THE UNIVERSITY OF CHICAGO

MULTISCALE METHODS FOR SEMIDEFINITE PROGRAMMING

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCES
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

DEPARTMENT OF STATISTICS

BY

YI WANG

CHICAGO, ILLINOIS
DECEMBER 2024

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

During my PhD studies, I received tremendous support from many people I've met at UChicago. Without them the completion of this thesis would not have been possible. First and foremost, I am deeply grateful to my thesis advisor, Professor Yuehaw Khoo, for all his invaluable advice and insights throughout my PhD journey. I began working with him in the middle of my third year, initially lacking substantial training in applied mathematics and facing a rather tight timeline. I am thankful that Yuehaw accepted me as his student and was always available to provide the essential resources I needed for my project and to guide me through the challenges I faced. Without his assistance, this thesis would not have been possible.

I would also like to thank my thesis committee members, Professor Rina Foygel Barber and Professor Jeremy G. Hoskins, for their invaluable support during my PhD studies. I am grateful for their numerous and insightful recommendations in the preparation of this thesis, as well as for their guidance during my proposal and defense. In addition, I admire them both as outstanding scholars, and their warmth of character has always reassured me that I had more than one person to rely on throughout my studies.

I would like to extend my gratitude to all the faculty members and administrative staff in the department for their support and assistance during this process. Among them, Professor Lek-Heng Lim deserves special mention. Professor Lim is a great teacher. His course on matrix computation provided me with the solid knowledge I needed for my project. Mastering his class gave me the much-needed confidence to tackle the matrix-related projects in my thesis, especially after experiencing several setbacks in research that were particularly challenging. Additionally, I would like to thank John Zekos for his invaluable assistance. His knowledge of computer systems is astonishing, and whenever I encountered problems with running code on the server or software installation, I always knew I could rely on John for help.

I would like to thank all my friends and peers I met at UChicago, particularly in the Statistics and CAM departments, for their support, numerous thoughtful discussions, and many many many enjoyable conversations: Lijia Zhou, Zhen Dai, Yanqing Gui, Hongli Zhao, Yi Wei, Qi Zhu, Haochen Wang, Dongyue Xie, Wanrong Zhu, Yunqi Yang, Zehao Niu, Hai Tran Bach, Chih-hsuan Wu, Yuguan Wang, Yian Chen, Yifan Peng, Huanlin Zhou, Daniel Xiang, Soumyabrata Kundu and many many others. I am also thankful for all the support I got from my high school friends: Shiliang Zuo, Yang Guo and Jialin Ouyang.

Finally, I am grateful to have met Rui Wang, who has been my girlfriend for over six years now, at UChicago. Rui has been a constant source of love and support, and she never fails to surprise me with her spontaneity, her talent in many many areas, and her persistence when facing difficult situations. Without her support and optimism, I wouldn't have been able to persist through all the challenges in my PhD studies. I also wish to express my deepest gratitude to my parents, Chunmei Cui and Jingwen Wang, as well as all of my grandparents. I thank them for their unconditional love and support and for all their advice and guidance, which have shaped me into who I am today.

ABSTRACT

In this thesis, we examine two problems of convex relaxation. The first problem involves the recovery of the ground-state energy in the Ferromagnetic Ising model with localized corruption patterns in the interaction matrix. The second problem focuses on the recovery of the ground-state energy for the quantum many-body problem. For both cases, we propose relaxations that result in semidefinite programming (SDP) problems, which can be solved in polynomial time. Additionally, we explore the inherent properties of the matrix variables from the SDP problems, and impose further structures on them to reduce the computation time for the problems. Furthermore, we conducted numerical experiments in various scenarios to compare the objective values from our relaxed problems to the true ground-state energy, to assess the exactness of our relaxations.

CHAPTER 1

INTRODUCTION

Determining the lowest energy state of a many-body system is one of the most fundamental problems in science and engineering. This problem arises in various fields, including the study of the Ising model [18], graphical modeling [58], sensor network localization [39], and the structure from motion problem [48], among others. In this thesis, we investigate two model problems in this category: the spin problem in the classical and quantum settings. We propose novel relaxations for the original problems, which are generally NP-hard to solve, and develop accelerated algorithms to solve these relaxed problems by exploiting their inherent structures.

For the spin problem in the classical setting, we leverage the fact that the ground-state configuration yields a low-rank moment matrix. We begin by reformulating the original ground-state energy problem as a standard SDP problem with an additional rank constraint. We then propose a relaxation based on this rank constraint, incorporating aspects of the embedding method, which combines an expensive local constraint with a relatively cheap global positive semidefinite (PSD) constraint. The local constraint is designed using techniques from the sum-of-squares relaxation. To solve the resulting relaxed problem, we employ an ADMM-type (Alternating Direction Method of Multipliers) method. By exploiting the low-rank structure of the primal PSD variable, we further reduce the computation time for the most computationally intensive step in the ADMM algorithm. We demonstrate the effectiveness of this modified ADMM algorithm by testing it on systems with as many as 10000 variables. We also examine the tightness of our relaxation by applying the algorithm to a 2-D lattice system. Furthermore, we address the issue of multiple solutions arising from our relaxation and demonstrate that this issue exists in the "corrupted pairs" model.

For the spin problem in the quantum setting, however, exploiting the low-rank structure alone is not sufficient to achieve a significant reduction in computation time. The natural

correlations present in the PSD matrix of the dual problem dictate that the matrix's rank increases linearly with the matrix size. Therefore, we develop a hierarchical matrix representation, another type of data-sparse representation, for the PSD matrix of the dual problem. We begin by reformulating the original ground-state energy problem in the quantum setting as a linear problem over the "moments" of the density matrix. Inspired by the 2-RDM (2-electron reduced density matrix) method, we replace the constraint that these "moments" must originate from a true density matrix with several necessary conditions, resulting in an SDP problem. By utilizing the hierarchical structure in the PSD variable for the dual problem, we successfully accelerate the most computationally expensive step of the ALM (Augmented Lagrangian Method), reducing the per-iteration time complexity from cubic to quadratic. Furthermore, if the primal PSD variable is also assumed to possess a hierarchical structure, the per-iteration time complexity can be further reduced to approximately linear. We test our algorithm on the 1-D transverse field Ising model, demonstrating its effectiveness for systems of sizes up to approximately 4000, with a relatively accurate recovery of the ground-state energy.

CHAPTER 2

A CONVEX EMBEDDING METHOD FOR CLASSICAL ISING MODEL

In this chapter, we introduce a novel method for determining the ground-state of the Ising model on a lattice, particularly in cases where some local regions are corrupted. The Ising model, originally proposed by Wilhelm Lenz and Ernst Ising to study phase transitions, has since been widely used in various areas of science. One of the most fundamental questions in the study of the Ising model is finding its ground-state, which corresponds to the configuration of particles in the system with the least energy. Mathematically, determining the ground-state on an Ising lattice of size n involves solving the following optimization problem:

$$\underset{x\in\{-1,+1\}^n}{\text{minimize}} \; H(x) := \sum_{i\sim j} J_{ij}x_i x_j. \tag{2.1}$$

Here, the vertices of the lattice are indexed by $\{1, 2, \cdots, n\}$, where $\{J_{ij}\}_{ij}$ represents known data. The notation $i \sim j$ denotes the adjacency relationship between site i and site j on the lattice, and H is the Hamiltonian function that describes the energy for any configuration $x \in \{-1, +1\}^n$. The ground-state $x_{\text{gs}} \in \{-1, +1\}^n$ is defined to be the solution to the above optimization problem, and the ground-state energy is defined to be $H(x_{\text{gs}})$. Without loss of generality, we can set $J_{ij} = 0$ for any pair (i, j) not adjacent on the lattice, and require that $J_{ij} = J_{ji}$ for all sites i and j. Thus, we can identify the optimization problem in (2.1) by a known symmetric matrix $J \in \mathbb{R}^{n\times n}$, which we also refer to as the interaction matrix.

The ground-state problem described by equation (2.1) is generally NP-hard without additional assumptions on the structure of the lattice or the interaction matrix J [4]. There are only a few known solutions for particular special cases documented in the literature [20, 21, 22, 57, 29, 23].

Recently, embedding theories [56] have emerged and have been successfully applied in

various areas of quantum physics. These theories combine high-fidelity yet computationally expensive models for local regions of interest with low-fidelity and cost-effective models that approximate the interactions between these regions of interest and their environment.

In this work, we leverage this intuition to solve (2.1) for a specific type of interaction matrix J. We assume J consists of two parts: $J = J_{\text{clean}} + J_{\text{noise}}$, where J_{clean} is -1 for all edges on the lattice, leading to a ground-state of all $+1$ or -1, and J_{noise} is zero almost everywhere except for on some "local regions" of the lattice where it introduces corruption patterns.

2.1 Notations

We use $\mathbb{S}^n$ to denote the set of real symmetric matrices of size $n \times n$, and $\mathbb{S}^n_+$ to denote the set of PSD matrices in $\mathbb{S}^n$. For any matrix X in $\mathbb{S}^n$, we may also use $X \succeq 0$ to indicate that X is PSD. When discussing a matrix $X \in \mathbb{R}^{n\times n}$ and two sets $I_1, I_2 \subseteq \{1, \cdots, n\}$, we use $X(I_1, I_2) \in \mathbb{R}^{|I_1|\times|I_2|}$ to denote the sub-matrix in X whose row numbers are in I_1 and column numbers are in I_2. We use $\text{vec}(X)$ to denote the vectorization of a matrix X. We use $\text{diag}(X)$ for a square matrix X to denote a vector of its diagonal terms, and we also use $\text{diag}(y)$ for a vector y to denote a diagonal matrix whose main diagonal is y. For any linear operator $\mathcal{A}$ on matrices or vectors, we denote its adjoint by $\mathcal{A}^*$. At last, we use $\Pi_{\mathcal{P}}$ to denote the projection operator onto the space $\mathcal{P}$.

2.2 Problem formulation

Firstly, the optimization problem described by equation (2.1) can be equivalently reformulated as follows:

$$\begin{aligned} &\underset{X\in\mathbb{S}_+^n}{\text{minimize}} && \langle J, X\rangle, \\ &\text{subject to} && \text{diag}(X) = 1_n, \quad \text{rank}(X) = 1. \end{aligned} \tag{2.2}$$

Here, 1_n denotes a vector of ones with dimension equal to $n \times 1$. Note that (2.2) is an equivalent formulation of (2.1), as any $X \in \mathbb{S}_+^n$ of rank one can be written as $X = xx^T$ for some vector $x \in \mathbb{R}^n$. Moreover, the constraint $\text{diag}(X) = 1$ implies that x must be a vector of -1 and 1. The alternative formulation presented in equation (2.2) is equally challenging to solve as the original problem (2.1), and the main difficulty arises from the rank constraint imposed on the PSD variable X. We will conduct experiments applying the embedding theories to (2.2), by replacing the rank constraint with a set of necessary constraints that is computational tractable, and solving a relaxed optimization problem of the following formulation:

$$\underset{X\in\mathbb{S}_+^n}{\text{minimize}} \quad \langle J, X\rangle, \tag{2.3}$$

$$\text{subject to} \quad \text{diag}(X) = 1_n, \tag{2.4}$$

$$X(I_k, I_k) \ \text{ satisfies some constraints for } k = 1, \cdots K. \tag{2.5}$$

Here, each $I_k \subseteq \{1, \cdots, n\}$, and $X(I_k, I_k)$ denotes the block matrix within X whose row and column numbers are in I_k, for $k = 1, \cdots, K$. In addition to the global PSD constraint and linear constraint on X in (2.4), we also apply constraints on local regions of X in (2.5). These local constraints are necessary constraints implied by the rank constraint in (2.2), and thus the problem described in (2.3)-(2.5) is a relaxed version of (2.2), and will always

provide a lower bound of the true objective value. The selection of the PSD constraint and linear constraint is based on the observation that in many problem scenarios, these constraints provide reasonable approximations to the true solutions. Additionally, there exist well-tested and highly efficient solvers specifically designed for addressing problems with PSD and linear constraints on matrices.

Consider a lattice on an undirected graph $G = (V, E)$, where $V = \{1, 2, \cdots, n\}$ represents the set of vertices and $E \subseteq \{\{i, j\} : i, j \in V,\ i \neq j\}$ represents the set of edges of the graph. Each vertex $i \in V$ has an associated binary variable $x_i \in \{+1, -1\}$. Additionally, for each edge $\{i, j\} \in E$ (which was previously denoted by $i \sim j$), there is an associated interaction term $J_{ij} = J_{ji} \in \mathbb{R}$. We form a symmetric matrix $J \in \mathbb{R}^{n\times n}$ from the set $\{J_{ij}\}_{ij}$, and our goal is to solve the optimization problem in (2.2) and find the corresponding objective value $\langle J, X\rangle$ of the solution.

Example 1. *Consider the Ferromagnetic Ising model on a lattice $G = (V, E)$ with $|V| = n$. The Hamiltonian function is given by*

$$H(x) := -\sum_{\{i,j\}\in E} x_i x_j. \tag{2.6}$$

The interaction matrix $J_F \in \mathbb{R}^{n\times n}$ corresponds to this Hamiltonian on the graph G is given by

$$J_F(i, j) = \begin{cases} -1, & \{i, j\} \in E, \\ 0, & \{i, j\} \notin E. \end{cases} \tag{2.7}$$

Assuming the graph G is connected, the ground-state is $\pm 1_n$.

In this work, we consider the Ferromagnetic Ising model with localized corruption. To define this model, we first let $N_1, N_2, \cdots, N_m$ be a collection of disjoint subsets of the vertex

set V. Then, define:

$$J_{\text{noise}}(i,j) = \begin{cases} a_{ij}, & i,j \in N_k \text{ for some } k = 1, \cdots, m, \\ 0, & \text{otherwise}, \end{cases} \tag{2.8}$$

where $a_{ij} \in \mathbb{R}$ is generated from some noise distribution. Without loss of generality we can assume $a_{ij} = a_{ji}$ for all i and j. Let $J := J_{\text{F}} + J_{\text{noise}}$. In other words, the interaction matrix J_{F} for the Ferromagnetic Ising model is randomly corrupted in a few subsets (also referred to as clusters) of V. We assume that the size of the corrupted clusters and the number of corrupted local clusters are constants and do not scale with the size of the lattice n. Additionally, each vertex is connected to a constant number of vertices, leading to a total number of edges that grows linearly with n.

As an example, a 2-D Ferromagnetic Ising model with localized corruption is shown in Figure 2.1. In the 2-D lattice, two vertices are connected by a solid line if their interaction is -1, by a dashed line if their interaction is sampled from a random distribution, and are not connected if their interaction is zero.

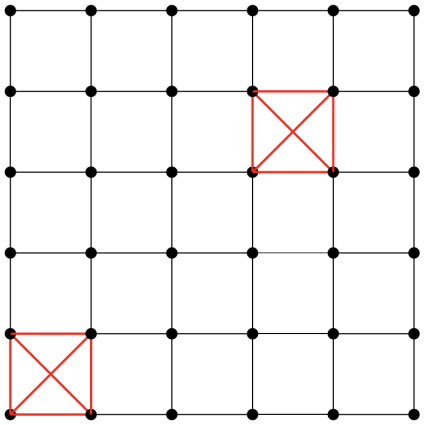

Figure 2.1: An example of the 2-D Ferromagnetic Ising model with localized corruption. Corrupted regions are 2×2 squares on the lattice.

2.3 Convex relaxation for the Ferromagnetic Ising problem with localized corruption

First, note that the problem described in (2.2) is difficult to solve due to the rank constraint imposed on the PSD matrix X. To address this issue, one could drop the rank constraint and solve a relaxed version of problem (2.2), which has the following formulation:

$$\underset{X \in \mathbb{S}_+^n}{\text{minimize}} \quad \langle J, X \rangle \quad \text{subject to} \quad \text{diag}(X) = 1_n. \tag{2.9}$$

This type of relaxation is known as the Max-Cut SDP relaxation [25].

The Max-Cut SDP relaxation described by equation (2.9) for a moderately large problem size n can be readily solved using the interior point method [44]. This approach offers a relatively accurate approximation to the solution of the problem defined in (2.2) [25]. However, the Max-Cut SDP relaxation is not tight enough, and almost always returns non-integer solutions. To address this issue, we propose a tighter relaxation for our model problem by coupling the global PSD and linear constraint on the matrix X with local, more expensive constraints on a collection of regions of interest in X.

We use $\{I_k\}_{k=1}^K$ to denote the indices of vertices where we want to impose the more expensive constraints, with $|I_k| = n_k$ and $I_k \subseteq \{1, \cdots, n\}$ for $k = 1, \ldots, K$. One way to choose these indices is to set $K = m$, where m is the number of corrupted regions, and $I_k = N_k$ for $k = 1, \cdots, K$. In other words, we only apply the more expensive constraints on the corrupted regions of the interaction matrix J. There are other ways to choose these indices as well. For instance, we can let I_k be the union of a corrupted region N_k and its one-hop or two-hop neighbors on the graph G, for $k = 1, \cdots, K$. One an undirected graph $G = (V, E)$ where the vertex set E is identified with $\{1, \cdots, n\}$ for some positive integer n,

the one-hop and two-hop neighbors of a set $S \subseteq \{1, \cdots, n\}$ are defined by

$$\begin{aligned} &\text{one-hop}(S) := \{t | \{t, s\} \in E \text{ for some } s \in S\}, \\ &\text{two-hop}(S) := \{t | \{t, s\} \in E \text{ for some } s \in \text{one-hop}(S)\}. \end{aligned} \tag{2.10}$$

After we have fixed a collection of sets $\{I_k\}_{k=1}^K$ based on the structure of J, our proposal is to solve the following optimization problem as a tighter relaxation for the optimization problem (2.2):

$$\begin{aligned} \underset{X \in \mathbb{S}_+^n}{\text{minimize}} \quad & \langle J, X \rangle, \\ \text{subject to} \quad & \text{diag}(X) = 1_n, \\ & X(I_k, I_k) = \mathcal{A}_k(M_k) \text{ for some } M_k \in \mathcal{C}_k, \quad k = 1, \cdots, K, \end{aligned} \tag{2.11}$$

for some sets $\{\mathcal{C}_k\}_k$ and functions $\{\mathcal{A}_k\}_k$ to be specified below. In other words, in addition to the constraints inherited from the Max-Cut SDP relaxation (2.9), we impose extra local constraints on blocks of X. These local constraints are necessary conditions implied by the rank constraint in (2.2), ensuring that the resulting relaxed problem provides a valid lower bound for the original problem. We will discuss two types of local constraints implied by the rank constraint in this work.

The first type of local constraint, which we refer to as the exact local constraint in the following sections, requires that each block $X(I_k, I_k)$ for $k = 1, \cdots, K$ must come from the product of a vector of -1 and 1 with its transpose, i.e.,

$$\mathcal{C}_k := \{vv^T, \ v \in \{-1, 1\}^{n_k}\}, \tag{2.12}$$

and $\mathcal{A}_k : \mathbb{S}^{n_k} \to \mathbb{S}^{n_k}$ is the identity operator for $1 \leq k \leq K$. Locally on $X(I_k, I_k)$ for $k = 1, \cdots, K$, the exact local constraint is equivalent to the rank constraint. However, the

optimization problem (2.11) with the exact local constraint constraint is non-convex. To address this issue, we also propose a second type of local constraint.

The second type of local constraint, which we refer to as the inexact local constraint in the following sections, is constructed using techniques from sum-of-squares optimization [35]. Suppose that X is a feasible solution for problem (2.2). Because of the constraint imposed on X, for every $k = 1, \cdots, K$, $X(I_k, I_k)$ is also a rank-one matrix whose diagonal elements are all ones. Thus, we must have $X(I_k, I_k) = x_k x_k^T$ for some $x_k \in \{-1, +1\}^{n_k}$. Consider the matrix M_k defined by:

$$M_k := y_k y_k^T \in \mathbb{S}^{1+n_k^2}, \quad y_k := (1; \mathrm{vec}(X(I_k, I_k))) \in \mathbb{R}^{n_k^2+1}. \tag{2.13}$$

One can observe that the matrix M_k is symmetric and PSD. Besides, the identities $x_k^2(i) = 1$ for all $1 \leq i \leq n_k$ give rise to several equality constraints on the entries of M_k, which can be described by a linear equality:

$$\mathcal{G}_k(M_k) = g_k, \quad \mathcal{G}_k : \mathbb{S}_+^{n_k^2+1} \to \mathbb{R}^{p_k}. \tag{2.14}$$

The constant vector g_k and the dimension p_k depend on n_k and will be specified later. The above describes a closed convex set $\mathcal{C}_k \subseteq \mathbb{S}_+^{n_k^2+1}$:

$$\mathcal{C}_k := \{M_k \in \mathbb{S}_+^{n_k^2+1} \ : \ \mathcal{G}_k(M_k) = g_k\}. \tag{2.15}$$

Additionally, from the construction of M_k in (2.13), we can see that $X(I_k, I_k)$ can be extracted from M_k, by reshaping the last n_k^2 entries of its first column into a square matrix, and the equality

$$\mathcal{A}_k(M_k) = X(I_k, I_k), \quad \mathcal{A}_k : \mathbb{S}_+^{n_k^2+1} \to \mathbb{R}^{n_k \times n_k} \tag{2.16}$$

uniquely defines the linear operator $\mathcal{A}_k$ for $k = 1, \cdots, K$.

Example 2. *Let X be any feasible PSD matrix for (2.2). Consider a simple example with $|I_k| = 3$. From the discussion above, we know $X(I_k, I_k) = x_k x_k^T$ for a vector $x_k = (x_k(1), x_k(2), x_k(3)) \in \{-1, +1\}^3$. To simplify notation, we will drop the dependence of x_k on k, and rewrite x_k as $(x_1, x_2, x_3) \in \{-1, +1\}^3$. In this case, we have*

$$y_k = (1; \mathrm{vec}(X(I_k, I_k))) = \left(1,\ x_1^2,\ x_1x_2,\ x_1x_3,\ x_1x_2,\ x_2^2,\ x_2x_3,\ x_1x_3,\ x_2x_3,\ x_3^2\right)^T \in \mathbb{R}^{10},$$

and

$$M_k = y_k y_k^T = \begin{pmatrix} 1 & x_1^2 & x_1x_2 & \cdots & x_3^2 \\ x_1^2 & x_1^4 & x_1^3x_2 & \cdots & x_1^2x_3^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_2x_3 & x_1^2x_2x_3 & x_1x_2^2x_3 & \cdots & x_2x_3^3 \\ x_3^2 & x_1^2x_3^2 & x_1x_2x_3^2 & \cdots & x_3^4 \end{pmatrix} \in \mathbb{S}_+^{10}.$$

Since $x_i \in \{-1, 1\}$ for $i = 1, 2, 3$, x_i^2 is always one, and we can simplify M_k as

$$M_k = \begin{pmatrix} 1 & 1 & x_1x_2 & \cdots & 1 \\ 1 & 1 & x_1x_2 & \cdots & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_2x_3 & x_2x_3 & x_1x_3 & \cdots & x_2x_3 \\ 1 & 1 & x_1x_2 & \cdots & 1 \end{pmatrix}.$$

From the above detailed description of the matrix M_k, we see that all entries of M_k can be partitioned into four groups, each of which shares the same value in $\{1, x_1x_2, x_1x_3, x_2x_3\}$.

Similarly, for $n_k = 3$, all entries in M_k take values from one of the two sets: $\{1\}$ and $\{x_1x_2, x_1x_3, x_2x_3\}$, and for a general $n_k \geq 4$, all entries in M_k take values from one of the three sets: $\{1\}, \{x_ix_j\}_{1 \leq i < j \leq n_k}$ and $\{x_ix_jx_kx_l\}_{1 \leq i < j < k < l \leq n_k}$. These sets collectively

include $1 + \binom{n_k}{2} + \binom{n_k}{4}$ many unique values. Given that M_k has $(n_k^2+1)^2$ many entries, it follows that there are $(n_k^2+1)^2 - \binom{n_k}{2} - \binom{n_k}{4}$ equality constraints on the entries of M_k, which are summarized by the linear equality constraint in (2.14). Consequently, the dimension p_k is given by $(n_k^2+1)^2 - \binom{n_k}{2} - \binom{n_k}{4}$ and $g_k = (1, 0, \cdots, 0)^T \in \mathbb{R}^{p_k}$.

It is straightforward to verify that for any feasible X in the original problem (2.2) with the rank constraint, the local blocks of X also satisfy the inexact local constraint. Therefore, (2.11) with the inexact local constraint is indeed a convex relaxation of the original problem. In the following sections in Chapter 2, we will primarily focus on solving (2.11), with either the exact or the inexact local constraint. We will use the term "the relaxed problem with the exact/inexact local constraint" to refer to (2.11) with the exact/inexact local constraint.

2.4 The alternating direction method of multipliers algorithm for the problem with embedding

In the following section, we will discuss how to solve the relaxed problem with the exact/inexact local constraint. If the problem's scale is relatively small and the local constraints defined by the sets $\{\mathcal{C}_k\}_k$ and functions $\{\mathcal{A}_k\}_k$ consist of PSD and linear constraints, the relaxed problem is convex and can be solved using the interior point method available in many software packages. However, if the local constraint is non-convex or the problem size is too large to manage directly, we need to solve either the primal or the dual problem using the alternating direction method of multipliers (ADMM). In case of the primal problem, we first form the augmented Lagrangian function $L_\sigma(X, \{M_k\}; y, \{v_k\})$ for primal variables $X \in \mathbb{S}^n_+$ and $M_k \in \mathcal{C}_k$ for $k = 1, \cdots, K$, dual variables $y \in \mathbb{R}^n$ and $v_k \in \mathbb{S}^{n_k}$ for $k = 1, \cdots, K$, and

penalty parameter $\sigma > 0$:

$$L_\sigma(X, \{M_k\}_k; y, \{v_k\}_k) = \begin{cases} \langle J, X \rangle + \frac{\sigma}{2} \sum_{k=1}^K \|X(I_k, I_k) - \mathcal{A}_k(M_k) + \frac{1}{\sigma} v_k\|_F^2 \\ -\frac{1}{2\sigma} \sum_{k=1}^K \|v_k\|_F^2 + \frac{\sigma}{2} \|\mathrm{diag}(X) - 1_n + \frac{1}{\sigma} y\|_F^2 - \frac{1}{2\sigma} \|y\|_F^2. \end{cases} \tag{2.17}$$

Then, we iterate Algorithm 1 until convergence.

Algorithm 1 ADMM for the primal problem

Require: $X, y, \{v_k\}_k$, $\{M_k\}_k$, and penalty parameter $\sigma > 0$

1: **while** not converged **do**
2: $X \leftarrow \mathrm{argmin}_{X \succeq 0}\ L_\sigma(X, \{M_k\}_k; y, \{v_k\}_k)$
3: **for** $k = 1, \cdots, K$ **do**
4: $M_k \leftarrow \mathrm{argmin}_{M_k \in \mathcal{C}_k} \|X(I_k, I_k) - \mathcal{A}_k(M_k) + \frac{1}{\sigma} v_k\|_F^2$
5: $v_k \leftarrow v_k + \sigma(X(I_k, I_k) - \mathcal{A}(M_k))$
6: **end for**
7: $y \leftarrow y + \sigma(\mathrm{diag}(X) - 1_n)$
8: **end while**

2.4.1 Solving the local problem

In Algorithm 1, we minimize the augmented Lagrangian function $L_\sigma(X, \{M_k\}_k; y, \{v_k\}_k)$ with respect to X while keeping $\{M_k\}_k$ fixed, and then minimize the augmented Lagrangian function with respect to each M_k for $k = 1, \cdots, K$ while keeping X fixed. For the M-subproblem in line 4 of Algorithm 1, we need to solve the following problem with two types of local constraints:

$$M_k = \mathrm{argmin}_{M_k \in \mathcal{C}_k} \|\mathcal{A}_k(M_k) - b_k\|_F^2, \quad k = 1, \cdots, K.$$

Here, the constant vector b_k is given by

$$b_k := X(I_k, I_k) + \frac{1}{\sigma} v_k, \quad k = 1, \cdots, K.$$

If we use the exact local constraint, we can afford to perform a brute force search among all feasible solutions, provided that the sizes of the local clusters $\{n_k\}_{k=1}^N$ are small constants that do not grow with the size of the matrix, n.

If instead, we use the inexact local constraint, we need to solve the following convex constrained optimization problem:

$$M_k = \operatorname{argmin}_{M_k \in \mathbb{S}_+^{n_k^2+1}} \|\mathcal{A}_k(M_k) - b_k\|_F^2 \quad \text{s.t.} \quad G_k(M_k) = g_k. \tag{2.18}$$

The above problem can be solved using the interior point method implemented in the Mosek package. Additionally, once a Mosek model for the problem has been generated, it can be reused without regenerating the data. The only requirement is to modify the vector b_k as needed, which enables fast solving of the M-subproblem.

2.4.2 Solving the global problem

If the problem size is large, the main computational burden in Algorithm 1 arises from line 2. Suppose we intend to address a lattice system with thousands of sites. In the current form of our optimization algorithm (2.11), we would be working with a large PSD matrix variable X. While we can solve the sub-minimization problem by introducing another matrix variable $\tilde{X}$, applying the PSD constraint on $\tilde{X}$, requiring $X = \tilde{X}$, and forming a new augmented Lagrangian function, this method still requires calculating the projection of a large matrix onto the PSD cone. The complexity of this projection is cubic, making it unfeasible for large systems.

Due to the complexity of this projection, we propose solving (2.11) using a low-rank factorization of the primal variable X. Following the approach introduced in [12], we perform a low-rank factorization of the PSD variable in the form $X = xx^T$ where $x \in \mathbb{R}^{n \times r}$ is a tall matrix with $r < n$. This low-rank factorization eliminates the difficult PSD constraint,

converting the optimization problem into an unconstrained one, which can subsequently be solved using the limited-memory BFGS algorithm (L-BFGS) [38]. Additionally, r is chosen minimally and adapted throughout the ADMM updates, enabling the development of a fast algorithm. In the following, we will adopt this strategy and develop an algorithm based on the low-rank factorization of the primal PSD variable.

2.5 The primal method with low-rank positive semidefinite variable

In this section, we propose a variable substitution for the PSD variable X in algorithm 1, expressed as $X = xx^T$. Previous work [5, 50, 36] has shown that for every feasible SDP problem with m linear constraints, there exists an optimal solution X with $\text{rank}(X) = r$ such that $r(r+1)/2 \leq m$. Consequently, when addressing large SDP problems where the number of linear constraints is relatively small, we can opt to solve the SDP problem via the low-rank formulation in order to save computation time. Given that we assume the number and the size of the corrupted clusters on the lattice do not grow with the size of the lattice n, we have reason to believe that a low-rank solution to (2.11) also exists. Therefore, we propose solving the following optimization problem instead of (2.11):

$$\begin{aligned} \underset{x\in\mathbb{R}^{n\times r}}{\text{minimize}} \quad & \langle J, xx^T\rangle, \\ \text{subject to} \quad & \text{diag}(xx^T) = 1, \\ & (xx^T)(I_k, I_k) = \mathcal{A}_k(M_k) \text{ for some } M_k \in \mathcal{C}_k, \quad k = 1, \cdots, K, \end{aligned} \tag{2.19}$$

with either the exact or inexact local constraint. We now present Algorithm 2, which uses a low-rank formulation of the primal PSD variable.

According to the theoretical and experimental results presented in [12, 9], the rank parameter r should be updated dynamically so that x remains rank-deficient throughout the

Algorithm 2 ADMM for the primal problem with low-rank PSD variable

Require: $x, y, \{v_k\}_k$, $\{M_k\}_k$ and penalty parameter $\sigma > 0$

1: **while** not converged **do**
2: $\quad x \leftarrow \text{argmin}_x \; L_\sigma(xx^T, \{M_k\}_k; y, \{v_k\}_k)$
3: $\quad$ **for** $k = 1, \cdots, K$ **do**
4: $\quad\quad M_k \leftarrow \text{argmin}_{M_k \in \mathcal{C}_k} \|(xx^T)(I_k, I_k) - \mathcal{A}_k(M_k) + \frac{1}{\sigma} v_k\|_F^2$
5: $\quad\quad v_k \leftarrow v_k + \sigma \left((xx^T)(I_k, I_k) - \mathcal{A}_k(M_k) \right)$
6: $\quad$ **end for**
7: $\quad y \leftarrow y + \sigma(\text{diag}(xx^T) - 1_n)$
8: **end while**

execution of Algorithm 2. For our problem however, fixing $r = 10$ appears to generate good results. Additionally, it is important to note that we never need to form the full matrix xx^T for Algorithm 2. Instead, we only need to compute the entries used in $(xx^T)(I_k, I_k)$ for $k = 1, \cdots, K$ and $\text{diag}(xx^T)$.

2.5.1 Complexity analysis

In this section, we conduct a complexity analysis of Algorithm 2. Throughout this section, x has a fixed dimension with $x \in \mathbb{R}^{n \times r}$. We assume the number of columns r, the number of local clusters K, and the size of each local cluster n_k for $k = 1, \cdots, K$ are all constants with respect to n. Additionally, the interaction matrix J is assumed to be a sparse matrix with $O(n)$ many non-zero entries. As a result, the main computational burden for Algorithm 2 is in line 2.

With the variables $\{M_k\}_k$ fixed, we seek to minimize the augmented Lagrangian function with respect to the variable x:

$$\underset{x \in \mathbb{R}^{n \times r}}{\text{minimize}} \; f(x) := \langle J, xx^T \rangle + \frac{\sigma}{2} \sum_{k=1}^{K} \|(xx^T)(I_k, I_k) - a_k\|_F^2 + \frac{\sigma}{2} \|\text{diag}(xx^T) - c\|_F^2 \quad (2.20)$$

for constant vectors $\{a_k\}_{k=1}^K$ and c defined by:

$$\begin{aligned} a_k &:= \mathcal{A}_k(M_k) - \frac{1}{\sigma} v_k, \quad k = 1, \cdots, K, \\ c &:= 1_n - \sigma^{-1} y. \end{aligned}$$

We propose solving (2.20) by running the L-BFGS algorithm. The main computational costs in the L-BFGS algorithm lie in the evaluation of the function value and the calculation of its gradient. We claim both of these can be computed efficiently for our problem. To see this, we analyse the complexity of the function value evaluation, and the complexity of the gradient calculation can be analyzed in a similar way.

To start with, the first part of $f(x)$ in (2.20) is

$$\langle J, xx^T \rangle = \sum_{i,j} J_{ij}(xx^T)_{ij} = \sum_{J_{ij} \neq 0} J_{ij} \sum_{k=1}^{r} x_{ik} x_{jk}.$$

Let $\|J\|_0$ denote the number of non-zero elements in J. The total computation time for the first part of $f(x)$ is apparently $O(\|J\|_0 r)$.

In the second component in (2.20), we must compute expressions of the type

$$\|(xx^T)(I_k, I_k) - a_k\|_F^2$$

for $k = 1, \cdots, K$. For each k, we are required to calculate n_k^2 many entries in the matrix xx^T, and the total computation time is $O(n_k^2 r)$.

In the third component in (2.20), we must evaluate

$$\|\mathrm{diag}(xx^T) - c\|_F^2,$$

and using similar reasoning as with the first two components, the computational complexity

for this part is $O(nr)$.

In conclusion, the overall time complexity for the loss function evaluation is $O((n + \sum_{k=1}^{K} n_k^2 + \|J\|_0)r)$. A similar analysis for the gradient evaluation of the loss function shows that its total computational complexity is also $O((n + \sum_{k=1}^{K} n_k^2 + \|J\|_0)r)$. Thus, both the cost function evaluation and gradient computation require $O((n + \sum_{k=1}^{K} n_k^2 + \|J\|_0)r)$ time. For the Ising model with localized corruption under consideration, each site on the lattice is assumed to be connected to only a constant number of sites, making the matrix J sparse with $O(n)$ many non-zero entries. Additionally, the cluster size n_k for $k = 1, \cdots, K$ and the number of clusters K are constant values with respect to n. Therefore, the total per-iteration cost for Algorithm 2 scales linearly with the lattice size n.

2.5.2 Stopping criteria

For $k = 1, \cdots, K$, define a linear function $\mathcal{D}_k : \mathbb{S}^n \to \mathbb{S}^{n_k}$ by:

$$\mathcal{D}_k(X) = X(I_k, I_k). \tag{2.21}$$

The dual problem to (2.19) is:

$$\begin{aligned} \underset{y,\{v_k\}_k}{\text{minimize}} \quad & \big[\, \langle -1, y\rangle + \sum_k \max_{M_k \in \mathcal{C}_k} \langle -v_k, \mathcal{A}_k(M_k)\rangle \,\big], \\ \text{subject to} \quad & J - \operatorname{diag}(y) - \sum_{k=1}^{K} \mathcal{D}_k^*(v_k) \succeq 0, \end{aligned} \tag{2.22}$$

with $y \in \mathbb{R}^n$ and $v_k \in \mathbb{S}^{n_k}$ for $k = 1, \cdots, K$.

Throughout the execution of Algorithm 2, we measure the accuracy of an approximate solution $(x, \{M_k\}_{k=1}^K, y, \{v_k\}_{k=1}^N)$ for the primal and dual problems (2.19) and (2.22) by monitoring the relative primal feasibility η_P, the relative dual feasibility η_D and the relative

duality gap η_g. Let $X = xx^T$. For the primal feasibility measure, we define

$$P_x = \big[\mathrm{diag}(X);\ \mathrm{vec}(\mathcal{D}_1(X))\ \cdots\ \mathrm{vec}(\mathcal{D}_k(X))\big],$$

$$P_M = \big[1_n;\ \mathrm{vec}(\mathcal{A}_1(M_1))\ \cdots\ \mathrm{vec}(\mathcal{A}_k(M_k))\big],$$

and

$$\eta_P = \frac{\|P_x - P_M\|_F}{1 + \|P_x\|_F + \|P_M\|_F}.$$

For the dual feasibility, let $\lambda_{\min}$ and $\lambda_{\max}$ be the smallest and largest eigenvalues of the matrix $J - \mathrm{diag}(y) - \sum_{k=1}^{K} \mathcal{D}_k^*(v_k)$, and we define

$$\eta_D = \frac{\max(0, -\lambda_{\min})}{1 + \max(0, \lambda_{\max})}.$$

For the duality gap, we compute

$$\eta_g = \frac{|\text{primal objective} - \text{dual objective}|}{1 + |\text{primal objective}| + |\text{dual objective}|},$$

where primal objective and dual objective are the objective values of (2.19) and (2.22) with the variables $(x, \{M_k\}_{k=1}^K)$ and $(y, \{v_k\}_{k=1}^K)$.

If we use the inexact local constraint described in (2.15), we terminate our algorithm when $\eta = \min\{\eta_P, \eta_D, \eta_g\} \leq 10^{-4}$. If we use the exact local constraint described in (2.12), the optimization problem is non-convex, and the relative duality gap usually does not converge to 0, and we terminate our algorithm when $\eta = \min\{\eta_P, \eta_D\} \leq 10^{-4}$.

2.6 The dual method with low-rank positive semidefinite variable

Alternatively, we can solve the dual problem of (2.11) with the inexact local constraint in (2.15):

$$\begin{aligned} \underset{y,\{v_k\}_k,S}{\text{minimize}} \quad & \big[\langle -1, y\rangle + \sum_{k=1}^{K} \delta^*_{\mathcal{C}_k}(-\mathcal{A}_k^*(v_k))\big], \\ \text{subject to} \quad & S + \text{diag}(y) + \sum_{k=1}^{K} \mathcal{D}_k^*(v_k) = J, \end{aligned} \tag{2.23}$$

with $y \in \mathbb{R}^n$, $v_k \in \mathbb{S}^{n_k}$ for $k = 1, \cdots, K$ and $S \in \mathbb{S}^n_+$. The function $\delta^*_{\mathcal{C}_k}(-w_k)$ for $k = 1, \cdots, K$ is defined by:

$$\delta^*_{\mathcal{C}_k}(-w_k) = \sup\{\langle -w_k, p_k\rangle \mid p_k \in \mathcal{C}_k\}.$$

To apply the ADMM algorithm for (2.23), we write down its associated augmented Lagrangian function $L_\sigma(S, y, \{v_k\}; X)$ with $S \in \mathbb{S}^n_+$, $v_k \in \mathbb{S}^{n_k}$ for $k = 1, \cdots, K$, $y \in \mathbb{R}^n$, $X \in \mathbb{S}^n$ and penalty parameter $\sigma > 0$ as follows:

$$L_\sigma(S, y, \{v_k\}; X) = \begin{cases} \langle -1, y\rangle + \sum_{k=1}^{K} \delta^*_{\mathcal{C}_k}(-\mathcal{A}_k^*(v_k)) \\ \\ +\frac{\sigma}{2}\|S + \text{diag}(y) + \sum_{k=1}^{K} \mathcal{D}_k^*(v_k) - J + \frac{1}{\sigma}X\|_F^2 - \frac{1}{2\sigma}\|X\|_F^2. \end{cases} \tag{2.24}$$

We then iterate Algorithm 3 until convergence. It is important to note that, for the dual problem, we assume that the set $\{I_k\}_{k=1}^K$ is pairwise disjoint, allowing the local problems to be solved in parallel.

Step 2 in Algorithm 3 involves a least square problem which can be solved in $O(n)$ time. In step 4, for $k = 1, \cdots, K$, define

$$\tilde{\mathcal{C}}_k := \{\tilde{M}_k \in \mathbb{S}^{n_k} : \tilde{M}_k = \mathcal{A}_k(M_k) \text{ for some } M_k \in \mathcal{C}_k\},$$

Algorithm 3 ADMM for the dual problem

Require: $S, y, \{v_k\}_k, X$, and penalty parameter $\sigma > 0$

1: **while** not converged **do**
2: $y \leftarrow \text{argmin}_y L_\sigma(S, y, \{v_k\}_k; X)$
3: **for** $k = 1, \cdots, K$ **do**
4: $v_k \leftarrow \text{argmin}_{v_k} \left[\delta^*_{\mathcal{C}_k}(-\mathcal{A}^*_k(v_k)) + \frac{\sigma}{2}\|S + \text{diag}(y) + \mathcal{D}^*_k(v_k) - J + \frac{1}{\sigma}X\|^2_F\right]$
5: **end for**
6: $S \leftarrow \text{argmin}_{S\succeq 0} L_\sigma(S, y, \{v_k\}; X)$
7: $X \leftarrow X + \sigma(S + \text{diag}(y) + \sum_{k=1}^K \mathcal{D}^*_k(v_k))$
8: **end while**

and

$$R_k := \mathcal{D}_k(S + \text{diag}(y) - J + \frac{1}{\sigma}X).$$

Step 4 involves solving the following problem:

$$\begin{aligned} v_k &= \text{argmin}_{v_k}\big[\delta^*_{\mathcal{C}_k}(-\mathcal{A}^*_k(v_k)) + \frac{\sigma}{2}\|v_k - R_k\|^2_F\big] \\ &= \text{argmin}_{v_k}\big[\delta^*_{\tilde{\mathcal{C}}_k}(-v_k) + \frac{\sigma}{2}\|v_k - R_k\|^2_F\big] \\ &= \sigma^{-1}\Pi_{\tilde{\mathcal{C}}_k}(-\sigma R_k) + R_k, \end{aligned} \tag{2.25}$$

and the last equality holds due to the Moreau decomposition of proximal operators and several basic properties of proximal operators [6]. Thus, the minimization problem in (2.25) can be solved via the interior point method implemented in the Mosek package.

The most computationally demanding task continues to be the projection of a large matrix onto the PSD cone in step 6. To address this concern, we modify Algorithm 3 for the dual problem, taking advantage of the low-rank characteristic of X. Interested readers can consult [28] for a reference of this method.

To start with, from the update rule of X in Algorithm 3 step 7, we can rewrite the update rule of S in the following form:

$$S^{t+1} = (X^{t+1} - X^t)/\sigma + J - \text{diag}(y^{t+1}) - \sum_{k=1}^K \mathcal{D}^*_k(v^{t+1}_k). \tag{2.26}$$

Here, we use superscripts on variables to indicate the iteration number within the ADMM algorithm. We notice that we can combine steps 6 and 7 of Algorithm 3 to amend the update rule of the primal variable X:

$$\begin{aligned} X^{t+1} &= X^t + \sigma\big(S^{t+1} + \operatorname{diag}(y^{t+1}) + \sum_{k=1}^{K} \mathcal{D}_k^*(v_k^{t+1}) - J\big) \\ &= \big(X^t + \sigma(\operatorname{diag}(y^{t+1}) + \sum_{k=1}^{K} \mathcal{D}_k^*(v_k^{t+1}) - J)) \\ &\qquad + \Pi_{\mathbb{S}_n^+}\big(- X^t - \sigma(\operatorname{diag}(y^{t+1}) + \sum_{k=1}^{K} \mathcal{D}_k^*(v_k^{t+1}) - J)) \\ &= \Pi_{\mathbb{S}_n^+}\big(X^t + \sigma(\operatorname{diag}(y^{t+1}) + \sum_{k=1}^{K} \mathcal{D}_k^*(v_k^{t+1}) - J)). \end{aligned} \tag{2.27}$$

The last step is true because for any symmetric matrix $M \in \mathbb{S}^n$, we have

$$M + \Pi_{\mathbb{S}_n^+}(-M) = \Pi_{\mathbb{S}_n^+}(M).$$

Given this updated insight, we can replace every instance of S^{t+1} as a function of X^{t+1}, X^t, y^{t+1} and $\{v_k^{t+1}\}_{k=1}^K$ using (2.26) throughout the updates of Algorithm 3. The primary challenge now lies in efficiently computing the update of the primal variable X^t to X^{t+1} in (2.27). To this end, we propose working again with a low-rank representation of X in the form of $X^t = x^t(x^t)^T$ with $x^t \in \mathbb{R}^{n\times r}$ for some $r < n$. This modification will enable us to reduce the overall computation time and memory used regarding the update of X^t. However, in each ADMM update of this revised formulation of the primal variable, we still need to compute the projection on the PSD cone of the matrix

$$\tilde{X}^{t+1} := x^t(x^t)^T + \sigma\left(\operatorname{diag}(y^{t+1}) + \sum_{k=1}^{K} \mathcal{D}_k^*(v_k^{t+1}) - J\right). \tag{2.28}$$

To this end, we seek an approximate solution by computing only the top eigenvalues and eigenvectors of $\tilde{X}^{t+1}$. This can be achieved efficiently using the Lanczos process, whose computational complexity is determined by the complexity of the matrix-vector product between $\tilde{X}^{t+1}$ and vectors in $\mathbb{R}^n$. Given that $x^t(x^t)^T$ is a low-rank matrix and $\mathrm{diag}(y^{t+1}) + \sum_{k=1}^{K} \mathcal{D}_k^*(v_k^{t+1}) - J$ is sparse, the complexity of this matrix-vector product is linear with respect to the lattice size n. Therefore, this approximate projection onto the PSD cone operation can be performed efficiently. We now give a step-by-step modified version of Algorithm 3.

Step 1. Compute

$$y^{t+1} = y^t - \sigma^{-1}\mathrm{diag}\left(2x^t(x^t)^T - x^{t-1}(x^{t-1})^T\right) + \sigma^{-1}1_n.$$

Step 2. For $k = 1, \cdots, K$, define

$$R_k^{t+1} = v_k^t + \mathcal{D}_k\left(\mathrm{diag}(y^t) - \mathrm{diag}(y^{t+1}) - \sigma^{-1}(2x^t(x^t)^T - x^{t-1}(x^{t-1})^T)\right),$$

and compute

$$v_k^{t+1} = \mathrm{argmin}_{v_k}\{\delta_{C_k}^*(-\mathcal{A}_k^*(v_k)) + \frac{\sigma}{2}\|v_k - R_k^{t+1}\|_F^2\},$$

via the interior point method in the Mosek package.

Step 3. Compute the top r eigenvectors and eigenvalues of the matrix

$$\tilde{X}^{t+1} = x^t(x^t)^T + \sigma\left(\mathrm{diag}(y^{t+1}) + \sum_{k=1}^{K}\mathcal{D}_k^*(v_k^{t+1}) - J\right) \tag{2.29}$$

via the Lanczos process and denote them by $U^{t+1} = (u_1^{t+1}, \cdots, u_r^{t+1}) \in \mathbb{R}^{n\times r}$ and $\Sigma^{t+1} =$

$\mathrm{Diag}(\sigma_1^{t+1}, \cdots, \sigma_r^{t+1}) \in \mathbb{R}^{r\times r}$. Let

$$x^{t+1} = U^{t+1}\mathrm{sqrt}(\Sigma_+^{t+1}), \tag{2.30}$$

where Σ_+^{t+1} is constructed by setting negative entries in Σ^{t+1} to zeroes.

We are now ready to present Algorithm 4, which summarises the above modifications to Algorithm 3.

Algorithm 4 ADMM for the dual problem with low-rank X

Require: $t = 0, y^0, \{v_k^0\}_k$, x^0, x^{-1}, and penalty parameter $\sigma > 0$

while not converged **do**
  $y^{t+1} \leftarrow y^t - \sigma^{-1}\mathrm{diag}(2x^t(x^t)^T - x^{t-1}(x^{t-1})^T) + \sigma^{-1}1_n$
  for $k = 1, \cdots, K$ **do**
    $R_k^{t+1} \leftarrow v_k^t + \mathcal{D}_k\left(\mathrm{diag}(y^t) - \mathrm{diag}(y^{t+1}) - \sigma^{-1}(2x^t(x^t)^T - x^{t-1}(x^{t-1})^T)\right)$
    $v_k^{t+1} \leftarrow \mathrm{argmin}_{v_k}\{\delta^*_{C_k}(-\mathcal{A}_k^*(v_k)) + \frac{\sigma}{2}\|v_k - R_k^{t+1}\|_F^2\}$
  end for
  Compute x^{t+1} according to (2.29)-(2.30)
  $t \leftarrow t + 1$
end while

2.6.1 Stopping criteria

When applying the dual method to for a large-scale system, we suggest replacing the update of S with the update of a low-rank variable x. This adjustment aims to mitigate the challenges related to computation and storage, and the dual constraint $S \succeq 0$ is automatically satisfied with our new method. So the only dual constraint remaining is the linear constraint

$$\mathrm{diag}(y^t) + S^t + \sum_{k=1}^{K} \mathcal{D}_k^*(v_k^t) = J.$$

Substituting S^t with

$$\sigma^{-1}\left(x^t(x^t)^T - x^{t-1}(x^{t-1})^T\right) + J - \mathrm{diag}(y^t) - \sum_{k=1}^{K} \mathcal{D}_k^*(v_k^t),$$

we can rewrite the dual equality constraint as

$$\mathrm{diag}(y^t) + \sum_{k=1}^{K} \mathcal{D}_k^*(v_k^t) + \sigma^{-1}\left(x^t(x^t)^T - x^{t-1}(x^{t-1})^T\right) + J - \mathrm{diag}(y^t) - \sum_{k=1}^{K} \mathcal{D}_k^*(v_k^t) = J,$$

or equivalently

$$\sigma^{-1}\left(x^t(x^t)^T - x^{t-1}(x^{t-1})^T\right) = 0.$$

So, to monitor the dual feasibility measure, we track

$$\eta_D := \frac{\|x^t(x^t)^T - x^{t-1}(x^{t-1})^T\|_F}{\sigma(1 + \|J\|_F)}.$$

For the primal feasibility constraint, as our primal matrix variable X is represented via a low-rank decomposition $X = xx^T$, the PSD constraint is also automatically satisfied. As we do not keep a set of primal variables $\{M_k\}_{k=1}^K$ for the dual method to test if X satisfies the local constraints, for $k = 1, \cdots, K$, we first compute

$$M_k = \mathrm{argmin}_{M_k \in \mathcal{C}_k} \|\mathcal{D}_k(X) - \mathcal{A}_k(M_k)\|_F^2,$$

and then we repeat the steps applied in the primal method, i.e., we define:

$$P_x := \big[\mathrm{diag}(X);\ \mathrm{vec}(\mathcal{D}_1(X))\ \cdots\ \mathrm{vec}(\mathcal{D}_k(X))\big],$$

$$P_M := \big[1_n;\ \mathrm{vec}(\mathcal{A}_1(M_1))\ \cdots\ \mathrm{vec}(\mathcal{A}_k(M_k))\big],$$

and

$$\eta_P := \frac{\|P_x - P_M\|_F}{1 + \|P_x\|_F + \|P_M\|_F}.$$

We also track the relative duality gap defined by

$$\eta_g = \frac{|\text{primal objective} - \text{dual objective}|}{1 + |\text{primal objective}| + |\text{dual objective}|},$$

and terminate our algorithm when $\eta := \min\{\eta_P, \eta_D, \eta_g\}$ is below a pre-specified threshold.

2.7 Numerical experiments

In this section, we present several experiment results for the recovery of the ground-state and ground-state energy of the Ising model with localized corruption on a square 2-D lattice, as illustrated in Figure 2.1. We identify a square 2-D lattice with an undirected graph $G = (V, E)$, with

$$\begin{aligned} V &= \{(x, y) : \ x, y \in \mathbb{N}, \ 1 \leq x, y \leq n\}, \\ E &= \{\{v_1, v_2\} : \ v_1, v_2 \in V, \ \text{dist}(v_1, v_2) = 1\} \end{aligned} \tag{2.31}$$

for varying sizes of n. The distance on the 2-D lattice is defined by

$$\text{dist}(v, w) := \|v - w\|_1, \ v, w \in V.$$

Notice that the distance between v and w is also the number of edges needed to travel from v to w on the 2-D lattice. Additionally, we randomly sample m clusters $\{N_1, N_2, \cdots N_m\}$ of size 2×2 on the 2-D lattice, assuming these clusters are not on the edge of the lattice. Furthermore, we assume these clusters are "well-separated," i.e., for any two clusters N_i, N_j, we have

$$\text{dist}(N_i, N_j) := \min\left\{\text{dist}(v_i, v_j), \ v_i \in N_i, v_j \in N_j\right\} \geq 2.$$

We define an interaction matrix $J : V \times V \to \mathbb{R}$ on the graph G such that

$$J(v,w) := \begin{cases} \epsilon\, a_{v,w}, & v, w \in N_i \text{ for some } N_i, \\ -1, & \text{dist}(v,w) = 1, v, w \notin N_i \text{ for any } N_i, \\ 0, & \text{othewise.} \end{cases} \tag{2.32}$$

Here, $\epsilon > 0$ is a parameter that controls the strength of the corruption, and $a_{v,w}$ is sampled from the standard normal distribution.

2.7.1 Tightness of the proposed relaxation

We first examine the tightness of the relaxation in equation (2.11) for the original problem (2.2). To achieve this, we solve the relaxed problem (2.11) with several types of convex relaxation using the Mosek package. This approach provides valid lower bounds for the ground-state energy in (2.2). Additionally, we propose two methods to construct feasible primal variables for the original problem, yielding valid upper bounds for the ground-state energy. This allows us to establish a "confidence interval" for the ground-state energy. If the solution for (2.11) returned by Mosek is also feasible for the original unrelaxed problem, or if the width of the confidence interval is zero, we can conclude that the relaxation is tight.

We consider an Ising model with localized corruption on a 10×10 lattice with three local clusters N_1, N_2, N_3, each of size 2×2, and the noise level $\epsilon = 2$. Note that we can choose between two types of relaxation, with the exact or the inexact local constraint, specified in (2.12) or in (2.15) respectively. Additionally, we have multiple ways to select indices $I_1, \cdots, I_K$ where we apply the local constraints. For each instance of the randomly generated interaction matrix J, we test the relaxation with the following three types of relaxation:

- with the exact local constraint described in (2.12), and $I_k = \text{one-hop}(N_k)$ for $k = 1, 2, 3$.

- with the inexact local constraint described in (2.15), and I_v = one-hop($\{v\}$) for all $v \in E$.

- with the inexact local constraint described in (2.15), and I_k = one-hop(N_k) for $k = 1, 2, 3$.

Here, the function one-hop is defined in (2.10). The relaxed problem (2.11) with the first type of relaxation is non-convex, and we solve it by running Algorithm 2 until both the relative primal and dual feasibility measures, as defined in section 2.5.2, are below 10^{-4}. In contrast, the relaxed problem (2.11) with the second or third type of constraint is convex, and we solve it using the Mosek package.

For the first type of relaxation, the duality gap remains non-zero until the algorithm converges, meaning we do not have a certificate of accuracy for the solution. Instead, we check if the resulting x is of rank one by calculating the ratio of its largest and second-largest singular values. If this ratio is greater than 10^6, we consider x to have rank one, and if x is of rank one, xx^T is a feasible solution for (2.2). Consequently, we obtain an upper bound for the ground-state energy by substituting xx^T into (2.2).

For the second and third type of relaxation, the Mosek package can solve the relaxed problems to high precision, and the objective values from these two problems serve as lower bounds for the ground-state energy. Additionally, if the ratio between the largest and second-largest eigenvalues of the solution X is greater than 10^{12}, we consider X to have rank one, indicating that the relaxation is tight.

In addition to solving the relaxed problem with the above three types of relaxation to approximate the ground-state energy, we employ two additional methods for approximation. The first method is coordinate descent, detailed in Algorithm 5. The resulting x from this method is always feasible, and thus $H(x)$ in (2.1) provides an upper bound for the ground-state energy. Furthermore, we solve the Max-Cut relaxation in (2.9) using the Mosek package, with the objective value serving as an additional lower bound for the ground-state energy.

Algorithm 5 Coordinate descent for (2.1)

Require: $x \in \{-1, +1\}^n$
1: **while** not converged **do**
2: **for** $i = 1, \cdots, n$ **do**
3: Let $\tilde{x} = x$, and then let $\tilde{x}(i) = -\tilde{x}(i)$
4: Set $x = \tilde{x}$ if $H(x) > H(\tilde{x})$
5: **end for**
6: **end while**

We repeated the experiment for 20 times. In all 20 trials, the problem with the first type of relaxation always generated rank one solutions. Thus, the objective values obtained from these experiments serve as valid upper bounds for the ground-state energy.

We verify the tightness of the relaxation by checking the rank of the solutions from the convex relaxation of the problem (relaxation types 2 and 3, and Max-Cut relaxation) or by comparing the lower and upper bounds of the ground-state energy. We observe that among all 20 experiments, only experiments 1 and 6 are not tight. Thus, in 18 out of 20 trials, the convex relaxation returns the exact ground-state energy of the original problem. Furthermore, in all cases, our relaxation problems provides much tighter upper and lower bounds for the ground-state energy than the coordinate descent and Max-Cut relaxation methods.

2.7.2 Computational speed of the algorithm with low-rank formulation of the matrix variable

In this section, we examine the efficiency of Algorithm 2 with two types of relaxation specified in (2.12) and (2.15), on 2-D lattices of size $10 \times 10, 20 \times 20, 40 \times 40, 60 \times 60, 80 \times 80$ and 100×100. We assume there are two well-separated local clusters N_1, N_2, each of size 2×2 on the lattice, and the strength of the corruption is $\epsilon = 2$. The corresponding interaction matrix is described in (2.32). We impose the local exact/inexact constraint on the one-hop neighbors (defined in (2.10)) of N_1 and N_2.

Experiment	Upper bounds		Lower bounds		
	Coord Descent	Type 1	Type 2	Type 3	Max-Cut
1	-316.4203	-326.6678	-326.6747	-326.8839	-337.7989
2	-314.6473	-316.8713	-316.8713*	-316.8713*	-328.1423
3	-313.0874	-321.6074	-321.6073*	-322.06	-336.215
4	-321.9702	-321.9703	-321.9702	-322.3243	-335.313
5	-320.9977	-337.3389	-337.7763*	-337.7792	-347.8577
6	-325.9782	-329.6889	-329.7011	-331.539	-345.2598
7	-332.9327	-337.3438	-337.3437*	-337.3437*	-344.589
8	-326.6027	-335.2236	-336.7875*	-336.7875*	-345.2221
9	-329.0979	-329.3048	-333.8823*	-334.002	-345.8276
10	-316.8543	-325.8311	-326.0037*	-326.0037*	-335.6402
11	-329.1189	-331.2435	-331.2434*	-331.265	-342.5557
12	-323.5778	-332.0073	-332.0072	-332.0072	-342.4065
13	-318.8727	-323.4228	-323.4228*	-323.4228*	-331.9729
14	-314.4191	-314.4192	-314.4303	-314.4191	-327.324
15	-334.7562	-337.4867	-337.4867*	-337.7093	-348.9947
16	-321.0631	-324.6084	-324.9322*	-325.2647	-338.7968
17	-326.5052	-327.5403	-327.5402*	-328.3839	-341.0507
18	-332.4943	-334.8598	-334.8597*	-334.9907	-345.4378
19	-322.7339	-318.0348	-322.7339	-326.0967	-339.7306
20	-322.0832	-332.4221	-332.422	-333.9074	-346.0861

Figure 2.2: Lower and upper bounds of the ground-state energy. The smaller of the two upper bounds is colored red, while the largest of the three lower bounds is colored blue. A lower bound value is marked with an asterisk ($*$) if the relaxation is tight (solution is rank one).

For the relaxation problem with the inexact local constraint, we run Algorithm 2 until the relative primal feasibility, relative dual feasibility and relative duality gap defined in section 2.5.2 are all below 10^{-4}. For the relaxation problem with the exact local constraint, we run Algorithm 2 until the relative primal feasibility and relative dual feasibility are below 10^{-4}.

We present the total computation time needed, as well as the computation time needed in step 2 of Algorithm 2 (the L-BFGS step) for each lattice size, averaged over 12 trials.

As shown in Table 2.2, the computation time for the L-BFGS step scales approximately linearly with the size of the lattice. However, due to the large amount of time needed for the Mosek package to solve the M-subproblem for local constraints described in (2.15), the total computation time remains substantial.

size of lattice	10×10	20×20	40×40	60×60	80×80	100×100
Exact local constraint	11.78	7.89	15.6	28.42	40.29	64.99
Inexact local constraint	2549.8	2882.9	2240.2	3052.6	3081.3	2332.4

Table 2.1: Total computation time of Algorithm 2 (in seconds).

size of lattice	10×10	20×20	40×40	60×60	80×80	100×100
Exact local constraint	3.06	3.58	8.13	19.12	31.07	51.3
Inexact local constraint	14.35	35.64	47.54	90.96	186.54	200.39

Table 2.2: L-BFGS step computation time of Algorithm 2 (in seconds).

2.7.3 The issue of multiple solutions for the proposed relaxation

We have observed from Figure 2.2 that when using a relaxation with the inexact local constraint, the resulting solution might not be of rank one. One reason for this phenomena is the potential existence of multiple solutions to the original problem (2.2).

To illustrate this, suppose $X_1, X_2 \in \mathbb{S}^n_+$ are two solutions to the original problem (2.2). Since (2.11) is a relaxed problem for (2.2), there exist two sets of variables $\{M_k^1\}_{k=1}^K$ and

$\{M_k^2\}_{k=1}^K$ such that both $(X_1, \{M_k^1\}_{k=1}^K)$ and $(X_2, \{M_k^2\}_{k=1}^K)$ are feasible for the relaxed problem (2.11). It is easy to check that any weighted average of these two solutions, i.e.,

$$\Big(wX_1 + (1-w)X_2, \big\{wM_k^1 + (1-w)M_k^2\big\}_{k=1}^K\Big), \quad \forall w \in [0,1] \tag{2.33}$$

is feasible for the relaxed problem (2.11) as well. Thus, it is possible that Algorithm 2 can return a PSD variable that is not of rank one, even though the objective value from the PSD variable is the ground-state energy. This has occurred in experiments $4, 12, 14, 19$ and 20 as illustrated in Figure 2.2.

However, the problem of obtaining a single solution from a linear combination of solutions remains an open question in the field of convex relaxation. Consequently, we have yet to develop an extraction algorithm for our relaxed problem. However, we conducted the following experiment to verify that the issue of multiple solutions does exist for the model under consideration.

We tested our relaxation on the "corrupted pairs model", as shown in Figure 2.3. Unlike the model used in section 2.7.1, the local clusters $N_1, \cdots, N_m$ in the corrupted pairs model have sizes equal to 1×2 or 2×1. The primary reason for testing our algorithm on this model is that the analytic solution of its ground-state is known. Specifically, if $J_{v,w} > 3$ for any v, w in a cluster, then multiple solutions for (2.2) exist.

We conducted an experiment on a 10×10 lattice with two local clusters N_1, N_2, each of size 2×1 or 1×2. We set $\epsilon = 3$ and ran Algorithm 2 with the inexact local constraint for 20 times. We discovered that all the resulting PSD variables lay in the linear space spanned by the true ground-states. Thus, the issue of multiple solutions does exist for our relaxation.

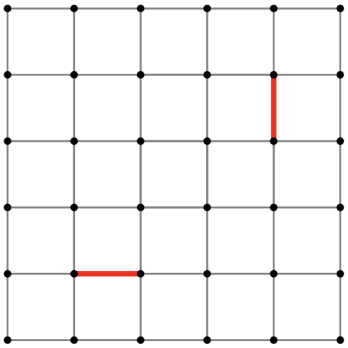

Figure 2.3: Corrupted pairs model on a 2-D lattice.

2.8 Summary

In this chapter, we explored a novel method to recover the ground-state energy for the Ferromagnetic model with localized corruption. We formulated a relaxed problem that combines a relatively inexpensive global PSD constraint with several costly local constraints to accurately capture fluctuations in the corrupted regions. Additionally, we utilized the low-rank decomposition of the global PSD variable to eliminate the expensive step of performing a projection onto the PSD cone, reducing the per-iteration cost of the ADMM algorithm from cubic to linear complexity. Our algorithm was tested on 2-D lattices of varying sizes, demonstrating its effectiveness on lattices with up to 10000 sites. Furthermore, we evaluated the tightness of our relaxation by examining the recovered energy and the rank of the solution on a 10×19 lattice. We demonstrated that our relaxation is able to recover the exact ground-state energy in most cases, and in the remaining cases, it provides an accurate "confidence interval" for the ground-state energy. Lastly, we analyzed a potential reason why the relaxation does not always yield a solution of rank one, showing that our algorithm can return a linear combination of the ground truths. Extracting a single solution from this linear combination will be a focus for future research.

CHAPTER 3

CONVEX RELAXATION FOR QUANTUM SPIN PROBLEM

In this chapter, we explore a novel method for studying the ground-state energy of the quantum many-body problem, following the work in [59]. The quantum many-body problem involves analyzing systems composed of a large number (denoted by N) of interacting quantum particles. The Hamiltonian, a fundamental operator in these systems, describes their total energy. Among the many questions posed in quantum many-body problems, determining the ground-state is crucial for various applications across physics, chemistry, material sciences, and beyond. The ground-state problem involves identifying a configuration with the lowest energy within a quantum system and determining the corresponding energy associated with it. From a mathematical standpoint, this requires finding the lowest eigenvalue and its associated eigenvector of the Hamiltonian, which is a Hermitian operator defined on a Hilbert space. Specifically, we aim to find the wavefunction Φ that solves the following optimization problem:

$$E_0 := \inf_{\Phi \in Q} \Big\{ \langle \Phi, \hat{H}\Phi \rangle,\ \langle \Phi, \Phi \rangle = 1 \Big\}, \tag{3.1}$$

where Q denotes the underlying Hilbert space. As we will see soon, this problem can be written equivalently as a standard SDP problem, similar to the ground-state energy problem we considered for the Ferromagnetic Ising model with localized corruption in chapter 2. The primary challenge of the problem in chapter 2 arises from the rank constraint imposed on the PSD variable. In this chapter, however, the main challenge arises from the exponential growth of the dimension of the Hilbert space Q with the system's size, i.e., the number of interacting quantum particles N in the system, making it infeasible to solve the eigenvalue problem (3.1) directly using standard numerical methods for large quantum systems.

Various methods have been proposed to overcome this difficulty, which can be categorized into several classes. One such class is Quantum Monte-Carlo, which forgoes representing

the full wavefunction and instead maintains a set of random walkers to calculate quantum expectation [14, 15]. Another class is wavefunction methods, which usually restrict the search space to a small subspace of the exponentially large Hilbert space [47, 46, 16]. A third class of methods involves reframing the minimization problem using reduced density matrices (RDM), in a manner similar to mean-field methods [11]. To understand this class of methods, note that the ground-state energy problem (3.1) can be equivalently reformulated as an optimization problem over the density operators (or density matrices):

$$E_0 := \inf_{\rho \in \mathcal{D}(Q)} \operatorname{Tr}[\hat{H}\rho], \tag{3.2}$$

where $\mathcal{D}(Q)$ denotes the collection of all density operators on Q, i.e., all PSD operators on Q with unit trace. Subsequently, the loss function in (3.2) is equivalently rewritten as a linear function of the "moments" of the density matrices. After this transformation, certain necessary constraints that ensure these moments originate from a true density matrix are lifted, leading to a relaxed convex optimization problem that can be solved within polynomial time. This type of method is guaranteed to return a lower bound of the true ground-state energy, i.e., a lower bound on the objective value obtained in (3.2). A well-known application of this class of methods is the 2-RDM method [3, 13, 19, 40, 41, 42, 43, 61].

Another approach to addressing the quantum many-body problem is through quantum embedding theories [56]. These theories involve partitioning the system into smaller regions of interest, which can be effectively treated using highly accurate yet computationally intensive methods. Concurrently, the local problems are glued together self-consistently through a global, less accurate but more computationally efficient approach. Numerous techniques fall under this category, such as dynamical mean-field theory [24, 33] and density matrix embedding theory [31, 32] to name a few. Recently, the variational embedding method [37, 30] has emerged. As an alternative to using the "moments" as the optimization variables in the 2-RDM method, the variational embedding method reformulates the loss function in (3.2)

in terms of the "quantum marginals" derived from a full density matrix. This approach retains the inherent constraints on the quantum marginals, ensuring they remain accurate local constraints. Simultaneously, the global constraint that all quantum marginals originate from a single global quantum density matrix is replaced with a necessary PSD constraint. In this sense, this method can be seen as a hybrid of embedding methods and RDM methods.

In this paper, we adopt a strategy similar to the variational embedding method, where we try to determine local cluster density matrices (quantum marginals) and combine them through a global PSD constraint. The difference is that the local cluster density matrices are represented through their moments. In this case, the decision variable is a PSD moment matrix. The main point of this paper is to propose a method to accelerate the PSD optimization problem therein. Typically, the most computationally expensive step in such an optimization problem is the projection onto the PSD cone, which scales cubically. In [30], translation invariance of the Hamiltonian is exploited in order to diagonalize the PSD matrix in the Fourier basis with a linear time complexity. However, it is unclear how such computational scaling can be achieved for general systems.

Contributions:

For spin systems, we propose representing the dual variable of the PSD moment matrix with a specific type of hierarchical matrix [8]. In our experiments, such a structure of the dual matrix seems to hold for the transverse field Ising model, even at the quantum phase transition point. We show that with such a structure of the dual variable, updates within an augmented Lagrangian method can be carried out with quadratic complexity. Furthermore, if one assumes the primal moment matrix also takes the form of a hierarchical PSD matrix, near-linear per-iteration complexity can be achieved. Our designed algorithm is able to solve the 1-D transverse field Ising model for systems of up to 4000 sites, and recover a lower bound of the ground-state energy with a relative error of approximately 2% compared to the true value.

3.1 Problem formulation

Let's consider a spin-$\frac{1}{2}$ quantum many-body problem with N sites, each having a quantum state space $Q_i := \mathbb{C}^2$, and the global quantum space is defined by $Q := \otimes_{i=1}^{N} Q_i \simeq \mathbb{C}^{2^N}$. Let H_i be any Hermitian operator $Q_i \to Q_i$, and H_{ij} be any Hermitian operator $Q_i \otimes Q_j \to Q_i \otimes Q_j$. We use $\hat{H}_i$ to denote the operator $Q \to Q$ obtained by tensoring H_i by the identity operators on all sites $k \neq i$, and similarly, we use $\hat{H}_{ij}$ to denote the operator $Q \to Q$ obtained by tensoring H_{ij} by the identity operators on all sites $k \notin \{i, j\}$. Then, we consider a pairwise Hamiltonian $\hat{H} : Q \to Q$ of the form

$$\hat{H} = \sum_{1 \leq i \leq N} \hat{H}_i + \sum_{1 \leq i < j \leq N} \hat{H}_{ij}, \tag{3.3}$$

and $\hat{H}$ is a Hermitian matrix of size 2^N. Our goal is to determine the ground-state energy of the system as defined in (3.1). Now, let's delve into a specific example.

Example 3. *Consider the Pauli matrices, and the* 2*-dimensional identity matrix defined below:*

$$\sigma^x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma^y = \begin{pmatrix} 0 & i \\ -i & 0 \end{pmatrix}, \quad \sigma^z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad I_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \tag{3.4}$$

These matrices form a basis for the real vector space of Hermitian operators on $\mathbb{C}^2$*. Furthermore, let* $\sigma_i^{x/y/z}$ *denote the operator obtained by tensoring* $\sigma^{x/y/z}$ *on the* i*-th site with identities* I_2 *on all other sites, i.e.*

$$\sigma_i^{\alpha} := I_2^{\otimes(i-1)} \otimes \sigma^{\alpha} \otimes I_2^{\otimes(N-i)}, \ \alpha \in \{x, y, z\}, \tag{3.5}$$

and let $I : Q \to Q$ *be the identity operator on* Q*. The* 1*-D transverse field Ising (TFI) model,*

a spin-$\frac{1}{2}$ quantum model, is defined by its Hamiltonian:

$$\hat{H}_{TFI} = -h\sum_{i=1}^{N}\sigma_i^x - \sum_{i=1}^{N}\sigma_i^z\sigma_{i+1}^z, \tag{3.6}$$

where the TFI model is assumed to have periodic boundary conditions, i.e., $\sigma_{N+1}^{x/y/z}$ should be identified with $\sigma_1^{x/y/z}$, and $h \in \mathbb{R}$ is a scalar parameter controlling the strength of the external magnetic field along the x axis. To demonstrate that the Hamiltonian defined in equation (3.6) *is a special case of the Hamiltonian in equation* (3.3)*, we can observe that the term $-h\sigma_i^x$ in* (3.6) *corresponds to the $\hat{H}_i$ component in* (3.3)*, while $-\sigma_i^z\sigma_{i+1}^z$ in* (3.6) *corresponds to the $\hat{H}_{i,i+1}$ component in* (3.3)*. For indices i and j where $|i-j| \geq 2$, $\hat{H}_{ij}$ is simply the zero matrix.*

3.1.1 Notations

We use I_n to denote the identity matrix of size $n \times n$. Additionally, we use $0_{m,n}$ to denote a zero matrix of size $m \times n$, and when the context is clear, we will omit m and n. Furthermore, let $\mathbb{S}^n$ be the space of real symmetric matrices of size $n \times n$, and let $\mathbb{S}_+^n$ be the PSD matrices in $\mathbb{S}^n$. Similarly, let $\mathbb{H}^n$ be the space of Hermitian matrices of size $n \times n$, and let $\mathbb{H}_+^n$ be the PSD matrices in $\mathbb{H}^n$. For any matrix X in $\mathbb{S}^n$ or $\mathbb{H}^n$, we may also use $X \succeq 0$ to denote that X is PSD.

When discussing a matrix A, the notation $A(p,q)$ refers to its (p,q)-th entry. Likewise, for a vector x, $x(p)$ denotes its p-th entry. When dealing with block matrices, we use A_{ij} to represent its (i,j)-th block. For a complex-valued matrix A, $\mathrm{Re}(A)$ and $\mathrm{Im}(A)$ denote its real and imaginary parts, respectively. In addition, for a linear operator $\mathcal{A}$ on matrices or vectors, we denote its adjoint by $\mathcal{A}^*$.

3.2 Preliminaries

We first rewrite the ground-state energy problem (3.1) for a spin-$\frac{1}{2}$ quantum many-body problem with N sites as an optimization problem over the density operator ρ, as shown in (3.2):

$$E_0 = \min_{\rho \in \mathbb{H}_+^{2^N}} \mathrm{Tr}(\hat{H}\rho), \quad \mathrm{Tr}(\rho) = 1. \tag{3.7}$$

Here, the space of Hermitian PSD operators on $Q \simeq \mathbb{C}^{2^N}$ can be identified with $\mathbb{H}_+^{2^N}$. As we consider Hamiltonians with only pairwise interactions in (3.3), the loss function in (3.7) is equal to

$$\sum_{1 \le i \le N} \mathrm{Tr}(\hat{H}_i \rho) + \sum_{1 \le i < j \le N} \mathrm{Tr}(\hat{H}_{ij} \rho). \tag{3.8}$$

In the context of quantum spin-$\frac{1}{2}$ systems, without sacrificing generality, we assume that each $\hat{H}_i$ and $\hat{H}_{ij}$ can be decomposed according to the following formula

$$\begin{aligned} \hat{H}_i &= \sum_{\alpha \in \{x,y,z\}} a_i^\alpha \sigma_i^\alpha, \quad 1 \le i \le N, \\ \hat{H}_{ij} &= \sum_{\alpha,\beta \in \{x,y,z\}} a_{ij}^{\alpha\beta} \sigma_i^\alpha \sigma_j^\beta, \quad 1 \le i < j \le N, \end{aligned}$$

for real constants $\{a_i^\alpha\}$ and $\{a_{ij}^{\alpha\beta}\}$. Consequently the loss function in (3.8) can be reformulated as:

$$\begin{aligned} \sum_{1 \le i \le N} \mathrm{Tr}(\hat{H}_i \rho) + \sum_{1 \le i < j \le N} \mathrm{Tr}(\hat{H}_{ij} \rho) = &\sum_{1 \le i \le N} \sum_{\alpha \in \{x,y,z\}} a_i^\alpha \mathrm{Tr}(\sigma_i^\alpha \rho) \\ &+ \sum_{1 \le i < j \le N} \sum_{\alpha,\beta \in \{x,y,z\}} a_{ij}^{\alpha\beta} \mathrm{Tr}(\sigma_i^\alpha \sigma_j^\beta \rho), \end{aligned} \tag{3.9}$$

which is a weighted sum of terms in $\{\mathrm{Tr}(\sigma_i^\alpha \rho)\}_{i,\alpha}$ and $\{\mathrm{Tr}(\sigma_i^\alpha \sigma_j^\beta \rho)\}_{i<j,\alpha,\beta}$.

Let

$$\mathbf{v}_N := \left(\sigma_1^x, \sigma_1^y, \sigma_1^z, \cdots, \sigma_N^x, \sigma_N^y, \sigma_N^z, I\right) \tag{3.10}$$

be a vector of operators of length $(3N+1)$. For any density operator $\rho \in \mathbb{H}_+^{2^N}$ and $\mathrm{Tr}(\rho) = 1$, define a square matrix M_ρ of size $(3N+1) \times (3N+1)$ by:

$$M_\rho(i,j) := \mathrm{Tr}(\mathbf{v}_N(i)\mathbf{v}_N(j)^*\rho), \quad 1 \le i,j \le 3N+1. \tag{3.11}$$

As all operators in $\mathbf{v}_N$ are Hermitian, it is apparent that the loss function in (3.9) is a linear function of the matrix M_ρ. For any Hamiltonian $\hat{H}$ with only pairwise interactions, let $J \in \mathbb{S}^{3N+1}$ be its corresponding real symmetric matrix such that the equality $\mathrm{Tr}(\hat{H}\rho) = \langle J, M_\rho \rangle$ holds for any density operator ρ, and thus we have a new formulation of the ground-state energy problem:

$$\begin{aligned} \underset{M}{\text{minimize}} \quad & \langle J, M \rangle, \\ \text{subject to} \quad & M = M_\rho \ \text{ for some } \ \rho \in \mathcal{D}(Q), \end{aligned} \tag{3.12}$$

with $Q \simeq \mathbb{C}^{2^N}$. However, the constraint in (3.12) on the moment matrix M remains overly restrictive and challenging to handle. Our objective is to retain a subset of the necessary conditions implied by the constraint in (3.12), and formulate a relaxed optimization problem that not only returns a reasonable lower bound for the exact ground-state energy, but also has a fast algorithm to solve it, even for large systems.

Remark 1. *In the analysis of a system with N sites, we frequently use Hermitian block matrices of size $(3N+1) \times (3N+1)$, such as the M_ρ matrix defined above. Let A be one such matrix. We partition A into 4 parts based on the locations of its entries:*

$$A := \begin{pmatrix} A^{(2)} & A^{(1)} \\ A^{(1)*} & A^{(0)} \end{pmatrix}, \tag{3.13}$$

where $A^{(2)} \in \mathbb{H}^{3N}$, $A^{(1)} \in \mathbb{C}^{3N}$ *and* $A^{(0)} \in \mathbb{R}$. *Furthermore, we adopt the notation* $A^{(2)}_{ij}$ *for* $1 \leq i, j \leq N$ *to represent the* (i,j)*-th* 3×3 *block of* $A^{(2)}$, *and* $A^{(1)}_i$ *for* $1 \leq i \leq N$ *to represent the* i*-th* 3×1 *block of* $A^{(1)}$. *To summary this notation, consider the following representation of* A*:*

$$A = \begin{pmatrix} A^{(2)}_{11} & A^{(2)}_{12} & \cdots & A^{(2)}_{1N} & A^{(1)}_1 \\ A^{(2)}_{21} & A^{(2)}_{22} & \cdots & A_{2N} & A^{(1)}_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A^{(2)}_{N1} & A^{(2)}_{N2} & \cdots & A^{(2)}_{NN} & A^{(1)}_N \\ A^{(1)^*}_1 & A^{(1)^*}_2 & \cdots & A^{(1)^*}_N & A^{(0)} \end{pmatrix}. \tag{3.14}$$

For the ground-state energy problem (3.12), if $M = M(\rho)$ *for some* $\rho \in \mathcal{D}(Q)$, *and we partition* M *in the form of (3.14), we can verify that for* $1 \leq i, j \leq N$*:*

$$M^{(2)}_{ij} = \begin{pmatrix} \mathrm{Tr}(\sigma^x_i \sigma^{x*}_j \rho) & \mathrm{Tr}(\sigma^x_i \sigma^{y*}_j \rho) & \mathrm{Tr}(\sigma^x_i \sigma^{z*}_j \rho) \\ \mathrm{Tr}(\sigma^y_i \sigma^{x*}_j \rho) & \mathrm{Tr}(\sigma^y_i \sigma^{y*}_j \rho) & \mathrm{Tr}(\sigma^y_i \sigma^{z*}_j \rho) \\ \mathrm{Tr}(\sigma^z_i \sigma^{x*}_j \rho) & \mathrm{Tr}(\sigma^z_i \sigma^{y*}_j \rho) & \mathrm{Tr}(\sigma^z_i \sigma^{z*}_j \rho) \end{pmatrix}, \tag{3.15}$$

and for $1 \leq i \leq N$*:*

$$M^{(1)}_i = \begin{pmatrix} \mathrm{Tr}(\sigma^x_i \rho) \\ \mathrm{Tr}(\sigma^y_i \rho) \\ \mathrm{Tr}(\sigma^z_i \rho) \end{pmatrix}, \tag{3.16}$$

and

$$M^{(0)} = \mathrm{Tr}(I\rho). \tag{3.17}$$

From (3.5), we can derive for any $1 \leq i < j \leq N$, *and for any* $\alpha, \beta \in \{x, y, z\}$, *the operator* $\sigma^\alpha_i \sigma^{\beta^*}_j$ *takes the following form:*

$$\sigma^\alpha_i \sigma^{\beta^*}_j = I_2^{\otimes(i-1)} \otimes \sigma^\alpha \otimes I_2^{\otimes(j-i-1)} \otimes \sigma^{\beta^*} \otimes I_2^{\otimes(N-j)}, \tag{3.18}$$

and for $i = j$, we have:

$$\sigma_i^{\alpha}\sigma_i^{\beta^*} = I_2^{\otimes(i-1)} \otimes (\sigma^{\alpha}\sigma^{\beta^*}) \otimes I_2^{\otimes(N-i)}. \tag{3.19}$$

3.3 Convex relaxation for the quantum spin problem

In this section, we propose a convex relaxation of the ground-state energy problem (3.12) based on only enforcing several necessary constraints. These necessary constraints comprise of various linear constraints on local blocks of the moment matrix M, along with a global PSD constraint. Our relaxed problem takes the following form:

$$(\mathbf{P}) \qquad \min \Big\{ \langle J, M\rangle \mid \mathcal{A}(M) = b,\ M \in \mathbb{H}_+^{3N+1} \Big\}. \tag{3.20}$$

Here, $\mathcal{A}$ is a linear map defined from $\mathbb{H}^{3N+1}$ to $\mathbb{S}^{3N+1}\times\mathbb{R}^{3N}$, and $b = (b_1, b_2) \in \mathbb{S}^{3N+1}\times\mathbb{R}^{3N}$ is given data. We will discuss these constraints in more details in section 3.3.1.

The dual problem of (**P**) takes the form of:

$$(\mathbf{D}) \qquad \min \Big\{ \langle -b, \Lambda\rangle \mid S + \mathcal{A}^*(\Lambda) = J,\ S \in \mathbb{H}_+^{3N+1},\ \Lambda = (\Lambda_1, \Lambda_2) \in \mathbb{S}^{3N+1} \times \mathbb{R}^{3N} \Big\}, \tag{3.21}$$

where $\mathcal{A}^*$ is the adjoint of $\mathcal{A}$. Here, for any two elements $x = (x_1, x_2)$ and $y = (y_1, y_2)$ in the product space $\mathbb{S}^{3N+1} \times \mathbb{R}^{3N}$, their inner product $\langle x, y\rangle$ is defined by:

$$\langle x, y\rangle := \langle x_1, y_1\rangle + \langle x_2, y_2\rangle.$$

In addition, for $x = (x_1, x_2) \in \mathbb{S}^{3N+1}\times\mathbb{R}^{3N}$, we occasionally use x/a for a scalar a to denote $(x_1/a, x_2/a)$.

3.3.1 Local linear constraints on the primal variable

The operators in (3.10) have some relations between them due to the properties of the Pauli matrices and I_2 in (3.4). We consider relating these operators with polynomial of operators up to degree 2:

$$p(\{\sigma_i^\alpha\}) = 0, \quad \deg(p) \leq 2,$$

which results in linear constraints on the moment matrix M in (3.11). These linear constraints on M are derived by taking the inner product of $p(\{\sigma_i^\alpha\})$ and a Hermitian PSD, unit-trace density operator ρ, and are summarized by the equality $\mathcal{A}(M) = b$ in (**P**). The correspondence between polynomial relations of operators and linear constraints on M is detailed in Table 3.1, in which the linear operator $\mathrm{odiag}(A) : \mathbb{H}^3 \to \mathbb{C}^3$, which extracts the off-diagonal entries of a 3×3 matrix, is defined as follows:

$$\mathrm{odiag}(A) := \begin{pmatrix} A(2,3) \\ A(3,1) \\ A(1,2) \end{pmatrix}. \tag{3.22}$$

Relations between the operators	Constraints on M
$\sigma_j^\alpha = \sigma_j^{\alpha *}$, $j \in [N]$, $\alpha \in \{x, y, z\}$	$\mathrm{Im}(M_j^{(1)}) = 0$, $j \in [N]$
$\sigma_j^\alpha \sigma_k^{\beta *} = \left(\sigma_j^\alpha \sigma_k^{\beta *}\right)^*$, $j, k \in [N]$, $j \neq k$, $\alpha, \beta \in \{x, y, z\}$	$\mathrm{Im}(M_{jk}^{(2)}) = 0$, $j, k \in [N], j \neq k$
$\sigma_j^\alpha \sigma_j^{\beta *} = i\sigma_j^\gamma$, $(\alpha, \beta, \gamma) = (x, y, z), (y, z, x), (z, x, y), j \in [N]$	$\mathrm{odiag}(M_{jj}^{(2)}) = iM_j^{(1)}, j \in [N]$
$\sigma_j^\alpha \sigma_j^{\alpha *} = I$, $\alpha, \beta \in \{x, y, z\}, \alpha \neq \beta, j \in [N]$	$\mathrm{diag}(M_{jj}^{(2)}) = M^{(0)} = 1, j \in [N]$

Table 3.1: Details of the local linear constraints on M.

As M is Hermitian, and $M_j^{(1)}$ is real-valued for all $j \in [N]$, the last two sets of constraints

imply $\mathrm{Re}(M_{jj}^{(2)}) = I_3$ and $M^{(0)} = 1$. Let $\mathcal{B}(A) : \mathbb{H}^3 \to \mathbb{R}^3$ be defined as:

$$\mathcal{B}(A) := \mathrm{Im}(\mathrm{odiag}(A)). \tag{3.23}$$

The linear constraints in Table 3.1 can be summarized by:

1. $\mathrm{Im}(M_j^{(1)}) = 0,\ j \in [N]$.
2. $\mathrm{Im}(M_{jk}^{(2)}) = 0,\ j,k \in [N],\ j \neq k$.
3. $\mathrm{Re}(M_{jj}^{(2)}) = I_3,\ j \in [N]$.
4. $M^{(0)} = 1$.
5. $\mathcal{B}(M_{jj}^{(2)}) = M_j^{(1)},\ j \in [N]$.

3.3.2 Global positive semidefinite constraint on the primal variable

In addition to the local linear constraints on the moment matrix M, we can also derive a global PSD constraint on M as follows. Let $a := (a_1^x, a_1^y, a_1^z, \cdots, a_N^x, a_N^y, a_N^z, a_0)^*$ be any complex-valued vector of size $(3N+1)$. Consider an operator $\hat{O} : Q \to Q$ (not necessarily Hermitian) defined by

$$\hat{O} := \sum_{i=1}^{N} \sum_{\alpha \in \{x,y,z\}} a_i^\alpha \sigma_i^\alpha + a_0 I.$$

Now $\hat{O}\hat{O}^* \succeq 0$, so

$$\mathrm{Tr}(\hat{O}\hat{O}^* \rho) \geq 0.$$

Expanding the above, we have

$$\begin{aligned} 0 &\leq \mathrm{Tr}(\hat{O}\hat{O}^*\rho) \\ &= \mathrm{Tr}\Big[\big(\sum_i \sum_{\alpha \in \{x,y,z\}} a_i^\alpha \sigma_i^\alpha + a_0 I\big)\big(\sum_j \sum_{\beta \in \{x,y,z\}} a_j^{\beta^*} \sigma_j^{\beta^*} + a_0^* I^*\big)\rho\Big] \end{aligned}$$

$$
\begin{aligned}
&= \sum_{i,j,\alpha,\beta} a_i^{\alpha}\ \mathrm{Tr}(\sigma_i^{\alpha}\sigma_j^{\beta^*}\rho)\ a_j^{\beta^*} + \sum_{i,\alpha} a_i^{\alpha}\ \mathrm{Tr}(\sigma_i^{\alpha}\rho)\ a_0{}^* + \\
&\quad \sum_{j,\beta} a_0\ \mathrm{Tr}(\sigma_j^{\beta^*}\rho) a_j^{\beta^*} + a_0(a_0)^*. \\
&= a^* M a.
\end{aligned}
$$

The second equality holds because $I^* = I$, and $\mathrm{Tr}(I\rho) = 1$, as ρ has unit trace. Since the choice of the vector a is completely arbitrary, it follows that M must be PSD.

3.4 Standard augmented Lagrangian method

A common approach to solve either the primal problem (3.20) or the dual problem (3.21) is the Augmented Lagrangian method [7, 54, 55]. In the case of the dual problem (3.21), we first form the augmented Lagrangian function $L_\sigma\big(S, \Lambda, M\big)$ with a penalty parameter $\sigma > 0$, for dual variables $S \in \mathbb{H}_+^{3N+1}$, $\Lambda \in \mathbb{S}^{3N+1} \times \mathbb{R}^{3N}$, and primal variable $M \in \mathbb{H}^{3N+1}$:

$$
L_\sigma(S, \Lambda, M) := \langle -b, \Lambda \rangle + \frac{\sigma}{2} \| S + \mathcal{A}^*(\Lambda) - J + M/\sigma \|_F^2 - \frac{\|M\|_F^2}{2\sigma}. \tag{3.24}
$$

Then, we iterate Algorithm 6 until convergence. In the following subsection, we provide a detailed explanation of how to perform the minimization sub-problem in Algorithm 6. It is important to note that all linear constraints in (**P**) are local in nature (see the linear constraints below equation (3.23)), making it straightforward to choose an initial M that satisfies all the linear constraints.

Algorithm 6 ALM for the dual problem

Require: S, Λ, and M satisfying linear constraints in (**P**), and penalty parameter $\sigma > 0$

1: **while** not converged **do**
2: $\quad S, \Lambda \leftarrow \mathrm{argmin}_{S \succeq 0, \Lambda}\ L_\sigma(S, \Lambda, M)$
3: $\quad M \leftarrow M + \sigma\big(S + \mathcal{A}^*(\Lambda) - J\big)$
4: **end while**

3.4.1 Solving the joint optimization problem

In this section, we provide details on how to solve the sub-problem involving both S and Λ in Algorithm 6. Since the sub-problem is unconstrained with respect to Λ, we first eliminate Λ from the sub-problem using the first-order optimality condition of Λ, and express the optimal Λ in terms of S and M. We are then left with a minimization sub-problem that involves only one variable, S, and the optimization of this sub-problem is the topic of the next section.

For any fixed primal variable M and dual variable S, the first order condition for the loss function (3.24) requires the optimal dual variable Λ to take the following form:

$$\sigma\mathcal{A}\mathcal{A}^*(\Lambda) + \sigma\mathcal{A}(S - J + M/\sigma) - b = 0,$$

and consequently the optimal Λ as a function of S and M is:

$$\Lambda(S, M) := \big(\mathcal{A}\mathcal{A}^*\big)^{-1}\big(\mathcal{A}(J - S - M/\sigma) + b/\sigma\big). \tag{3.25}$$

By substituting the optimal Λ in (3.25) into the loss function in (3.24), we arrive at a loss function which involves only one PSD variable S, and a fixed primal variable M:

$$f(S, M) = \langle -b, \Lambda(S, M)\rangle + \frac{\sigma}{2}\|S + \mathcal{A}^*(\Lambda(S, M)) - J + M/\sigma\|_F^2. \tag{3.26}$$

For the particular dual problem (**D**) we have, we can show that $f(S, M)$ takes the following equivalent form, up to a constant independent of S:

$$\begin{aligned} f(S; M) = \mathrm{Tr}(S) &+ \frac{\sigma}{2}\sum_{i\neq j}\|\mathrm{Re}(S_{ij}^{(2)}) + \mathrm{Re}(M_{ij}^{(2)})/\sigma - J_{ij}^{(2)}\|_F^2 \\ &+ \frac{\sigma}{2}\sum_i \|\mathrm{Re}\big(J_i^{(1)} - S_i^{(1)} - M_i^{(1)}/\sigma\big) + \mathcal{B}\big(J_{ii}^{(2)} - S_{ii}^{(2)} - M_{ii}^{(2)}/\sigma\big)\|_F^2, \end{aligned} \tag{3.27}$$

where the linear operator $\mathcal{B}$ is defined in (3.23), and the details of the calculation are de-

ferred to Appendix A. Thus, solving the sub-problem in Algorithm 6 amounts to solving the following optimization problem involving one PSD variable S:

$$\underset{S \succeq 0}{\text{minimize}} \quad f(S, M). \tag{3.28}$$

In Algorithm 6, it is worth noting that the primal variable M always satisfies the linear constraint in (**P**) during the updates. This arises from the property that ALM always gives dual-feasible variables [17] ((**P**) is the dual problem of (**D**)).

A significant challenge in directly solving (3.28) arises from the PSD constraint imposed on S. While it's feasible to solve a standard SDP problem using an ADMM-type method [54, 55], this approach involves computing the projection onto the PSD cone, and the computational complexity of this projection is cubic, making it impractical for large-scale problems.

In [12], the authors propose a solution by introducing a change of variables for the PSD variable in the form of $S = RR^*$. This strategy effectively circumvented the difficult PSD constraint in the optimization problem, converting it into an unconstrained optimization problem. Moreover, when low-rank solutions of the SDP problem exist, the number of columns of R is chosen minimally, enabling the development of an efficient algorithm using the limited-memory BFGS algorithm. However, experiments conducted using the CVX package [26] to directly solve either the primal problem (3.20) or the dual problem (3.21) for smaller systems indicate a linear increase in the rank in N of both the primal PSD variable M and the dual PSD variable S. Hence, employing a vanilla low-rank decomposition of M or S to solve either the primal or dual problem via the limited-memory BFGS algorithm is unlikely to yield substantial reductions in the computation time.

Fortunately, the dual PSD variable S exhibits a hierarchically low-rank structure that can be exploited to reduce the computational complexity of the ALM algorithm. Exploiting such a structure is the focus of subsequent sections.

3.5 Hierarchical dual positive semidefinite variable

In this section, we propose a structure for dual variable S in Algorithm 6, that allows us to perform the ALM updates with a reduced time complexity. For a 1-D TFI model (3.6) with small system size $N = 128$ and field strength parameter $h = 1$, we solve (3.21) using the CVX package. We show a heatmap of the PSD variable S in Figure 3.1. We can see from the plot that even though S is not low-rank, it is nearly zero except on a few diagonals near the main diagonal. This observation inspired us to represent the dual PSD variable S using a hierarchical low-rank matrix [8], resulting in an algorithm with $O(N^2)$ scaling of the per-iteration time complexity. In this section, for simplicity, we assume that the system size N is a power of 2.

3.5.1 Approximating the dual positive semidefinite variable with a hierarchical matrix

Let S denote the solution obtained from the dual problem (3.21) with N sites. We first look at the $S^{(2)}$ block of the matrix S (as defined in (3.14)). While our objective is to use a hierarchical matrix to represent $S^{(2)}$, we must also ensure that $S^{(2)}$ remains a PSD matrix. To this end, we use m levels of hierarchies to characterize $S^{(2)}$. For the i-th level,

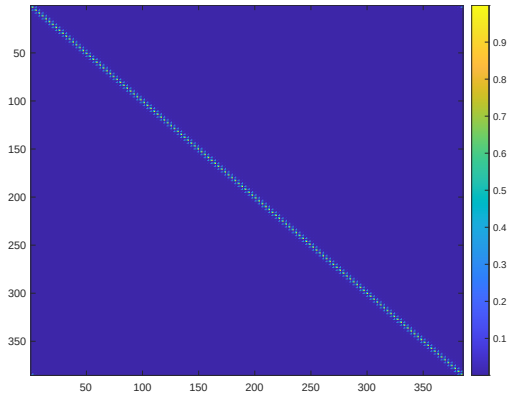

Figure 3.1: Heatmap of the dual PSD variable, system size $N = 128$.

one defines a block diagonal matrix having n_i diagonal blocks, where each block is of size $c_i \times c_i$. Furthermore, we want the block diagonal matrix to be PSD. Therefore, for the i-th level, we form a matrix:

$$\begin{pmatrix} y_1^{(i)}(y_1^{(i)})^* & & & \\ & y_2^{(i)}(y_2^{(i)})^* & & \\ & & \ddots & \\ & & & y_{n_i}^{(i)}(y_{n_i}^{(i)})^* \end{pmatrix} \in \mathbb{H}_+^{3N}, \quad y_j^{(i)} \in \mathbb{C}^{c_i \times r_i}, \quad 1 \leq j \leq n_i.$$

This naturally requires $n_i c_i = 3N$, since the size of the matrix matrix $S^{(2)}$ is $3N \times 3N$. Then, as an approximation to $S^{(2)}$, we define

$$\begin{aligned} \mathcal{H}^{(2)}(y) := y^{(1)}(y^{(1)})^* + & \begin{pmatrix} y_1^{(2)}(y_1^{(2)})^* & 0 \\ 0 & y_2^{(2)}(y_2^{(2)})^* \end{pmatrix} \\ & + \cdots + \begin{pmatrix} y_1^{(m)}(y_1^{(m)})^* & 0 & \cdots & 0 \\ 0 & y_2^{(m)}(y_2^{(m)})^* & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & y_{n_m}^{(m)}(y_{n_m}^{(m)})^* \end{pmatrix}, \end{aligned} \tag{3.29}$$

with

$$y := \left\{y^{(1)}, y^{(2)}, \cdots, y^{(m)}\right\}, \quad \text{and} \quad y^{(i)} := \begin{pmatrix} y_1^{(i)} \\ y_2^{(i)} \\ \vdots \\ y_{n_i}^{(i)} \end{pmatrix} \in \mathbb{C}^{3N \times r_i}.$$

Our proposal involves representing $S^{(2)}$ with $\mathcal{H}^{(2)}(y)$, where the number of levels m, and the number of columns for each level r_i for $1 \leq i \leq m$ are determined based on the desired accuracy of the algorithm.

To approximate the full matrix S, which has one extra row and extra column compare to $S^{(2)}$, we just pad $\mathcal{H}^{(2)}(y)$ with an extra row and column of zeros, plus a low-rank matrix:

$$\mathcal{H}(y,z) := \begin{pmatrix} \mathcal{H}^{(2)}(y) & 0_{3N\times 1} \\ 0_{1\times 3N} & 0_{1\times 1} \end{pmatrix} + zz^* \approx S, \tag{3.30}$$

where $y := \left\{y^{(1)}, \cdots, y^{(m)}\right\}$ is a set of matrices with $y^{(i)} \in \mathbb{C}^{3N\times r_i}$, and $z \in \mathbb{C}^{(3N+1)\times r_z}$. Let $Z := zz^* \in \mathbb{H}_+^{3N+1}$. We partition the low-rank matrix zz^* into 4 blocks based on the locations of its entries as in (3.13), and we use $(zz^*)^{(2)}, (zz^*)^{(1)}$, $\left((zz^*)^{(1)}\right)^*$ and $(zz^*)^{(0)}$ to denote its blocks. Additionally, we use $\mathcal{H}_{ij}^{(2)}(y)$ and $(zz^*)_{ij}^{(2)}$ for $1 \le i,j \le N$ to denote the (i,j)-th 3×3 block of $\mathcal{H}^{(2)}(y)$ and $(zz^*)^{(2)}$, and use $(zz^*)_i^{(1)}$ for $1 \le i \le N$ to denote the i-th 3×1 block of $(zz^*)^{(1)}$.

Remark 2. *Much like the low-rank representation of PSD matrices discussed in chapter 2, the expressive power of the hierarchical representation of PSD matrices also depends on the parameters chosen for the representation. In chapter 2, we represent a PSD matrix $X \in \mathbb{S}_+^n$ via the low-rank factorization of the form $X = xx^T$, with $x \in \mathbb{R}^{n\times r}$, and the parameter r controls the expressive power of this representation. If r is set to be equal to n, then all PSD matrices can be written as xx^T for some $x \in \mathbb{R}^{n\times r}$, and as r decreases, the expressive power of the low-rank representation also decreases.*

For the hierarchical representation of PSD matrices, it is easy to see that any $S \in \mathbb{H}_+^{3N+1}$ can be represented by one level of hierarchy, where the number of columns r in this level is equal to $3N$, plus one low-rank matrix. This representation is essentially the same as the low-rank representation of S with $r = 3N+1$.

On the other hand, neither representations naturally offers an advantage in terms of computational speed. If we use the number of parameters used in the representation as an approximate measure of computational speed, the computational complexity is approximately $O(nr)$ for the low-rank representation. A significant improvement in computational speed

can only be achieved if we choose a rank r such that the representation is accurate enough, while $r \ll n$.

Similarly, for a hierarchical representation of PSD matrices with m levels, where the number of columns in each level and in the rank-correction term are $r_1, r_2, \cdots, r_m, r_z$, the total number of variables needed for this hierarchical representation is $(3N\sum_{i=1}^{m} r_i + (3N + 1)r_z)$. An improvement in computational speed is possible only if we can find $m, r_1, \cdots, r_m, r_z$ such that the hierarchical representation provides an accurate enough approximation to the PSD matrix of interest, while $\sum_{i=1}^{m} r_i + r_z \ll N$.

The next section presents a numerical study to investigate the existence of such a representation. In the following sections, we assume that each n_i is a power of two, specifically $n_i = 2^{i-1}$.

3.5.2 Existence of a data-sparse hierarchical matrix representation for the dual positive semidefinite variable

To investigate the existence of a data-sparse and relatively accurate hierarchical matrix representation for the dual PSD variable S, we conducted an experiment as follows. We first solved the dual problem (3.21) with the Hamiltonian specified in (3.6), with system sizes $N = 64, 128, 256$, and a field strength parameter $h = 1$. For these problem sizes, we solved for S in the full PSD cone rather easily with an accuracy of 10^{-4}. Then, to see how well these PSD S can be approximated by a hierarchical structure, we fitted the resulting PSD dual variable S with the structure outlined in (3.30). Let S^* denote the solution to the dual problem (3.21). Besides, let y denote a set of matrices and z be a complex-valued matrix that contain the information for a hierarchical matrix of the corresponding size, with number of levels $m = 3, 4, 5$ respectively for $N = 64, 128, 256$. The number of columns r_i in each level and r_z were fixed to be 20 for all system sizes. We solved the following optimization

problem for system size $N = 64, 128, 256$ for the variables y and z:

$$\min_{y,z} \quad \mathrm{Err}_S^2 := \frac{\|\mathcal{H}(y,z) - S^*\|_F^2}{\|S^*\|_F^2}, \tag{3.31}$$

by running the limited-memory BFGS algorithm provided in the MANOPT toolbox [10] for 1000 iterations. The approximation errors are given in Table 3.2. From the table, we can see that even with fixed $r_i, r_z \leq 20$, we obtain a similar accuracy for different system sizes. We therefore assume one can use a fixed rank approximation in the hierarchical matrix even for large system sizes.

N	64	128	256
Err_S	$3.0968e-04$	$3.7045e-04$	$4.6712e-04$

Table 3.2: Relative errors of the fitted dual PSD variables.

3.5.3 Update rule with a hierarchically structured matrix variable

By substituting the PSD variable S in (3.24) with a data-sparse hierarchical PSD representation, it seems like we can eliminate the challenging PSD constraint on S, significantly reducing the per-iteration computational costs. With this hierarchical representation of S, when performing Algorithm 6 where one needs to minimize the augmented Lagrangian function $L_\sigma(S, \Lambda; M)$ in (3.24), we now solve

$$\underset{y,z}{\text{minimize}} \quad L_\sigma(\mathcal{H}(y,z), \Lambda(\mathcal{H}(y,z), M); M) \tag{3.32}$$

where the hierarchical matrix $\mathcal{H}(y,z)$ is defined in (3.30) and the optimal $\Lambda(\mathcal{H}(y,z), M)$ is defined in (3.25). Here, y is a collection of matrices $y := \left\{y^{(1)}, y^{(2)}, \cdots, y^{(m)}\right\}$ for $y^{(i)} \in \mathbb{C}^{3N \times r_i}$ and $z \in \mathbb{C}^{(3N+1)\times r_z}$, with pre-specified number of levels m and number of columns $r_1, \cdots, r_m, r_z$. The resulting algorithm is outlined in Algorithm 7.

Algorithm 7 Pseudo code for ALM for the dual problem with a hierarchical dual PSD variable

Require: $y := \{y^{(1)}, y^{(2)}, \cdots, y^{(m)}\}$, z, M satisfying the linear constraint in (**P**), and penalty parameter $\sigma > 0$

1: **while** not converged **do**
2: $y, z \leftarrow \mathrm{argmin}_{y,z} L_\sigma(\mathcal{H}(y,z), \Lambda(\mathcal{H}(y,z), M); M)$
3: $S \leftarrow \mathcal{H}(y,z)$
4: Compute $\Lambda(S, M)$ according to formula (3.25)
5: $M \leftarrow M + \sigma\big(S + \mathcal{A}^*(\Lambda(S,M)) - J\big)$
6: **end while**

We now conduct a complexity analysis of Algorithm 7, examining its computational scaling step by step. Throughout this subsection, we approximate the PSD dual variable $S \in \mathbb{H}_+^{3N+1}$ using a hierarchical structure constructed from $y := \{y^{(1)}, \cdots, y^{(m)}\}$ and z, where each $y^{(i)} \in \mathbb{C}^{3N \times r}$ and $z \in \mathbb{C}^{(3N+1)\times r}$. We assume the number of levels m scales logarithmically with the system size N, and the number of columns in each level r is kept a constant.

We start by analyzing the complexity of approximately solving the sub-problem in Algorithm 7 at line 2. To tackle this sub-problem, we run the limited-memory BFGS algorithm available in the MANOPT toolbox for a fixed number of steps. The main computational burden for the limited-memory BFGS algorithm is the evaluation of the loss function, which is equal to $f(\mathcal{H}(y,z); M)$ in (3.27), up to a constant independent of y and z. Therefore, it suffices to determine the computational complexity of evaluating $f(\mathcal{H}(y,z); M)$ as N increases. We first rewrite the objective $f(\mathcal{H}(y,z); M)$ in (3.32) as the sum of two parts, $f(\mathcal{H}(y,z); M) = f_1(y,z;M) + f_2(y,z;M)$, with

$$f_1(y,z;M) := \frac{\sigma}{2}\|\mathrm{Re}(\mathcal{H}^{(2)}(y)) + \mathrm{Re}((zz^*)^{(2)}) + \mathrm{Re}(M^{(2)})/\sigma - J^{(2)}\|_F^2, \tag{3.33}$$

and

$$
\begin{aligned}
f_2(y,z;M) :&= f(\mathcal{H}(y,z);M) - f_1(y,z;M) \\
&= \mathrm{Tr}(\mathcal{H}(y,z)) - \frac{\sigma}{2}\sum_i \|\mathrm{Re}(\mathcal{H}_{ii}^{(2)}(y)) + \mathrm{Re}((zz^*)_{ii}^{(2)}) + \mathrm{Re}(M_{ii}^{(2)})/\sigma - J_{ii}^{(2)}\|_F^2 \\
&\quad + \frac{\sigma}{2}\sum_i \|\mathrm{Re}\big(J_i^{(1)} - (zz^*)_i^{(1)} - M_i^{(1)}/\sigma\big) + \\
&\qquad\qquad \mathcal{B}\big(J_{ii}^{(2)} - \mathcal{H}_{ii}^{(2)}(y) - (zz^*)_{ii}^{(2)} - M_{ii}^{(2)}/\sigma\big)\|_F^2.
\end{aligned}
\tag{3.34}
$$

The second equality holds because of the definition of $f(\mathcal{H}(y,z);M)$ in (3.27), and the representation of $\mathcal{H}(y)$ and zz^* as block matrices is outlined in the comment that follows (3.30).

By splitting the loss function f, we reveal that the computational bottleneck arises primarily from computing inner products of matrices within f_1. As for the residual portion of the loss in f_2, it is apparent from its definition that f_2 depends solely on a subset of $O(N)$ elements within $\mathcal{H}^{(2)}(y)$ and zz^*. Moreover, computing any element in $\mathcal{H}^{(2)}(y)$ or zz^* from y and z incurs a computational cost at most $O(mr)$, and consequently, the computational complexity for evaluating f_2 is $O(Nmr)$, which scales approximately linearly with the system size.

On the other hand, evaluating f_1 involves three terms: $\langle \mathrm{Re}(\mathcal{H}^{(2)}(y) + (zz^*)^{(2)}), J^{(2)}\rangle$, $\langle \mathrm{Re}(\mathcal{H}^{(2)}(y) + (zz^*)^{(2)}), \mathrm{Re}(M^{(2)})\rangle$ and $\langle \mathrm{Re}(\mathcal{H}^{(2)}(y) + (zz^*)^{(2)}), \mathrm{Re}(\mathcal{H}^{(2)}(y) + (zz^*)^{(2)})\rangle$, up to a constant independent of y and z. For many well-known spin-$\frac{1}{2}$ quantum models, the J matrix associated with its Hamiltonian typically contains only $O(N)$ many non-zero entries, making the computational cost for the first term scale linearly with the system size N. For the third term, noting that the real part of a matrix A is equal to the average of A and its

complex conjugate $\bar{A}$, we get:

$$\mathrm{Re}(\mathcal{H}^{(2)}(y)) = \frac{1}{2}\big(\mathcal{H}^{(2)}(y) + \overline{\mathcal{H}^{(2)}(y)}\big),$$
$$\mathrm{Re}((zz^*)^{(2)}) = \frac{1}{2}\big((zz^*)^{(2)} + \overline{(zz^*)^{(2)}}\big).$$

Using Proposition 1, the third term's complexity is $O(Nm^2r^2)$. It turns out that the most time-consuming aspect in Algorithm 7 at line 2 arises from computing the second term. Without additional assumptions on M, the time complexity for the inner product between a dense matrix $\mathrm{Re}(M^{(2)})$ and a hierarchical matrix is $O(N^2)$. Moreover, if the number of levels m in the hierarchy scales logarithmically with the system size N, and the number of columns r within each level remains constant, the complexity of all other terms in f is at most $O(N \log^2 N)$.

Additionally, lines 3-5 in Algorithm 7, which update the primal variable M, also have an $O(N^2)$ scaling. To reduce the time complexity for each ALM iteration, our next step is to represent the primal variable M with a hierarchical structure, aiming for an algorithm with almost linear per-iteration scaling.

Proposition 1. *Suppose we have two sets of matrices*

$$a := \big\{a^{(1)}, a^{(2)}, \cdots, a^{(m)}\big\}, \quad a^{(i)} \in \mathbb{C}^{3N\times r}, \quad 1 \le i \le m,$$
$$b := \big\{b^{(1)}, b^{(2)}, \cdots, b^{(m)}\big\}, \quad b^{(j)} \in \mathbb{C}^{3N\times r}, \quad 1 \le j \le m,$$

and two rank correction matrices

$$c \in \mathbb{C}^{(3N+1)\times r}, \quad d \in \mathbb{C}^{(3N+1)\times r}.$$

The complexity of evaluating the inner product

$$\left\langle \mathcal{H}^{(2)}(a) + (cc^*)^{(2)}, \mathcal{H}^{(2)}(b) + (dd^*)^{(2)} \right\rangle \tag{3.35}$$

is $O(Nm^2r^2)$.

Proof. The inner product in (3.35) can be split into three groups: the inner product between two hierarchical matrices, the inner product between one hierarchical matrix and one low-rank matrix, and the inner product between two low-rank matrices. It is apparent that the inner product between low-rank matrices of the form $\langle (cc^*)^{(2)}, (dd^*)^{(2)} \rangle$ can be computed in $O(Nr^2)$ time. For the inner product between two hierarchical matrices, it can be rewritten using the following form from the definition of the hierarchical matrix in (3.29):

$$\begin{aligned} \left\langle \mathcal{H}^{(2)}(a), \mathcal{H}^{(2)}(b) \right\rangle = \sum_{i=1}^{m} \sum_{j=1}^{m} \langle &\text{a block-diagonal matrix with block size equal to } c_i \times c_i, \\ &\text{a block-diagonal matrix with block size equal to } c_j \times c_j \rangle, \end{aligned} \tag{3.36}$$

and the inner product of block-diagonal matrices can be written as n_k many inner products of low-rank matrices of size $c_k \times c_k$, whose ranks are equal to r. Here $k = \max(i, j)$. Consequently, the complexity of the inner product of two block-diagonal matrices in (3.36) is $O(n_k c_k r^2) = O(Nr^2)$, and the overall complexity of (3.36) is $O(Nm^2r^2)$.

The inner product between one hierarchical matrix and one low-rank matrix can be analyzed similarly as for (3.36), and we can show that its complexity is $O(Nmr^2)$. In conclusion, the complexity of (3.35) is $O(Nm^2r^2)$.

□

3.6 Hierarchical primal positive semidefinite variable

As analyzed in section 3.5.3, Algorithm 7 requires repeatedly computing the inner product of $\mathrm{Re}(M^{(2)})$ and hierarchical matrices in the limited-memory BFGS algorithm, which is computationally intensive due to the lack of structure in M. We propose replacing the direct update rule in lines 3-5 of Algorithm 7 with a projection step that compresses M, reducing the computational cost of this inner product. This results in an algorithm with nearly linear per-iteration cost.

Before discussing how to form a compressed representation for the primal variable, we will revisit the exactly updated primal variable in Algorithm 7, which is the solution to the following problem:

$$\begin{aligned} \mathrm{argmin}_{\tilde{M}} \quad & \|\tilde{M} - \big(M + \sigma(S + \mathcal{A}^*(\Lambda(S, M)) - J)\big)\|_F^2, \\ \text{s.t.} \quad & \mathcal{A}(\tilde{M}) = b. \end{aligned} \tag{3.37}$$

Here, S is represented hierarchically as $S = \mathcal{H}(y, z)$, with the optimal $\Lambda(S, M)$ defined in (3.25). The linear constraint $\mathcal{A}(\tilde{M}) = b$ includes all constraints from section 3.3.1. Although this constraint in (3.37) may seem redundant since the exactly updated primal variable is always feasible, it becomes crucial when replacing $\tilde{M}$ with a compressed representation, as the constraint won't be naturally satisfied.

Next, we examine the linear constraint $\mathcal{A}(\tilde{M}) = b$. Define the function $g : \mathbb{H}^{3N+1} \to \mathbb{H}^{3N+1}$ by:

$$g(A) = I_{3N+1} + \mathrm{Re}\left(\begin{pmatrix} 0 & A_{12}^{(2)} & \cdots & A_{1N}^{(2)} & A_1^{(1)} \\ A_{21}^{(2)} & 0 & \cdots & A_{2N}^{(2)} & A_2^{(1)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A_{N1}^{(2)} & A_{N2}^{(2)} & \cdots & 0 & A_N^{(1)} \\ (A_1^{(1)})^* & (A_2^{(1)})^* & \cdots & (A_N^{(1)})^* & 0 \end{pmatrix}\right)$$

$$+ 2 \begin{pmatrix} \mathcal{B}^*(\mathrm{Re}(A_1^{(1)})) & & & \\ & \ddots & & \\ & & \mathcal{B}^*(\mathrm{Re}(A_N^{(1)})) & \\ & & & 0 \end{pmatrix}, \quad (3.38)$$

where the operator $\mathcal{B}$ is defined in (3.23). We can show that $g(A)$ is always primal-feasible for any $A \in \mathbb{H}^{3N+1}$ by checking the constraints in section 3.3.1. Additionally, for any primal-feasible $\tilde{M} \in \mathbb{H}^{3N+1}$, there exists an $A \in \mathbb{H}^{3N+1}$ such that $g(A) = \tilde{M}$. This reparameterization of primal-feasible matrices allows us to rewrite (3.37) in an equivalent form:

$$\mathrm{argmin}_{A \in \mathbb{H}^{3N+1}} \quad \|g(A) - \big(M + \sigma(S + \mathcal{A}^*(\Lambda(S, M)) - J)\big)\|_F^2. \quad (3.39)$$

Our final step is to approximate $A \in \mathbb{H}^{3N+1}$ with a hierarchical matrix to reduce the cost of the loss function evaluation:

$$\begin{aligned} \mathrm{argmin}_{x,v,A} \quad & \|g(A) - \big(M + \sigma(S + \mathcal{A}^*(\Lambda(S, M)) - J)\big)\|_F^2, \\ \text{s.t.} \quad & A = \mathcal{H}(x, v), \end{aligned} \quad (3.40)$$

where $x := \{x^{(i)} \in \mathbb{C}^{3N \times r_i}\}_{1 \le i \le m}$ and $v \in \mathbb{C}^{(3N+1) \times r_v}$ for some pre-specified number of levels m, and number of columns $r_1, \cdots, r_m, r_v$. In other words, we propose a hierarchical representation for the parameterization of any primal-feasible variable M. This allows efficient evaluation of the loss function in (3.40). If M is represented as $M = g(\mathcal{H}(x_0, v_0))$ for some x_0 and v_0, and $S = \mathcal{H}(y, z)$, then we can show that $g(\mathcal{H}(x, v))$ and $M + \sigma(S + \mathcal{A}^*(\Lambda(S, M)) - J)$ are sums of hierarchical and sparse matrices (see Appendix A). Thus, the main computational task in evaluating the loss function (3.40) is the evaluation of inner products between structured matrices, which can be computed efficiently as discussed in section 3.5.3.

We now introduce Algorithm 8, which utilizes hierarchical representation for both the primal and the dual PSD variables.

Algorithm 8 Pseudo code for ALM for the dual problem with two hierarchical PSD variables

Require: $t = 0$, $x_0 := \{x_0^{(1)}, \cdots, x_0^{(m_1)}\}$ and v_0 for the primal variable $M_0 := g(\mathcal{H}(x_0, v_0))$, $y_0 := \{y_0^{(1)}, \cdots, y_0^{(m_2)}\}$ and z_0 for the dual variable $S_0 = \mathcal{H}(y_0, z_0)$, and penalty parameter $\sigma > 0$

1: **while** not converged **do**

2: $y_{t+1}, z_{t+1} \leftarrow \text{argmin}_{y,z} L_\sigma(\mathcal{H}(y, z), \Lambda(\mathcal{H}(y, z), M_t); M_t)$

3: $S_{t+1} \leftarrow \mathcal{H}(y_{t+1}, z_{t+1})$

4: $x_{t+1}, v_{t+1}, A_{t+1} \leftarrow \text{argmin}_{x,v,A=\mathcal{H}(x,v)} \|g(A) - \big(M_t + \sigma(S_{t+1} + \mathcal{A}^*(\Lambda(S_{t+1}, M_t)) - J)\big)\|_F^2$

5: $M_{t+1} \leftarrow g(A_{t+1})$

6: $t \leftarrow t + 1$

7: **end while**

We assume $x_0^{(i)} \in \mathbb{C}^{3N \times r_{1i}}$ for $1 \leq i \leq m_1$, $y_0^{(j)} \in \mathbb{C}^{3N \times r_{2j}}$ for $1 \leq j \leq m_2$, $v_0 \in \mathbb{C}^{(3N+1) \times r_v}$ and $z_0 \in \mathbb{C}^{(3N+1) \times r_z}$. The parameters $r_{11}, \cdots, r_{1m_1}, r_{21}, \cdots, r_{2m_2}, r_v, r_z$ are chosen based on algorithm's required accuracy. We highlight that it is not necessary to explicitly form the matrices A, M or S during the execution of Algorithm 8. Instead, the loss functions in steps 2 and 4 can be efficiently evaluated using the compressed representation of these matrices.

3.6.1 Complexity analysis

In this section, we analyze the computational complexity of Algorithm 8, following the approach in section 3.5.3 for Algorithm 7. We assume the hierarchical representation of the primal and dual PSD variables has the same number of levels m, with a constant number of columns r for each level, and the rank correction matrices z and v also have r columns.

The sub-problems in step 2 and step 4 of Algorithm 8 are solved approximately by running the limited-memory BFGS algorithm in MANOPT for a fixed number of steps. Step 2 in Algorithm 8 resembles step 2 in Algorithm 7, but the expensive inner products between dense and hierarchical matrices are replaced with inner products between hierarchical matrices. Thus, the time complexity for step 2 is $O(Nm^2r^2)$. Similarly, step 4 involves evaluating inner products between structured matrices, making its time complexity also $O(Nm^2r^2)$.

In summary, if the number of levels m in the hierarchy scales logarithmically with the system size N, and the number of columns for each level r remains constant, the per-iteration time complexity for Algorithm 8 is $O(N \log^2 N)$.

3.6.2 Existence of a data-sparse hierarchical matrix representation for the primal positive semidefinite variable

In this section, we investigate the existence of a data-sparse and relatively accurate hierarchical representation of M, using the same method as in Section 3.5.2 for the dual PSD variable S. Instead of fitting the dual PSD variable S, we fitted the primal PSD variable M resulting from solving (**D**) with the Hamiltonian specified in (3.6), for system sizes $N = 64, 128, 256$, an external magnetic field strength parameter $h = 1$ and an accuracy of 10^{-4}. Let M^* be the approximate solution. Let x, v be complex-valued parameters that parameterize a hierarchical matrix of the corresponding size, with the number of levels $m = 3, 4, 5$ respectively for $N = 64, 128, 256$. The number of columns for each level was fixed at 20 for all system sizes. We solved the following optimization problem with system sizes $N = 64, 128, 256$ for the variables x and t:

$$\min_{x,v} \quad \mathrm{Err}_M^2 := \frac{\|\mathcal{H}(x, v) - M^*\|_F^2}{\|M^*\|_F^2}, \tag{3.41}$$

by running the limited-memory BFGS algorithm provided in the MANOPT toolbox [10] for 1000 iterations. The approximation errors are presented in Table 3.3. The experiment results indicate that as the system size increases, the relative error of the fitted primal PSD variable also tends to increase. This likely contributes to the failure of Algorithm 8 for large system sizes, such as $N = 4096$ (see Section 3.7 for further details on the experiments).

N	64	128	256
Err_M	$4.8663e-04$	0.0020	0.0119

Table 3.3: Relative errors of the fitted primal PSD variables.

3.7 Numerical experiments

In this section, we present numerical experiments for the 1-D transverse field Ising (TFI) model using Algorithms 7 and 8, with system sizes $N \in \{64, 128, 256, 512, 1024, 2048, 4096\}$. The penalty parameter is initialized at $\sigma = 0.1$, and is adjusted dynamically [60] based on primal and dual feasibility to speed up the convergence of the ALM algorithm. For both algorithms, the number of levels m in the hierarchy is set to be $m = 3, 4, \cdots, 9$ for $N = 64, 128, \cdots, 4096$, and the number of columns for all levels and the rank correction matrices is set to be $r = 20$. In Algorithm 7, y, z and M are initialized from the standard normal distribution, with M being primal-feasible. This is achieved by initializing $\text{Re}(M_{ij}^{(2)})$ for $i \neq j$ and $\text{Re}(M^{(1)})$ randomly, with the rest of M determined by the constraint $\mathcal{A}(M) = b$. In Algorithm 8, x_0, v_0, y_0, and z_0 are randomly initialized from the standard normal distribution.

Throughout the updates of Algorithm 7 and 8, we evaluate the accuracy of approximate solutions by monitoring the relative primal feasibility, the relative dual feasibility, and the relative duality gap.

To evaluate the feasibility of a candidate primal variable M, we must ensure that M

is PSD and satisfies the linear constraint in (**P**). In Algorithm 7, M is directly initialized and updated, while in Algorithm 8, M is maintained using a compressed representation as specified in steps 4-5. Since M is guaranteed to satisfy the linear constraint in (**P**), we only monitor its relative PSD-ness using the following measure:

$$\eta_P := \frac{\max(0, -\lambda_{\min})}{1 + \max(0, \lambda_{\max})},$$

where $\lambda_{\min}$ and $\lambda_{\max}$ are the smallest and largest eigenvalues of M.

For the dual problem (**D**) and a candidate dual variable S, the PSD constraint for S is automatically satisfied because S is maintained as a hierarchical matrix in both algorithms. For the linear constraint involving both S and Λ, given that we only keep the information of S in these two algorithms, we need to find its corresponding dual variable $\Lambda(S)$. This is achieved by solving for Λ from the linear equality in (**D**):

$$\Lambda(S) = (\mathcal{A}\mathcal{A}^*)^{-1}\mathcal{A}(J - S),$$

and we monitor the relative dual feasibility using the following measure:

$$\eta_D := \frac{\|S + \mathcal{A}^*(\Lambda(S)) - J\|_F}{1 + \|J\|_F}.$$

Finally, we monitor the relative duality gap by:

$$\eta_g := \frac{|\text{primal objective} - \text{dual objective}|}{1 + |\text{primal objective}| + |\text{dual objective}|}.$$

We terminate the algorithm when $\eta := \max(\eta_P, \eta_D, \eta_g) \leq 10^{-3}$, or when the ALM algorithm has run for 1500 iterations. It's important to highlight that we exploit the hierarchical structure present in the PSD primal and dual variables to efficiently evaluate these convergence metrics.

We examine the ground-state energy recovery for the TFI model on an $N \times 1$ lattice, for system sizes $N \in \{64, 128, 256, 512, 1024, 2048, 4096\}$ and external magnetic field strength parameter $h \in \{0.1, 1, 1.5\}$. Let E_0 denote the true ground-state energy and $\tilde{E}_0$ the lower bound of the ground-state energy obtained from (**P**). The ground-state energy of the 1-D transverse field Ising model can be computed analytically [52]. The relative error is defined as:

$$\text{Err}_{\text{rel}} := \frac{E_0 - \tilde{E}_0}{|E_0|}.$$

The relative errors for Algorithms 7 and 8 are given in Tables 3.4 and 3.5. Additionally, we present the evolution of our convergence metrics as a function of the ALM iteration number in Figures 3.2 and 3.3, focusing on the 1-D TFI model with a fixed external magnetic field strength parameter $h = 1$ and various system sizes. Alongside the relative primal and dual feasibility measures and the relative duality gap, we also track the per-site primal objective change between subsequent iterations. Let p_i be value of the primal objective function at ALM iteration number i, the per-site primal objective change at iteration i is defined as

$$\text{Energy-Change}_i := |\frac{p_i}{N} - \frac{p_{i+1}}{N}|, \tag{3.42}$$

where N is the number of sites in the system.

All convergence metrics are transformed using a base-10 logarithm function. Within 1500 ALM iterations, all metrics drop below 10^{-3} for experiments with the external magnetic field strength parameter $h = 1$.

	N=64	N=128	N=256	N=512	N=1024	N=2048	N=4096
$h = 0.5$	1.12%	1.14%	1.17%	1.23%	1.24%	1.24%	1.22%
$h = 1$	2.73%	2.75%	2.76%	2.76%	2.76%	2.77%	2.79%
$h = 1.5$	0.69%	0.69%	0.69%	0.7%	0.7%	0.7%	0.71%

Table 3.4: Relative errors of the recovered energy from Algorithm 7.

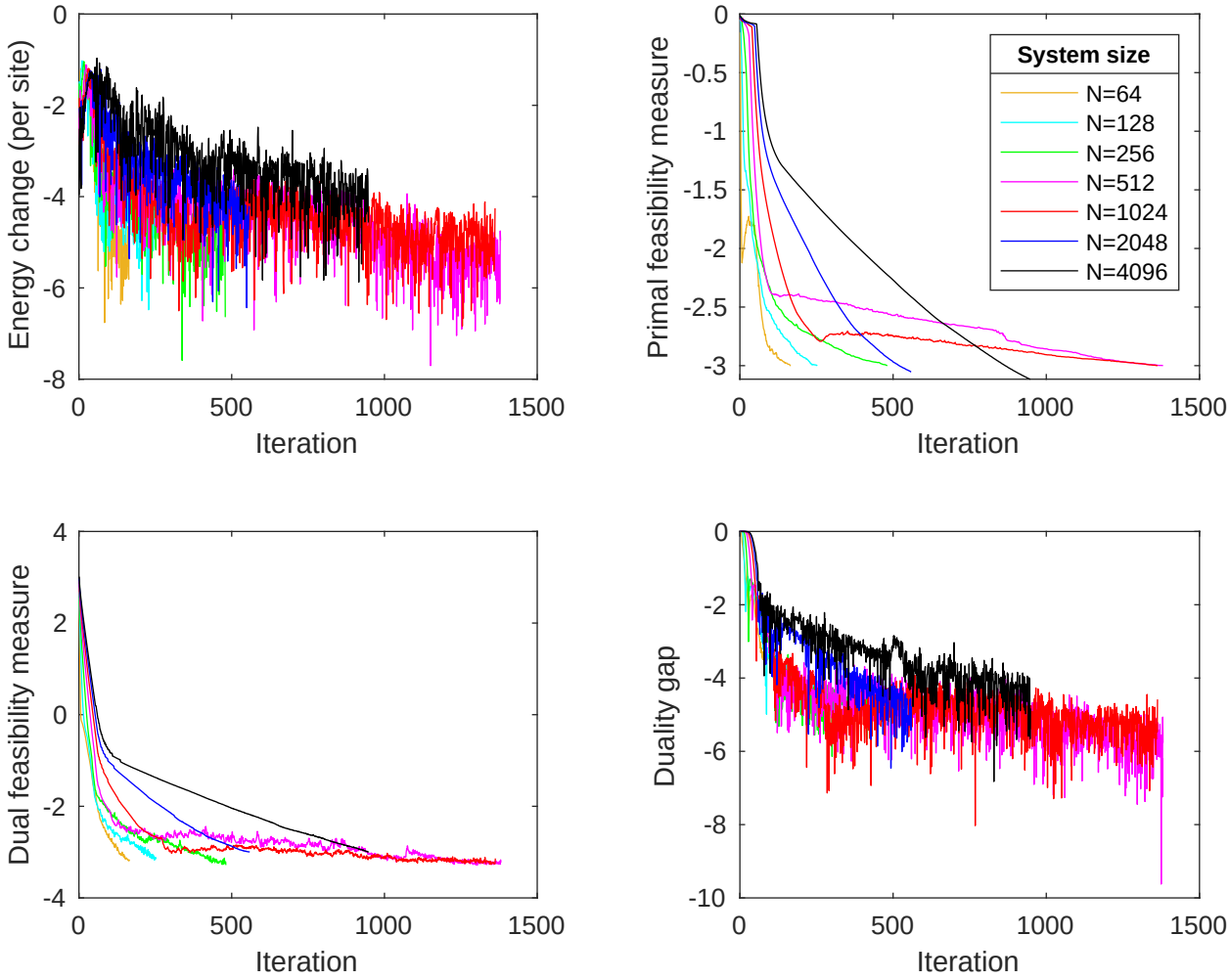

Figure 3.2: Convergence metrics for Algorithm 7. The primal feasibility measure (η_P), dual feasibility measure (η_D), duality gap (η_g) and per-site energy change in equation (3.42) are all transformed using the $\log_{10}$ function.

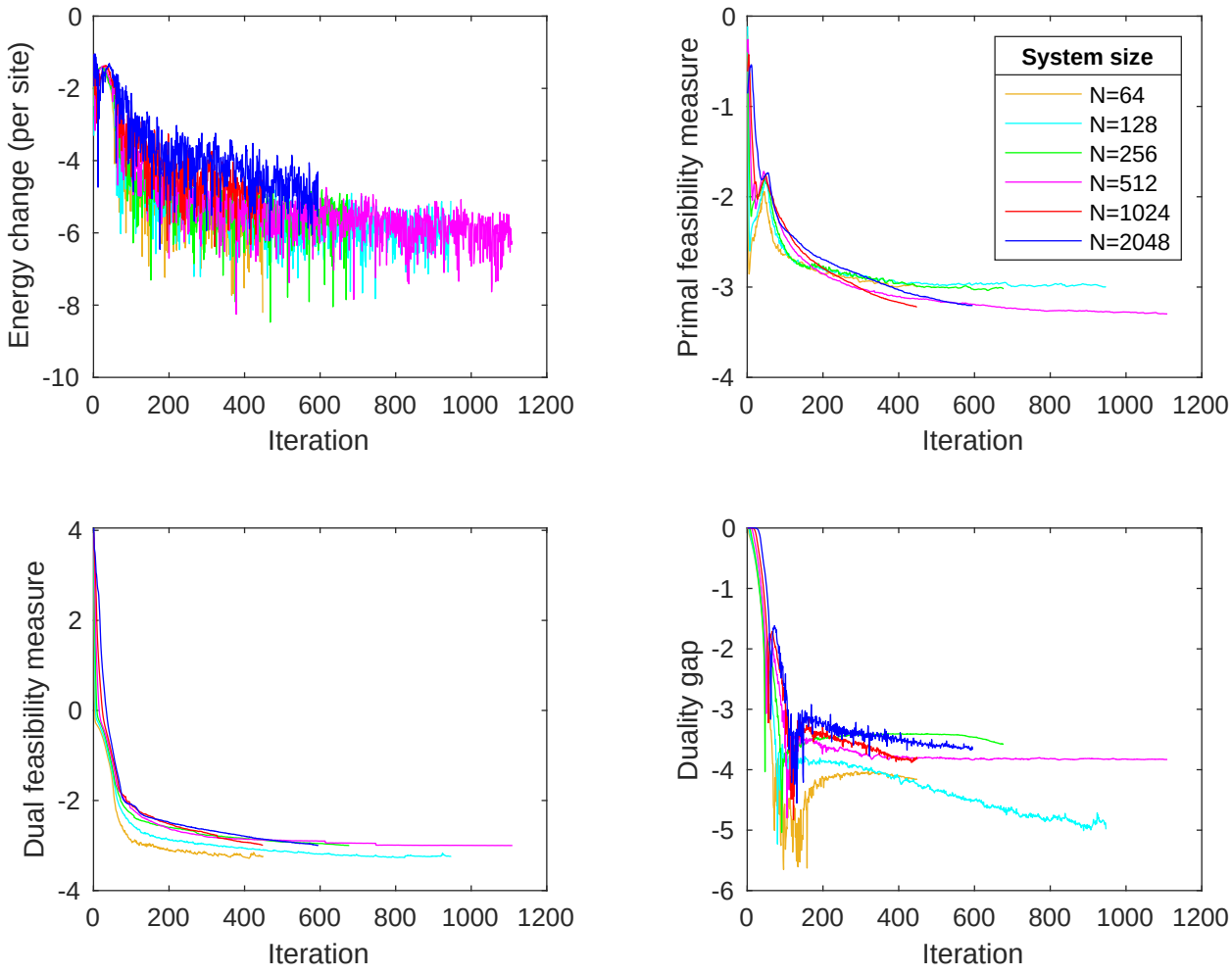

Figure 3.3: Convergence metrics for Algorithm 8. The primal feasibility measure (η_P), dual feasibility measure (η_D), duality gap (η_g) and per-site energy change in equation (3.42) are all transformed using the $\log_{10}$ function.

	N=64	N=128	N=256	N=512	N=1024	N=2048
$h = 0.5$	1.08%	1.19%	1.21%	1.23%	1.24%	1.23%
$h = 1$	2.75%	2.75%	2.70%	2.73%	2.73%	2.72%
$h = 1.5$	0.69%	0.68%	0.68%	0.69%	0.65%	0.64%

Table 3.5: Relative errors of the recovered energy from Algorithm 8.

3.8 Summary

In this chapter, we explored a novel convex relaxation framework for determining the ground-state energy of the quantum many-body problem. Mathematically, the ground-state energy problem is an eigenvalue problem, but the matrix size increases exponentially with the size of the physical system. By formulating the ground-state energy problem with density operators and applying only a subset of necessary constraints derived from the properties of Pauli matrices, we propose a relaxation of the original problem via semi-definite programming (SDP), which can be solved in polynomial time and provides a reasonable lower bound for the ground-state energy. Additionally, we identified a hierarchical structure in both the positive semi-definite (PSD) primal and dual variables, allowing us to circumvent the expensive step of performing a projection onto the PSD cone, thereby reducing the per-iteration complexity of the ALM-type algorithm from cubic to quadratic or almost linear.

The relaxed problem provides only a lower bound for the exact ground-state energy. To evaluate the effectiveness of our approach, we compare the recovered lower bound with the true ground-state energy for the 1-D transverse field Ising model. Notably, for the most challenging case of $h = 1$, where the system undergoes a quantum phase transition, our algorithm still produces a reasonable lower bound. Furthermore, our algorithm can handle systems consisting of up to 4096 qubits, whereas previous work on the variational embedding method, such as [37, 30], which employs more accurate yet more expensive constraints, can only manage systems of a few dozen qubits without leveraging the periodicity of the model.

CHAPTER 4

CONCLUSION

In this thesis, we discussed two problems in convex relaxation. For both problems, we explored the computation of a specific SDP relaxation to determine the lowest possible energy of a system of many interacting bodies. These SDP relaxations can be solved in polynomial time and provide reasonable lower bounds for the lowest energy. Additionally, by analyzing the resulting PSD variables from these relaxed problems, we identified structures in these matrices, which we subsequently exploited to develop algorithms with reduced per-iteration complexity.

To further evaluate our algorithms, we tested them on systems comprising 10,000 and 4,000 particles, respectively. We demonstrated that both algorithms are capable of solving large systems and returning relatively accurate lower bounds for the true ground-state energy. However, in most of the cases we tested, the relaxation was not exact. Furthermore, due to the issue of multiple solutions, extracting a single solution from a linear combination of solutions is not currently possible, which will be a direction for future studies.

APPENDIX A

DETAILS OF THE DERIVATION FOR FORMULA (3.27)

Firstly, we describe how to merge all the linear constraints from Table 3.1 into an equality constraint $\mathcal{A}(M) = b$ for the primal problem (**P**). Next, we show that $\mathcal{A}\mathcal{A}^*$ is the identity operator, simplifying the optimal Λ defined in (3.25). Finally, we substitute this simplified Λ into (3.26) and prove that it is equal to (3.27) up to a constant independent of S.

To begin with, we explicitly write down the adjoint of the operator $\mathcal{B}$ defined in (3.23), which will be used later:

$$\mathcal{B}^*(\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}) = \frac{i}{2}\begin{pmatrix} 0 & x_3 & -x_2 \\ -x_3 & 0 & x_1 \\ x_2 & -x_1 & 0 \end{pmatrix}, \quad \mathcal{B}^* : \mathbb{R}^3 \to \mathbb{H}^3. \tag{A.1}$$

It is worth noting that the real part of the output of $\mathcal{B}^*$ is a zero matrix.

A.1 Detailed breakdown of the linear constraint $\mathcal{A}(M) = b$

We partition all linear constraints on M from Table 3.1 into two groups. The first group is described by the linear function $\mathcal{A}^{(1)}$, defined as:

$$\mathcal{A}^{(1)}(M) := \begin{pmatrix} \mathrm{Re}(M_{11}^{(2)}) & \mathrm{Im}(M_{12}^{(2)}) & \cdots & \mathrm{Im}(M_{1N}^{(2)}) & \mathrm{Im}(M_1^{(1)}) \\ -\mathrm{Im}(M_{21}^{(2)}) & \mathrm{Re}(M_{22}^{(2)}) & \cdots & \mathrm{Im}(M_{2N}^{(2)}) & \mathrm{Im}(M_2^{(1)}) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -\mathrm{Im}(M_{N1}^{(2)}) & -\mathrm{Im}(M_{N2}^{(2)}) & \cdots & \mathrm{Re}(M_{NN}^{(2)}) & \mathrm{Im}(M_N^{(1)}) \\ -\mathrm{Im}((M_1^{(1)})^*) & -\mathrm{Im}((M_2^{(1)})^*) & \cdots & -\mathrm{Im}((M_N^{(1)})^*) & \mathrm{Re}(M^{(0)}) \end{pmatrix} = I_{3N+1}, \tag{A.2}$$

with $\mathcal{A}^{(1)} : \mathbb{H}^{3N+1} \to \mathbb{S}^{3N+1}$. The second group is described by the linear function $\mathcal{A}^{(2)}$, defined as:

$$\mathcal{A}^{(2)}(M) := \begin{pmatrix} \mathrm{Re}(M_1^{(1)}) - \mathcal{B}(M_{11}^{(2)}) \\ \mathrm{Re}(M_2^{(1)}) - \mathcal{B}(M_{22}^{(2)}) \\ \vdots \\ \mathrm{Re}(M_N^{(1)}) - \mathcal{B}(M_{NN}^{(2)}) \end{pmatrix} = 0_{3N\times 1}, \tag{A.3}$$

with $\mathcal{A}^{(2)} : \mathbb{H}^{3N+1} \to \mathbb{R}^{3N}$, and $\mathcal{B}$ defined in (3.23). Thus, we can merge the constraints in (A.2) and (A.3) into $\mathcal{A}(M) = b$, where

$$\mathcal{A}(M) := (\mathcal{A}^{(1)}(M), \mathcal{A}^{(2)}(M)), \quad \mathcal{A} : \mathbb{H}^{3N+1} \to \mathbb{S}^{3N+1} \times \mathbb{R}^{3N}, \tag{A.4}$$

and $b := (b_1, b_2) \in \mathbb{S}^{3N+1} \times \mathbb{R}^{3N}$, with

$$b_1 := I_{3N+1} \text{ and } b_2 := 0_{3N\times 1}. \tag{A.5}$$

A.2 Proof that $\mathcal{A}\mathcal{A}^*$ is the identity operator

In this section, we prove that $\mathcal{A}\mathcal{A}^*$ is the identity operator on $\mathbb{S}^{3N+1} \times \mathbb{R}^{3N}$. Let (X, Y) be any element in this space. We partition X into blocks as in (3.14) and Y into N blocks of size 3×1, with Y_i as its i-th block. Our goal is to show that $(X, Y) = (\mathcal{A}\mathcal{A}^*)((X, Y))$.

To this end, first let $Z := \mathcal{A}^*((X, Y)) = \mathcal{A}_1^*(X) + \mathcal{A}_2^*(Y)$. We can derive $\mathcal{A}_1^*$ and $\mathcal{A}_2^*$ from

(A.2) and (A.3), respectively, and show that Z takes the following form:

$$Z = \begin{pmatrix} X_{11}^{(2)} - \mathcal{B}^*(Y_1) & iX_{12}^{(2)} & \cdots & iX_{1N}^{(2)} & Y_1/2 + iX_1^{(1)} \\ -iX_{21}^{(2)} & X_{22}^{(2)} - \mathcal{B}^*(Y_2) & \cdots & iX_{2N}^{(2)} & Y_2/2 + iX_2^{(1)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -iX_{N1}^{(2)} & -iX_{N2}^{(2)} & \cdots & X_{NN}^{(2)} - \mathcal{B}^*(Y_N) & Y_N/2 + iX_N^{(1)} \\ Y_1/2 - iX_1^{(1)} & Y_2/2 - iX_2^{(1)} & \cdots & Y_N/2 - iX_N^{(1)} & X^{(0)} \end{pmatrix}. \tag{A.6}$$

Let

$$(\tilde{X}, \tilde{Y}) := (\mathcal{A}\mathcal{A}^*)((X, Y)) = \mathcal{A}(Z) = (\mathcal{A}^{(1)}(Z), \mathcal{A}^{(2)}(Z)).$$

Since the real part of the output of $\mathcal{B}^*$ is zero, it follows from (A.2) that $\tilde{X} = \mathcal{A}^{(1)}(Z) = X$. Additionally, we have:

$$\tilde{Y} = \mathcal{A}^{(2)}(Z) = \begin{pmatrix} Y_1/2 - \mathcal{B}(X_{11}^{(2)} - \mathcal{B}^*(Y_1)) \\ Y_2/2 - \mathcal{B}(X_{22}^{(2)} - \mathcal{B}^*(Y_2)) \\ \vdots \\ Y_N/2 - \mathcal{B}(X_{NN}^{(2)} - \mathcal{B}^*(Y_N)) \end{pmatrix} = \begin{pmatrix} Y_1/2 + \mathcal{B}(\mathcal{B}^*(Y_1)) \\ Y_2/2 + \mathcal{B}(\mathcal{B}^*(Y_2)) \\ \vdots \\ Y_N/2 + \mathcal{B}(\mathcal{B}^*(Y_N)) \end{pmatrix} = Y. \tag{A.7}$$

The second equality holds due to the definition of $\mathcal{A}^{(2)}$ in (A.3), and the last two equalities hold due to the definitions of $\mathcal{B}$ and $\mathcal{B}^*$ in (3.23) and (A.1). Consequently, $(\tilde{X}, \tilde{Y}) = (X, Y)$, and $\mathcal{A}\mathcal{A}^*$ is the identity operator on $\mathbb{S}^{3N+1} \times \mathbb{R}^{3N}$.

A.3 Equivalence of the loss functions (3.26) and (3.27)

In this section, we show the equivalence of (3.26) and (3.27) by substituting the optimal Λ from (3.25) into (3.26). Since both loss functions are used to update S, we just need to establish their equivalence up to a constant independent of S, and we will omit this condition when the context is clear.

For any primal and dual variables $M, S \in \mathbb{H}^{3N+1}$, let

$$R(S, M) := J - S - M/\sigma \in \mathbb{H}^{3N+1}. \tag{A.8}$$

Since $\mathcal{A}\mathcal{A}^*$ is the identity operator, the optimal Λ from (3.25) can be equivalently expressed as:

$$\Lambda(S, M) = \mathcal{A}(R(S, M)) + b/\sigma = \big(\mathcal{A}^{(1)}(R(S, M)) + b_1/\sigma,\ \mathcal{A}^{(2)}(R(S, M)) + b_2/\sigma\big), \tag{A.9}$$

where $b = (b_1, b_2)$ is defined in (A.5).

We first prove the first components in (3.26) and (3.27), $\langle -b, \Lambda(S, M)\rangle$ and $\mathrm{Tr}(S)$, are equal up to a constant independent of S. Then, we show that the remaining parts are also equal. Let $\Lambda(S, M) = (\Lambda_1(S, M), \Lambda_2(S, M))$. We have:

$$\begin{aligned}
\langle -b, \Lambda(S, M)\rangle &= \langle -b_1, \Lambda_1(S, M)\rangle + \langle -b_2, \Lambda_2(S, M)\rangle \\
&= \langle -I_{3N+1}, \Lambda_1(S, M)\rangle \\
&= -\mathrm{Tr}(\Lambda_1(S, M)) \\
&= -\mathrm{Tr}(\mathcal{A}^{(1)}(R(S, M))) - \mathrm{Tr}(b_1)/\sigma \\
&= -\mathrm{Tr}(R(S, M)) - \mathrm{Tr}(b_1)/\sigma \\
&= \mathrm{Tr}(S) + \mathrm{Tr}(M)/\sigma - \mathrm{Tr}(J) - \mathrm{Tr}(b_1)/\sigma.
\end{aligned} \tag{A.10}$$

The first and second equalities hold due to the definition of b in (A.5). The fourth equality follows from (A.9), and the fifth and sixth equalities result from the definitions of $\mathcal{A}^{(1)}$ and $R(S, M)$ in (A.2) and (A.8).

To show the remaining parts in (3.26) and (3.27) are equal, we rewrite the remaining

part in (3.26), ignoring the constant $\sigma/2$, as:

$$\begin{aligned}\|S+\mathcal{A}^*(\Lambda(S,M))-J+M/\sigma\|_F^2 &= \|\mathcal{A}^*(\Lambda(S,M))-R(S,M)\|_F^2\\ &= \|(\mathcal{A}^*\mathcal{A})(R(S,M))+\mathcal{A}^*(b/\sigma)-R(S,M)\|_F^2\\ &= \|(\mathcal{A}^*\mathcal{A})(R(S,M))-R(S,M)+I_{3N+1}/\sigma\|_F^2\\ &= \|(\mathcal{A}^{(1)})^*\mathcal{A}^{(1)}(R(S,M))+(\mathcal{A}^{(2)})^*\mathcal{A}^{(2)}(R(S,M))\\ &\qquad -R(S,M)+I_{3N+1}/\sigma\|_F^2.\end{aligned} \tag{A.11}$$

Let $P(S,M) = (\mathcal{A}^{(1)})^*\mathcal{A}^{(1)}(R(S,M)) + (\mathcal{A}^{(2)})^*\mathcal{A}^{(2)}(R(S,M))$. We partition $P(S,M)$ into blocks as in (3.14). From the definitions of $\mathcal{A}^{(1)}$ and $\mathcal{A}^{(2)}$ in (A.2) and (A.3), the blocks of $P(S,M)$ take the following form:

$$\begin{cases} P_{ii}^{(2)}(S,M) = \mathrm{Re}(R_{ii}^{(2)}(S,M)) - \mathcal{B}^*\big(\mathrm{Re}(R_i^{(1)}) - \mathcal{B}(R_{ii}^{(2)})\big), \quad 1\le i\le N,\\ P_{ij}^{(2)}(S,M) = i\,\mathrm{Im}(R_{ij}^{(2)}(S,M)), \quad i\neq j,\\ P_i^{(1)}(S,M) = \frac{1}{2}(\mathrm{Re}(R_i^{(1)}) - \mathcal{B}(R_{ii}^{(2)})) + i\,\mathrm{Im}(R_i^{(1)}), \quad 1\le i\le N,\\ P^{(0)}(S,M) = \mathrm{Re}(R^{(0)}(S,M)). \end{cases} \tag{A.12}$$

To simplify notations, the dependence of P and R on S and M is omitted in the analysis below, and (A.11) can be reformulated as follows:

$$\begin{aligned}\|P-R+I_{3N+1}/\sigma\|_F^2 = &\sum_{i=1}^N \|P_{ii}^{(2)}-R_{ii}^{(2)}+I_3/\sigma\|_F^2 + \sum_{i\neq j}\|P_{ij}^{(2)}-R_{ij}^{(2)}\|_F^2\\ &+2\sum_{i=1}^N\|P_i^{(1)}-R_i^{(1)}\|_F^2 + (P^{(0)}-R^{(0)}+\sigma^{-1})^2,\end{aligned} \tag{A.13}$$

and these four terms will be analyzed one by one.

Since the real part of the output of $\mathcal{B}^*$ is a zero matrix, for $1 \leq i \leq N$ we have:

$$\begin{aligned}\|P_{ii}^{(2)} - R_{ii}^{(2)} + I_3/\sigma\|_F^2 &= \|\mathrm{Im}(R_{ii}^{(2)}) + \mathrm{Im}\big(\mathcal{B}^*(\mathrm{Re}(R_i^{(1)}) - \mathcal{B}(R_{ii}^{(2)}))\big)\|_F^2 + \|I_3/\sigma\|_F^2 \\ &= \|\mathrm{Im}(R_{ii}^{(2)}) - \mathrm{Im}(\mathcal{B}^*\mathcal{B}(R_{ii}^{(2)})) + \mathrm{Im}(\mathcal{B}^*(\mathrm{Re}(R_i^{(1)})))\|_F^2 + 3/\sigma^2 \\ &= \frac{1}{2}\|\mathcal{B}(R_{ii}^{(2)}) + \mathrm{Re}(R_i^{(1)})\|_F^2 + 3/\sigma^2,\end{aligned} \tag{A.14}$$

where the last equality follows from the definitions of $\mathcal{B}$ and $\mathcal{B}^*$ in (3.23) and (A.1). For $i \neq j$ we have:

$$\|P_{ij}^{(2)} - R_{ij}^{(2)}\|_F^2 = \|i\,\mathrm{Im}(R_{ij}^{(2)}) - R_{ij}^{(2)}\|_F^2 = \|\mathrm{Re}(R_{ij}^{(2)})\|_F^2. \tag{A.15}$$

For $1 \leq i \leq N$ we have:

$$\begin{aligned}\|P_i^{(1)} - R_i^{(1)}\|_F^2 &= \|\frac{1}{2}(\mathrm{Re}(R_i^{(1)}) - \mathcal{B}(R_{ii}^{(2)})) + i\,\mathrm{Im}(R_i^{(1)}) - R_i^{(1)}\|_F^2 \\ &= \|\frac{1}{2}(\mathrm{Re}(R_i^{(1)}) - \mathcal{B}(R_{ii}^{(2)})) - \mathrm{Re}(R_i^{(1)})\|_F^2 \\ &= \frac{1}{4}\|\mathcal{B}(R_{ii}^{(2)}) + \mathrm{Re}(R_i^{(1)})\|_F^2,\end{aligned} \tag{A.16}$$

and lastly, we have:

$$(P^{(0)} - R^{(0)} + \sigma^{-1})^2 = (\mathrm{Re}(R^{(0)}) - R^{(0)} + \sigma^{-1})^2 = 1/\sigma^2, \tag{A.17}$$

since $R^{(0)}$ lies on the diagonal of a Hermitian matrix and is real-valued. Combining the results from (A.14)-(A.17), we conclude that:

$$\begin{aligned}\|S + \mathcal{A}^*(\Lambda(S,M)) - J + M/\sigma\|_F^2 = &\sum_{i \neq j} \|\mathrm{Re}(R_{ij}^{(2)}(S,M))\|_F^2 + \frac{3N+1}{\sigma^2} \\ &+ \sum_{i=1}^{N} \|\mathcal{B}(R_{ii}^{(2)}(S,M)) + \mathrm{Re}(R_i^{(1)}(S,M))\|_F^2.\end{aligned} \tag{A.18}$$

Given the definition of $R(S, M)$ in (A.8), we confirm from (A.10) and (A.18) that (3.26) and (3.27) are equal up to a constant independent of S.

REFERENCES

[1] A. Altland and B. D. Simons. *Condensed matter field theory*. Cambridge university press, 2010.

[2] G. An. A note on the cluster variation method. *Journal of Statistical Physics*, 52(3-4):727–734, 1988.

[3] J. S. M. Anderson, M. Nakata, R. Igarashi, K. Fujisawa, and M. Yamashita. The second-order reduced density matrix method and the two-dimensional hubbard model. *Computational and Theoretical Chemistry*, 1003:22–27, 2013.

[4] F. Barahona. On the computational complexity of ising spin glass models. *J. Phys. A: Math. Gen.*, 15:3241, 1982.

[5] A. I. Barvinok. Problems of distance geometry and convex properties of quadratic maps. *Discrete Computational Geometry*, 13:189, 1995.

[6] A. Beck. *First-Order Methods in Optimization*. SIAM-Society for Industrial and Applied Mathematics, 2017.

[7] D. P. Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.

[8] S. Börm, L. Grasedyck, and W. Hackbusch. Introduction to hierarchical matrices with applications. *Eng Anal Bound Elem*, 27(5):405–422, 2003.

[9] N. Boumal. A riemannian low-rank method for optimization over semidefinite matrices with block-diagonal constraints. *arXiv:1506.00575*, 2016.

[10] N. Boumal, B. Mishra, P.-A. Absil, and R. Sepulchre. Manopt, a matrix toolbox for optimization on manifolds. *J. Mach. Learn. Res.*, 15(42):1455–1459, 2014.

[11] E. Brézin. *Introduction to Statistical Field Theory*. Cambridge University Press, 2014.

[12] S. Burer and R. Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Math. Program., Ser. B*, 95:329–357, 2003.

[13] E. Cancès, G. Stoltz, and M. Lewin. The electronic ground-state energy problem: A new reduced density matrix approach. *J. Chem. Phys.*, 125(6):064101, 2006.

[14] D. Ceperley. Ground state of the fermion one-component plasma: A monte carlo study in two and three dimensions. *Phys. Rev. B*, 18:3126, 1978.

[15] D. M. Ceperley and B. J. Alder. Ground state of the electron gas by a stochastic method. *Phys. Rev. Lett.*, 45:566, 1980.

[16] Y. Chen and Y. Khoo. Combining monte-carlo and tensor-network methods for partial differential equations via sketching. *arXiv:2305.17884*, 2023.

[17] S. Cipolla and J. Gondzio. Admm and inexact alm: the qp case. 2020.

[18] B. A. Cipra. An introduction to the ising model. *The American Mathematical Monthly*, 94(10):937–959, 1987.

[19] A. E. DePrince and D. A. Mazziotti. Exploiting the spatial locality of electron correlation within the parametric two-electron reduced-density-matrix method. *J. Chem. Phys.*, 132:034110, 2010.

[20] Y. I. Dublenych. Ground states of the lattice-gas model on the triangular lattice with nearest- and next-nearest-neighbor pairwise interactions and with three-particle interaction: Full-dimensional ground states. *Phys. Rev. E*, 84:011106, 2011.

[21] Y. I. Dublenych. Ground states of the lattice-gas model on the triangular lattice with nearest- and next-nearest-neighbor pairwise interactions and with three-particle interaction: Ground states at boundaries of full-dimensional regions. *Phys. Rev. E*, 84:061102, 2011.

[22] Y. I. Dublenych. Ground states of the ising model on the shastry-sutherland lattice and the origin of the fractional magnetization plateaus in rare-earth-metal tetraborides. *Phys. Rev. Lett.*, 109:167202, 2012.

[23] A. Finel and F. Ducastelle. On the phase diagram of the fcc ising model with antiferromagnetic first-neighbour interactions. *EPL*, 1:135, 1986.

[24] A. Georges, G. Kotliar, W. Krauth, and M. J. Rozenberg. Dynamical mean-field theory of strongly correlated fermion systems and the limit of infinite dimensions. *Rev. Mod. Phys.*, 68:13, 1996.

[25] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42:1115, 1995.

[26] M. Grant and S. Boyd. *CVX: Matlab software for disciplined convex programming, version 2.0 beta*, 2013.

[27] A. Griewank and A. Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Society for Industrial and Applied Mathematics, 2008.

[28] Q. Huang, Y. Chen, and L. Guibas. Scalable semidefinite relaxation for maximum a posterior estimation. In *International Conference on Machine Learning*, 2014.

[29] M. Kaburagi and J. Kanamori. Ground state structure of triangular lattice gas model with up to 3rd neighbor interactions. *J. Phys. Soc. Jpn.*, 44:718, 1978.

[30] Y. Khoo and M. Lindsey. Scalable semidefinite programming approach to variational embedding for quantum many-body problems. *arXiv:2106.02682*, 2021.

[31] G. Knizia and G. Chan. Density matrix embedding: A simple alternative to dynamical mean-field theory. *Phys. Rev. Lett.*, 109:186404, 2012.

[32] G. Knizia and G. K.-L. Chan. Density matrix embedding: A strong-coupling quantum embedding theory. *J. Chem. Theory Comput.*, 9:1428–1432, 2013.

[33] G. Kotliar, S. Y. Savrasov, K. Haule, V. S. Oudovenko, O. Parcollet, and C. A. Marianetti. Electronic structure calculations with dynamical mean-field theory. *Rev. Mod. Phys.*, 78:865, 2006.

[34] J. B. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on optimization*, 11(3):796–817, 2001.

[35] J. B. Lasserre. *Moments, Positive Polynomials and Their Applications*. Imperial College Press, 2010.

[36] A. Lemon, A. M.-C. So, and Y. Ye. *Low-Rank Semidefinite Programming: Theory and Applications*. Now Publishers, 2016.

[37] L. Lin and M. Lindsey. Variational embedding for quantum many-body problems. *Comm. Pure Appl. Math.*, 75:2033–2068, 2022.

[38] DC. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45:503, 1989.

[39] G. Mao, B. Fidan, and B. DO. Anderson. Wireless sensor network localization techniques. *Computer networks*, 51(10):2529–2553, 2007.

[40] D. Mazziotti. Realization of quantum chemistry without wave functions through first-order semidefinite programming. *Phys. Rev. Lett.*, 93:213001, 2004.

[41] D. Mazziotti. Structure of fermionic density matrices: Complete n-representability conditions. *Phys. Rev. Lett.*, 108:263002, 2012.

[42] D. A. Mazziotti. Contracted schrödinger equation: Determining quantum energies and two-particle density matrices without wave functions. *Phys. Rev. A*, 57:4219, 1998.

[43] M. Nakata, H. Nakatsuji, M. Ehara, M. Fukuda, K. Nakata, and K. Fujisawa. Variational calculations of fermion second-order reduced density matrices by semidefinite programming algorithm. *J. Chem. Phys.*, 114:8282–8292, 2001.

[44] Y. Nesterov and A. Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*. Society for Industrial and Applied Mathematics, 1994.

[45] J. Nie. Optimality conditions and finite convergence of Lasserre's hierarchy. *Mathematical programming*, 146(1-2):97–121, 2014.

[46] R. Orús. A practical introduction to tensor networks: Matrix product states and projected entangled pair states. *Ann. Phys.*, 349:117–158, 2014.

[47] I. V. Oseledets and E. E. Tyrtyshnikov. Breaking the curse of dimensionality, or how to use svd in many dimensions. *SIAM J. Sci. Comput.*, 31:3744–3759, 2009.

[48] O. Ozyesil, V. Voroninski, R. Basri, and A. Singer. A survey of structure from motion. *arXiv preprint arXiv:1701.08493*, 2017.

[49] P. A. Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Mathematical programming*, 96(2):293–320, 2003.

[50] G. Pataki. On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues. *Mathematics of Operations Research*, 23:339, 1998.

[51] A. Pelizzola. Cluster variation method in statistical physics and probabilistic graphical models. *Journal of Physics A: Mathematical and General*, 38(33):R309, 2005.

[52] P. Pfeuty. The one-dimensional ising model with a transverse field. *Annals of Physics*, 57(1):79–90, 1970.

[53] Amit Singer. Angular synchronization by eigenvectors and semidefinite programming. *Applied and computational harmonic analysis*, 30(1):20–36, 2011.

[54] D. Sun, K.-C. Toh, and L. Yang. A convergent 3-block semiproximal alternating direction method of multipliers for conic programming with 4-type constraints. *SIAM journal on Optimization*, 25:882–915, 2015.

[55] D. Sun, K.-C. Toh, Y. Yuan, and X.-Y. Zhao. Sdpnal+: A matlab software for semidefinite programming with bound constraints (version 1.0). *Optimization Methods and Software*, 35:87–115, 2020.

[56] Q. Sun and G. K.-L. Chan. Quantum embedding theories. *Acc. Chem. Res.*, 49:2705–2712, 2016.

[57] M. Teubner. Ground states of classical one-dimensional lattice models. *Physica A*, 169:407, 1990.

[58] M. J. Wainwright and M. I. Jordan. *Graphical models, exponential families, and variational inference*. Now Publishers Inc, 2008.

[59] Y. Wang and Y. Khoo. Solving cluster moment relaxation with hierarchical matrix. *arXiv:2408.00235*, 2024.

[60] B. Wohlberg. Admm penalty parameter selection by residual balancing. *arXiv:1704.06209*, 2017.

[61] Z. Zhao, B. J. Braams, M. Fukuda, M. L. Overton, and J. K. Percus. The reduced density matrix method for electronic structure calculations and the role of three-index representability conditions. *J. Chem. Phys.*, 120:2095–2104, 2004.