# Learning and clustering graphs from high dimensional data

Doctoral Dissertation submitted to the

Faculty of Informatics of the Università della Svizzera Italiana

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

presented by

Dimosthenis Pasadakis

under the supervision of

Prof. Olaf Schenk

February 2023

Dissertation Committee

| | |
|---|---|
| **Prof. Illia Horenko** | Università della Svizzera italiana, Switzerland |
| **Prof. Stefan Wolf** | Università della Svizzera italiana, Switzerland |
| **Prof. Theodoros Damoulas** | University of Warwick, United Kingdom |
| **Dr. Albert-Jan Yzelman** | Huawei Zürich Research Lab, Switzerland |
| **Prof. Inderjit Dhillon** | University of Texas at Austin, USA |

Dissertation accepted on 21 February 2023

[illegible]

Research Advisor

**Prof. Olaf Schenk**

PhD Program Director

**Prof. Walter Binder, Prof. Stefan Wolf**

I certify that except where due acknowledgement has been given, the work presented in this thesis is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; and the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program.

[illegible]

Dimosthenis Pasadakis
Lugano, 21 February 2023

# Abstract

Estimating the graphical structures of high dimensional data and identifying the presence of clusters in them are ubiquitous tasks in every scientific domain that deals with interacting or interconnected variables. We participate in the advance of these research fields with efficient and accurate algorithms that learn and cluster graphs. Initially, we contribute in the development of a performant precision matrix estimation routine based on the sparse quadratic approximation of the $\ell_1$ regularized Gaussian maximum likelihood method. The proposed method exploits the presence of block structure in the underlying computations, and is suitable for datasets characterized by reduced sparsity. Motivated by its effectiveness in high dimensional problems, we extend the capabilities of this method to the retrieval of graphs of only non-negatively correlated variables, and introduce two algorithms for sparse $M$-matrix estimation. The first one is based on consecutive precision matrix estimations, while the second one performs constrained optimization for the retrieval of the final graphical structure. Finally, we present a nonlinear reformulation of direct multiway spectral clustering that is formulated as an unconstrained minimization problem. Our method promotes sharp indicator vectors that correspond to optimal graph cuts and improved clustering assignments. The advantages of all introduced algorithms are showcased in a series of comparative tests with the current state-of-the-art on artificial datasets, and their real-world applicability is demonstrated with numerical experiments on biological, medical, and image data.

# Acknowledgements

I consider this thesis as the conclusion of an enriching and enjoyable chapter. The people mentioned here have played a key role in both these aspects.

Professor Olaf Schenk, you have simultaneously treated me as a colleague since early in our academic interaction, and supported me as a student personally and professionally. This has liberated me in a multitude of ways, and has allowed me to work and produce with joy. Professor Matthias Bollhöfer, I feel fortunate to have collaborated with you. You have expanded my scientific interests, and made me more competent along the way. Dr. Drosos Kourounis, you have defined the way I think in science. They say life is composed of distinct turning points, and maybe that's a platitude. Coming to Lugano, following your suggestion, certainly feels like such a moment for me. Professor Gerhard Wellein and Dr. Albert-Jan Yzelman the opportunities you have granted me to work with you and your groups have been defining moments in my career so far. Professor Stefan Wolf, you have fostered an environment of living and thinking that has marked these past years for me, and for many people that I care for. Aryan, your tenacity in work, among other of your unique traits, is an inspiration to me, and Christie working and spending time with you is such a pleasure. Professor Theodoros Damoulas, Professor Illia Horenko, and Professor Inderjit Dhillon, I deeply appreciate your acceptance to be part of my committee, and the time and effort you have devoted in reviewing my work.

To the friends that live far, but feel close, you've made every transition smoother and easier. To the ones that live close, and share their everyday lives with me, these years have shaped us. To my family, that made every step of the way possible, we are what we have always been - the greatest team. For my partner, that just makes everything so much better, I have to come up with new ingenious ways to express all my appreciation. And to our cat, that is our leader at home, and constantly reminds us what comfort means, you get a treat tonight.

Thank you all.

# Contents

# Chapter 1

# Introduction

Representing data in the form of graphs and discovering groups of interest in them are problems with application fields that span many disciplines. Graphs are fundamental mathematical entities with nodes (or vertices) and edges connecting them. The relationship between two connected nodes is usually captured by the scalar weight value of the edge that links them. In many domains, data is commonly available in the form of an unstructured list of samples or variables, with no available relational information among them. The construction of the latent graphical structure of such a dataset often offers an intuitive representation, and, subsequently, the identification of clusters within this structure further enhances one's insight on it. Due to the ever-growing size of data than can be stored and analyzed, performing learning and clustering tasks in high-dimensional scenarios, where the number of dimensions surpass the number of available observations, is of critical importance.

## 1.1 Estimating graphical structures

Undirected weighted graphs, with edges representing the conditional dependence among the variables, are typically constructed with a Gaussian graphical modeling (GGM) approach [1]. In this context, each vertex corresponds to a variable, with edges being present between the vertices only if they are conditionally dependent. These dependencies among the data points can be both positive and negative, and are encoded in a matrix that represents the graph, whose non-zero entries correspond to the dependencies between two variables. This matrix is the inverse of the covariance matrix, also known as the precision matrix, and encodes the positive definite graphical structure of a Gaussian Markov random field (GMRF) [2]. A common prior imposed on the estimation of the precision matrix is that the conditional correlations among the random variables are sparse [3], i.e., that there is a limited number of conditional correlations between the variables. This prior corresponds to imposing a degree of sparsity on the estimated precision matrix, and a widely used approach for this is the $\ell_1$ regularized maximum likelihood estimation (MLE), commonly referred to as the "graphical LASSO" [4]. This method minimizes the negative log-likelihood of the data in question, and considers an $\ell_1$ regularization term that enforces the much desired sparsity property in the estimates. A popular solution approach for this problem is to quadratically approximate this objective, and solve the resulting series of minimization problems with a proximal Newton method [5, 6]. This technique enjoys superlinear convergence rates, and, as an attribute of the regularization that is employed, successfully reduces the size of the variables that will be updated. However, for high dimensional datasets the estimation of precision matrices continues to pose a computational challenge, particularly in cases that exhibit reduced sparsity in their estimates. The Sparse QUadratic Inverse Covariance matrix estimation (SQUIC) algorithm [7, 8] continues the progress on large-scale, second-order methods by exploiting the inherent sparsity in the underlying linear algebra operations. This thesis contributes to this field of research with block-oriented algorithmic routines that further accelerate the retrieval of precision matrices, enhance the performance of SQUIC, and enable its applicability on a wide range of real-world problems which exhibit limited sparsity [9].

Graphical models under the constraint that all partial correlations are non-negative is an important subclass of GGMs. The problem of finding variables that are non-negatively correlated corresponds to enforcing an $M$-matrix structure on the precision matrix [10, 11]. Therefore, MLE methods are faced with the additional challenge that the optimization problem is now constrained, and aims to retrieve positive definite and symmetric precision matrices restricted to non-positive off-diagonal elements. This problem has attracted a lot of attention, and is featured in recent graph learning surveys [12, 13]. However, the applicability of the proposed approaches is limited for large-scale and real-world data. We tackle these limitations, and extend the concept of quadratically approximating the $\ell_1$ regularized MLE minimization problem to the estimation of $M$-matrices. In [14] we introduce two algorithms that learn high-dimensional graphs of only non-negatively correlated random variables, one based on consecutive precision matrix estimations, and one based on a constrained optimization approach.

## 1.2 Clustering graphical clustures

Among a plethora of applications, $M$-matrices are commonly used in partitioning applications [15, 16], and their spectrum is utilized in graph clustering tasks [17, 18]. Spectral clustering is a popular graph-based method due to the simplicity of its implementation, the reasonable computation time, and the fact that it overcomes the NP-hardness of other graph-theoretic approaches by solving a relaxed optimization problem in polynomial time. Its idea is based on the eigendecomposition of matrices that describe the connectivity of a graph [19]. Reformulating the spectral method from the traditional 2-norm to the $p$-norm, for $p \in (1,2]$, has proven to lead to a sharp approximation of balanced cut metrics and improved clustering assignments [20]. Additionally, these nonlinear reformulations result in a tight relaxation of the spectral clustering problem, with the resulting solutions approximating closely the solution of the original discrete problem [21]. The graph cut theoretically converges to the optimal Cheeger cut [22] for $p \to 1$, thus highlighting the superiority of $p$-spectral methods over their traditional 2-norm counterparts [23]. The resulting indicator functions exhibit non-smooth jumps between nodes belonging to different groups of a graph, and are thus particularly well suited for clustering applications [24]. We contribute in this research field by recasting this problem as an unconstrained optimization procedure over the Grassmann manifold [25], and by proposing a simple algorithm for direct multiway (or $k$-way) $p$-spectral clustering that effectively minimizes graph cuts [26].

## 1.3 Outline and notation

This document is composed of two parts; in Part I we discuss the problem of estimating graphical structures from data through MLE methods, and in Part II we present research related to the nonlinear spectral clustering of graphs. In particular, we begin in Chapter 2 with offering a brief background on the estimation of precision and $M$-matrices, and proceed in Chapter 3 with introducing the sparse quadratic approximate methods that lead to their efficient and accurate retrieval. Numerical experiments that showcase the capabilities of the proposed graph learning methods are offered in Chapter 4. Then, in Chapter 5 we revisit the problem of performing spectral and $p$-spectral clustering, and in Chapter 6 we introduce our algorithm for the direct clustering of multiple eigenvectors of the graph $p$-Laplacian, a nonlinear reformulation of the traditional graph Laplacian matrix. The numerical results that support the efficacy of this method are offered in Chapter 7. The author's contributions in the aforementioned areas have additionally been published in [9, 26, 27], and are under review in [14, 28].

In what follows, we denote scalar quantities with lowercase, vectors with lowercase bold, and matrices with uppercase bold characters. The $(i, j)$th entry of a matrix $\mathbf{A}$ is symbolized by $\mathbf{A}_{ij}$ and all entries in row $i$ or column $j$ by $\mathbf{A}_{i:}$ and $\mathbf{A}_{:j}$, respectively. The $i$th element of a vector $\mathbf{v}$ is denoted by $v_i$. The $i$th column vector of a matrix $\mathbf{V}$ is denoted by either $\mathbf{v}_i$, or $\mathbf{v}^i$. The latter is used in case that the subscript is occupied by the element index number or the norm of the vector. For example, when comparing the $i$th eigenvector computed in the 2 and the $p$-norm, we denote them as $\mathbf{v}_2^i$ and $\mathbf{v}_p^i$ respectively. Sets are denoted by capital calligraphic characters, for example, $\mathscr{A}$, the identity matrix as $\mathbf{I}$ and the vector of all ones as $\mathbf{e}$.

# Part I

# Sparse graph learning

# Chapter 2

# Graphical model estimation

Graphical models are a statistical framework widely used for estimating the graphical structure of high dimensional data. Any collection $\mathbf{y} = \left(y_1, y_2, \ldots, y_p\right)$ of random variables can be associated with an underlying graph $\mathcal{G}(V, E)$, characterized by its node set $V = \{1, 2, \ldots, p\}$ and edge set $E = V \times V$. The properties of $\mathcal{G}$ dictate the conditional independence of the variables in $\mathbf{y}$. Typically, this latent graphical structure is unknown a priori; hence, it is desirable to estimate the graph's edge set using the available samples. Applications of this problem abound in scientific fields, including computer vision [29], social science [30], finance [31], and biology [32], where it may be desirable to infer the connectivity between individual pixels, people, financial institutions, or genes. Our focus is on undirected graphical models, also known as Markov random fields, for which no direction exists in the edges of $\mathcal{G}$ and there is no distinction between an edge specified by the node pairs $(i, j)$ and $(j, i) \in E$. This naturally leads to symmetric matrices that capture the connectivity of the underlying graph. Additionally, we are interested in scenarios commonly encountered in real-world datasets, where the number of nodes $p$, or dimensions of the problem, far exceed the number of available samples $n$, or observations.

In this chapter we do not intend to review all the fundamental properties of graphical models. For that purpose [33] and [34] serve as excellent sources. We hereby confine ourselves to the most relevant topics, and begin in Section 2.1 by outlining the factorization and conditional independence properties of a graph. Subsequently, in Section 2.2 we narrow our focus on graphs emerging from distributions assumed to be Gaussian and discuss the retrieval of precision matrices that capture the underlying structure and conditional dependence between the random variables. Then, in Section 2.3 we recap a popular approach that induces the desirable property of sparsity in precision matrices by regularizing the estimates. In Section 2.4 the problem of retrieving $M$-matrices, with non-zero entries that are positively correlated, is outlined, and, last, in Section 2.5 we present the landscape of existing solution methods for Gaussian graphical model estimation.

## 2.1 Factorization and conditional independency

The probabilistic structure of a random vector can be related to the structure of a graph through its factorization and conditional independence properties, which are based on the existence of cliques and the node cut set of the graph, respectively. A clique $C \subseteq V$ of a graph $\mathcal{G}$ is defined as a fully connected subgraph of its node set $V$, i.e., for an edge connecting two vertices $i, j$ we have that $(i, j) \in E \ \forall \ i, j \ \in C$. Maximal cliques are fully connected subgraphs that are not contained within any other clique. The clique set $\mathcal{C}$ of a graph may be used to characterize the probability distribution, or density function, of the $p$-dimensional random vector $\mathbf{y} = (y_1, y_2, \ldots, y_p)$. For each given clique $C \subset \mathcal{C}$ a real valued positive function $\psi_C$ of the vector $\mathbf{y}_C = (y_i, i \in C)$ is associated with it. Then, a probability distribution $P$ factorizes over the graph if

$$P(y_1, y_2, \ldots, y_p) = \frac{1}{Z} \prod_{c \in \mathcal{C}} \psi_C(\mathbf{y}_C), \tag{2.1.1}$$

with $Z$ being the function given by the sum

$$Z = \sum_{y_1, y_2, \ldots, y_p} \prod_{C \in \mathcal{C}} \psi_C(\mathbf{y}_C). \tag{2.1.2}$$

We consider as an illustrative example the clique structure of the graph in Figure 2.1a. Any probability distribution that factorizes over it is of the form

$$\begin{aligned} P(y_1, \ldots, y_9) = \frac{1}{Z}\, & \psi_{1234}(y_1, y_2, y_3, y_4)\, \psi_{457}(y_4, y_5, y_7) \\ & \psi_{56}(y_5, y_6)\, \psi_{789}(y_7, y_8, y_9). \end{aligned} \tag{2.1.3}$$

In the following subsection 2.2 we will see the form that such probability distributions attain when the underlying data is assumed to be Gaussian.

A second way in which graphical structure can be related to the probabilistic properties of a random vector is via conditional independency. For undirected graphs, this property can be specified in terms of the node cut sets of the graph. A vertex subset $S$ is the collection of entries of $V$ that, if removed, separates the

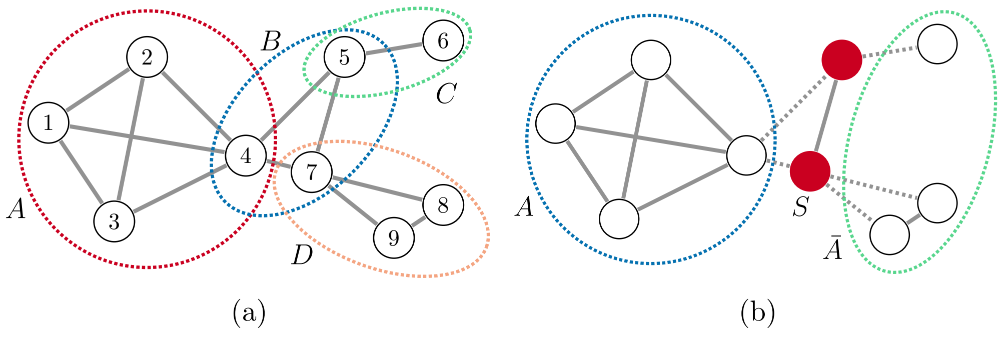


Figure 2.1: *Illustration of the ways a graphical structure can be related to the probabilistic properties of a random vector. (a) Maximal clique structure of a graph with 9 nodes. (b) Node cut set of graph, that separates it into two disconnected components A and $\bar{A}$. Based on concepts from [33].*

graph into two or more disconnected subsets. Looking again into this relationship illustratively, removing the subset $S$ in Figure 2.1b, comprising of nodes 5 and 7, splits the graph into two disconnected components, $A$ with nodes $1,2,3,4$ and its complement $\bar{A}$ with nodes $6,8,9$. In this example, $A$ is conditionally independent from $\bar{A}$, with this relationship denoted as $\perp\!\!\!\perp$. The random vector $\mathbf{y}$ exhibits the Markov property with respect to the graph $\mathcal{G}$ as

$$\mathbf{y}_A \perp\!\!\!\perp \mathbf{y}_{\bar{A}} | \, \mathbf{y}_S \;\; \forall \text{ vertex cut sets } S \subset V. \tag{2.1.4}$$

This property has a particular interpretation for chain graphs, which capture the graphical structure of a Markov chain process [33]. Their edge set is

$$E = \{(1,2),(2,3),\ldots,(p-1,p)\}, \tag{2.1.5}$$

and any vertex $i \in \{2,3,\ldots,p-1\}$ forms a cut set separating the graph into past $A = \{1,\ldots,i-1\}$ and future nodes $\bar{A} = \{i+1,\ldots,p\}$. The conditional independence relation then reads

$$\underbrace{\mathbf{y}_A = (y_1,y_2,\ldots,y_{i-1})}_{\text{Past}} \perp\!\!\!\perp \underbrace{\mathbf{y}_{\bar{A}} = (y_{i+1},y_{i+2},\ldots,y_p)}_{\text{Future}} \mid \underbrace{\; y_i \;}_{\text{Present}}, \tag{2.1.6}$$

and translates to the fact that the past and the future of a Markov chain process are conditionally independent given the present [35].

These two probabilistic properties, the factorization and the Markov property, are equivalent for undirected graphical models as demonstrated in the Hammersley-Clifford theorem [36]. This critical equivalence states that positive distributions

$P(\mathbf{y})$ factorize the graph $\mathcal{G}$ if and only if the random vector $\mathbf{y}$ exhibits the Markov property on the graph.

## 2.2 Gaussian graphical models

Gaining insights into complex continuous phenomena requires characterizing the relationships within multivariate distributions. In many domains, it is considered that the underlying data is Gaussian. This hypothesis is primarily based on the central limit theorem [37], and on the fact that the normal distribution arises from maximizing the entropy across all real valued distributions with specified mean and variance [38]. Hence, estimating the mean, the covariance matrix and it's inverse of Gaussian distributions are ubiquitous tasks. The resulting graphical models capture the conditional dependencies between the random variables in the form of a network, which in turn is encoded in the entries of the inverse covariance, or precision, matrix.

In order to estimate such models we consider $\mathbf{Y} \in \mathbb{R}^{p \times n}$ to be $n$ data samples drawn by a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ in $p$ dimensions with mean $\boldsymbol{\mu} \in \mathbb{R}^p$ and covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{p \times p}$. The probability distribution is then a special form of 2.1.1 and is defined as

$$\begin{aligned} P_{\mu,\Sigma}(\mathbf{y}_i) &= \frac{1}{(2\pi)^{\frac{p}{2}} \det[\boldsymbol{\Sigma}]^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{y}_i - \boldsymbol{\mu})^\intercal \boldsymbol{\Sigma}^{-1}(\mathbf{y}_i - \boldsymbol{\mu})\right) \\ &= \frac{1}{(2\pi)^{\frac{p}{2}} \det[\boldsymbol{\Sigma}]^{\frac{1}{2}}} \exp\left(-\frac{1}{2} \operatorname{tr} \boldsymbol{\Sigma}^{-1}(\mathbf{y}_i - \boldsymbol{\mu})(\mathbf{y}_i - \boldsymbol{\mu})^\intercal\right), \end{aligned} \tag{2.2.1}$$

where we have used the trace operator to rearrange the quadratic part of the exponent [39]. In order for (2.2.1) to result in a well-defined probability function the covariance matrix $\boldsymbol{\Sigma}$ must be positive definite, i.e., for any $\mathbf{x} \in \mathbb{R}^p$ such that $\mathbf{x} \neq 0$ we must have that $\mathbf{x}^\intercal \boldsymbol{\Sigma} \mathbf{x} > 0$. Positive definite matrices are non-singular, and hence the determinant appearing in the denominator of (2.2.1) is nonzero. This property also ensures that the inverse of the covariance matrix will be positive definite itself. Considering a Gaussian distribution whose mean is the all-zero vector $\mathbf{0}$, and denoting the precision matrix as $\boldsymbol{\Theta} = \boldsymbol{\Sigma}^{-1}$ the rescaled negative log-likelihood function for the distribution (2.2.1) reads

$$\begin{aligned}\mathcal{L}(\boldsymbol{\Theta}) &= -\frac{1}{n}\sum_{i=1}^{n} \log P_{\mathbf{0},\boldsymbol{\Sigma}}(\mathbf{y}_i) \\ &= -\log\det\boldsymbol{\Theta} + \operatorname{tr}\left(\boldsymbol{\Sigma}^{-1}\frac{1}{n}\sum_{i=1}^{n}\mathbf{y}_i\mathbf{y}_i^{\mathsf{T}}\right) \\ &= -\log\det\boldsymbol{\Theta} + \operatorname{tr}(\mathbf{S}\boldsymbol{\Theta}), \end{aligned} \tag{2.2.2}$$

where $\mathbf{S}$ is the empirical, or sample, covariance matrix defined as

$$\mathbf{S} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{y}_i\mathbf{y}_i^{\mathsf{T}}, \tag{2.2.3}$$

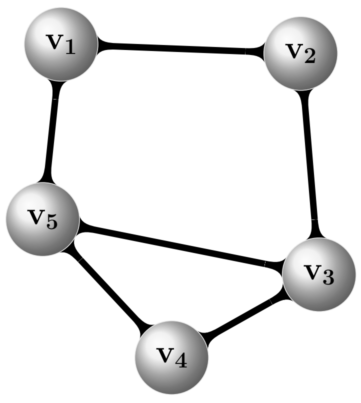
for zero mean distributions. The objective function (2.2.2) is strictly convex with a unique minimum, and its minimization defines the maximum likelihood estimator

$$\widehat{\boldsymbol{\Theta}}_{\mathrm{MLE}} = \arg\min_{\boldsymbol{\Theta}} \mathcal{L}(\boldsymbol{\Theta}), \tag{2.2.4}$$

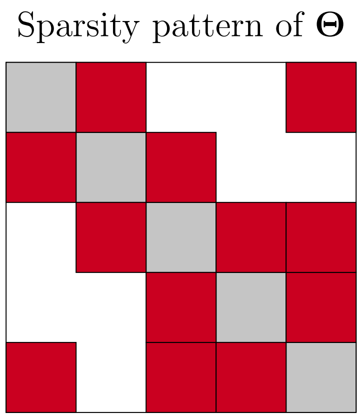
denoted also as MLE. The MLE converges to the true precision matrix $\widehat{\boldsymbol{\Theta}} = \mathbf{S}^{-1} = \boldsymbol{\Sigma}^{-1}$ when the sample size $n \to \infty$. It is a common phenomenon, however, in real-world problems to have a number of nodes $p$ of the same order of magnitude, or even larger than the number of available samples $n$. In these low sampling regimes we have that $\mathbf{S} \nsucc 0$ and thus the empirical covariance matrix does not accurately model the true covariance. Moreover, $\mathbf{S}$ cannot be inverted in order to obtain an estimate of the precision matrix, thus the MLE problem is not well defined. Additionally, the estimate $\widehat{\boldsymbol{\Theta}}$ can be ill-conditioned even when $n \geq p$. Therefore, constrained or regularized variants of the MLE method must be considered.

### 2.2.1 Sparsity considerations

Regardless of the ratio between the available samples and the dimensionality of the MLE problem, enforcing sparsity in the retrieved precision matrix has additional advantages. In the context of undirected Gaussian graphical models, the zero entries entries in the precision matrix indicate conditional independence between the random variables. In particular, a collection of random variables $\mathbf{Y}$ with an associated precision matrix $\boldsymbol{\Theta}$ is a Gaussian Markov Random Field (GMRF) with respect to the underlying graph $\mathcal{G}(\boldsymbol{\Theta})$. The graph's edge structure



(a)



(b)

Figure 2.2: *Illustration of the ways a graphical structure can be related to the probabilistic properties of a random vector. (a) An undirected graph $\mathcal{G}$ with 5 vertices and 6 edges. (b) The associated sparsity pattern of the precision matrix $\mathbf{\Theta}$. Zero entries are depicted with white, diagonal entries with gray, and nonzero off-diagonal entries with red squares.*

$E$ then reflects the nonzero pattern of $\mathbf{\Theta}$. For any pair of nodes $(i, j)$ for which $\mathbf{\Theta}_{ij} = 0$ we have that $(i, j) \notin E$. We illustrate this correspondence in Figure 2.2 for an undirected graph. Sparsity of the true precision matrix is a prevailing assumption [40, 41], and models which only involve a small number variables, i.e. sparse models, are inherently simpler [42]. Thus, it is often preferable to retrieve the structure of complex phenomena with a sparse model, that is easier to interpret [43]. Additionally, in many applications it is assumed that few conditionally dependent factors govern the overall behavior of the model, see for example the sparsity imposed in financial [44] and brain studies [45]. Similar arguments in favor of sparsity are commonly encountered in Bayesian approaches for the retrieval of Gaussian graphical models [46], and in approximate techniques that scale variational inference in Gaussian processes to large datasets [47, 48].

As a consequence, the problem of learning sparse undirected Gaussian graphical models in high dimensional settings has been central in machine learning, statistics and optimization. A class of algorithms that focuses on estimating sparse precision matrices is based on the graphical lasso [4, 49], which applies an $\ell_1$ penalty to the negative log-likelihood function 2.2.2. We discuss in the following section the problem formulation for the graphical lasso, and outline alternative sparse estimation methods in Section 2.5.1.

## 2.3 $\ell_1$ regularized negative log-likelihood

Let $\mathbf{Y} \in \mathbb{R}^{p\times n}$ be once more a dataset of $n$ independently drawn samples from a $p$-variate Gaussian distribution characterized by a precision matrix $\boldsymbol{\Theta} \in \mathbb{R}^{p\times p}$ and a mean $\boldsymbol{\mu} \in \mathbb{R}^{p}$. Given a matrix sparsity parameter $\boldsymbol{\Lambda} \in \mathbb{R}^{p\times p}$ with positive elements $\boldsymbol{\Lambda}_{ij} > 0$, the negative $\ell_1$ regularized log-likelihood function can be written as the sum of two parts,

$$\begin{gathered} f(\boldsymbol{\Theta}) = \mathcal{L}(\boldsymbol{\Theta}) + h(\boldsymbol{\Theta}), \quad \text{where} \\ \mathcal{L}(\boldsymbol{\Theta}) = -\log\det(\boldsymbol{\Theta}) + \operatorname{tr}(\boldsymbol{\Theta}\mathbf{S}), \quad \text{and} \quad h(\boldsymbol{\Theta}) = \|\boldsymbol{\Lambda} \odot \boldsymbol{\Theta}\|_1. \end{gathered} \tag{2.3.1}$$

The estimated precision matrix is the solution of the minimization problem

$$\widehat{\boldsymbol{\Theta}} = \arg\min_{\boldsymbol{\Theta}\succ 0} f(\boldsymbol{\Theta}). \tag{2.3.2}$$

The likelihood, i.e., the first component $\mathcal{L}(\boldsymbol{\Theta})$ of the objective function (2.3.1) is smooth and strictly convex, as $\boldsymbol{\Theta} \succ 0$ and thus $\nabla^2\mathcal{L}(\boldsymbol{\Theta}) \succ 0$. Its gradient and hessian are defined as [50]

$$\nabla\mathcal{L}(\boldsymbol{\Theta}) = \mathbf{S} - \boldsymbol{\Theta}^{-1} \quad \text{and} \quad H = \nabla^2\mathcal{L}(\boldsymbol{\Theta}) = \boldsymbol{\Theta}^{-1} \otimes \boldsymbol{\Theta}^{-1}. \tag{2.3.3}$$

With $h(\boldsymbol{\Theta})$, the second component of $f(\boldsymbol{\Theta})$, being also convex for $\boldsymbol{\Lambda}_{ij} > 0$, we have that (2.3.1) is strongly convex with $\widehat{\boldsymbol{\Theta}}$ its unique minimizer, even for rank deficient systems with $n < p$. The first order optimality conditions of (2.3.2) read

$$\partial f(\boldsymbol{\Theta}) = \nabla\mathcal{L}(\boldsymbol{\Theta}) + \boldsymbol{\Lambda} \odot \boldsymbol{\Xi}, \quad \text{with} \quad \boldsymbol{\Xi} = \begin{cases} \operatorname{sign}(\boldsymbol{\Theta}_{ij}), & \text{if } \boldsymbol{\Theta}_{ij} \neq 0, \\ [-1, +1], & \text{if } \boldsymbol{\Theta}_{ij} = 0, \end{cases} \tag{2.3.4}$$

with the subgradient denoted as $\partial$.

This convex composite optimization problem enjoys significant computational advantages when the sought precision matrices have block diagonal structure [51, 52], as the final solution can be composed by the individual result of each block. We will exploit this and other advantages of performing block computations in the following Chapter 3, where we introduce a high performance algorithm based on the graphical lasso for the retrieval of precision matrices.

## 2.4 $M$-matrices

Matrices that arise from Markov processes in probability and statistics often have some particular structure [53]. One of the most frequent occurrences is when the matrix $\boldsymbol{\Theta}$ under consideration includes non-positive off-diagonal and non-negative diagonal elements, i.e., when $\boldsymbol{\Theta}$ is a finite matrix of the type

$$\boldsymbol{\Theta} = \begin{bmatrix} \boldsymbol{\Theta}_{11} & -\boldsymbol{\Theta}_{12} & -\boldsymbol{\Theta}_{12} & \cdots \\ -\boldsymbol{\Theta}_{21} & \boldsymbol{\Theta}_{22} & -\boldsymbol{\Theta}_{23} & \cdots \\ -\boldsymbol{\Theta}_{31} & -\boldsymbol{\Theta}_{32} & \boldsymbol{\Theta}_{33} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \tag{2.4.1}$$

with non-negative entries $\boldsymbol{\Theta}_{ij} \geq 0$. In the context of multivariate statistics, the problem of retrieving precision matrices of the form (2.4.1) corresponds to finding only the variables that are non-negatively correlated. The resulting matrices are referred to as $M$-matrices [10, 11], and are part of the symmetric set

$$\mathcal{S}_M = \left\{ \boldsymbol{\Theta} \in \mathbb{R}^{p \times p} | \boldsymbol{\Theta}_{ij} = \boldsymbol{\Theta}_{ji} \leqslant 0, \ \forall\, i \neq j, \ \boldsymbol{\Theta} \succ 0 \right\}, \tag{2.4.2}$$

where $\succ$ denotes positive definiteness. When matrices of the set (2.4.2) encode the graphical structure a Markov random field, they induce total positivity between the random variables, which is a special form of positive dependence. This property stems from the fact that the precision matrix having nonpositive off-diagonal entries is equivalent to the existence of nonnegative partial correlations $-\boldsymbol{\Theta}_{ij}/\sqrt{\boldsymbol{\Theta}_{ii}\boldsymbol{\Theta}_{jj}}$ [10]. The underlying distribution of such undirected Gaussian graphical models is called multivariate totally positive of order 2, or MTP$_2$. A probability distribution $P$ on $\mathbb{R}^p$ is MTP$_2$ if

$$P(\mathbf{x})P(\mathbf{y}) \leq P(\mathbf{x} \wedge \mathbf{y})P(\mathbf{x} \vee \mathbf{y}), \quad \forall \ \mathbf{x}, \mathbf{y} \in \mathbb{R}^p, \tag{2.4.3}$$

with $\wedge, \vee$ denoting the coordinate-wise maximum and minimum respectively [54]. In particular, a multivariate Gaussian distribution with mean $\boldsymbol{\mu}$ and a positive definite covariance matrix $\boldsymbol{\Sigma}$ is MTP$_2$ if and only if the precision matrix is a symmetric $M$-matrix in the set (2.4.2), with such models also known as attractive Gaussian Markov random fields [55].

It has been proven in [56] that for the negative log-likelihood function $\mathcal{L}(\boldsymbol{\Theta})$ the optimizer

$$\widehat{\boldsymbol{\Theta}} = \underset{\boldsymbol{\Theta} \in \mathcal{S}_M}{\arg\min}\ \mathcal{L}(\boldsymbol{\Theta}), \tag{2.4.4}$$

exists and is unique for

$$\mathbf{S}_{ij} < \sqrt{\mathbf{S}_{jj}\mathbf{S}_{ii}}, \ \forall\, i \neq j, \quad \text{and} \quad \mathbf{S}_{ii} > 0 \ \forall\, i, \tag{2.4.5}$$

i.e., as long as no two variables $i, j$ are perfectly positively positively correlated, and no variable $i$ is constant. In contrast to the unconstrained MLE estimator (see Section 2.2), $\widehat{\boldsymbol{\Theta}}$ exists for $n \geq 2$, thus, the estimator (2.4.4) is well defined in the high dimensional regime $p > n$ [57]. Additionally, for MTP$_2$ distributions, similarly to (2.1.4), it holds that two random variables $\mathbf{y}_i, \mathbf{y}_j$ are conditionally independent if and only if $i, j$ are separated in the underlying graphical structure $\mathcal{G}(\boldsymbol{\Theta})$ [58].

Nevertheless, the arguments for sparsity in the retrieved graphical structures of Section 2.2.1 regarding performance and explainability of the results hold when estimating $M$-matrices from high dimensional attractive GMRFs. It is therefore common practise to include an $\ell_1$ regularization terms in the minimization objective of (2.4.4) to promote sparsity in the estimates. The graphical lasso estimator is now constrained to the set (2.4.2), and defined as

$$\widehat{\boldsymbol{\Theta}} = \underset{\boldsymbol{\Theta} \in \mathcal{S}_M}{\arg\min}\ \mathcal{L}(\boldsymbol{\Theta}) + \|\boldsymbol{\Lambda} \odot \boldsymbol{\Theta}\|_1. \tag{2.4.6}$$

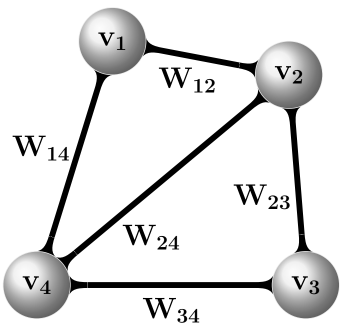
The retrieved $M$-matrices $\widehat{\boldsymbol{\Theta}}$ are tightly connected with the combinatorial graph Laplacian, a matrix with numerous applications in graph learning and clustering tasks. We analyze this connection in the following section, and list alternative methods for $M$-matrix estimation in Section 2.5.2.

### 2.4.1 Connection to graph Laplacians

The combinatorial graph Laplacian $\mathbf{L} \in \mathbb{R}^{p \times p}$ is a symmetric positive semidefinite matrix. If we allow $\boldsymbol{\Theta} \succcurlyeq 0$ in (2.4.2), then graph Laplacians would be part of the subset of $M$-matrices

$$\begin{aligned} \mathcal{S}_L = \{\boldsymbol{\Theta} \in \mathbb{R}^{p\times p} \,|\, \boldsymbol{\Theta}_{ij} = \boldsymbol{\Theta}_{ji} \leqslant 0\ \forall i \neq j; \\ \boldsymbol{\Theta}_{ii} = -\sum_{j:i\neq j} \boldsymbol{\Theta}_{ij},\ \ \boldsymbol{\Theta} \succcurlyeq 0\}. \end{aligned} \tag{2.4.7}$$

The constant vector of ones $\mathbf{e}$ lies in its nullspace, i.e. $\mathbf{L} \cdot \mathbf{e} = 0$, because the row and column sums of $\mathbf{L}$ are zero, i.e. $\mathbf{L}_{ii} + \sum_{i\neq j} \mathbf{L}_{ij} = 0$. A very important property of the spectrum of $\mathbf{L}$ is that the multiplicity of the zero eigenvalue corresponds to the number of connected components $k$ of the graph [15]. This also implies



$$\mathbf{W} = \begin{bmatrix} 0 & \mathbf{A}_{12} & 0 & \mathbf{W}_{14} \\ \mathbf{W}_{12} & 0 & \mathbf{W}_{23} & \mathbf{W}_{24} \\ 0 & \mathbf{W}_{23} & 0 & \mathbf{W}_{34} \\ \mathbf{W}_{14} & \mathbf{W}_{24} & \mathbf{W}_{34} & 0 \end{bmatrix},$$

$$\mathbf{D}_{ii} = \begin{bmatrix} \sum_j \mathbf{W}_{1j} \\ \sum_j \mathbf{W}_{2j} \\ \sum_j \mathbf{W}_{3j} \\ \sum_j \mathbf{W}_{4j} \end{bmatrix}, \quad \mathbf{L} = \mathbf{D} - \mathbf{W}.$$

Figure 2.3: *A simple, undirected, and connected graph $\mathcal{G}(V, E, \mathbf{W})$ with 4 vertices and 5 edges, with its weighted adjacency* **W**, *degree* **D**, *and combinatorial graph Laplacian* **L** *matrices.*

that the Laplacian matrix is singular with rank $p - k > 0$. We then consider that $\mathbf{L}$ encodes an improper GMRF (IGMRF) [2, 59] of rank $p - k$, as opposed to $\boldsymbol{\Theta}$ in (2.4.2) which is of full rank.

An undirected weighted graph $\mathcal{G}(V, E, \mathbf{W})$, as illustrated in Figure 2.3, is defined by its node set $V = \{1, 2, \dots, p\}$ representing the data points, and the similarity between the edges $E$ which is encoded in the elements of the weighted adjacency matrix $\mathbf{W} \in \mathbb{R}^{p \times p}$. Its combinatorial graph Laplacian $\mathbf{L}$ can be understood in terms of $\mathbf{W}$, that encodes the weights $\mathbf{W}_{ij} \geq 0$ of the edges, and the diagonal degree matrix $\mathbf{D} \in \mathbb{R}^{p \times p}$, which captures the degree of each node $\mathbf{D}_{ii} = \sum_{j=1}^{p} \mathbf{W}_{ij}$, as $\mathbf{L} = \mathbf{D} - \mathbf{W}$. The positive entries of $\mathbf{W}$, or equivalently the negative off-diagonal entries of $\mathbf{L}$, represent the edge weights of a graph, while zero entries $\mathbf{W}_{ij} = 0, i \neq j$, imply that there is no connection between nodes $i$ and $j$. For a simple and undirected graph we additionally consider that $\mathbf{W}_{ii} = 0$, and $\mathbf{W}_{ij} = \mathbf{W}_{ji}$. The graph Laplacian is often also realized as the linear operator whose action on a vector $\mathbf{u} \in \mathbb{R}^n$ induces the following quadratic form

$$\langle \mathbf{u}, \mathbf{L}\mathbf{u} \rangle = \mathbf{u}^{\mathsf{T}} \mathbf{L} \mathbf{u} = \frac{1}{2} \sum_{i,j=1}^{n} \mathbf{W}_{ij} \left( u_i - u_j \right)^2, \tag{2.4.8}$$

demonstrating the positive semidefiniteness of $\mathbf{L}$. The eigenvalues of $\mathbf{L}$ can be ordered as $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$, with the eigenvector associated with $\lambda_1 = 0$ being the constant one, i.e., $\mathbf{v}^{(1)} = c \cdot \mathbf{e}$, where $c \in \mathbb{R}$.

Different variants of graph Laplacian matrices have also been extensively studied. The normalized symmetric $\mathbf{L}_{\text{sym}} = \mathbf{D}^{-1/2} \mathbf{L} \mathbf{D}^{-1/2}$ and random walk $\mathbf{L}_{\text{rw}} = \mathbf{D}^{-1} \mathbf{L}$ Laplacians [60] are both scaled by the degree of the edges and have been successfully used for clustering tasks [17, 18, 61]. Additionally, nonlinear reformulations of the graph Laplacian from the traditional 2-norm to the $p$-norm for $p \in (1, 2]$ have proven to lead to a sharp approximation of balanced cut metrics and improved clustering assignments [23, 62].

All abovementioned graph Laplacian variants can be constructed after obtaining the weights of the graph's edges, encoded in the adjacency matrix $\mathbf{W}$. Following the $M$-matrix estimation routines introduced in Chapter 3 we can estimate the non-negatively correlated variables of a GMRF, and then set $\mathbf{W} = -\widehat{\mathbf{\Theta}}$. The appropriate type of graph Laplacian is subsequently built according to the application at hand.

## 2.5 Related work on graph learning

We present here a short overview of the various available precision and $M$-matrix estimation methods. Accuracy and performance comparisons with some of these algorithms are offered in Chapter 4.

### 2.5.1 Precision matrix estimation

There are two main groups of methods for the estimation of undirected Gaussian graphical models in the high dimensional setting. A first class attempts to directly recover the precision matrix through the solution of the $\ell_1$ regularized MLE problem (2.3.1). The graphical lasso problem was first addressed in [4] with a pathwise-coordinate-descent approach, and in [49] with a block coordinate descent and an accelerated gradient descent method. The convexity of the problem also enables the usage of interior point methods [42, 63], and more recently, of second-order methods [64, 65]. The second-order approach presented in [5, 6], and extended in [66] for large scale computations, is based on a quadratic approximation of the MLE problem. This method inspired the development of Sparse QUadratic Inverse Covariance matrix estimation (SQUIC), a performant and parallel precision matrix estimation routine presented in [7, 8]. We contribute in this state-of-the-art solution method with algorithmic routines presented in Chapter 3.

Parallel to the graphical lasso approach, many recent methods have been proposed to estimate precision matrices using alternative objectives. For example, the authors in [67, 68] utilize the coordinate wise minimization of a regression-based formulation which has been shown to have robust model selection properties compared to other Gaussian approaches. In [69] and [70] the explicit recovery of the edges that belong to the graphical model is achieved through conditional independence tests and neighborhood selection, respectively. Alternative approaches include EQUAL [71], which utilizes the penalized quadratic loss functions introduced in [72], and FASTCLIME [73, 74, 75] which casts sparse precision matrix estimation as a linear programming problem and solves it with the parametric simplex algorithm. In [40] an edge-cardinality constrained negative log-likehood problem is solved with mixed-integer optimization techniques and ridge regularization. Last, the MDMC algorithm [76] approximates the graphical lasso problem by soft thresholding the sample covariance matrix and performing a maximum determinant matrix completion.

### 2.5.2 $M$-matrix estimation

Naturally, many of the methods employed for the retrieval of non-negatively correlated variables from GRMFs are based on algorithms initially proposed for precision matrix estimation, since $M$-matrix estimation can be viewed as a constrained version of the MLE under Gaussianity assumptions. Our proposed algorithms for $M$-matrix retrieval are based on the sparse quadratic approximation of the constrained MLE objective (2.4.6), and are presented in the following Chapter 3. In [56] $M$-matrices are estimated with a sign-constrained log-determinant divergence minimization algorithm without regularization, thus limiting the applicability of the algorithm to smaller datasets. In the same work, the fact that an a-posteriori thresholding of the off-diagonal entries of the precision matrix successfully retrieves matrices that encapsulate only the positively correlated variables is established. In [77] an algorithm based on conditional independence testing that does not require any tuning parameters is proposed that estimates only the graphical structure without the weights. In [78] the optimization problem is solved with an alternating direction method of multipliers (ADMM) algorithm with LASSO and adaptive LASSO penalties.

Graph Laplacian matrices are singular, with their off-diagonal entries capturing the weight of the edges of the graph in reversed sign. Thus, their retrieval entails additional difficulties from an optimization perspective. Here, the initial work of Lake and Tenenbaum [79] focuses on the optimization of an $\ell_1$ regularized MLE problem by adding positive constant values to the diagonal entries of the graph Laplacian to account for its singularity and enforce positive definiteness. In [80], the authors build upon their previous work in the field [81], and propose a framework for the estimation of graph Laplacian matrices by introducing new problem formulations with sign and structural (i.e., connectivity) constraints, and develop tailored algorithms for these problems using again an

$\ell_1$ regularization term to enforce sparsity. Similarly, the work in [59] converts combinatorial structural constraints into spectral ones on graph matrices, and develops an optimization framework based on block majorization-minimization. In [82] nonconvex regularization terms are proposed in order to enforce sparsity in the retrieved matrices.

Additionally, various $M$-matrix learning algorithms have been proposed based on the assumption that the graph structure emerges from a set of smooth signals. The authors in [83] adopted a factor analysis model, and imposed a Gaussian probabilistic prior on the latent variables that control these signals, in order to obtain a graphical representation. In [84] the same problem is formulated as a weighted $\ell_1$ minimization, and in [85] a scalable variant is proposed that utilizes approximate nearest neighbors techniques to reduce the dimensionality of the problem.

# Chapter 3

# Sparse quadratic approximation for graph learning

The estimation of precision matrices is a challenging problem in high dimensional settings, with the maximum likelihood problem being ill-posed and potentially ill-conditioned. Regularization with the $\ell_1$-norm is commonly used to circumvent these issues and to induce sparsity in the estimates. However, scenarios where the number of dimensions far exceed the available samples still present difficulties attributed to the significant increases in computational costs. In this chapter we present our Sparse QUadratic Inverse Covariance matrix estimation (SQUIC) method, a second-order approach for the retrieval of large-scale precision matrices, that addresses the computational challenges associated with the graphical lasso problem. We begin in Section 3.1 with the quadratic approximation of the problem, which is the basis upon our method is developed. The SQUIC algorithm is presented in Section 3.2, and its optimized components that exploit the presence of block structure in Section 3.3. Last, in Section 3.4 we extend the idea of sparse quadratic approximation to $M$-matrices, and introduce two algorithms for their retrieval.

## 3.1 Quadratic approximation

We consider again $n$ independent and identically distributed samples $\mathbf{Y} \in \mathbb{R}^{p\times n}$ of a $p$-variate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, with the true covariance matrix and mean denoted as $\boldsymbol{\Sigma} \in \mathbb{R}^{p\times p}$ and $\boldsymbol{\mu} \in \mathbb{R}^{p}$, respectively. The graphical lasso problem for the retrieval of the precision matrix $\mathbf{\Theta}$ reads

$$\widehat{\mathbf{\Theta}} = \arg\min_{\mathbf{\Theta} \succ \mathbf{0}} \left\{ \underbrace{-\log\det(\mathbf{\Theta}) + \operatorname{tr}(\mathbf{\Theta}\mathbf{S})}_{\mathcal{L}(\mathbf{\Theta})} + \underbrace{\|\mathbf{\Lambda} \odot \mathbf{\Theta}\|_1}_{h(\mathbf{\Theta})} \right\}, \tag{3.1.1}$$

where $\mathbf{S} \in \mathbb{R}^{p\times p}$ is the sample covariance matrix. We denote the log-likelihood function with $\mathcal{L}(\cdot)$, and the regularization term with $h(\cdot)$. Positive-definiteness of the estimated precision matrix $\widehat{\mathbf{\Theta}} \succ 0$ is established since it is an approximate of the inverse of $\mathbf{\Sigma} \succ 0$. Thus, precision matrices lie in the cone

$$\mathcal{S}^p_{++} = \left\{ \mathbf{\Theta} \in \mathbb{R}^{p\times p} \mid \mathbf{\Theta} = \mathbf{\Theta}^\intercal, \mathbf{\Theta} \succ 0 \right\}. \tag{3.1.2}$$

### 3.1.1 Motivation

First-order methods for the solution of (3.1.1) are based on the first-order optimality conditions and the direction of the subgradient

$$\partial\left(\mathcal{L}(\mathbf{\Theta}) + h(\mathbf{\Theta})\right) = \mathbf{S} - \mathbf{\Theta}^{-1} + \mathbf{\Lambda} \odot \mathbf{\Xi}, \text{ with } \mathbf{\Xi} = \begin{cases} \operatorname{sign}(\mathbf{\Theta}_{ij}), & \text{if } \mathbf{\Theta}_{ij} \neq 0, \\ [-1,+1], & \text{if } \mathbf{\Theta}_{ij} = 0, \end{cases} \tag{3.1.3}$$

with $\mathbf{\Xi}$ reducing to the sign of $\mathbf{\Theta}_{ij}$ for entries $\mathbf{\Theta}_{ij} \neq 0$, and lying in the range $[-1,+1]$ for $\mathbf{\Theta}_{ij} = 0$. Such method have enjoyed increased popularity in high dimensional problems due to their ease of implementation, and their low requirements in memory and computation at each gradient step. However, they exhibit at most linear convergence rates that render them inapplicable for problems whose dimensionality $p$ exceeds the order of thousands [86, 87]. The usage of second-order methods which consider the Hessian, or at least part of it, in order to reach superlinear convergence rates is an attractive alternative, but one that is faced with computational challenges due to the prohibitively large size of the Hessian

$$H = \mathbf{\Theta}^{-1} \otimes \mathbf{\Theta}^{-1} \in \mathbb{R}^{p^2\times p^2}. \tag{3.1.4}$$

The difficulties of applying a Newton-type algorithm for the solution of (3.1.1) are further exacerbated by the positive-definiteness requirement imposed by the set (3.1.2). In the original graphical lasso problem, the log-determinant function in the Gaussian MLE objective acts as a barrier function for the positive definite cone [50], highly penalizing the objective when the estimate $\widehat{\mathbf{\Theta}}$ becomes semi-definite. This property simplifies the optimization problem, and has been exploited by gradient-based methods [88]. Approximating the objective function quadratically enables the usage of second-order information, but simultaneously results in losing the barrier property of the logarithm. As a consequence, the Newton-type updates can potentially yield matrices that are not part of the cone (3.1.2), and positive-definiteness must be explicitly enforced.

### 3.1.2 The quadratic model

Addressing the aforementioned challenges, a piecewise quadratic model has been proposed [5, 6] that uses both first- and second-order information of the graphical lasso. For composite functionals with continuously differentiable (e.g., the log-likelihood $\mathcal{L}(\mathbf{\Theta})$) and non-differentiable (e.g., the $\ell_1$ regularization term $h(\mathbf{\Theta})$) components, like the one in (2.3.1), this computational approach is referred to as proximal Newton method. We refer to [89, 90] for a detailed analysis of proximal methods for composite and convex functions. The QUadratic approximation for Inverse Covariance estimation method (QUIC) method [5, 6] is part of this family of algorithms and enjoys multiple computational advantages. First, it converges with quadratic rates at the global optimum, thus drastically decreasing the number of required iterations. Additionally, due to the $\ell_1$ regularization term, the second-order updates need to be computed only for a subset of the total variables. If the size of this subset is much smaller than $p^2$, significant computational benefits are reported [6]. Finally, the direction of Newton's method is determined via a coordinate descent update strategy that considers the inherent symmetry in the Hessian of the problem and speeds up the computation.

In QUIC, similarly to [90], the smooth component $\mathcal{L}(\mathbf{\Theta})$ of the graphical lasso objective is approximated with the second-order Taylor expansion around an iterate $\mathbf{\Theta}_t$, with $t$ denoting the current iteration. Let $\mathcal{L}(\mathbf{\Theta}_t + \mathbf{\Delta})$ be the Taylor expansion for some perturbation $\mathbf{\Delta} \in \mathbb{R}^{p\times p}$. Setting the inverse of the precision $\mathbf{\Theta}_t^{-1} = \mathbf{W}$ for ease of notation,[1] and using the derivation of gradient and Hessian of the log-likelihood (2.3.3), the approximation reads

$$\mathcal{L}(\mathbf{\Theta}_t + \mathbf{\Delta}) = \mathcal{L}(\mathbf{\Theta}_t) + \operatorname{vec}(\nabla g(\mathbf{\Theta}_t))^\intercal \operatorname{vec}(\mathbf{\Delta}) + \frac{1}{2}\operatorname{vec}(\mathbf{\Delta})^\intercal \nabla^2 g(\mathbf{\Theta}_t) \operatorname{vec}(\mathbf{\Delta}). \tag{3.1.5}$$

Discarding the term $\mathcal{L}(\mathbf{\Theta}_t)$, as it is constant with respect to $\mathbf{\Delta}$, and setting $g_t(\mathbf{\Delta}) =$

[1] Note that $\mathbf{W}$ here refers to the inverse of the precision matrix, and not to the adjacency matrix.

$\mathcal{L}(\mathbf{\Theta}_t + \mathbf{\Delta})$ we get

$$\begin{aligned} g_t(\mathbf{\Delta}) &= \mathrm{vec}(\nabla g(\mathbf{\Theta}_t))^\intercal \mathrm{vec}(\mathbf{\Delta}) + \frac{1}{2}\mathrm{vec}(\mathbf{\Delta})^\intercal \nabla^2 g(\mathbf{\Theta}_t)\mathrm{vec}(\mathbf{\Delta}) \\ &= \mathrm{vec}(\mathbf{S}-\mathbf{W})^\intercal \mathrm{vec}(\mathbf{\Delta}) + \frac{1}{2}\mathrm{vec}(\mathbf{\Delta})^\intercal (\mathbf{W}\otimes\mathbf{W})\,\mathrm{vec}(\mathbf{\Delta}) \\ &= \mathrm{tr}\,(\mathbf{S}-\mathbf{W})\mathbf{\Delta} + \frac{1}{2}\,\mathrm{tr}(\mathbf{W\Delta W\Delta}), \end{aligned} \tag{3.1.6}$$

where vec is the vectorized equivalent of a matrix, such that for $\mathbf{\Theta} \in \mathbb{R}^{p\times p}$ we have $\mathrm{vec}(\mathbf{\Theta}) \in \mathbb{R}^{p^2}$. Here, we have also used the fact that $\mathrm{vec}(\mathbf{\Delta})^\intercal(\mathbf{W}\otimes\mathbf{W})\mathrm{vec}(\mathbf{\Delta}) = \mathrm{tr}(\mathbf{W\Delta W\Delta})$ (see Chapter 10.2 in [39] for this property of the Kronecker product). The Newton direction $\mathbf{D}_t$ for the composite objective $f(\mathbf{\Theta})$ can now be written as

$$\begin{aligned} \mathbf{D}_t &= \operatorname*{arg\,min}_{\mathbf{\Delta}} \left\{ g_t(\mathbf{\Delta}) + h(\mathbf{\Theta}_t + \mathbf{\Delta}) \right\} \\ &= \operatorname*{arg\,min}_{\mathbf{\Delta}} \left\{ g_t(\mathbf{\Delta}) + \|\mathbf{\Lambda} \odot (\mathbf{\Theta}_t + \mathbf{\Delta})\| \right\}. \end{aligned} \tag{3.1.7}$$

In what follows, we simplify further the notation and drop the subscript $t$, to denote with $\mathbf{\Theta}$ the precision matrix, with $\mathbf{D}$ the Newton direction, and with $g$ the Taylor approximation of the log-likelihood at a given iteration.

### 3.1.3 Newton direction via coordinate descent

Finding the optimal Newton direction in (3.1.7) is closed-form problem that can be solved by coordinate descent update, which is an efficient strategy for $\ell_1$ regularized problems [91]. Additionally, exploiting the structure of the second-order term $\mathrm{tr}(\mathbf{W\Delta W\Delta})$ in (3.1.6) can significantly reduce the cost of the update.

We consider the update value $\nu$ for one variable $\mathbf{\Theta}_{ij}$, with $i < j$, that preserves symmetry in the updated Newton direction $\mathbf{D}'$ as $\mathbf{D}'_{ij} = \mathbf{D}_{ij} + \nu(\mathbf{e}_i\mathbf{e}_j^\intercal + \mathbf{e}_j\mathbf{e}_i^\intercal)$, with $\mathbf{e}_i$ being the $i$-th entry of the all-ones vector. The coordinate descent procedure for the one-variable problem corresponding to (3.1.7) is

$$\mathbf{D}'_{ij} \leftarrow \mathbf{D}_{ij} + \operatorname*{arg\,min}_{\nu} \underbrace{g\left(\mathbf{D}_{ij} + \nu(\mathbf{e}_i\mathbf{e}_j^\intercal + \mathbf{e}_j\mathbf{e}_i^\intercal)\right)}_{\text{Taylor approx.}} + \underbrace{2\mathbf{\Lambda}_{ij} \odot |\mathbf{\Theta}_{ij} + \mathbf{D}_{ij} + \nu|}_{\text{reg. term}}. \tag{3.1.8}$$

Expanding the terms in the quadratic approximation of the log-likelihood (3.1.6) with $\mathbf{D}_{ij} + \nu(\mathbf{e}_i\mathbf{e}_j^\intercal + \mathbf{e}_j\mathbf{e}_i^\intercal)$ in the place of $\mathbf{\Delta}$, the trace $\mathrm{tr}(\mathbf{WDWD})$ is the only remaining quadratic term with respect to $\mathbf{D}$. The minimum of (3.1.8) is achieved for

$$\nu = -\gamma + S(\gamma - \beta/\alpha, \mathbf{\Lambda}_{ij}/\alpha), \tag{3.1.9}$$

where the soft thresholding operator

$$S(\frac{\gamma-\beta}{\alpha}, \frac{\mathbf{\Lambda_{ij}}}{\alpha}) = \operatorname{sign}\left(\frac{\gamma-\beta}{\alpha}\right) \max\left\{ |\frac{\gamma-\beta}{\alpha} - \frac{\mathbf{\Lambda_{ij}}}{\alpha}, 0| \right\} \tag{3.1.10}$$

moves $\frac{\gamma-\beta}{\alpha}$ towards zero by the amount $\frac{\mathbf{\Lambda_{ij}}}{\alpha}$, and sets it to zero if $|\frac{\gamma-\beta}{\alpha}| \leq \frac{\mathbf{\Lambda_{ij}}}{\alpha}$. We refer to Chapter 2 in [33] for an in-depth discussion of the role of the soft thresholding function in coordinate descent solutions of $\ell_1$ regularized quadratic problems. In the graphical lasso case, the scalar values $\alpha, \beta, \gamma$, derived in detail in [5], are defined as

$$\begin{aligned} \alpha &= \mathbf{W}_{ij}^2 + \mathbf{W}_{ii}\mathbf{W}_{jj} \\ \beta &= \mathbf{S}_{ij} - \mathbf{W}_{ij} + \mathbf{W}_{:i}^{\mathsf{T}}\mathbf{D}\mathbf{W}_{:j} \\ \gamma &= \mathbf{\Theta}_{ij} + \mathbf{D}_{ij}. \end{aligned} \tag{3.1.11}$$

Note that with $\alpha$ and $\gamma$ being easy to evaluate the main computational bottleneck lies in the evaluation of the term $\mathbf{W}_{:i}^{\mathsf{T}}\mathbf{D}\mathbf{W}_{:j}^{\mathsf{T}}$ in $\beta$, which requires $\mathcal{O}(p^2)$ time computed directly. An efficient alternative is to keep a $p \times p$ matrix $\mathbf{F} = \mathbf{D}\mathbf{W}$ in buffer, and compute instead $\mathbf{W}_{:i}^{\mathsf{T}}\mathbf{F}_{:j}$ in $\mathcal{O}(p)$ flops.

### 3.1.4 Reducing the size of the search space

The cost of computing the Newton direction (3.1.7) can be further reduced by limiting the set of variables for which an update, and thus also the precision matrix, will be computed at each iteration. We denote the indices for which the computation will take place as $\mathcal{I}_{\text{free}}$, and those that remain constant throughout the optimization as $\mathcal{I}_{\text{fixed}}$. The full index set is $\mathcal{I} := \{1, 2, \ldots, p\} \times \{1, 2, \ldots, p\}$. To derive a reduction of the search space of the graphical lasso problem we consider an optimal update $\mathbf{\Delta}^*$ that results in $\partial f(\mathbf{\Theta} + \mathbf{\Delta}^*) = 0$ (see Section 3.1.1 for the subgradient of the composite functional $f$). The stationary conditions for the smooth part of the composite objective read

$$\nabla\mathcal{L}(\mathbf{\Theta} + \mathbf{\Delta}^*) = \begin{cases} -\mathbf{\Lambda}_{ij}, & \text{if } \mathbf{\Theta}_{ij} + \mathbf{\Delta}_{ij}^* > 0, \\ \mathbf{\Lambda}_{ij}, & \text{if } \mathbf{\Theta}_{ij} + \mathbf{\Delta}_{ij}^* < 0, \\ [-\mathbf{\Lambda}_{ij}, \mathbf{\Lambda}_{ij}], & \text{if } \mathbf{\Theta}_{ij} + \mathbf{\Delta}_{ij}^* = 0. \end{cases} \tag{3.1.12}$$

It follows immediately that $\mathbf{\Delta}^*_{ij} = 0$ for $\mathbf{\Theta}_{ij} = 0$ and for $\nabla\mathcal{L}(\mathbf{\Theta})_{ij} \in [-\mathbf{\Lambda}_{ij}, \mathbf{\Lambda}_{ij}]$, or equivalently for $|\nabla\mathcal{L}(\mathbf{\Theta})_{ij}| \leq \mathbf{\Lambda}_{ij}$. This observation allows us to split the full index set $\mathcal{I}$ into two complementary parts

$$\begin{aligned} \mathcal{I}_{fixed} &:= \Big\{\{i,j\} \in \mathcal{I} : |\nabla\mathcal{L}(\mathbf{\Theta})_{ij}| \leq \mathbf{\Lambda}_{ij} \text{ and } \mathbf{\Theta}_{ij} = 0\Big\}, \\ \mathcal{I}_{free} &:= \Big\{\{i,j\} \in \mathcal{I} : |\nabla\mathcal{L}(\mathbf{\Theta})_{ij}| > \mathbf{\Lambda}_{ij} \text{ or } \mathbf{\Theta}_{ij} \neq 0\Big\}, \\ &:= \mathcal{I} \backslash \mathcal{I}_{fixed}. \end{aligned} \tag{3.1.13}$$

Noting that the gradient of the negative log-likehood is elementwise $\nabla\mathcal{L}(\mathbf{\Theta})_{ij} = \mathbf{S}_{ij} - \mathbf{W}_{ij}$ offers a deeper insight in this relationship and the way that the index sets are computed. The key assumption here is that $\mathbf{\Theta}_{ij} = 0$ is already optimal for an index pair $(i,j) \in \mathcal{I}_{fixed}$, thus no update $\mathbf{D}_{ij}$ will be computed. The coordinate descent updates to find the Newton direction are restricted to the $\mathcal{I}_{free}$ subset. For carefully selected values of $\mathbf{\Lambda}_{ij}$, the number of variables in $\mathcal{I}_{free}$ will be much less than $p^2$, i.e., the variables of the entire index set $\mathcal{I}$. This results in significant computational advantages for larger values of $\mathbf{\Lambda}_{ij}$, and sparser graphical structures.

### 3.1.5 Step size computation

In each Newton step, computed only for the indices in $\mathcal{I}_{\text{free}}$, an appropriate step size $\kappa \in (0,1)$ has to be selected that leads the quadratic approximate function (3.1.6) towards sufficient decrease, and also ensures that the next iterate $\mathbf{\Theta}' = \mathbf{\Theta} + \kappa\mathbf{D}$ will be positive definite. Similarly to [92], the updated precision matrix is required to satisfy an Armijo line-search criterion [87], which for $\ell_1$ regularized problems reads

$$f(\mathbf{\Theta} + \kappa\mathbf{D}) \leq f(\mathbf{\Theta}) + \kappa\sigma\zeta \;\text{ with }\; \zeta = \text{tr}(\mathbf{D}\nabla\mathcal{L}(\mathbf{\Theta})) + \|\mathbf{\Lambda} \odot (\mathbf{\Theta} + \mathbf{D})\|_1 - \|\mathbf{\Lambda} \odot \mathbf{\Theta}\|_1, \tag{3.1.14}$$

with $\sigma \in (0, 0.5)$. Condition (3.1.14) ensures that the objective function decreases by a certain amount, with $\zeta$ measuring the proximity of the current solution $\mathbf{\Theta}$ to the global optimum [93]. Following the computation of the step size $\kappa$, the positive-definiteness of the updated precision matrix can be verified by the Cholesky decomposition $\mathbf{\Theta}' = \mathbf{L}\mathbf{L}^{\mathsf{T}}$, which only exists if $\mathbf{\Theta} + \kappa\mathbf{D} \succ 0$.

**Algorithm 1** The key algorithmic operations present in QUIC.

```
1: Estimate the sample covariance matrix $\mathbf{S}$                          ➜ acc.(2.2.3)
2: Evaluate the quadratic approximation of $\mathcal{L}(\mathbf{\Theta})$      ➜ acc.(3.1.6)
3: Compute Newton's direction $\mathbf{D}$ restricted to $\mathcal{I}_{\text{free}}$   ➜ acc.(3.1.7), (3.1.13)
4: Update of the current estimate $\mathbf{\Theta}$ via line search            ➜ acc. (3.1.14)
5: repeat
6:     steps 2 – 4
7: until convergence criteria are met
```

### 3.1.6 Key steps and computational challenges

The key steps of QUIC for the estimation of precision matrices are summarized in Algorithm 1. It is important to note that the majority of the computational operations are dense, and therefore not applicable in high-dimensional settings. In particular, in step 1 the sample covariance matrix **S** is computed via a matrix-matrix multiplication taking $O(np^2)$ time. Then in step 2 the quadratic approximation of $\mathcal{L}$ is evaluated, and in step 3 the inversion of the precision matrix is performed on every Newton iteration in order to compute the free index set (3.1.13) and the coordinate descent update (3.1.8). Each inversion requires $O(p^3)$ time. Finally, the $O(p^3)$ cost of Cholesky factorization dominates the computational cost in the estimation of the step-size. At every second-order iteration multiple line-searches are potentially performed, all of which will be checked for positive-definiteness. The process is repeated by approximating the objective around the updated optimizer until convergence, which is determined by examining the difference of the current composite functional from its previous value. We present in the following section the SQUIC algorithm, which addresses these computational challenges, and enables the retrieval of large-scale sparse precision matrices.

## 3.2 The SQUIC algorithm

The Sparse QUadratic Inverse Covariance matrix estimation (SQUIC) is a fast and scalable precision matrix estimation package that extends the original QUIC algorithm [5, 6] to large-scale applications. The algorithm is a second-order method that solves the $\ell_1$ regularized maximum likelihood problem using highly optimized linear algebra subroutines, which leverage the underlying sparsity and the intrinsic parallelism in the computation. It is written in C++ and parallelized with OpenMP. The SQUIC library is available on the USI GitLab at

https://www.gitlab.ci.inf.usi.ch/SQUIC/libSQUIC,

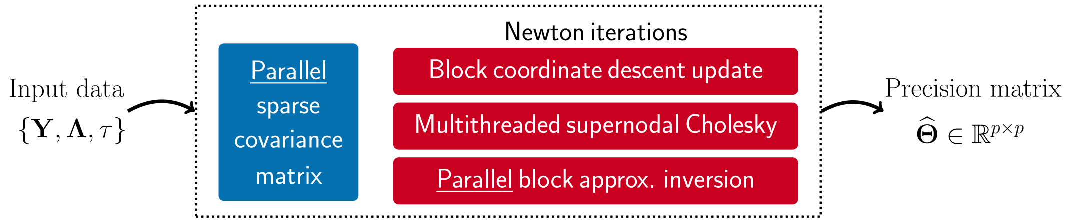

Input data
$\{\mathbf{Y}, \mathbf{\Lambda}, \tau\}$
Parallel sparse covariance matrix
Newton iterations
Block coordinate descent update
Multithreaded supernodal Cholesky
Parallel block approx. inversion
Precision matrix
$\widehat{\mathbf{\Theta}} \in \mathbb{R}^{p\times p}$

Figure 3.1: *An illustration of the optimized algorithmic components present in SQUIC.*

and is the synthesis of the work presented in [7, 8, 9]. An illustration of the critical components of the MLE method based on quadratic approximations employed in SQUIC is offered in Figure 3.1, and they key algorithmic steps are summarized in Algorithm 2. Throughout the computation efficient data structures, i.e. compressed sparse column storage (CRS), are utilized for all matrix computations, thus replacing several dense matrix routines by state-of-the-art sparse operations. The algorithm's required inputs are the data $\mathbf{Y} \in \mathbb{R}^{p\times n}$, the regularization matrix $\mathbf{\Lambda} \in \mathbb{R}^{p\times p}$ and the convergence tolerance $\tau \in \mathbb{R}$, and the output the estimated sparse precision matrix $\widehat{\mathbf{\Theta}} \in \mathbb{R}^{p\times p}$. Convergence in SQUIC is determined by measuring that the relative difference between the objective function at the updated $\mathbf{\Theta}$ and the previous $\mathbf{\Theta}_{\text{prev}}$ is below a threshold $\tau$, i.e. $\frac{\|f(\mathbf{\Theta}_{\text{prev}})-f(\mathbf{\Theta})\|}{\|f(\mathbf{\Theta}_{\text{prev}})\|} < \tau$. The initial guess for the precision matrix and its approximate inverse is set in step 1 to any sparse symmetric positive-definite matrix, which is most cases will be the identity $\mathbf{I}$. In step 2 a sparse representation of the sample covariance matrix $\mathbf{S}$ is computed in parallel fashion. We provide some details on this computation in Section 3.2.1. Entering the kernel of the algorithm in step 3, we compute the free index set $\mathcal{I}_{\text{free}}$ in step 4, and then in step 5 the composite quadratic functional is evaluated for the current value of the precision matrix. In step 6 the computation of the Newton direction $\mathbf{D}_{ij}$ is limited only to the indices that are part of $\mathcal{I}_{\text{free}}$, and is achieved with a block coordinate descent update that we present in Section 3.3.3. The line-search process in step 7 updates the current iterate $\mathbf{\Theta}$ with a step-size $\kappa \in [0, 1)$ that ensures the positive-definiteness and satisfies the Armijo criterion (3.1.14). We discuss the supernodal sparse Cholesky factorization with which we accelerate this computation and safeguard that $\mathbf{\Theta} \succ 0$ in Section 3.3.1. Last, in step 9 a parallel block approach, presented in Section 3.3.2, returns the approximate inverse of the precision matrix $\mathbf{\Theta}^{\text{inv}}$, which will be used in the next iteration.

**Algorithm 2** The key algorithmic operations present in SQUIC.

```
Input: $\mathbf{Y}, \mathbf{\Lambda}, \tau$
 1: Initialize $\mathbf{\Theta} \leftarrow \mathbf{\Theta}^{\text{inv}} \leftarrow \mathbf{I}$
 2: Estimate the sparse sample covariance matrix $\mathbf{S}$            ➜ See sec. 3.2.1
 3: while Not converged do
 4:     Compute the free index set $\mathcal{I}_{\text{free}}$          ➜ See sec. 3.1.4
 5:     Evaluate the composite quadratic functional $f(\mathbf{\Theta})$
 6:     Compute Newton's direction $\mathbf{D}$ restricted to $\mathcal{I}_{\text{free}}$   ➜ See sec. 3.3.3
 7:     Update $\mathbf{\Theta} \leftarrow \mathbf{\Theta} + \kappa\mathbf{D}$
 8:     Ensure that $\mathbf{\Theta} \succ 0$ and suff. decrease of $f(\mathbf{\Theta})$   ➜ See sec. 3.3.1
 9:     Approximate the inverse of the precision $\mathbf{\Theta}^{\text{inv}}$   ➜ See sec. 3.3.2
10: end while
Output: $\widehat{\mathbf{\Theta}}$
```

### 3.2.1 Sparse sample covariance matrix

The sample covariance matrix $\mathbf{S}$ is dense, and its computation is challenging in high dimensional settings. The elements of the $\mathbf{S}$ can be computed on the fly, with this, nonetheless, not being a cache efficient approach. We instead compute an initial sparse representation for $\mathbf{S}$ by accepting the values of indices that satisfy

$$\mathbf{\Lambda}_{ij} \leqslant |\mathbf{S}_{ij}| \leqslant \sqrt{\mathbf{S}_{ii}\mathbf{S}_{jj}} \quad \text{or } i = j. \tag{3.2.1}$$

In the main loop of Algorithm 2, when performing Newton iterations, we compute on the fly the entries $(i, j)$ of $\mathbf{S}$ that are pending computation and have a corresponding nonzero value in the approximated inverse $\mathbf{\Theta}^{\text{inv}}$. With this approach we ensure that the nonzero structure of $\mathbf{S}$ and $\mathbf{\Theta}^{\text{inv}}$ overlap.

SQUIC exploits the highly parallelizable nature of the matrix-matrix multiplication that computes the sparse sample covariance matrix. The multicore approach that is followed is presented in detail in [7], and a scalable distributed-memory version in [8].

### 3.2.2 Matrix regularization parameter

The matrix regularization parameter $\mathbf{\Lambda} \in \mathbb{R}^{p\times p}$ is a dense matrix that cannot be explicitly represented for large $p$. We therefore combine a sparse symmetric matrix $\mathbf{M} \in \mathbb{R}^{p\times p}$ with $\mathbf{M}_{ij} \geq 0$, and a scalar regularization parameter $\lambda \in \mathbb{R} > 0$, such that

$$\boldsymbol{\Lambda}_{ij} := \begin{cases} \mathbf{M}_{ij} & \text{for} \quad \mathbf{M}_{ij} \neq 0, \\ \lambda & \text{for} \quad \mathbf{M}_{ij} = 0. \end{cases} \tag{3.2.2}$$

With this approach, the $p^2$ entries of $\boldsymbol{\Lambda}$ can be represented with a total number of entries equalling the nonzeros of $\mathbf{M}$, and an additional entry for the scalar $\lambda$. It also enables the encoding of a graphical bias in the optimization problem for the graphical lasso. The nonzero structure of $\mathbf{M}$ is utilized to penalize the expected nonzero pattern of $\widehat{\boldsymbol{\Theta}}$, usually with some small scalar value. The entries that are expected to be zero, for the conditionally independent variables, are regularized with a scalar parameter $\lambda$, that usually attains a larger value. Uniform regularization for all the variables, when no knowledge on the anticipated graphical structure is available, is achieved by setting $\mathbf{M} = \mathbf{0}$, which results in $\boldsymbol{\Lambda}_{ij} = \lambda, \ \forall \ (i, j)$.

## 3.3 Exploiting block structure

Enforcing sparsity in the retrieval of inverse covariance matrices simplifies the optimization process and provides interpretable results via sparse graphical representations (see Section 2.2.1 for a relevant discussion). However, real-world applications are commonly associated with community structure, or clustered dependencies that exhibit limited sparsity. As examples we may consider biological networks with genes exhibiting a hub structure [94], and grouped stocks in financial portfolios [95]. When considering such structures, the selection of the regularization parameters is pivotal, and is an active area of research [96, 97]. Large values in the regularization term correspond to increasing the threshold that controls which links will be included, thus resulting in sparser networks. If only the important connections remain, the community structure can offer insights on the modular nature of the underlying data. However, removing too many links leads to a significant loss of information [98]. An example of this behaviour is illustrated in Figure 3.2. In a dataset with $p = 100$ variables and 5 underlying communities, denoted by A - E, a large regularization parameter results in a sparse graph, but also to a loss of the community structure (see Figure 3.2a). In such clustered datasets, decreasing the threshold and including more edges in the retrieved precision matrix is critical in order to accurately retrieve the structure (see Figure 3.2b). This results in a reduction of the sparsity of the precision matrix, which is also accompanied by a drastic decrease in the sparsity of the inverse and, thus, an overall increase in the computational cost.

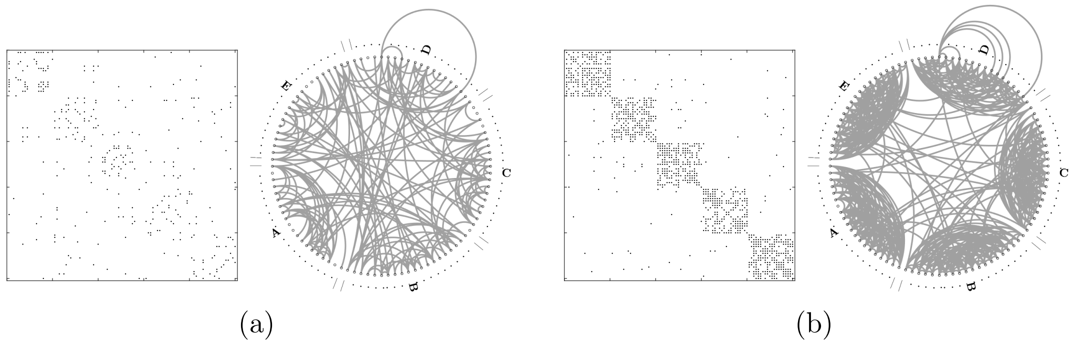


Figure 3.2: *Retrieving the graphical structure of datasets with clustered dependencies. We visualize the sparsity pattern of the precision matrix, and the graphical structure with the clusters in a circular layout. (a) For a large regularization parameter a sparser graph is estimated. However, the community structure is lost. (b) A clear community structure is retrieved with a smaller regularization parameter. The underlying precision matrix exhibits reduced sparsity and, thus, presents a computational challenge.*

In the SQUIC algorithm, the approximate matrix inversion of the precision matrix and the coordinate descent updates are both routines severely affected by any reduction in the sparsity of the estimated inverse $\boldsymbol{\Theta}^{\text{inv}}$. A popular optimization approach that mitigates such shortcomings is developing algorithms that work with submatrices or blocks of the entire dataset, instead of rows or columns of a sparse matrix [99]. With this motivation, we introduce in this section algorithmic approaches that reduce the adverse effects of limited sparsity in the precision matrix, and in the intermediary computations of the graphical lasso problem. These methods are incorporated in the SQUIC library, and are based on a supernodal strategy for sparse Cholesky decompositions (Section 3.3.1), an approximate blockwise inversion of $\boldsymbol{\Theta}$ (Section 3.3.2), and a block coordinate descent update (Section 3.3.3).

### 3.3.1 Supernodal sparse Cholesky factorization

Performant Cholesky factorizations play a central role in the overall computational efficiency of SQUIC. For each line-search iteration (step 7 in Algorithm 2) a Cholesky factorization is performed. The resulting factors are used to check the updated $\boldsymbol{\Theta}$ for positive-definiteness, and assist in the evaluation of the log-determinant term of the objective function (step 5) and in the approximate inversion of the precision matrix (step 8). For efficient computations SQUIC uses the algorithm CHOLMOD [100] for the Cholesky factorization, which is part of the SuiteSparse Matrix Collection.[2] CHOLMOD is based on the supernodal [101, 102] approach, which successively detects dense block structures during the factorization and produces a matrix in a hybrid format. The idea of a supernode is to group together consecutive columns with the same nonzero structure in order to treat them as a dense block, with benefits in both storage and computation. These dense blocks can be efficiently handled with high-performance libraries such as the Intel(R) Math Kernel Library (MKL) [103]. We note that the CHOLMOD supernodal sparse Cholesky factorization library provides internal parallelization, as it utilizes the optimized BLAS Level-3 routines. We will provide a brief discussion on its parallel performance in the numerical results presented in Chapter 4.

Our decomposition strategy is based on a combinatorial analysis that increases sparsity in the Cholesky factors with an a priori permutation of $\mathbf{\Theta}$. The supernodes, are subsequently computed as part of the symbolic analysis and set up for the factorization. Last, the sparse block Cholesky factorization is performed with dense matrix operations implemented using level-3 BLAS and LAPACK [100]. We use the $\mathbf{LDL}^\mathsf{T}$ decomposition,

$$\mathbf{\Theta} = \mathbf{PLDL}^\mathsf{T}\mathbf{P}^\mathsf{T}, \tag{3.3.1}$$

with $\mathbf{P}$ being a permutation matrix computed by CHOLMOD, $\mathbf{D}$ a block diagonal matrix with symmetric positive-definite submatrices as diagonal blocks, and $\mathbf{L}$ a block lower triangular matrix with identities as its diagonal blocks.[3] In the $\mathbf{LDL}^\mathsf{T}$ factorization, which is a variant of the more common $\mathbf{LU}$ Cholesky decomposition, a supernode is a range of $(r : s)$ of columns of $\mathbf{L}$ with the same nonzero pattern below the diagonal. We illustrate this data structure in Figure 3.3a. For the block 1 this results in $\mathbf{L}_{11} = \mathbf{L}(r_1 : s_1, r_1 : s_1)$ being a full lower triangular matrix, and every row of $\mathbf{L}(s_1 + 1 : p, r_1 : s_1)$ being either full or zero. The block diagonal $\mathbf{D}_{11}$ can then be represented as $\mathbf{L}_{11}\mathbf{L}_{11}^\mathsf{T}$. These blocks are stored with compressed rows and contiguous columns. The main assumption here is that if the underlying precision matrix is sparse then the fill-in of $\mathbf{L}$ will also be small. This naturally depends on both the true structure of $\mathbf{\Theta}$, and on the selection of the regularization coefficients.

Using this decomposition (3.3.1), the log-determinant term of the objective

[2]The SuiteSparse collection of sparse matrix algorithms is available at https://people.engr.tamu.edu/davis/suitesparse.html.

[3]Note that $\mathbf{L}$ here refers to the Cholesky factors, and not to the graph Laplacian matrix.

and the inverse $\mathbf{W} = \mathbf{\Theta}^{-1}$ are expressed as

$$\begin{aligned} \log\det\mathbf{\Theta} &= \sum_{b=1}^{q} \log\det\mathbf{D}_{bb}, \\ \mathbf{W} &= \mathbf{P}\mathbf{L}^{-\intercal}\mathbf{D}^{-1}\mathbf{L}^{-1}\mathbf{P}^{\intercal}, \end{aligned} \tag{3.3.2}$$

where we assume that $\mathbf{D}$ consists of $q$ total diagonal blocks. This routine also factorizes $\mathbf{D}_{bb} = \mathbf{L}_{bb}\mathbf{L}_{bb}^{\top}$ as a dense Cholesky decomposition such that $\mathbf{L}_{bb}$ is a lower triangular matrix. This in turn allows us to compute

$$\log\det\mathbf{D}_{bb} = 2\log\det\mathbf{L}_{bb} = 2\sum_{i}\log\mathbf{L}_{ii} \tag{3.3.3}$$

from the diagonal entries of $\mathbf{L}_{bb}$.

To compute the inverse $\mathbf{W}$, the exact inversion of the block diagonal matrix $\mathbf{D}$ is straightforward. In contrast, inverting $\mathbf{L}$ and computing the product of the factors in (3.3.2) is challenging and may result in a dense final inverse. This issue is addressed in the following section with a sparse approximate inversion of $\mathbf{L}$, to obtain the approximate inverse $\mathbf{\Theta}^{\text{inv}}$.

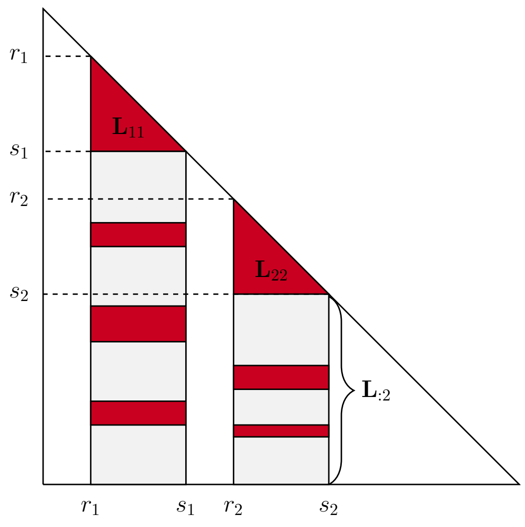
### 3.3.2 Block approximate matrix inversion

Approximating the inverse of the precision matrix entails two main computational obstacles. The first one is computing the approximate inverse of the Cholesky factors $\mathbf{L}^{\text{inv}} \approx \mathbf{L}^{-1}$, and the second calculating the matrix multiplication in (3.3.2). We alleviate the first bottleneck by considering the lower triangular matrix $\mathbf{L} = \mathbf{I} - \mathbf{E}$, with values $\mathbf{E}_{ij}$ in the off-diagonal entries and ones in the diagonal. The matrix $\mathbf{E}$ represents the strictly lower triangular part of $\mathbf{L}$. Now the exact inverse of the factor can be computed with a Neumann series as

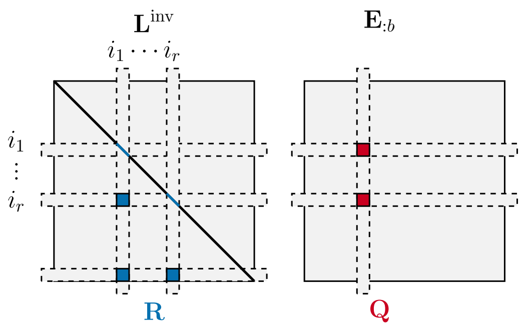
$$\mathbf{L}^{-1} = (\mathbf{I} - \mathbf{E})^{-1} = \sum_{k=0}^{p-1}\mathbf{E}^{k}. \tag{3.3.4}$$

This summation requires at most $k = p - 1$ terms, as $\mathbf{E}$ is strictly lower triangular and thus $\mathbf{E}^{p} = \mathbf{0}$. We approximate this Neumann series successively via Horner's scheme in order to compute

$$\mathbf{L}_{k+1}^{\text{inv}} = \mathbf{L}_{k}^{\text{inv}}\mathbf{E} + \mathbf{I},\ k = 1, 2\ldots, p-1, \tag{3.3.5}$$



(a)



(b)

Figure 3.3: *Exploiting block structure in the retrieval of precision matrices. (a) An illustration of a supernode set up for the Cholesky factorization* $\mathbf{\Theta} = \mathbf{LDL}^{\mathsf{T}}$*. Red elements correspond to dense regions of the matrix, and white to zero entries. (b) The data structures in the block sparse matrix-matrix multiplication used for the approximate matrix inversion. The buffer* $\mathbf{Q}$ *captures the nonzero rows and the contiguous columns of block* $\mathbf{E}_{:b}$*. The elements of* $\mathbf{L}^{\text{inv}}$ *that contribute to the multiplication are stored in* $\mathbf{R}$*.*

where $\mathbf{L}_1^{\text{inv}} := \mathbf{I} + \mathbf{E}$. This iterative process converges when the condition

$$|(\mathbf{L}_{k+1}^{\text{inv}})_{ij} - (\mathbf{L}_k^{\text{inv}})_{ij}| \leqslant \tau_{\text{inv}} \tag{3.3.6}$$

is fulfilled, with $\tau_{\text{inv}} > 0$ being a dropout threshold. To prevent excessive error propagation in the values of the updates $\mathbf{L}_{k+1}^{\text{inv}} \leftarrow \mathbf{L}_k^{\text{inv}}\mathbf{E} + \mathbf{I}$ we use a scaling factor $\gamma \in (0, 1)$ such that only the elements that satisfy

$$|(\mathbf{L}_{k+1}^{\text{inv}})_{ij} - (\mathbf{L}_k^{\text{inv}})_{ij}| > \gamma \tau_{\text{inv}} \tag{3.3.7}$$

are updated. In practice, we set $\gamma = 0.1$ in SQUIC. The final approximation $\mathbf{L}^{\text{inv}}$ is sparsified by discarding entries smaller than the tolerance level $\tau_{\text{inv}}$, and subsequently $\mathbf{\Theta}^{\text{inv}}$ is computed as per (3.3.2). The resulting entries must finally satisfy

$$|\mathbf{\Theta}_{ij}^{\text{inv}}|^2 > \tau_{\text{inv}}^2 \mathbf{\Theta}_{ii}^{\text{inv}} \mathbf{\Theta}_{jj}^{\text{inv}}. \tag{3.3.8}$$

**Algorithm 3** Parallel block approximate matrix inversion.

```
Input: $\mathbf{L}, \mathbf{D}^{\text{inv}} := \mathbf{D}^{-1}, \mathbf{P}, \tau_{\text{inv}}$
 1: Initialize $\mathbf{E} \leftarrow \mathbf{I} - \mathbf{L}, \mathbf{L}^{\text{inv}} \leftarrow \mathbf{I} + \mathbf{E}, \mathbf{u} \leftarrow \mathbf{0}$
 2: while $\max(\mathbf{u}) \leqslant \tau_{\text{inv}}$ do
 3:     for blocks $b = 1, \ldots, q$ parallel do
 4:         $r \leftarrow$ thread_id, $\mathbf{Q} \leftarrow \mathbf{E}_{:b}, \mathbf{V} \leftarrow \mathbf{0}$
 5:         for $c = 1, 2, \ldots, q$ do
 6:             Compute $\mathbf{R}_c$ given $\mathbf{L}^{\text{inv}}_{:c}$
 7:             $\mathbf{V} \leftarrow \mathbf{V} + \mathbf{R}_c \mathbf{Q}$
 8:         end for
 9:         $\mathbf{u}_r \leftarrow \max(\mathbf{u}_r, \max_i \|\mathbf{L}^{\text{inv}}_{ib} - \mathbf{V}_{i:}\|_\infty)$
10:         for all $i > b$ and $\|\mathbf{L}^{\text{inv}}_{ib} - \mathbf{V}_{i:}\|_\infty > \gamma\tau_{\text{inv}}$ do
11:             $\bar{\mathbf{L}}^{\text{inv}}_{ib} \leftarrow \mathbf{V}_{i:}$
12:         end for
13:     end for
14:     $\mathbf{L}^{\text{inv}} \leftarrow \bar{\mathbf{L}}^{\text{inv}}$
15: end while
16: $\mathbf{L}^{\text{inv}} \leftarrow$ sparsify($\mathbf{L}^{\text{inv}}$)
17: Compute the block graph $(\mathbf{L}^{\text{inv}})^\intercal$
18: for $b = 1, \ldots, q$ parallel do
19:     $\boldsymbol{\Theta}^{\text{inv}}_{:b} \leftarrow (\mathbf{L}^{\text{inv}})^\top \mathbf{D}^{\text{inv}} \mathbf{L}^{\text{inv}}_{:b}$
20: end for
21: $\boldsymbol{\Theta}^{\text{inv}} \leftarrow$ sparsify($\mathbf{P}\boldsymbol{\Theta}^{\text{inv}}\mathbf{P}^\intercal$)
Output: $\boldsymbol{\Theta}^{\text{inv}}$
```

The computational efficiency and accuracy of this approach relies on (i) a small number of iterations $k$ in Horner's scheme (3.3.5), and (ii) the fact that both the inverse of the Cholesky factors $\mathbf{L}^{-1}$ and of the precision matrix $\boldsymbol{\Theta}^{-1}$ are sparse, or can be accurately approximated as such. Even though both assumptions depend on the dataset in question, in our numerical experiments presented in Chapter 4 we observe that our approximate inversion routine is accurate and performant for a variety of synthetic and real-world instances.

Computing the product of the factors in (3.3.2) is the second significant computational challenge in the estimation of $\boldsymbol{\Theta}^{\text{inv}}$. The kernel operation in this product is a sequence of sparse block matrix-matrix multiplications, with the individual products relying on dense matrix operations. We parallelize this portion of the algorithm and exploit the block structure induced by the supernodes of the Cholesky factorization described previously in order to accelerate the computation. Since the exact inversion of the block diagonal matrix $\mathbf{D}$ in (3.3.1) does not pose computational or storage problems, the parallelized block approximate matrix inversion Algorithm 3 comprises of two parts. First, the computation of $\mathbf{L}^{\mathrm{inv}}$ as per (3.3.5), and second the reconstruction of $\mathbf{\Theta}^{\mathrm{inv}}$ (3.3.1).

The algorithmic inputs are $\mathbf{L}$, $\mathbf{D}^{\mathrm{inv}} := \mathbf{D}^{-1}$, the permutation matrix $\mathbf{P}$ provided by CHOLMOD, and the threshold $\tau_{\mathrm{inv}}$ that is used to drop entries of negligible magnitude. We initialize in step 1 the variables and a buffer $\mathbf{u}$ of size equal to the maximum numbers of threads. We evaluate with this buffer the convergence of the approximate Neumann iteration in step 2, and store the maximum magnitude of the incremental updates in step 9. As illustrated in Figure 3.3a, each lower diagonal block of $\mathbf{L}$, $\mathbf{E}$, and $\mathbf{L}^{\mathrm{inv}}$ is stored as a dense matrix with compressed rows and contiguous columns. The parallel portion of the algorithm begins in step 3, where we consider $q$ total diagonal blocks in $\mathbf{L}$, and allocate each thread $r$ to a specific block $b$. Additionally, we initialize here a dense buffer $\mathbf{V}$ of size $t \times s$ that will store the multiplication updates. The sparse matrix-matrix multiplication is performed in steps 5—8, and illustrated in Figure 3.3b. Each block $\mathbf{E}_{:b}$ consists of a dense buffer $\mathbf{Q}$ with rows $i_1, \dots, i_r$ and columns $j, j+1, \dots, j+s-1$, that captures the nonzero rows and the contiguous columns within block $b$. We compute the partial matrix multiplication and update the buffer as $\mathbf{V} \leftarrow \mathbf{V} + \mathbf{R}_c\mathbf{Q}$ for each block column $c$ of $\mathbf{L}^{\mathrm{inv}}$ that intersects with $\{i_1, \dots, i_r\}$. In order to do so, we initially gather the associated columns of $c$ into a buffer $\mathbf{R}$ and then we compute $\mathbf{V} \leftarrow \mathbf{RQ}$. In steps $10-12$ the dropout rule 3.3.8 is applied, and the remaining rows of $\{i_1, \dots, i_t\}$ and columns $j, \dots, j+s-1$ from $\mathbf{V}$ are scattered to the new approximate inverse factor $\bar{\mathbf{L}}^{\mathrm{inv}}_{:b}$.

Before approximating the inverse of the precision matrix, we discard entries smaller than $\tau_{\mathrm{inv}}$ from $\mathbf{L}^{\mathrm{inv}}$ in step 16 and then compute the block graph of $(\mathbf{L}^{\mathrm{inv}})^{\intercal}$ in step 17, based on the partitioning induced by the diagonal blocks. Subsequently, we store the start of the dense supernodal diagonal blocks, if any exist. This is followed by the computation of the sparse approximate $\mathbf{\Theta}^{\mathrm{inv}}$ in steps $18-20$. Notice that $(\mathbf{L}^{\mathrm{inv}})^{\intercal}$ is a unit upper triangular matrix with dense superdiagonal blocks that stores only the nonzero columns. When performing the matrix multiplications in step 18 the nonzero rows $i_1, \dots, i_r$ of $\mathbf{L}^{\mathrm{inv}}_{:b}$ are associated with the diagonal blocks $c_1, \dots, c_r$ of $\mathbf{D}^{\mathrm{inv}}$ and block columns $c_1, \dots, c_r$ of $(\mathbf{L}^{\mathrm{inv}})^{\intercal}$, which now can be easily accessed via the computed block graph. Similarly to our approach at step 7, we exploit the symmetry in the computation, and compute only the lower block triangular part of $\mathbf{\Theta}^{\mathrm{inv}}$. Our last operation, before returning the estimated $\mathbf{\Theta}^{\mathrm{inv}}$, is to apply the permutation matrix $\mathbf{P}$, and to discard entries smaller than the selected threshold $\tau_{\mathrm{inv}}$.

For the parallel portion of Algorithm 3, the computation associated with each block cannot be determined before evaluating $\mathbf{R}$ for each block column $c$ of $\mathbf{L}^{\mathrm{inv}}$, and each thread $r$ operates on a different block of data. We thus consider a static OpenMP scheduling.

### 3.3.3 Block coordinate descent update

The minimization of the negative log-likelihood objective (3.1.1) requires the computation of the Newton direction $\mathbf{D}_{ij}$ in step 6 of Algorithm 2 for the set of free indices $\{i, j\} \in \mathcal{I}_{\text{free}}$. A block coordinate descent update can be computationally infeasible if the set $\mathcal{I}_{\text{free}}$ is large, and particularly if $\mathbf{\Theta}^{\text{inv}}$ is not a sparse matrix. We accelerate this segment of the algorithm with a blocking approach that reorders the indices of the matrix in order to increase contiguous data patterns and to reduce cache missses.

Similar to Section 3.1.3, to determine the Newton direction we compute

$$\begin{aligned}
\mathbf{A}_{ij} &= \mathbf{W}_{ij}^2 + \mathbf{W}_{ii}\mathbf{W}_{jj}, \\
\mathbf{B}_{ij} &= \mathbf{S}_{ij} - \mathbf{W}_{ij} + \mathbf{W}_{i:}\mathbf{D}_{ij}\mathbf{W}_{:j}, \text{and} \\
\mathbf{\Gamma}_{ij} &= \mathbf{\Theta}_{ij} + \mathbf{D}_{ij},
\end{aligned} \tag{3.3.9}$$

limited to all indices $\{i, j\} \in \mathcal{I}_{\text{free}}$. Instead of accessing the indices in the free set in a randomized manner, we consider a regrouped approach such that $\{i_1, j\}, \ldots, \{i_l, j\}$ refer to the same column $j$, and sort the indices in ascending order $i_1 < i_2 < \cdots < i_l$. This approach allows us to access the columns $\mathbf{W}_{:j}$, $\mathbf{S}_{:j}$, and $\mathbf{\Theta}_{:j}$ once in an ascending row of indices, and compares favorably from a computational perspective to a randomized access approach that requires recomputing these quantities. Additionally, this access pattern enables the efficient computation of the matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{\Gamma}$ in (3.3.9), and the update of $\mathbf{B}$ and $\mathbf{\Gamma}$ whenever the new Newton direction $\mathbf{D}$ is found.

## 3.4 Sparse $M$-matrix estimation

As outlined in the previous Chapter 2, $M$-matrices are part of the set

$$\mathcal{S}_M = \left\{ \mathbf{\Theta} \in \mathbb{R}^{p \times p} | \mathbf{\Theta}_{ij} = \mathbf{\Theta}_{ji} \leqslant 0 \; \forall i \neq j, \;\; \mathbf{\Theta} \succ 0 \right\}, \tag{3.4.1}$$

and can be considered as precision matrices whose partial correlations

$$-\mathbf{\Theta}_{ij} / \sqrt{\mathbf{\Theta}_{ii}\mathbf{\Theta}_{jj}}, \; i \neq j, \tag{3.4.2}$$

are all non-negative [10]. The GMRF corresponding to a precision matrix of that form is referred to as attractive [104], and the constrained GLASSO estimator for $M$-matrices is defined as

$$\widehat{\mathbf{\Theta}} = \underset{\mathbf{\Theta} \in \mathcal{S}_M}{\arg\min}\ f(\mathbf{\Theta}) + \|\mathbf{\Lambda} \odot \mathbf{\Theta}\|_1. \tag{3.4.3}$$

In this section we present two algorithms developed for the MLE of $M$-matrices emerging from high dimensional datasets. Our contributions build on top of the SQUIC library for large scale precision matrix estimation. In Section 3.4.1 we present the SQUIC-fit algorithm, an unconstrained approach to $M$-matrix estimation based on two consecutive $\ell_1$ regularized optimization problems. Then, in Section 3.4.2 we introduce SQUIC-sqp, a constrained sequential quadratic programming approach for the solution of problems of the form (3.4.3).

### 3.4.1 A post processing approach

The first learning algorithm of $M$-matrices in the set (3.4.1) that we present hereby can be considered as an unconstrained $\ell_1$ regularized technique. In SQUIC-fit we do not enforce additional sign constraints in the estimation of the precision matrix, but instead follow a post processing approach coupled with the utilization of a matrix sparsity parameter in order to obtain the graphical structure of non-negatively correlated variables. Our approach consists of two consecutive estimations of precision matrices $\widehat{\mathbf{\Theta}}^{(1)}, \widehat{\mathbf{\Theta}}^{(2)}$ that are solutions of a graphical lasso problem. The first precision matrix $\widehat{\mathbf{\Theta}}^{(1)}$ is estimated with the aid of a scalar regularization parameter $\lambda$ that penalizes equally all the variables. The entries of the resulting precision matrix are utilized in order to estimate the binary graphical structure of the non-negatively correlated variables in the data $\mathbf{Y} \in \mathbb{R}^{p \times n}$ under question. The second precision matrix $\widehat{\mathbf{\Theta}}^{(2)}$ is estimated with a matrix sparsity parameter $\mathbf{\Lambda}$ that encodes this graphical structure.The final $M$-matrix is then extracted by post-processing the entries of $\widehat{\mathbf{\Theta}}^{(2)}$.

An outline of the algorithmic scheme for SQUIC-fit is presented in Algorithm 4. In step 1 we solve the $\ell_1$-regularized negative log-likelihood problem, that is,

$$\widehat{\mathbf{\Theta}}^{(1)} = \underset{\mathbf{\Theta} \succ 0}{\arg\min} \left\{ -\log\det \mathbf{\Theta} + \operatorname{tr} \mathbf{S}\mathbf{\Theta} + \lambda \|\mathbf{\Theta}\|_1 \right\}. \tag{3.4.4}$$

The scalar tuning parameter $\lambda$ is set such that the resulting graph is sparse, and its values usually adjust the regularization according to the number of variables

**Algorithm 4** SQUIC-fit

```
Input: data $\mathbf{Y}$, tuning parameters $\lambda, \eta$, thresholds $\kappa, \tau$
1: estimate : $\widehat{\mathbf{\Theta}}^{(1)}$                          ➜ acc. (3.4.4)
2: Build graphical bias $\mathbf{G}$                                      ➜ acc. (3.4.5)
3: Build matrix regularization parameter $\mathbf{\Lambda}$               ➜ acc. (3.4.7)
4: estimate : $\widehat{\mathbf{\Theta}}^{(2)}$                          ➜ acc. (3.4.6)
5: Build $M$-matrix $\widehat{\mathbf{\Theta}}$                           ➜ acc. (3.4.8)
Output: $\widehat{\mathbf{\Theta}}$
```

$p$ and the number of features $n$. Then, in step 2 we estimate the structure of the negative off-diagonal entries of $\widehat{\mathbf{\Theta}}^{(1)}$ as

$$\mathbf{G}_{ij} = \begin{cases} 0, & \text{if } i = j, \\ \mathbf{I}\Big(-\widehat{\mathbf{\Theta}}^{(1)}_{ij} > \kappa\Big), & \text{if } i \neq j. \end{cases} \tag{3.4.5}$$

The thresholding parameter $\kappa \geqslant 0$ is chosen sufficiently small so that all negative off-diagonal elements $\widehat{\mathbf{\Theta}}^{(1)}_{ij}$ of significant magnitude are detected. These values correspond to an attractive GMRF, and capture, for nonzero entries, the notion of positive correlation between two nodes (variables) $i, j$ of the graph.

In steps $3-4$ we subsequently utilize the graphical structure of $\mathbf{G} \in \mathbb{R}^{p\times p}$ in the composition of the matrix tuning parameter $\mathbf{\Lambda}$ for solving

$$\widehat{\mathbf{\Theta}}^{(2)} = \underset{\mathbf{\Theta}\succ 0}{\arg\min} \Big\{ -\log\det\mathbf{\Theta} + \operatorname{tr}\mathbf{S}\mathbf{\Theta} + \|\mathbf{\Lambda}\odot\mathbf{\Theta}\|_1 \Big\}. \tag{3.4.6}$$

The matrix sparsity parameter is composed as

$$\mathbf{\Lambda}_{ij} = \begin{cases} \eta & \text{for} \quad \mathbf{G}_{ij} \neq 0, \\ \lambda & \text{for} \quad \mathbf{G}_{ij} = 0. \end{cases} \tag{3.4.7}$$

where $\eta < \lambda \in \mathbb{R}$, thus the regularization matrix $\mathbf{\Lambda}$ effectively uses the sparsity pattern of $\mathbf{G}$ as a graphical bias in the estimation of the structure of $\widehat{\mathbf{\Theta}}^{(2)}$. The final step 5 of SQUIC-fit involves a post-processing procedure to construct the $M$-matrix from the entries of $\widehat{\mathbf{\Theta}}^{(2)}$. The final matrix $\widehat{\mathbf{\Theta}}$ is formed by selecting the structure and the weights of the non-positive off-diagonal entries of the estimated

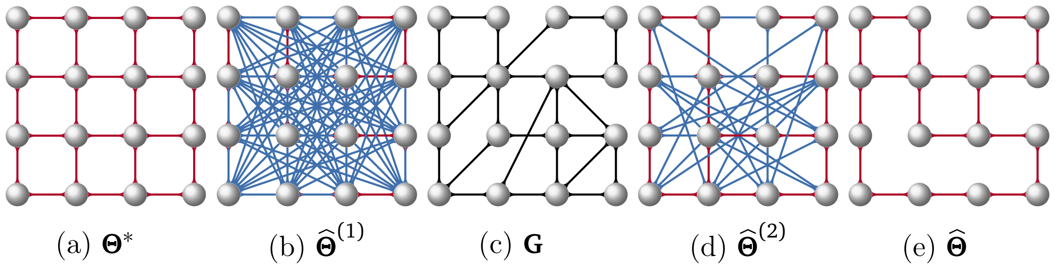

(a) $\mathbf{\Theta}^*$
(b) $\widehat{\mathbf{\Theta}}^{(1)}$
(c) $\mathbf{G}$
(d) $\widehat{\mathbf{\Theta}}^{(2)}$
(e) $\widehat{\mathbf{\Theta}}$

Figure 3.4: *An illustrated example of the key algorithmic operations of SQUIC-fit. Red edges correspond to negative off-diagonal entries $\mathbf{\Theta}_{ij} < 0$, and blue edges to positive $\mathbf{\Theta}_{ij} > 0$, $i \neq j$. (a) The true underlying structure of the $M$-matrix. (b) The first estimated precision matrix with the scalar regularization parameter $\lambda$. (c) The graphical bias of the negative off-diagonals. (d) The second estimated precision matrix with the matrix regularization term $\mathbf{\Lambda}$. (e) The final estimated $M$-matrix after the post-processing step.*

precision matrix $\widehat{\mathbf{\Theta}}^{(2)}$ as

$$\widehat{\mathbf{\Theta}}_{ij} = \begin{cases} 0, & \text{if } i = j, \\ \mathbf{I}\left(-\widehat{\mathbf{\Theta}}^{(2)}_{ij} > \kappa\right)\widehat{\mathbf{\Theta}}^{(2)}_{ij}, & \text{if } i \neq j. \end{cases} \tag{3.4.8}$$

In both steps 1 and 4 the SQUIC algorithm is executed up to a convergence tolerance $\tau$. These key operations of SQUIC-fit are illustrated in Figure 3.4, where we attempt to retrieve a grid structure with $p = 16$ nodes from $n = 100$ samples drawn from the zero-mean multivariate Gaussian distribution $\mathbf{Y} \sim \mathcal{N}(0, \mathbf{\Theta}_*^{-1})$, with $\mathbf{\Theta}_*$ being the true underlying $M$-matrix of the grid.

Incorporating available connectivity information for the graphical structure of non-negatively correlated variables in the data $\mathbf{Y}$ is also possible in the Algorithm 4. In this case the structure of $\mathbf{G}$ is part of the input, and the algorithm is reduced to steps $3-5$.

### 3.4.2 A constrained optimization approach

The second learning algorithm that we introduce, SQUIC-sqp, is a constrained approach for the estimation of $M$-matrices based on sequential quadratic programming. The minimization of the $\ell_1$ regularized log-likelihood, restricted to the set $\mathcal{S}_M$ in (3.4.1), can be reformulated as the constrained minimization problem

$$\underset{\boldsymbol{\Theta}\succ 0}{\text{minimize}} \left\{ f(\boldsymbol{\Theta}) + \|\boldsymbol{\Lambda} \odot \boldsymbol{\Theta}\|_1 \right\}, \tag{3.4.9a}$$

$$\text{subject to } \boldsymbol{\Theta}_{ij} \leqslant 0 \text{ for all } i \neq j. \tag{3.4.9b}$$

In order to approximate this optimization problem locally with a sequence of quadratic reformulations, we employ again the second-order Taylor expansion of $f$ around $\boldsymbol{\Theta}$, similarly to our approach for precision matrices (3.1.6). The free index set $\mathcal{I}_{free}$, as defined in (3.1.13), restricts the optimization problem (3.4.9) to pairs of indices that satisfy $|\mathbf{S}_{ij} - \boldsymbol{\Theta}_{ij}^{-1}| > \boldsymbol{\Lambda}_{ij}$, or pairs for which $\boldsymbol{\Theta}_{ij}$ is potentially nonzero. Therefore, in the context of $M$-matrix estimation, the matrix can be assumed to have either positive sign for the indices $i = j$ in the diagonal, or negative sign for the $i \neq j$ off-diagonal elements. This observation allows us to rewrite the local regularized quadratic objective as

$$\begin{aligned} q(\boldsymbol{\Delta}) = \operatorname{tr}(\mathbf{S} - \mathbf{W})\boldsymbol{\Delta} + \frac{1}{2}\operatorname{tr}\mathbf{W}\boldsymbol{\Delta}\mathbf{W}\boldsymbol{\Delta} \\ + \sum_i \boldsymbol{\Lambda}_{ii}(\boldsymbol{\Theta}_{ii} + \boldsymbol{\Delta}_{ii}) - \sum_{i\neq j} \boldsymbol{\Lambda}_{ij}(\boldsymbol{\Theta}_{ij} + \boldsymbol{\Delta}_{ij}). \end{aligned} \tag{3.4.10}$$

where $\mathbf{W} = \boldsymbol{\Theta}^{-1}$ is again the inverse of the $M$-matrix. This local approximate function is quadratic and differentiable, and, similarly to SQUIC-fit, prior knowledge on the latent graphical structure can be incorporated in the objective function through a matrix sparsity parameter of the form (3.4.7). In order to account for the constraints of the minimization (3.4.9b), we reformulate the problem as

$$\widehat{\boldsymbol{\Theta}} = \arg\min_{\boldsymbol{\Delta}} \ q(\boldsymbol{\Delta}) \tag{3.4.11a}$$

$$\text{subject to } \boldsymbol{\Theta}_{ij} + \boldsymbol{\Delta}_{ij} \leqslant 0 \text{ for all } i \neq j, \tag{3.4.11b}$$

The differentiable objective, in conjuction with the linear inequality constraints, enables the usage of sequential quadratic programming (SQP) [64, 105] for the solution of this problem.

At the core of the SQP method lies a successive distinction between constraints that are considered active and inactive in the optimization. The active ones refer to $\boldsymbol{\Delta}_{ij}$ such that $\boldsymbol{\Delta}_{ij} \approx -\boldsymbol{\Theta}_{ij}$, and the inactive ones such that $\boldsymbol{\Delta}_{ij} \ll -\boldsymbol{\Theta}_{ij}$. Since (3.4.11b) corresponds to box constraints, the Karush–Kuhn–Tucker (KKT) system for the unconstrained variables can be solved using a straight projection, i.e., systems (3.4.11) and (3.4.12) are restricted to the diagonal entries $\boldsymbol{\Delta}_{ii}$, which are always unconstrained, and the inactive off-diagonal entries $\boldsymbol{\Delta}_{ij}$.

For the active constraints, the entries $\mathbf{\Delta}_{ij}$ enter as inhomogeneity. We denote the affiliated index subsets of $\mathcal{I}_{\text{free}}$ by $\mathcal{I}_u$ for the unconstrained indices and by $\mathcal{I}_c$ for the active ones. The minimization problem is therefore reduced to solving the projected system $\nabla q(\mathbf{\Delta}) = 0$ restricted to $\mathcal{I}_u$, where the gradient of the quadratic function $q(\mathbf{\Delta})$ in (3.4.10) reads

$$\nabla q(\mathbf{\Delta}) = \mathbf{W\Delta W} + \mathbf{S} - \mathbf{W} + \mathbf{\Lambda} \odot (2\mathbf{I} - \mathbf{ee}^{\mathsf{T}}). \tag{3.4.12}$$

Note that the matrix associated with the term $\mathbf{W\Delta W}$ in (3.4.12) is equivalent to the Kronecker product $\mathbf{W}\otimes\mathbf{W}$, and the size of the resulting matrix is squared with respect to $\mathbf{W}$. Additionally, the positive-definiteness of $\mathbf{W}$ leads to $\mathbf{W} \otimes \mathbf{W} \succ 0$. We are therefore solving systems with the submatrix of $\mathbf{W} \otimes \mathbf{W}$ belonging to the unconstrained indices $\{i, j\} \in \mathcal{I}_u$.

In the solution of the linear system $\nabla q(\mathbf{\Delta}) = 0$ the sought matrix $\mathbf{\Delta} \in \mathbb{R}^{p\times p}$ is treated as a vector in a subspace of $\mathbb{R}^{p^2}$ defined on $\mathcal{I}_u$. The prohibitively large size of this system, that is quadratic with respect to the dimensions $p$, even when projected to the set of unconstrained variables renders an iterative approach the only viable option. We use the preconditioned conjugate gradient (PCG) method [64] with diagonal preconditioning, implemented in a parallel fashion with static OpenMP scheduling. It should be noted that the vectors used in the PCG method are sparse symmetric matrices stored in compressed column storage format (CSC), and as a result, the data is naturally partitioned. The parallelization is performed along the set of columns of the underlying matrices. For details regarding the benefits of parallelizing the conjugate gradient method we refer to [106], and for an implementation of the matrix-by-vector product for matrices stored in CSC format to [107].

An outline of the SQP part of our implementation is offered in Algorithm 5. The SQP method successively evaluates the computed $\mathbf{\Delta}_{ij}$, checks the constraints, activates and de-activates them until eventually the solution is computed. The set of unconstrained indices $\mathcal{I}_u$ is computed in step 3, and in step 4 the constrained ones are determined as $\mathcal{I}_{\text{free}} \setminus \mathcal{I}_u$, and the matrix $\mathbf{\Delta}$ is initialized. The parallel diagonally preconditioned CG for the solution of $\nabla q(\mathbf{\Delta})$ is called in step 5 with the indices that are part of $\mathcal{I}_u$ as unknowns, and the ones in $\mathcal{I}_c$ as constants. The existence of further descent directions is examined in steps $6-13$. Due to the simplicity of the box constraints in (3.4.11b), the Lagrangian multiplier $\mathbf{T}_{ij}$ of the augmented system

$$q(\mathbf{\Delta}) + \sum_{i\neq j} \mathbf{T}_{ij}(\mathbf{\Theta}_{ij} + \mathbf{\Delta}_{ij}) \tag{3.4.13}$$

is obtained by the negative gradient $\mathbf{T} = -\nabla q(\mathbf{\Delta})$ for all indices $\{i, j\} \in \mathcal{I}_c$. If the

**Algorithm 5** SQP-loop of SQUIC-sqp

```
Input: objective function $q(\mathbf{\Delta})$ from (3.4.10)
 1: $\mathbf{\Theta}_{\text{old}} \leftarrow \mathbf{\Theta}$
 2: while not satisfied do
 3:     Compute set $\mathcal{I}_u \subset \mathcal{I}_{\text{free}}$ of inactive constraints and diagonal indices
 4:     $\mathcal{I}_c \leftarrow \mathcal{I}_{\text{free}} \setminus \mathcal{I}_u$, $\mathbf{\Delta} \leftarrow 0$
 5:     Call PCG with diagonal preconditioning for solving $\nabla q(\mathbf{\Delta}) = 0$ from
        (3.4.12), restricted to variables associated with $\mathcal{I}_u$ as unknowns and $\mathcal{I}_c$
        as constants.
 6:     if $\mathbf{\Delta} \approx 0$ and there is no further descent direction then
 7:         break
 8:     else if $\mathbf{\Delta} \approx 0$ but there exists a further descent direction then
 9:         de-activate a promising constraint and update $\mathcal{I}_u$, $\mathcal{I}_c$
10:     else
11:         Compute $\nu \in (0, 1]$ such that $\mathbf{\Theta}_{ij} + \nu\mathbf{\Delta}_{ij} \leqslant 0$ for all $\{i, j\} \in \mathcal{I}_u$, $i \neq j$
12:         $\mathbf{\Theta} \leftarrow \mathbf{\Theta} + \nu\mathbf{\Delta}$
13:     end if
14: end while
15: $\mathbf{\Delta} \leftarrow \mathbf{\Theta} - \mathbf{\Theta}_{\text{old}}$
Output: $\mathbf{\Delta}$
```

update $\mathbf{\Delta} \approx 0$ we consider the signs of the Lagrange multipliers $\mathbf{T}_{ij}$ on the active set $\mathcal{I}_c$. If they are non-negative $\mathbf{T}_{ij} \geq 0$, the optimal $\mathbf{\Delta}$ has been found, while if there exist negative components $\mathbf{T}_{ij} < 0$, then there must be further descent directions. The index pair $\{i, j\}$ associated with the most negative entry of $\mathbf{T}$ is the first promising candidate to be de-activated. If the update $\mathbf{\Delta} \neq 0$ is nonzero, we perform a line-search to determine the largest step-length value $\nu$ in the range $(0, 1]$ for which all the constraints are satisfied, and update the $M$-matrix with $\mathbf{\Theta} \leftarrow \mathbf{\Theta} + \nu\mathbf{\Delta}$. Finally, in step 15 we compute the new update $\mathbf{\Delta} \leftarrow \mathbf{\Theta} - \mathbf{\Theta}_{\text{old}}$.

# Chapter 4

# Numerical results for graph learning

In this chapter we demonstrate the effectiveness of the previously introduced algorithms in the retrieval of graphical structures with a series of numerical experiments. We begin in Section 4.1 with results concerning the estimation of sparse precision matrices, and then in Section 4.2 we shift our attention to sparse $M$-matrices. To evaluate and analyze the efficacy of our methods we present results on both synthetic and real-world datasets. Synthetic cases are associated with ground-truth solutions and are useful for evaluating the accuracy of the retrieved graphs. Simultaneously, artificial data generation provides a suitable environment to investigate the strong scaling capabilities of our algorithms for an increasing number of dimensions $p$. Our real-world experiments, instead, are usually not equipped with labelled or annotated solutions, but represent challenging realistic large-scale problems whose solution depends on the accurate estimation of latent graphical structures.

All experiments presented in this chapter are conducted on a single node with 1 TB main memory and 4 Intel(R) Xeon E7-4880 v2 @ 2.5 GHz each with 15 cores per socket, totaling 60 cores, and the numerical values presented hereby are averaged over 10 runs.

## 4.1 Estimating precision matrices

This section outlines the analysis and test results that validate the performance, accuracy, and scalability of the SQUIC algorithm. We begin in Section 4.1.1 with tests on synthetic data to compare the accuracy and time-to-solution of SQUIC against various external state-of-the-art methods. Then, in Section 4.1.2 we measure the strong scaling capabilities of its individual algorithmic components. Last, in Section 4.1.3 we perform a real-world experiment, where we classify high dimensional DNA microarray data with limited samples with a linear discriminant analysis (LDA) study.

### Experimental setup

The following publicly available sparse precision matrix estimation packages are included in our comparative results:[1]

1. GLASSO [4]: A first order method based on coordinate descent updates for the solution of the graphical lasso problem.

2. BigQUIC [66]: A second order method based on quadratic approximation, that superseded QUIC [6], and solves issues associated with the memory footprint of the method.

3. EQUAL [71]: An approximation approach of the $\ell_1$ regularized negative log-likelihood with a trace-based quadratic loss function.

4. FASTCLIME [73, 75]: A constrained $\ell_1$ regularized method using the parametric simplex algorithm for the estimation of sparse inverse covariance matrices.

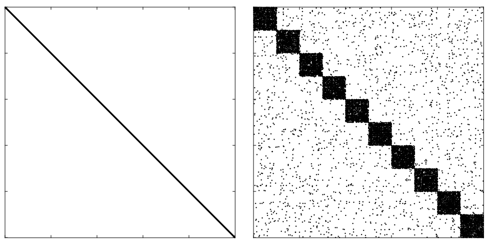
5. MDMC [76]: Approximation of the graphical lasso estimator through soft-thresholding of the sample covariance matrix and solving a second-order maximum determinant matrix completion problem.

We base our comparative results on two synthetic datasets generated from Gaussian distributions with a mean of zero and the following types of predefined true precision matrices:

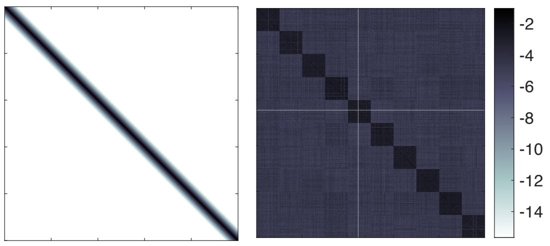
- *Tridiagonal*, $\mathbf{\Theta}^*_{(1)}$ — A tridiagonal matrix with off-diagonal values of $-0.5$ and 1.25 on the diagonal, and

- *Clusters*, $\mathbf{\Theta}^*_{(2)}$ — A random structured matrix representing a graphical structure of $p/100$ clusters of size 100 and an average degree of 20 with 90% of the edges contained within the clusters [108].

[1]BigQUIC, GLASSO, EQUAL and FASTCLIME are available as R packages at: https://cran.r-project.org/web/packages/BigQuic, https://cran.r-project.org/web/packages/glasso/, https://github.com/cescwang85/EQUAL, and https://github.com/cescwang85/EQUAL, respectively. The MATLAB code for MDMC is available at: https://ryz.ece.illinois.edu/software.html.



(a)



(b)

Figure 4.1: *The sparsity structure of the precision matrices of two synthetic datasets with dimension $p = 10^3$ are shown in (a) with the left panel being the tridiagonal matrix $\mathbf{\Theta}^*_{(1)}$ and the left one a clustered matrix $\mathbf{\Theta}^*_{(2)}$. The respective inverse of the precision matrices is shown in (b), where the colors represent the magnitude of the matrix values in* log *base* 10*. Note that both inverse matrices are dense; however, the tridiagonal matrix's inverse has exponentially decaying values in the off-diagonals.*

In what follows, we refer to the respective datasets with the terms described above. In all the tests outlined below, the number of samples is fixed at $n = 500$ and the convergence tolerance is $\tau = 10^{-4}$. In order to construct the input data $\mathbf{Y} \in \mathbb{R}^{p\times n}$, we consider a matrix $\mathbf{Z} \sim \mathcal{N}(0, \mathbf{I}) \in \mathbb{R}^{p\times n}$ of normally distributed and uncorrelated random variables. The synthetic datasets is then created as $\mathbf{Y} = \mathbf{ZL}$, where $\mathbf{L}$ is the lower-triangular Cholesky factor of the inverse of the true precision $\mathbf{\Theta}^*$.

These datasets are selected to highlight two key points. First, the accuracy of the introduced SQUIC algorithm is equivalent or better compared to the aforementioned packages. Second, SQUIC provides significant speedups in comparison to the other methods both in scenarios when the inverse $\mathbf{\Theta}^{-1}$ exhibits reduced sparsity, and when both $\mathbf{\Theta}$ and $\mathbf{\Theta}^{-1}$ can be approximated as being very sparse. We also note that the exact inverse of $\mathbf{\Theta}^*_{(1)}$ is dense but has exponentially decaying values as we move further from the diagonal. This property makes the tridiagonal dataset well suited for a sparse approximation of the inverse precision matrix via a thresholded Neumann approach (see Section 3.3.2 for further details). For $\mathbf{\Theta}^*_{(2)}$ we also have a dense exact inverse; however, the magnitude difference between large and small values is much smaller than in the tridiagonal example, and thus a dropout tolerance $\tau_{\text{inv}}$ does not prevent increased density in $\mathbf{\Theta}^{-1}$. This behavior is illustrated in Figure 4.1. For this reason we can expect that the clusters dataset will pose a more significant computational challenge in both the approximate matrix inversion and, in turn, the coordinate descent update.

The accuracy in the estimation of $\widehat{\mathbf{\Theta}}$ is measured in terms of $\mathrm{F-score} \in [0, 1]$,

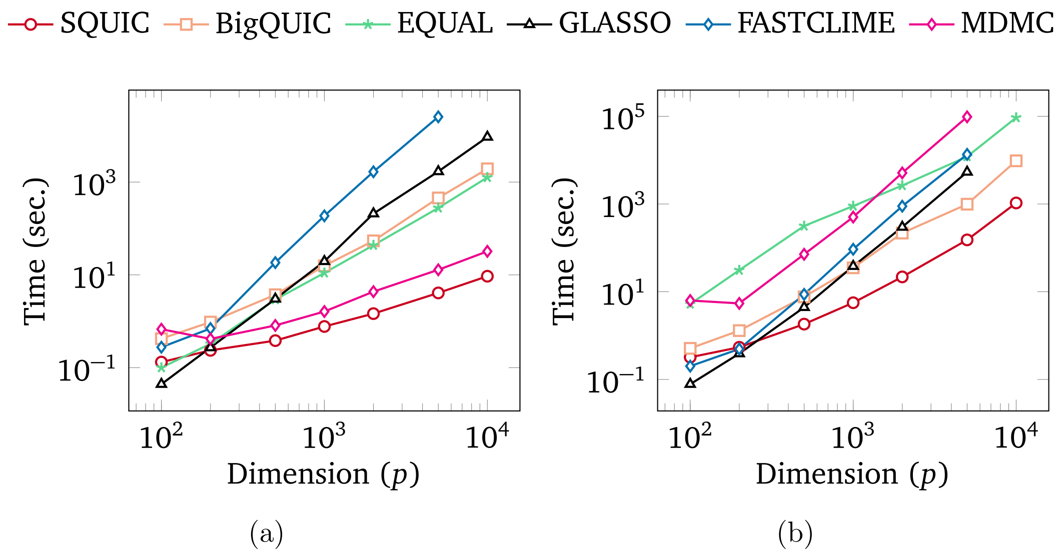


Figure 4.2: *A comparison of the runtimes of precision matrix estimation packages for (a) the tridiagonal and (b) the clusters dataset. At each dimension p, the runtime is the total compute time for a path of* 10 *sparsity parameters* $\lambda$.

with a value of $F = 1$ suggesting that the matrix has been fully recovered, while smaller values of $F$ suggest worse recovery success. We additionally consider the unsupervised clustering accuracy metric ACC $\in [0, 1]$ to measure the correctness of the classification solutions in Section 4.1.3. Again, a value of ACC $= 1$ suggests a perfect grouping of the nodes of the graph according to the true labels. Note that results regarding the relative error in the estimated $\widehat{\mathbf{\Theta}}$ have been previously presented in [7]. For further details on these accuracy metrics we refer to [109].

### 4.1.1 Comparisons with other methods

We present here results that validate the performance and accuracy of SQUIC, with a comparative analysis on a set of datasets of medium size. The results outlined in Figure 4.2 correspond to the two synthetic datasets with dimensions $10^2 \leqslant p \leqslant 10^4$. We penalize equally all entries of the precision matrix with a scalar paramemer $\lambda$. Each runtime represents a path of 10 different $\lambda$ values, which have been determined experimentally as the range for the best recovery of the true precision matrix. The timing results for EQUAL, FASTCLIME, GLASSO, and MDMC are excluded for $p > 6400$, as the runtimes exceed $2 \cdot 10^5$ seconds. We observe that SQUIC outperforms the competing algorithms when $p \geqslant 500$ for both datasets. For the tridiagonal dataset tests, presented in Figure 4.2a,

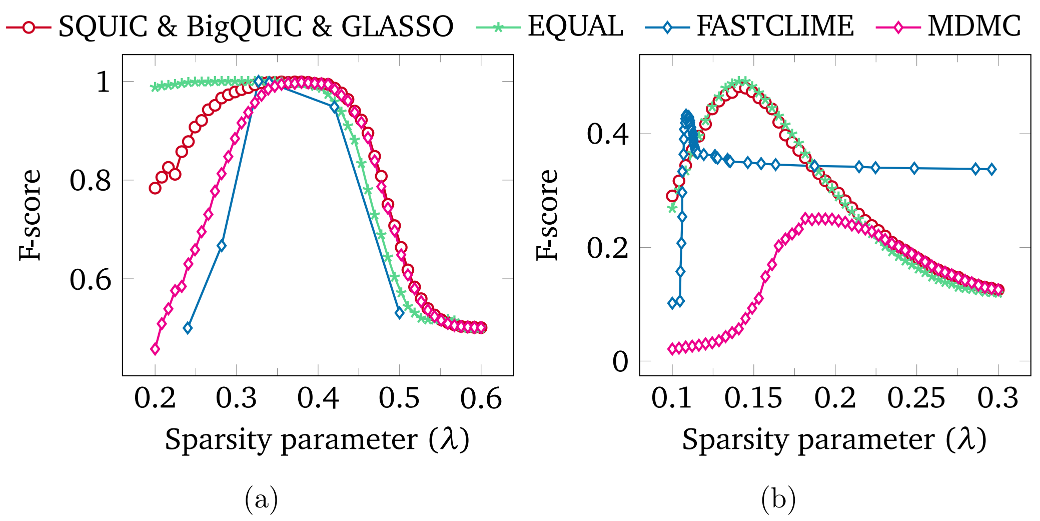


Figure 4.3: *A comparison of the F-score achieved by the different methods in the recovery of the precision matrices with respect to the regularization parameter $\lambda$, for (a) $\mathbf{\Theta}^*_{(1)}$ the tridiagonal and (b) $\mathbf{\Theta}^*_{(2)}$ the clusters datasets.*

SQUIC is consistently 5 times faster than the second fastest method (MDMC), and orders of magnitude faster than the other methods. While the tridiagonal dataset is a didactic example with approximately 3 nonzeros per row in the inverse of the precision matrix $\mathbf{\Theta}^{-1}$, the cluster dataset aims to resemble the structure of real-world data, with both $\mathbf{\Theta}$ and $\mathbf{\Theta}^{-1}$ being significantly denser. As shown in Figure 4.2b, the SQUIC algorithm is consistently orders of magnitude faster than the other methods for $p \geq 10^3$.

In Figure 4.3 we show the F-scores for the various algorithms for $p = 10^3$ dimensions with respect to a varying regularization parameter $\lambda$. Notice that SQUIC, BigQUIC, and GLASSO solve the same $\ell_1$ regularized MLE problem (3.1.1) and, thus, the recovered precision matrices have the same or similar F-score. Any differences between these MLE methods are due to numerical errors or the approximate inversion of the precision. Both are shown in our experiments to have a negligible impact on the estimated graphical structure. For the tridiagonal dataset in Figure 4.3a, we can see that all methods reach the maximum F-score of 1 for similar $\lambda$ values. For the clusters dataset in Figure 4.3b, the MLE methods reach a maximum F-score of 0.48 while EQUAL is slightly higher at 0.49. In contrast to the tridiagonal dataset, the remaining algorithms do not reach the same F-score, but remain in lower levels.

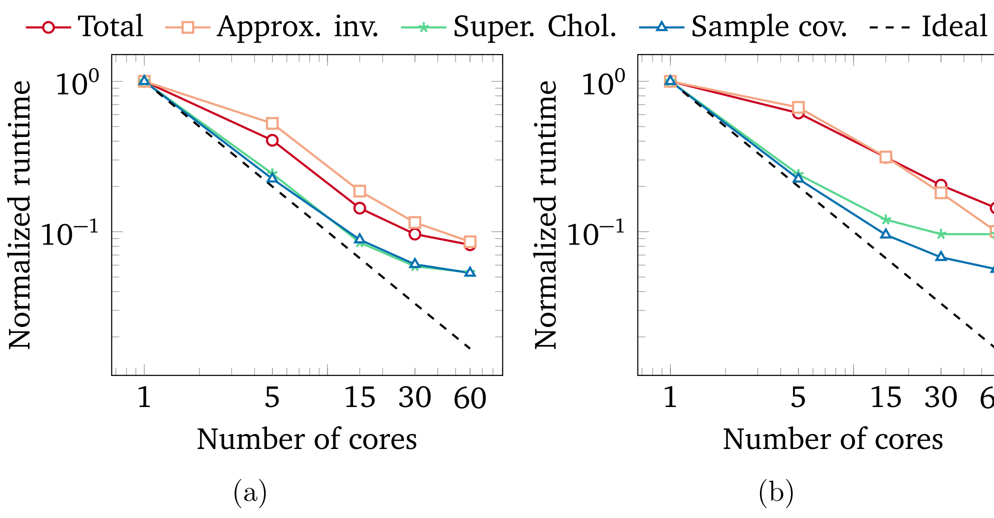


Figure 4.4: *Strong scaling for SQUIC at dimension* $p = 10^5$ *for (a) the tridiagonal and (b) the clusters datasets. The sparsity parameters used are* $\lambda = 0.5$ *and* $\lambda = 0.15$*, respectively. For all tests we consider* $\tau = \tau_{\text{inv}} = 10^{-4}$.

## 4.1.2 Scalability

In Figure 4.4 we present the strong scaling results of SQUIC and its internal parallel components discussed in Sections 3.2, 3.3, i.e., the introduced parallel block approximate matrix inversion, the supernodal sparse matrix factorization from the package CHOLMOD, and the sparse sample covariance matrix. In both plots, the dashed black line indicates the ideal scalability. For the tridiagonal dataset, shown in Figure 4.4a, the dominant algorithmic component is the approximate matrix inversion, which has a similar scalability profile to the total runtime. In contrast, the Cholesky factorization component accounts for very little of the runtime, and scale equivalently. In total, the parallel implementation of the algorithm exhibits 13 times speedup over its sequential variant.

Similarly to the tridiagonal case, the block approximate matrix inversion for the clusters dataset in Figure 4.4b requires over 79% of the serial runtime. As such, the overall scalability matches the approximate matrix inversion, which scales well to 60 cores with minimal degradation. Overall, the parallel execution of SQUIC is 7 times faster than its single-core execution for the clusters dataset.

## 4.1.3 Classification of microarray data

As a final case study, we utilize SQUIC to perform the classification of high dimensional microarray data concerning gene expressions with a linear discriminant analysis (LDA) approach. This study is intended to demonstrate (i) the performance of SQUIC in estimating precision matrices emerging from large real-world datasets and (ii) the increase in classification accuracy by utilizing a graphical bias in the matrix tuning parameter $\mathbf{\Lambda}$. The datasets used are a subset of the collection available in the R package datamicroarray [110], and are presented in Table 4.1. The challenging ratio between the available samples $n$ and the dimensionality $p$ renders such data unfavorable for LDA approaches. As a result, various approximate methods have been proposed that reduce the dimensionality of the problem [111, 112]. We demonstrate here that SQUIC enables applying traditional LDA with high classification scores within a reasonable time.

We estimate two precision matrices for these cases, one based on a scalar regularization parameter $\lambda$, and one utilizing a graphical bias in the composition of the matrix parameter $\mathbf{\Lambda}$. Subsequently, we apply the LDA method to group them into classes. Based on the classification scores in grouping the DNA microarray genes, we determine the accuracy in the computation of the inverse covariance. For a detailed analysis of the method, we refer to [113]. We follow the approach of [40, 114] and randomly select 70% of the genes from each class to form the training set, with the rest of the genes being in the testing set.

Initially, we normalize the input such that the sample covariance matrix $\mathbf{S}$ satisfies $\mathbf{S}_{ij} \in [-1, 1]$ with $\mathbf{S}_{ii} = 1$. To assign the matrix tuning parameter $\mathbf{\Lambda}$, we consider the minimal spanning tree (MST) [115] of the dataset. As a first step we generate a complete graph where the weights of the edges are defined as the Euclidean distance similarity measure (see, e.g., [116] for a similar approach). We note that using the Pearsons distance similarity provides similar accuracy results for the datasets under consideration, and refer to [117] for a relevant discussion. Next, we compute the adjacency matrix $\mathbf{W}$ of the MST for the complete graph, and use $\mathbf{W}$ to estimate the off-diagonal nonzero pattern of $\widehat{\mathbf{\Theta}}$. The MST provides a very sparse estimate for the off-diagonal elements of $\widehat{\mathbf{\Theta}}$, as $\mathbf{W}$ will have 2 nonzeros per row, or equivalently 2 edges per node. The nonzero pattern of $\mathbf{W}$ corresponds to low values of the used similarity measure, that is, the Euclidean distance (see e.g., [118] for a detailed discussion). In practice, we used $\mathbf{W} = \mathbf{W} + \mathbf{I}$ to also account for the diagonal values of $\widehat{\mathbf{\Theta}}$. The matrix regularization parameter $\mathbf{\Lambda}$ is then defined as

$$\mathbf{\Lambda}_{ij} = \begin{cases} \eta & \text{for} \quad \mathbf{W}_{ij} \neq 0, \\ \lambda & \text{for} \quad \mathbf{W}_{ij} = 0. \end{cases} \tag{4.1.1}$$

We set $\eta = 0.1$ and $\lambda = 0.95$ for all the tests. For comparison purposes, we retrieve another version of the precision matrix using a scalar regularization pa-

| **Dataset** | | **Specification** | | | |
|---|---|---|---|---|---|
| Name | Disease | $K$ | $n$ | $p$ | $\lambda$ |
| burczynski | Crohn's | 3 | 127 | 22,283 | 0.7 |
| yeoh | Leukemia | 6 | 248 | 12,625 | 0.8 |
| shipp | Lymphoma | 2 | 58 | 6,817 | 0.8 |
| alon | Colon Cancer | 2 | 62 | 2,000 | 0.6 |

Table 4.1: *The specifications of the microarray datasets under consideration. Starting from the left, we report the name, the type of disease the dataset describes, the number of ground-truth classes $K$, the number of samples $n$, the dimension $p$, and the scalar regularization parameter $\lambda$ that leads to the highest classification accuracy.*

rameter $\lambda$, penalizing equally all elements of $\widehat{\mathbf{\Theta}}$. In this case we perform a grid search for the values of $\lambda$ which provide the best classification, and report the results with this optimal parameter. The $\lambda$ values for the respective datasets are listed in Table 4.1.

In order to apply the LDA method, we consider $k \in \mathbb{N}_+$ to be the class index of a given sample, with each class having mean $\boldsymbol{\mu}_k$, and all classes sharing the same precision matrix. Assuming a Gaussian distribution $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_k, \mathbf{\Theta}^{-1})$ for the normalized medical data the linear discriminant function is defined as

$$\rho_k(\mathbf{x}) = \mathbf{x}^{\mathsf{T}}\mathbf{\Theta}\mu_k - \frac{1}{2}\boldsymbol{\mu}_k^{\mathsf{T}}\mathbf{\Theta}\mu_k + \log \hat{\pi}_k, \tag{4.1.2}$$

where $\hat{\pi}_k = n_k/n$ is the ratio of the number of samples of each class $n_k$ over the number of total number of samples $n$. The class-average vector for each class is computed as $\boldsymbol{\mu}_k = \left(\frac{1}{n_k}\right)\sum_{i\in k}\mathbf{x}_i$. Then, the class $C$ of a vector $\mathbf{x}$ is defined as

$$C(\mathbf{x}) := \arg\max_k \rho_k(\mathbf{x}). \tag{4.1.3}$$

We perform training and testing incrementally for each class $k$. In the training phase, we compute $\hat{\pi}_k$ and $\boldsymbol{\mu}_k$, and in the testing phase we evaluate $\rho_k$ and assign the samples to classes according to 4.1.3.

In Figure 4.5 we present the results comparing the matrix and scalar regularization parameter in the runtime and accuracy of the LDA method. We observe in Figure 4.5a that utilizing $\mathbf{\Lambda}$ may, in some cases, lead to a slight increase in the runtime. However, this increase is not significant when compared to the performance advantages of SQUIC over other state-of-the-art packages (see, e.g, results in Figure 4.2). The runtime for the largest dataset, burczynski with $p = 22283$,

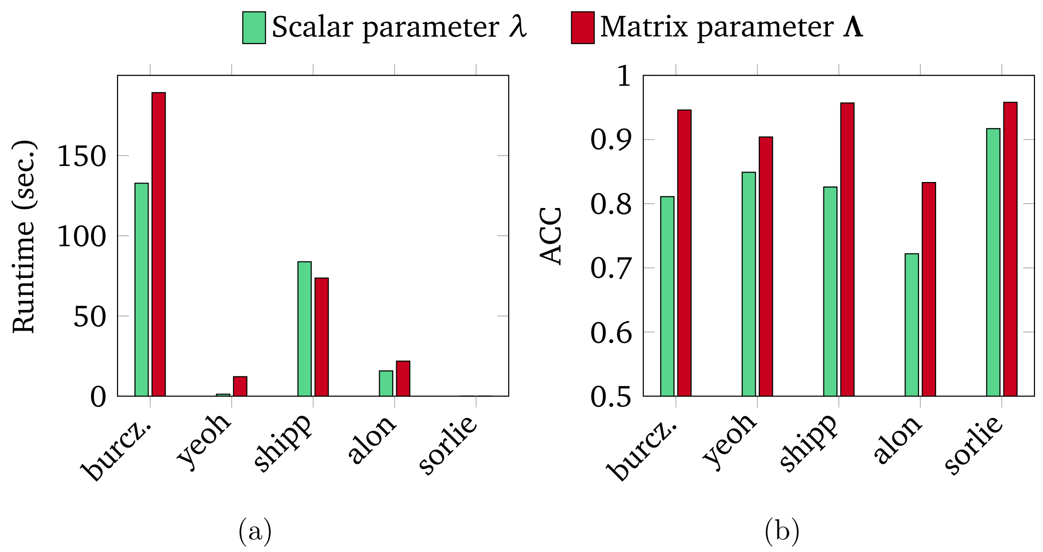


Figure 4.5: *A comparison of (a) the runtime of SQUIC using scalar and matrix regularization parameters, and (b) the LDA accuracy, measured in terms of ACC. The datasets used are outlined in Table 4.1. Note that the runtimes show here are for the computation of $\widehat{\mathbf{\Theta}}$ using SQUIC which is the majority of the total runtime. The runtime does not include the creation of the graphical bias emerging from the MST of the dataset.*

is estimated in 190 and 130 seconds when considering matrix and scalar parameters, respectively. Notably, in Figure 4.5b we see that the accuracy of all tests is increased when using $\mathbf{\Lambda}$ with the MST graphical bias. The classification scores are on average 14% higher when compared against the scalar counterpart.

## 4.2 Estimating $M$-matrices

This section outlines the analysis and test results that validate the performance, accuracy, and real-world applicability of the introduced $M$-matrix retrieval algorithms, SQUIC-fit and SQUIC-sqp. We begin in Section 4.2.1 with tests on synthetic data to compare the accuracy and time-to-solution against various external state-of-the-art methods, and then shift in Section 4.2.2 our attention to the way incorporating prior knowledge of the graphical structure of a dataset influences the accuracy of the $M$-matrix retrieval. Then, in Section 4.2.3 we identify the largest connected components of a graph emerging from the COVID-19 daily cases in the USA, and perform spectral clustering with the $M$-matrix of the largest component. Last, in Section 4.2.4 we classify image datasets of up to

$p = 7 \cdot 10^4$ dimensions based on the eigenvectors of the estimated $M$-matrices.

## Experimental setup

We will compare our methods against the following state-of-the-art graph learning packages:[2]

1. Combinatorial Graph Laplacian (CGL) [80]: Graph Laplacian estimation via an iterative block-coordinate descent algorithm. The authors decompose the original problem into a series of lower dimensional subproblems. We use a cycle of 100 row/column updates for the minimization of the objective function.

2. Structured Graph Learning (SGL) [59]: The graph Laplacian is estimated by converting combinatorial structural constraints into spectral constraints, and the resulting optimization problem is solved with an algorithm based on quadratic methods. The parameter controlling the quadratic approximation term is set at $\beta = 20$, as suggested by the authors.

Since both external algorithms directly estimate the combinatorial graph Laplacian in the set (2.4.7), we compare the accuracy of our proposed methods by estimating the $M$-matrix $\widehat{\mathbf{\Theta}}$ and then building the combinatorial graph Laplacian as

$$\widehat{\mathbf{L}}_{ij} = \begin{cases} \widehat{\mathbf{\Theta}}_{ij}, & \text{for all } i \neq j \\ -\sum_{r:r\neq i}^{p} \widehat{\mathbf{\Theta}}_{ir}, & \text{for all } i = j \end{cases} \tag{4.2.1}$$

The accuracy in the estimation of $\widehat{\mathbf{L}}$ is measured again in terms of F-score, and in terms of relative error (RE) defined as

$$\mathrm{RE} = \frac{||\widehat{\mathbf{L}} - \mathbf{L}_{\mathrm{true}}||_F}{||\mathbf{L}_{\mathrm{true}}||_F}, \tag{4.2.2}$$

where $\widehat{\mathbf{L}}$ is the estimated matrix and $\mathbf{L}_{\mathrm{true}}$ the true reference graph Laplacian. Then, in Section 4.2.4 the accuracy of the classification assignments is measured in terms of the unsupervised clustering accuracy (ACC $\in [0,1]$), and the normalized mutual information (NMI $\in [0,1]$) [109]. For both metrics a value of 1 suggests a perfect grouping of the nodes according to the true labels.

[2]The CGL code is available at: https://github.com/STAC-USC/Graph_Learning. The code for SGL is available as an R package at: https://cran.r-project.org/web/packages/spectralGraphTopology/index.html.

We base our results on two synthetic datasets generated from Gaussian distributions with a mean of zero and the following types of predefined graphical structures:

- A grid graph structure denoted as $\mathcal{G}_{\text{grid}}^{(p)}$, where $p$ is the number of nodes. Each node is connected to its four nearest neighbors (except the nodes at the boundaries).

- A random structured matrix denoted as $\mathcal{G}_{\text{clust}}^{(p)}$ representing a graphical structure of $p/100$ balanced clusters with an average node degree of 20 and with 90% of the edges contained within the clusters [108].

Edge weights are then randomly selected based on a uniform distribution from the interval $[0.1, 3]$. From these structures we generate an IGMRF model parametrized by the true graph Laplacian $\mathbf{L}_{\text{true}}$. From this IGMRF model $n$ samples are drawn from the degenerate zero-mean multivariate Gaussian distribution $\mathbf{y}_i \sim \mathcal{N}\left(\mathbf{0}, \mathbf{L}_{\text{true}}^{\dagger}\right)$, where $\mathbf{L}_{\text{true}}^{\dagger}$ is the Moore-Penrose pseudoinverse of $\mathbf{L}_{\text{true}}$. The sample covariance matrix $\mathbf{S}$ is computed as

$$\mathbf{S} = \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{y}_i - \bar{\mathbf{y}}_i\right)\left(\bar{\mathbf{y}}_i - \mathbf{y}_i\right)^{\intercal}, \text{ with } \bar{\mathbf{y}}_i = \frac{1}{n}\sum_{i=1}^{n}\mathbf{y}_i. \tag{4.2.3}$$

We follow the approach of [119] and define the regularization parameter as

$$\lambda = c \cdot \sqrt{\log(p)/n}, \tag{4.2.4}$$

unless specified otherwise. The scaling term $\sqrt{\log(p)/n}$ adjusts the regularization according to $p$ and $n$, and $c \in \mathbb{R}$ is based on experimental results. The convergence tolerance for all the methods is set to $\tau = 10^{-4}$, the threshold parameter for SQUIC-fit to $\kappa = 0$, and all the results reported hereby correspond to their mean value after 10 runs.

## 4.2.1 Comparisons with other methods

Our first round of numerical experiments is designed to evaluate and compare the accuracy of the two proposed algorithms in the retrieval of synthetic combinatorial graph Laplacian matrices emerging from the graphical structure of $\mathcal{G}_{\text{grid}}^{(64)}$ and $\mathcal{G}_{\text{clust}}^{(60)}$.

We generate 10 instances of each synthetic graph, and present in Figure 4.6 the mean accuracy results in terms of F-score and then in Figure 4.7 the associated

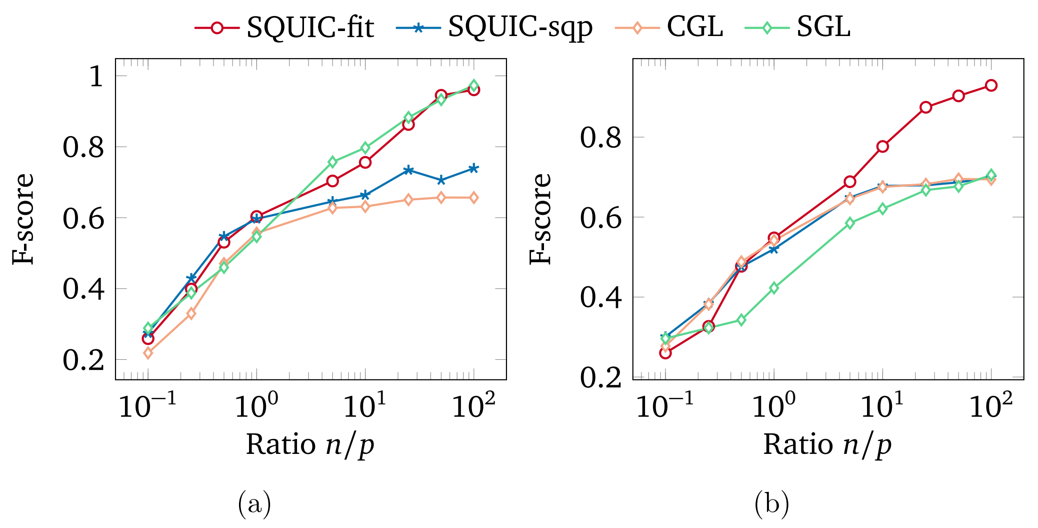


Figure 4.6: *Accuracy comparisons between the different combinatorial graph Laplacian estimation methods measured in terms of F-score for (a) the lattice grid graph $\mathcal{G}_{\text{grid}}^{(64)}$, and (b) the random clusters graph $\mathcal{G}_{\text{clust}}^{(60)}$.*

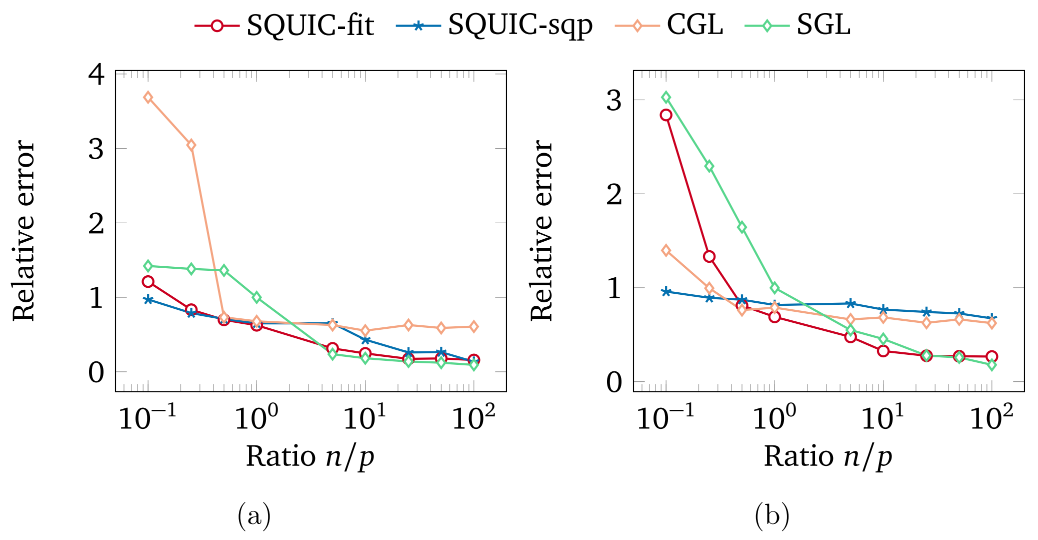


Figure 4.7: *Accuracy comparisons between the different combinatorial graph Laplacian estimation methods measured in terms of relative error (RE) for (a) the lattice grid graph $\mathcal{G}_{\text{grid}}^{(64)}$, and (b) the random clusters graph $\mathcal{G}_{\text{clust}}^{(60)}$.*

RE. The performance of the algorithms is compared for different ratios of sample sizes $n/p = \{0.1, 0.25, 0.5, 1, 5, 10, 25, 50, 100\}$. The parameter $c$ in (4.2.4) is selected independently at each level for each method, and corresponds to the

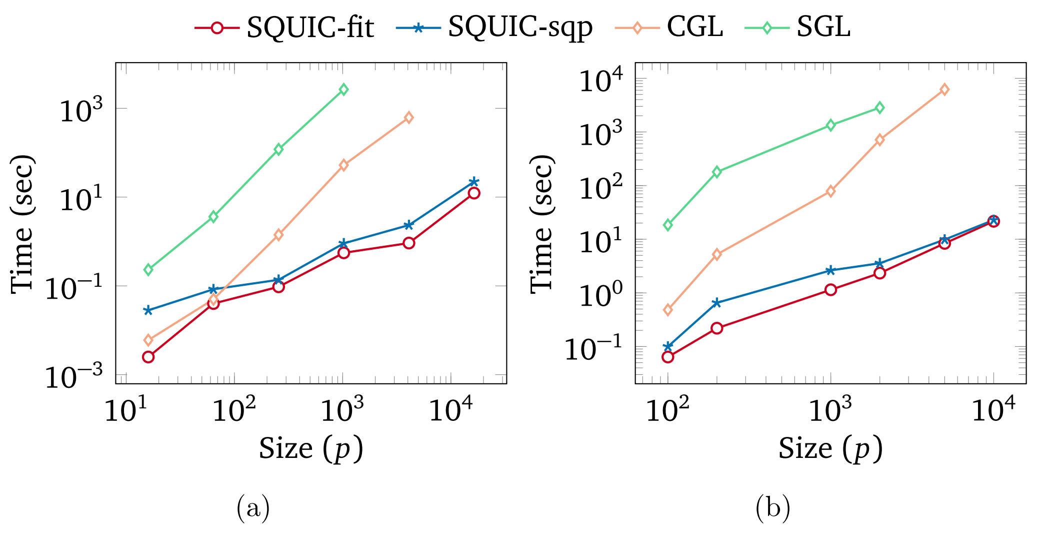


Figure 4.8: *Timing comparisons between the different graph Laplacian estimation methods when learning $\widehat{\boldsymbol{\Theta}}$ from synthetic graphs with an increasing number of $p$. (a) Results for the lattice grid graph $\mathcal{G}_{\text{grid}}^{(p)}$ with $p \in \{16, \dots, 16384\}$. (b) Results for the random clusters graph $\mathcal{G}_{\text{clust}}^{(p)}$ with $p \in \{100, \dots, 10000\}$.*

one that maximizes the F-score. Additionally, for SQUIC-fit we set in (3.4.7) $\eta = \lambda/10$.

For the lattice grid graphs (Figures 4.6a, 4.7a) SQUIC-fit achieves very high F-scores and low RE for higher sampling ratios $n/p > 1$, and remains competitive in the low sampling regimes $n/p \leq 1$ in terms of F-score. The accuracy of our post processing approach is similar to that of SGL, both in terms of F-score and RE. For low sampling ratios $n/p \leq 1$ SQUIC-sqp reports the best F-scores and RE, an expected behavior, as exploiting $M$-matrix constraints satisfies the model assumptions of attractive GMRFs. For the random clusters graphs (Figures 4.6b, 4.7b) SQUIC-fit achieves the highest F-scores for sampling ratios $n/p > 1$, with its RE reported being comparable with that of SGL. Our constrained approach SQUIC-sqp reports here similar F-score and RE with CGL for all sampling regimes.

We then proceed with a comparison of the runtimes of the methods under question. To this end, we consider a sequence of 6 true combinatorial graph Laplacian matrices $\mathbf{L}_{\text{true}}$ of increasing size. In particular, for the lattice grid graph $\mathcal{G}_{\text{grid}}^{(p)}$ we consider graphs of dimension $p = \{16, 64, 256, 1024, 4096, 16384\}$, and for the random clusters graph $\mathcal{G}_{\text{clust}}^{(p)}$ of $p = \{100, 200, 1000, 2000, 5000, 10000\}$. The number of samples is fixed in both cases at $n = 500$ and the parameter $c$ in (4.2.4) is again set for each method such that the best solution in terms of

F-score is reported at each $p$ level. We report these timing comparisons in Figure 4.8. The timing results for CGL and SGL are excluded when the runtimes exceed $10^4$ seconds. For the lattice grid graph (Figure 4.8a) SGL exceeds this time limit for $p \geq 4096$ and CGL for $p = 16384$. For the random clusters graph (Figure 4.8b) the time limit is exceeded by SGL at $p \geq 5 * 10^3$ and by CGL at $p = 10^4$. In Figure 4.8 we additionally observe that SQUIC-fit outperforms all competing algorithms across all dimensions for both graphical structures. This is an expected behaviour, as SQUIC-fit is the only unconstrained method included in the comparisons. SQUIC-sqp is outperformed by CGL only in the lattice grid experiments for the low dimensional cases $p \leq 64$. In both synthetic cases the SQUIC variants are up to 3 orders of magnitude faster that the competing methods for $p \geq 256$.

### 4.2.2 Incorporating graphical bias

We study here the recovery accuracy of the two introduced SQUIC algorithms when using prior graphical knowledge in the estimation of the sparse $M$-matrix $\widehat{\mathbf{\Theta}}$. The graphical structure of the bias $\mathbf{G}$ is defined as a corrupted version of the structure of the true graph Laplacian matrix $\mathbf{L}_{\text{true}}$. We control the degree of this corruption with a random symmetric sparse matrix $\mathbf{Z} \in \mathbb{R}^{p \times p}$ with $\delta \cdot |\mathbf{L}_{\text{true}}|/p$ number of nonzeros per row. The structure of $\mathbf{G}$ is then defined as

$$\mathbf{G}_{ij} = \begin{cases} 0, & \text{if } i = j, \\ \mathbf{I}\left(\mathbf{L}_{ij}^{\text{true}} > 0\right) + \mathbf{I}\left(\mathbf{Z}_{ij} > 0\right), & \text{if } i \neq j. \end{cases} \tag{4.2.5}$$

Notice that for $\delta = 0$ we retrieve the exact structure of $\mathbf{L}_{\text{true}}$, while for an increasing $\delta > 0$ the structure of $\mathbf{G}$ has an increasing number of noisy entries. The matrix regularization parameter is composed in similar fashion to (3.4.7) as

$$\mathbf{\Lambda}_{ij} = \begin{cases} \lambda_{\text{opt}}/b & \text{for} \quad \mathbf{G}_{ij} \neq 0, \\ b \cdot \lambda_{\text{opt}} & \text{for} \quad \mathbf{G}_{ij} = 0, \end{cases} \tag{4.2.6}$$

with $\lambda_{\text{opt}}$ being the scalar coefficient resulting in the highest F-score, and $b \in \mathbb{R}$ a scalar parameter controlling the effect of the matrix bias on the regularization. Larger values of $b$ in (4.2.6) result in the matrix bias $\mathbf{G}$ being more strictly enforced. We select $b = 2$ for a moderate influence of $\mathbf{G}$ on the estimated $\widehat{\mathbf{L}}$.

We consider two true graph Laplacian matrices emerging from the graphical structure of $\mathcal{G}_{\text{grid}}^{1024}$ and $\mathcal{G}_{\text{clust}}^{1000}$ with $n = 500$ number of samples. In Figure 4.9

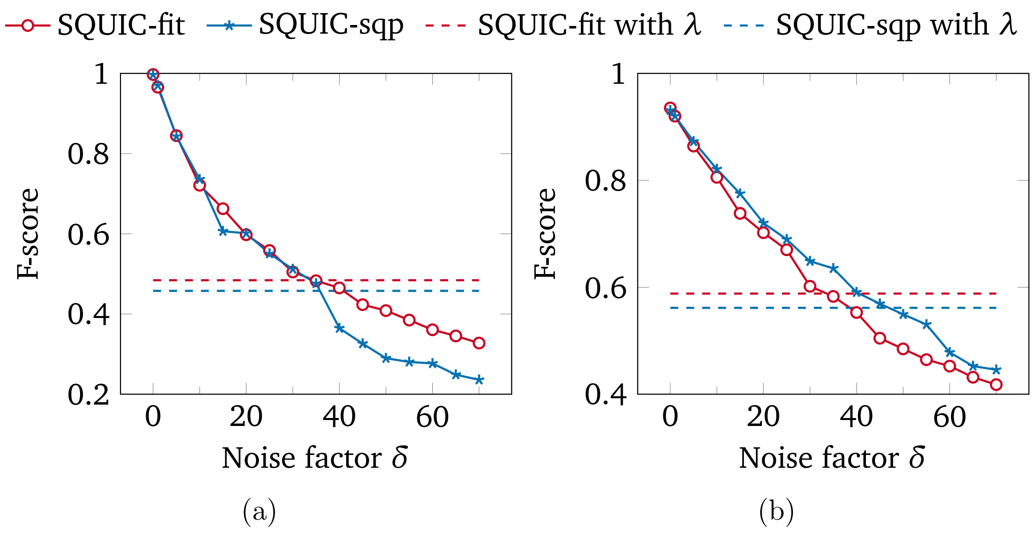


Figure 4.9: *Studying the effect that incorporating the structure of* $\mathbf{G}$ *in the matrix tuning parameter* $\mathbf{\Lambda}$ *has on the retrieval accuracy of* $\widehat{\mathbf{L}}$*. a) Results for the lattice grid graph* $\mathcal{G}_{\text{grid}}^{(1024)}$*. b) Results for the random clusters graph* $\mathcal{G}_{\text{clust}}^{(1000)}$*. We use* $n = 500$ *samples in both cases.*

we present the effect that an increasing noise factor $\delta = \{0, 1, \ldots, 70\}$ has on the retrieval accuracy of both SQUIC-fit and SQUIC-sqp in terms of F-score, and compare it with the retrieval accuracy achieved by the algorithms when using a scalar regularization parameter $\lambda$ with no graphical bias. Note that the corruption matrix $\mathbf{Z}$ has no effect on the retrieval accuracy when using the scalar parameter $\lambda$, as no graphical bias is utilized in the composition of the penalty term. The best F-scores achieved at the optimal scalar $\lambda_{\text{opt}}$ are represented with the horizontal dashed lines. Both introduced algorithms greatly outperform their scalar counterparts when taking into account a noisy graphical bias $\mathbf{G}$ in the matrix sparsity parameter $\mathbf{\Lambda}$ (solid lines). In particular, for the lattice grid graph $\mathcal{G}_{\text{grid}}^{(1024)}$ (Figure 4.9a) utilizing the graphical structure with SQUIC-fit improves the achieved F-score of 0.49 for noise factors of $\delta \leq 35$. For SQUIC-sqp improvements over the baseline of $\mathrm{F-score} = 0.46$ are observed for $\delta \leq 40$. For the random clusters graph $\mathcal{G}_{\text{clust}}^{(1000)}$ (Figure 4.9b) the baseline of SQUIC-fit is $\mathrm{F-score} = 0.59$, and is improved for $\delta \leq 30$, while for SQUIC-sqp the best F-score of 0.56 is improved when considering a graphical bias with $\delta \leq 45$.

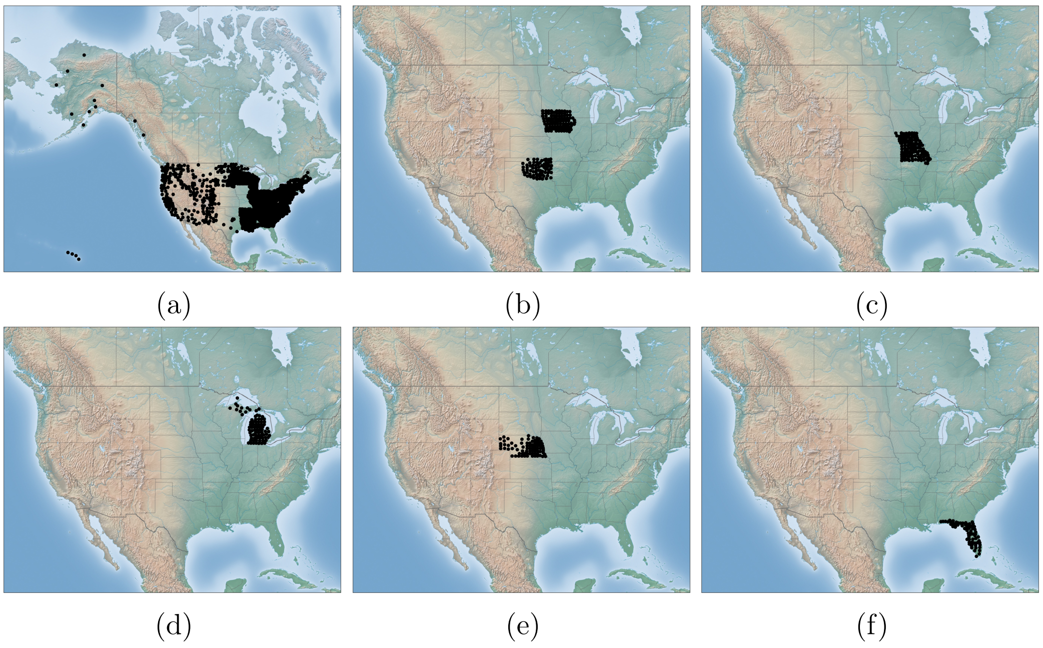


Figure 4.10: *Visualizing the US counties corresponding to the six largest connected components of the estimated $\widehat{\boldsymbol{\Theta}}$ with $p = 3209$ dimensions. The available $n = 671$ samples describe the number of COVID-19 daily instances. The six components are illustrated in a – f in descending order according to their size.*

### 4.2.3 Clustering of COVID-19 daily cases

For this real-world study we consider the publicly available data for the US confirmed daily cases, reported at the county level [120], and emphasize that results presented here are intended to highlight the capabilities of the proposed algorithms and not to propose any course of COVID-19 related actions.[3]

The dataset under consideration consists of $p = 3342$ counties and reports the number of daily COVID-19 cases $C$ for $n = 671$ days for the time window 22 January 2020 to 23 November 2021. Counties with a small total number of cases $\sum_n C < 100$ are discarded, resulting in $p = 3209$. We normalize the remaining daily cases by the number of residents per county in order to obtain information on the infection rate per capita.[4]

Subsequently, the $M$-matrix $\widehat{\boldsymbol{\Theta}}$ of the positively correlated counties is con-

[3]The COVID-19 Data Repository is provided by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University at https://github.com/CSSEGISandData/COVID-19. Puerto Rico municipalities are included.

[4]Demographic information of the USA at the county level is available at https://www.census.gov/data/datasets/time-series/demo/popest/2010s-counties-total.html.

structed with SQUIC-fit in 72 seconds and with SQUIC-sqp in 211 seconds, and the largest connected components of the resulting graphical structure are identified. For the SQUIC-sqp variant we use a scalar regularization parameter of $\lambda = 0.7$, and for the SQUIC-fit algorithm we set in (3.4.4) $\lambda = 0.7$ and $\eta = 2\lambda/3$ in (3.4.7). The matrices retrieved from both algorithms are almost identical, thus in what follows we report the results obtained with SQUIC-fit.

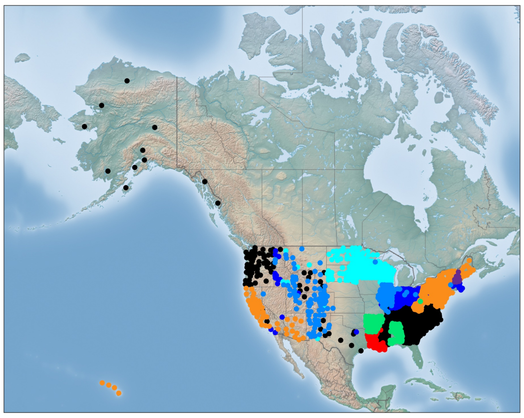
We illustrate in Figure 4.10 the six largest connected components of $\widehat{\mathbf{\Theta}}$. The largest component (Figure 4.10a) includes 1774 counties from the entire USA, the second one (Figure 4.10b) captures 165 counties from the states of Oklahoma and Iowa, the third one (Figure 4.10c) 113 counties from Missouri, the fourth one (Figure 4.10d) 81 counties from Michigan, the fifth one (Figure 4.10e) 79 counties from Nebraska, and the sixth largest connected component of $\widehat{\mathbf{\Theta}}$ (Figure 4.10f) includes 66 counties from the state of Florida. The clear geographic partition of the components $2-6$ demonstrates that SQUIC-fit successfully captures the positively correlated variables of the dataset.

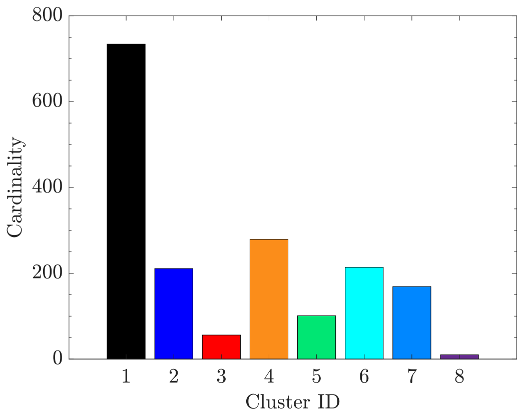
We proceed with an analysis of the clusters present in the largest connected component of $\widehat{\mathbf{\Theta}}$. This component is denoted as $\widehat{\mathbf{\Theta}}_{\mathrm{a}}$, and is used to build the random-walk normalized graph Laplacian $\widehat{\mathbf{L}}_{\mathrm{rw}} = \mathbf{D}^{-1}\widehat{\mathbf{L}}$, where $\widehat{\mathbf{L}}$ is defined as in (4.2.1) and $\mathbf{D}$ is the diagonal degree matrix satisfying $\mathbf{D}_{ii} = \widehat{\mathbf{L}}_{ii}$ for all $i$. After computing the eigenvalues $\lambda_k$ of $\widehat{\mathbf{L}}_{\mathrm{rw}}$ the number of natural clusters present in the dataset is estimated with the relative eigengap

$$\gamma_k = \frac{\lambda_{k+1} - \lambda_k}{\lambda_k}, \quad k \geqslant 2. \tag{4.2.7}$$

A high value of $\gamma_k$ indicates that $\widehat{\mathbf{\Theta}}_{\mathrm{a}}$ admits a natural decomposition into at least $k$ clusters [121]. In order to obtain discrete partitions, the eigenvectors corresponding to the $k$ smallest eigenvalues of $\widehat{\mathbf{L}}_{\mathrm{rw}}$ are clustered with the k-means algorithm with 20 orthogonal and 10 random initializations [122].

We present the clustering results using $\widehat{\mathbf{\Theta}}_{\mathrm{a}}$ in Figure 4.11. According to the relative eigengap, 8 distinct clusters are present in the subgraph. The locations of the counties of each cluster are illustrated in Figure 4.11a, and the cardinality of the respective clusters in Figure 4.11b. The largest cluster (black) captures 734 counties located mostly in the south and mideast states of Georgia, South and North Carolina, Virgina, Tenessee, Kentucky, and the northwest states of Washington and Oregon and Alaska. The second largest cluster (orange) includes 279 counties in the northeast states of West Virginia, Pennsylvania, New York, Maine, Delaware, the District of Columbia, the southwest state of California and Hawaii. The third largest cluster (cyan) has 214 counties mostly located in the


(a)



(b)

Figure 4.11: *Spectral clustering of the largest connected component of* $\widehat{\mathbf{\Theta}}$ *a) Geographical locations of the nodes belonging to each cluster. b) Cardinality of each cluster. (Best viewed in color.)*

neighboring states of North and South Dakota, Minnesota and Wisconsin. The fourth cluster in size (dark blue) is comprised of 211 nodes in the states of Massachusetts, Ohio and Indiana. The fifth cluster (light blue) includes 169 nodes mostly located in Illinois, Utah, Colorado and New Mexico. The sixth (green) captures 101 counties of Arkansas and Alabama, the seventh (red) 56 counties of Louisiana and finally the eighth (purple) 10 counties of New Hampshire.

We observe again that there is a clear geographical pattern in the retrieved clusters. This indicates that the $M$-matrix $\widehat{\mathbf{\Theta}}_{\mathrm{a}}$ estimated by SQUIC-fit accurately captures the latent graphical structure of the dataset, and that the resulting eigenvectors of $\widehat{\mathbf{L}}_{\mathrm{rw}}$ are well suited for spectral clustering tasks.

### 4.2.4 Image classification

In this case study we demonstrate the applicability of the introduced algorithms in the estimation of $M$-matrices emerging from image applications. We study the problem of classifying facial images and handwritten characters according to their labels by applying a spectral clustering routine on the eigenvectors of the estimated random walk Laplacian $\widehat{\mathbf{L}}_{\mathrm{rw}}$. We consider the following publicly available datasets

- YaleA [123]: A collection of $p = 165$ grayscale images of 15 individuals at resolution $n = 64 \times 64$ pixels. There are 11 images per subject, one per different facial expression.

- Olivetti [124]: A set of 10 different facial images of 40 distinct subjects, resulting in $p = 400$ instances at resolution $64 \times 64$ pixels, taken at different times, varying lighting, facial expressions and facial details.
- USPS [125]: A balanced set of $p = 11000$ images of 10 distinct handwritten digits with $n = 16 \times 16$ pixels.
- KMNIST [126]: The entire Kuzushiji-MNIST balanced dataset with $p = 70000$ images of 10 modern Japanese hiragana characters at resolution $n = 28 \times 28$.

The high dimensionality $p$ of these datasets renders them computationally unfavorable for the CGL [127] and SGL [59] methods. We therefore compare here our proposed algorithms against the traditional approach of building adjacency matrices $\mathbf{W}$. This approach consists of initially creating the connectivity matrix $\mathbf{G} \in \mathbb{R}^{p \times p}$ from a $k$-nearest neighbors routine, with the number of nearest neighbors (NN) set such that the resulting graph is connected. For these datasets the number of nearest neighbors needed for a connected graph is NN $= 12$ for YaleA and Olivetti, and NN $= 11$ for both USPS and KMNIST. Subsequently, the similarity matrix $\mathbf{H} \in \mathbb{R}^{p \times p}$ between the data points is defined similarly to [128] as $\mathbf{H}_{ij} = \max\{\mathbf{H}_i(j), \mathbf{H}_j(i)\}$ with $\mathbf{H}_i(j) = \exp\left(-4\frac{\|\mathbf{y}_i - \mathbf{y}_j\|^2}{\sigma_i^2}\right)$, with $\sigma_i$ standing for the Euclidean distance between the $i$-th data point and its $k$-th nearest neighbor. The adjacency matrix $\mathbf{W}$ is then created as

$$\mathbf{W} = \mathbf{G} \odot \mathbf{H}. \tag{4.2.8}$$

We utilize the kNN connectivity matrix $\mathbf{G}$ as graphical bias for SQUIC-fit and SQUIC-sqp and find the optimal scalar tuning parameter $\lambda = \lambda_{\text{opt}}$ for each case. The matrix parameter $\mathbf{\Lambda}$ in (3.4.7) is then set with $\eta = \lambda_{\text{opt}}/\sqrt{p}$ for both SQUIC-fit and SQUIC-sqp. Our strategy is thus penalizing the graphical bias $\mathbf{G}$ with a decreasing rate for an increasing number of dimensions $p$. The goal is to obtain within a reasonable amount of time graphical representations of the datasets that are sparser than $\mathbf{G}$ and more accurate. Due to these favorable properties, we anticipate that the retrieved $M$-matrices will lead to an increase in the classification accuracy metrics after applying a spectral clustering routine. Sparsity in the graph is measured in term of edge density, defined as $\epsilon = |E|/(|V| * (|V| - 1))$, which is a ratio reflecting how close the graph is to a complete graph, with $\epsilon = 1$ for a complete graph.

The $M$-matrices of the 4 datasets under consideration are retrieved in $t = 0.9, 6.7, 46.3$ and $1255.6$ seconds with SQUIC-fit, and in $t = 2.4, 3.6, 31.5$ and

| Method | YaleA density | YaleA ACC | YaleA NMI | Olivetti density | Olivetti ACC | Olivetti NMI | USPS density | USPS ACC | USPS NMI | KMNIST density | KMNIST ACC | KMNIST NMI |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| kNN | −16.81% | −5.13% | −7.26% | −9.39% | −5.57% | −5.29% | −0.28% | −12.63% | −7.73% | −0.24% | −15.72% | −11.54% |
| SQUIC-fit | **0.083** | **0.613** | **0.650** | **0.04** | **0.646** | **0.7852** | −0.02% | **0.652** | **0.683** | **0.0003** | **0.61** | **0.587** |
| SQUIC-sqp | −8.32% | −3.06% | −4.62% | −8.16% | −1.87% | −1.81% | **0.002** | −1.89% | −1.85% | −0.05% | −0.23% | −0.37% |

Table 4.2: *Classification results for the image datasets under consideration. We report the edge density $\epsilon$, and the accuracy metrics ACC, NMI. The best value achieved is presented in bold, and the percentages show how inferiorly the other methods fared against it.*

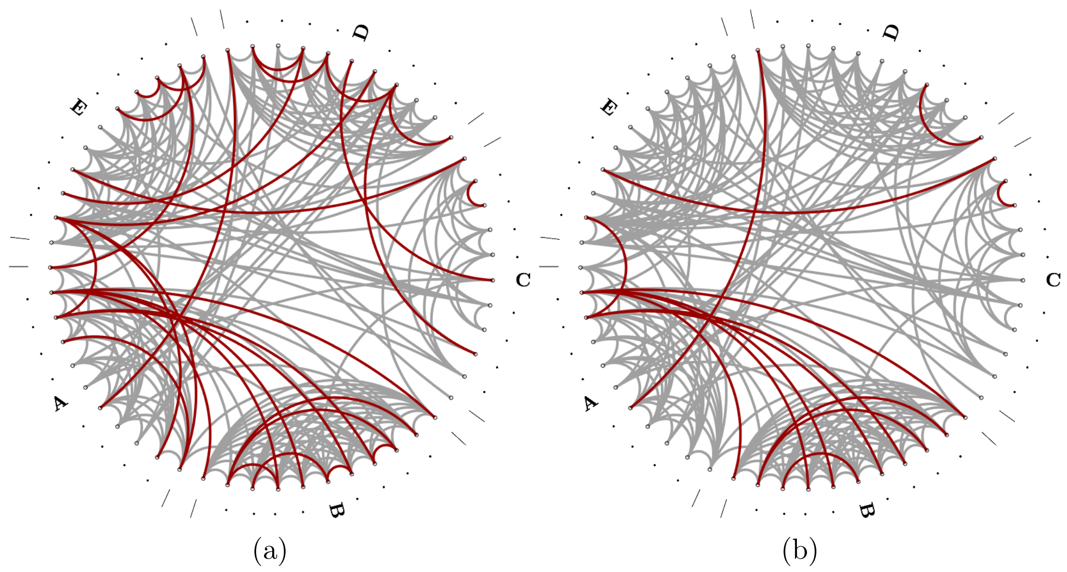


Figure 4.12: *Comparison of the graphical structure of the adjacency matrix* $\mathbf{A}$ *for a subset of the dataset YaleA. The coloring indicates the edges that were removed (in red) from the initial kNN graphical bias, and the edges (in gray) that remained after the application of the two proposed algorithms. (a) Graph estimated with SQUIC-fit with 398 remaining and 68 removed edges. (c) Graph estimated with SQUIC-sqp with 432 remaining and 28 removed edges.*

2024 seconds with SQUIC-sqp respectively. We summarize the rest of our results in Table 4.2. For each dataset we report the edge density, the ACC and the NMI achieved by the best method, and the percentage the remaining methods are inferior to that value. Inferiority in percentage values is defined as $I = 100 \cdot \gamma \cdot (e_{\text{ref}} - e_{\text{best}}) / e_{\text{best}}$, where $e_{\text{best}}$ is the best value, $e_{\text{ref}}$ the value it is compared against, and $\gamma = -1$ for minimization scenarios, as the edge density $\epsilon$, and $\gamma = 1$ for maximization ones, as the accuracy metrics ACC and NMI.

Both SQUIC-fit and SQUIC-sqp improve the classification accuracy of the traditional kNN graph for all the datasets considered. In particular, SQUIC-fit achieves the highest accuracy metrics for all cases, and the lowest edge density for all cases except USPS. The reduction of the edge density is more evident for YaleA and Olivetti, as the tuning parameter $\eta = \lambda_{\text{opt}}/\sqrt{p}$ applied on the entries of the graphical bias **G** has a lesser impact for graphs of low dimensions $p$. In Figure 4.12 we illustrate this reduction in $\epsilon$ for the YaleA dataset. For visual clarity we select a subset (variables 100 to 155) of the dataset, organized in five distinct classes, denoted by the letters A to E, with each class composed by eleven variables. We order the variables in a circular layout and compare the graphical structure obtained by SQUIC-fit (398 gray edges in Figure 4.12a) and SQUIC-sqp (432 gray edges in Figure 4.12b). The red edges in both figures represent the edges that were removed from the graphical bias **G**, estimated with a kNN routine, after applying SQUIC-fit (68 edges) and SQUIC-sqp (28 edges). Multiple edges that connect variables belonging to different classes are removed in both cases, thus reducing the interclass connectivity of the graph. The advantages of these sparser graphical structures, with edge weights assigned by solving the MLE problem, are verified by the increased classification scores of Table 4.2.

# Part II

# Nonlinear spectral clustering

# Chapter 5

# Spectral methods for graph clustering

The aim of graph clustering is to distinguish groups of points according to their similarities. If these data points are defined by a matrix describing pointwise similarities, the problem of grouping them in $k$ parts is treated as a graph partitioning problem with an undirected weighted graph $\mathcal{G}(V, E, \mathbf{W})$ being constructed. Its nodes $V$ represent the data points, and the similarity between the connected edges $E$ is encoded in the elements $\mathbf{W}_{ij} \geq 0$ of the weight matrix $\mathbf{W}$. Graph-theoretic approaches have proven to be highly successful in characterizing and extracting clusters. However, the resulting combinatorial optimization problems frequently appear to be NP-hard [129].

Spectral clustering is a popular graph-based method due to the simplicity of its implementation, the reasonable computation time, and the fact that it overcomes the NP-hardness of other graph-theoretic approaches by solving a relaxed optimization problem in polynomial time. Its idea is based on the eigendecomposition of matrices that describe the connectivity of a graph [19]. The spectral clustering of the total number of nodes $n = |V|$ into groups $C_1, \dots, C_k$ is equivalent to a partitioning problem, usually with a dual objective: high intracluster similarity and low intercluster similarity is desired, while at the same time the cardinality or the volume of the clusters should not differ excessively.[1] Depending on the application domain, the idea of spectral decomposition can be applied to either partitioning or clustering problems. The fundamental difference between them is the fact that in partitioning scenarios tightly balanced partitions are favored [130], and are possibly enforced using additional constraints, while in clustering the focus is on identifying the existence of communities [131] that are not necessarily of equal size.

[1] In what follows $n$ refers to the number of nodes in a graph, and not the available samples. We reserve $p$ to denote the value of the $p$-norm.

In this chapter, we review some fundamental notions related to measuring the quality of a graph partition in Section 5.1, and review the problem of splitting a graph into two roughly equal segments with spectral methods in Section 5.2. In Section 5.3 we consider a nonlinear reformulation of spectral bipartitioning that leads to superior graph cut results, and in Section 5.4 we outline how multiple clusters can be obtaining simultaneously from the eigenvectors of the graph Laplacian operator. Last, in Section 5.5 we present the landscape of existing solutions methods related to the clustering of eigenvectors of nonlinear variants of the graph Laplacian.

## 5.1 Graphs and graph cut metrics

The dual objective of graph clustering, discussed previously, is reflected in the balanced cut metrics presented below. When bisecting a graph $\mathcal{G}(V, E, \mathbf{W})$ into two subsets $C$ and its complement $\overline{C} = (V \backslash C)$ the cut between them is defined as

$$\text{cut}(C, \overline{C}) = \sum_{i \in C, j \in \overline{C}} \mathbf{W}_{ij}. \tag{5.1.1}$$

There are two main ways of measuring the size of one of these subsets $C \subset V$, and to thus balance the partitions accordingly. The first one is the cardinality, which considers the number of nodes in each subset, and the second one the volume, which measures the size of $C$ by summing the weights of all edges that are part of $C$. We define these metrics as

$$|C| = \text{number of nodes in } C, \text{ and, } \text{vol}(C) = \sum_{i \in C} \mathbf{D}_{ii} = \sum_{i \in C} \sum_{j=1}^{n} \mathbf{W}_{ij}. \tag{5.1.2}$$

As balanced graph cut criteria we consider the ratio [132] and normalized cut [61], which in the case of bisection read

$$\text{RCut}(C, \overline{C}) = \frac{\text{cut}(C, \overline{C})}{|C|} + \frac{\text{cut}(\overline{C}, C)}{|\overline{C}|}, \tag{5.1.3}$$

$$\text{NCut}(C, \overline{C}) = \frac{\text{cut}(C, \overline{C})}{\text{vol}(C)} + \frac{\text{cut}(\overline{C}, C)}{\text{vol}(\overline{C})}. \tag{5.1.4}$$

Alternatively, one can consider the ratio Cheeger cut $\mathrm{RCC}(C,\overline{C})$ [22], which controls the balance between the bipartitions in a marginally different way, as

$$\mathrm{RCC}(C,\overline{C}) = \frac{\mathrm{cut}(C,\overline{C})}{\min\{|C|,|\overline{C}|\}}. \tag{5.1.5}$$

The RCut and RCC partitioning metrics are related to each other

$$\mathrm{RCC}(C,\overline{C}) \le \mathrm{RCut}(C,\overline{C}) \le 2\mathrm{RCC}(C,\overline{C}), \tag{5.1.6}$$

thus, minimizing for the ratio cut results in reducing the value of the Cheeger cut as well. Extending (5.1.5) to its normalized counterpart follows naturally by considering the volume of the clusters in the place of cardinality for balancing.

When trying to identify $k$ clusters $C_1,\dots,C_k$ in the entire node set $V$, the RCut and NCut metrics are commonly adapted to the task, and their expressions are formalized as [133]

$$\mathrm{RCut}(C_1,\dots,C_k) = \sum_{i=1}^{k} \frac{\mathrm{cut}(C_i,\overline{C_i})}{|C_i|}, \tag{5.1.7}$$

$$\mathrm{NCut}(C_1,\dots,C_k) = \sum_{i=1}^{k} \frac{\mathrm{cut}(C_i,\overline{C_i})}{\mathrm{vol}(C_i)}. \tag{5.1.8}$$

It is important to note that the graph cut criteria discussed here correspond to nearly optimal clusters when their value approaches zero.

In spectral methods, the connectivity of $\mathcal{G}$ is usually described by means of the 2-norm graph Laplacian matrix $\mathbf{L} \in \mathbb{R}^{n\times n}$. The graph Laplacian matrix $\mathbf{L}$ is a symmetric, positive semi-definite and diagonally dominant matrix whose spectral properties reveal a number of important topological characteristics of the graph [60, 134]. We refer to our previous Section 2.4.1 for properties and normalized variants of the graph Laplacian.

## 5.2 Spectral bipartitioning

In the case of bipartitioning, i.e., $k=2$, we consider two complementary subsets $C,\overline{C}$ such that $C\cup\overline{C}=V, C\cap\overline{C}=\varnothing$. An indicator vector $\mathbf{u}=(u_1,\dots,u_n)^\intercal \in \mathbb{R}^n$

is defined for the vertex set $V = \{v_1, \dots, v_n\}$ with

$$u_i = \begin{cases} \sqrt{\frac{|\overline{C}|}{|C|}} & \text{if } v_i \in C, \\ -\sqrt{\frac{|C|}{|\overline{C}|}} & \text{if } v_i \in \overline{C}. \end{cases} \tag{5.2.1}$$

The ratio cut partitioning metric (5.1.3) can now be expressed in terms of the graph Laplacian $\mathbf{L}$ with

$$\operatorname{RCut}\left(C, \overline{C}\right) = \frac{\mathbf{u}^\intercal \mathbf{L} \mathbf{u}}{\mathbf{u}^\intercal \mathbf{u}}. \tag{5.2.2}$$

Furthermore, it can be seen from (5.2.1) that the indicator vector of node assignments $\mathbf{u}$ is orthogonal to the constant vector $\mathbf{e}$, i.e., $\mathbf{u}^\intercal \cdot \mathbf{e} = 0$. Therefore, the problem of minimizing the ratio cut (5.1.3) can be expressed as

$$\underset{C, \overline{C} \in V}{\text{minimize}} \; \frac{\mathbf{u}^\intercal \mathbf{L} \mathbf{u}}{\mathbf{u}^\intercal \mathbf{u}}. \tag{5.2.3}$$

This optimization problem is NP-hard, due to the discreteness of the indicator vector, thus a relaxation approach is followed by allowing $\mathbf{u}$ to attain values in all of $\mathbb{R}$, i.e., $u_i \in \mathbb{R}$. This relaxed optimization problem reads

$$\underset{\mathbf{u} \in \mathbb{R}^n}{\text{minimize}} \; \frac{\mathbf{u}^\intercal \mathbf{L} \mathbf{u}}{\mathbf{u}^\intercal \mathbf{u}} \tag{5.2.4a}$$

$$\text{subject to } \mathbf{u}^\intercal \cdot \mathbf{e} = 0. \tag{5.2.4b}$$

The objective function (5.2.4a) is the Rayleigh quotient of the graph Laplacian matrix $\mathbf{L}$. The minimum of the quotient is attained by the smallest eigenvalue $\lambda_1 = 0$ of $\mathbf{L}$, with the associated eigenvector $\mathbf{v}^{(1)} = c \cdot \mathbf{e}$ being the minimizer. However, this eigenpair corresponds to the trivial partition $V = V \cup \emptyset$. Additionally, for nonconnected graphs, the multiplicity of the zero eigenvalue corresponds to the number of connected components. Therefore, taking into account the constraint (5.2.4b) we seek the second-smallest eigenvalue, called the algebraic connectivity of the graph [15], and its associated eigenvector. For a connected graph $\mathcal{G}$, this corresponds to $\mathbf{v}^{(2)}$, also termed Fiedler's eigenvector. It enables the partitioning of $\mathcal{G}$ into the two complementary sets $C, \overline{C}$ by thresholding its entries around zero, or their median value for tightly balanced partitioning applications. A computationally more expensive alternative, used more widely in clustering applications, is to perform a sweep cut on the Fiedler eigenvector by sorting the entries of $\mathbf{v}^{(2)}$, considering each of the $n-1$ cuts possible, and selecting the one that minimizes the RCut (5.1.3). This process can be easily generalized to the normalized case $\mathbf{L}_{\text{rw}}$ [17], corresponding to a a minimization of the NCut (5.1.4).

Obtaining $k$ clusters from the spectral graph bisection method is possible by recursively bipartitioning the graph until the desired number of $k$ clusters is reached. At each recursive step, the partition whose bisection leads to smaller values of the global ratio cut (5.1.7) is split into two. Alternatively, in order to directly realize multiple strongly connected components of $\mathcal{G}$ the procedure outlined in Section 5.4 is followed.

## 5.3 Bipartitioning with the graph $p$-Laplacian

Reformulating spectral graph partitioning in the $p$-norm, for $p \in (1, 2]$, is based on the fact that better theoretical bounds on the balanced partitioning metrics, introduced in Section 5.1, are achieved at the limit $p \to 1$. As $p$ approaches one, the resulting bipartition indicator vector $\mathbf{u}$ attains more discrete values and leads to nearly optimal balanced graph cut metrics and tighter partitionings [23, 135, 136]. At the limit of $p = 1$, solving the total variation problem [137, 138] has also been proven to be a tighter relaxation for balanced discrete graph cut metrics than the 2-norm relaxation (5.2.3).

Let $\kappa_{\text{RCC}}$ be the optimal value of the Cheeger cut (5.1.5) for the bisection of a graph into two complements $C, \overline{C}$, i.e.,

$$\kappa_{\text{RCC}} = \inf_{C} \text{RCC}(C, \overline{C}), \tag{5.3.1}$$

and let the value obtained by thresholding the entries of the second eigenvector of the graph $p$-Laplacian be denoted by $\kappa^*_{\text{RCC}}$. The theoretical bounds for the approximation of an optimal cut with $p$-spectral bisection read [23]

$$\kappa_{\text{RCC}} \leq \kappa^*_{\text{RCC}} \leq p \left( \max_{i \in V} \mathbf{D}_{ii} \right)^{\frac{p-1}{p}} (\kappa_{\text{RCC}})^{\frac{1}{p}} . \tag{5.3.2}$$

The above inequality implies that as $p \to 1$ we have $\kappa^*_{\text{RCC}} \to \kappa_{\text{RCC}}$, thus the Cheeger cut obtained by the second $p$-eigenvector approximates its optimal value. Additionally, due to the relationship between the Cheeger and the ratio cut (5.1.6), the ratio cut also approaches its optimal value for $p \to 1$. This suggests that $p$-spectral bipartitioning is superior to its traditional 2-norm counterpart.

The graph Laplacian operator is redefined in the $p$-norm for a node $i \in V$ as

$$\left(\boldsymbol{\Delta}_p \mathbf{u}\right)_i = \sum_{j \in V} \mathbf{W}_{ij} \phi_p \left(u_i - u_j\right) \tag{5.3.3}$$

and its normalized counterpart as $\left(\boldsymbol{\Delta}_p^{(n)} \mathbf{u}\right)_i = \frac{1}{\mathbf{D}_{ii}} \sum_{j \in V} \mathbf{W}_{ij} \phi_p \left(u_i - u_j\right)$, with the scalar function $\phi_p : \mathbb{R} \to \mathbb{R}$ being $\phi_p(x) = |x|^{p-1} \text{sign}(x)$, for $x \in \mathbb{R}$. In what follows we focus on the standard graph $p$-Laplacian case, but all concepts can be easily generalized to the normalized one. The $p$-Laplacian operator is nonlinear, with $\boldsymbol{\Delta}_p(\gamma \mathbf{x}) \neq \gamma \boldsymbol{\Delta}_p(\mathbf{x})$ for $\gamma \in \mathbb{R}$ and $p \in (1, 2)$, and the linear counterpart $\mathbf{L}$ is recovered for $p = 2$, as $\phi_2(x) = x$ and $\mathbf{L}(\cdot) = \boldsymbol{\Delta}_p(\cdot)$. Therefore, the action of the standard graph Laplacian operator on a vector $\mathbf{u} \in \mathbb{R}^n$ can be generalized in the $p$-norm as

$$\langle \mathbf{u}, \boldsymbol{\Delta}_p \mathbf{u} \rangle = \frac{1}{2} \sum_{i,j=1}^{n} \mathbf{W}_{ij} \left|u_i - u_j\right|^p . \tag{5.3.4}$$

Similar to the approach followed in Section 5.2, we wish to obtain the second-smallest eigenvector of the symmetric graph $p$-Laplacian $\boldsymbol{\Delta}_p \in \mathbb{R}^{n \times n}$ in order to minimize the value of the RCut (5.1.3). The Rayleigh-Ritz principle, extended to the nonlinear case, states that a scalar value $\lambda_p \in \mathbb{R}$ is called an eigenvalue of $\boldsymbol{\Delta}_p$ if there exists a vector solution $\mathbf{v} \in \mathbb{R}^n$ such that $\left(\boldsymbol{\Delta}_p \mathbf{v}\right)_i = \lambda_p \phi_p(v_i)$, with $i = 1, \ldots, n$. In order to obtain the smallest eigenpair of the $p$-Laplacian operator, we reformulate the Rayleigh quotient minimization problem from the linear 2-norm case $F_2(\mathbf{u}) : \mathbb{R}^n \to \mathbb{R}$,

$$F_2(\mathbf{u}) = \frac{\langle \mathbf{u}, \mathbf{L}\mathbf{u} \rangle}{\|\mathbf{u}\|_2^2} = \frac{1}{2} \frac{\sum_{i,j=1}^{n} \mathbf{W}_{ij} \left(u_i - u_j\right)^2}{\|\mathbf{u}\|_2^2}, \tag{5.3.5}$$

to the nonlinear one $F_p(\mathbf{u}) : \mathbb{R}^n \to \mathbb{R}$ as

$$F_p(\mathbf{u}) = \frac{\langle \mathbf{u}, \boldsymbol{\Delta}_p \mathbf{u} \rangle}{\|\mathbf{u}\|_p^p} = \frac{1}{2} \frac{\sum_{i,j=1}^{n} \mathbf{W}_{ij} \left|u_i - u_j\right|^p}{\|\mathbf{u}\|_p^p}, \tag{5.3.6}$$

with the $p$-norm defined as $\|\mathbf{u}\|_p = \sqrt[p]{\sum_{i=1}^{n} |u_i|^p}$. A vector $\mathbf{v} \in \mathbb{R}^n$ is an eigenvector of $\boldsymbol{\Delta}_p$ if and only if it is a critical point of (5.3.6) [139]. The associated $p$-eigenvalue is given by $F_p(\mathbf{v}) = \lambda_p$. The functional $F_p$ is nonconvex, and it is easy to notice that for some scalar $\gamma \in \mathbb{R}$ it is invariant under scaling, and thus $F_p(\gamma \mathbf{u}) = F_p(\mathbf{u})$.

Additionally, fundamental properties of the graph Laplacian in the linear case

$p = 2$, which relate the eigenspectrum of $\mathbf{L}$ to the algebraic connectivity of the graph [15], can be extended to the nonlinear one for $p \in (1,2]$. The multiplicity of the smallest $p$-eigenvalue $\lambda_p^{(1)} = 0$ corresponds to the number of connected components in the graph [23], and the associated eigenvector is constant. Therefore, for a connected graph, we are searching for the second eigenvalue $\lambda_p^{(2)}$ of $F_p$ and the associated eigenvector $\mathbf{v}^{(2)}$ in order to obtain a bipartition. Furthermore, any two eigenvectors $\mathbf{v}^{(\alpha)}, \mathbf{v}^{(\beta)}$, with $\alpha \neq \beta$, of the $p$-Laplacian operator associated with nonzero eigenvalues are approximately $p$-orthogonal [140], i.e., $\sum_i \phi_p(v_i^{(\alpha)})\phi_p(v_i^{(\beta)}) \approx 0$.

## 5.4 Direct multiway spectral clustering

Exploiting information from $k$ eigenvectors of the graph Laplacian matrix $\mathbf{L}$ allows the direct $k$-way partitioning of a graph into $C_1, \ldots, C_k$ clusters, thus circumventing the need for a recursive strategy. The potential benefits of this approach are discussed in Section 6.1.

A relaxation approach is followed again for the minimization of RCut (5.1.7). We define $k$ indicator vectors $\mathbf{u}_j = \left(u_{1,j}, \ldots, u_{n,j}\right)^\intercal$ such that for $i = \{1, \ldots, n\}$, $j = \{1, \ldots, k\}$,

$$u_{i,j} = \begin{cases} \frac{1}{\sqrt{|C_j|}} & \text{if } v_i \in C_j, \\ 0 & \text{otherwise.} \end{cases} \tag{5.4.1}$$

The matrix $\mathbf{U} \in \mathbb{R}^{n \times k}$ contains these $k$ orthonormal vectors in its columns, thus $\mathbf{U}^\intercal\mathbf{U} = \mathbf{I}$. The expression for estimating the global ratio cut (5.1.7) is now $\mathrm{RCut}(C_1, \ldots, C_k) = \mathrm{tr}(\mathbf{U}^\intercal\mathbf{L}\mathbf{U})$ with tr being the trace of a matrix. The discrete optimization problem for the minimization of (5.1.7) reads

$$\underset{C_1,\ldots,C_k}{\text{minimize}} \ \mathrm{tr}(\mathbf{U}^\intercal\mathbf{L}\mathbf{U}). \tag{5.4.2a}$$

Finding globally optimum solutions for this expression is again a known NP-hard problem [129]. The optimization problem is therefore relaxed by allowing the entries of matrix $\mathbf{U}$ to attain any value in $\mathbb{R}$, i.e., $\mathbf{u}_j \in \mathbb{R}^n$. The relaxed optimization problem now reads

$$\underset{\mathbf{U}\in\mathbb{R}^{n\times k}}{\text{minimize}} \ F_2(\mathbf{U}) = \mathrm{tr}(\mathbf{U}^\intercal\mathbf{L}\mathbf{U}), \tag{5.4.3a}$$

$$\text{subject to } \mathbf{U}^\intercal\mathbf{U} = \mathbf{I}. \tag{5.4.3b}$$

Fan's trace min/max principle [139] dictates that the solution to this minimiza-

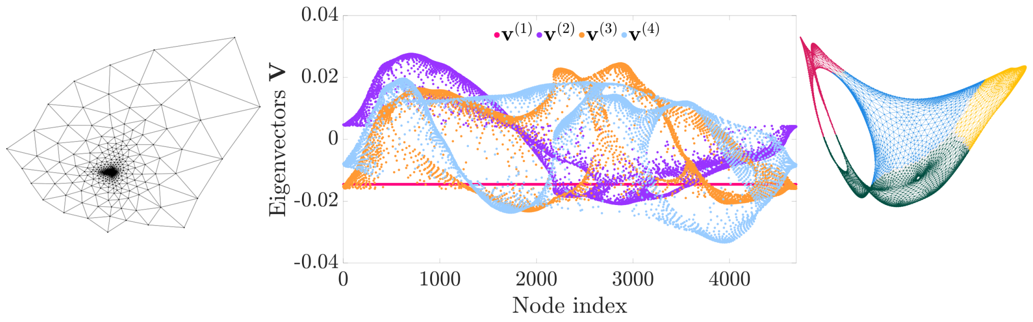


Figure 5.1: *Spectral clustering of the "3elt" graph with* 4720 *nodes and* 27444 *edges. (Left) The graph visualized with its spatial coordinates* $(x, y)$*. (Center) The eigenvectors associated with the* 4 *smallest eigenvalues of the resulting graph Laplacian. (Right) The discrete partitions obtained by the eigenvectors. We utilize* $\mathbf{v}^{(2)}$ *and* $\mathbf{v}^{(3)}$ *as the spectral coordinates.*

tion problem is given by a matrix $\mathbf{U}$ whose first $k$ columns are spanned by the eigenvectors associated with the $k$ smallest eigenvalues of $\mathbf{L}$. In order to obtain discrete clusters from the resulting real valued eigenvectors we consider for the $n$ nodes of the graph $n$ vectors $\mathbf{h}_i = \mathbf{U}_i^\intercal \in \mathbb{R}^k$, $\forall\, i \in [1, n]$. These are considered the spectral coordinates of the graph and have to be divided into $k$-groups $C_1, \ldots, C_k$. Obtaining discrete partitions from the eigenvectors of the graph Laplacian is commonly achieved with an external algorithm based on the relative distance between the eigenvectors. We provide more details on this procedure in Section 6.2.2. These spectral coordinates can be also be utilized for graph drawing purposes, and enjoy the advantages of leading to highly readable layouts and a fast computation time. More specifically, spectral layouts place a vertex at the centroid of its neighbors, with some deviation defined by the degree of each node [141]. We visualize in Figure 5.1 the entries of multiple eigenvectors of the graph Laplacian, the clusters which they induce, and the alternative graph drawing that they enable for the finite element mesh "3elt" from the AG-Monien Graph Collection [142].

Similar to the previous bisection approaches, this procedure can be generalized to the normalized functional $F_2^{(n)}$ that minimizes the NCut (5.1.8). In this case the entries of the indicator vectors are defined as

$$u_{i,j} = \begin{cases} \frac{1}{\sqrt{\mathrm{vol}(C_j)}} & \text{if } v_i \in C_j, \\ 0 & \text{otherwise.} \end{cases} \tag{5.4.4}$$

With the relaxed constraint now reading $\mathbf{U}^\intercal \mathbf{D} \mathbf{U} = \mathbf{I}$, the $k$ columns of the matrix solution $\mathbf{U}$ correspond to the eigenvectors spanned by the $k$ smallest eigenvalues of $\mathbf{L}_{\text{rw}}$. This is the normalized spectral clustering approach introduced in [61].

As a final remark here, we note that the functional $F_2$ is invariant under a change of basis, i.e., $F_2(\mathbf{UQ}) = F_2(\mathbf{U})$, for all $\mathbf{Q}$ belonging to the group of $k \times k$ orthogonal matrices, $\mathcal{O}(k) = \{\mathbf{Q} \in \mathbb{R}^{k\times k} \mid \mathbf{Q}^\intercal\mathbf{Q} = \mathbf{I}\}$. This property will enable the reformulation of our $p$-spectral clustering problem into an unconstrained manifold minimization problem in the following Chapter 6.

## 5.5 Related work on $p$-spectral methods

Spectral clustering algorithms and extensions of the method have enjoyed tremendous popularity since their introduction [18, 61]. Offering a comprehensive list of all the algorithmic development since then goes beyond the scope of this thesis. We refer to [17] and [143] for an introduction to the topic and the theoretical background, and to [21] for a more recent list of available algorithms.

We offer, instead, in what follows an overview of the methods related to the nonlinear variants of spectral clustering The favorable theoretical properties of the $p$-Laplacian have also attracted significant recent algorithmic development. Following the seminal work of [23], using the same objective function, a self-tuning $p$-spectral algorithm is proposed that determines the optimal value of $p$ [144]. The authors in [140] generalize this approach to multiway partitioning by employing a modified gradient descent update that converges to multiple $p$-eigenvectors. The nodal properties of multiple eigenvectors of the graph $p$-Laplacian were investigated in [62]. In [27] we introduced an explicit way to handle the constraints between the first two eigenvectors of the $p$-Laplacian, and a hybrid scheme to recursively partition large-scale graphs. In [145] the authors express the $p$-Laplacian eigenproblem into a nonlinear eigenproblem with eigenvector nonlinearity (NEPv), commonly encountered in quantum chemistry applications. Tight relaxations based on the concept of total variation, leading to similar sharp indicator eigenfunctions, have also been proposed for bipartitioning [138] and multiway problems [24, 146]. In [147] the concept of total variation was utilized in multiclass transductive learning problems, and in [148] to find the leading community of a graph. Reformulations of the spectral method in different $p$-norms have also been employed in local graph clustering methods [149, 150], in hypergraph partitioning [151, 152], as well as in the clustering of signed graphs [153].

# Chapter 6

# Direct multiway $\boldsymbol{p}$-spectral clustering

We devote this chapter to the presentation of our multiway $p$-spectral clustering algorithm on Grassmann manifolds. We motivate the development of our method with some practical evidence in 6.1, and then we introduce the formulation of the unconstrained minimization problem that leads to the clustering of graphs in Section 6.2.

## 6.1 Motivation for multiple $\boldsymbol{p}$-eigenvectors

Besides the theoretical advantages of performing spectral bipartitioning in the $p$-norm, discussed in Section 5.3, we further show some practical considerations that motivate our research on $p$-spectral clustering. The results that follow are obtained by our algorithm, that will be introduced in the next section. We begin by calculating the second eigenvector of the graph Laplacian $\mathbf{L}$ and of the graph $p$-Laplacian $\mathbf{\Delta}_p$ for the 2-dimensional (2D) finite element mesh “grid1_dual" from the AG-Monien Graph Collection [142], with 224 nodes and 420 edges. Subsequently, we attempt to extract two clusters ($k = 2$) from the entries of the second eigenvector by thresholding it around zero. The results are illustrated in Figure 6.1. We plot the mesh (graph) on the horizontal axis ($x, y$ coordinates) and the eigenvector entries on the vertical one ($z$ coordinate). Each eigenvector entry is visualized using the $x$ and $y$ coordinates of the associated node of the mesh, in order to demonstrate the clusters. In the standard spectral computations ($p = 2$) the entries of the Fiedler eigenvector $\mathbf{v}^{(2)}$ are distributed uniformly around zero. The number of cut edges is 20 and the value of the RCut $= 0.179$. In contrast, the entries of the second $p$-eigenvector $\mathbf{v}_p^{(2)}$ for $p = 1.1$ are organized into two easily distinguishable partitions, while at the same time the size of the edge cut is reduced to 16, and the value of RCut to 0.143. The reason

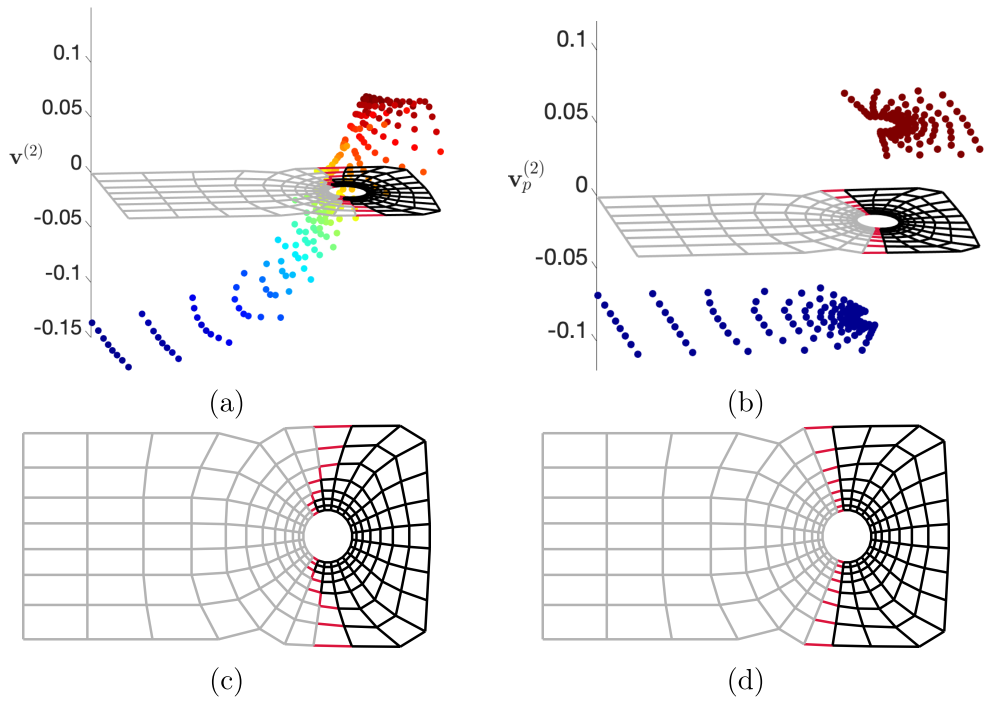
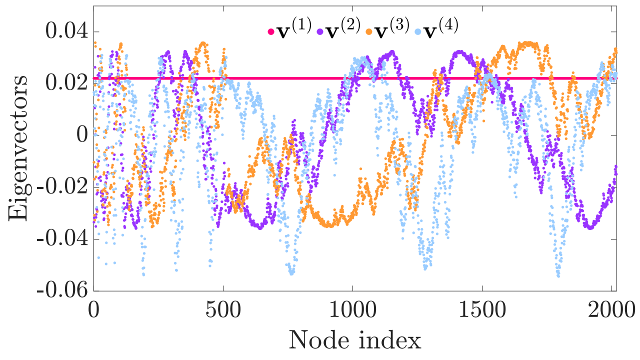
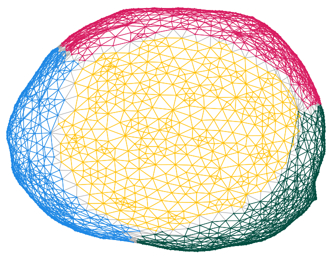


Figure 6.1: *Finding two clusters based on the entries of the second eigenvector of the graph Laplacian* $\mathbf{L}$ *and of the graph p-Laplacian* $\boldsymbol{\Delta}_p$ *for a finite element mesh (see text for details). The two partitions are depicted in black and gray, while the cut edges are depicted in red. The z-axis represents the value of the entries of the eigenvector, with their coloring indicating their distance from zero. (a) Standard spectral computation* ($p = 2$)*. (b) Spectral computation in the p-norm for* $p = 1.1$*. (c) The standard spectral clusters. (d) The p-spectral clusters for* $p = 1.1$*.*

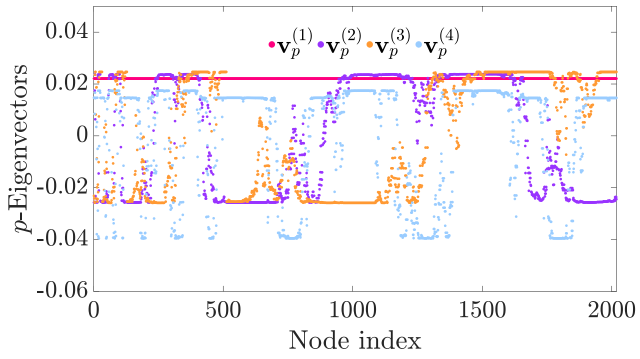
for this improved performance in the $p$-norm is the fact that as $p \to 1$, the cut obtained by thresholding $\mathbf{v}_p^{(2)}$ approaches its optimal value [20, 23]. When considering multiple eigenvectors ($k > 2$), this tendency towards optimal cut values as $p$ approaches one has been proven for graphs for which the number of strong nodal domains of the eigenvector corresponding to the $k$-th smallest eigenvalue $\lambda_k$ is equal to $k$ [62], e.g., the unweighted path graph. However, the application of $p$-Laplacian direct multiway clustering in more general graphs has shown promising results [140]. We demonstrate in Figure 6.2 the 2-norm and $p$-norm eigenvectors associated with the 4 smallest eigenvalues of the random Delaunay triangulation “Delaunay11" from the 10th DIMACS Implementation Challenge [154]. The more discrete organization of the entries of the eigenvectors in the $p = 1.1$ case is evident. We remind here that the last step of direct multiway spectral clustering algorithms involves the application of a distance-based method (e.g., k-means) on the eigenvector entries. Clearly, the distribution of



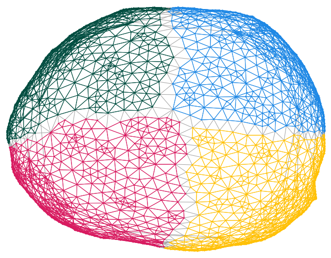
(a)



(b)



(c)



(d)

Figure 6.2: *Finding $k = 4$ clusters based on the entries of the eigenvectors of the graph Laplacian $\mathbf{L}$ and of the graph $p$-Laplacian $\boldsymbol{\Delta}_p$ for a random Delaunay triangulation (see text for details). The cut edges are depicted in gray and the graphs are visualized with $\mathbf{v}^{(2)}$ and $\mathbf{v}^{(3)}$ as the spectral coordinates. (a) The spectral eigenvectors for $p = 2$. (b) The clusters for $p = 2$ with* RCut $= 0.901$. *(c) The $p$-spectral eigenvectors for $p = 1.1$. (d) The clusters for $p = 1.1$ with* RCut $= 0.707$.

points in Figure 6.2a is more favorable than the one in Figure 6.2c for a distance based algorithm, thus leading to easily separable clusters that will approximate the optimal RCut (5.1.7) or NCut (5.1.8) values. This hypothesis is validated by the resulting clusters, visualized in Figures 6.2b and 6.2d using a spectral graph layout. The 2-norm clusters have an associated RCut value of 0.901, while the $p$-norm clusters reduce the RCut metric to 0.707.

In order to motivate the computation of multiple $p$-eigenvectors we additionally consider the fact that recursive bisection is highly dependant on the decisions made during the early stages of the process. Recursive methods suffer from the lack of global information, as they do not optimize over the entire node set in order to find $k$ optimal partitions, but instead focus on finding optimal bisections at each recursive step. Thus, they may result in suboptimal partitions [155]. This necessitates the further advancement of methods for direct multiway $p$-spectral clustering.

## 6.2 A Grassmannian approach to $p$-spectral clustering

Taking into account the objective function for spectral bipartitioning in the $p$-norm (5.3.6), the relaxed optimization problem of estimating multiple eigenvectors of the graph Laplacian (5.4.3) can be reformulated in the $p$-norm as

$$\underset{\mathbf{U}\in\mathbb{R}^{n\times k}}{\text{minimize}}\; F_p(\mathbf{U}) = \sum_{l=1}^{k}\sum_{i,j=1}^{n} \frac{\mathbf{W}_{ij}|u_i^l - u_j^l|^p}{2\|\mathbf{u}^l\|_p^p} \tag{6.2.1a}$$

$$\text{subject to} \sum_{i=1}^{n} \phi_p(u_i^l)\phi_p(u_i^m) = 0,\;\; \forall\, l \neq m,\; p \in (1,2],\; l \in [1,k],\; m \in [1,k]. \tag{6.2.1b}$$

The cluster indices are denoted by $l, m = 1, 2, \dots, k$. The final number of clusters $k$ can be considered predetermined, or estimated based on the relative eigengap as in (4.2.7). The matrix $\mathbf{U} = (\mathbf{u}_1, \dots, \mathbf{u}_k)$ contains the eigenvectors associated with the smallest $k$ eigenvalues of the $p$-Laplacian operator $\mathbf{\Delta}_p$ in its columns. In the case of normalized $p$-spectral clustering the normalized functional reads $F_p^{(n)}(\mathbf{U}) = \sum_{l=1}^{k}\sum_{i,j=1}^{n}\left(\mathbf{W}_{ij}|u_i^l - u_j^l|^p\right)/\left(2\mathbf{D}_{ii}\|\mathbf{u}^l\|_p^p\right)$. This scaling by the degree $\mathbf{D}_{ii}$ of the corresponding row $i$ results in the matrix $\mathbf{U}$ containing the eigenvectors of the normalized $p$-Laplacian operator $\mathbf{\Delta}_p^{(n)}$ (see subsection 5.3) in its columns. For brevity we restrict our analysis in this section in the case of unnormalized $p$-spectral clustering.

The constraint for $p$-orthogonal eigenvectors (6.2.1b) renders the optimization problem intractable. Therefore, we replace it with the traditional constraint $\mathbf{U}^\intercal\mathbf{U} = \mathbf{I}$ (5.4.3b), a tight approximation as shown in [140]. This constraint corresponds to the Stiefel manifold, which is composed of all orthogonal column matrices

$$\mathcal{S}t(k,n) = \{\mathbf{U} \in \mathbb{R}^{n\times k} \mid \mathbf{U}^\intercal\mathbf{U} = \mathbf{I}\}. \tag{6.2.2}$$

That is, a point in the Stiefel manifold is a specific orthogonal matrix [25].

Similar to standard direct $k$-way spectral clustering, we are interested in converging to some orthonormal basis of the eigenspace and not on the exact eigenvectors [17]. The final transformation of the $p$-spectral coordinates into clusters can be performed by either a flat algorithm like k-means or by rotating the normalized eigenvectors as shown in [156]. Both algorithms are based on the relative distances between points and not on the exact values of their coordinates. However, every set of orthonormal eigenvectors forming the matrix $\mathbf{U}$ is considered to be unique on $\mathcal{S}t$, even if they correspond to the same basis. Therefore, optimizing our objective (6.2.1a) over the Stiefel manifold leads to the well known identifiability issue [157], with the redundantly big search space of the Stiefel manifold causing slow convergence and the increased probability of getting stuck in local minima for a nonconvex function. This behavior is illustrated in Figure 6.3 for the problem of finding the first constant eigenvector of the graph $p$-Laplacian $\mathbf{\Delta}_p$ for a graph representing the Ecoli dataset from the UCI collection [158]. Optimizing over the Stiefel manifold (in cyan) leads to a failure to converge to the known constant solution $\mathbf{v}_p^{(1)} = c\mathbf{1}$ [23]. Thus, in this case, additional constraints have to be imposed, i.e., the Stiefel gradient corresponding to the first eigenvector has to be set to zero in order for it to attain constant values. This gradient correction approach guides the algorithm towards the correct solution, but is not applicable to the rest of the $k-1$ eigenvectors of $\mathbf{\Delta}_p$ as there is no theoretical guarantee for the values they should attain.

We thus consider the group of all $k \times k$ orthogonal matrices $\mathcal{O} = \{Q \in \mathbb{R}^{k\times k} \,|\, \mathbf{Q}^\intercal\mathbf{Q} = \mathbf{I}\}$. Searching for $k$ nonspecific and mutually orthogonal vectors as the solution to (6.2.1a) means that two solutions $\mathbf{U}_1$ and $\mathbf{U}_2$ belonging to the Stiefel manifold are considered equivalent if there exists some $\mathbf{Q} \in \mathcal{O}(k)$ such that $\mathbf{U}_1 = \mathbf{U}_2\mathbf{Q}$. This corresponds to the Grassmann manifold, a quotient space of $\mathcal{S}t(k,n)$ [159], defined as

$$\mathcal{G}r(k,n) \simeq \mathcal{S}t(k,n)/\mathcal{O}(k) = \{\text{span}(\mathbf{U}) : \mathbf{U} \in \mathbb{R}^{n\times k}, \mathbf{U}^\intercal\mathbf{U} = \mathbf{I}\}. \tag{6.2.3}$$

Points on $\mathcal{G}r(k,n)$ are understood as linear subspaces represented by an arbitrary basis stored as an $n$-by-$k$ orthonormal matrix [25]. The choice of the matrix $\mathbf{U}$ for these points is not unique, unlike for the ones on $\mathcal{S}t(k,n)$, with points on $\mathcal{G}r$ being defined through the relationship

$$\mathbf{U}^{\mathcal{G}r} = \{\mathbf{UQ} \,|\, \forall\ \mathbf{Q} \in \mathcal{O}(k)\},\ \ \mathbf{U} \in \mathbb{R}^{n\times k},\ n \gg k. \tag{6.2.4}$$

Optimizing our objective over the Grassmann manifold results in a reduced search space, with the solutions being an approximation of the orthonormal eigenvectors of $\mathbf{\Delta}_p$, that are satisfying fundamental properties of spectral graph theory without imposing additional constraints. This behavior can be observed in Figure 6.3, where optimizing over the Grassmann manifold leads to the constant first eigenvector of $\mathbf{\Delta}_p$ (in red) for the Ecoli dataset.

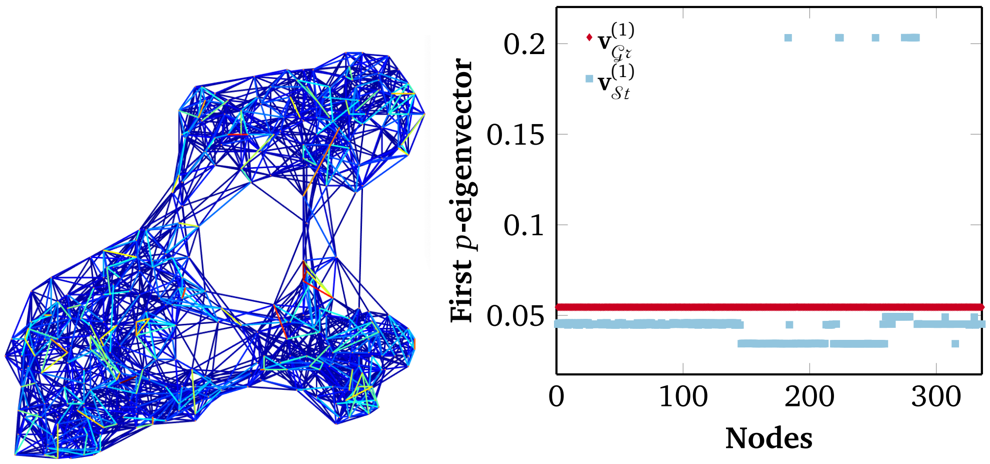


Figure 6.3: *Values of the entries of the first eigenvector* $\mathbf{v}^{(1)}$ *of* $\mathbf{\Delta}_p$ *for the Ecoli graph (illustrated), after minimizing the functional* (6.2.1a) *over the Stiefel* $\mathcal{S}t$ *and the Grassmann* $\mathcal{G}r$ *manifold. The graph in question is connected and thus* $\mathbf{v}^{(1)}$ *should be constant. This behavior is observed only on* $\mathcal{G}r$ *(in red), as* $\mathbf{v}^{(1)}$ *does not converge to a constant vector on* $\mathcal{S}t$ *(in blue).*

Thus, we approximate function (6.2.1a) as being invariant to any choice of basis and only depending on the subspace spanned by the $p$-eigenvectors, i.e., the columns of $\mathbf{U}$. The optimization problem of (6.2.1) can now be reformulated as an unconstrained problem on the Grassmann manifold as follows:

$$\underset{\mathbf{U}\in\mathcal{G}r(k,n)}{\text{minimize}}\ F_p(\mathbf{U}) = \sum_{l}^{k}\sum_{i,j=1}^{n} \frac{\mathbf{W}_{ij}|u_i^l - u_j^l|^p}{2\|\mathbf{u}^l\|_p^p}, \quad p \in (1,2]. \tag{6.2.5}$$

## 6.2.1 Optimization techniques

The previous section revealed that the direct multiway $p$-spectral clustering problem can be approximated as an optimization problem on a Grassmann manifold. Manifold optimization has been extensively developed over the last couple of decades, with the intention of providing robust numerical algorithms for problems on subspaces with a Riemannian structure. The work of [25] and [160] set the foundation to analyze such problems, with a focus on establishing a theory that leads to efficient numerical algorithms on the Stiefel $\mathcal{S}t(k,n)$ and Grassmann $\mathcal{G}r(k,n)$ manifolds. Specifically, they determine the Riemannian gradient and Hessian as the most critical ingredients in order to design first- and second-order algorithms on these subspaces. In particular, the Riemannian gradient and Hessian are projections of their Euclidean counterparts onto the tangent space of the manifold and the mapping between them is well established. Thus, in our case, the primary inputs to the manifold optimisation routines are the functional $F_p$ (6.2.5) along with its Euclidean gradient and optionally Hessian when using second-order algorithms.

The entries of the Euclidean gradient ($\mathbf{g}^k$) of $F_p$ with respect to $u_m^k$ read[1]

$$g_m^k = \frac{\partial F_p}{\partial u_m^k} = \frac{p}{\|\mathbf{u}^k\|_p^p}\left[\sum_{j=1}^{n} \mathbf{W}_{mj}\phi_p\left(u_m^k - u_j^k\right) - \phi_p\left(u_m^k\right)\sum_{i,j=1}^{n}\frac{\mathbf{W}_{ij}|u_i^k - u_j^k|^p}{2\|\mathbf{u}^k\|_p^p}\right]. \tag{6.2.6}$$

The Hessian of the functional is not sparse and can cause storage and scaling problems for big problem sizes. Hence, we use a sparse approximation of the Hessian by discarding the low rank terms as shown in [23]. The Euclidean Hessian follows the sparsity pattern of $\mathbf{W}$ and is approximated as

$$h_{ml}^k = \frac{\partial g_m^k}{\partial u_l^k} \approx \begin{cases} \dfrac{p(p-1)}{\|\mathbf{u}^k\|_p^p}\displaystyle\sum_{j=1}^{n}\mathbf{W}_{mj}|u_m^k - u_j^k|^{p-2} & \text{if } m = l, \\ \dfrac{-p(p-1)}{\|\mathbf{u}^k\|_p^p}\mathbf{W}_{ml}|u_m^k - u_l^k|^{p-2} & \text{otherwise.} \end{cases} \tag{6.2.7}$$

Our objective function $F_p(\mathbf{U})$ (6.2.5) is nonconvex for $p \in (1,2)$, and thus convergence to a global minimum cannot be guaranteed. Minimizing $F_p$ directly for a small value of $p$ results, in most cases, in convergence to a nonoptimal local minimum. Therefore, we take advantage of the fact that our minimization problem (6.2.5) exhibits a convex behavior for $p = 2$, and thus the global minimizer can be computed. The fact that $F_p$ is also continuous in $p$ suggests that for close values of $p_1, p_2$, the solution of $F_{p_1}(\mathbf{U}), F_{p_2}(\mathbf{U})$ will be close as well [23]. Accordingly, to find a solution at a given $p \in (1,2)$ we solve (6.2.5) by gradually reducing the value of $p$ (starting from $p = 2$), with the solution at the current $p$ serving as the initial iterate for the next $p$-level. In previous works [23, 140] the value of $p$ was decreased linearly. We instead decrease $p$ in a pseudocontinuous fashion, inspired by second order interior point methods and the way they handle the barrier parameter in order to achieve a superlinear rate of convergence [161]. The update rule for the value of $p$ reads

$$p = 1 + \max\left(\text{tol}, \min\left(\kappa \cdot (p-1), (p-1)^\theta\right)\right), \tag{6.2.8}$$

with $\kappa \in (0,1), \theta \in (1,2)$, and tol $= 10^{-1}$. The lower bound of this update

[1]The detailed derivation of the gradient is offered in Appendix A.

rule is $p \geq 1 + \text{tol}$, thus avoiding numerical instabilities with the discontinuity at $p = 1$. The value of $p$ is decreased at a superlinear rate, with the majority of the evaluations taking place close to $p = 1$, where the highest quality clusters are expected to be obtained.

At each level of $p$, we minimize our objective with a Grassmannian Newton's method, as it has proven to have a superlinear convergence rate close to the local optima and quadratic elsewhere [160]. The linear substeps within the Newton method are handled by a Grassmannian truncated conjugate gradient scheme [162]. For sparse or banded adjacency matrices $\mathbf{W}$ with bandwidth $2q + 1$ the computational cost per Newton iteration on the Grassmann is $O(nq^2k) + O(nk^2)$. Such matrices are commonly encountered in practical real-world applications. If the bandwidth is significantly more narrow, i.e. $q^2 \approx k$, or if $\mathbf{W}$ is tridiagonal then the cost becomes $O(nk^2)$ [163]. This reduction in the cost per iteration suggests that for very large and sparse adjacency matrices one can exploit the benefits from reordering methods that reduce the bandwidth size [164].

There exists various software packages for Riemannian optimization. One of the most popular choices, offering a wide variety of Riemannian optimization algorithms and a user-friendly environment in MATLAB, is Manopt [165]. A more recent library for manifold optimization is ROPTLIB [166]. Most of the kernels in ROPTLIB are written in C++ and use the highly optimized BLAS and LAPACK libraries for efficient linear algebra operations. We use ROPTLIB [166] to perform the Grassmannian Newton's steps, due to its superiority in terms of computational runtimes. The Newton's minimization procedure is terminated if the norm of the gradient ($\|\mathbf{g}_m^k\|$) at iteration $m$ is close to zero, i.e., $\|\mathbf{g}_m^k\|/\|\mathbf{g}_0^k\| < 10^{-6}$. In addition to the stopping criteria for Newton's method within each $p$ level we use a global stopping criterion based on cut values (RCut or NCut). If the cut value increases by at least 5% compared to its value at the previous $p$ level, we terminate the algorithm and choose the cluster corresponding to the smallest cut value, thus ensuring the semi-monotonic descent of our discrete objective.

### 6.2.2 Discretizing the $p$-eigenvectors

Similar to multiway spectral clustering in 2-norm (see Section 5.4), the final clustering solution is obtained by discretizing the multiple $p$-eigenvectors, stored in the matrix $\mathbf{U}$, obtained after solving the Grassmannian optimization problem (6.2.5). Various approaches have been proposed for this discretization in the 2-norm [21].

We consider two different methods for the discretization of the $p$-eigenvectors.

The first one is k-means, which is the most commonly used algorithm for the clustering of eigenvectors. However, the results of k-means depend heavily on the initial guess, and, therefore, the algorithm is usually run multiple times with different initial guesses and the best result is picked. We follow the approach of [122] with multiple orthogonal and random initial guesses that generally lead to a stable result. The second algorithm that we employ for the clustering of the $p$-eigenvectors is applicable only when minimizing the NCut graph cut metric. It is based on the fact that the application of a rotation matrix $\mathbf{P}$ transforms the matrix $\mathbf{U}$, containing the normalized $p$-eigenvectors in its columns, into a cluster indicator matrix containing only one nonzero entry per row that indicates the cluster index [21]. We consider the set of indicator matrices $J = \{\mathbf{J} \in \{0,1\}^{n\times k} : \mathbf{J}\cdot\mathbf{e}_k = \mathbf{e}_n\}$ and search for the matrices $\mathbf{J}$ and $\mathbf{P}$ that minimize the functional

$$\min_{\substack{\mathbf{P}^\intercal\mathbf{P}=\mathbf{I}\\ \mathbf{J}\in J}} f(\mathbf{P},\mathbf{J}) = \min_{\substack{\mathbf{P}^\intercal\mathbf{P}=\mathbf{I}\\ \mathbf{J}\in J}} \|\mathbf{UP}-\mathbf{J}\|_F. \tag{6.2.9}$$

We follow the approach of [156] for the solution of this optimization problem, that iteratively computes this discretization using singular value decomposition and non-maximum supression. In what follows, the clustering solutions obtained by employing k-means on the $p$-eigenvectors are denoted as pGrass-kmeans, and the ones obtained by the rotation of the normalized eigenvectors after solving (6.2.9) are denoted as pGrass-disc.

### 6.2.3 Multiway $p$-Grassmann clustering algorithm

A general summary of the algorithmic scheme employed for the unnormalized (RCut based) or normalized (NCut based) multiway $p$-Grassmann spectral clustering is offered in Algorithm 6. The MATLAB source code is available on GitHub at

https://github.com/DmsPas/Multiway-p-spectral-clustering.

The inputs of the algorithm are the adjacency matrix $\mathbf{W}$ of the graph in question, the number of the desired clusters $k$, the parameters $\kappa$, $\theta$ from (6.2.8), the final value of $p$ denoted as $p_\omega$, and whether the clustering will be based on the unnormalized (RCut) or the normalized (NCut) objective. As output we obtain the indices of the vertices forming the clusters and the discrete cut value of the clustering. In steps $2-8$ we solve the optimization problem for $p = 2$ and obtain $k$-eigenvectors stored in matrix $\mathbf{U}$. Their discretization, through the k-means algorithm or the solution of (6.2.9), is performed in step 9, and the cut

**Algorithm 6** $p$-Grassmann spectral clustering

```
Input: adjacency matrix $\mathbf{W}$, number of clusters $k$, $\kappa$, $\theta$, final $p$ value $p_\omega$, normalized
Output: cluster indices $\mathbf{c}_{\text{best}}$, cut value $\text{r}_{\text{best}}$
 1: function pGrassmannClustering
 2: if normalized then
 3:     Find U: $\underset{\mathbf{U}\in\mathbb{R}^{(k,n)}}{\text{minimize}}\, F_2^{(n)}(\mathbf{U})$ using W      ➜ See section 5.4
 4:     Cut = NCut
 5: else
 6:     Find U: $\underset{\mathbf{U}\in\mathbb{R}^{(k,n)}}{\text{minimize}}\, F_2(\mathbf{U})$ using W
 7:     Cut = RCut
 8: end if
 9: $\mathbf{c}_{\text{best}} = \mathbf{c} = \text{discretize}(\mathbf{U})$      ➜ Obtain discrete solution; see section 6.2.2
10: $\text{r}_{\text{best}} = \text{r}_{\text{new}} = \text{r}_{\text{old}} = \text{Cut}(\mathbf{c})$      ➜ Initialize the cut value acc. (5.1.7) or acc. (5.1.8)
11: $p = 2$      ➜ Initialize the value of $p$.
12: while $p \geq p_w$ && $\text{r}_{\text{new}} \leq 1.05 \cdot \text{r}_{\text{old}}$ do
13:     Reduce $p$      ➜ Pseudocontinuous reduction acc. (6.2.8)
14:     if normalized then
15:         Find U: $\underset{\mathbf{U}\in\mathcal{G}r(k,n)}{\text{minimize}}\, F_p^{(n)}(\mathbf{U})$ using W      ➜ See section 6.2.
16:     else
17:         Find U: $\underset{\mathbf{U}\in\mathcal{G}r(k,n)}{\text{minimize}}\, F_p(\mathbf{U})$ using W
18:     end if
19:     $\mathbf{c} = \text{discretize}(\mathbf{U})$      ➜ Obtain discrete solution; see section 6.2.2
20:     $\text{r}_{\text{old}} = \text{r}_{\text{new}}$
21:     $\text{r}_{\text{new}} = \text{Cut}(\mathbf{c})$      ➜ Update the cut value acc. (5.1.7) or acc. (5.1.8)
22:     if $\text{r}_{\text{new}} < \text{r}_{\text{best}}$ then
23:         $\text{r}_{\text{best}} = \text{r}_{\text{new}}$
24:         $\mathbf{c}_{\text{best}} = \mathbf{c}$
25:     end if
26: end while
27: return $\mathbf{c}_{\text{best}}, \text{r}_{\text{best}}$      ➜ optimal solution
28: end function
```

value is initialized accordingly in step 10. The main loop of the algorithm in steps $12-25$ terminates if the value of $p$, which is initialized in 11 and reduced in a pseudocontinuous fashion in 13, reaches the final value $p_\omega$ or the cut value stops decreasing monotonically, with a tolerance of 5% on this monotonic reduction. The multiple unormalized or normalized $p$-eigenvectors are estimated on the manifold $\mathcal{G}r(k,n)$ in steps $14-18$, and the discrete solution is obtained in 19. Then the cut values are updated in steps $20-24$ if they are smaller than their value in the previous iteration.

# Chapter 7

# Numerical results for graph clustering

We demonstrate in what follows the effectiveness of the $p$-Grassmann spectral clustering method, summarized in Algorithm 6. We begin by reporting the setup of our numerical experiments, listing the external methods considered in our comparisons, and discussing on the key differences between our approach and the most closely related method considered. Our results on synthetic graphs are presented in Section 7.1, and on graphs emerging from real-world classification problems in Section 7.2.

**Experimental setup**

For all test cases we report results concerning the quality of the cut in terms of RCut (5.1.7) and NCut (5.1.8), unless specified otherwise. The corresponding accuracy of the labelling assignment is again measured in terms of the unsupervised clustering accuracy (ACC $\in [0, 1]$) and the normalized mutual information (NMI $\in [0, 1]$) [109]. We remind here that for both metrics a value of 1 suggests a perfect grouping of the nodes according to the true labels. To this end, we work strictly with graphs that have ground-truth labels, and set the number of clusters $k$ equal to the total number of labelled classes. However, our approach is directly applicable to graphs with no ground-truth information for unsupervised community detection. We use MATLAB R2020a for our implementation, and run experiments on a total of 80 graphs, organized in 2 sets. The first one comprises 27 artificial test cases, with the purpose of demonstrating the impact of different optimization aspects of $p$-Grassmann spectral clustering. The second one includes 53 graphs originating from image classification and text recognition applications. For all methods under consideration we report the mean results after 10 runs. The connectivity and similarity matrices, that compose the adjacency, are created in the same fashion as in Section 4.2.4, with a $k$-nearest neighbors routine and a Gaussian similarity kernel based on Euclidean distances respectively.

The maximum number of Newton iterations for our method is set to 20 for every $p$-level, and the final $p$-level is set to $p_\omega = 1.1$. We fix the parameters of (6.2.8) at $\kappa = 0.9, \theta = 1.25$. This selection results in the following 8 total $p$-levels, i.e. $p = \{2, 1.9, 1.71, 1.539, 1.3851, 1.2466, 1.171, 1.1\}$. When using k-means for the discretization of the $p$-eigenvectors (pGrass-kmeans) we consider 10 orthogonal and 20 random initial guesses. In order to select the best result out of the different k-means runs we use our discrete objective, i.e., RCut or NCut (lower the better) as the primary ranking metric. Then the labelling accuracy metrics (ACC, NMI) are calculated based on the clusters obtained from the minimization of the cut values. When solving (6.2.9) for the clustering of the $p$-eigenvectors (pGrass-disc) the solution is unique.

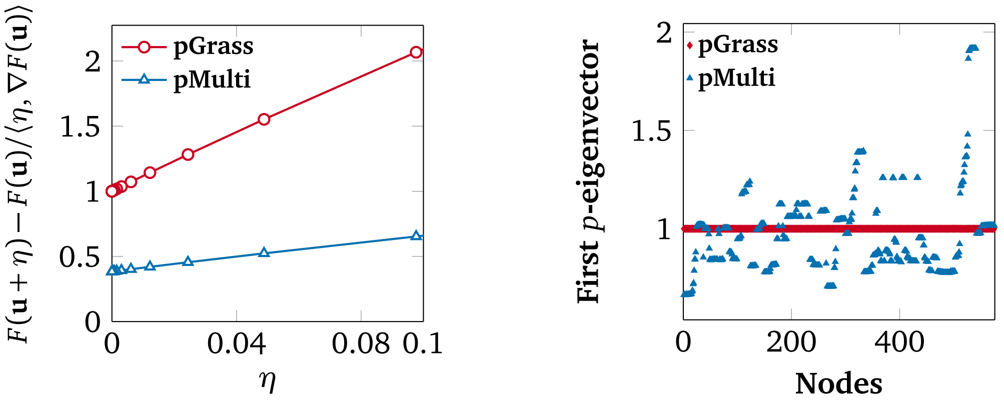
We compare our method against a diverse selection of state-of-the-art clustering algorithms, that target the minimization of RCut or NCut, namely

1. Spec [17]: Traditional direct multiway spectral clustering. We consider the eigenvectors of the combinatorial graph Laplacian $\mathbf{L}$ for unnormalized clustering, and follow the approach of [156] with $\mathbf{L}_{\text{rw}}$ for the normalized case.

2. pSpec [23]: Recursive bi-partitioning with the unnormalized and normalized graph $p$-Laplacian, using a hybrid Newton-gradient descent scheme for the minimization of the nonlinear objective.

3. kCuts [146]: A tight continuous relaxation for the balanced direct $k$-cut problem, using a monotonically descending algorithm for the minimization of the resulting sum of Rayleigh quotients. The method is applicable for the minimization of a variety of discrete graph cut metrics, including RCut and NCut. We use 12 starting initializations for the routine, as suggested by the authors.

4. Graclus [167]: A multilevel algorithm that optimizes for various weighted graph clustering objectives using a weighted kernel k-means objective, thus eliminating the need for eigenvector computations. We use the Kerhighan-Lin [168] algorithm at the coarsest level clustering and perform 10 local searches at each level for increased accuracy. Graclus minimizes directly only the NCut, thus it is omitted from any comparisons in the computation of the RCut and the associated accuracy metrics.



(a) *Accuracy of gradient.* (b) *The first eigenvectors* $\mathbf{v}_p^{(1)}$ *of* $\boldsymbol{\Delta}_p$.

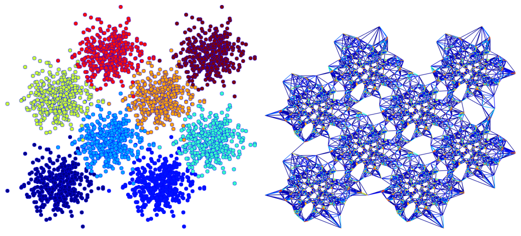
Figure 7.1: *Analysis of key differences between the pMulti [140] and the pGrass (ours) algorithms. (a) The accuracy of the approximated gradients compared against their numerical approximation using a first order Taylor approximation. The $x$-axis denotes the different step size ($\eta$) used in the Taylor expansion (see text for details). The experiment is conducted using the UMIST dataset with a p value of* 1.8 *and* $k = 20$ *number of clusters. (b) The values of the first eigenvector* $\mathbf{v}_p^{(1)}$ *of the graph p-Laplacian* $\boldsymbol{\Delta}_p$ *for the UMIST dataset, estimated by the two methods.*

5. pMulti [140]: The first full eigenvector analysis of the $p$-Laplacian leading to direct multiway clustering, and the most directly related method to our $p$-Grassmann approach. The discrete minimization objective for this approach is the RCut, thus we omit it from any NCut based comparisons.

The code for the methods outlined in 1-4 is available online.[1] We implement method 5, as described in [140], briefly outline here the key differences from our approach, and visualize them in Figure 7.1. The minimization of the constrained multiway $p$-spectral problem (6.2.1) is achieved through an approximated gradient descent scheme which suffers from inaccuracies. This is illustrated in Figure 7.1a, where the ratio of directional derivative $F'$ obtained using a first order Taylor expansion is compared to that of the computed gradient from [140] and ours (6.2.6), for the UMIST dataset [169] at $p = 1.8$.[2] The ratio of $(F(\mathbf{u}+\boldsymbol{\eta})-F(\mathbf{u}))/\langle\eta,\nabla F(\mathbf{u})\rangle$ should ideally approach one as the step size $\eta$

[1]The Spec code is available at: https://github.com/panji530/Ncut9. The pSpec code is available at: https://www.ml.uni-saarland.de/code/pSpectralClustering. The kCuts code is available at: https://www.ml.uni-saarland.de/code/balancedKCuts. The Graclus code is available at: https://www.cs.utexas.edu/users/dml/Software/graclus.html.

[2]The first order Taylor expansion reads $F(\mathbf{u}+\boldsymbol{\eta}) = F(\mathbf{u}) + \langle\boldsymbol{\eta},\nabla F(\mathbf{u})\rangle$, where $\eta$ is the step size.


(a)

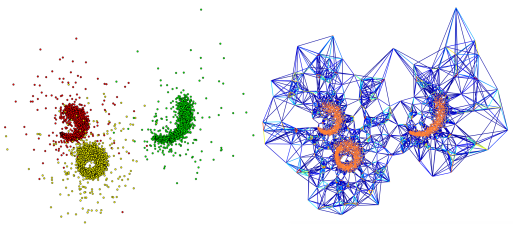

(b)

Figure 7.2: *A subset of the synthetic datasets used. (a) One of the Gaussian datasets considered in Subsection 7.1.2 with $k = 8$ ground-truth clusters, $n = 3200$ nodes and $m = 19319$ edges. b) The worms dataset, considered in Subsection 7.1.3, consists of $n = 5967$ points with three ground-truth communities. The resulting graph has $m = 36031$ edges.*

in the Taylor expansion decreases. However, with the approximated gradient defined in [140] this is not the case. Due to this gradient inaccuracy, fundamental properties of the spectrum of $\boldsymbol{\Delta}_p$ are no longer valid. For example, the degeneracy of the eigenvalues, corresponding to the constant eigenvectors $\mathbf{v} = c \cdot \mathbf{e}$, no longer indicates the number of connected components in the graph. In contrast, our $p$-Grassmann approach, referred to as pGrass, preserves this fundamental property of $\boldsymbol{\Delta}_p$, as illustrated in Figure 7.1b. Furthermore, since the functional $F$ is nonconvex the modified gradient descent approach used in their work has a suboptimal convergence rate, as opposed to the properties of our method. Finally, the linear reduction rate of $p$ in [140] results in fewer evaluations taking place close to $p \approx 1$, and their method considers only the minimization of the unnormalized $p$-spectral objective, associated with the RCut metric.

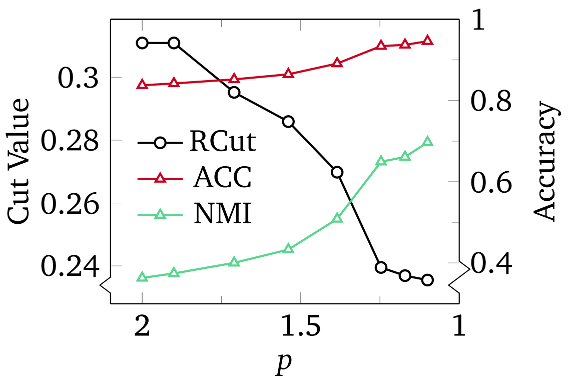
## 7.1 Experiments with artificial data

In this section we focus on artificial datasets widely used as test cases for clustering algorithms, in order to display the behavior of our pGrassmann clustering algorithm in challenging scenarios. Some of the synthetic cases that we consider are visualized in Figure 7.2. In Section 7.1.1 we are interested in studying the effect that the reduction of the value of $p$ has on the clustering result for a graph corrupted by high dimensional noise, and for a set of 16 stochastic block model graphs. In Section 7.1.2, we shift our attention to Gaussian datasets, and study the impact a large number of ground-truth classes has on the accuracy of our method. Last, in Section 7.1.3 we take a closer look at the eigenvectors of the graph $p$-Laplacian and the differences between standard spectral and $p$-spectral embedding on a synthetic dataset with three ground-truth clusters. The results obtained by discretizing the $p$-eigenvectors with the k-means algorithm and by solving (6.2.9) are almost identical for these artificial datasets, therefore in what follows in this section only the results of pGrass-kmeans are presented, and are referred to as pGrass.

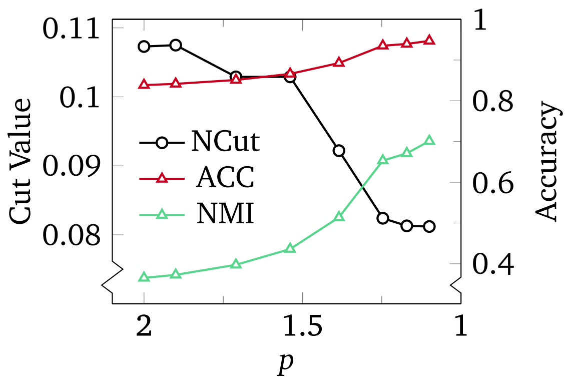
### 7.1.1 Reducing the value of $p$

We initially study the impact of the reduction of the value of $p \in (1,2]$ in (6.2.5) on the high dimensional two-moons dataset, which is frequently used in evaluating graph clustering algorithms. It consists of two half-circles in $\mathbb{R}^2$ embedded into a 100-dimensional space with Gaussian noise $\mathcal{N}(0, \sigma^2 \mathbb{I}_{100})$. This high dimensional noise results in a complex edge formation, as illustrated in Figure 7.3 for $n = 2,000$ points and a variance of $\sigma^2 = 0.02$. In Figure 7.3a we show the effect of reducing $p$ from 2 towards 1 on the resulting RCut and on the associated labelling accuracy metrics (ACC, NMI), and in Figure 7.3b we show the accuracy results of the normalized $p$-Grassmann clustering variant with NCut as its objective. In both cases the monotonic descent of the graph cut metrics leads to nearly perfect accuracies at $p = 1.1$. In Figure 7.3c we present the results obtained by all the methods considered. Our algorithm performs significantly better than Spec, pMulti and Graclus, while it achieves almost identical cut and accuracy values to the pSpec and kCuts methods. The identical results with pSpec are expected in this case, as for a number of clusters $k = 2$ the minimization objective of [23] is equivalent with ours (6.2.5).

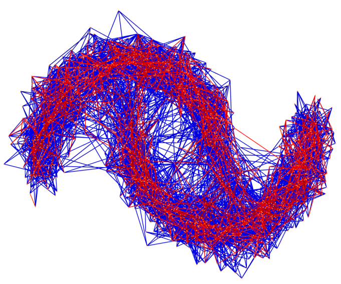
We further demonstrate in Figure 7.4 the effectiveness of the introduced Algorithm 6 in finding the best available clusters even in scenarios where the discrete graph cut metrics are not monotonically descending for a decreasing value of $p$. To this end, we consider the LFR model [170], which is a stochastic block model whose nodes' degrees follow the power law distribution with a parameter $\mu$ controlling what fraction of a node's neighbours is outside the node's block. We follow the approach of [149] and pick $\mu \in [0.1, 0.4]$, with this range giving rise to graphs that contain increasingly noisy clusters for an increasing value of $\mu$. The number of clusters in this benchmark ranges from $k = 17$ to $k = 20$. In Figure 7.4a we show the value of NCut and of the associated accuracy metrics ACC and NMI for the case with $\mu = 0.38$. The monotonic minimization of NCut, with a tolerance of 5% as specified in Algorithm 6, is interrupted at $p = 1.539$. At this $p$-level the graph cut reaches a minimum value of NCut $= 7.187$, with the corresponding accuracy metrics being at their maximum



(a)



(b)

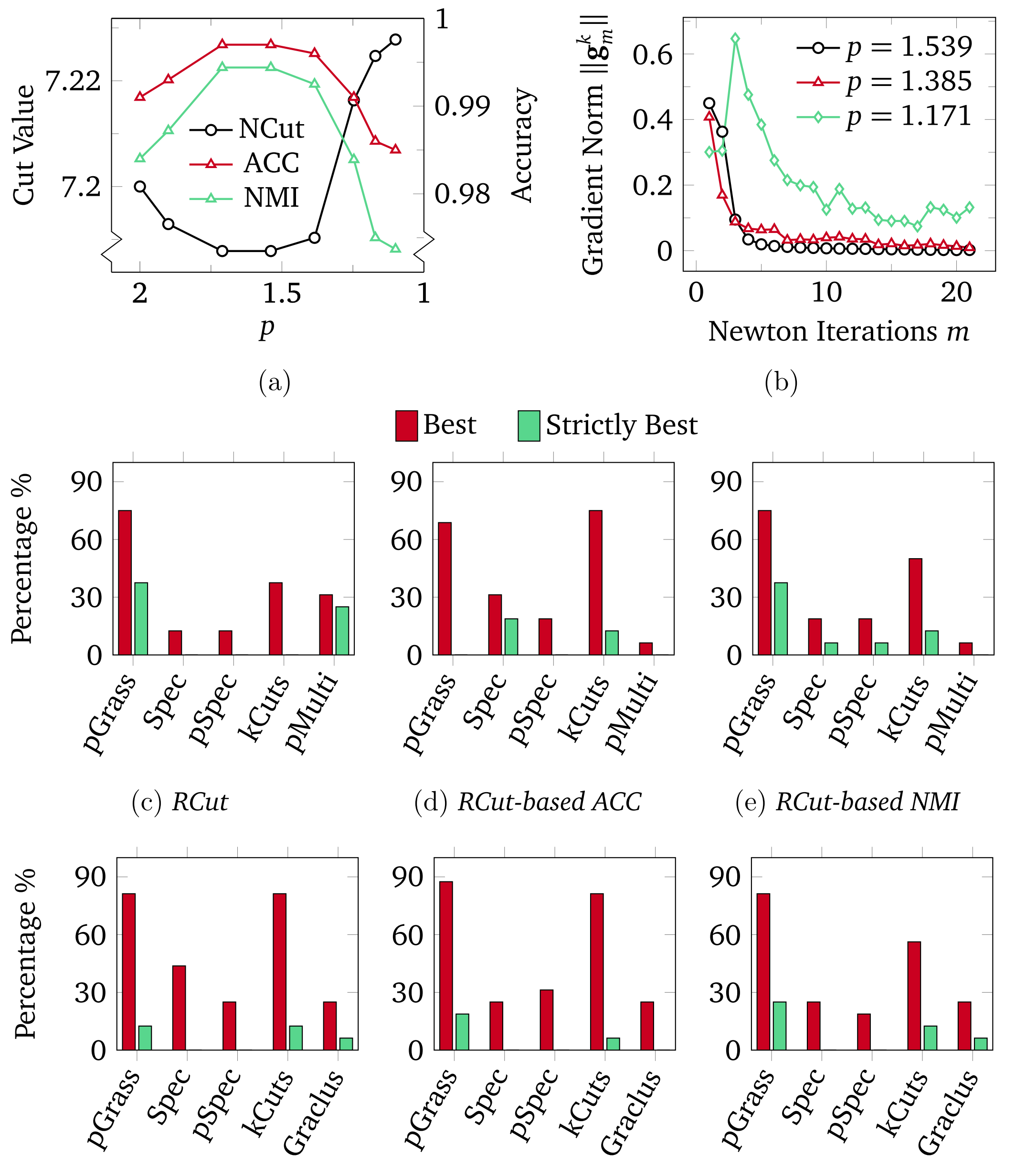


| Method | RCut | ACC | NMI | NCut | ACC | NMI |
|---|---|---|---|---|---|---|
| pGrass | 0.2355 | 0.9460 | 0.6968 | 0.0812 | 0.9470 | 0.7011 |
| Spec | 0.3109 | 0.8375 | 0.3631 | 0.1073 | 0.8385 | 0.3653 |
| pSpec | 0.2355 | 0.9460 | 0.6968 | 0.0812 | 0.9470 | 0.7011 |
| kCuts | 0.2344 | 0.9455 | 0.6953 | 0.0812 | 0.9470 | 0.7010 |
| Graclus | - | - | - | 0.0836 | 0.8405 | 0.4064 |
| pMulti | 0.2677 | 0.8660 | 0.4437 | - | - | - |

(c)

Figure 7.3: *Clustering the two-moons dataset (illustrated). (a) The estimation of RCut and of the associated ACC and NMI with the pGrass algorithm for $p \in [1.1, 2]$. (b) The estimation of NCut and of the associated ACC and NMI with the pGrass algorithm for $p \in [1.1, 2]$. (c) Comparative results for the clustering methods under consideration.*

values $\text{ACC} = 0.9970, \text{NMI} = 0.9944$. Our algorithm stops the reduction of the value of $p$ at this level, however we report the results of the optimization procedure up to the final level of $p = 1.1$, in order to demonstrate that the increasing nonlinearity close to $p \approx 1$ may lead to unfavorable groupings of the nodes. At the final $p$-level the value of the graph cut has ascended to $\text{NCut} = 7.228$, with the values of the accuracy metrics being decreased at $\text{ACC} = 0.9850$ and $\text{NMI} = 0.9737$. In Figure 7.4b we plot the norm of the gradient $\|\mathbf{g}_m^k\|$ over the Newton iterations $m$ for the levels $p = 1.539$ (best solution) and two subsequent levels closer to $p \approx 1$ ($p = 1.385, p = 1.171$). The monotonic minimization of $\|\mathbf{g}_m^k\|$ at the best $p$-level is followed by an increasingly oscillating behavior as $p \to 1$. This showcases that the monotonic minimization of our discrete graph



(a) (b)

(c) *RCut* (d) *RCut-based ACC* (e) *RCut-based NMI*

(f) *NCut* (g) *NCut-based ACC* (h) *NCut-based NMI*

Figure 7.4: *Clustering the LFR benchmark datasets with a noise component $\mu$. (a) NCut values and the associated ACC and NMI for $\mu = 0.38$ for a decreasing value of $p$. (b) Norm of the gradient $\|\mathbf{g}_m^k\|$ over Newton iterations $m$ for $\mu = 0.38$ for three different $p$-levels. (c)–(h) Collective results of the fraction of times a method achieves the best and the strictly best metrics for the entire benchmark with $\mu \in [0.1, 0.4]$.*

cut metric NCut is directly associated with the monotonic decrease of the gradient norm of our continuous objective (6.2.6). The proposed pGrass algorithm is guaranteed to find the best available solution from all $p$-levels under consideration. This is highlighted in Figures 7.4c – 7.4h, where the results for all LFR benchmark datasets (in total 16 cases) for all the methods under consideration are collected. We present the percentage of times a method found the best and the strictly best solution in terms of graph cut metrics (RCut, NCut), and the associated labelling accuracy values in ACC and NMI. Our $p$-Grassmann clustering routine outperforms the external methods Spec, pSpec, pMulti and Graclus in all the metrics under question, and achieves comparable scores with the kCuts algorithm. In particular, the unnormalized pGrass algorithm achieves the best - strictly best solutions in 75% - 37.5% of the cases when minimizing the RCut, in 68.75% - 0% when finding the associated ACC and in 75% - 37.5% when finding the associated NMI. The corresponding percentages for the kCuts algorithm are 37.5% - 0% for RCut, 75% - 12.5% for ACC, and 50% - 12.5% for NMI. The normalized pGrass algorithm achieves the best - strictly best solutions in 81.25% - 12.50% of the cases when minimizing the NCut, in 87.50% - 18.75% when finding the associated ACC, and in 81.25% - 25% when finding the associated NMI. The corresponding percentages for the kCuts algorithm are 81.25% - 12.5% for NCut, 81.25% - 6.25% for ACC, and 56.25% - 12.5% for NMI. The numeric values of the results for the LFR benchmark datasets are summarized in Table B.1 in Appendix B.

### 7.1.2 Increasing the number of clusters

In order to study the clustering quality of our algorithm as the number of clusters ($k$) increases we utilize a set of synthetic Gaussian datasets with an increasing number of ground-truth communities. Each dataset consists of $k$ clusters containing 400 points each. The clusters are generated using a Gaussian distribution with a variance of $\sigma^2 = 0.055$, with the mean of each cluster then placed equidistantly on a 2D square grid (see Figure 7.2a). For the experiment we generated datasets with varying $k = \{2, 5, 8, 18, 25, 32, 41, 50, 61\}$, resulting in 9 graphs with an increasing number of nodes, edges and clusters.

In Figure 7.5 we present the mean values and the standard deviation of the cut metrics and the associated accuracy metric NMI for the Gaussian datasets. In Figure 7.5a we show the results obtained when minimizing RCut and in Figure 7.5b the corresponding NMI. Our pGrass clustering routine finds the minimum RCut in 7/9 cases and the strictly minimum in 5/9 cases. In terms of NMI, pGrass attains the maximum in 8/9 cases and the strictly maximum in 7/9

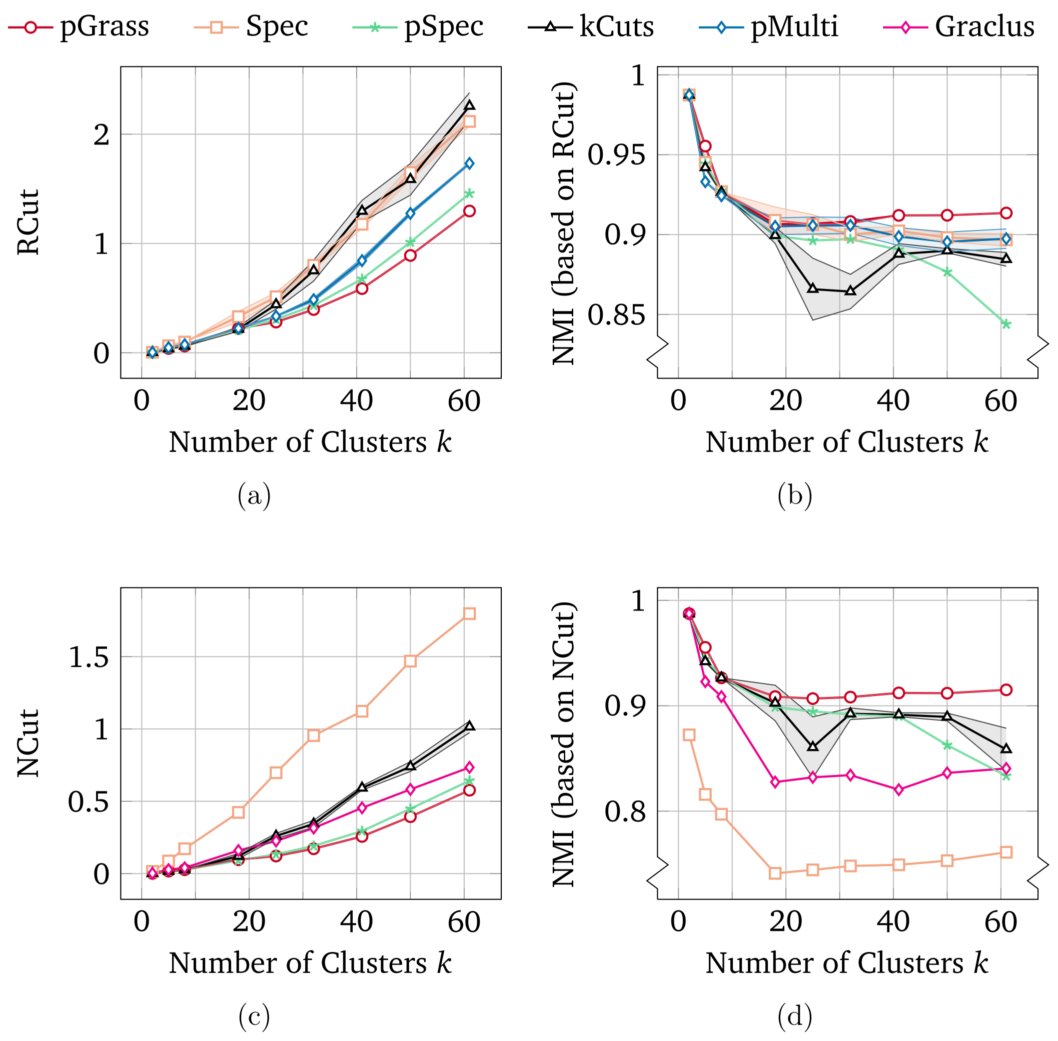


Figure 7.5: *Clustering the Gaussian datasets with an increasing number of clusters k. (a) RCut values for all the methods under consideration. (b) NMI values for all the methods under consideration based on the RCut. (c) NCut values for all the methods under consideration. (d) NMI values for all the methods under consideration based on the NCut.*

cases. The results obtained when attempting to minimize the NCut are shown in Figure 7.5c, with the corresponding NMI values shown in Figure 7.5d. Our algorithm finds the best NCut in 7/9 cases and the strictly best in 5/9 cases. In terms of NMI our algorithm fares the best in 7/9 cases and the strictly best in 6/9 cases. No significant deviations from the mean values are reported for pGrass.

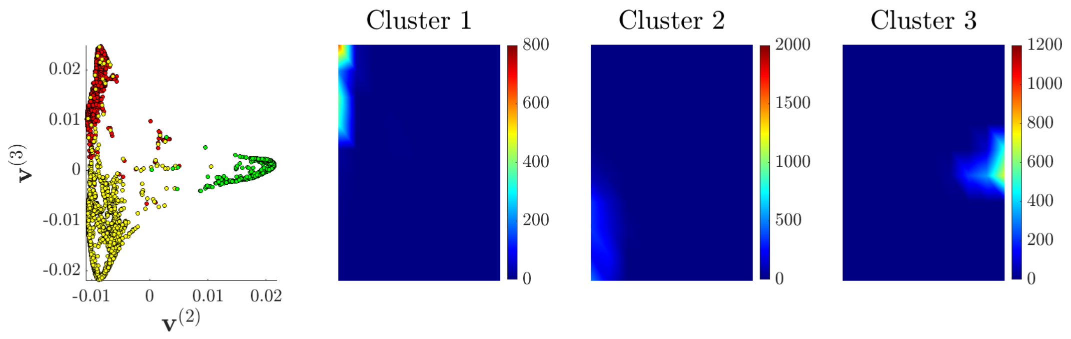
We note that in both the normalized and unormalized experiments the benefits of our method are becoming more evident as the number of clusters $k$ increases. In particular, for $k \geq 25$ pGrass attains the strictly best results in terms of both graph cut and labelling accuracy. This behavior demonstrates that the $p$-Grassmann algorithm is favorable for clustering datasets with a large number of clusters not only from recursive approaches (pSpec, Graclus), which is expected due to the recursive methods' instabilities [155], but also from the direct multiway methods under consideration (Spec, kCuts, pMulti).

### 7.1.3 $p$-spectral embedding

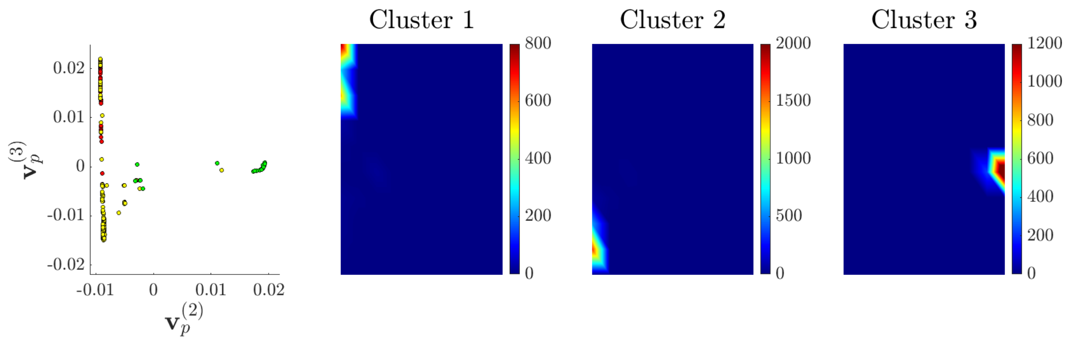
In order to highlight the differences between the embeddings achieved using the eigenvectors of the combinatorial graph Laplacian $\mathbf{\Delta}_2$ and those of the graph $p$-Laplacian $\mathbf{\Delta}_p$, we utilize the "Worms2" dataset [171]. This dataset is composed of three individual worm-like shapes that start from a random position and move towards a random direction. Points are drawn according to a Gaussian distribution with both low and high variance components that are gradually increasing as the points populate the 2D space. The direction of the generation of each worm-like shape is orthogonal to the previous one. The dataset consists of $n = 5,967$ points with three ground-truth communities and the resulting graph has $m = 36,031$ edges (see Figure 7.2b).

We visualize the embedding results obtained by standard spectral clustering (Spec) and our method (pGrass) in Figure 7.6. There are three distinct clusters in the dataset. We utilize the second and third eigenvectors as the $x$- and $y$-axis, respectively. Note that this analysis is not possible for recursive bisection methods, since information regarding only $\mathbf{v}^{(2)}$ is available at each step. The $p$-spectral embedding (Figure 7.6b) organizes the nodes of the dataset in clearly distinguishable groups, as opposed to the spectral embedding (Figure 7.6a). The heat maps illustrate the density of points from each cluster in the two different embeddings ($p = 2, p = 1.2$). We consider ten bins for each direction in order to measure the density for each cluster. The limits of the colorbar are set in both cases to the maximum density values obtained by our method, for a clear comparison.

Upon visual inspection, the pGrass algorithm performs superior to its the traditional Spec routine in the task of creating sharp cuts of the data. Since the last stage of both algorithms is to cluster these points according to their relative distances (see Section 6.2.2 on discretization), the $p$-spectral coordinates of Figure 7.6b are expected to lead to clusters of higher quality. This hypothesis is supported by our numerical results. The quality of the cut achieved by our algorithm (RCut $= 0.0062$) is 49.6% better compared to the one obtained by



(a) *Results obtained by the traditional spectral clustering algorithm (Spec).*



(b) *Results obtained by our proposed p-norm algorithm (pGrass).*

Figure 7.6: *Embedding results for the worms dataset. Starting from the left, the points of the dataset are illustrated using the entries of the second and third eigenvectors of the* $\mathbf{L}$ *in (a), and of* $\mathbf{\Delta}_p$ *for* $p = 1.1$ *in (b), as x and y coordinates. The heat maps that follow depict the density of the points from each of the three clusters.*

spectral clustering (RCut $= 0.0123$). This improvement in terms of graph cut criteria also leads to a slightly better labelling accuracy for this dataset, with our method achieving scores of $\mathrm{ACC} = 0.985, \mathrm{NMI} = 0.93$ as opposed to the traditional spectral routine which achieves scores of $\mathrm{ACC} = 0.981, \mathrm{NMI} = 0.92$. The fact that this big improvement in the quality of the cut is not accompanied by an equivalent increase in the accuracy of the labelling assignment showcases that for the dataset under question the predefined labels can be accurately retrieved with both methods.

## 7.2 Experiments with real-world data

We proceed here with the application of our $p$-Grassmann spectral clustering in graphs emerging from real-world applications. In Section 7.2.1 we consider the problem of classifying facial images according to their labels and in Section 7.2.2 the problem of distinguishing handwritten characters. Both the results obtained after applying k-means for the clustering of the $p$-eigenvectors (pGrass-kmeans), and the results after rotating the eigenvectors to find an optimal partition (pGrass-disc), are presented

### 7.2.1 Classification of facial image datasets

We consider the following publicly available [3] datasets depicting facial expressions

- Olivetti [124]: A set of 10 different facial images of 40 distinct subjects at resolution $64 \times 64$ pixels, taken at different times, varying lighting, facial expressions and facial details.
- Faces95 [172]: A collection of 1440 pictures with resolution $180 \times 200$ pixels from 72 individuals that were asked to move while a sequence of 20 images was taken.
- FACES [173]: A set of images with resolution $2835 \times 3543$ pixels of naturalistic faces of 171 individuals displaying 6 facial expressions. The database consists of 2 sets of pictures per person and per facial expression, resulting in a total of 2052 images. We downsample the images at 20% of their initial resolution to decrease the problem size when creating the adjacency matrix $\mathbf{W}$.

For these datasets the number of nearest neighbors needed for a connected graph is NN = 6 for Olivetti faces and NN = 10 for both Faces95 and FACES. We summarize our results in Table 7.1. For each dataset we report the mean value and the standard deviation of NCut, ACC and NMI achieved by the best method, and the percentage the remaining methods are inferior to that value. Inferiority in percentage values is defined as $I = 100 \cdot \gamma \cdot (e_{\text{ref}} - e_{\text{best}}) / e_{\text{best}}$, where $e_{\text{best}}$ is the best value, $e_{\text{ref}}$ the value it is compared against, and $\gamma = -1$ for minimization scenarios (NCut) and $\gamma = 1$ for maximization ones (ACC, NMI). Our algorithmic variant pGrass-kmeans finds the best NCut result in all three datasets. However, these minimum cut values do not correspond to a maximization of the labelling accuracy metrics. Instead, our algorithmic variant pGrass-disc, which discretizes

[3]The Olivetti dataset is available at https://cam-orl.co.uk/facedatabase.html. The faces95 is available at https://cmp.felk.cvut.cz/spacelib/faces/. The FACES dataset is available after registration at https://faces.mpdl.mpg.de/imeji/.

| | Olivetti | | | Faces95 | | | FACES | | |
|---|---|---|---|---|---|---|---|---|---|
| Method | NCut | ACC | NMI | NCut | ACC | NMI | NCut | ACC | NMI |
| pGrass - kmeans | **3.984**$\pm 8\cdot 10^{-3}$ | -4.15% | -2.28% | **2.658**$\pm 6\cdot 10^{-3}$ | -5.77% | -4.24% | **29.42**$\pm 7\cdot 10^{-3}$ | -3.58% | -2.41% |
| pGrass - disc | -4.50% | **0.716**$\pm 6\cdot 10^{-3}$ | **0.831**$\pm 2\cdot 10^{-3}$ | -4.50% | **0.609**$\pm 4\cdot 10^{-4}$ | **0.758**$\pm 1\cdot 10^{-3}$ | -6.08% | **0.802**$\pm 4\cdot 10^{-3}$ | **0.91**$\pm 2\cdot 10^{-3}$ |
| Spec | -24.84% | -9.19% | -5.27% | -24.84% | -4.23% | -0.90% | -15.05% | -2.50% | -1.23% |
| pSpec | -8.04% | -7.41% | -3.06% | -8.04% | -6.86% | -6.02% | -4.34% | -6.73% | -2.71% |
| kCuts | -1.41% | -6.78% | -3.20% | -1.41% | -10.37% | -7.70% | -7.67% | -13.0% | -6.99% |
| Graclus | -23.10% | -6.36% | -2.25% | -23.11% | -9.25% | -2.38% | -9.98% | -3.70% | -2.56% |

Table 7.1: *Clustering results for the facial image datasets of subsection 7.2.1. Both variants of our algorithm, pGrass-kmeans and pGrass-disc are considered. We report in bold the mean results of the best method, and their standard deviation, and the percentage the remaining methods are inferior to that value.*

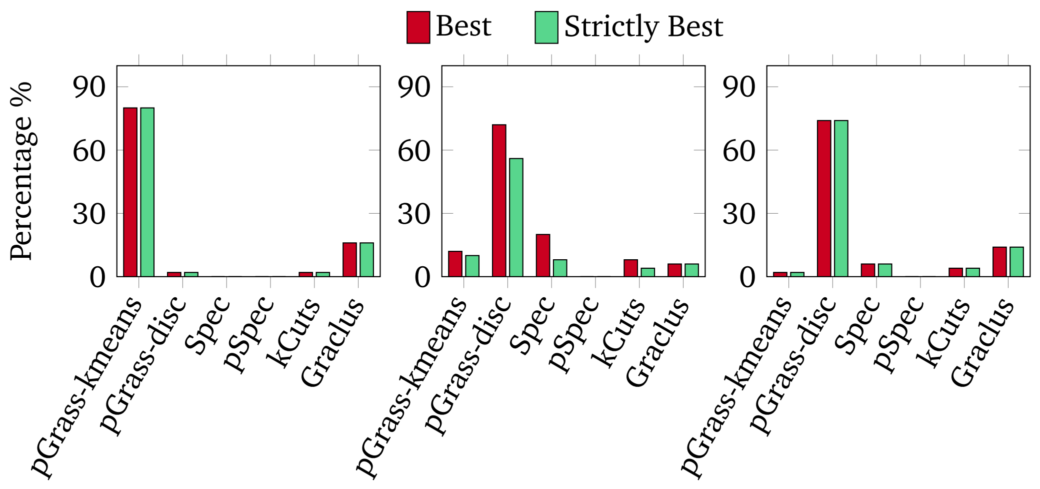
the eigenvectors of the normalized graph $p$-Laplacian $\boldsymbol{\Delta}_p^{(n)}$ with the orthonormal transformation described in [156], achieves the highest ACC and NMI values in all cases. Similarly to the numerical experiments on artificial datasets, no significant deviations ($< 1\%$) from the mean reported values are observed for pGrass.

### 7.2.2 Classification of handwritten characters

For the problem of classifying handwritten characters we consider the Omniglot database [174].[4] It consists of 1623 different handwritten characters from 50 alphabets. Each of the 1623 characters was drawn online via Amazon's Mechanical Turk by 20 different people, with each drawing having dimensions of $105 \times 105$ pixels. For each of these 50 alphabets we consider the problem of grouping the symbols in their respective classes. The number of nearest neighbors in the creation of the adjacency matrix is set to $\mathrm{NN} = 10$ for all cases.

We present the percentage of times a method achieved the best and the strictly best solution in Figure 7.7. In Figure 7.7a we see that our variant pGrass-kmeans obtains the best NCut values in 80% of the cases, with the remaining methods pGrass-disc, Spec, pSpec, kCuts and Graclus in 2%, 0%, 0%, 2%, and 16% respectively. There are no ties in the NCut results, thus best and strictly best percentages are identical. In terms of ACC, illustrated in Figure 7.7b, our variant pGrass-disc find the best solution in 72% of the cases, and the strictly best in 56%. The remaining methods pGrass-kmeans, Spec, pSpec, kCuts, Graclus find the best-strictly best solution in 12%—10%, 20%—8%, 0%—0%, 8.0%—4.0% and 6.0%—6.0% of the cases respectively. Finally, the NMI results of Figure 7.7c indicate that the pGrass-disc variant finds the best solution in 74% of the cases, with the remaining methods pGrass-kmeans, Spec, pSpec, kCuts, Graclus in 2%, 6%,

[4]The Omniglot database is available at https://github.com/brendenlake/omniglot.



(a) *NCut* (b) *NCut-based ACC* (c) *NCut-based NMI*

Figure 7.7: *Clustering the Omniglot database of handwritten digits. The red bar indicated the percentage of times that a method achieved the best solution and the green bar the percentage of times it achieved the strictly best solution. (a) NCut values, (b) ACC values based on NCut, (c) NMI values based on NCut.*

0%, 4%, and 14% respectively. Similarly to the NCut results all NMI solutions are unique, thus best and strictly best percentages are identical. The numerical values of the results for the Omniglot database are summarized in Table B.2 in Appendix B.

### 7.2.3 Discussion of real-world results

All real-world numerical experiments presented above demonstrate that clustering with the pGrass algorithm leads to either obtaining the best (minimum) graph cut values, or the best (maximum) labelling accuracy metrics. In particular, the pGrass-kmeans variant, that discretizes the resulting $p$-eigenvectors from the Grassmannian optimization problem (6.2.5) with the k-means algorithm, showcases superior results in terms of the balanced graph cut metric NCut (5.1.8). This variant (pGrass-kmeans) attains the minimum NCut value from all the methods under consideration in all the facial expression datasets of Section 7.2.1, and the minimum NCut value in 80% of the total 50 handwritten datasets of Section 7.2.2. All these solutions were unique, i.e., none of the external graph clustering methods under consideration reported the same cut. However, as reported in multiple related works [23, 146, 149], the minimization of balanced graph cut metrics does not necessarily lead to a increase in the accuracy of the labelling assignment for real-world data. The creation of the adjacency matrix plays a vital role in this discrepancy, and is an active field of research [12, 13]. We demonstrate that for widely used adjacency matrices, employing a different technique for the discretization of the $p$-eigenvectors (pGrass-disc) leads to favorable labelling accuracy assignments, even if the graph cut values are inferior than the ones by pGrass-kmeans. Rotating the eigenvectors in order to obtain discrete partitions has been reported to be particularly successful in maximizing the clustering accuracy metrics of labelled image data [128, 175], and our numerical experiments further support this observation. The algorithmic variant pGrass-disc minimizes (6.2.9) as suggested in [156], and results in the maximum ACC and NMI for all three facial expressions datasets. In the classification of handwritten digits this variant finds the best ACC in 72%, the strictly best ACC in 56%, and the best NMI in 74% of the cases, with the NMI solutions being unique. This showcases that the $p$-spectral embeddings found by (6.2.5) can lead to the minimization of the balanced graph cut metric (pGrass-kmeans), which is the primary objective of graph partitioning applications, or to the maximization of the labelling assignment accuracy (pGrass-disc), which is the goal in classification problems.

# Chapter 8

# Conclusions

This thesis aims to provide a stand-alone synopsis of research conducted by the author in the field of high dimensional precision and $M$-matrix estimation, and in the field of nonlinear spectral clustering.

Initially, a performant and accurate second-order algorithm was developed, based on the $\ell_1$ regularized optimization of a quadratic approximation of the maximum likelihood estimation problem. The proposed method is suitable for datasets characterized by reduced sparsity and showcases significant performance gains over the current state-of-the-art. This was achieved by exploiting the presence of block structure in the underlying computations, namely, in the approximate inversion of the precision matrix, and in the coordinate descent update that determines the direction of Newton's method. The block structures utilized were retrieved by incorporating a high-performance supernodal sparse Cholesky factorization routine.

Motivated by the effectiveness of this method in problems of very large dimensions, we introduced two algorithms for learning $M$-matrices, that represent graphs whose nodes are non-negatively correlated random variables. Both proposed algorithms are again based on the graphical lasso problem, and are able to incorporate prior available information about the latent graphical structure of the data under question. The first one, SQUIC-fit, is an unconstrained approach that performs two consecutive precision matrix estimations, and utilizes the positively correlated variables identified in the first run as graphical bias for the retrieval of the second precision matrix. Subsequent post-processing of its entries guarantees that the resulting matrix is a positive definite $M$-matrix. The second one, SQUIC-sqp, is a constrained method that enforces the $M$-matrix structure during the optimization procedure. This constrained minimization is achieved by means of a sequential quadratic programming algorithm, with the corresponding KKT system being solved with a preconditioned conjugate gradient method.

This work finally contributes in the area of nonlinear spectral methods with an algorithm for direct multiway $p$-spectral clustering, that reformulates the problem of obtaining multiple eigenvectors of the graph $p$-Laplacian as an unconstrained minimization problem on a Grassmann manifold. Our method reduces the value of $p$ in a pseudocontinuous manner, and ensures that the best result with respect to balanced graph cut metrics is retrieved from the various $p$-levels. The retrieved $p$-eigenvectors lead to either superior graph cut values or labelling accuracy metrics, depending on the method that transforms them into discrete partitions.

For all introduced methods numerical experiments are conducted on artificial cases and real-world instances, including biological, medical, and image data. The consistency of our results, from the synthetic tests to the real-world problems, highlights the effectiveness of the introduced graph learning and clustering algorithms and the broad applicability of the presented work.

# Appendix A

# Derivation of gradient and Hessian for the pGrass algorithm

In this appendix we show the derivation of the Euclidean gradient $g_m^k$ (introduced in (6.2.6)) and the approximate Hessian $h_{mn}^k$ (introduced in (6.2.7)) of the functional $F_p$.

The $m$-th entry of the Euclidean gradient ($\mathbf{g}^k$) of $F_p$ with respect to $\mathbf{u}^k$ is:

$$g_m^k = \frac{\partial F_p}{\partial u_m^k} = \frac{\partial}{\partial u_m^k} \sum_{i,j=1}^{n} \frac{\mathbf{W}_{ij} \left| u_i^k - u_j^k \right|^p}{2\|\mathbf{u^k}\|_p^p} = \frac{\partial}{\partial u_m^k} \frac{A}{B},$$

$$\text{where } A = \sum_{i,j=1}^{n} \mathbf{W}_{ij} \left| u_i^k - u_j^k \right|^p \text{ and } B = 2\|\mathbf{u^k}\|_p^p.$$

$$g_m^k = \frac{1}{B} \frac{\partial A}{\partial u_m^k} - \frac{\partial B}{\partial u_m^k} \frac{A}{B^2} = \frac{1}{B} \left[ \frac{\partial A}{\partial u_m^k} - \frac{\partial B}{\partial u_m^k} \frac{A}{B} \right], \text{ and applying the product rule} \tag{A.0.1}$$

$$\frac{\partial A}{\partial u_m^k} = \frac{\partial}{\partial u_m^k} \sum_{i,j=1}^{n} \mathbf{W}_{ij} \left| u_i^k - u_j^k \right|^p = \sum_{i,j=1}^{n} \mathbf{W}_{ij} p \left| u_i^k - u_j^k \right|^{p-1} \text{sign}(u_i^k - u_j^k) \frac{\partial}{\partial u_m^k} (u_i^k - u_j^k)$$

$$= p \sum_{i,j=1}^{n} \mathbf{W}_{ij} \phi_p(u_i^k - u_j^k) \frac{\partial}{\partial u_m^k} (u_i^k - u_j^k), \text{ since } \phi_p(x) = |x|^{p-1} \text{sign}(x) \tag{A.0.2}$$

$$\frac{\partial}{\partial u_m^k}(u_i^k - u_j^k) = \left\{ \begin{matrix} 1 & \text{if } i = m \text{ and } j \neq m \\ -1 & \text{if } j = m \text{ and } i \neq m \\ 0 & \text{else} \end{matrix} \right\} \tag{A.0.3}$$

Using (A.0.3) in (A.0.2) we get,

$$\begin{aligned}\frac{\partial A}{\partial u_m^k} &= p\sum_{j=1}^{n}\mathbf{W}_{mj}\phi_p(u_m^k-u_j^k)-p\sum_{i=1}^{n}\mathbf{W}_{im}\phi_p(u_i^k-u_m^k)\\ &= p\sum_{j=1}^{n}\Big[\mathbf{W}_{mj}\phi_p(u_m^k-u_j^k)-\mathbf{W}_{jm}\phi_p(u_j^k-u_m^k)\Big]\\ &= 2p\sum_{j=1}^{n}\mathbf{W}_{mj}\phi_p(u_m^k-u_j^k),\text{ since }\phi_p(-x)=-\phi_p(x)\text{ and }\mathbf{W}_{mj}=\mathbf{W}_{jm}.\end{aligned} \tag{A.0.4}$$

$$\begin{aligned}\frac{\partial B}{\partial u_m^k} &= \frac{\partial}{\partial u_m^k}2\|\mathbf{u^k}\|_p^p=\frac{\partial}{\partial u_m^k}2\sum_{i=1}^{n}|u_i^k|^p=2\sum_{i=1}^{n}p|u_i^k|^{p-1}\text{sign}(u_i^k)\frac{\partial u_i^k}{\partial u_m^k}\\ &= 2p\phi_p(u_m^k),\text{ since }\phi_p(x)=|x|^{p-1}\text{sign}(x)\text{ and }\frac{\partial u_i^k}{\partial u_m^k}=\begin{Bmatrix}1 & \text{if } i=m\\ 0 & \text{else}\end{Bmatrix}.\end{aligned} \tag{A.0.5}$$

Substituting (A.0.4) and (A.0.5) in (A.0.1) we get,

$$g_m^k=\frac{p}{\|\mathbf{u^k}\|_p^p}\left[\sum_{j=1}^{n}\mathbf{W}_{mj}\phi_p(u_m^k-u_j^k)-\phi_p(u_m^k)\sum_{i,j=1}^{n}\frac{\mathbf{W}_{ij}\left|u_i^k-u_j^k\right|^p}{2\|\mathbf{u^k}\|_p^p}\right]. \tag{A.0.6}$$

The Euclidean Hessian of $F_p$ with respect to $\mathbf{u}^k$ is the matrix $\mathbf{H^k}$. Its $m$-th row and $l$-th column entry is

$$h_{ml}^k=\frac{\partial g_m^k}{\partial u_l^k}=\frac{\partial}{\partial u_l^k}\frac{p}{\|\mathbf{u^k}\|_p^p}\left[\sum_{j=1}^{n}w_{mj}\phi_p(u_m^k-u_j^k)-\phi_p(u_m^k)\sum_{i,j=1}^{n}\frac{w_{ij}\left|u_i^k-u_j^k\right|^p}{2\|\mathbf{u^k}\|_p^p}\right]. \tag{A.0.7}$$

The Hessian matrix as such (A.0.7) is not sparse and will cause storage problems. Therefore, we neglect the lower rank updates (refer to [23] for details). The existing higher rank term can then be simplified, in the same way with the gradient derivation seen above, to arrive at the approximated Hessian

$$\frac{\partial g_m^k}{\partial u_l^k}\approx\begin{cases}\dfrac{p(p-1)}{\|\mathbf{u}^k\|_p^p}\displaystyle\sum_{j=1}^{n}w_{mj}|u_m^k-u_j^k|^{p-2} & \text{if } m=l,\\[2ex] \dfrac{-p(p-1)}{\|\mathbf{u}^k\|_p^p}w_{ml}|u_m^k-u_l^k|^{p-2} & \text{otherwise.}\end{cases} \tag{A.0.8}$$

# Appendix B

# Full list of clustering numerical results

We present in this Appendix the numerical results for the experiments of Sections 7.1.1 and 7.2.2 on the LFR benchmark datasets and the Omniglot database of handwritten characters respectively. For the number of times each method under consideration found the best and the strictly best solutions we refer to Figures 7.4 and 7.7 for the LFR datasets and the Omniglot cases respectively. The clustering methods under consideration are listed in Chapter 7.

| Case | Measure | pGrass | Spec | pSpec | kCuts | pMulti | Measure | pGrass | Spec | pSpec | kCuts | Graclus |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RCut | **18.82** | **18.82** | -0.1% | **18.82** | -13.4% | NCut | **1.92** | **1.92** | -0.75% | **1.92** | -37.2% |
| $\mu = 0.10$ | ACC | **1.0** | **1.0** | -0.1% | **1.0** | -10.6% | ACC | **1.0** | **1.0** | -0.1% | **1.0** | -5.3% |
| | NMI | **1.0** | **1.0** | -0.2% | **1.0** | -2.6% | NMI | **1.0** | **1.0** | -0.16% | **1.0** | -4.0% |
| | RCut | **21.6** | **21.6** | **21.6** | **21.6** | -17.89% | NCut | **2.18** | **2.18** | -1.1% | **2.18** | -19.3% |
| $\mu = 0.12$ | ACC | **1.0** | **1.0** | **1.0** | **1.0** | -6.3% | ACC | **1.0** | **1.0** | -0.2% | **1.0** | -2.7% |
| | NMI | **1.0** | **1.0** | **1.0** | **1.0** | -2.4% | NMI | **1.0** | **1.0** | -0.34% | **1.0** | -2.0% |
| | RCut | **29.75** | -3.53% | -15.3% | **29.75** | -20.3% | NCut | **2.92** | **2.92** | **2.92** | **2.92** | -36.2% |
| $\mu = 0.14$ | ACC | **1.0** | -8.2% | -10.4% | **1.0** | -7.2% | ACC | **1.0** | **1.0** | **1.0** | **1.0** | -8.5% |
| | NMI | **1.0** | -2.4% | -4.6% | **1.0** | -4.7% | NMI | **1.0** | **1.0** | **1.0** | **1.0** | -5.4% |
| | RCut | **31.37** | -4.6% | **31.37** | **31.37** | -3.0% | NCut | **3.17** | **3.17** | **3.17** | **3.17** | -13.5% |
| $\mu = 0.16$ | ACC | **1.0** | -2.8% | **1.0** | **1.0** | -3.1% | ACC | **1.0** | **1.0** | **1.0** | **1.0** | -2.6% |
| | NMI | **1.0** | -1.7% | **1.0** | **1.0** | -1.6% | NMI | **1.0** | **1.0** | **1.0** | **1.0** | -1.8% |
| | RCut | **30.56** | -0.15% | -1.82% | **30.56** | **30.56** | NCut | **3.21** | -0.1% | **3.21** | **3.21** | -11.8% |
| $\mu = 0.18$ | ACC | **1.0** | -0.1% | -0.7% | **1.0** | **1.0** | ACC | **1.0** | -0.1% | **1.0** | **1.0** | -2.5% |
| | NMI | **1.0** | -0.2% | -1.1% | **1.0** | **1.0** | NMI | **1.0** | -0.19% | **1.0** | **1.0** | -1.6% |
| | RCut | **34.45** | -4.1% | -2.77% | **34.45** | -19.0% | NCut | **3.65** | **3.65** | -1.53% | **3.65** | -11.1% |
| $\mu = 0.20$ | ACC | -0.1% | -3.2% | -12.0% | **1.0** | -24.1% | ACC | **1.0** | -0.1% | -0.5% | **1.0** | -2.4% |
| | NMI | **1.0** | -2.1% | -5.2% | **1.0** | -17.0% | NMI | **1.0** | -0.19% | -0.9% | **1.0** | -2.4% |
| | RCut | **39.01** | -5.0% | -4.0% | -0.34% | -1.00% | NCut | **3.96** | **3.96** | **3.96** | **3.96** | **3.96** |
| $\mu = 0.22$ | ACC | **1.0** | -15.0% | -11.0% | **1.0** | -7.2% | ACC | **1.0** | -0.3% | **1.0** | **1.0** | **1.0** |
| | NMI | **1.0** | -7.2% | -6.7% | -0.5% | -3.3% | NMI | **1.0** | -0.2% | -0.9% | **1.0** | **1.0** |
| | RCut | **49.76** | -7.3% | -7.2% | -1.4% | -2.3% | NCut | **5.02** | **5.02** | **5.02** | **5.02** | -8.8% |
| $\mu = 0.24$ | ACC | -13.8% | **0.88** | -12.8% | -10.1% | -39.0% | ACC | **1.0** | -0.3% | **1.0** | **1.0** | -3.3% |
| | NMI | -0.46% | **0.94** | -7.3% | -3.8% | -16.7% | NMI | **1.0** | -0.19% | -0.9% | -0.2% | -1.9% |
| | RCut | **41.95** | -4.5% | -1.2% | -1.9% | -11.4% | NCut | **4.4** | -0.14% | -41.0% | **4.4** | **4.4** |
| $\mu = 0.26$ | ACC | -1.9% | -5.2% | -13.7% | **0.87** | -48.0% | ACC | **1.0** | -0.1% | -9.2% | **1.0** | **1.0** |
| | NMI | -1.0% | -7.3% | -8.7% | **0.94** | -32.5% | NMI | **1.0** | -0.17% | -14.7% | -0.18% | **1.0** |
| | RCut | **53.38** | -8.4% | -5.2% | -4.4% | -1.0% | NCut | **5.59** | -0.1% | -0.4% | **5.59** | -6.6% |
| $\mu = 0.28$ | ACC | **0.83** | -44.4% | -19.6% | **0.83** | -152.7% | ACC | **1.0** | -0.2% | -0.4% | **1.0** | -3.1% |
| | NMI | -0.3% | -23.3% | -6.3% | **0.894** | -78.3% | NMI | **1.0** | -0.5% | -0.7% | -0.18% | -1.8% |
| | RCut | -1.74% | -10.5% | -10.9% | -4.9% | **56.0** | NCut | **6.00** | -0.14% | -1.96% | **6.00** | **6.00** |
| $\mu = 0.30$ | ACC | **0.77** | -26.4% | -68.2% | **0.77** | -67.1% | ACC | **1.0** | -0.3% | -1.9% | **1.0** | **1.0** |
| | NMI | **0.866** | -18.3% | -37.8% | -0.2% | -34.2% | NMI | **1.0** | -0.52% | -3.0% | -0.74% | **1.0** |
| | RCut | **43.73** | -13.5% | -9.4% | -4.9% | -0.35% | NCut | **5.40** | -0.48% | -1.0% | -0.04% | -11.8% |
| $\mu = 0.32$ | ACC | **0.51** | -3.7% | -17.5% | **0.51** | -126.1% | ACC | **0.998** | -0.8% | -2.0% | -0.2% | -6.9% |
| | NMI | -1.6% | -6.4% | **0.67** | -0.3% | -95.0% | NMI | **0.996** | -1.5% | -3.5% | -0.34% | -4.5% |
| | RCut | -1.47% | -9.7% | -14.0% | -6.5% | **46.24** | NCut | -0.1% | -0.6% | -1.4% | -0.1% | **5.74** |
| $\mu = 0.34$ | ACC | -7.9% | **0.53** | -57.4% | -94.9% | -247.7% | ACC | -0.1% | -0.8% | -1.7% | **0.995** | **0.995** |
| | NMI | **0.62** | -0.7% | -4.7% | -35.6% | -213.9% | NMI | -0.19% | -1.6% | -2.9% | -0.01% | **0.990** |
| | RCut | -1.3% | -6.3% | -19.6% | -4.5% | **56.11** | NCut | -0.04% | -1.00% | -5.9% | **7.216** | -0.02% |
| $\mu = 0.36$ | ACC | **0.38** | -7.8% | -15.4% | **0.38** | -51.2% | ACC | -0.6% | -2.3% | -7.2% | **0.998** | -0.5% |
| | NMI | **0.51** | -2.4% | -8.0% | -1.4% | -42.1% | NMI | -1.1% | -3.7% | -11.0% | **0.996** | -0.74% |
| | RCut | -0.08% | -10.3% | -14.7% | -0.2% | **56.25** | NCut | **7.18** | -1.0% | -6.8% | -0.08% | -4.5% |
| $\mu = 0.38$ | ACC | -0.9% | **0.34** | -54.4% | -43.8% | -117.5% | ACC | **0.992** | -2.0% | -9.0% | -0.4% | -4.4% |
| | NMI | **0.51** | -12.6% | -36.3% | -61.0% | -156.0% | NMI | **0.996** | -3.8% | -14.2% | -0.76% | -3.6% |
| | RCut | **53.45** | -10.9% | -18.9% | -1.9% | -1.4505 | NCut | -0.2% | -0.9% | -8.1% | **7.99** | -3.5% |
| $\mu = 0.40$ | ACC | **0.25** | -12.8% | **0.25** | -28.1% | -39.0% | ACC | **0.998** | -2.15% | -12.2% | -0.8% | -3.9% |
| | NMI | **0.43** | -64.2% | -58.6% | -69.2% | -102.0% | NMI | -1.55% | -4.1% | -17.1% | **0.994**% | -3.7% |

Table B.1: *Clustering results for the LFR benchmark datasets with an increasing noise component $\mu \in [0.1, 0.4]$. The numerical value of the best solutions for each metric is presented in bold. The percentage signs indicate how much inferior a solution is to the best solution.*

| Measure | Case | pGr-A | pGr-B | Spec | pSpec | kCuts | Graclus | Case | pGr-A | pGr-B | Spec | pSpec | kCuts | Graclus |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NCut | | **13.63** | -2.57% | -3.54% | -8.95% | -2.89% | -0.63% | | -1.80% | -3.50% | -6.10% | -12.62% | **4.85** | -3.68% |
| ACC | AngloSaxon | -5.04% | **0.358** | -2.46% | -30.83% | -14.28% | -11.23% | Ojibwe | -2.65% | **0.41** | -0.88% | -0.88% | -8.43% | -7.42% |
| NMI | | -2.14% | **0.477** | -1.06% | -9.99% | -5.16% | -5.68% | | -0.59% | **0.46** | -1.61% | -1.07% | -2.51% | -3.90% |
| NCut | | **15.42** | -1.86% | -2.12% | -6.23% | -1.05% | -0.22% | | -0.50% | -2.42% | -2.19% | -5.43% | -3.30% | **0.319**% |
| ACC | Arcadian | -2.85% | **0.29** | -0.10% | -8.04% | -2.85% | -5.02% | Sanskrit | -1.46% | **0.166** | **0.166** | -5.31% | -2.96% | -6.86% |
| NMI | | -4.89% | **0.384** | -2.32% | -16.53% | -7.29% | -5.20% | | -1.39% | **0.33** | -1.33% | -9.70% | -4.27% | -4.66% |
| NCut | | **23.02** | -2.64% | -2.83% | -4.20% | -0.96% | -1.12% | | **12.59** | -3.00% | -4.11% | -3.71% | -1.70% | -3.59% |
| ACC | Armenian | **0.24** | -2.65% | -4.88% | -10.87% | -3.77% | -3.18% | Syriac | -8.01% | **0.263** | -4.28% | -17.46% | -8.01% | -7.04% |
| NMI | | -1.08% | **0.40** | -1.08% | -8.18% | -4.16% | -2.43% | | -8.10% | **0.338** | -2.57% | -18.41% | -10.21% | -8.80% |
| NCut | | **14.31** | -3.42% | -4.92% | -5.79 % | -2.14 % | -2.59 % | | **28.84** | -5.21% | -7.44% | -6.45% | -3.83% | -3.05% |
| ACC | Balinese | -4.16% | -4.16% | -6.85% | -8.68 % | **0.26** | -2.44 % | Tifinagh | -1.68% | **0.279** | -0.98% | -13.69% | -9.24% | -2.01% |
| NMI | | -4.02% | -2.84% | -7.09% | -8.56% | -3.2% | **0.33** | | -4.97% | **0.498** | -0.97% | -9.63% | -8.74% | -4.42% |
| NCut | | **26.31** | -1.88% | -2.34% | -6.52% | -1.20% | -0.75% | | **8.90** | -4.99% | -6.97% | -6.89% | -1.61% | -5.15% |
| ACC | Bengali | -8.57% | **0.223** | -4.16% | -19.25% | -2.06% | -10.29% | Angelic | -11.90% | -3.30% | **0.47** | -20.51% | -12.57% | -14.63% |
| NMI | | -4.41% | -0.75% | -1.36% | -9.46% | **0.40** | -2.73% | | -5.86% | **0.538** | -0.73% | -18.12% | -9.14% | -6.55% |
| NCut | | **4.39** | -2.87% | -7.29% | -5.78% | -0.81% | -10.92% | | -1.09% | -3.08% | -5.85% | -7.18% | -2.44% | **13.26** |
| ACC | Blackfoot | -7.93% | -4.82% | -6.86% | -17.22% | -4.82% | **0.389** | Atemayar | -7.30% | **0.26** | -2.44% | -15.56% | -9.01% | -18.61% |
| NMI | | -8.24% | -4.12% | -8.92% | -9.02% | -0.27% | **0.452** | | -4.62% | **0.351** | -0.54% | -12.14% | -6.36% | -8.67% |
| NCut | | **13.76** | -4.83% | -6.82% | -4.90% | -1.92% | -1.99% | | **16.49** | -4.76% | -5.54% | -4.86% | -1.16% | -0.39% |
| ACC | Braille | -16.69% | **0.298** | -2.12% | -27.24% | -13.28% | -17.61% | Atlantean | -12.05% | **0.34** | **0.34** | -18.81% | -14.92% | -5.35% |
| NMI | | -11.61% | **0.387** | -1.76% | -18.21% | -12.71% | -11.90% | | -8.53% | -0.79% | -3.12% | -16.91% | -11.42% | **0.422** |
| NCut | | **19.40** | -3.76% | -3.89% | -5.73% | -1.48% | -1.30% | | **11.66** | -4.92% | -6.80% | -5.25% | -1.82% | -2.72% |
| ACC | Burmese | -5.39% | **0.23** | -1.32% | -15.45% | **0.23** | -3.96% | Aurek-Besh | -3.21% | **0.37** | -3.21% | -21.53% | -1.04% | -8.46% |
| NMI | | -3.50% | -0.43% | -0.89% | -9.92% | -4.40% | **0.372** | | -2.34% | **0.47** | -1.67% | -9.71% | -2.05% | -5.24% |
| NCut | | **16.79** | -3.28% | -4.25% | -8.08% | -0.80% | -2.69% | | **15.25** | -3.67% | -4.53% | -6.37% | -2.40% | -1.86% |
| ACC | Cyrillic | -0.45% | -0.18% | **0.334** | -19.44% | -0.90% | -9.38% | Avesta | -6.11% | **0.27** | -0.72% | -21.94% | -1.44% | -7.74% |
| NMI | | -2.91 % | **0.467** | -0.65% | -15.15% | -2.64% | -4.19% | | -6.36% | **0.366** | -2.00% | -20.24% | -8.14% | -5.83% |
| NCut | | **10.37** | -4.34% | -4.35% | -8.21% | -2.58% | -3.40% | | -0.48% | -3.36% | -4.02% | -3.57% | -1.20% | **15.11**% |
| ACC | Aramaic | -1.32% | **0.352** | -4.73% | -13.98% | -2.65% | -3.34% | Ge-ez | -10.00% | **0.275** | **0.275** | -16.28% | -17.22% | -15.30% |
| NMI | | -0.28% | **0.427** | -0.99% | -16.10% | -0.12% | -2.64% | | -4.42% | -5.41% | **0.355** | -6.29% | -6.55% | -0.82% |
| NCut | | **15.97** | -5.47% | -6.89% | -3.90% | -0.83% | -1.46% | | **27.84** | -3.96% | -5.82% | -5.11% | -1.52% | -0.03% |
| ACC | Futurama | -6.53% | **0.344** | -1.12% | -13.30% | -14.01% | -15.46% | Glagolitic | -8.02% | **0.3** | -1.12% | -23.97% | -5.10% | -10.25% |
| NMI | | -9.48% | **0.446** | -0.59% | -8.21% | -15.66% | -9.35% | | -5.13% | **0.45** | -1.21% | -19.61% | -4.14% | -2.25% |
| NCut | | **18.70** | -1.63% | -4.70% | -5.19% | -2.98% | -2.38% | | **24.96** | -2.17% | -3.62% | -5.87% | -1.25% | -2.79% |
| ACC | Georgian | -2.89% | **0.313** | -1.62% | -11.08% | -8.21% | -3.29% | Gurmukhi | -1.16% | -5.52% | -8.15% | -5.52% | -5.52% | **0.19** |
| NMI | | -1.92% | **0.488** | -1.66% | -4.82% | -5.23% | -1.92% | | -2.27% | -1.01% | -1.61% | -3.98% | -1.24% | **0.36** |
| NCut | | **22.37** | -2.06% | -2.52% | -5.40% | -3.20% | -0.89% | | -0.26% | -3.01% | -3.68% | -5.51% | -1.21% | **24.32** |
| ACC | Grantha | -0.35% | -0.50% | **0.341** | -9.72% | -3.54% | -6.91% | Kannada | -2.96% | **0.21** | -5.47% | -4.17% | -9.44% | -2.96% |
| NMI | | -0.14% | -2.05% | **0.498** | -4.71% | -3.99% | -2.62% | | -2.35% | **0.37** | -0.75% | -4.23% | -5.61% | -0.70% |
| NCut | | **13.19** | -3.60% | -6.96% | -6.47% | -1.15% | -1.12% | | **8.91** | -2.60% | -4.86% | -6.40% | -0.26% | -3.59% |
| ACC | Greek | **0.329** | -2.75% | -5.34% | -22.52% | -11.29% | -8.22% | Keble | -2.57% | **0.30** | 0.00% | -10.47% | -3.26% | -2.57% |
| NMI | | **0.410** | -2.91% | -2.04% | -13.63% | -6.00% | -2.60% | | -3.40% | -0.30% | -1.36% | -10.41% | -4.77% | **0.402** |
| NCut | | **29.86** | -2.62% | -3.44% | -4.83% | -0.86% | -0.33% | | **29.38** | -1.71% | -3.49% | -4.66% | -0.95% | -0.39% |
| ACC | Gujarati | -1.54% | **0.21** | -3.57% | -14.70% | -4.14% | -10.33% | Malayalam | -5.58% | **0.22** | -1.98% | -10.65% | -0.50% | -6.14% |
| NMI | | -3.08% | **0.41** | -1.13% | -8.74% | -4.20% | -3.69% | | -7.34% | **0.41** | -1.72% | -13.90% | -6.25% | -9.88% |
| NCut | | **11.73** | -4.09% | -4.29% | -4.49% | -1.67% | -0.31% | | -0.16% | -2.07% | -4.92% | -4.88% | -1.08% | **24.08** |
| ACC | Hebrew | -9.37% | **0.248** | -5.16% | -16.39% | -3.20% | -4.19% | Manipuri | -7.23% | **0.22** | -1.69% | -16.31% | -2.30% | -7.23% |
| NMI | | -4.18% | **0.333** | -0.33% | -13.11% | -4.08% | -6.20% | | -2.02% | **0.369** | -0.22% | -10.42% | -0.55% | -5.40% |
| NCut | | -0.70% | -3.96% | -4.21% | -7.52% | -2.01% | **33.68** | | **17.49** | -2.31% | -4.18% | -5.07% | -1.33% | -2.51% |
| ACC | Hiragana | -6.25% | -8.30% | -2.36% | -13.62% | **0.268** | -9.58% | Mongolian | -7.41% | **0.27** | -1.91% | -20.30% | -11.92% | -7.41% |
| NMI | | -6.40% | **0.452** | -1.16% | -9.18% | -3.93% | -5.90% | | -4.27% | **0.39** | -2.39% | -12.58% | -8.48% | -5.03% |
| NCut | | -4.73% | **4.55** | -10.97% | -9.34% | -5.28% | -7.74% | | **29.29** | -3.53% | -4.53% | -4.10% | -1.22% | -1.14% |
| ACC | Inuktitut | -6.44% | **0.515** | -1.86% | -9.26% | -0.60% | -22.21% | Slavonic | -5.94% | **0.24** | -2.48% | -20.34% | -5.94% | -9.03% |
| NMI | | -2.57% | **0.582** | -1.25% | -7.59% | -2.37% | -6.95% | | -6.85% | **0.41** | -1.72% | -17.94% | -6.76% | -5.30% |
| NCut | | **26.18** | -3.49% | -4.70% | -6.39% | -1.32% | -1.12% | | **28.63** | -2.68% | -3.06% | -4.56% | -0.79% | -0.55% |
| ACC | Katakana | -6.43% | **0.246** | -4.49% | -7.40% | -10.47% | -9.45% | Oriya | -6.89% | **0.203** | -1.09% | -5.06% | -3.88% | -10.01% |
| NMI | | -5.29% | **0.435** | -1.63% | -10.72% | -5.32% | -3.54% | | -4.33% | **0.37** | -0.46% | -5.36% | -2.95% | -2.61% |
| NCut | | -0.04% | -2.26% | -3.20% | -7.17% | -1.76 % | **22.47** | | **14.43** | -2.10% | -4.00% | -5.58% | -0.72% | -1.05% |
| ACC | Korean | **0.23** | -3.31% | -5.65% | -16.90% | -2.19% | -1.08% | Sylhetti | -3.68% | -6.67% | -0.91% | -2.77% | -14.29% | **0.2%** |
| NMI | | -1.51% | **0.40** | -3.01% | -11.13% | -3.99% | -1.97% | | -5.43% | **0.293** | -1.52% | -8.19% | -9.36% | -0.93% |
| NCut | | **11.91** | -3.66% | -4.79% | -3.54% | -0.34% | -1.32% | | **12.02** | -5.93% | -6.23% | -4.66% | -3.97% | -3.99% |
| ACC | Magi | -9.09% | -0.76% | **0.33** | -13.79 % | -3.94% | -13.79% | Syriac-Serto | -5.97% | **0.31** | **0.308** | -8.39% | -14.50% | -3.66% |
| NMI | | -3.32% | -1.00% | -2.36% | -14.31% | **0.373** | -2.98% | | -2.58% | -2.99% | **0.41** | -14.43% | -16.87% | -5.46% |
| NCut | | **12.76** | -4.37% | -4.42% | -5.08% | -2.15% | -1.46% | | **13.7** | -2.43% | -4.42% | -7.76% | -1.25% | -2.28% |
| ACC | Latin | -6.35% | **0.296** | -1.44% | -7.82% | -10.92% | -7.82% | Tengwar | -7.91% | **0.30** | -4.17% | -7.14% | -4.17% | -7.91% |
| NMI | | -4.58% | -2.18% | -1.26% | -9.13% | -7.29 % | **0.426** | | -6.39% | **0.37** | -0.73% | -5.70% | -2.33% | -0.05% |
| NCut | | **21.85** | -3.43% | -4.72% | -7.05% | -1.94% | -2.88 % | | **25.38** | -2.76% | -2.76% | -5.59% | -1.06% | -0.30% |
| ACC | Malay | **0.213** | **0.213** | -10.27% | -10.27% | **0.213** | -2.99% | Tibetan | **0.24** | -3.11% | -3.65% | -21.49% | -5.32% | -6.46% |
| NMI | | -4.27% | **0.395** | -0.89% | -8.39% | -2.86% | -1.70% | | -0.67% | **0.41** | -0.77% | -14.48% | -4.88% | -6.54% |
| NCut | | **24.10** | -3.21% | -3.73% | -4.96% | -1.38% | -2.40% | | **14.29** | -3.49% | -4.29% | -2.41% | -1.37% | -2.71% |
| ACC | Mkhedruli | -13.96% | **0.25** | -0.60% | -16.49% | -16.49% | -6.30% | ULOG | -3.87% | **0.26** | -0.74% | -16.50% | -3.87% | -11.66% |
| NMI | | -8.05% | **0.42** | -1.70% | -15.27% | -9.57% | -5.59% | | -5.56% | **0.36** | -1.06% | -13.17% | -5.13% | -10.47% |
| NCut | | **14.91** | -3.26% | -4.41% | -6.80% | -1.29% | -3.09% | | -0.10% | -2.98% | -5.69% | -7.39% | -1.11% | **8.71** |
| ACC | N-Ko | -2.45% | **0.318** | -0.47% | -9.38% | -1.95% | -7.14% | Tagalog | **0.37** | -7.57% | -2.42% | -17.44% | -8.47% | -7.57% |
| NMI | | -1.24% | **0.464** | -0.43% | -8.78% | -2.67% | -1.51% | | -1.37% | **0.41** | -0.39% | -6.96% | -3.17% | -0.69% |

Table B.2: *Clustering results for the Omniglot database. The numerical value of the best solutions for each metric is presented in bold. The percentage signs indicate how much inferior a solution is to the best solution. pGr-A and pGr-B refer to the algorithmic variants pGrass-kmeans and pGrass-disc, respectively.*

# Bibliography

[1] S. L. Lauritzen. *Graphical models*. Number 17 in Oxford Statistical Science Series. Clarendon Press, 1996.

[2] H. Rue and L. Held. *Gaussian Markov Random Fields: Theory And Applications (Monographs on Statistics and Applied Probability)*. Chapman & Hall/CRC, 2005.

[3] A. P. Dempster. Covariance Selection. *Biometrics*, 28(1):157, mar 1972. doi: 10.2307/2528966.

[4] J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 12 2007. doi: 10.1093/biostatistics/kxm045.

[5] C.-J. Hsieh, M. A. Sustik, I. S. Dhillon, and P. K. Ravikumar. Sparse Inverse Covariance Matrix Estimation Using Quadratic Approximation. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2330–2338. Curran Associates, Inc., 2011.

[6] C.-J. Hsieh, M. A. Sustik, I. S. Dhillon, and P. K. Ravikumar. Quic: Quadratic approximation for sparse inverse covariance estimation. *Journal of Machine Learning Research*, 15(83):2911–2947, 2014.

[7] M. Bollhöfer, A. Eftekhari, S. Scheidegger, and O. Schenk. Large-scale Sparse Inverse Covariance Matrix Estimation. *SIAM Journal on Scientific Computing*, 41(1):A380–A401, 2019. doi: 10.1137/17M1147615.

[8] A. Eftekhari, M. Bollhöfer, and O. Schenk. Distributed Memory Sparse Inverse Covariance Matrix Estimation on High-Performance Computing Architectures. In *ACM/IEEE International Conference on High Performance Computing, Networking, Storage and Analysis (SC18)*, 2018. doi: 10.1109/SC.2018.00023.

[9] A. Eftekhari, D. Pasadakis, M. Bollhöfer, S. Scheidegger, and O. Schenk. Block-enhanced precision matrix estimation for large-scale datasets. *Journal of Computational Science*, 53:101389, 2021. doi: 10.1016/j.jocs.2021.101389.

[10] E. Bølviken. Probability inequalities for the multivariate normal with non-negative partial correlations. *Scandinavian Journal of Statistics*, 9(1):49–58, 1982.

[11] S. Karlin and Y. Rinott. M-matrices as covariance matrices of multinormal distributions. *Linear Algebra and its Applications*, 52-53:419–438, 1983. doi: 10.1016/0024-3795(83)80027-5.

[12] L. Stanković, D. Mandic, M. Daković, M. Brajović, B. Scalzo, S. Li, and A. G. Constantinides. Data analytics on graphs part iii: Machine learning on graphs, from graph topology to applications. *Foundations and Trends® in Machine Learning*, 13(4):332–530, 2020. doi: 10.1561/2200000078-3.

[13] F. Xia, K. Sun, S. Yu, A. Aziz, L. Wan, S. Pan, and H. Liu. Graph learning: A survey. *IEEE Transactions on Artificial Intelligence*, 2(2):109–127, 2021. doi: 10.1109/TAI.2021.3076021.

[14] D. Pasadakis, M. Bollhöfer, and O. Schenk. Sparse quadratic approximation for graph learning. April 2022. doi: 10.36227/techrxiv.19635990.v1.

[15] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(2):298–305, 1973.

[16] A. Pothen, H. D. Simon, and K.-P. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal on Matrix Analysis and Applications*, 11(3):84521, May 1990. doi: 10.1137/0611030.

[17] U. Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17 (4):395–416, December 2007. doi: 10.1007/s11222-007-9033-z.

[18] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, NIPS'01, page 849–856, Cambridge, MA, USA, 2001. MIT Press. doi: 10.5555/2980539.2980649.

[19] C. E. Bichot and P. Siarry. *Graph Partitioning*. ISTE. Wiley, 2013.

[20] S. Amghibech. Bounds for the largest p-Laplacian eigenvalue for graphs. *Discrete Mathematics*, 306(21):2762 – 2771, 2006. doi: 10.1016/j.disc.2006.05.012.

[21] S. T. Wierzchoń and M. A. Kłopotek. *Spectral Clustering*, pages 181–259. Springer International Publishing, Cham, 2018. doi: 10.1007/978-3-319-69308-8_5.

[22] J. Cheeger. *A lower bound for the smallest eigenvalue of the Laplacian*, pages 195–199. Princeton Univ. Press, Princeton, 1969.

[23] T. Bühler and M. Hein. Spectral clustering based on the graph p-Laplacian. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 81–88, New York, NY, USA, 2009. ACM. doi: 10.1145/1553374.1553385.

[24] X. Bresson, T. Laurent, D. Uminsky, and J. von Brecht. Multiclass total variation clustering. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[25] A. Edelman, T. A. Arias, and S. T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, 20(2):303–353, April 1999. doi: 10.1137/S0895479895290954.

[26] D. Pasadakis, C. L. Alappat, O. Schenk, and G. Wellein. Multiway p-spectral graph cuts on Grassmann manifolds. *Machine Learning*, 111(2):791–829, Feb 2022. doi: 10.1007/s10994-021-06108-1.

[27] T. Simpson, D. Pasadakis, D. Kourounis, K. Fujita, T. Yamaguchi, T. Ichimura, and O. Schenk. Balanced graph partition refinement using the graph p-Laplacian. In *Proceedings of the Platform for Advanced Scientific Computing Conference*, PASC '18, New York, NY, USA, 2018. Association for Computing Machinery. doi: 10.1145/3218176.3218232.

[28] A. Eftekhari, L. Gaedke-Merzhaeuser, D. Pasadakis, M. Bollhöfer, S. Scheidegger, and O. Schenk. Large-scale precision matrix estimation with SQUIC. *Available at SSRN*, 2021. doi: 10.2139/ssrn.3904001.

[29] A. Berge, A. C. Jensen, and A. H. Schistad Solberg. Sparse inverse covariance estimates for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 45(5):1399–1407, 2007. doi: 10.1109/TGRS.2007.892598.

[30] S. de Vos, S. Patten, E. C. Wit, E. H. Bos, K. J. Wardenaar, and P. de Jonge. Subtyping psychological distress in the population: a semi-parametric network approach. *Epidemiology and Psychiatric Sciences*, 29:e36, 2020. doi: 10.1017/S204579601900026X.

[31] G. Fatima, P. Babu, and P. Stoica. Covariance matrix estimation under positivity constraints with application to portfolio selection. *IEEE Signal Processing Letters*, 29:2487–2491, 2022. doi: 10.1109/lsp.2022.3226117.

[32] Y. Ni, V. Baladandayuthapani, M. Vannucci, and F. C. Stingo. Bayesian graphical models for modern biological applications. *Statistical Methods & Applications*, 31(2):197–225, May 2021. doi: 10.1007/s10260-021-00572-8.

[33] T. Hastie, R. Tibshirani, and M. Wainwright. *Statistical Learning with Sparsity: The Lasso and Generalizations*. Chapman & Hall/CRC, 2015. ISBN 1498712169.

[34] M. Maathuis, M. Drton, S. Lauritzen, and M. Wainwright. *Handbook of Graphical Models*. CRC Press, Inc., USA, 1st edition, 2018. ISBN 1498788629.

[35] M. Wainwright. *Graphical Models and Message-Passing Algorithms: Some Introductory Lectures*, pages 51–108. Springer International Publishing, Cham, 2015. doi: 10.1007/978-3-319-16967-5_3.

[36] P. Clifford. Markov random fields in statistics. In G. Grimmett and D. Welsh, editors, *Disorder in Physical Systems: A Volume in Honour of John M. Hammersley*, pages 19–32. Oxford University Press, Oxford, 1990.

[37] L. Le Cam. The central limit theorem around 1935. *Statistical science*, pages 78–91, 1986.

[38] H. K. Kesavan. *Jaynes' maximum entropy principle*, pages 1779–1782. Springer US, Boston, MA, 2009. doi: 10.1007/978-0-387-74759-0_312.

[39] K. B. Petersen and M. S. Pedersen. The matrix cookbook, nov 2012. URL `http://www2.compute.dtu.dk/pubdb/pubs/3274-full.html`.

[40] D. Bertsimas, J. Lamperski, and J. Pauphilet. Certifiably optimal sparse inverse covariance estimation. *Mathematical Programming*, 184(1–2): 491–530, nov 2020. doi: 10.1007/s10107-019-01419-7.

[41] C. Lam and J. Fan. Sparsistency and rates of convergence in large covariance matrix estimation. *The Annals of Statistics*, 37(6B):4254 – 4278, 2009. doi: 10.1214/09-AOS720.

[42] M. Yuan and Y. Lin. Model selection and estimation in the Gaussian graphical model. *Biometrika*, 94(1):19–35, 03 2007. ISSN 0006-3444. doi: 10.1093/biomet/asm018.

[43] Z. Wu, C. Wang, and W. Liu. A unified precision matrix estimation framework via sparse column-wise inverse operator under weak sparsity. *Annals of the Institute of Statistical Mathematics*, December 2022. doi: 10.1007/s10463-022-00856-0.

[44] J. Fan, Y. Fan, and J. Lv. High dimensional covariance matrix estimation using a factor model. *Journal of Econometrics*, 147(1):186–197, November 2008.

[45] A. Das, A. L Sampson, C. Lainscsek, L. Muller, W. Lin, J. C. Doyle, S. S. Cash, E. Halgren, and T. J. Sejnowski. Interpretation of the precision matrix and its application in estimating sparse brain connectivity during sleep spindles from human electrocorticography recordings. *Neural Computation*, 29(3):603–642, March 2017.

[46] A. Mohammadi and E. C. Wit. Bayesian Structure Learning in Sparse Gaussian Graphical Models. *Bayesian Analysis*, 10(1):109 – 138, 2015. doi: 10.1214/14-BA889.

[47] M. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In D. van Dyk and M. Welling, editors, *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 567–574, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 16–18 Apr 2009. PMLR.

[48] M. Lemercier, C. Salvi, T. Cass, E. V. Bonilla, T. Damoulas, and T. J. Lyons. SigGPDE: Scaling sparse Gaussian processes on sequential data. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 6233–6242. PMLR, 18–24 Jul 2021.

[49] O. Banerjee, L. El Ghaoui, and A. d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data. *The Journal of Machine Learning Research*, 9:485–516, 2008.

[50] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. doi: 10.1017/CBO9780511804441.

[51] D. M. Witten, J. H. Friedman, and N. Simon. New insights and faster computations for the graphical lasso. *Journal of Computational and Graphical Statistics*, 20(4):892–900, 2011. doi: 10.1198/jcgs.2011.11051a.

[52] R. Mazumder and T. Hastie. The graphical lasso: New insights and alternatives. *Electronic Journal of Statistics*, 6:2125 – 2149, 2012. doi: 10.1214/12-EJS740.

[53] A. Berman and R. J. Plemmons. *Nonnegative Matrices in the Mathematical Sciences*. Society for Industrial and Applied Mathematics, 1994. doi: 10.1137/1.9781611971262.

[54] S. Karlin and Y. Rinott. Classes of orderings of measures and related correlation inequalities. I. Multivariate totally positive distributions. *Journal of Multivariate Analysis*, 10(4):467–498, 1980. doi: 10.1016/0047-259X(80)90065-2.

[55] J. A. Soloff, A. Guntuboyina, and M. I. Jordan. Covariance estimation with nonnegative partial correlations, 2020. URL `https://arxiv.org/abs/2007.15252`.

[56] M. Slawski and M. Hein. Estimation of positive definite M-matrices and structure learning for attractive Gaussian Markov random fields. *Linear Algebra and its Applications*, 473:145 – 179, 2015. doi: 10.1016/j.laa.2014.04.020. Special issue on Statistics.

[57] S. Lauritzen, C. Uhler, and P. Zwiernik. Maximum likelihood estimation in Gaussian models under total positivity. *The Annals of Statistics*, 47(4):1835 – 1863, 2019. doi: 10.1214/17-AOS1668.

[58] S. Fallat, S. Lauritzen, K. Sadeghi, C. Uhler, N. Wermuth, and P. Zwiernik. Total positivity in Markov structures. *The Annals of Statistics*, 45(3):1152 – 1184, 2017. doi: 10.1214/16-AOS1478.

[59] S. Kumar, J. Ying, J. V. de M. Cardoso, and D. P. Palomar. A unified framework for structured graph learning via spectral constraints. *Journal of Machine Learning Research*, 21(22):1–60, 2020.

[60] F. R. K. Chung. *Spectral Graph Theory*, volume 92. American Mathematical Society, 1997. doi: 10.1090/cbms/092.

[61] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000. doi: 10.1109/34.868688.

[62] F. Tudisco and M. Hein. A nodal domain theorem and a higher-order Cheeger inequality for the graph p-Laplacian. *Journal of Spectral Theory*, March 2017. doi: 10.4171/JST/216.

[63] L. Li and K.-C. Toh. An inexact interior point method for $\ell_1$-regularized sparse covariance selection. *Mathematical Programming Computation*, 2 (3):291–315, Dec 2010. doi: 10.1007/s12532-010-0020-6.

[64] F. Oztoprak, J. Nocedal, S. Rennie, and P. A. Olsen. Newton-like methods for sparse inverse covariance estimation. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[65] E. Treister and J. S. Turek. A block-coordinate descent approach for large-scale sparse inverse covariance estimation. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[66] C.-J. Hsieh, M. A. Sustik, I. S Dhillon, P. K. Ravikumar, and R. Poldrack. BIG & QUIC: Sparse Inverse Covariance Estimation for a Million Variables. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3165–3173. Curran Associates, Inc., 2013.

[67] S. Oh, O. Dalal, K. Khare, and B. Rajaratnam. Optimization methods for sparse pseudo-likelihood graphical model selection. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[68] P. Koanantakool, A. Ali, A. Azad, A. Buluc, D. Morozov, L. Oliker, K. Yelick, and S.-Y. Oh. Communication-Avoiding Optimization Methods for Distributed Massive-Scale Sparse Inverse Covariance Estimation. In A. Storkey and F. Perez-Cruz, editors, *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 1376–1386, Playa Blanca, Lanzarote, Canary Islands, 09–11 Apr 2018. PMLR.

[69] A. Anandkumar, V. Y. F. Tan, F. Huang, and A. S. Willsky. High-dimensional Gaussian graphical model selection: Walk summability and local separation criterion. *Journal of Machine Learning Research*, 13(76):2293–2337, 2012.

[70] N. Meinshausen and P. Bühlmann. High-dimensional graphs and variable selection with the Lasso. *The Annals of Statistics*, 34(3):1436 – 1462, 2006. doi: 10.1214/009053606000000281.

[71] C. Wang and B. Jiang. An efficient ADMM algorithm for high dimensional precision matrix estimation via penalized quadratic loss. *Computational Statistics & Data Analysis*, 142(C), 2020. doi: 10.1016/j.csda.2019.10681.

[72] W. Liu and X. Luo. Fast and adaptive sparse precision matrix estimation in high dimensions. *Journal of Multivariate Analysis*, 135:153 – 162, 2015. doi: 10.1016/j.jmva.2014.11.005.

[73] T. Cai, W. Liu, and X. Luo. A constrained $\ell_1$ minimization approach to sparse precision matrix estimation. *Journal of the American Statistical Association*, 106(494):594–607, 2011. doi: 10.1198/jasa.2011.tm10155.

[74] T. Tony Cai, W. Liu, and H. H. Zhou. Estimating sparse precision matrix: Optimal rates of convergence and adaptive estimation. *The Annals of Statistics*, 44(2):455 – 488, 2016. doi: 10.1214/13-AOS1171.

[75] H. Pang, H. L., and R. Vanderbei. The FASTCLIME package for linear programming and large-scale precision matrix estimation in R. *Journal of Machine Learning Research*, 15(14):489–493, 2014.

[76] R. Zhang, S. Fattahi, and S. Sojoudi. Large-scale sparse inverse covariance estimation via thresholding and max-det matrix completion. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5766–5775, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[77] Y. Wang, U. Roy, and C. Uhler. Learning high-dimensional Gaussian graphical models under total positivity without adjustment of tuning parameters. In S. Chiappa and R. Calandra, editors, *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August*

*2020, Online [Palermo, Sicily, Italy]*, volume 108 of *Proceedings of Machine Learning Research*, pages 2698–2708. PMLR, 2020.

[78] J. K. Tugnait. Sparse graph learning under Laplacian-related constraints. *IEEE Access*, 9:151067–151079, 2021. doi: 10.1109/ACCESS.2021.3126675.

[79] B. M. Lake and J. B. Tenenbaum. Discovering structure by learning sparse graphs. In *Proceedings of the 33rd Annual Cognitive Science Conference*, 2010.

[80] H. E. Egilmez, E. Pavez, and A. Ortega. Graph learning from data under Laplacian and structural constraints. *IEEE Journal of Selected Topics in Signal Processing*, 11(6):825–841, 2017. doi: 10.1109/JSTSP.2017.2726975.

[81] H. E. Egilmez, E. Pavez, and A. Ortega. Graph learning with Laplacian constraints: Modeling attractive gaussian markov random fields. In *2016 50th Asilomar Conference on Signals, Systems and Computers*, pages 1470–1474, 2016. doi: 10.1109/ACSSC.2016.7869621.

[82] J. Ying, J. V. de Miranda Cardoso, and D. Palomar. Nonconvex sparse graph learning under Laplacian constrained graphical model. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7101–7113. Curran Associates, Inc., 2020.

[83] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst. Laplacian matrix learning for smooth graph signal representation. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3736–3740, 2015. doi: 10.1109/ICASSP.2015.7178669.

[84] V. Kalofolias. How to learn a graph from smooth signals. In A. Gretton and C. C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 920–929, Cadiz, Spain, 09–11 May 2016. PMLR.

[85] V. Kalofolias and N. Perraudin. Large scale graph learning from smooth signals. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[86] A. Agarwal, S. Negahban, and M. J. Wainwright. Fast global convergence rates of gradient methods for high-dimensional statistical recovery. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010.

[87] D.P. Bertsekas. *Nonlinear Programming*. Athena scientific optimization and computation series. Athena Scientific, 1999.

[88] A. d'Aspremont, O. Banerjee, and L. El Ghaoui. First-order methods for sparse covariance selection. *SIAM Journal on Matrix Analysis and Applications*, 30(1):56–66, 2008. doi: 10.1137/060670985.

[89] J. D. Lee, Y. Sun, and M. A. Saunders. Proximal Newton-type methods for convex optimization. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[90] J. D. Lee, Y. Sun, and M. A. Saunders. Proximal Newton-type methods for minimizing composite functions. *SIAM Journal on Optimization*, 24(3): 1420–1443, 2014. doi: 10.1137/130921428.

[91] J. Friedman, T. Hastie, H. Höfling, and R. Tibshirani. Pathwise coordinate optimization. *The Annals of Applied Statistics*, 1(2):302 – 332, 2007. doi: 10.1214/07-AOAS131.

[92] P. Tseng and S. Yun. A coordinate gradient descent method for nonsmooth separable minimization. *Mathematical Programming*, 117(1-2):387–423, August 2009. doi: 10.1007/s10107-007-0170-0.

[93] S. Yun and K.-C. Toh. A coordinate gradient descent method for $\ell_1$-regularized convex minimization. *Computational Optimization and Applications*, 48(2):273–307, May 2009. doi: 10.1007/s10589-009-9251-8.

[94] M. Neuman, V. Jonsson, J. Calatayud, and M. Rosvall. Cross-validation of correlation networks using modular structure. *Applied Network Science*, 7 (1), November 2022. doi: 10.1007/s41109-022-00516-5.

[95] D. Disatnik and S. Katz. Portfolio optimization using a block structure for the covariance matrix. *Journal of Business Finance & Accounting*, March 2012. doi: 10.1111/j.1468-5957.2012.02279.x.

[96] H. Liu, K. Roeder, and L. Wasserman. Stability approach to regularization selection (stars) for high dimensional graphical models. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 2*, NIPS'10, page 1432–1440, Red Hook, NY, USA, 2010. Curran Associates Inc.

[97] L. Wasserman and K. Roeder. High-dimensional variable selection. *The Annals of Statistics*, 37(5A):2178 – 2201, 2009. doi: 10.1214/08-AOS646.

[98] J. Smiljanić, D. Edler, and M. Rosvall. Mapping flows on sparse networks with missing links. *Physical Review E*, 102:012302, Jul 2020. doi: 10.1103/PhysRevE.102.012302.

[99] M. D. Lam, E. E. Rothberg, and M. E. Wolf. The cache performance and optimizations of blocked algorithms. In *Proceedings of the Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS IV, page 63–74, New York, NY, USA, 1991. Association for Computing Machinery. ISBN 0897913809. doi: 10.1145/106972.106981.

[100] Y. Chen, T. A. Davis, W. W. Hager, and S. Rajamanickam. Algorithm 887: CHOLMOD, Supernodal Sparse Cholesky Factorization and Update/Downdate. *ACM Transactions on Mathematical Software*, 35(14), 2008. doi: 10.1145/1391989.1391995.

[101] J. W. H. Liu, E. G. Ng, and B. W. Peyton. On finding supernodes for sparse matrix computations. *SIAM Journal on Matrix Analysis and Applications*, 14(1):242–252, 1993. doi: 10.1137/0614019.

[102] X. S. Li. *Sparse Gaussian Elimination on High Performance Computers*. PhD thesis, EECS Department, University of California, Berkeley, Sep 1996.

[103] T. A. Davis, S. Rajamanickam, and W. M. Sid-Lakhdar. A survey of direct methods for sparse linear systems. *Acta Numerica*, 25:383–566, 2016. doi: 10.1017/S0962492916000076.

[104] A. Anandkumar, V. Y. F. Tan, and A. S. Willsky. High-dimensional graphical model selection: Tractable graph families and necessary conditions. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, page 1863–1871, Red Hook, NY, USA, 2011. Curran Associates Inc.

[105] J. Nocedal and S. Wright. *Numerical Optimization*. Operations Research and Financial Engineering. Springer, 2006. ISBN 978-0-387-30303-1. doi: 10.1007/978-0-387-40065-5.

[106] A. N. Yzelman. High performance sparse computations applied to a parallel conjugate gradient solver. Preprint, 2015.

[107] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, second edition, 2003. doi: 10.1137/1.9780898718003.

[108] J. Ballani and D. Kressner. Sparse inverse covariance estimation with hierarchical matrices. Technical report, EPFL Technical Report, 2014.

[109] H. Dalianis. *Evaluation Metrics and Evaluation*, pages 45–53. Springer International Publishing, Cham, 2018. ISBN 978-3-319-78503-5. doi: 10.1007/978-3-319-78503-5_6.

[110] J. A. Ramey. *datamicroarray: Collection of Data Sets for Classification*, 2016. URL https://github.com/ramhiser/datamicroarray.

[111] E. I. G. Nassara, E. Grall-Maës, and M. Kharouf. Linear discriminant analysis for large-scale data: Application on text and image data. In *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 961–964, 2016. doi: 10.1109/ICMLA.2016.0173.

[112] B. Tu, Z. Zhang, S. Wang, and H. Qian. Making Fisher discriminant analysis scalable. In E. P. Xing and T. Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 964–972, Bejing, China, 22–24 Jun 2014. PMLR.

[113] D. Calvetti and E. Somersalo. *Mathematics of Data Science: A Computational Approach to Clustering and Classification*, chapter 4: Linear discriminant analysis, pages 49–61. Data Science. SIAM, 2020.

[114] J. Fan, Y. Feng, and Y. Wu. Network exploration via the adaptive lasso and scad penalties. *Annals of Applied Statistics*, 3(2):521–541, 06 2009. doi: 10.1214/08-AOAS215.

[115] R. C. Prim. Shortest connection networks and some generalizations. *The Bell System Technical Journal*, 36(6):1389–1401, 1957. doi: 10.1002/j.1538-7305.1957.tb01515.x.

[116] F. D. Gibbons and F. P. Roth. Judging the quality of gene expression-based clustering methods using gene annotation. *Genome research*, 12 (10):1574–1581, oct 2002. doi: 10.1101/gr.397002.

[117] M. R. Berthold and Frank Höppner. On clustering time series using Euclidean distance and Pearson correlation, 2016. URL https://arxiv.org/abs/1601.02213.

[118] P. D'haeseleer. How does gene expression clustering work? *Nature Biotechnology*, 23(12):1499–1501, 2005. doi: 10.1038/nbt1205-1499.

[119] S. Zhou, P. Rütimann, M. Xu, and P. Bühlmann. High-dimensional covariance estimation based on Gaussian graphical models. *Journal of Machine Learning Research*, 12(91):2975–3026, 2011.

[120] E. Dong, H. Du, and L. Gardner. An interactive web-based dashboard to track COVID-19 in real time. *The Lancet. Infectious Diseases*, 20(5):533–534, May 2020. doi: 10.1016/S1473-3099(20)30120-1.

[121] R. J. Sánchez-García, M. Fennelly, S. Norris, N. Wright, G. Niblo, J. Brodzki, and J. W. Bialek. Hierarchical spectral clustering of power grids. *IEEE Transactions on Power Systems*, 29(5):2229–2237, 2014. doi: 10.1109/TPWRS.2014.2306756.

[122] D. Verma and M. Meila. A comparison of spectral clustering algorithms. Technical report, Department of CSE University of Washington Seattle, WA98195-2350, 2005.

[123] P.N. Belhumeur, J.P. Hespanha, and D.J. Kriegman. Eigenfaces vs. fisherfaces: recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, 1997. doi: 10.1109/34.598228.

[124] F.S. Samaria and A.C. Harter. Parameterisation of a stochastic model for human face identification. In *Proceedings of 1994 IEEE Workshop on Applications of Computer Vision*, pages 138–142, 1994. doi: 10.1109/ACV.1994.341300.

[125] J.J. Hull. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554, 1994. doi: 10.1109/34.291440.

[126] T. Clanuwat, M. Bober-Irizar, A. Kitamoto, A. Lamb, K. Yamamoto, and D. Ha. Deep learning for classical japanese literature. *CoRR*, abs/1812.01718, 2018.

[127] H. E. Egilmez, E. Pavez, and A. Ortega. Graph learning from data under Laplacian and structural constraints. *IEEE Journal of Selected Topics in Signal Processing*, 11(6):825–841, 2017. doi: 10.1109/JSTSP.2017.2726975.

[128] L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1601–1608. MIT Press, 2005.

[129] D. Wagner and F. Wagner. Between min cut and graph bisection. In A. M. Borzyszkowski and S. Sokołowski, editors, *Mathematical Foundations of Computer Science 1993*, pages 744–750, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

[130] M. Bollhöfer, O. Schenk, R. Janalik, S. Hamm, and K. Gullapalli. *State-of-the-Art Sparse Direct Solvers*, pages 3–33. Springer International Publishing, Cham, 2020. doi: 10.1007/978-3-030-43736-7_1.

[131] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006. doi: 10.1073/pnas.0601602103.

[132] L. Hagen and A. B. Kahng. Fast spectral methods for ratio cut partitioning and clustering. In *1991 IEEE International Conference on Computer-Aided Design Digest of Technical Papers*, pages 10–13, Nov 1991. doi: 10.1109/ICCAD.1991.185177.

[133] L. Hagen and A. B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(9):1074–1085, Sep. 1992. doi: 10.1109/43.159993.

[134] B. Bollobás. *Graphs, Groups and Matrices*, pages 253–293. Springer New York, New York, NY, 1998. doi: 10.1007/978-1-4612-0619-4_8.

[135] H. Gajewski and K. Gärtner. Domain separation by means of sign changing eigenfunctions of p-Laplacians. *Applicable Analysis*, 79(3-4):483–501, 2001. doi: 10.1080/00036810108840974.

[136] S. Amghibech. Eigenvalues of the discrete p-Laplacian for graphs. *Ars Combinatoria*, 67, 2003.

[137] A. Szlam and X. Bresson. Total variation and Cheeger cuts. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 1039–1046, Madison, WI, USA, 2010. Omnipress.

[138] X. Bresson, T. Laurent, D. Uminsky, and J. Brecht. Convergence and energy landscape for Cheeger cut clustering. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[139] R. Bhatia. *Matrix Analysis*, volume 169. Springer, 1997.

[140] D. Luo, H. Huang, C. Ding, and F. Nie. On the eigenvectors of p-Laplacian. *Machine Learning*, 81(1):37–51, 2010. doi: 10.1007/s10994-010-5201-z.

[141] Y. Koren. Drawing graphs by eigenvectors: theory and practice. *Computers & Mathematics with Applications*, 49(11):1867–1888, 2005. doi: 10.1016/j.camwa.2004.08.015.

[142] R. Diekmann and R. Preis. AG-Monien Graph Collection. `http://www2.cs.uni-paderborn.de/fachbereich/AG/monien/RESEARCH/PART/graphs.html`, January 2018.

[143] D. Spielman. *Spectral graph theory*, chapter 5, pages 129 – 162. Chapman & Hall/CRC, 1st edition, 2012. ISBN 1439827354. doi: 10.5555/2141107.

[144] H. Jia, S. Ding, and M. Du. Self-tuning p-spectral clustering based on shared nearest neighbors. *Cognitive Computation*, 7(5):622–632, Oct 2015. doi: 10.1007/s12559-015-9331-2.

[145] P. Upadhyaya, E. Jarlebring, and F. Tudisco. The self-consistent field iteration for p-spectral clustering, 2021. URL `https://arxiv.org/abs/2111.09750`.

[146] S. S. Rangapuram, P. K. Mudrakarta, and M. Hein. Tight continuous relaxation of the balanced k-cut problem. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3131–3139, Cambridge, MA, USA, 2014. MIT Press.

[147] X. Bresson, X.-C. Tai, T. F. Chan, and A. Szlam. Multi-class transductive learning based on $\ell_1$ relaxations of Cheeger cut and Mumford-Shah-Potts model. *Journal of Mathematical Imaging and Vision*, 49(1):191–201, May 2014. ISSN 0924-9907. doi: 10.1007/s10851-013-0452-5.

[148] A Cristofari, F Rinaldi, and F Tudisco. Total variation based community detection using a nonlinear optimization approach. *SIAM Journal on Applied Mathematics*, 80(3):1392–1419, 2020. doi: 10.1137/19M1270446.

[149] K. Fountoulakis, D. Wang, and S. Yang. p-norm flow diffusion for local graph clustering. In H. Daumé III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3222–3232. PMLR, 13–18 Jul 2020.

[150] M. Liu and D. F. Gleich. Strongly local p-norm-cut algorithms for semi-supervised learning and local graph clustering. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 5023–5035. Curran Associates, Inc., 2020.

[151] P. Li, N. He, and O. Milenkovic. Quadratic decomposable submodular function minimization: Theory and practice. *Journal of Machine Learning Research*, 21(106):1–49, 2020.

[152] S. Saito and M. Herbster. Generalizing p-Laplacian: spectral hypergraph theory and a partitioning algorithm. *Machine Learning*, November 2022. doi: 10.1007/s10994-022-06264-y.

[153] P. Mercado, F. Tudisco, and M. Hein. Spectral clustering of signed graphs via matrix power means. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4526–4536. PMLR, 09–15 Jun 2019.

[154] D. A. Bader, H. Meyerhenke, P. Sanders, C. Schulz, A. Kappes, and D. Wagner. *Benchmarking for Graph Clustering and Partitioning*, pages 73–82. Springer New York, New York, NY, 2014. doi: 10.1007/978-1-4614-6170-8_23.

[155] H. D. Simon and S.-H. Teng. How good is recursive bisection? *SIAM Journal on Scientific Computing*, 18(5):1436–1445, September 1997. doi: 10.1137/S1064827593255135.

[156] S. X. Yu and J. Shi. Multiclass spectral clustering. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 313–319 vol.1, 2003. doi: 10.1109/ICCV.2003.1238361.

[157] Q. Wang, J. Gao, and H. Li. Grassmannian manifold optimization assisted sparse spectral clustering. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3145–3153, July 2017. doi: 10.1109/CVPR.2017.335.

[158] D. Dua and C. Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

[159] H. Sato and T. Iwai. Optimization algorithms on the Grassmann manifold with application to matrix eigenvalue problems. *Japan Journal of Industrial and Applied Mathematics*, 31(2):355–400, April 2014. doi: 10.1007/s13160-014-0141-9.

[160] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, USA, 2007.

[161] R. H. Byrd, G. Liu, and J. Nocedal. On the local behavior of an interior point method for nonlinear programming. In *Numerical Analysis 1997*, pages 37–56. Addison Wesley Longman, 1998.

[162] A. Antoniou and L. Wu-Sheng. *Practical Optimisation: Algorithms and Engineering Applications*. ISTE. Springer Science + Business Media, LLC, New York, USA, 2017.

[163] P. A. Absil, R. Sepulchre, P. Van Dooren, and R. Mahony. Cubically convergent iterations for invariant subspace computation. *SIAM Journal on Matrix Analysis and Applications*, 26(1):70–96, 2004. doi: 10.1137/S0895479803422002.

[164] I. S. Duff, J. K. Reid, and J. A. Scott. The use of profile reduction algorithms with a frontal code. *International Journal for Numerical Methods in Engineering*, 28(11):2555–2568, 1989. doi: https://doi.org/10.1002/nme.1620281106.

[165] N. Boumal, B. Mishra, P.-A. Absil, and R. Sepulchre. Manopt, a Matlab toolbox for optimization on manifolds. *Journal of Machine Learning Research*, 15(1):1455–1459, January 2014.

[166] W. Huang, P.-A. Absil, K. A. Gallivan, and P. Hand. Roptlib: An object-oriented C++ library for optimization on Riemannian manifolds. *ACM Transactions on Mathematical Software*, 44(4), July 2018. doi: 10.1145/3218822.

[167] I. S. Dhillon, Y. Guan, and B. Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(11):1944–1957, November 2007. ISSN 0162-8828. doi: 10.1109/TPAMI.2007.1115.

[168] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49(2):291–307, 1970. doi: 10.1002/j.1538-7305.1970.tb01770.x.

[169] D. B. Graham and N. M. Allinson. *Characterising Virtual Eigensignatures for General Purpose Face Recognition*, pages 446–456. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998. doi: 10.1007/978-3-642-72201-1_25.

[170] A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E*, 78:046110, Oct 2008. doi: 10.1103/PhysRevE.78.046110.

[171] S. Sieranoja and P. Fränti. Fast and general density peaks clustering. *Pattern Recognition Letters*, 128:551 – 558, 2019. doi: 10.1016/j.patrec.2019.10.019.

[172] D. Hond and L. Spacek. Distinctive descriptions for face processing. In *Proceedings of the 8th British Machine Vision Conference BMVC97, Colchester, England*, pages 320–329, September 1997.

[173] N. C. Ebner, M. Riediger, and U. Lindenberger. FACES–a database of facial expressions in young, middle-aged, and older women and men: development and validation. *Behavior Research Methods*, 42(1):351–362, Feb 2010. doi: 10.3758/BRM.42.1.351.

[174] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266): 1332–1338, 2015. doi: 10.1126/science.aab3050.

[175] X. Zhu, Y. Zhu, and W. Zheng. Spectral rotation for deep one-step clustering. *Pattern Recognition*, 105:107175, 2020. doi: 10.1016/j.patcog.2019.107175.