Positive Semi-Definite Sparse Low-Rank Matrix Decomposition

Floor Eijkelboom (6432115)

July 2, 2021

Supervisor: Thijs van Ommen

Second Reader: Johannes Korbmacher

BSc Artificial Intelligence

Utrecht University

15 ECT

Acknowledgements

First and foremost, I would like to thank Thijs van Ommen for his guidance during my thesis. Not only his expertise but also his involvement have been incredibly valuable, especially given the peculiar global circumstances in which this thesis has been written.

Furthermore, I would like to thank Johannes Korbmacher for his aid when starting my thesis. His advice on both reading and writing technical materials has made this thesis of much higher quality than it would have been without.

Last, my sincere thanks to Marilva for her support throughout writing my thesis.

Abstract

In recent years, various research fields have been incorporating machine learning techniques to interpret high-dimensional data. Often, most of the information in such datasets is contained in a small subspace of the data. General algorithms that find these subspaces are found but do not make use of prior knowledge of the problem. An example of a situation where we have prior knowledge about both the initial dataset and its relevant subset is found in Structural Equation Modeling, where both sets are represented by positive semi-definite matrices. This study aimed to improve the performance of these general algorithms on positive semi-definite matrices.

We first showed that the mathematical properties on which the general algorithms are based hold in our constrained problem. We then derived both the optimality conditions of our problem and a closed-form algorithm that can find the positive semi-definite decomposition. When comparing the efficiency of our algorithm to its general counterpart, the results showed that our constrained algorithm converges roughly 29% faster. This suggests that our algorithm performs significantly better on the space of positive semi-definite matrices than the general algorithm does.

Contents

1 Introduction

Given the recent developments in Machine Learning, various research fields are using techniques based on analyzing high-dimensional datasets [1]. Employing such datasets is seen in many branches of Artificial Intelligence, such as computer vision [11], face recognition [43] and deep learning [44]. Direct applications have also been found, including in biology [21], sound engineering [20], electromagnetism [35] and geophysics [27]. Such datasets are often hard to interpret, given the vast amount of information that has to be processed. One way to better comprehend the properties of these complex systems is to break them down into smaller parts that are easier to interpret.

Some systems can be represented by a matrix containing exactly the information the system has. When modeling a social network, for instance, we can make a matrix representation of the interconnections among people, by assigning each individual an index and letting the corresponding entry of this matrix represent the strength of their relation. A neat consequence of representing a system with a matrix is that a matrix can be decomposed into a sum or product of other matrices. If we demand that the decomposition matrices obey certain properties, we can use them as indicators for various characteristics of the initial matrix.

One type of matrix decomposition often used in statistics is one where we aim to decompose the matrix into a sparse and low-rank component. This decomposition is used frequently because the important information of high-dimensional datasets is often contained in a low-dimensional subspace of the dataset. This is a consequence of the many dependencies that tend to exist between the observed variables contained in the initial matrix. In this case, most of the data will have an almost identical structure, which we can describe by filtering out the noise of the dataset. The low-rank part of the decomposition then tells us which dependencies there exist. This process is referred to as Sparse Low-Rank Matrix Decomposition (SLRMD).

It turns out, however, that SLRMD is NP-hard in general [38], which makes it infeasible to efficiently find the decomposition that yields the lowest rank and sparsest components. A common way to find a (near) solution to an NP-hard problem is to use an approximation algorithm. Such algorithms have been used to study SLRMD, as seen in [45], [24] and [33], among others. In many statistical models, however, we have prior information about the matrix we aim to deconstruct, which is generally not used in approximating this decomposition. Having more information about the matrix and which matrices can make up the decomposition can greatly narrow our search space.

An example of a situation where we have prior knowledge about the decomposition is in analyzing causal relationships between a set of variables using Linear Structural Equation Models. In a specific case of Linear Structural Equation Modeling, we know that the model itself consists of only positive definite matrices and that the matrices making up the relevant decomposition are positive semi-definite, as we will argue in chapter 2. Given that SLRMD searches over the space of matrices in general, constraining the algorithm to consider only positive semi-definite matrices reduces its search space.[1] Since search space reduction is used as a common way to improve the performance of algorithms [3], (e.g. [36]), we can use this constraint to our advantage.

The primary goal of this thesis is to redefine the approximation algorithms found to estimate SLRMD by preserving the positive semi-definiteness of the matrix to be decomposed. To attain this, we will first aim to improve the algorithms by only preserving the symmetry of matrices when decomposed and then refine this improvement to preserve positive semi-definiteness as well.

We will study the mathematical foundations of SEM in chapter 2, to comprehend which consequences are implied by both constraints. In chapter 3, we will examine if our problem constrained by symmetry satisfies the conditions on which the approximation algorithms for SLRMD are based, by studying the mathematics of convex optimization theory. In chapter 4 and 5, we will consider

[1] Appendix D addresses the validity of this claim.

variations of gradient descent and lagrangian optimization respectively, two first-order classes of approximation algorithms used to approximate SLRMD. To do this, the mathematics on which the two classes rely will be extended in a way it can be applied to our problem. Having derived an algorithm for the symmetric case, we will refine this algorithm to preserve positive semi-definiteness in chapter 6. The performances of the two algorithms will be compared in chapter 7, in which the methodology for the relevant experiments will be laid out.

Causal models are often used as a more interpretable alternative to 'black-box' machine learning methods such as deep learning [41]. Given that the graphs that SEMs are based on grow exponentially in the number of variables measured, this decomposition can help us understand causal relations among a high-dimensional set of variables we would otherwise not be able to analyze. Improving our understanding of such causal models can help us improve the explainability of machine learning models, which is a property sought after extensively in the field of explainable AI [15].

The layout of the individual chapters is illustrative of the fashion in which this thesis came to be. Rather than presenting the mathematics used as a theoretical background separate from our problem, we opted to introduce a given piece of mathematics at the point it becomes relevant to the problem. By doing this, it is always clear *why* the mathematics is introduced beforehand, instead of explaining its relevance afterward. This approach made the piece a little unconventional, perhaps, but all together easier to comprehend.

For the Humanities Honours Program, a supplementary element is included in the thesis. This element consists primarily of a combination of a wide variety of mathematics, namely matrix calculus, optimization theory, theoretical machine learning, abstract algebra, linear algebra, and statistics. I chose to deepen my knowledge about these fields of research, for I wish to continue my academic career by researching geometric deep learning, which combines many of the above fields with insights from topology and differential geometry. Moreover, all the proofs in this work are my own, together accounting for the additional element.

The basic definitions and mathematical preliminaries are summarized in appendices A, B and C. To make the mathematics more comprehensible, matrices are denoted with bold, capital letters (i.e. $\mathbf{M}$), vectors are denoted as bold, lowercase letters (i.e. $\mathbf{v}$), and scalars are denoted as normal, lowercase letters (i.e. a). Random vectors will be denoted using the bold better 'X' as follows:

$$\mathbf{X} = \begin{bmatrix} X_1 \\ \vdots \\ X_n \end{bmatrix}.$$

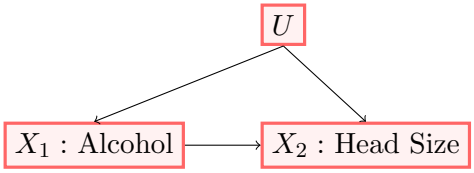
The implementation of the algorithms can be found at https://github.com/FloorEijkelboom/PSD-SLRMD.

2 Structural Equation Modeling

2.1 Introduction

Structural Equation Models (SEMs) are models used to study causal-effect relationships among a collection of variables. To use SEM, a graph is constructed incorporating domain knowledge the researcher has about the variables. The vertices in this graph correspond with the individual variables measured, and the directed edges show which variables can have a causal effect on which other variables. We refer to these variables as the **measured variables** and refer to this graph as the **path diagram**.

Example 2.1. (inpsired by [12]): Fetal alcohol spectrum disorders (FASDs) are a collection of conditions associated with newborns whose mothers consumed alcohol during pregnancy, estimated to affect up to 9.1 of every 1000 baby born in the US and Canada [40] [32]. Let us imagine researchers finding a significant negative correlation between the amount of alcohol consumed by the mother and head size in newborns. A skeptical person might consider the possibility of both the alcohol consumption and birth defect being caused by some confounding socio-economical influence. This situation can be represented by the following graph in SEM:



Here, the exclusion of a directed edge from X_2 to X_1 implies that we are certain X_2 has no causal effect on X_1. △

The major assumption in linear structural equation modeling (LSEM), is that all the causal relationships are linear. That is to say, we assume that the variable X_2 can be understood as a linear combination of X_1 and confounder U:

$$X_2 = \lambda_{02} + \lambda_{12}X_1 + \lambda_{u2}U + \varepsilon_2, \tag{1}$$

where λ_{ij} denotes the effect size of X_i on X_j, λ_{0i} denotes the expected value of X_i and ε_i denotes the the error term of X_i. Our goal in example 2.1 is to find λ_{12}, quantifying the relationship between alcohol consumption and head size.

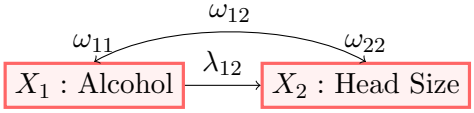
In statistics, SEMs are especially useful when studying (social-scientific) concepts that are not directly observable, such as the confounding variable mentioned in example 2.1. To study such an unobserved variable, its presumed manifestations are observed and used as indicators. We refer to this unobserved construct more generally as a **latent variable**.

Since we cannot directly measure this latent variable, it is not included in our graph explicitely. In example 2.1, exluding U from our graph would imply that

$$X_1 = \lambda_{01} + \tilde{\varepsilon_1} \quad \text{and} \quad X_2 = \lambda_{02} + \lambda_{12}X_1 + \tilde{\varepsilon_2}, \tag{2}$$

where $\tilde{\varepsilon_1} = \lambda_{u1}U + \varepsilon_1$ and $\tilde{\varepsilon_2} = \lambda_{u2}U + \varepsilon_2$. Since both new error terms are affected by the confounder U, we can indirectly observe the latent variable by measuring the covariances between the error terms.

Let ω_{ij} the covariance between $\tilde{\varepsilon}_i$ and $\tilde{\varepsilon}_j$ (and thus, ω_{ii} the variance of $\tilde{\varepsilon}_i$), and consider the following graph:



In SEM, we aim to assign values to the edges in this graph, quantifying the studied relationships.

2.2 Formalizing SEM

Formally, we imagine having measured variables $X_1, ..., X_n$, combined in random vector $\mathbf{X}$. Let $\mathcal{G} = (\mathcal{V}, \mathcal{D}, \mathcal{B})$ be a mixed graph with vertices $\mathcal{V} = \{1, ..., n\}$ such that $i \in \mathcal{V}$ represents X_i, together with directed and bidirected edges $\mathcal{D}, \mathcal{B} \subseteq \mathcal{V} \times \mathcal{V}$. Then the LSEM consists of the joint distribution of $\mathbf{X}$ and a random vector describing noise $\boldsymbol{\varepsilon}$, such that X_i can be understood as

$$X_i = \lambda_{0i} + \sum_{j \in \text{pa}(i)} \lambda_{ji} X_j + \varepsilon_i, \quad i \in \mathcal{V},$$

where $\text{pa}(i)$ denote the set of parents of i, i.e.

$$\text{pa}(i) = \{j : (j, i) \in \mathcal{D}\}.$$

Given that we assumed our causal relationships to be linear, we can represent this set of equation using matrix multiplication. Let $\boldsymbol{\Lambda} = (\lambda_{ij}) \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ denote the matrix of the unknown parameters, such that

$$\mathbf{X} = \boldsymbol{\Lambda}^{\mathbf{T}} \mathbf{X} + \boldsymbol{\varepsilon}.$$

Bringing $\mathbf{X}$ to one side, we see that

$$\mathbf{X} - \boldsymbol{\Lambda}^{\mathbf{T}} \mathbf{X} = (\mathbf{I} - \boldsymbol{\Lambda}^{\mathbf{T}}) \mathbf{X} = \boldsymbol{\varepsilon},$$

and hence this system is solved when

$$\mathbf{X} = (\mathbf{I} - \boldsymbol{\Lambda}^{\mathbf{T}})^{-\mathbf{1}} \boldsymbol{\varepsilon} = (\mathbf{I} - \boldsymbol{\Lambda})^{-\mathbf{T}} \boldsymbol{\varepsilon}, \tag{3}$$

if $\mathbf{I} - \boldsymbol{\Lambda}$ is invertible. The covariance matrix of the solution is given by

$$\mathbf{K_{XX}} = (\mathbf{I} - \boldsymbol{\Lambda})^{-\mathbf{T}} \boldsymbol{\Omega} (\mathbf{I} - \boldsymbol{\Lambda})^{-\mathbf{1}}, \tag{4}$$

where $\boldsymbol{\Omega}$ denotes the covariance of the error tems [12].

The goal in SEM is to find matrices $\boldsymbol{\Omega}$ and $\boldsymbol{\Lambda}$ such that (4) holds for our observed $\mathbf{K_{XX}}$. However, not all matrices $\boldsymbol{\Omega}$ and $\boldsymbol{\Lambda}$ obey the assumptions made in graph $\mathcal{G}$. In example 2.1 we know that the head size of a new born has no effect on alcohol consumption during pregnancy, that is $\lambda_{21} = 0$. In general, we want that if $(i, j) \notin \mathcal{D}$, then $[\boldsymbol{\Lambda}]_{ij} = 0$. Please note that this does not imply that if $(i, j) \in \mathcal{D}$ we assume $\lambda_{ij} \neq 0$, for the directed edge only allows for a causal relationship and does not require one. Hence the set of matrices $\boldsymbol{\Lambda}$ that are considered is

$$\mathbb{R}^{\mathcal{D}} := \{\boldsymbol{\Lambda} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|} \mid \lambda_{ij} = 0 \text{ if } (i, j) \notin \mathcal{D}\}. \tag{5}$$

Moreover, as seen in (3), we want $(\mathbf{I} - \boldsymbol{\Lambda})$ to be invertible. If $\mathcal{G}$ is acyclic, we can always reorder the vertices set $\mathcal{V}$ using the topological ordering defined on $\mathcal{G}$ such that $\mathbf{I} - \boldsymbol{\Lambda}$ is invertible. Hence, the set $\mathbb{R}^{\mathcal{D}}$ contains all possible matrices.

Moreover, we know that $\boldsymbol{\Omega}$ is positive semi-definite, for all covariance matrices are. If a covariance matrix is not strictly positive definite, however, the relationships between the variables are deterministic, which is a property not desired in LSEMs in general. We, therefore, want our matrix $\boldsymbol{\Omega}$ to be positive definite. Similarily as in (5), we want ω_{ij} to be zero if X_i and X_j are not affected by some confounder, that is

$$\text{PD}(\mathcal{B}) = \{\boldsymbol{\Omega} \in \mathbb{S}^n_{++} : \omega_{ij} = 0 \text{ if } i \neq j \text{ and } \{i, j\} \notin \mathcal{B}\}, \tag{6}$$

where $\mathbb{S}^n_{++}$ denotes the set of n-dimensional positive definite matrices.

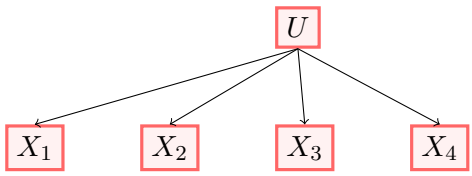
The LSEM given by $\mathcal{G}$ is the set of all random vectors with multivariate normal distributions on $\mathbb{R}^n$, such that its covariance matrix as defined in (4) and its $\mathbf{\Lambda}, \mathbf{\Omega}$ obey (5) and (6), i.e.

$$\mathcal{M}_{\mathcal{G}} = \{(\mathbf{I} - \mathbf{\Lambda})^{-\mathbf{T}} \mathbf{\Omega} (\mathbf{I} - \mathbf{\Lambda})^{-\mathbf{1}} : \mathbf{\Lambda} \in \mathbb{R}^{\mathcal{D}}, \mathbf{\Omega} \in \mathrm{PD}(\mathcal{B})\}. \tag{7}$$

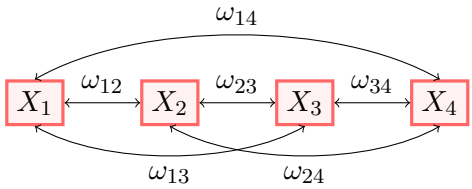
Ideally, we would have one matrix $\mathbf{\Lambda}$ and $\mathbf{\Omega}$ such that this equation holds, hence only one value for λ_{ij} to quantify the effect of X_i on X_j.

2.3 Defining the Problem

In this thesis, a special type of SEM is studied. There are many applications of SEM where we are predominantly interested in studying the latent variable itself [7]. When considering social-scientific concepts we cannot measure directly, we can often imagine a set of associated measurable indicators caused by this concept. E.g., the Epidemiological Studies Depression Scale (CESD) lists talking less and frequent crying as indicators for depression [30]. In general, we can imagine some latent confounding variable U being the cause of a set of aspects:



An interesting consequence of this graph, is that there are no directed edges between the observed variables, and hence $\mathbf{\Lambda} = \mathbf{0}$. This means that our entire model $\mathcal{M}_{\mathcal{G}}$ is defined by only the possible covariance matrices $\mathbf{\Omega}$ of the following graph $\mathcal{G}$:



By the definition of LSEM as seen in (7), this implies that $\mathcal{M}_{\mathcal{G}}$ is itself a set of positive definite matrices. Many positive definite matrices, however, do not correspond with the graph as given above. If a matrix corresponds with this graph, we know it can be written as a matrix of rank 1 plus some diagonal matrix, where the diagonal contains the noise terms independent of the confounder. The formal reason that any such matrix can be decomposed as such goes beyond the scope of this thesis but is a direct consequence of the shape of the graph, implied by 'vanishing tetrads' (see: [37]).

Given that the size of graphs grows exponentially in the number of vertices (i.e. in the number of variables measured), we aim to decompose the observed covariance matrix into a sparse part and a low-rank part. If this can be done in a way congruent to $\mathcal{G}$, we know that our dataset can be explained by one, confounding variable. Moreover, if this cannot be done, we know that our model cannot be explained by one confounding variable.

In general, we have that all positive definite matrices are positive semi-definite matrices, and all positive semi-definite matrices are symmetric matrices, i.e.

$$\mathbb{S}^n_{++} \subset \mathbb{S}^n_{+} \subset \mathbb{S}^n \subset \mathbb{R}^{n \times n}.$$

Given that SLRMD searches over the space of matrices in general, constraining the algorithm to consider only positive semi-definite matrices reduces its search space. In this thesis, we aim to make use of this reduction and redefine the algorithms that exist for sparse low-rank matrix decomposition

to perform better on positive semi-definite matrices. Given that making these sets more restrictive makes them differ more from the space on which the SLRMD algorithms are defined, we aim to first redefine these algorithms on $\mathbb{S}^n$ and then redefine this new algorithm on $\mathbb{S}^n_+$.

In the first case, we imagine a symmetric matrix $\mathbf{C} \in \mathbb{S}^n$ formed by adding two symmetric matrices $\mathbf{A}^*, \mathbf{B}^* \in \mathbb{S}^n$ such that $\mathbf{A}^*$ is sparse and $\mathbf{B}^*$ is low-rank. Our goal is to find $\mathbf{A}^*, \mathbf{B}^*$ given $\mathbf{C}$. We refer to this process as **Symmetric Sparse Low-Rank Matrix Decomposition** (S-SLRMD).

In the second case, we can make use of the fact that $\mathbf{B} = \mathbf{L^T L}$ implies that $\mathbf{B}$ is positive semi-definite, for some matrix $\mathbf{L} \in \mathbb{R}^{l \times m}$ (A.2). Formally, we imagine some matrix $\mathbf{C} \in \mathbb{S}^n_+$ formed by adding two positive semi-definite matrices $\mathbf{A}, \mathbf{B} \in \mathbb{S}^n_+$ such that $\mathbf{A}$ is sparse, and $\mathbf{B} = \mathbf{L^T L}$ for some matrix $\mathbf{L} \in \mathbb{R}^{l \times m}$, where possibly $l = 1$ if $\mathbf{B}$ is rank 1 (hence essentially being a vector). We will refer to this version of SLRMD as **Positive Semi-Definite Sparse Low-Rank Matrix Decomposition** (PSD-SLRMD).

Besides our main objective of deriving an algorithm to perform S-SLRMD and PSD-SLRMD, we will also conduct a series of small-scale experiments evaluating the performance of our algorithms. We hypothesize that the more constrained a problem is to the space of positive semi-definite matrices, the better the algorithms finding positive semi-definite decompositions work. That is to say, we expect our algorithms on PSD-SLRMD to work better on positive semi-definite matrices than those defined for S-SLRMD, and S-SLRMD algorithms better than SLRMD algorithms. Moreover, we will be looking at two types of algorithms to solve these problems, called Gradient Descent and the Alternating Directions Method. Since the latter is generally considered as an improved version of the former [17], we expect the Alternating Directions Method to perform better than gradient descent.

In chapter 3, we will argue that S-SLRMD obeys the essential properties used in the SLRMD approximation algorithms, hence allowing us to extend those algorithms to solve S-SLRMD.

3 Convex Optimization

Mathematical optimization is the study of finding the best element from some set given some criterion. Most often, such an optimization problem seeks to minimize or maximize some function, called the **objective function**. The criterion, called the **constraint**, is often expressed as a set of equalities and inequalities, called the **constraints functions**. This chapter aims to formulate S-SLRMD as an optimization problem that is feasible to solve.

3.1 Formalizing Optimization Problems

Let f the objective function and g_i and h_j the inequality and equality constraint functions of some arbitrary problem respectively. Our optimization problem in its most general form is

$$\begin{aligned} \min_{x \in D} \quad & f(x) \\ \text{subject to} \quad & g_i(x) \leq 0, \qquad i = 1, \dots, m \\ & h_j(x) = 0, \qquad j = 1, \dots, r, \end{aligned}$$

where D is the joint domain of the functions, i.e.

$$D = \text{dom}(f) \cap \bigcap_{i=1}^{m} \text{dom}(g_i) \cap \bigcap_{j=1}^{r} \text{dom}(h_j).$$

If some element $x \in D$ satisfies all the contraints, we call that element **feasible**. An element $x^* \in D$ is a **solution** to this problem if it has the smallest objective value among all feasible elements. The value of f takes on in x^* is denoted by f^*. We call the familly of all feasible elements the **feasible/constraint set**, denoted as $\mathcal{C}$, such that our problem becomes

$$\min_x f(x) \text{ subject to } x \in \mathcal{C}. \tag{8}$$

3.2 Formalizing S-SLRMD

3.2.1 Defining the Constraint Set

Let $\mathbf{C} \in \mathbb{S}^n$ be an arbitrary symmetric matrix we want to decompose. When considering pairs of matrices $(\mathbf{A}, \mathbf{B})$, we ask ourselves which constraints ensure that such a pair is feasible for a given matrix $\mathbf{C}$. First, we know that is must be that $\mathbf{A} + \mathbf{B} = \mathbf{C}$, giving use the equality constraint function evaluating if $\mathbf{A} + \mathbf{B} - \mathbf{C} = \mathbf{0}$. Moreover, we want $\mathbf{A}$ and $\mathbf{B}$ to be symmetric. Given that $\mathbf{C}$ is symmetric, either $\mathbf{A}$ or $\mathbf{B}$ being symmetric ensures the other to be symmetric as well, given the former constraint. We know that a matrix is symmetric if and only if the matrix is equal to its transpose. Without loss of generality, we choose $\mathbf{A}$ to be incorporated in the feasible set to ensure the symmetry, and hence we can add a constraint function evaluating if $\mathbf{A} - \mathbf{A}^\mathbf{T} = \mathbf{0}$ as the second constraint.

We will write the tuple of two matrices as one column vector to emphasize that the two matrices together form a single element of our constraint set, i.e.

$$\begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix} := (\mathbf{A}, \mathbf{B}).$$

Combining the two constraints defined, we can define our constraint set as

$$\mathcal{C}_{\text{S-SLRMD}} = \{ \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix} \mid \mathbf{A} + \mathbf{B} - \mathbf{C} = \mathbf{0}, \mathbf{A} - \mathbf{A}^\mathbf{T} = \mathbf{0} \}. \tag{9}$$

3.2.2 Defining the Objective Function

In order to find the point in our constraint set that is the solution to our problem, a metric of sparsity for $\mathbf{A}$ and low-rankness for $\mathbf{B}$ is needed. In order to estimate the former, the support of a matrix can be used. The **support** of a matrix is the set of all indices where the matrix contains a non-zero value, i.e.

$$\text{support}(\mathbf{M}) = \{(i,j) \mid m_{ij} \neq 0\}.$$

To ensure $\mathbf{M}$ is sparse, we would seek to minimize the cardinality of the support of $\mathbf{A}$. Combining this with the minimization of the rank of $\mathbf{B}$ gives us the following optimization problem:

$$\begin{aligned} \min \quad & \gamma|\text{support}(\mathbf{A})| + \text{rank}(\mathbf{B}) \\ \text{subject to} \quad & \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix} \in \mathcal{C}_{\text{S-SLRMD}} \end{aligned}$$

where $\gamma \in \mathbb{R}$ models the tradeoff between sparsity and rank. This problem, however, is still intractable to solve [10]. A common way to make optimization problems tractable is to make the problem convex [38], as is made use of in the major algorithms to solve SLRMD.

3.3 Convex Sets and Functions

3.3.1 Convex Sets

Let V be a vector space and let $S \subseteq V$. The **closed line segment** between two points $\mathbf{s}, \mathbf{t} \in S$ is the set of all points that lie on the line between $\mathbf{s}$ and $\mathbf{t}$, i.e.

$$\{\theta\mathbf{s} + (1-\theta)\mathbf{t} : 0 \leq \theta \leq 1\}. \tag{10}$$

We refer to the line segment between $\mathbf{s}$ and $\mathbf{t}$ excluding $\mathbf{s}$ and $\mathbf{t}$ as the **open line segment**. We call a set S **convex** if all points on the closed line segments of any two points in S also lie in S.

Proposition 3.1. *All linear (sub)spaces are convex sets.*

Proof. Let $S \subseteq V$ be a subspace of some vector space V and let $\mathbf{u}, \mathbf{v} \in S$. Since S is a subspace, we know that $\theta\mathbf{u} \in S$ and $(1-\theta)\mathbf{v} \in S$ for all $\theta \in \mathbb{R}$ (A.3.c), and hence for $\theta \in [0,1]$. Using (A.3.b) we see that $\theta\mathbf{u} + (1-\theta)\mathbf{v} \in S$, which implies that the line segment is contained in S, proving the proposition. □

A consequence of this proposition is that $\mathbb{S}^n$ is convex, for it forms a vector space (A.6).

3.3.2 Convex Functions

To make use of the advantages of convexity in S-SLRMD, however, we need the objective function itself to be convex as well. A function f is said to be **convex** if for any pair of points on f, the line segment that connects these points lies above f. Formally, this implies that $\text{dom}(f)$ is a convex set and if for all $x_1, x_2 \in \text{dom}(f)$ and $0 \leq \theta \leq 1$ the following inequality holds:

$$f(\theta x_1 + (1-\theta)x_2) \leq \theta f(x_1) + (1-\theta)f(x_2). \tag{11}$$

If this inequality is strict on the open line segment between x_1 and x_2, we call the function **strictly convex**, implying the curvature is more than linear. We call a function f **concave**, if $-f$ is convex. A problem is called **convex** if the objective function, and thus the constraint set, is convex.

Proposition 3.2. *All norms over vector spaces are convex functions.*

Proof. Let W be a vectorspace, let $||\cdot|| : W \to \mathbb{R}$ be a norm. Using proposition 3.1, we know that $\text{dom}(||\cdot||)$ is a convex set. Let $\mathbf{u}, \mathbf{v} \in W$ and $0 \leq \theta \leq 1$. Given that norms obey the triangle inequality (A.5.1.c), we know that

$$\begin{aligned} ||\theta\mathbf{u} + (1-\theta)\mathbf{v}|| &\leq ||\theta\mathbf{u}|| + ||(1-\theta)\mathbf{v}|| \\ &= \theta||\mathbf{u}|| + (1-\theta)||\mathbf{v}||, \qquad \text{(A.5.1.b)} \end{aligned}$$

which is what we wanted to show. □

This is a neat result, for many matrix norms are defined in such a way that they are indicative of certain properties of the matrix [4]. Which norms define the properties desired in S-SLRMD are covered in 3.4.

3.3.3 Minima of Convex Functions

Arguably the most important property of a convex function is that any local minimum is a global minimum. That is to say, if an algorithm finds a local minimum that is feasible, we are certain that it has converged to the optimal value.

Formally, for some normed vector space V with corresponding norm $||\cdot||$ and function $f : V \to \mathbb{R}$ defined on V, we call $\mathbf{v}^* \in V$ a **global minimum** of f if f takes on its smallest value in $\mathbf{v}^*$, i.e. for all feasible $\mathbf{v} \in V$

$$f(\mathbf{v}^*) \leq f(\mathbf{v}).$$

We define the $\boldsymbol{\varepsilon}-$**neighborhood** N around some vector $\mathbf{v} \in V$ as all the family of points in ε distance of $\mathbf{v}$ for some $\varepsilon > 0$, i.e.

$$N = \{\mathbf{v}' \in V : d(\mathbf{v}, \mathbf{v}') < \varepsilon\}.$$

Moreover, we call $\mathbf{x_l} \in V$ a **local minimum** of f if there exists some $\varepsilon > 0$ with an associated neighborhood around $\mathbf{x_l}$ such that $f(\mathbf{x_l}) \leq f(\mathbf{v})$ for all feasible $\mathbf{v} \in N$.

Proposition 3.3. *For any convex problem, all feasible local minima are global minima.*

Proof. Let f be the convex problem associated with the convex problem. Moreover, let $\mathbf{x_l}$ be a feasible local minimum and let N be the neighborhood around $\mathbf{x_l}$ for some $\varepsilon > 0$. Aiming for a contradiction, we assume that there exists some feasible $\mathbf{x_m} \in V$ such that $f(\mathbf{x_m}) < f(\mathbf{x_l})$ outside the neighborhood.

Let $0 \leq \theta \leq 1$. We define

$$\mathbf{n} = \theta\mathbf{x_m} + (1-\theta)\mathbf{x_l},$$

and hence $\mathbf{n} \in V$. Since the problem is convex, we know that the constraint set is convex. Given that $\mathbf{x_m}$ and $\mathbf{x_l}$ are feasible, the convexity of the constraint set implies that the point $\mathbf{n}$ must lie in the contraint set as well, and hence $\mathbf{n}$ is feasible. If we imagine f applied to the points between $\mathbf{x_l}$ and $\mathbf{x_m}$, we know that the convexity of f implies that the function values in those points lie under the line segment connecting $\mathbf{x_l}$ and $\mathbf{x_m}$, i.e. for some $0 < \theta < 1$ we have that

$$\begin{aligned} f(\mathbf{n}) &= f(\theta\mathbf{x_m} + (1-\theta)\mathbf{x_l}) \\ &\leq \theta f(\mathbf{x_m}) + (1-\theta)f(\mathbf{x_l}) \qquad \text{(convexity)} \\ &< f(\mathbf{x_l}). \end{aligned}$$

We note that $\mathbf{n}$ can be rewritten as

$$\mathbf{n} = \mathbf{x_l} + \theta(\mathbf{x_m} - \mathbf{x_l}),$$

and hence by choosing θ arbirarily close to zero, we can ensure that $\mathbf{n}$ lies arbitrarily close to $\mathbf{x_l}$, and thus we can ensure $\mathbf{n} \in N$. However, that would imply there exists some feasible point $\mathbf{n} \in N$ such that $f(\mathbf{n}) < f(\mathbf{x_l})$, which is a contradiction. We conclude that all local minima are global minimal. □

We have now seen that any local minimum of some arbitrary norm defined on a vector space is guaranteed to be a global minimum of that norm in that space. This is a useful result. If we can show that the constraint set of S-SLRMD is convex, we can use the fact that S-SLRMD is defined on a convex space and that we can use norms to characterize properties of matrices, which are also convex, to construct a convex problem.

3.4 Relaxing the Problem

A **relaxation** of some problem is a similar problem that is easier to solve, that resembles the initial problem. The main property the relaxation has to obey is that if some point is optimal for the relaxed problem and feasible for the original problem, it has to be optimal for the original problem as well.

To estimate sparsity, the ℓ_1-norm is used as an relaxation [4]. For some matrix, the ℓ_1 norm is defined by the maximum of the sum of absolute values of the columns of that matrix, i.e.

$$||\mathbf{M}||_1 := \max_j \sum_{i=1}^{m} |m_{ij}|. \tag{12}$$

Moreover, to estimate rank, the nuclear norm is used as a relaxation [4]. This relaxation makes use of the fact that the rank of a matrix is equal to the number of nonzero singular values (with repetitions). The nuclear norm over some matrix is defined by the sum of all its singular values, i.e.

$$||\mathbf{M}||_* := \sum_{i=1}^{\min\{m,n\}} \sigma_i(\mathbf{M}). \tag{13}$$

Using proposition 3.2, we know that $||\cdot||_1$ and $||\cdot||_*$ are convex funtions. We can then use lemma the following lemma to see that the linear combination of the individual norms is still a convex function.

Lemma 3.4. *Convex functions are closed under adition.*

Proof. Let f and g be convex functions. We notice that

$$\begin{aligned}(f+g)(\theta x + (1-\theta)y) &= f(\theta x + (1-\theta)y) + g(\theta x + (1-\theta)y) \\ &\leq \theta f(x) + (1-\theta)f(y) + \theta g(x) + (1-\theta)g(y) \\ &= \theta(f+g)(x) + (1-\theta)(f+g)(y),\end{aligned}$$

which is what we wanted to show. □

Thus, if $\mathcal{C}_{\text{S-SLRMD}}$ is convex, we know that S-SLRMD can be relaxed to the following convex optimization problem:

$$\begin{aligned} \min \quad & \gamma||\mathbf{A}||_1 + ||\mathbf{B}||_* \\ \text{subject to} \quad & \begin{bmatrix}\mathbf{A} \\ \mathbf{B}\end{bmatrix} \in \mathcal{C}_{\text{S-SLRMD}}. \end{aligned} \tag{14}$$

Proposition 3.5. *The constraint set $\mathcal{C}_{S\text{-}SLRMD}$ is convex.*

Proof. See Appendix E. □

In summary, we have shown that all vector spaces are convex, and hence is $\mathbb{S}^n$. We could therefore use the fact that all norms of vector spaces are convex functions, and norms can be indicative of properties of matrices, to find a convex objective function for S-SLRMD. Lastly, we showed that S-SLRMD is a convex set, making the entire problem convex, ensuring all feasible local minima are global minima.

3.5 Characterizations of Convex Functions

In order to know whether some point is a local minimum of a convex function, we use the two most important characterization of a convex functions, called the **first** and **second order characterizations**. The first order characterization tells us that f lies above all the tangent lines in the points on f. That is, if f is differentiable, then f is convex if and only if $\text{dom}(f)$ is convex and for all $x, y \in \text{dom}(f)$, we have that

$$f(y) \geq f(x) + \nabla f(x)^{\mathrm{T}}(y - x). \tag{15}$$

It follows that any differentiable convex function is minimal when its gradient is zero.

The second order characterization tell us that if f is twice differentiable, then f is convex if and only if $\text{dom}(f)$ is convex and for all $x \in \text{dom}(f)$ the Hessian is positive definite, i.e.

$$\nabla^2 f(x) \in \mathbb{S}^n_{++}.$$

In this thesis, first-order methods to solve S-SLRMD will be studies. The foundation of first-order methods comes from gradient descent, which will be studied in the next section.

4 Gradient Descent Methods

4.1 Gradient Descent

4.1.1 The Algorithm

Gradient descent is a first-order method for differentiable functions. **Gradient descent** makes use of the fact that a function decreases fastest in the direction of the negative gradient, giving rise to perhaps the most intuitive way to find the minimum of f: we start at some point x_0 and then 'take a step' of a given size in the direction of the negative gradient until we find the minimum, i.e.

$$x_{k+1} = x_k - t_k \nabla f(x_k).$$

The rate of convergence of gradient descent is dictated by the choice in step size. If we choose t too small, convergence will take too long. If we choose t too big, we might constantly overshoot our minimum value when minimizing, making convergence not possible. Moreover, in standard gradient descent, if we would choose t to be constant, we would have to find the perfect step size for a given problem, which is often not viable. Although many methods that estimate the most efficient step size in each iteration of the algorithm exist [29], they will turn out to not apply to our specific problem.

4.1.2 Subgradients

When considering the gradient of the objective function of S-SLRMD, a problem arises: when graphing any norm, one quickly learns that norms are not differentiable in all points. When considering a point that lies a given distance from the origin, there are many ways to increase the norm of this point optimally (i.e. every point that lies on some circle with a radius smaller than the distance between the origin and the point increase the norm equally), which implies there is no full derivative of any norm. This implies we cannot use regular gradient descent on our function to optimize it.

Since our function is convex, however, we can use a generalization of the gradient - called a subgradient, which gives us exactly enough information about the smoothness of the function to perform gradient descent and comprable methods on [31]. A **subderivative** of a function in some point is the set of all slopes of the tangent lines of the function in that point. We call the set of all slopes of the tangent lines in some point that lie under our curve the **subderivative** of our function in that point, i.e. a is a subgradient of f in p if

$$f(x) \geq f(p) + a(x - p), \tag{16}$$

for all $x \in D$ (c.f. B). The **subdifferential** of f in p would be the set of all such a, denoted by $\partial f(p)$.

Example 4.1. The absolute value is a convex function, which is non-smooth in the origin. The lines through the origin with a slope between -1 and 1 are exactly the tangent lines in the origin, implying that $\partial|0| = [-1, 1]$. △

The **partial subderivative** of a multivariate convex function f in some point $\mathbf{x}$ with resepct to x_i (notation: $\partial_{x_i} f(\mathbf{x})$) is the set of all subderivatives of f in $\mathbf{x}$ with resect to x_i. Similar to (B), the **subgradient** of f in some point $\mathbf{x}$, denoted a $\partial f(\mathbf{x})$, is the set of all vectors in which the ith element is a subderivate of f in $\mathbf{x}$ with respect to x_i, i.e.

$$\partial f(\mathbf{x}) = \{ \begin{pmatrix} g_1 \\ ... \\ g_n \end{pmatrix} \mid g_i \in \partial_{x_i} f(\mathbf{x}) \}.$$

A neat property of subgradients is that the subgradients of a differentiable function is the singleton containing its gradient, that is $\partial f(\mathbf{x}) = \{\nabla f(\mathbf{x})\}$ for all points where f is differentiable. That is to say, subgradients merely extend the definition of derivatives, hence we can speak of the 'gradient' of a non-smooth function. Moreover, our first order condition is preserved, for some point $\mathbf{x}^*$ is a global minimum of f if and only if zero is a subderivative of f in $\mathbf{x}x^*$. Moreover, subgradients can be scaled and added, giving us the following four properties of any subgradient:

(a) $\partial f(\mathbf{x}) = \{\nabla f(\mathbf{x})\}$, if f is differentiable in $\mathbf{x}$

(b) $0 \in \partial f(\mathbf{x}) \iff \mathbf{x}$ is a global minimum of f

(c) $\partial(af) = a\partial f$, if $a > 0$

(d) $\partial(f + g) = \partial f + \partial g$, where on the right hand side we have the Minkowski sum, i.e. for sets A, B we have

$$A + B := \{a + b \mid a \in A, b \in B\}.$$

To emphasize that partial subderivative of a differentiable function is the singleton containing the regular gradient of that function, we write $\partial f(\mathbf{x}) = \nabla f(\mathbf{x})$.

4.1.3 Subgradient Method

Revisiting gradient descent with subgradients gives us the so called **subgradient method**. For some starting point x_0, we use the following iterative scheme: we choose one of the subgradients of f in x_k, and take a step in the direction of the negative subgradient, i.e.

$$g_k \in \partial f(x_k)$$
$$x_{k+1} = x_k - tg_k.$$

One important consequence of using a single subgradient rather than the entire gradient is that the subgradient method is not guaranteed to increase in the direction of the negative subgradient. This is why we keep track of the best iterate, i.e.

$$f_k^{\text{best}} = \min_{i=0,\dots,k} f(x_i).$$

A major result by Shor proofs that if the step length chosen to update the step size is constant and all subgradients have a Frobenius norm equal to one, the subgradient method converges arbitrarily close to the optimum value [34], that is for all $\varepsilon > 0$ we have

$$\lim_{k\to\infty} f_k^{\text{best}} - f^* < \varepsilon.$$

Ensuring a fixed step length of size η is to say that the distance between the next point and current point is equal to η, which is achieved when

$$t_k = \frac{\eta}{||g_k||_{\text{F}}},$$

as shown in [6].

4.1.4 Subgradient Projection Method

In many cases, such as S-SLRMD, we do not want to just minimize some function, but we want to find a minimal element under a set of constraints. To ensure that after each iteration our new point still lies in the constraint set of the problem, we use a projection function. A **projection function** on some constraint set $\mathcal{C}$ (notation: $P_{\mathcal{C}}$) is a function that maps all points $x \in D$ to some

point $x' \in \mathcal{C}$. If $\mathcal{C}$ is a convex set, we know that any point projected on a convex set is closer to every other point in the convex set than the original points were. That is if x gets projected on x', we know that for all points $y \in \mathcal{C}$ we have that

$$||x' - y|| \leq ||x - y||. \tag{17}$$

The **subgradient projection method** makes use of this projection function, by projecting our point x_{k+1} obtained in the regular subgradient method back onto $\mathcal{C}$, giving us the following iterative scheme:

$$\begin{aligned} g_k &\in \partial f(x_k) \\ x_{k+1} &= P_{\mathcal{C}}[x_k - t g_k]. \end{aligned}$$

4.2 Gradient Descent on S-SLRMD

4.2.1 Matrix Calculus

When doing multivariate calculus over spaces of matrices, it is common to combine the partial derivaties of some function over matrices into new matrices. If we imagine some function f applied to matrix $\mathbf{X}$, we essentially consider a multivarite function and hence we can compute its partial subderivatives. We combine these partial derivatives into a new matrix, which we refer to as the **gradient** of f, such that the i, jth entry is the partial derivative with respect to x_{ij}, that is

$$\partial f(\mathbf{X}) = \begin{pmatrix} \frac{\partial f}{\partial x_{11}} & \cdots & \frac{\partial f}{\partial x_{1n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial x_{n1}} & \cdots & \frac{\partial f}{\partial x_{nn}}. \end{pmatrix} \tag{18}$$

We add a subscript to the '∂' to indicate to which matrix the gradient is computed when a function is defined over multiple matrices. If a function is defined over multiple matrices, we denote the gradient as a tuple of matrices, corresponding to the partial derivative to the entries in each matric, i.e.

$$\partial f(\mathbf{X}, \mathbf{Y}) =: \begin{bmatrix} \partial_{\mathbf{X}} f \\ \partial_{\mathbf{Y}} f \end{bmatrix}.$$

An elegant consequence of combining the partial derivatives of a function over matrices in a new matrix, is that many matrix derivates can be computed very straightforwardly, as seen in the following lemma.

Lemma 4.1. *The following matrix derivative rules hold:*

(a) $\partial_{\mathbf{A}} tr(\mathbf{ZA}) = \mathbf{Z^T}$

(b) $\partial_{\mathbf{A}} tr(\mathbf{A^T Z}) = \mathbf{Z}$.

(c) $\partial_{\mathbf{A}} tr(\mathbf{A^T A}) = 2\mathbf{A^T}$.

Proof. See Appendix E. □

4.2.2 Subderivative of S-SLRMD

Using the definition of a subgradient, we see that for some normed vector space V with corresponding norm $||\cdot||$, any matrix $\mathbf{G}$ is a subgradient of that norm in point $\mathbf{A}$ if and only if

$$\langle \mathbf{G}, \mathbf{B} - \mathbf{A} \rangle \leq ||\mathbf{B}|| - ||\mathbf{A}||, \tag{19}$$

where $\langle \cdot, \cdot \rangle$ denotes the standard inner product as defined in (A.6). Given that (19) does not give us a simple way of computing any subgradient, we will derive the subgradient of our objective function directly.

To find the subgradient of our objective function as defined in (14), we can make use of the linearity of the subgradient:

$$\partial(\gamma||\mathbf{A}||_1 + ||\mathbf{B}||_*) = \gamma\partial||\mathbf{A}||_1 + \partial||\mathbf{B}||_*.$$

We know that the ℓ_1-norm considers the maximum of the sum of absolute values for each column. Given that the absolute has a smooth linear increase when it considers values greater than zero, and smooth linear decreases for values smaller than zero, the derivative of the absolute value is defined for all entries unequal to zero, i.e.

$$\frac{d|x|}{dx} = \begin{cases} -1 & x < 0 \\ 1 & x > 0 \end{cases} = \text{sign}(x).$$

Moreover, we know from (4.1) that $\partial_x|0| = [-1, 1]$. Hence any subgradient $\mathbf{S} \in ||\mathbf{A}||_1$ will be of the form

$$\begin{cases} s_{ij} = \text{sign}(a_{ij}) & a_{ij} \neq 0 \\ s_{ij} \in [-1, 1] & a_{ij} = 0 \end{cases}.$$

Since $\mathbf{S} \in \mathbb{S}^n$, the entries s_{ij} are equal to s_{ji} for the uniform choices. Moreover, we make sure to normalize $\mathbf{S}$ when doing any computations to ensure convergence. A simple way to compute that subderivative of the ℓ_1 norm is to imagine $\mathbf{S} = \text{sign}(\mathbf{A}) + \mathbf{Q}$, where $\mathbf{Q}$ and $\mathbf{A}$ have disjoint support (i.e. a_{ij} is zero if and only if g_{ij} is not) and $||\mathbf{Q}||_\infty \leq 1$ [24], where

$$||\mathbf{M}||_\infty := \max_{1 \leq i \leq m} \sum_{j=1}^{n} m_{ij}.$$

In our case, we extend this definition to the symmetric case, giving us

$$\partial_{\mathbf{A}}||\mathbf{A}||_1 = \{\text{sign}(\mathbf{A}) + \mathbf{Q}, \text{ where } \mathbf{Q} = \mathbf{Q}^\mathbf{T}, ||\mathbf{Q}||_\infty \leq 1, \text{supp}(\mathbf{A}) \cap \text{supp}(\mathbf{Q}) = \emptyset\}. \tag{20}$$

It is shown by [39] that the subgradient of the nuclear norm of some matrix $\mathbf{B}$ is

$$\partial||\mathbf{B}||_* = \{\mathbf{U}\mathbf{V}^\mathbf{T} + \mathbf{W} : \sigma_{\max}(\mathbf{W}) \leq 1, \mathbf{U}^\mathbf{T}\mathbf{W} = \mathbf{0}, \mathbf{W}\mathbf{V} = \mathbf{0}\}, \tag{21}$$

where $\mathbf{B} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathbf{T}$ is the SVD of matrix $\mathbf{B}$. If $\mathbf{W} = \mathbf{0}$, it is easy to see that

$$\mathbf{U}\mathbf{V}^\mathbf{T} \in \partial||\mathbf{B}||_*, \tag{22}$$

but other elements of this set are computationally heavy to compute.

Hence, we now that the subgradient of S-SLRMD is given by

$$\partial(||\mathbf{A}||_1 + ||\mathbf{B}||_*) = \{\begin{bmatrix} \mathbf{G_1} \\ \mathbf{G_2} \end{bmatrix} \mid \mathbf{G_1} \in \partial_{\mathbf{A}}||\mathbf{A}||_1, \mathbf{G_2} \in \partial_{\mathbf{B}}||\mathbf{B}||_*\},$$

as defined in (20) and (21).

4.3 Redefining S-SLRMD

Given that the subderivative of the nuclear norm cannot be computed effieciently, we aim to replace it with a simpler expression that has a simpler derivative. To find this expression, we use the rank factorization of matrix $\mathbf{B}$, which makes use of the fact that every matrix $\mathbf{M} \in \mathbb{R}^{m \times n}$ of rank r can be decomposed into $\mathbf{M} = \mathbf{L}\mathbf{R}^{\mathbf{T}}$, such that $\mathbf{L} \in \mathbb{R}^{m \times r}$ and $\mathbf{R} \in \mathbb{R}^{n \times r}$. Using the approach suggested by [24], we can rewrite the nuclear norm, and therefore our problem, as follows:

$$\begin{aligned} \min \quad & \gamma||\mathbf{A}||_1 + \frac{1}{2}(||\mathbf{L}||_{\mathrm{F}}^2 + ||\mathbf{R}^{\mathbf{T}}||_{\mathrm{F}}^2) \\ \text{subject to} \quad & \begin{bmatrix} \mathbf{A} \\ \mathbf{L} \\ \mathbf{R} \end{bmatrix} \in \mathcal{C}'_{\text{S-SLRMD}}, \end{aligned} \tag{23}$$

where $\mathcal{C}'_{\text{S-SLRMD}}$ is the new constraint set, substituting $\mathbf{B} = \mathbf{L}\mathbf{R}^{\mathbf{T}}$ in (9), i.e.

$$\mathcal{C}'_{\text{S-SLRMD}} = \{ \begin{bmatrix} \mathbf{A} \\ \mathbf{L} \\ \mathbf{R} \end{bmatrix} \mid \mathbf{A} + \mathbf{L}\mathbf{R}^{\mathbf{T}} - \mathbf{C} = \mathbf{0}, \mathbf{A} - \mathbf{A}^{\mathbf{T}} = \mathbf{0} \}, \tag{24}$$

and $||\mathbf{M}||_{\mathrm{F}}$ denotes the Frobenius norm (A.6). The argument that $\mathcal{C}'_{\text{S-SLRMD}}$ is convex is identical to the argument that $\mathcal{C}_{\text{S-SLRMD}}$ is. In line with this approach, it holds that if some point $\begin{bmatrix} \mathbf{A} \\ \mathbf{L} \\ \mathbf{R} \end{bmatrix}$ is the optimum of (23) we find $\mathbf{B}$ by $\mathbf{B} = \mathbf{L}\mathbf{R}^{\mathbf{T}}$.

A major advantage that (23) has over our previous definition (14), is that the partial subderivatives with respect to $\mathbf{L}$ and $\mathbf{R}$ are straightforward to compute. Given that $||\mathbf{M}||_{\mathrm{F}}^2 = \text{trace}(\mathbf{M}^{\mathbf{T}}\mathbf{M})$, we can use lemma 4.1.c to see that

$$\partial_{\mathbf{L}}(\gamma||\mathbf{A}||_1 + \frac{1}{2}(||\mathbf{L}||_{\mathrm{F}}^2 + ||\mathbf{R}^{\mathbf{T}}||_{\mathrm{F}}^2)) = \mathbf{L}^{\mathbf{T}}$$

and

$$\partial_{\mathbf{R}}(\gamma||\mathbf{A}||_1 + \frac{1}{2}(||\mathbf{L}||_{\mathrm{F}}^2 + ||\mathbf{R}^{\mathbf{T}}||_{\mathrm{F}}^2)) = \mathbf{R}.$$

We can now compute the gradient S-SLRMD, resolving the issue.

We can thus implement subgradient descent, by updating $\mathbf{A}, \mathbf{L}, \mathbf{R}$ in the directions of their negative gradients. Let for some objective function f, the step size of the subgradient step of the derivative with resepct to matrix $\mathbf{M}$ be denoted as $t_k^{\mathbf{M}}$, i.e. for some $\eta > 0$ we have

$$t_k^{\mathbf{M}} := \frac{\eta}{||g_k^{\mathbf{M}}||_{\mathrm{F}}}, \quad g_k^{\mathbf{M}} \in \partial_{\mathbf{M}} f.$$

To ensure, however, that our new point lies in $\mathcal{C}_{\text{S-SLRMD}}$, we need to find a projection function.

4.3.1 S-SLRMD Projection

To project our new point back onto our constraint set, we consider two methods. First, we note that $\mathcal{C}'_{\text{S-SLRMD}}$ is the intersection of the two constraint functions seperately, i.e.

$$\mathcal{C}'_{\text{S-SLRMD}} = \mathcal{C}_1 \cap \mathcal{C}_2,$$

where $\mathcal{C}_1 = \{ \begin{bmatrix} \mathbf{A} \\ \mathbf{L} \\ \mathbf{R} \end{bmatrix} \mid \mathbf{A} + \mathbf{L}\mathbf{R}^{\mathbf{T}} - \mathbf{C} = \mathbf{0} \}$ and $\mathcal{C}_2 = \{ \begin{bmatrix} \mathbf{A} \\ \mathbf{L} \\ \mathbf{R} \end{bmatrix} \mid \mathbf{A} - \mathbf{A}^{\mathbf{T}} = \mathbf{0} \}$.

An important result by [2] tell us that if we can split a constraint set into multiple sets, that using the average projection of the individual parts is a valid approach to finding the projection of the combined set, i.e.

$$P_{\mathcal{C}'_{\text{SSLRD}}}(x) = \frac{1}{2}(P_{\mathcal{C}_1}(x) + P_{\mathcal{C}_2}(x)).$$

This approach is referred to as the **averaged projections method**.

A second method often used when is it possible to split the constraint set is Dykstra's projection algorithm.[2] **Dykstra's algorithm** presribes to perform the project after one other, that is

$$P_{\mathcal{C}'_{\text{SSLRD}}}(x) = P_{\mathcal{C}_1}(P_{\mathcal{C}_2}(x))),$$

as proposed in [8]. A major advantage of splitting the two constraint sets is that the projections on singular constraints are simpler than on combinations of constraints.

To project the point back onto $\mathcal{C}_2$, we simply have to find the closest symmetric matrix to $\mathbf{A}$. We know that for any matrix $\mathbf{X}$ it holds that $\mathbf{X}+\mathbf{X}^\mathbf{T}$ is symmetric (A.1). Moreover, it has been shown by [13] that $\frac{1}{2}(\mathbf{X}+\mathbf{X}^\mathbf{T})$ is the closest symmetric matrix to $\mathbf{X}$.

We now need to ensure that that two subgradients add up to the matrix we are trying to decompose. We do this by imagining $\mathbf{A}$ and $\mathbf{R}$ fixed, and finding the corresponding $\mathbf{L}$ that ensures that $\mathbf{A} + \mathbf{L}\mathbf{R}^\mathbf{T} - \mathbf{C} = \mathbf{0}$. Since

$$\mathbf{A} + \mathbf{L}\mathbf{R}^\mathbf{T} - \mathbf{C} = \mathbf{0} \iff \mathbf{L}\mathbf{R}^\mathbf{T} = \mathbf{C} - \mathbf{A} \iff \mathbf{L} = (\mathbf{C} - \mathbf{A})\mathbf{R}^{-\mathbf{T}},$$

we find

$$P_{\mathcal{C}_1}(\mathbf{A}, \mathbf{L}, \mathbf{R}) = (\mathbf{A}, (\mathbf{C} - \mathbf{A})\mathbf{R}^{-\mathbf{T}}, \mathbf{R}).$$

This gives us the Dykstra's projection

$$P_{\mathcal{C}_1}(P_{\mathcal{C}_2}(\mathbf{A}, \mathbf{L}, \mathbf{R})) = P_{\mathcal{C}_1}(\frac{\mathbf{A} + \mathbf{A}^\mathbf{T}}{2}, \mathbf{L}, \mathbf{R}) = (\frac{\mathbf{A} + \mathbf{A}^\mathbf{T}}{2}, (\mathbf{C} - \frac{\mathbf{A} + \mathbf{A}^\mathbf{T}}{\mathbf{2}})\mathbf{R}^{-\mathbf{T}}, \mathbf{R}),$$

and average projection

$$\frac{1}{2}(P_{\mathcal{C}_1}(\mathbf{A}, \mathbf{L}, \mathbf{R}) + P_{\mathcal{C}_1}(\mathbf{A}, \mathbf{L}, \mathbf{R})) = (\frac{\mathbf{3A} + \mathbf{A}^\mathbf{T}}{4}, \frac{(\mathbf{C} - \mathbf{A})\mathbf{R}^{-\mathbf{T}} + \mathbf{L}}{2}, \mathbf{R}).$$

We can now implement a projected subgradient method for S-SLRMD. We will use the same initial values as [24], letting $\mathbf{A_0} = \mathbf{0}$ and $\mathbf{L_0} = \mathbf{U}\mathbf{\Sigma}^{\frac{\mathbf{1}}{\mathbf{2}}}, \mathbf{R_0} = \mathbf{V}\mathbf{\Sigma}^{\frac{\mathbf{1}}{\mathbf{2}}}$, where $\mathbf{B} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathbf{T}$ is the singular value decomposition of $\mathbf{B}$.[3] This gives us the following algorithm:

Algorithm 1: Subgradient Descent - S-SLRMD

$\mathbf{A_0} = \mathbf{0}$;
$\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathbf{T} = \mathbf{B}$, $\mathbf{L_0} = \mathbf{U}\mathbf{\Sigma}^{\frac{\mathbf{1}}{\mathbf{2}}}, \mathbf{R_0} = \mathbf{V}\mathbf{\Sigma}^{\frac{\mathbf{1}}{\mathbf{2}}}$;
while *not converged* **do**
  $\mathbf{A_{k+1}} = \mathbf{A_k} - \mathbf{t_k^A}\gamma(\mathbf{A'}),\ \mathbf{A'} \in \partial_\mathbf{A}||\mathbf{A_k}||_1$;
  $\mathbf{L_{k+1}} = \mathbf{L_k} - \mathbf{t_k^L}\mathbf{L_k^T}$;
  $\mathbf{R_{k+1}} = \mathbf{R_k} - \mathbf{t_k^R}\mathbf{R_k}$;
  $(\mathbf{A_{k+1}}, \mathbf{L_{k+1}}, \mathbf{R_{k+1}}) = P_{\mathcal{C}'}(\mathbf{A_k}, \mathbf{L_k}, \mathbf{R_k})$;
end
$\mathbf{A} \leftarrow \mathbf{A_k}$, $\mathbf{B} \leftarrow \mathbf{L_k}\mathbf{R_k^T}$.

[2] Not to be misakten for Dijkstra's algorithm.

[3] Even though this implies that $\mathbf{L}$ and $\mathbf{R}$ are not symmetric, we know that after one iteration our new matrices will be projected on $\mathbb{S}^n$.

Having worked our way through defining gradient descent on S-SLRMD, a reasonable next class of algorithms to consider are the penalty methods. Penalty methods are one of the most used kinds of methods in constraint optimization theory and form the basis for one the most efficient first-order methods known to solve SLRMD: the alternation directions method.

5 Lagrangian Methods

5.1 Penalty Methods

A **penalty method** is an algorithm that characterizes itself by incorporating the constraints of the original problem in its objective function, making it no longer necessary to consider our constraint set explicitly [28]. A quadratic penalty is often used, to make sure that greater deviations from the solution to be punished harder than points that lie near the solution. Moreover, using the square of the constraint as an additional term, makes sure that all deviations increase the objective function.

In general, if we have some problem with some objective function f and equality contraints functions g_i, we can rewrite the problems as follows:

$$\min f(\mathbf{x}) + \xi \sum_{i=1}^{n} (g_i(\mathbf{x}))^2, \tag{25}$$

where ξ indicates the 'importance' of the violation. Given that for some feasible point $\mathbf{x}_f$ we have that $g_i(\mathbf{x}_f) = 0$, the problem reduces to our originial problem if $\mathbf{x}_f$ is feasible, i.e.

$$f(\mathbf{x}_f) + \xi \underbrace{(\sum_{i=1}^{n} g_i(\mathbf{x}_f))^2}_{=0} = f(\mathbf{x}_f). \tag{26}$$

It is important to note that therefore transforming our problem into a penalty problem does not change the solution to the problem.

In the case of S-SLRMD, we have our constraint set as defined in (24), containing only equality constraints. Given that two matrices $\mathbf{M}, \mathbf{N}$ are equal if $\mathbf{M} - \mathbf{N} = \mathbf{0}$, we can quantify the diviation of $\mathbf{M}$ with respect to $\mathbf{N}$ by taking the size of their difference, i.e. $||\mathbf{M} - \mathbf{N}||_{\mathrm{F}}$. Rewriting (25), we can rewrite S-SLRMD as

$$\min \gamma||\mathbf{A}||_1 + \frac{1}{2}(||\mathbf{L}||_{\mathrm{F}}^2 + ||\mathbf{R}^{\mathbf{T}}||_{\mathrm{F}}^2) + \xi(||\mathbf{A} + \mathbf{L}\mathbf{R}^{\mathbf{T}} - \mathbf{C}||_{\mathrm{F}}^2 + ||\mathbf{A} - \mathbf{A}^{\mathbf{T}}||_{\mathrm{F}}^2). \tag{27}$$

Please note that any algorithm minimizing (27) does not need to be projected onto $\mathcal{C}'_{\text{S-SLRMD}}$, for the optimum value found is guaranteed to satisfy the constraints as argued in (26).

A major issue quadratic methods can have when being minimized is that the problem becomes ill-conditioned when the quadratic term gets added. A problem is called **ill-conditioned** if a small change in the input can cause a large change in the output, making it difficult (or impossible) to find a solution [5].

5.2 Lagrangian Relaxation

5.2.1 Lagrangian

The key property we will make use of to improve our algorithm, is that the gradient of the objective function is parallel to the gradient of the constraint function in a point if and only if that point is an optimal point, i.e.

$$\nabla f(\mathbf{x}^*) = \lambda \nabla g(\mathbf{x}^*), \tag{28}$$

where λ is called a **lagrange multiplier**. We define a new function $\mathcal{L}$, called the **Langrangian**, incorporating this fact:

$$\mathcal{L}(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda g(\mathbf{x}). \tag{29}$$

By a similar argument as in (26), we see that the Langrangian operates as a relaxation for the original problem. Moreover, given that the constraints are added linearly, taking the derivative of Lagrangian becomes straightforward.

In general, the Lagrangian of an optimization problem is the weighted linear combination of the objective function and its contraint functions, that is

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i g_i(x), \tag{30}$$

for the Lagrangian Multiplier vector $\boldsymbol{\lambda}$.

Let $\boldsymbol{\Lambda}, \mathbf{K}$ denote our lagrangian multipliers for the first and second constraint respectively and let $\langle \mathbf{M}, \mathbf{N} \rangle$ denote the standard trace product used for square matrices, i.e.

$$\langle \mathbf{M}, \mathbf{N} \rangle = \text{tr}(\mathbf{N}^\mathbf{T}\mathbf{M}).$$

Given the objective function and constraint set as defined in (23) and (24) respectively, we find that the Lagrangian of S-SLRMD is

$$\mathcal{L}(\mathbf{A}, \mathbf{L}, \mathbf{R}, \boldsymbol{\Lambda}, \mathbf{K}) = \gamma||\mathbf{A}||_1 + \frac{1}{2}(||\mathbf{L}||_F^2 + ||\mathbf{R}^\mathbf{T}||_F^2) + \langle \boldsymbol{\Lambda}, \mathbf{A} + \mathbf{L}\mathbf{R}^\mathbf{T} - \mathbf{C} \rangle + \langle \mathbf{K}, \mathbf{A} - \mathbf{A}^\mathbf{T} \rangle. \tag{31}$$

5.2.2 Augmented Lagrangian

An algorithm combining the penalty method and the lagrangian is the **Augmented Lagrangian Method** (ALM), as first discussed by [17]. Adding the quadratic penalty to the Lagrangian makes sure that optimizing the Lagrangian becomes feasible even when we can not compute its derivative, by making the problem strongly convex. Adding a quadratic penalty for the violation of each equality contraint g_i, we define our **Augmented Lagrangian** as follows:

$$\mathcal{L}_\mathcal{A}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \sum_{i=1}^{n} \lambda_i g_i(\mathbf{x}) + \xi \sum_{j=1}^{n} (g_j(\mathbf{x}))^2. \tag{32}$$

Implementing a method using the augmented lagrangian is often a feasible approach to solve a constraint optimization problem, for the augmented Lagrangian is a smooth function, making it straightforward to differentiate and has the advantages as mentioned after (29).

Combining (27) and (31), we can express S-SLRMD as the following augmented lagrangian:

$$\begin{aligned} \mathcal{L}_\mathcal{A}(\mathbf{A}, \mathbf{L}, \mathbf{R}, \boldsymbol{\Lambda}, \mathbf{K}, \xi) = \gamma||\mathbf{A}||_1 + \frac{1}{2}(||\mathbf{L}||_F^2 + ||\mathbf{R}^\mathbf{T}||_F^2) + \langle \boldsymbol{\Lambda}, \mathbf{A} + \mathbf{L}\mathbf{R}^\mathbf{T} - \mathbf{C} \rangle \\ + \langle \mathbf{K}, \mathbf{A} - \mathbf{A}^\mathbf{T} \rangle + \frac{\xi}{2}(||\mathbf{A} + \mathbf{L}\mathbf{R}^\mathbf{T} - \mathbf{C}||_F^2 + ||\mathbf{A} - \mathbf{A}^\mathbf{T}||_F^2). \end{aligned} \tag{33}$$

5.3 Algorithms

5.3.1 Derivative Lagrangian S-SLRMD

First-order methods, as mentioned in chapter 3.5, make use of the gradient. In order to find the subgradient of (31), we need to find the subderivatives of the new terms $\langle \boldsymbol{\Lambda}, \mathbf{A} + \mathbf{L}\mathbf{R}^\mathbf{T} - \mathbf{C} \rangle$ and $\langle \mathbf{K}, \mathbf{A} - \mathbf{A}^\mathbf{T} \rangle$ with respect to $\mathbf{A}, \mathbf{L}$ and $\mathbf{R}$.

Proposition 5.1. $\partial_\mathbf{A} \langle \boldsymbol{\Lambda}, \mathbf{A} + \mathbf{L}\mathbf{R}^\mathbf{T} - \mathbf{C} \rangle = \boldsymbol{\Lambda}$.

Proof. Using the defintion of the inner product, we know that

$$\langle \boldsymbol{\Lambda}, \mathbf{A} + \mathbf{L}\mathbf{R}^\mathbf{T} - \mathbf{C} \rangle = \text{tr}((\mathbf{A} + \mathbf{L}\mathbf{R}^\mathbf{T} - \mathbf{C})^\mathbf{T}\boldsymbol{\Lambda}).$$

Furthermore, using the properties of the trace, we see that

$$\text{tr}((\mathbf{A} + \mathbf{L}\mathbf{R}^\mathbf{T} - \mathbf{C})^\mathbf{T}\boldsymbol{\Lambda} = \text{tr}(\mathbf{A}^\mathbf{T}\boldsymbol{\Lambda}) + \text{tr}(\mathbf{R}\mathbf{L}^\mathbf{T}\boldsymbol{\Lambda}) - \text{tr}(\mathbf{C}^\mathbf{T}\boldsymbol{\Lambda}).$$

Given that only the first term depends on $\mathbf{A}$ we get, using lemma (4.1), that

$$\partial_{\mathbf{A}}\langle \mathbf{\Lambda}, \mathbf{A} + \mathbf{L}\mathbf{R}^{\mathbf{T}} - \mathbf{C}\rangle = \partial_{\mathbf{A}} \text{tr}(\mathbf{A}^{\mathbf{T}}\mathbf{\Lambda}) = \mathbf{\Lambda},$$

which completes the proof. □

By a similar argument, we find

$$\partial_{\mathbf{L}}\langle \mathbf{\Lambda}, \mathbf{A} + \mathbf{L}\mathbf{R}^{\mathbf{T}} - \mathbf{C}\rangle = \mathbf{\Lambda}\mathbf{R} \text{ and } \partial_{\mathbf{R}}\langle \mathbf{\Lambda}, \mathbf{A} + \mathbf{L}\mathbf{R}^{\mathbf{T}} - \mathbf{C}\rangle = \mathbf{\Lambda}^{\mathbf{T}}\mathbf{L}.$$

To find the subgradient of the second term, we again use the properties of the trace to deduce that

$$\text{tr}((\mathbf{A} - \mathbf{A}^{\mathbf{T}})^{\mathbf{T}}\mathbf{K}) = \text{tr}(\mathbf{A}^{\mathbf{T}}\mathbf{K}) - \text{tr}(\mathbf{A}\mathbf{K}),$$

and hence

$$\partial_{\mathbf{A}}\langle \mathbf{K}, \mathbf{A} - \mathbf{A}^{\mathbf{T}}\rangle = \mathbf{K} - \mathbf{K}^{\mathbf{T}}.$$

Hence we find our optmiality conditions for the Lagrangian of S-SLRMD:

$$\partial\mathcal{L}(\mathbf{A}, \mathbf{L}, \mathbf{R}) = \mathbf{0} \iff \begin{cases} \mathbf{0} \in \gamma\partial_{\mathbf{A}}||\mathbf{A}||_1 + \mathbf{\Lambda} + \mathbf{K} - \mathbf{K}^{\mathbf{T}} \\ \mathbf{0} = \mathbf{L}^{\mathbf{T}} + \mathbf{\Lambda}\mathbf{R} \\ \mathbf{0} = \mathbf{R} + \mathbf{\Lambda}^{\mathbf{T}}\mathbf{L} \end{cases} \tag{34}$$

Moreover, since our problem is convex, we know that $\partial\mathcal{L} = \mathbf{0}$ if and only if the algorithm is converged, and hence has found the optimal decomposition.

5.3.2 Derivative Augmented Lagrangian S-SLRMD

To find the subgradient of (33), we need to find subgradients of the two quadratic terms.

Proposition 5.2. $\partial(||\mathbf{A} + \mathbf{L}\mathbf{R}^{\mathbf{T}} - \mathbf{C}||_F^2) = 2\begin{bmatrix} \mathbf{A} + \mathbf{L}\mathbf{R}^{\mathbf{T}} - \mathbf{C} \\ (\mathbf{A} + \mathbf{L}\mathbf{R}^{\mathbf{T}} - \mathbf{C})\mathbf{R} \\ (\mathbf{A} + \mathbf{L}\mathbf{R}^{\mathbf{T}} - \mathbf{C})^{\mathbf{T}}\mathbf{L} \end{bmatrix}.$

Proof. See Appendix E. □

By a similar argument, we find that

$$\partial||\mathbf{A} - \mathbf{A}^{\mathbf{T}}||_F^2 = \mathbf{0},$$

giving us the following optimality condition for the subgradient of the augmented lagrangian of our problem:

$$\partial\mathcal{L}_{\mathcal{A}}(\mathbf{A}, \mathbf{L}, \mathbf{R}) = \mathbf{0} \iff \begin{cases} \mathbf{0} \in \gamma\partial_{\mathbf{A}}||\mathbf{A}||_1 + \mathbf{\Lambda} + \mathbf{K} - \mathbf{K}^{\mathbf{T}} + \xi(\mathbf{A} + \mathbf{L}\mathbf{R}^{\mathbf{T}} - \mathbf{C}) \\ \mathbf{0} = \mathbf{L}^{\mathbf{T}} + \mathbf{\Lambda}\mathbf{R} + \xi(\mathbf{A} + \mathbf{L}\mathbf{R}^{\mathbf{T}} - \mathbf{C})\mathbf{R} \\ \mathbf{0} = \mathbf{R} + \mathbf{\Lambda}^{\mathbf{T}}\mathbf{L} + \xi(\mathbf{A} + \mathbf{L}\mathbf{R}^{\mathbf{T}} - \mathbf{C})^{\mathbf{T}}\mathbf{L} \end{cases} \tag{35}$$

5.4 Alternating Directions Method

An efficient method to solve augmented lagrangian optimization problems given their gradients is the Alternating Directions Method. The **Alternation Directions Method** (ADM) splits the minimization process up into separate parts which are solved separately [14]. In the case of S-SLRMD, the optimization process would be split into three parts, updating $\mathbf{A}, \mathbf{L}$ and $\mathbf{R}$ one after another. To make ADM efficient, however, we need to find closed forms for the updates.

We know that our problem is converged when (35) holds. Using that optimality condition, we can find three closed-form expressions, corresponding to updates to $\mathbf{A}, \mathbf{L}$ and $\mathbf{R}$ that would ensure that the subderivative to the given matrix is zero. ADM decreases $\mathbf{\Lambda}, \mathbf{K}$ proportional to the size of the corresponding constraint violation, to ensure the algorithm converges as quickly as possible.

Proposition 5.3.

$$\partial \mathcal{L}_{\mathcal{A}}(\mathbf{A}, \mathbf{L}, \mathbf{R}) = \mathbf{0} \iff \begin{cases} \mathbf{A} = \mathcal{S}_{\frac{\gamma}{\xi}}(\mathbf{C} - \mathbf{L}\mathbf{R}^{\mathbf{T}} + \frac{\mathbf{\Lambda}+\mathbf{K}-\mathbf{K}^{\mathbf{T}}}{\xi}) \\ \mathbf{L} = (-\mathbf{\Lambda} - \xi(\mathbf{A} - \mathbf{C}))\mathbf{R}(\mathbf{I} + \xi\mathbf{R}^{\mathbf{T}}\mathbf{R}))^{-\mathbf{1}} \\ \mathbf{R} = (-\mathbf{\Lambda} - \xi(\mathbf{A} - \mathbf{C}))^{\mathbf{T}}\mathbf{L}(\mathbf{I} + \xi\mathbf{L}^{\mathbf{T}}\mathbf{L}))^{-\mathbf{1}} \end{cases}, \tag{36}$$

where $\mathcal{S}_\alpha$ denotes the soft thresholding function, as defined in (44).

Proof. See Appendix E. □

For S-SLRMD, we can update now update $\mathbf{A}$, $\mathbf{L}$ and $\mathbf{R}$ one after another until convergence. After each iteration, ADM determines how far the new point lies from the constraint set by updating the size of $\mathbf{\Lambda}$ and $\mathbf{K}$. The 'severity' of this error is dictated by the paramter δ, which acts as a multiplier for the violation. Following a similar argument as in subgradient descent, our initial values are again given by the SVD of $\mathbf{B}$. Coming these results gives us the following algorithm S-SLRMD, which we will refer to as ADM-S:

Algorithm 2: ADM-S

$\mathbf{A_0} = \mathbf{0}$, $\mathbf{\Lambda_0} = \mathbf{0}$, $\mathbf{K_0} = \mathbf{0}$;
$\mathbf{U\Sigma V^T} = \mathbf{B}$, $\mathbf{L_0} = \mathbf{U\Sigma^{\frac{1}{2}}}, \mathbf{R_0} = \mathbf{V\Sigma^{\frac{1}{2}}}$;
while *not converged* **do**
  $\mathbf{A_{k+1}} = \mathcal{S}_{\frac{\gamma}{\xi}}(\mathbf{C} - \mathbf{L_k}\mathbf{R_k^T} + \frac{\mathbf{\Lambda_k}+\mathbf{K_k}-\mathbf{K_k^T}}{\xi})$;
  $\mathbf{L_{k+1}} = (-\mathbf{\Lambda_k} - \xi(\mathbf{A_{k+1}} - \mathbf{C}))\mathbf{R_k}(\mathbf{I} + \xi\mathbf{R_k^T}\mathbf{R_k}))^{-\mathbf{1}}$;
  $\mathbf{R_{k+1}} = (-\mathbf{\Lambda_k} - \xi(\mathbf{A_{k+1}} - \mathbf{C}))^{\mathbf{T}}\mathbf{L_k}(\mathbf{I} + \xi\mathbf{L_k^T}\mathbf{L_k}))^{-\mathbf{1}}$;
  $\mathbf{\Lambda_{k+1}} = \mathbf{\Lambda_k} - \delta(\mathbf{A_{k+1}} + \mathbf{L_{k+1}}\mathbf{R_{k+1}^T} - \mathbf{C})$;
  $\mathbf{K_{k+1}} = \mathbf{K_k} - \delta(\mathbf{A_{k+1}} - \mathbf{A_{k+1}^T})$;
end
$\mathbf{A} \leftarrow \mathbf{A_{k+1}}$, $\mathbf{B} \leftarrow \mathbf{L_{k+1}}\mathbf{R_{k+1}^T}$.

We have successfully found an algorithm to find a symmetric sparse low-rank decomposition of a given symmetric matrix. Its performance will be evaluated in chapter 7.

The next chapter will consider how we can use the results found for S-SLRMD in the case of PSD-SLRMD.

6 PSD-SLRMD

Having defined ADM for S-SLRMD, we aim to do the same for PSD-SLRMD. To do this, we will follow the same approach as done in S-SLRMD, i.e. proving that PSD-SLRMD can be relaxed to a convex problem, finding the optimality conditions, and associated closed-form updates.

6.1 Convexity of PSD-SLRMD

When relaxing our objective function, we will find that it is equivalent to the objective function of S-SLRMD, for we still aim to decompose our given matrix into some matrix $\mathbf{A}$ which is sparse, and $\mathbf{B}$ which is low-rank. The main difference is that in PSD-SLRMD, we imagine a positive semi-definite matrix $\mathbf{C}$ decomposed into positive semi-definite matrices matrix $\mathbf{A}$ and $\mathbf{B}$, rather than just symmetric matrices. We will first prove that our reduced search space is also convex.

Proposition 6.1. *The set of positive semi-definite matrices is convex.*

Proof. Let $\mathbf{M}, \mathbf{N} \in \mathbb{S}_+^n$. We know $\mathbf{S}$ is positive semi-definite if and only if for all $\mathbf{x} \in \mathbb{R}^n$ we have that $\mathbf{x^T S x} \geq 0$. We notice that for some $0 \leq \theta \leq 1$ we have that

$$\begin{aligned}\mathbf{x^T}(\theta\mathbf{M} + (1-\theta)\mathbf{N})\mathbf{x} &= \theta\mathbf{x^T M x} + (1-\theta)\mathbf{x^T N x} \\ &\geq 0,\end{aligned}$$

implying that $\mathbb{S}_+^n$ is convex. □

This result helps us in two ways. First, combining the facts that the objective function of S-SLRMD is convex (14), and the objective function of PSD-SLRMD is the same fuction as in S-SLRMD but over a restricted domain, which is convex as well, we know that the objective function of PSD-SLRMD is convex. Moreover, we can quite straightforwardly prove that the corresponding constraint set of PSD-SLRMD is convex, as well. To prove this, we will use the following lemma:

Lemma 6.2. *The set of positive semi-definite matrices is closed under addition.*

Proof. Let $\mathbf{M}, \mathbf{N} \in \mathbb{S}_+^n$. By definition, we know that $\mathbf{x^T M x} \geq 0$ and $\mathbf{x^T N x} \geq 0$ for all $x \in \mathbb{R}^n$. Hence we see that for all $x \in \mathbb{R}^n$, we have that

$$\begin{aligned}\mathbf{x^T}(\mathbf{M} + \mathbf{N})\mathbf{x} &= \mathbf{x^T M x} + \mathbf{x^T N x} \\ &\geq 0,\end{aligned}$$

which proves our lemma. □

Proposition 6.3. *The constraint set $\mathcal{C}_{PSD\text{-}SLRMD}$ is convex.*

Proof. Similar to before, we know that the constraint set of PSD-SLRMD is given by

$$\mathcal{C}_{\text{PSD-SLRMD}} = \{ \begin{bmatrix}\mathbf{A} \\ \mathbf{B}\end{bmatrix} \mid \mathbf{A} + \mathbf{B} - \mathbf{C} = \mathbf{0} \text{ and } \mathbf{A} \in \mathbb{S}_+^n \}.$$

By lemma 6.2, we to that $\mathbf{A}$ and $\mathbf{C}$ being positive semi-definite ensures $\mathbf{B}$ to be positive semi-definite as well. Using lemma E.1, we know that the the intersection of the sets of the individually constraints is itself convex. Moreover, we know by proposition 3.5 that the set corresponding with the first constraint set is convex, and by proposition 6.1 that the set corresponding with the second constraint set is convex as well, finishing our proof. □

Given that both the objective function and the constraint set of PSD-SLRMD are convex, we have successfully rewritten PSD-SLRMD as a convex optimization problem.

6.2 Algorithms for PSD-SLRMD

6.2.1 Subgradient Descent

Given that our objective function has not changed, our subgradient method does not change either, besides the projection used. Using the average projection method and Dykstra's method, we aim to find a projection of a matrix to the closed positive semi-definite matrix. Contrary to the symmetric case, this is computationally complex to do [19]. Even though an algorithm has been found by Higham [18], it is iterative and not certain to converge. Given the projection method itself is a major deciding factor in the performance of project subgradient descent [25], we conclude that subgradient descent cannot feasibly be used to do convex optimization on positive semi-definite matrices. This issue accentuates the strength of penalty methods like ADM, for their this problem is avoided all altogether.

6.2.2 Alternating Directions Method

Rewriting our algorithm for the case where we want $\mathbf{B}$ to be positive semi-definite and rank 1, we are looking for some matrix $\mathbf{L}$ such that $\mathbf{B} = \mathbf{L}^\mathbf{T}\mathbf{L}$ and $\mathbf{B}$ has rank 1. Since we have expressed $\mathbf{B} = \mathbf{L}\mathbf{R}^\mathbf{T}$ in S-SLRMD, we can add the constraint that $\mathbf{L} = \mathbf{R}$ to ensure this. Furthermore, we can drop the symmetric constraint, giving us the following augmented Lagrangian:

$$\begin{aligned}\mathcal{L}_\mathcal{A}(\mathbf{A}, \mathbf{B}, \mathbf{\Lambda}, \mathbf{M}, \xi) = \gamma||\mathbf{A}||_1 + \frac{1}{2}(||\mathbf{L}||_F^2 + ||\mathbf{R}^\mathbf{T}||_F^2) + \langle \mathbf{\Lambda}, \mathbf{A} + \mathbf{L}\mathbf{R}^\mathbf{T} - \mathbf{C}\rangle \\ + \langle \mathbf{M}, \mathbf{L} - \mathbf{R}\rangle + \frac{\xi}{2}(||\mathbf{A} + \mathbf{L}\mathbf{R}^\mathbf{T} - \mathbf{C}||_F^2 + ||\mathbf{L} - \mathbf{R}||_F^2).\end{aligned}$$

By a similar argument as made in prositions 5.1 and 5.2, we see that

$$\partial(\langle \mathbf{M}, \mathbf{L} - \mathbf{R}\rangle) = \begin{bmatrix}\mathbf{0} \\ \mathbf{M} \\ -\mathbf{M}\end{bmatrix} \quad \text{and} \quad \partial(||\mathbf{L} - \mathbf{R}||_\mathrm{F}^2) = 2\begin{bmatrix}\mathbf{0} \\ \mathbf{L} - \mathbf{R} \\ \mathbf{R} - \mathbf{L}\end{bmatrix}.$$

If we use the derivates for the parts we already found when differentiating the augmented lagrangian of S-SLRMD, we see that the optimality conditions of PSD-SLRMD become the following:

$$\partial\mathcal{L}_\mathcal{A}(\mathbf{A}, \mathbf{L}, \mathbf{R}) = \mathbf{0} \iff \begin{cases}\mathbf{0} \in \gamma\partial_\mathbf{A}||\mathbf{A}||_1 + \mathbf{\Lambda} + \xi(\mathbf{A} + \mathbf{L}\mathbf{R}^\mathbf{T} - \mathbf{C}) \\ \mathbf{0} = \mathbf{L}^\mathbf{T} + \mathbf{\Lambda}\mathbf{R} + \mathbf{M} + \xi[(\mathbf{A} + \mathbf{L}\mathbf{R}^\mathbf{T} - \mathbf{C})\mathbf{R} + \mathbf{L} - \mathbf{R}] \\ \mathbf{0} = \mathbf{R} + \mathbf{\Lambda}^\mathbf{T}\mathbf{L} - \mathbf{M} + \xi[(\mathbf{A} + \mathbf{L}\mathbf{R}^\mathbf{T} - \mathbf{C})^\mathbf{T}\mathbf{L} + \mathbf{R} - \mathbf{L}]\end{cases} \tag{37}$$

Similary as before, we can find closed form solution for our ADM algorithm:

Proposition 6.4.

$$\partial\mathcal{L}_\mathcal{A}(\mathbf{A}, \mathbf{L}, \mathbf{R}) = \mathbf{0} \iff \begin{cases}\mathbf{A} = \mathcal{S}_\frac{\gamma}{\xi}(\mathbf{C} - \mathbf{L}\mathbf{R}^\mathbf{T} + \frac{\mathbf{\Lambda}}{\xi}) \\ \mathbf{L} = [-\mathbf{M} - (\mathbf{\Lambda} + \xi(\mathbf{A} - \mathbf{C} + \mathbf{I}))\mathbf{R}](\mathbf{I} + \xi(\mathbf{R}^\mathbf{T}\mathbf{R} - \mathbf{I}))^{-\mathbf{1}} \\ \mathbf{R} = [\mathbf{M} - (\mathbf{\Lambda}^\mathbf{T} + \xi(\mathbf{A} - \mathbf{C} - \mathbf{I})^\mathbf{T})\mathbf{L}](\mathbf{I} + \xi(\mathbf{L}^\mathbf{T}\mathbf{L} + \mathbf{I}))^{-\mathbf{1}}\end{cases}. \tag{38}$$

Proof. See Appendix E. □

Having found these closed form updates, we can again define a ADM algorithm for PSD-SLRMD, which we will refer to as ADM-PSD:

Algorithm 3: ADM-PSD

$\mathbf{A_0} = \mathbf{0}$, $\mathbf{\Lambda_0} = \mathbf{0}$, $\mathbf{M_0} = \mathbf{0}$;
$\mathbf{U\Sigma V^T} = \mathbf{B}$, $\mathbf{L_0} = \mathbf{U\Sigma^{\frac{1}{2}}}, \mathbf{R_0} = \mathbf{V\Sigma^{\frac{1}{2}}}$;
while *not converged* **do**
  $\mathbf{A_{k+1}} = \mathcal{S}_{\frac{\gamma}{\xi}}(\mathbf{C} - \mathbf{L_k R_k^T} + \frac{\mathbf{\Lambda_k}}{\xi})$;
  $\mathbf{L_{k+1}} = [-\mathbf{M_k} - (\mathbf{\Lambda_k} + \xi(\mathbf{A_{k+1}} - \mathbf{C} + \mathbf{I}))\mathbf{R_k}](\mathbf{I} + \xi(\mathbf{R_k^T R_k} - \mathbf{I}))^{-1}$;
  $\mathbf{R_{k+1}} = [\mathbf{M_k} - (\mathbf{\Lambda_k^T} + \xi(\mathbf{A_{k+1}} - \mathbf{C} - \mathbf{I})^{\mathbf{T}})\mathbf{L_{k+1}}](\mathbf{I} + \xi(\mathbf{L_{k+1}^T L_{k+1}} + \mathbf{I}))^{-1}$;
  $\mathbf{\Lambda_{k+1}} = \mathbf{\Lambda_k} - \delta(\mathbf{A_{k+1}} + \mathbf{L_{k+1} R_{k+1}^T} - \mathbf{C})$;
  $\mathbf{M_{k+1}} = \mathbf{M_k} - \delta(\mathbf{L_{k+1}} - \mathbf{R_{k+1}})$;
end
$\mathbf{A} \leftarrow \mathbf{A_{k+1}}$, $\mathbf{B} \leftarrow \mathbf{L_{k+1} R_{k+1}^T}$.

We have now successfully found an algorithm for PSD-SLRMD. We will evaluate the performance of both the subgradient methods and lagrangian methods in chapter 7.

7 Experiment

This section covers the experiments done to evaluate our algorithms. It serves as a stand-alone section analyzing the two experimental hypotheses defined in chapter 2.3, i.e. asking if enforcing more constraints implies better performance and if ADM performs better than subgradient descent.

The implementation of the algorithms can be found in `models.py` and the implementation of the data generators can be found in `matgen.py`. Moreover, the algorithms can be executed from `main.py` (instructions are included in the file).

7.1 Evaluating the Algortihms

7.1.1 Generating the Data

To evaluate any algorithm, a set of decompositions to compare the results to is needed. We immediately run into a problem, for we cannot generate such a decomposition for some matrix, for that is exactly the problem we are trying to solve. Alternatively, we can generate a sparse matrix and a low-rank matrix and add them together. If the initial matrices were sparse and dependant enough respectively, we can safely assume that they form the decomposition.

To generate a symmetric sparse matrix of sparsity p, we first generate a standard normally distributed matrix $\mathbf{G}$, and obtain a symmetric standard normally distributed matrix by adding $\mathbf{G}$ to its transpose. We then cycle randomly through the entries, removing them and their symmetric counterparts until the sparsity is greater than or equal to p, giving us the algorithm `sparse_matrix`.

To generate a symmetric matrix such that its rank is smaller than some predetermined rank, we can make use of the fact that the rank of a product of matrices is smaller than the minimum of the individual ranks, i.e.

$$\text{rank}(\mathbf{AB}) \leq \min(\text{rank}(\mathbf{A}), \text{rank}(\mathbf{B})).$$

We also know that for some matrix $\mathbf{L} \in \mathbb{R}^{n \times k}$, the product $\mathbf{LL^T}$ is symmetric (A.1), and hence

$$\text{rank}(\mathbf{LL^T}) \leq \min(\text{rank}(\mathbf{L}), \text{rank}(\mathbf{L^T})) \leq \text{rank}(\mathbf{L}) \leq k,$$

given that $\text{rank}(\mathbf{L}) = \text{rank}(\mathbf{L^T})$ (A.4). We can therefore generate a symmetric matrix with a maximum rank of k, by generating an arbitrary matrix of dimensions n by k, and multiplying it by its transpose, giving us the algorithm `lowrank_matrix`.

To generate the low-rank part of the positive semi-definite decompositions, we simply generate a column vector which is then multiplied with its transpose, as described in `lowrank_matrix_PSD`. To generate a sparse semi-definite matrix, we generate a sparse symmetric matrix and square it (i.e. multiply it with its transpose), as seen in `sparse_matrix_PSD`.

7.1.2 Sparse and Low-Rank Criterions

If we consider an algorithm estimating some sparse matrix, it could be that our approximation might look something like

$$\begin{pmatrix} 1.001 & 0.001 & ... & 0.001 \\ 0.001 & 1.001 & ... & 0.001 \\ \vdots & \vdots & \ddots & 0.001 \\ 0.001 & 0.001 & ... & 1.001 \end{pmatrix}.$$

Strictly, we have no zero entries, hence the matrix is not sparse. However, given that many algorithms only approach the right value and might not reach it, it becomes necessary to implement

a criterion that signifies whether the entry is 'sparse enough', which we will refer to as the **sparse tolerance**.

Similarly, when considering a matrix-like

$$\begin{pmatrix} 50.001 & 50.002 & 50.0015 \\ 20.002 & 20.011 & 22.0013 \\ 40.004 & 40.005 & 40.002 \end{pmatrix},$$

it is full rank, even though it approaches a completely dependent system. The fact that this matrix approaches rank 1 is seen in the singular values, where all but one singular value will approach zero. We introduce a similar criterion that signifies when a singular value is small enough, which we will refer to as the **low-rank tolerance**.

Both criteria are not modeled explicitly, for NumPy (the linear algebra library used to implement the algorithms) has this functionality build in.

7.1.3 Overview of Experiments

The experiments done are of two types. Experiments 1 through 3 are meant to estimate the different effects of the parameters in our models. We refer to these experiments as parameter estimation experiments (PEEs). For subgradient descent, we estimate a single parameter in PEE.1, and for both ADMs we estimate two parameters, done in PEE.2 and PEE.3.

We use the results of the PEEs in the four algorithmic performance experiments (APEs), APE.1 through APE.4, which are described in chapter 7.3. The experiments can be found and executed from `main.py`, under the names `PEE_1` through `PEE_3` and `APE_1` through `APE_4`.

7.2 Parameter Estimation Experiments

7.2.1 Method

In the PEE.1 and PEE.2 experiments, we generate matrices of dimension 50, sparsity $p = 0.15$ and rank 5. Moreover, we chose to fix $\gamma = 0.05$, which performs well, as argued by [45]. We choose a relatively small sparsity and rank, to increase the odds that when the algorithm finds a solution of such sparsity and rank, this solution is the optimal decomposition of the generated matrix, as argued in chapter 7.1.1. For PEE.3 experiment, we generated positive semi-definite matrices of rank 1 and sparsity $p = 0.15$. Given that the algorithms are made to perform well on a given domain, we estimate the parameters on that domain as well.

For PEE.1, we first notice that subgradient descent is dependent on the stepsize parameter and hence we need to first find which stepsize yields the best results for our problem. Given that subgradient descent converges rather slowly in general, we estimate the performance of different choices in stepsize by letting the algorithm run for a set number of iterations (rather than until convergence) and considering its best objective value to that point. Given that the matrices are distributed equally, the expected objective value of a given matrix at the start is equal, which implies that when averaging over multiple decompositions we can compare the performances between the different stepsizes. We let the algorithm run for 1000 iterations per decomposition, and we averaged over 25 decompositions per stepsize.

Contrary to the previous method, our ADM algorithms converge. For ADM, we need to estimate both that ξ and δ parameters for both versions of ADM. To evaluate which parameters perform best, we recorded the average number of iterations until convergence for different values of ξ and δ.

7.2.2 Results

The PEE.1 showed that for our subgradient descent algorihm the stepsize that performed best was $\eta \approx 4.2$. The results are summarized in figure 1.

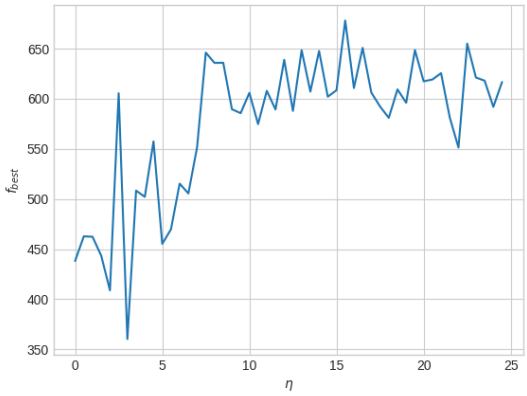
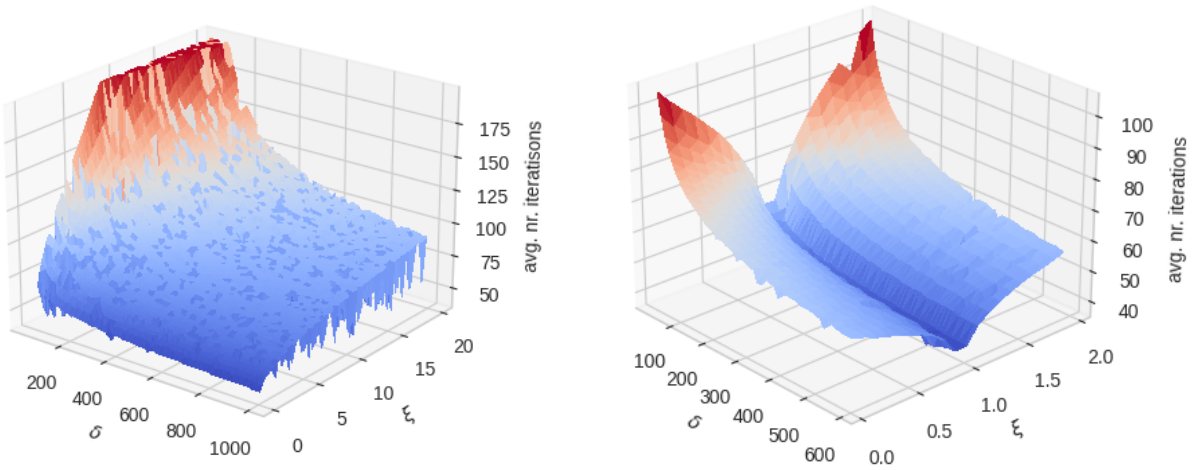


Figure 1: PEE.1 (SGD): Effect of stepsize on average best iteration in SGD.

The PEE.2 and PEE.3 showed that for ADM-S, we have that a $\xi \approx 0.2$ yields the optimal results together with a large enough $\delta \geq 500$. For ADM-PSD, we see that a $\xi \approx 1.08$ yields the optimal results together with a large enough $\delta \geq 800$. The effects of the individual parameters ξ and δ on both algorithms can be seen in appendix F. Their combined effect is seen in figure 2.



(a) PEE.2 (ADM-S)

(b) PEE.3 (ADM-PSD)

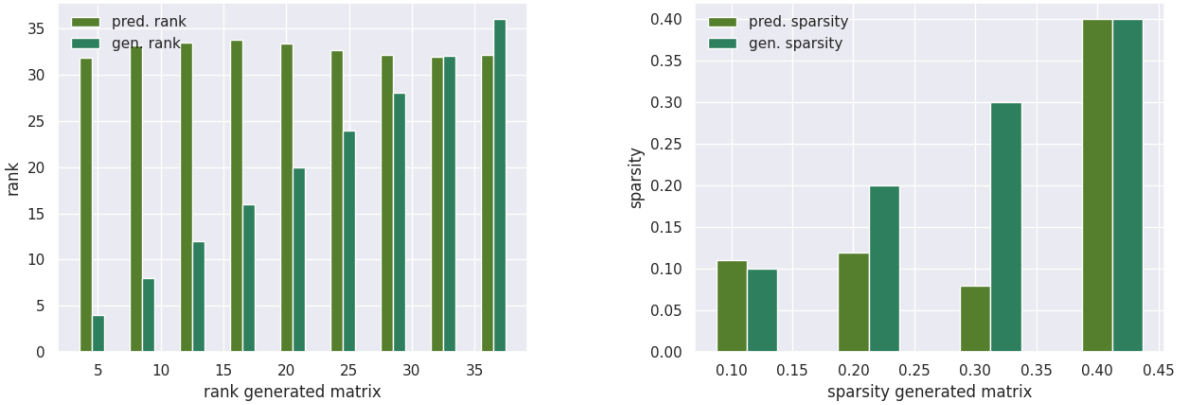
Figure 2: Effect of choice of δ and ξ on average number of iterations of Alternating Directions Method on S-SLRMD (left) and PSD-SLRMD (right).

7.3 Algorithmic Performance Experiments

7.3.1 Subgradient Descent (APE.1 and APE.2)

Given that our subgradient method did not converge consistently, our hypothesis that ADM performs is answered directly. However, this gives rise to the question of whether the lack of convergence in subgradient descent is caused by minimizing sparsity or by minimizing rank (or both). To answer this question, we conduct two APE experiments (APE.1 and APE.2), in which the ranks

and sparsities of generated matrices are compared to those predicted by a model with a stepsize of $\eta = 4.2$. In the APE.1 and APE.2 experiments, we again generate symmetric matrices of dimension 50, of different rank and sparsity. To estimate the effect of rank, we keep the sparsity fixed at $p = 0.15$, and to estimate sparsity we keep the rank fixed at 5. Moreover, we still fix $\gamma = 0.05$. The results are summarized in figure 3.



(a) (b) APE.2

Figure 3: Average best performance of SGD for matrices generated of differend ranks (left) and sparsities (right).

7.3.2 Alternating Directions Method

In the APE.3 and APE.4 algorithm, we evaluate if our AMD-S performs better than regular ADM on the positive semi-definite matrices, and ADM-PSD performs better than ADM-S. We, therefore, generate different positive semi-definite matrices and compare the average number of iterations until convergence for each of the three. These results are visualized in the figure 4a and 4b.

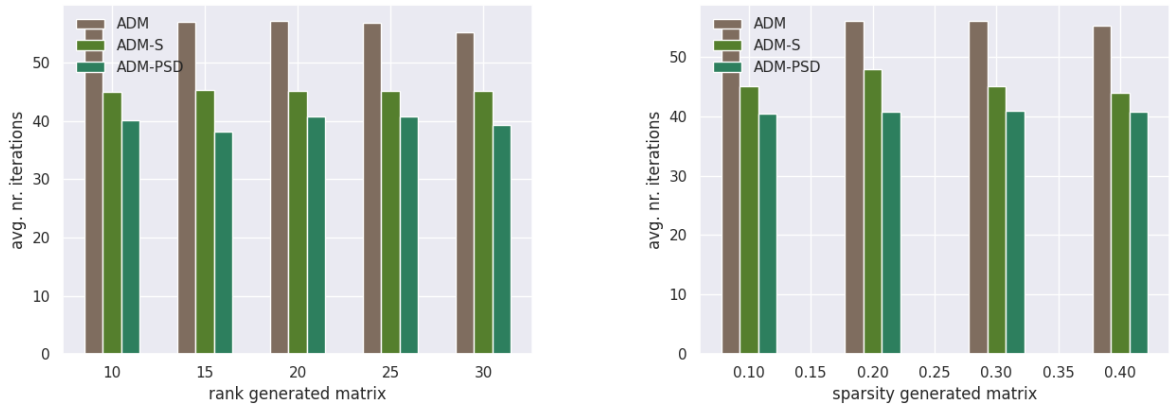
Table 1: APE.3 and APE.4: Average number of iteration until convergence for matrices generated of differend ranks (left) and sparsities (right).

rank	10	15	20	25	30		sparsity	0.1	0.2	0.3	0.4
ADM	55.9	57.0	57.1	56.8	55.2		ADM	55.9	56.0	56.0	55.2
ADM-S	45.0	45.3	45.1	45.1	45.1		ADM-S	45.0	47.9	45.0	44.0
ADM-PSD	40.1	38.2	40.8	40.8	39.3		ADM-PSD	40.5	40.7	40.9	40.7

7.4 Analysis and Discussion

PEE.1 showed that the subgradient descent method does not converge to an optimal value. APE.1 suggests that subgradient descent starts performing is worse the lower the initial rank, as seen in figure 3a. For sparsity, APE.2 shows no such decrease in performance when the sparsity changes 3b. Averaging per column in table 1, we see that ADM-S converges roughly 19.1% faster than regular ADM, and ADM-PSD converges approximately 28.6% faster than regular ADM.

In line with our hypothesis, the performance of our redefined subgradient descent algorithm is rather poor compared to the performance of ADM-S. This problem does not seem to occur when the sparsity changes, as seen in figure 3b, which makes it plausible that the reason that the algorithm does not converge lies in the rank-minimization. Given that we have rewritten our problem in such a way, that the subderivatives of our subgradient with respect of both $\mathbf{L}$ and $\mathbf{R}$ were singletons can



(a) APE.3 (b) APE.4

Figure 4: Average number of iteration until convergence for matrices generated of differend ranks (left) and sparsities (right).

be an explanation for this, for a lack of diversity in our subgradient often implies slower convergence. These results are in line with the results by [23] and [22].

Another problem mentioned in chapter 6 already, is that the projection function largely dictates the rate of convergence. This is because, depending on the projection function, in most steps, the improvement obtained by the algorithm is negated by the projection back on the constraint set. This latter issue is an issue inherent to the projection method, which has been a known problem with using projection methods [16]. This can have played a role in the lack of convergence, as well.

The results of the third and fourth APE support our hypothesis that the more we constrain our algorithms to the space of positive semi-definite matrices, the better they performed when doing sparse low-rank matrix decomposition on them. This is strongly reflected in that the average performance of AMD-S was roughly 19% more efficient than ADM, and the average performance of AMD-PSD was approximately 29% more efficient.

8 Conclusion

This thesis aimed to redefine the approximation algorithms for sparse low-rank matrix decomposition to perform better on the space of positive semi-definite matrices, as applicable in linear structural equation modeling. Using Lagrangian optimization theory, we derived the optimality conditions for the lagrangian of a convex relaxation of our problem, with which we derived closed-form updates for the Alternating Directions Method, ensuring our algorithm converges to the optimal decomposition. The experimental results indicate that the redefined ADM-S performs better than regular ADM, and ADM-PSD performs better than ADM-S, hence supporting our claim that the more constrained algorithms perform better than the standard algorithm when finding positive semi-definite decompositions.

To find these optimality conditions and closed-form updates, we first studied how the limitations imposed by our specific SEM graphs restricted our search space and hypothesized that restricting our search would improve the performance of the algorithm. In chapter 3, we then considered how we could relax our problem of S-SLRMD to a convex problem, such that it obeyed the properties used by the algorithms defined for SLRMD. In chapter 4, we laid the foundations for the first-order optimization, by defining a projected subgradient descent algorithm for S-SLRMD using the mathematics of matrix calculus and subgradients. This algorithm formed the basis for our second algorithm of Alternating Directions Method in chapter 5, for which we then derived the optimality conditions of the problem's Lagrangian and closed-form solutions for the updates of ADM. In chapter 6, we showed that PSD-SLRMD could be relaxed to a similar convex problem, and also derived the optimality conditions of the Lagrangian of PSD-SLRMD and closed-form updates for ADM-PSD. We then conducted a series of experiments, which indicated that our redefined approximation algorithms for SLRMD perform significantly better on positive semi-definite matrices than the regular algorithms do, support our hypothesis.

Given that ADM-PSD performs better on our problem than standard ADM on positive semi-definite matrices, the question arises if other first-order methods would perform better on the problem as well. To better understand the implication of our results, future studies could address the performance of other first-order methods, such as proximal gradient descent or the Frank-Wolfe algorithm. Moreover, second-order methods, such as Newton's method, could be considered.

Three SEM-specific limitations exist in our approach. First, as mentioned in 2.3, we would ideally have that our matrix $\mathbf{L}$ making up our decomposition is itself sparse in the context of SEM. Further research could be done in redefining the ADM-PSD algorithm in such a way that it ensures sparsity of $\mathbf{L}$. Second, our algorithm looks for sparse positive semi-definite components in general. The algorithm could be improved by analyzing how the sparse elements can be ensured to lie on the diagonal of our sparse matrix. Last, we studied one specific type of SEM, where there were no edges between the observed variables. Additional research has to be done to see how these algorithms can be extended to the case where $\mathbf{\Lambda}$ is non-empty (but sparse, for instance). Furthermore, the number of experiments conducted is limited. Further research could be done further by studying the interdependencies between the parameters γ, δ and ξ in ADM.

In many applications of SEM, researchers are restricted by the computational complexity of high dimensional data, for graphs grow exponentially in the number of vertices. This improved algorithm can be used to efficiently find sparse low-rank decompositions of the models based on these graphs, potentially aiding in the understanding of high dimensional SEMs. Understanding the causal relationships in a set of variables (especially understanding which relationships are not causal), can be of substantial benefit in interpreting machine learning models. This interpretability is crucial given the expected growth of societal impact of AI [26], especially considering the explainability of the ethical dilemmas autonomous systems will encounter, such as in self-driving cars, AI-powered analytics, and automated bias.

References

[1] Yonatan Amit, Michael Fink, Nathan Srebro, and Shimon Ullman. Uncovering shared structures in multiclass classification. In *Proceedings of the 24th international conference on Machine learning*, pages 17–24, 2007.

[2] Alfred Auslender. *Méthodes numériques pour la résolution des problèmes d'optimisation avec contraintes*. Faculté des sciences, Université de Grenoble, 1969.

[3] Abu SSM Barkat Ullah, Ruhul Sarker, and David Cornforth. Search space reduction technique for constrained optimization with tiny feasible space. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 881–888, 2008.

[4] G Belitskii et al. *Matrix norms and their applications*, volume 36. Birkhäuser, 2013.

[5] Dimitri P Bertsekas. On penalty and multiplier methods for constrained minimization. *SIAM Journal on Control and Optimization*, 14(2):216–235, 1976.

[6] Dimitri P Bertsekas and Athena Scientific. *Convex optimization algorithms*. Athena Scientific Belmont, 2015.

[7] Kenneth A Bollen and Rick H Hoyle. Latent variables in structural equation modeling. 2012.

[8] James P Boyle and Richard L Dykstra. A method for finding projections onto the intersection of convex sets in hilbert spaces. In *Advances in order restricted statistical inference*, pages 28–47. Springer, 1986.

[9] Emmanuel J Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):1–37, 2011.

[10] Venkat Chandrasekaran, Sujay Sanghavi, Pablo A Parrilo, and Alan S Willsky. Sparse and low-rank matrix decompositions. *IFAC Proceedings Volumes*, 42(10):1493–1498, 2009.

[11] Pei Chen and David Suter. Recovering the missing components in a large noisy low-rank matrix: Application to sfm. *IEEE transactions on pattern analysis and machine intelligence*, 26(8):1051–1063, 2004.

[12] Mathias Drton et al. Algebraic problems in structural equation modeling. In *The 50th anniversary of Gröbner bases*, pages 35–86. Mathematical Society of Japan, 2018.

[13] Ky Fan and Alan J Hoffman. Some metric inequalities in the space of matrices. *Proceedings of the American Mathematical Society*, 6(1):111–116, 1955.

[14] Roland Glowinski and Patrick Le Tallec. *Augmented Lagrangian and operator-splitting methods in nonlinear mechanics*. SIAM, 1989.

[15] Hani Hagras. Toward human-understandable, explainable ai. *Computer*, 51(9):28–36, 2018.

[16] Shih-Ping Han. A successive projection method. *Mathematical Programming*, 40(1):1–14, 1988.

[17] Magnus R Hestenes. Multiplier and gradient methods. *Journal of optimization theory and applications*, 4(5):303–320, 1969.

[18] Nicholas J Higham. Computing the nearest correlation matrix—a problem from finance. *IMA journal of Numerical Analysis*, 22(3):329–343, 2002.

[19] Richard D Hill and Steven R Waters. On the cone of positive semidefinite matrices. *Linear Algebra and its Applications*, 90:81–88, 1987.

[20] Po-Sen Huang, Scott Deeann Chen, Paris Smaragdis, and Mark Hasegawa-Johnson. Singing-voice separation from monaural recordings using robust principal component analysis. In *2012*

IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 57–60. IEEE, 2012.

[21] Donald A Jackson and Yong Chen. Robust principal component analysis and outlier detection with ecological data. *Environmetrics: The official journal of the International Environmetrics Society*, 15(2):129–139, 2004.

[22] Krzysztof C Kiwiel, Torbjörn Larsson, and Per Olov Lindberg. Lagrangian relaxation via ballstep subgradient methods. *Mathematics of Operations Research*, 32(3):669–686, 2007.

[23] Claude Lemaréchal. Lagrangian relaxation. In *Computational combinatorial optimization*, pages 112–156. Springer, 2001.

[24] ZS Liu, JC Li, Guo Li, JC Bai, and XN Liu. A new model for sparse and low-rank matrix decomposition. *J. Appl. Anal. Comput*, 7(2):600–616, 2017.

[25] Paul-Emile Maingé. Strong convergence of projected subgradient methods for nonsmooth and nonstrictly convex minimization. *Set-valued analysis*, 16(7-8):899–912, 2008.

[26] Spyros Makridakis. The forthcoming artificial intelligence (ai) revolution: Its impact on society and firms. *Futures*, 90:46–60, 2017.

[27] Matthew P Masarik, Joseph Burns, Brian T Thelen, Jack Kelly, and Timothy C Havens. Gpr anomaly detection with robust principal component analysis. In *Detection and Sensing of Mines, Explosive objects, and Obscured targets XX*, volume 9454, page 945414. International Society for Optics and Photonics, 2015.

[28] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[29] Yuchuan Qiao, Baldur van Lew, Boudewijn P. F. Lelieveldt, and Marius Staring. Fast automatic step size estimation for gradient descent optimization of image registration. *IEEE Transactions on Medical Imaging*, 35(2):391–403, 2016.

[30] Donald J Robinaugh, Nicole J LeBlanc, Heidi A Vuletich, and Richard J McNally. Network analysis of persistent complex bereavement disorder in conjugally bereaved adults. *Journal of abnormal psychology*, 123(3):510, 2014.

[31] Ralph Tyrell Rockafellar. *Convex analysis*. Princeton university press, 2015.

[32] Paul D Sampson, Ann P Streissguth, Fred L Bookstein, Ruth E Little, Sterling K Clarren, Philippe Dehaene, James W Hanson, and John M Graham Jr. Incidence of fetal alcohol syndrome and prevalence of alcohol-related neurodevelopmental disorder. *Teratology*, 56(5):317–326, 1997.

[33] Yuan Shen, Zaiwen Wen, and Yin Zhang. Augmented lagrangian alternating direction method for matrix separation based on low-rank factorization. *Optimization Methods and Software*, 29(2):239–263, 2014.

[34] Naum Zuselevich Shor. *Minimization methods for non-differentiable functions*, volume 3. Springer Science & Business Media, 2012.

[35] M Yu Smirnov and GD Egbert. Robust principal component analysis of electromagnetic arrays with missing data. *Geophysical Journal International*, 190(3):1423–1438, 2012.

[36] M Srinivas and LM Patnaik. Learning neural network weights using genetic algorithms-improving performance by search-space reduction. In *[Proceedings] 1991 IEEE International Joint Conference on Neural Networks*, pages 2331–2336. IEEE, 1991.

[37] Seth Sullivant, Kelli Talaska, and Jan Draisma. Trek separation for gaussian graphical models. *The Annals of Statistics*, 38(3):1665–1685, 2010.

[38] Lieven Vandenberghe and Stephen Boyd. Semidefinite programming. *SIAM review*, 38(1):49–95, 1996.

[39] G Alistair Watson. Characterization of the subdifferential of some matrix norms. *Linear algebra and its applications*, 170:33–45, 1992.

[40] Robert J Williams, Felix S Odaibo, and Janet M McGee. Incidence of fetal alcohol syndrome in northeastern manitoba. *Canadian Journal of Public Health*, 90(3):192–194, 1999.

[41] Guandong Xu, Tri Dung Duong, Qian Li, Shaowu Liu, and Xianzhi Wang. Causality learning: A new perspective for interpretable machine learning. *arXiv preprint arXiv:2006.16789*, 2020.

[42] Di Yan, Tao Wu, Ying Liu, and Yang Gao. An efficient sparse-dense matrix multiplication on a multicore system. In *2017 IEEE 17th International Conference on Communication Technology (ICCT)*, pages 1880–1883. IEEE, 2017.

[43] Shicheng Yang, Le Zhang, Lianghua He, and Ying Wen. Sparse low-rank component-based representation for face recognition with low-quality images. *IEEE Transactions on Information Forensics and Security*, 14(1):251–261, 2018.

[44] Xiyu Yu, Tongliang Liu, Xinchao Wang, and Dacheng Tao. On compressing deep models by low rank and sparse decomposition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7370–7379, 2017.

[45] Xiaoming Yuan and Junfeng Yang. Sparse and low-rank matrix decomposition via alternating direction methods. *preprint*, 12(2), 2009.

A Linear Algebra

A.1 Basic Definitions

The **real numbers** (denoted as $\mathbb{R}$) is the set of all - possibly infinite - decimal expansions. A **real matrix** is a two dimensional array of real numbers. We denote the set of all real matrices with m rows and n columns as $\mathbb{R}^{m\times n}$ and we call a matrix **square** if the number of rows is equal to the number of columns. For some arbitrary real matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, we refer to its entry in row i and column j as a_{ij} or $[\mathbf{A}]_{ij}$, i.e.

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & ... & a_{1m} \\ a_{21} & a_{22} & ... & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & ... & a_{nm} \end{pmatrix}. \tag{39}$$

We call $\mathbf{A^T}$ the **transpose** of $\mathbf{A}$, if the i-th row, j-th column element of $\mathbf{A^T}$ is the j-th row, i-th column element of $\mathbf{A}$, i.e.

$$[\mathbf{A^T}]_{ij} = [\mathbf{A}]_{ji}.$$

A matrix equal to its transpose is called **symmetric**. We denote the set of n by n symmetric matrices as $\mathbb{S}^n$.

Example: The matrix $\begin{pmatrix} 5 & 10 & 7 \\ 10 & 6 & 8 \\ 7 & 8 & 9 \end{pmatrix}$ in an element of $\mathbb{S}^3$.

Four main properties of the transpose used are the following:

- $(\mathbf{M^T})^\mathbf{T} = \mathbf{M}$
- $(\mathbf{M} + \mathbf{N})^\mathbf{T} = \mathbf{M^T} + \mathbf{N^T}$
- $(\lambda\mathbf{M})^\mathbf{T} = \lambda\mathbf{M^T}$
- $(\mathbf{AB})^\mathbf{T} = \mathbf{B^T A^T}$.

If two matrices $\mathbf{A}, \mathbf{B}$ have the same number of rows and columns, we define addition as elementwise addition, i.e.

$$[\mathbf{A} + \mathbf{B}]_{ij} = [\mathbf{A}]_{ij} + [\mathbf{B}]_{ij}.$$

Moreover, we can multiply some matrix $\mathbf{A}$ by a number $\lambda \in \mathbb{R}$, called a **scalar**, by multiplying each element of $\mathbf{A}$ by λ, i.e.

$$[\lambda\mathbf{A}]_{ij} = \lambda[\mathbf{A}]_{ij}.$$

Let $\mathbf{A} \in \mathbb{R}^{m\times n}, \mathbf{B} \in \mathbb{R}^{n\times p}$. Using standard matrix multiplication, we have that the i, j entry of the multiple $\mathbf{AB}$ is

$$[\mathbf{AB}]_{ij} = \sum_{k=1}^{n} a_{ik}b_{kj}.$$

Let $\mathbf{a}_i$ refer to the row i of the matrix $\mathbf{A}$, and $\mathbf{b}_j$ refer to column j of $\mathbf{B}$, such that

$$[\mathbf{AB}]_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} =: \mathbf{a_i^T} \cdot \mathbf{b_j},$$

where $\cdot$ denotes the dot product. We can therefore write

$$\mathbf{AB} = \begin{pmatrix} \mathbf{a_1^T b_1} & \mathbf{a_1^T b_2} & ... & \mathbf{a_1^T b_n} \\ \mathbf{a_2^T b_1} & \mathbf{a_2^T b_2} & ... & \mathbf{a_2^T b_n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{a_n^T b_1} & \mathbf{a_n^T b_2} & ... & \mathbf{a_n^T b_n} \end{pmatrix}.$$

A few important properties of matrix multiplication are the following:

(a) $(\mathbf{AB})\mathbf{C} = \mathbf{A}(\mathbf{BC})$

(b) $\mathbf{A}(\mathbf{B} + \mathbf{C}) = \mathbf{AB} + \mathbf{AC}$ and $(\mathbf{B} + \mathbf{C})\mathbf{A} = \mathbf{BA} + \mathbf{CA}$.

Specifically, matrix multiplication is not commutative, i.e. $\mathbf{AB} \neq \mathbf{BA}$ in general.

Let $\mathbf{I_n}$ denote the n by n matrix with ones on the diagonal, and zeros elsewhere, i.e.

$$\mathbf{I_n} = \begin{pmatrix} 1 & 0 & ... & 0 \\ 0 & 1 & ... & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & ... & 1 \end{pmatrix},$$

called the **identity matrix**. If the dimension of $\mathbf{I}$ is clear from the context, the subscript may be dropped. For all matrices $\mathbf{A}$, we have that

$$\mathbf{AI} = \mathbf{IA} = \mathbf{A}.$$

If for some matrix $\mathbf{A}$, there exists some matrix $\mathbf{B}$ such that

$$\mathbf{AB} = \mathbf{I},$$

we call $\mathbf{A}$ **invertible** and $\mathbf{B}$ its **inverse**, denoted as $\mathbf{A^{-1}} = \mathbf{B}$. We occasionally take the inverse and transpose of some matrix $\mathbf{M}$, which we denote as $\mathbf{M^{-T}}$.

A matrix is called unitary, if its transpose is its inverse, i.e. $\mathbf{A}$ is unitary if and only if

$$\mathbf{AA^T} = \mathbf{I} = \mathbf{A^T A}.$$

We denote the n by n matrix of zeros as $\mathbf{0_n}$. For all matrices $\mathbf{A}$, we have that

$$\mathbf{A0} = \mathbf{0A} = \mathbf{0}.$$

Please note that again subscripts may be dropped when the dimension of $\mathbf{0}$ are evident.

The trace of $\mathbf{A}$ (notation: $\mathrm{tr}(\mathbf{A})$) is defined as the sum of the elements on the diagonal from the upper left to lower right of $\mathbf{A}$, i.e.

$$\mathrm{tr}(\mathbf{A}) := \sum_{i=1}^{n} [\mathbf{A}]_{ii}.$$

The traces obeys the following properties:

(a) $\text{tr}(\mathbf{A} + \mathbf{B}) = \text{tr}(\mathbf{A}) + \text{tr}(\mathbf{B})$

(b) $\text{tr}(c\mathbf{A}) = c\text{tr}(\mathbf{A})$

(c) $\text{tr}(\mathbf{A}) = \text{tr}(\mathbf{A}^\mathbf{T})$.

(d) $\text{tr}(\mathbf{AB}) = \text{tr}(\mathbf{BA})$.

Please note that (d) implies that $\text{tr}(\mathbf{ABC}) = \text{tr}(\mathbf{BCA}) = \text{tr}(\mathbf{CAB})$, but does not imply that $\text{tr}(\mathbf{ABC}) = \text{tr}(\mathbf{BAC})$. In general, the trace in invariant under cycle permutations, as seen above.

The **Hadamard product** is the element-wise multiplication, denoted by $\circ$, that is

$$[\mathbf{A} \circ \mathbf{B}]_{ij} = [\mathbf{A}_{ij}] \cdot [\mathbf{B}]_{ij}.$$

A matrix is called **sparse** if that matrix contains mostly zero entries.[4] The number of zero-valued entries of that matrix divided by the total number of elements is referred to as its **sparsity**.

Proposition A.1. *Adding some matrix to its transpose yields a symmetric matrix. Moreover, multiplying a matrix by its transpose yields a symmetric matrix as well.*

Proof. Let $\mathbf{M} \in \mathbb{R}^{m \times n}$ be some arbitratry matrix. Since $(\mathbf{M} + \mathbf{M}^\mathbf{T})^\mathbf{T} = \mathbf{M}^\mathbf{T} + \mathbf{M} = \mathbf{M} + \mathbf{M}^\mathbf{T}$, we conclude that $\mathbf{M} + \mathbf{M}^\mathbf{T}$ is symmetric. Moreover, since $(\mathbf{MM}^\mathbf{T})^\mathbf{T} = (\mathbf{M}^\mathbf{T})^\mathbf{T}\mathbf{M}^\mathbf{T} = \mathbf{MM}^\mathbf{T}$, we conclude that $\mathbf{MM}^\mathbf{T}$ is symmetric, as well. □

A.2 Vector Spaces

A vector is often depicted as an object describing both magnitude and direction. The operations such vectors can perform are quite restricted: two vectors can either be added together to form a third vector or be scaled by some constant. If we restrict ourselves to arrows in two-dimensional space we can see that the product of applying only addition and scalar multiplication always still lie within two-dimensional space (and thus are vectors still).

Arrows in Euclidian spaces are, however, not the only collection obeying these properties. One of many examples of collections that also have this property is the set of linear function over x with real coefficients, denoted $\mathbb{R}[x]_{\leq 1}$. Given some function $ax + b \in \mathbb{R}[x]_{\leq 1}$, we can quickly see that multiplying this function by some scalar $c \in \mathbb{R}$ we get that

$$c \cdot (ax + b) = (ca)x + cb \in \mathbb{R}[x]_{\leq 1},$$

hence being closed under scalar multiplication. Moreover, adding two functions $ax + b, a'x + b' \in \mathbb{R}[x]_{\leq 1}$ gives us

$$ax + b + a'x + b' = (a + a')x + (b + b') \in \mathbb{R}_{\leq 1},$$

hence also being closed under addition. One other such example is the set of all continuous functions.

We can therefore directly translate all the concepts and ideas from linear algebra that are intuitively defined for arrows in space to such sets, allowing us to deduce properties of these families using linear algebra.

In general, we call every set the obeys these properties a **vector space**, and the elements in the sets are called **vectors**. A more rigorous definition of a vector space is listed in C.

[4] There is no strict definition of how many elements need to be zero for a matrix to be considered sparse [42].

A.3 Subspaces

Let V be a vectorspace and let $S \subseteq V$. We call S a **subspace** of V if S itself forms a linear space, that is

(a) S is non empty.

(b) If $\mathbf{u}, \mathbf{v} \in V$, then $\mathbf{u} + \mathbf{v} \in V$.

(c) If $\mathbf{v} \in V$, then $\lambda \mathbf{v} \in V$ for all $\lambda \in \mathbb{R}$.

Example: In $\mathbb{R}^2$, every line through the origin is a linear subspace.

Please note that every vector space is a subspace of itself. A few important subspaces of any vector space are:

- The column space Col($\mathbf{M}$), the set of all linear combinations of the columns of $\mathbf{M}$, referred to as **column vectors**.
- The row space Row($\mathbf{M}$), the set of all linear combinations of the rows of $\mathbf{M}$, referred to as **row vectors**.
- The null space Nul($\mathbf{M}$), the set of all solution of the equation $\mathbf{Mx} = \mathbf{0}$.

A.4 Dependence and Rank

Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ with column vectors $\mathbf{a_1}, \mathbf{a_2}, ..., \mathbf{a_n}$. We call any set of vectors **dependent** if we can find $\lambda_1, \lambda_2, ..., \lambda_n$ not all zero such that

$$\lambda_1 \mathbf{a_1} + \lambda_2 \mathbf{a_2} + \lambda_n \mathbf{a_n} = \mathbf{0}.$$

If we could find such λ for which the above equation hold such that $\lambda_i \neq 0$, we could rewrite $\mathbf{a_i}$ as a linear combination of the vectors in the original set, i.e.

$$\mathbf{a}_i = -\frac{\lambda_1}{\lambda_i}\mathbf{a_1} - ... - \frac{\lambda_{i-1}}{\lambda_i}\mathbf{a_{i-1}} - \frac{\lambda_{i+1}}{\lambda_i}\mathbf{a_{i+1}} - ... - \frac{\lambda_n}{\lambda_i}\mathbf{a_n}. \tag{40}$$

The **column rank** of some matrix, is the maximum number of independent column vectors it has. Similarly, the **row rank** is the maximum number of independent row vectors it has. A famous result in linear algebra shows us that the two are always equal, hence we often refer to **rank** in general. A matrix has **full rank** if its rank is maximized, i.e. equal to the lesser of the number of columns and rows. A major theory in linear algebra tells us that a matrix is invertible if and only if it has full rank.

Given that the columns of $\mathbf{A}$ are the rows of $\mathbf{A^T}$ and vice versa, we know that the rank of a matrix is equal to the rank of its transpose.

A.5 Inner Products

Often, we want to define concepts such as length, distance, and angles on our vector space. In order to do this, we define an inner product on our vector space, describing how much of one vector is pointing in the direction of another one. Intuitively, we want this similarity between a vector and itself to be zero and the similarity between one vector and another to be equal to the similarity between the other and the one. Lastly, we want our metric to be linear.

Let V be an arbitrary vector space and let $\langle \cdot, \cdot \rangle : V \times V \to \mathbb{R}$ be a function obeying the following properties:

- $\langle \mathbf{v}, \mathbf{u} \rangle \geq 0$ for all $\mathbf{v}, \mathbf{u} \in V$, and $\langle \mathbf{v}, \mathbf{v} \rangle = 0 \iff \mathbf{v} = \mathbf{0}$

- $\langle \mathbf{v}, \mathbf{u} \rangle = \langle \mathbf{u}, \mathbf{v} \rangle$ for all $\mathbf{v}, \mathbf{u} \in V$
- $\langle a\mathbf{v} + b\mathbf{u}, \mathbf{w} \rangle = a\langle \mathbf{u}, \mathbf{w} \rangle + b\langle \mathbf{v}, \mathbf{w} \rangle$ for all $\mathbf{u}, \mathbf{v}, \mathbf{w} \in V$ and $a, b \in \mathbb{R}$.

Example: The function

$$\langle f, g \rangle = \int_0^1 f(x)g(x)dx$$

is a inner product on the vector space of continous functions.

A.5.1 Norms

A **norm** defined on V is a function

$$|| \cdot || : V \to \mathbb{R}$$

that defines the length of a vector, that is the distance from the origin. Intuitively, we want the norm of a vector to be non-negative, and only zero if the vector itself is zero. Moreover, we want our distances to obey the triangle inequality, in order for our metric to resemble standard Euclidean geometry. The triangle inequality states that the sum of the lenghts of two sides of any triangle must be greater than or equal to the remaining side. In other words, that the direct path between points a and b is always smaller than or equal to taking the detour through c. Lastly, we want the length of a some n times some vector to be equal to n times the length of that vector, giving us the following properties:

(a) $||\mathbf{v}|| \geq 0$, and $||\mathbf{v}|| = 0 \iff \mathbf{v} = \mathbf{0}$

(b) $||a\mathbf{v}|| = a||\mathbf{v}||$

(c) $||\mathbf{v} + \mathbf{u}|| \leq ||\mathbf{v}|| + ||\mathbf{u}||$ (**triangle inequality**)

It neat property of an inner product is that every inner product on V induces a norm on V, that is

$$||\mathbf{v}|| := \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle}$$

is always a norm on V. The converse is, however, not true in general.

We can now straightforwardly define the distance between two vectors, by considering the length of the difference between the two, i.e.

$$d(\mathbf{v}, \mathbf{u}) = ||\mathbf{v} - \mathbf{u}||.$$

We refer to a vector space with an inner product as an **inner product space** and refer to a vector space with a norm as a **normed space**.

A.6 Vector Space of Matrices

Two vector spaces extensively used are the space of m by n matrices, i.e. $\mathbb{R}^{m \times n}$, and the space of symmetric matrices, i.e. $\mathbb{S}^n$. That is to say, adding or scaling any (symmetric) matrix, is guaranteed to yield another (symmetric) matrix. The standard inner product used on $\mathbb{R}^{m \times n}$ and $\mathbb{S}^n$ is the **Frobenius inner product**, defined as

$$\langle \mathbf{M}, \mathbf{N} \rangle_F = \text{tr}(\mathbf{M}^\mathbf{T}\mathbf{N}),$$

inducing the **Frobenius norm**, i.e.

$$||\mathbf{M}||_F := \sqrt{\text{tr}(\mathbf{M}^\mathbf{T}\mathbf{M})}.$$

The details why these sets form vector spaces will be discussed in C.

A.7 Eigenvalues

If we consider some matrix $\mathbf{M} \in \mathbb{R}^{m \times n}$, we can consider the effect of multiplying it times some arbitrary vector $\mathbf{v} \in \mathbb{R}^n$. Every nonzero vector for which the above multiplication results in only a change by a scalar factor is called an **eigenvector**, and corresponding scalar is called an **eigenvalue**, i.e. if

$$\mathbf{Mv} = \lambda \mathbf{v},$$

we call $\mathbf{v}$ an eigenvector and λ an eigenvalue of $\mathbf{M}$.

A major result in linear algebra tells us that all eigenvalues of symmetric matrices are real. If we assume - without loss of generality - that $\lambda_1 \geq \lambda_2 \geq ... \geq 0$, we call

$$\sigma_i = \sqrt{\lambda_i}$$

a **singular value** of this symmetric matrix. Moreover, it can be shown that the rank of a matrix is equal to the number of nonzero singular values.

The **singular value decomposition** of matrix $\mathbf{M}$ is given by

$$\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathbf{T}},$$

such that $\mathbf{U}, \mathbf{V}$ are unitary matrices and $\mathbf{\Sigma}$ is a matrix with the singular values on the diagonal.

A.8 Definite Matrices

If for some symmetric matrix $\mathbf{M}$ with real entries it holds that $\mathbf{x}^{\mathbf{T}}\mathbf{M}\mathbf{x} > 0$ for all $\mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$, we call $\mathbf{M}$ **positive definite**. Moreover, if we have that $\mathbf{x}^{\mathbf{T}}\mathbf{M}\mathbf{x} \geq 0$ for all $\mathbf{x} \in \mathbb{R}^n$, we call $\mathbf{M}$ **positive semi-definite**. We denote the set of positive definite matrices and positive semi-definite matrices as $\mathbb{S}^n_{++}$ and $\mathbb{S}^n_{+}$ respectively.

A property of positive definite matrices is that its eigenvalues are all positive. Comparably, if a matrix is positive semi-definite its eigenvalues are all non-negative.

Proposition A.2. *If* $\mathbf{M} = \mathbf{L}^{\mathbf{T}}\mathbf{L}$*, then* $\mathbf{M}$ *is postive semi-definite.*

Proof. If we can express $\mathbf{M} = \mathbf{L}^{\mathbf{T}}\mathbf{L}$ for some matrix $\mathbf{L}$, we see that

$$\begin{aligned} \mathbf{x}^{\mathbf{T}}\mathbf{M}\mathbf{x} &= \mathbf{x}^{\mathbf{T}}\mathbf{L}^{\mathbf{T}}\mathbf{L}\mathbf{x} \\ &= (\mathbf{L}\mathbf{x})^{\mathbf{T}}(\mathbf{L}\mathbf{x}) \\ &= ||\mathbf{L}\mathbf{x}||^2_{\mathrm{F}} \\ &\geq 0, \end{aligned}$$

where $||\cdot||_{\mathrm{F}}$ denotes the Frobenius norm as defined in (A.6), finishing our proof. □

B Calculus

B.1 Derivatives and Gradients

One of the major ideas studied in calculus is the derivative. The **derivative** of a function is a new function that describes the amount by which the original function is changing in each point, denoted as $\frac{df}{dx}$ or $f'(x)$ for some original function f in x. More exactly, it is the change in f evaluated in some point x and some second point $x+h$ as the two get closer together, i.e.

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}.$$

If some function has multiple variables, we define the **partial derivative** (notation: $\frac{\partial f}{\partial x_i}$ or $\partial_{x_i} f$), describing the change in f given a infintely small change in variable x_i, keeping the other variables constant, i.e.

$$\frac{\partial f}{\partial x_i} = \lim_{h \to 0} \frac{f(x_1, ..., x_i + h, ..., x_n) - f(x_1, ..., x_n)}{h}.$$

We can combine all possible partial derivatives of some function into one vector, called the **gradient** (notation: ∇f), such that

$$\nabla f : \mathbb{R}^n \to \mathbb{R}^n, \quad \nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(\mathbf{x}) \\ ... \\ \frac{\partial f}{\partial x_n}(\mathbf{x}) \end{bmatrix}.$$

The **Hessian** of f (notation: $\mathbf{H}_f$) is the $n \times n$ matrix, containing all second partial derivative of f, i.e.

$$[\mathbf{H}_f]_{i,j} = \frac{\partial}{\partial x_i} \left(\frac{\partial f}{\partial x_j} \right) = \frac{\partial^2 f}{\partial x_i \partial x_j}.$$

A **tangent line** of a point of some function is a line that 'touches' the function in that point. Formally, this implies that a line $l : y = ax + b$ is a tangent line of f in p if $ap + b = f(p)$ and $a = f'(p)$, with can more consisely be stated as

$$y = f(p) + a(x - p).$$

C Abstract Algebra

A **binary operation** $*$ on set S is a mapping of the Cartesian product of S to S:

$$* : S \times S \to S.$$

Given that applying the operation to any two elements in a set yields an element in the set, we call the set **closed** under the operation.

C.1 Groups and Fields

We call a set S together with an operation $*$ a **group** if the following axioms are satisfied:

- The operation $*$ is **associative**, i.e. for all $a, b, c \in S$, one has that $(a * b) * c = a * (b * c)$.
- There exists some $e \in S$ such that $a * e = e * a = a$ for all $a \in S$. We call e an **identity element**.
- For all elements $s \in S$ there is some element $s' \in S$ such that $s * s' = e$. We call s' the **inverse** of s.

Moreover, if the operation is **commutative**, i.e. for all $a, b \in S$ we have that $a * b = b * a$, the group is said to be **Abelian**.

Example: The integers form an Abelian group under addition, denoted $(\mathbb{Z}, +)$. We know that $(a + b) + c = a + (b + c)$ for all $a, b, c \in \mathbb{Z}$. Moreover, we have 0 as an identity element, since $a + 0 = 0 + a = a$ for all $a \in \mathbb{Z}$. Furthermore, there exists an inverse for all $a \in \mathbb{Z}$, for $a + (-a) = 0$. Lastly, $a + b = b + a$ for all $a, b \in \mathbb{Z}$.

We call a set F together with two operations, called addition $+$ and multiplication $\cdot$, a **field** if the following axioms are satisfied:

- Additive and multiplicative associativity: $a + (b + c) = (a + b) + c$ and $a \cdot (b \cdot c) = (a \cdot b) \cdot c$ for all $a, b, c \in F$.
- Additive and multiplicative commutativity: $a + b = b + a$, and $a \cdot b = b \cdot a$.
- Additive and multiplicative identities: there exist two distinct elements 0 and 1 in F such that $a + 0 = a$ and $a \cdot 1 = a$.
- Additive inverses: for every $a \in F$, thre exists some $(-a) \in F$ such that $a + (-a) = 0$, called its inverse.
- Multiplicative inverses: for every $a \in F - \{0\}$, there exists an $a^{-1} \in F$ such that $a \cdot a^{-1} = 1$.
- Distributivity: $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$ for all $a, b, c \in F$.

In other words, a field is a set with two operations, such that it forms an abelian group under addition with 0 as its identity, and all the nonzero elements also form an abelian group under multiplication with 1 as its identity. Moreover, the distributivity 'connects' the operations.

Example: The real numbers form a field, with the number 0 as its additive identity and 1 as its multiplicative identity.

C.2 Vector Spaces

We call some set V, together with two operations addition and multiplication a **vector space** over some field F (notation: $(V, +, \cdot)_F$), if

(a) V is an Abelian group under addition

(b) For every vector $v \in V$ and every scalar $f \in F$ we have that $f \cdot v \in V$.

(c) $f \cdot (v_1 + v_2) = f \cdot v_1 + f \cdot v_2$ and $(f_1 + f_1) \cdot v = f_1 \cdot v + f_2 \cdot v$

(d) $f_1 \cdot (f \cdot v) = (f_1 \cdot f_2) \cdot v$

(e) $1 \cdot v = v$.

All the vector spaces considered in this thesis are defined over the reals.

Proving all the individual properties mentioned in C.2 for both $\mathbb{R}^{m \times n}$ and $\mathbb{S}^n$ are vector spaces can be done by using the properties of matrices defined in A.1, where $\mathbf{I}_n$ is used as the multiplicative identity and $\mathbf{0_n}$ as the additive identity.

D Reduction

In this section, we shortly reflect on what is meant when we say that $\mathbb{S}^n_+$ is smaller than $\mathbb{R}^{n\times n}$.

We know that $[a, b] \sim \mathbb{R}$ for all $a, b \in \mathbb{R}$ such that $a < b$. Moreover, we know that

$$[-3, 3] \subset [-5, 5].$$

Hence, there exist (uncountably many) elements in $[-5, 5]$ that are not in $[-3, 3]$. However, we know that since $[-3, 3] \sim \mathbb{R}$ and $[-5, 5] \sim \mathbb{R}$ and $\sim$ is an equivalence relation, that

$$[-3, 3] \sim [-5, 5].$$

A veridical paradox arises when one asks if $[-5, 5]$ contains more elements than $[-3, 3]$.

We ask ourselves what percentage of squared real matrices is symmetric, or equivalently what the probability are that a given square real matrix is symmetric, i.e.

$$\mathbb{P}[\mathbf{M} \in \mathbb{S}^n \mid \mathbf{M} \in \mathbb{R}^{n\times n}].$$

On one hand, we have that all symmetric matrices are square real matrices, but not all square real matrices are symmetric, that is

$$\mathbb{S}^n \subset \mathbb{R}^{n\times n}.$$

However, given that matrices are mere collections of real numbers, we know that $\mathbb{R}^{n\times n} \sim \mathbb{S}^n$, and a similar veridical paradox arises.

There is a (pragmatic) cheat out, for we cannot generate uncountably many matrices in our simulations. Imagine a n by n matrix of real numbers, where each entry is represented by a B-bit number. Since each number consists of B bits, we have a total of 2^B different values any entry can take on. Hence we have $(2^B)^{n^2} = 2^{Bn^2}$ different real matrices.

Moreover, a matrix $\mathbf{M}$ is symmetric if for all entries $m_{ij} = m_{ji}$ for all i, j. Considering that the n diagonal entries do not matter for the symmetry, we can imagine the other $n^2 - n$ elements in such a way that half of them lie under the diagonal and the other half above. Without loss of generality, we 'fix' the lower half and want all upper $\frac{n^2-n}{2}$ elements to match the lower ones. Assuming that all values have an equal probability of being chosen and each entries is chosen independently of the others, the odds of such a match is $\frac{1}{2^B}$, giving a total probability of

$$\mathbb{P}[\mathbf{M} \in \mathbb{S}^n \mid \mathbf{M} \in \mathbb{R}^{n\times n}] = (\frac{1}{2^B})^{\frac{n^2-n}{2}} = 2^{-\frac{1}{2}Bn(n-1)}.$$

That is, for some 32 bit number and matrix of dimension 10, we have that

$$\mathbb{P}[\mathbf{M} \in \mathbb{S}^n \mid \mathbf{M} \in \mathbb{R}^{n\times n}] \approx \frac{1}{10^{434}}.$$

For comparison, is is estimated that there are roughly 10^{80} atoms in the observable universe.

A similar argument can be made when considering if $\mathbb{S}^n_+$ is smaller than $\mathbb{S}^n$, hence in an applied setting, we can say that $\mathbb{S}^n_+$ is smaller than $\mathbb{S}^n$, which again is smaller than $\mathbb{R}^{n\times n}$, even though the sets are equinumerous in a theoretical sense.

E Proofs

Lemma E.1: The intersection of convex sets is a convex set.

Proof. Let $C_1, C_2, ..., C_n$ be convex sets and let s and t lie in their intersection. If the intersection is empty, the result follows trivially. If we assume that the intersection is non-empty, then by definition of the intersection we know that s and t lie in the individual sets. The convexity of these sets implies that the line segment between s and t lie in the individual sets as well, hence implying it lies in their intersection, proving our lemma. □

Proposition 3.5: The constraint set $\mathcal{C}_{\text{S-SLRMD}}$ is convex.

Proof. To see that $\mathcal{C}_{\text{S-SLRMD}}$ is convex, we first notice that $\mathcal{C}_{\text{S-SLRMD}}$ is the intersection of the sets of the individual contraints, that is

$$\mathcal{C}_{\text{S-SLRMD}} = \{\begin{bmatrix}\mathbf{A}\\ \mathbf{B}\end{bmatrix} \mid \mathbf{A}+\mathbf{B}-\mathbf{C}=\mathbf{0_n}\} \cap \{\begin{bmatrix}\mathbf{A}\\ \mathbf{B}\end{bmatrix} \mid \mathbf{A}-\mathbf{A^T}=\mathbf{0_n}\}.$$

Let $\begin{bmatrix}\mathbf{A_1}\\ \mathbf{B_1}\end{bmatrix}, \begin{bmatrix}\mathbf{A_2}\\ \mathbf{B_2}\end{bmatrix} \in \{\begin{bmatrix}\mathbf{A}\\ \mathbf{B}\end{bmatrix} \mid \mathbf{A}+\mathbf{B}-\mathbf{C}=\mathbf{0_n}\}$. We notice that for some $\theta \in [0,1]$ we have the following convex combination:

$$\theta \begin{bmatrix}\mathbf{A_1}\\ \mathbf{B_1}\end{bmatrix} + (1-\theta)\begin{bmatrix}\mathbf{A_2}\\ \mathbf{B_2}\end{bmatrix} = \begin{bmatrix}\theta\mathbf{A_1}+(1-\theta)\mathbf{A_2}\\ \theta\mathbf{B_1}+(1-\theta)\mathbf{B_2}\end{bmatrix}.$$

Using the fact that $\mathbf{A_1}+\mathbf{B_1}=\mathbf{A_2}+\mathbf{B_2}=\mathbf{C}$, we see that

$$\begin{aligned}[\theta\mathbf{A_1}+(1-\theta)\mathbf{A_2}]+[\theta\mathbf{B_1}+(1-\theta)\mathbf{B_2}] &= \theta(\mathbf{A_1}+\mathbf{B_1})+(1-\theta)(\mathbf{A_2}+\mathbf{B_2})\\ &= \theta\mathbf{C}+(1-\theta)\mathbf{C}\\ &= \mathbf{C},\end{aligned}$$

which implies that $\begin{bmatrix}\theta\mathbf{A_1}+(1-\theta)\mathbf{A_2}\\ \theta\mathbf{B_1}+(1-\theta)\mathbf{B_2}\end{bmatrix} \in \{\begin{bmatrix}\mathbf{A}\\ \mathbf{B}\end{bmatrix} \mid \mathbf{A}+\mathbf{B}-\mathbf{C}=\mathbf{0_n}\}$. We conclude that $\{\begin{bmatrix}\mathbf{A}\\ \mathbf{B}\end{bmatrix} \mid \mathbf{A}+\mathbf{B}-\mathbf{C}=\mathbf{0_n}\}$ is convex.

Moreover, using the convex combination already found, we also see that

$$\begin{aligned}[\theta\mathbf{A_1}+(1-\theta)\mathbf{A_2}]-[\theta\mathbf{A_1}+(1-\theta)\mathbf{A_2}]^{\mathbf{T}} &= \theta\mathbf{A_1}+(1-\theta)\mathbf{A_2}-\theta\mathbf{A_1^T}+(1-\theta)\mathbf{A_2^T}\\ &= \theta(\mathbf{A_1}-\mathbf{A_1^T})+(1-\theta)(\mathbf{A_2}-\mathbf{A_2^T})\\ &= \theta\mathbf{0}-(1-\theta)\mathbf{0}\\ &= \mathbf{0}.\end{aligned}$$

Hence, since $\begin{bmatrix}\theta\mathbf{A_1}+(1-\theta)\mathbf{A_2}\\ \theta\mathbf{B_1}+(1-\theta)\mathbf{B_2}\end{bmatrix} \in \{\begin{bmatrix}\mathbf{A}\\ \mathbf{B}\end{bmatrix} \mid \mathbf{A}-\mathbf{A^T}\}$, we conclude that $\{\begin{bmatrix}\mathbf{A}\\ \mathbf{B}\end{bmatrix} \mid \mathbf{A}-\mathbf{A^T}\}$ is convex. Using lemma E.1, we conclude that $\mathcal{C}_{\text{S-SLRMD}}$ is a convex set. □

Proposition 4.1: The following matrix derivative rules hold::

(a) $\partial_{\mathbf{A}} \text{tr}(\mathbf{ZA}) = \mathbf{Z^T}$

(b) $\partial_{\mathbf{A}} \text{tr}(\mathbf{A^T Z}) = \mathbf{Z}$.

(c) $\partial_{\mathbf{A}} \text{tr}(\mathbf{A^T A}) = 2\mathbf{A^T}$.

Proof. Using the definition of the trace, we see that

$$\begin{aligned}\text{tr}(\mathbf{ZA}) &= \text{tr}\left(\begin{pmatrix} \mathbf{z_1^T a_1} & \mathbf{z_1^T a_2} & ... & \mathbf{z_1^T a_n} \\ \mathbf{z_2^T a_1} & \mathbf{z_2^T a_2} & ... & \mathbf{z_2^T a_n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{z_n^T a_1} & \mathbf{z_n^T a_2} & ... & \mathbf{z_n^T a_n} \end{pmatrix}\right) \\ &= \mathbf{z_1^T a_1} + \mathbf{z_2^T a_2} + ... + \mathbf{z_n^T a_n} \\ &= \sum_{k=1}^{n} z_{1k} a_{k1} + \sum_{k=1}^{n} z_{2k} a_{k2} + ... + \sum_{k=1}^{n} z_{nk} a_{kn}.\end{aligned}$$

Hence, $\partial_{a_{ij}} \text{tr}(\mathbf{ZA}) = z_{ji}$. We conclude that $\partial_{\mathbf{A}} \text{tr}(\mathbf{ZA}) = \mathbf{Z^T}$. Moreover, since $\text{tr}(\mathbf{ZA}) = \text{tr}(\mathbf{AZ})$, we know that

$$\partial_{\mathbf{A}} \text{tr}(\mathbf{ZA}) = \partial_{\mathbf{A}} \text{tr}(\mathbf{AZ}) = \mathbf{Z^T}.$$

Similarily, we have that

$$\text{tr}(\mathbf{A^T Z}) = \mathbf{a_1 z_1} + ... + \mathbf{a_n z_n} = \sum_{k=1}^{n} a_{k1} z_{k1} + ... + \sum_{k=1}^{n} a_{kn} z_{kn},$$

and hence $\partial_{a_{ij}} \text{tr}(\mathbf{A^T Z}) = z_{ij}$. We conclude that

$$\partial_{\mathbf{A}} \text{tr}(\mathbf{A^T Z}) = \partial_{\mathbf{A}} \text{tr}(\mathbf{ZA^T}) = \mathbf{Z}.$$

Lastly, we see that

$$\begin{aligned}\text{tr}(\mathbf{A^T A}) &= \mathbf{a_1 a_1} + \mathbf{a_2 a_2} + ... + \mathbf{a_n a_n} \\ &= \sum_{i=1}^{n} x_{1i} x_{i1} + \sum_{i=1}^{n} x_{2i} x_{i2} + ... + \sum_{i=1}^{n} x_{ni} x_{in}.\end{aligned}$$

Hence, $\partial_{a_{ij}} \text{tr}(\mathbf{A^T A}) = a_{ji} + a_{ji} = 2a_{ji}$ and we have our derivative

$$\partial_{\mathbf{A}} \text{tr}(\mathbf{A^T A}) = \partial_{\mathbf{A}} \text{tr}(\mathbf{AA^T}) = 2\mathbf{A^T},$$

proving the last rule. □

Proposition 5.2: $\partial(||\mathbf{A}+\mathbf{L}\mathbf{R}^\mathbf{T}-\mathbf{C}||_\mathrm{F}^2) = 2\begin{bmatrix}\mathbf{A}+\mathbf{L}\mathbf{R}^\mathbf{T}-\mathbf{C}\\(\mathbf{A}+\mathbf{L}\mathbf{R}^\mathbf{T}-\mathbf{C})\mathbf{R}\\(\mathbf{A}+\mathbf{L}\mathbf{R}^\mathbf{T}-\mathbf{C})^\mathbf{T}\mathbf{L}\end{bmatrix}$

Proof. We will show $\partial_\mathbf{A}||\mathbf{A}+\mathbf{L}\mathbf{R}^\mathbf{T}-\mathbf{C}||_\mathrm{F}^2 = 2(\mathbf{A}+\mathbf{L}\mathbf{R}^\mathbf{T}-\mathbf{C})$, the other cases follow by a similar argument. Using the properties of the trace, we see that

$$\begin{aligned}||\mathbf{A}+\mathbf{L}\mathbf{R}^\mathbf{T}-\mathbf{C}||_F^2 &= \mathrm{tr}((\mathbf{A}+\mathbf{L}\mathbf{R}^\mathbf{T}-\mathbf{C})^\mathbf{T}(\mathbf{A}+\mathbf{L}\mathbf{R}^\mathbf{T}-\mathbf{C}))\\ &= \mathrm{tr}(\mathbf{A}^\mathbf{T}\mathbf{A}) + \mathrm{tr}(\mathbf{R}\mathbf{L}^\mathbf{T}\mathbf{A}) - \mathrm{tr}(\mathbf{C}^\mathbf{T}\mathbf{A}) + \mathrm{tr}(\mathbf{A}^\mathbf{T}\mathbf{L}\mathbf{R}^\mathbf{T}) + \mathrm{tr}(\mathbf{R}\mathbf{L}^\mathbf{T}\mathbf{L}\mathbf{R}^\mathbf{T})\\ &\quad - \mathrm{tr}(\mathbf{C}^\mathbf{T}\mathbf{L}\mathbf{R}^\mathbf{T}) - \mathrm{tr}(\mathbf{A}^\mathbf{T}\mathbf{C}) - \mathrm{tr}(\mathbf{R}\mathbf{L}^\mathbf{T}\mathbf{C}) + \mathrm{tr}(\mathbf{C}^\mathbf{T}\mathbf{C})\end{aligned}$$

Dropping all the terms independent of $\mathbf{A}$, we see that

$$\partial_\mathbf{A}||\mathbf{A}+\mathbf{L}\mathbf{R}^\mathbf{T}-\mathbf{C}||_\mathrm{F}^2 = \partial_\mathbf{A}(\mathrm{tr}(\mathbf{A}^\mathbf{T}\mathbf{A}) + \mathrm{tr}(\mathbf{R}\mathbf{L}^\mathbf{T}\mathbf{A}) - \mathrm{tr}(\mathbf{C}^\mathbf{T}\mathbf{A}) + \mathrm{tr}(\mathbf{A}^\mathbf{T}\mathbf{L}\mathbf{R}^\mathbf{T}) - \mathrm{tr}(\mathbf{A}^\mathbf{T}\mathbf{C})),$$

which using lemma 4.1 we see gives

$$2\mathbf{A}+\mathbf{L}\mathbf{R}^\mathbf{T}-\mathbf{C}+\mathbf{L}\mathbf{R}^\mathbf{T}-\mathbf{C} = 2(\mathbf{A}+\mathbf{L}\mathbf{R}^\mathbf{T}-\mathbf{C}),$$

which completes the proof. □

Proposition 5.3:

$$\nabla_{\mathbf{A},\mathbf{L},\mathbf{R}}\mathcal{L}_{\mathcal{A}} = \mathbf{0} \iff \begin{cases} \mathbf{A} = \mathcal{S}_{\frac{\gamma}{\xi}}(\mathbf{C} - \mathbf{L}\mathbf{R}^{\mathbf{T}} + \frac{\mathbf{\Lambda}+\mathbf{K}-\mathbf{K}^{\mathbf{T}}}{\xi}) \\ \mathbf{L} = (-\mathbf{\Lambda} - \xi(\mathbf{A}-\mathbf{C}))\mathbf{R}(\mathbf{I}+\xi\mathbf{R}^{\mathbf{T}}\mathbf{R}))^{-\mathbf{1}} \\ \mathbf{R} = (-\mathbf{\Lambda} - \xi(\mathbf{A}-\mathbf{C}))^{\mathbf{T}}\mathbf{L}(\mathbf{I}+\xi\mathbf{L}^{\mathbf{T}}\mathbf{L}))^{-\mathbf{1}} \end{cases},$$

Proof. We will show that

$$\partial_{\mathbf{R}}\mathcal{L}_{\mathcal{A}} = \mathbf{0} \iff \mathbf{R} = (-\mathbf{\Lambda} - \xi(\mathbf{A}-\mathbf{C}))^{\mathbf{T}}\mathbf{L}(\mathbf{I}+\xi\mathbf{L}^{\mathbf{T}}\mathbf{L}))^{-\mathbf{1}},$$

and

$$\partial_{\mathbf{A}}\mathcal{L}_{\mathcal{A}} = \mathbf{0} \iff \mathbf{A} = \mathcal{S}_{\frac{\gamma}{\lambda}}(\mathbf{C} - \mathbf{L}\mathbf{R}^{\mathbf{T}} + \frac{\mathbf{\Lambda}}{\xi}).$$

The argument that

$$\partial_{\mathbf{L}}\mathcal{L}_{\mathcal{A}} = \mathbf{0} \iff \mathbf{L} = (-\mathbf{\Lambda} - \xi(\mathbf{A}-\mathbf{C}))\mathbf{R}(\mathbf{I}+\xi\mathbf{R}^{\mathbf{T}}\mathbf{R}))^{-\mathbf{1}}.$$

is similar to the former.

We recall that the optimality conditions for $\partial\mathcal{L}_{\mathcal{A}}$ are given by

$$\partial\mathcal{L}_{\mathcal{A}}(\mathbf{A},\mathbf{L},\mathbf{R}) = \mathbf{0} \iff \begin{cases} \mathbf{0} \in \gamma\partial_{\mathbf{A}}||\mathbf{A}||_1 + \mathbf{\Lambda} + \mathbf{K} - \mathbf{K}^{\mathbf{T}} + \xi(\mathbf{A} + \mathbf{L}\mathbf{R}^{\mathbf{T}} - \mathbf{C}) \\ \mathbf{0} = \mathbf{L}^{\mathbf{T}} + \mathbf{\Lambda}\mathbf{R} + \xi(\mathbf{A} + \mathbf{L}\mathbf{R}^{\mathbf{T}} - \mathbf{C})\mathbf{R} \\ \mathbf{0} = \mathbf{R} + \mathbf{\Lambda}^{\mathbf{T}}\mathbf{L} + \xi(\mathbf{A} + \mathbf{L}\mathbf{R}^{\mathbf{T}} - \mathbf{C})^{\mathbf{T}}\mathbf{L} \end{cases} \tag{41}$$

Using this conditions, we know that $\partial_{\mathbf{R}}\mathcal{L}_{\mathcal{A}} = \mathbf{0}$ if and only if

$$\begin{aligned} \mathbf{0} &= \mathbf{R} + \mathbf{\Lambda}^{\mathbf{T}}\mathbf{L} + \xi(\mathbf{A} + \mathbf{L}\mathbf{R}^{\mathbf{T}} - \mathbf{C})^{\mathbf{T}}\mathbf{L} \\ &= \mathbf{R} + \mathbf{\Lambda}^{\mathbf{T}}\mathbf{L} + \xi\mathbf{A}^{\mathbf{T}}\mathbf{L} + \xi\mathbf{R}\mathbf{L}^{\mathbf{T}}\mathbf{L} - \xi\mathbf{C}^{\mathbf{T}}\mathbf{L} \end{aligned}$$

Taking all the terms containing $\mathbf{R}$ one side, we see that

$$\begin{aligned} \mathbf{R} + \xi\mathbf{R}\mathbf{L}^{\mathbf{T}}\mathbf{L} &= -\mathbf{\Lambda}^{\mathbf{T}}\mathbf{L} - \xi\mathbf{A}^{\mathbf{T}}\mathbf{L} + \xi\mathbf{C}^{\mathbf{T}}\mathbf{L} \\ \mathbf{R}(\mathbf{I} + \xi\mathbf{L}^{\mathbf{T}}\mathbf{L}) &= (-\mathbf{\Lambda} - \xi(\mathbf{A}-\mathbf{C}))^{\mathbf{T}}\mathbf{L} \\ \mathbf{R} &= (-\mathbf{\Lambda} - \xi(\mathbf{A}-\mathbf{C}))^{\mathbf{T}}\mathbf{L}(\mathbf{I}+\xi\mathbf{L}^{\mathbf{T}}\mathbf{L}))^{-\mathbf{1}}, \end{aligned}$$

which proves the first equivalence.

The prove the second equivalence, we again use (35), implying that $\partial_{\mathbf{A}}\mathcal{L}_{\mathcal{A}} = \mathbf{0_n}$ if and only if

$$\mathbf{0} \in \gamma\partial_{\mathbf{A}}||\mathbf{A}||_1 + \mathbf{\Lambda} + \mathbf{K} - \mathbf{K}^{\mathbf{T}} + \xi(\mathbf{A} + \mathbf{L}\mathbf{R}^{\mathbf{T}} - \mathbf{C}).$$

Taking out a factor of γ, we see that $\partial_{\mathbf{A}}\mathcal{L}_{\mathcal{A}} = \mathbf{0_n}$ if an only if

$$\mathbf{0} \in \frac{\gamma}{\xi}\partial_{\mathbf{A}}||\mathbf{A}||_1 + \frac{\mathbf{\Lambda} + \mathbf{K} - \mathbf{K}^{\mathbf{T}}}{\xi} + (\mathbf{A} + \mathbf{L}\mathbf{R}^{\mathbf{T}} - \mathbf{C}).$$

Hence, we see that $\partial_{\mathbf{A}}\mathcal{L}_{\mathcal{A}} = \mathbf{0}$ if and only if $\mathbf{A}$ is the matrix to minimize

$$\mathbf{A} = \arg\min_{\mathbf{A}} \frac{\gamma}{\xi}||\mathbf{A}||_1 + \frac{1}{2}(||\mathbf{A} + \mathbf{L}\mathbf{R}^{\mathbf{T}} - \mathbf{C} - \frac{\mathbf{\Lambda} + \mathbf{K} - \mathbf{K}^{\mathbf{T}}}{\xi}||_{\mathrm{F}}^2). \tag{42}$$

Using a result by [9], we know that the following two are equivalent:

$$\mathbf{A} = \arg\min \alpha||\mathbf{A}||_1 + \frac{1}{2}(||\mathbf{A} - \mathbf{\Psi}||_{\mathrm{F}}^2) \iff \mathbf{A} = \mathcal{S}_{\alpha}(\mathbf{\Psi}), \tag{43}$$

where $\mathcal{S}_\alpha(\mathbf{M})$ is the soft thresholding function, replacing each element $m_{ij} \in \mathbf{M}$ with the maximum of $|m_{ij}| - \alpha$ and zero, i.e.

$$[\mathcal{S}_\alpha(\mathbf{M})]_{ij} := \max(|[\mathbf{M}]_{ij}| - \alpha, 0). \tag{44}$$

Rewriting (42) in the form of (43) gives

$$\mathbf{A} = \arg\min \frac{\gamma}{\xi}||\mathbf{A}||_1 + \frac{1}{2}(||\mathbf{A} - (\mathbf{C} - \mathbf{L}\mathbf{R}^\mathbf{T} + \frac{\mathbf{\Lambda} + \mathbf{K} - \mathbf{K}^\mathbf{T}}{\xi})||_\mathrm{F}^2),$$

and hence $\mathbf{A}$ is minimized when

$$\mathbf{A} = \mathcal{S}_{\frac{\gamma}{\xi}}(\mathbf{C} - \mathbf{L}\mathbf{R}^\mathbf{T} + \frac{\mathbf{\Lambda} + \mathbf{K} - \mathbf{K}^\mathbf{T}}{\xi}),$$

proving the second equivalence. □

Proposition 6.4:

$$\nabla_{\mathbf{A},\mathbf{L},\mathbf{R}}\mathcal{L}_{\mathcal{A}} = \mathbf{0} \iff \begin{cases} \mathbf{A} = \mathcal{S}_{\frac{\gamma}{\lambda}}(\mathbf{C} - \mathbf{L}\mathbf{R}^{\mathbf{T}} + \frac{\mathbf{\Lambda}}{\xi}) \\ \mathbf{L} = [-\mathbf{M} - (\mathbf{\Lambda} + \xi(\mathbf{A} - \mathbf{C} + \mathbf{I}))\mathbf{R}](\mathbf{I} + \xi(\mathbf{R}^{\mathbf{T}}\mathbf{R} - \mathbf{I}))^{-\mathbf{1}} \\ \mathbf{R} = [\mathbf{M} - (\mathbf{\Lambda}^{\mathbf{T}} + \xi(\mathbf{A} - \mathbf{C} - \mathbf{I})^{\mathbf{T}})\mathbf{L}](\mathbf{I} + \xi(\mathbf{L}^{\mathbf{T}}\mathbf{L} + \mathbf{I}))^{-\mathbf{1}}. \end{cases}.$$

Proof. The $\partial_{\mathbf{A}}$ biconditional is almost unchanged, only dropping the terms that used to ensure symmetry (i.e. the terms containing $\mathbf{K}$). We will show that

$$\partial_{\mathbf{R}}\mathcal{L}_{\mathcal{A}} = 0 \iff \mathbf{R} = [\mathbf{M} - (\mathbf{\Lambda}^{\mathbf{T}} + \xi(\mathbf{A} - \mathbf{C} - \mathbf{I})^{\mathbf{T}})\mathbf{L}](\mathbf{I} + \xi(\mathbf{L}^{\mathbf{T}}\mathbf{L} + \mathbf{I}))^{-\mathbf{1}}.$$

The argument that

$$\partial_{\mathbf{L}}\mathcal{L}_{\mathcal{A}} = 0 \iff \mathbf{L} = [-\mathbf{M} - (\mathbf{\Lambda} + \xi(\mathbf{A} - \mathbf{C} + \mathbf{I}))\mathbf{R}](\mathbf{I} + \xi(\mathbf{R}^{\mathbf{T}}\mathbf{R} - \mathbf{I}))^{-\mathbf{1}}$$

is again similar to the case we show.

We recall that our optimality conditions were given by

$$\partial\mathcal{L}_{\mathcal{A}}(\mathbf{A}, \mathbf{L}, \mathbf{R}) = \mathbf{0} \iff \begin{cases} \mathbf{0} \in \gamma\partial_{\mathbf{A}}||\mathbf{A}||_1 + \mathbf{\Lambda} + \xi(\mathbf{A} + \mathbf{L}\mathbf{R}^{\mathbf{T}} - \mathbf{C}) \\ \mathbf{0} = \mathbf{L}^{\mathbf{T}} + \mathbf{\Lambda}\mathbf{R} + \mathbf{M} + \xi[(\mathbf{A} + \mathbf{L}\mathbf{R}^{\mathbf{T}} - \mathbf{C})\mathbf{R} + \mathbf{L} - \mathbf{R}] \\ \mathbf{0} = \mathbf{R} + \mathbf{\Lambda}^{\mathbf{T}}\mathbf{L} - \mathbf{M} + \xi[(\mathbf{A} + \mathbf{L}\mathbf{R}^{\mathbf{T}} - \mathbf{C})^{\mathbf{T}}\mathbf{L} + \mathbf{R} - \mathbf{L}] \end{cases} \tag{45}$$

We therefore know that $\partial_{\mathbf{A}} = 0$ if and only if

$$\mathbf{0} = \mathbf{R} + \mathbf{\Lambda}^{\mathbf{T}}\mathbf{L} - \mathbf{M} + \xi[(\mathbf{A} + \mathbf{L}\mathbf{R}^{\mathbf{T}} - \mathbf{C})^{\mathbf{T}}\mathbf{L} + \mathbf{R} - \mathbf{L}].$$

Hence, we see that

$$\begin{aligned} \mathbf{0} &= \mathbf{R} + \mathbf{\Lambda}^{\mathbf{T}}\mathbf{L} - \mathbf{M} + \xi[(\mathbf{A} + \mathbf{L}\mathbf{R}^{\mathbf{T}} - \mathbf{C})^{\mathbf{T}}\mathbf{L} + \mathbf{R} - \mathbf{L}] \\ &= \mathbf{R} + \mathbf{\Lambda}^{\mathbf{T}}\mathbf{L} - \mathbf{M} + \xi((\mathbf{A} + \mathbf{L}\mathbf{R}^{\mathbf{T}} - \mathbf{C})^{\mathbf{T}}\mathbf{L} + \mathbf{R} - \mathbf{L}) \\ &= \mathbf{R} + \mathbf{\Lambda}^{\mathbf{T}}\mathbf{L} - \mathbf{M} + \xi\mathbf{A}^{\mathbf{T}}\mathbf{L} + \xi\mathbf{R}\mathbf{L}^{\mathbf{T}}\mathbf{L} - \xi\mathbf{C}^{\mathbf{T}}\mathbf{L} + \xi\mathbf{R} - \xi\mathbf{L}. \end{aligned}$$

Isolating $\mathbf{R}$ gives us

$$\begin{aligned} \mathbf{R} + \xi\mathbf{R}\mathbf{L}^{\mathbf{T}}\mathbf{L} + \xi\mathbf{R} &= -\mathbf{\Lambda}^{\mathbf{T}}\mathbf{L} + \mathbf{M} - \xi\mathbf{A}^{\mathbf{T}}\mathbf{L} + \xi\mathbf{C}^{\mathbf{T}}\mathbf{L} + \xi\mathbf{L} \\ \mathbf{R}(\mathbf{I} + \xi(\mathbf{L}^{\mathbf{T}}\mathbf{L} + \mathbf{I})) &= \mathbf{M} - (\mathbf{\Lambda}^{\mathbf{T}} + \xi(\mathbf{A}^{\mathbf{T}} - \mathbf{C}^{\mathbf{T}} - \mathbf{I}))\mathbf{L} \\ &= \mathbf{M} - (\mathbf{\Lambda}^{\mathbf{T}} + \xi(\mathbf{A} - \mathbf{C} - \mathbf{I})^{\mathbf{T}})\mathbf{L}, \end{aligned}$$

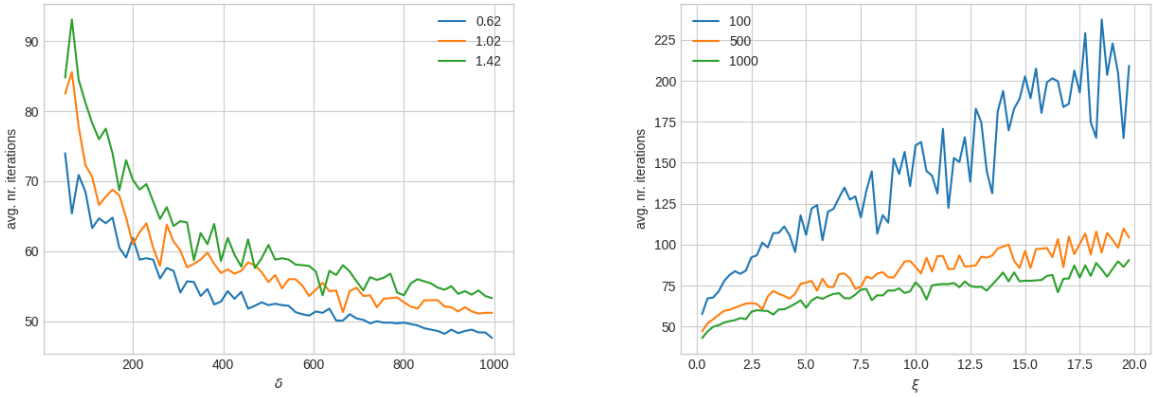
and hence we conclude that

$$\mathbf{R} = [\mathbf{M} - (\mathbf{\Lambda}^{\mathbf{T}} + \xi(\mathbf{A} - \mathbf{C} - \mathbf{I})^{\mathbf{T}})\mathbf{L}](\mathbf{I} + \xi(\mathbf{L}^{\mathbf{T}}\mathbf{L} + \mathbf{I}))^{-\mathbf{1}},$$

which is what we wanted to show. □

F Figures

F.1 ADM-S Parameters

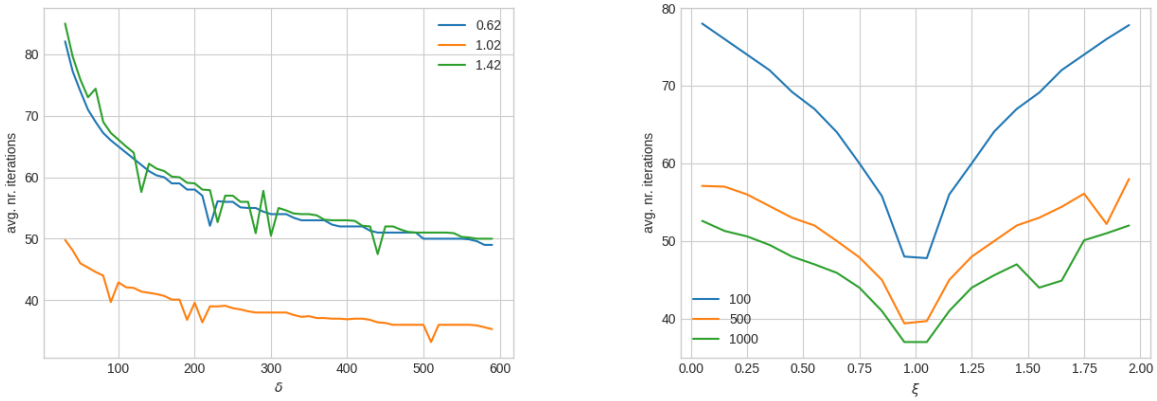


(a) Effect of choice of δ on number of iterations (for $\xi = 0.62, \xi = 1.02, \xi = 1.42$).

(b) Effect of choice of ξ on number of iterations (for $\delta = 100, \delta = 500, \delta = 1000$).

Figure 5: Alternating Directions Method (S-SLRMD)

F.2 ADM-PSD Parameters



(a) Effect of choice of δ on number of iterations (for $\xi = 0.62, \xi = 1.02, \xi = 1.42$).

(b) Effect of choice of ξ on number of iterations (for $\delta = 100, \delta = 500, \delta = 1000$).

Figure 6: Alternating Directions Method (PSD-SLRMD)