107–128, 2013.

[8] J.A. Bazerque and G.B. Giannakis. Distributed spectrum sensing for cognitive radio networks by exploiting sparsity. *IEEE Transactions on Signal Processing*, 58(3):1847–1862, 2010.

[9] G. Bazerque, J.A.and Mateos and G.B. Giannakis. Group-lasso on splines for spectrum cartography. *IEEE Transactions on Signal Processing*, 59(10):4648–4663, 2011.

[10] A.H. Sayed. *Fundamentals of adaptive filtering*. John Wiley & Sons, 2003.

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

[12] D. Bertsekas, A. Nedić, and A.E. Ozdaglar. Convex analysis and optimization. 2003.

[13] J. Stewart. *Calculus: Early transcendentals 6e*. Cengage Learning, 2015.

[14] D.G. Luenberger and Y. Ye. *Linear and nonlinear programming*, volume 2. Springer, 1984.

[15] G.H. Golub and C.F. Van Loan. *Matrix computations*, volume 3. JHU Press, 1996.

[16] R.B. Bapat. The Laplacian matrix of a graph. *Mathematics Student-India*, 65(1):214–223, 1996.

[17] L. Xiao, S. Boyd, and S. Lall. Distributed average consensus with time-varying Metropolis weights. *Automatica*, 2006.

[18] V. Blondel, J.M. Hendrickx, A. Olshevsky, J. Tsitsiklis, et al. Convergence in multiagent coordination, consensus, and flocking. In *IEEE Conference on Decision and Control*, volume 44, page 2996. IEEE; 1998, 2005.

[19] L. Xiao and S. Boyd. Fast linear iterations for distributed averaging. *Systems & Control Letters*, 53(1):65–78, 2004.

[20] L. Xiao, S. Boyd, and S. Lall. A scheme for robust distributed sensor fusion based on average consensus. In *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pages 63–70. IEEE, 2005.

[21] A. Nedić and A.E. Ozdaglar. Distributed subgradient methods for multi-agent optimization. *Automatic Control, IEEE Transactions on*, 54(1):48–61, 2009.

[22] C.G. Lopes and A.H. Sayed. Diffusion least-mean squares over adaptive networks: Formulation and performance analysis. *Signal Processing, IEEE Transactions on*, 56(7):3122–3136, 2008.

[23] F.S. Cattivelli and A.H. Sayed. Diffusion LMS strategies for distributed estimation. *Signal Processing, IEEE Transactions on*, 58(3):1035–1048, 2010.

[24] S. Mou, J. Liu, and A.S. Morse. A distributed algorithm for solving a linear algebraic equation. *Automatic Control, IEEE Transactions on*, 60(11):2863–2878, 2015.

[25] T. Knutson, A.and Tao. Honeycombs and sums of hermitian matrices. *Notices Amer. Math. Soc*, 48(2), 2001.

[26] S. Mou and A.S. Morse. A fixed-neighbor, distributed algorithm for solving a linear algebraic equation. In *Proc. European Control Conference*, pages 2269–2273, 2013.

[27] G. Mateos, I.D. Schizas, and G.B. Giannakis. Performance analysis of the consensus-based distributed LMS algorithm. *EURASIP Journal on Advances in Signal Processing*, 2009(1):1–19, 2009.

[28] I.D. Schizas, G. Mateos, and G.B. Giannakis. Distributed LMS for consensus-based in-network adaptive processing. *Signal processing, IEEE transactions on*, 57(6):2365–2382, 2009.

[29] K.E. Prikopa, H. Straková, and W.N. Gansterer. Analysis and comparison of truly distributed solvers for linear least squares problems on wireless sensor networks. In *Euro-Par 2014 Parallel Processing*, pages 403–414. Springer, 2014.

[30] H. Straková, W.N. Gansterer, and T. Zemen. Distributed QR factorization based on randomized algorithms. In *Parallel Processing and Applied Mathematics*, pages 235–244. Springer, 2011.

[31] F. Dufossé and B. Uçar. Notes on Birkhoff–von Neumann decomposition of doubly stochastic matrices. *Linear Algebra and its Applications*, 497:108–115, 2016.