Imperial College London

MEng Individual Project

Imperial College London

Department of Computing

Quantum Belief Propagation

Author:
Jian Zhao

Supervisor:
Dr. Roberto Bondesan

Second Marker:
Prof. Wayne Luk

June 19, 2024

Abstract

Reduced density matrices are central to studies of quantum systems. Due to the size of the Hamiltonian scaling exponentially as system size increases, it is often infeasible to obtain the exact density matrices of systems of interest. Belief propagation algorithms are one of the candidates for obtaining approximate solutions. They have produced good approximation in probabilistic graphical models, which is the classical analogue of quantum systems. In this project, we work through the derivation of a quantum belief propagation algorithm by taking the steps from the derivation of classical algorithm. The derivation is made under fewer assumptions than some in literature, resulting in a more general algorithm. We implement the resulting algorithms as software modules for 1D systems and 2D lattice-like systems. We then study the performance of the algorithm including computation time, correctness, convergence, and scalability. The 1D version of the algorithm demonstrated outstanding performance. The 2D version demonstrated good performance for systems at high temperatures but required more care regarding numerical problems at lower temperatures.

Acknowledgements

I would like to thank my supervisor Dr. Roberto Bondesan for introducing the topic of the project to me in a concise and effective way, and offering me great advice and feedback throughout the project. His guidance has helped me improve my ability in academic research.

I would also like to thank my second marker Prof. Wayne Luk for the valued feedback in the interim meeting.

Contents

Chapter 1

Introduction

1.1 Motivation

Reduced density operators are powerful tools for describing subsystems of a composite quantum system. For example, such an operator can describe the state of one particular qubit in a system of many qubits [14]. Knowing the reduced density operator is useful for many analyses of quantum systems, such as studying observable quantities in entangled multi-body systems [2], determine the energy in electronic systems [17], and even computing the quantum Fisher information of an entire many-body system [3].

Therefore, computing the reduced density operator is an important task in quantum many-body physics. The exact computation is generally intractable because the size of the problem grows exponentially as the number of particles grows. However, for practical problems, there is a lot of research in developing efficient algorithms that can be implemented on a classical computer [4, 8].

One of the widely studied approximation methods to study quantum systems is Quantum Monte Carlo (QMC) [21]. QMC uses the Lie-Trotter formula $e^{A+B} \approx (e^{\frac{A}{n}} e^{\frac{B}{n}})^n$ [24] to break the density matrix up into a product, where resolutions of identity can be applied to convert most of the computation into scalar values [21]. Then, the Monte Carlo simulation can be applied to generate approximation to these values. This approach has been studied on some systems and produced good results [11].

However, QMC has several shortcomings. First, by introducing the resolution of identity, we need to compute the sum of product based on the original algorithm. This effectively creates another dimension to the problem, which makes it more difficult and negatively impacts scaling. Second, the Trotter error from the Lie-Trotter formula increases with the size of the system [5]. This also makes it hard to scale to larger systems. T a very the approximation in the formula converges to the exact value as $n \to \infty$ [28], which means that, depending on the nature of A and B, we might need a very large n for the approximation to be good enough. Forth, when applied to fermionic systems, QMC has the "sign problem", where negative weights appear in the simulation, causing exponential growth in error [25], which is undesirable. Finally, QMC inherits all the shortcomings of regular Monte Carlo methods, such as being dependant on having sufficient knowledge or data about the system to form a good model with good results [6].

Therefore, researchers have explored different ways to study quantum systems, such as variational algorithms. They started with density matrix renormalisation group (DMRG), which decomposes the space of a quantum system into small subsystems and build from these systems using matrix renormlisation [15, 26]. There are also attempts to draw on the similarity of quantum systems and classical probability models to build a quantum belief propagation algorithm [10]. This will be the topic of this project.

In the classical setting, there is an analogue of the problem of finding the reduced density operator given the overall density operator of the system. It is computing marginal distributions on a graphical model given the joint probability distribution of the entire model [10]. Quantum systems become more and more similar to classical systems as temperature approaches 0, where

higher energy states become less significant.

This classical problem in itself is a very interesting one, as such graphical models are used for many practical applications such as classical statistical physics [22], image processing [13] and mechanism design in strategic settings [12].

Given an exponential family of probability distributions, we can reformulate the problem by relating the marginal distribution to the optimum of some value of interest. This effectively converts the marginalisation problem into an optimisation problem using a variational method. The belief propagation sum-product algorithm provides an efficient way to compute the exact marginal distribution on tree-like graphs via this method. Although the algorithm is not exact for graphs with cycles, it provides a very useful approximation [27].

Similarly, variational methods can be applied to the computation of reduced density operators, which leads to a quantum version of the belief propagation algorithm [18]. The main additional consideration from a mathematical point of view in the quantum version is dealing with the non-commutativity of the operators.

With the increased interest in applying machine learning in quantum technology, some research on quantum belief propagation algorithms has been done regarding the simulation of quantum computations in tensor networks [1, 23].

Despite the increased interest and the fact that the existence of many classical belief propagation software packages, there are no quantum belief propagation software available, which indicates a gap between the research tools of classical statistians and quantum physics.

This project aims to bridge the gap between classical belief propagation and quantum belief propagation. It will be focusing on the quantum belief propagation algorithm in the context of studying thermal states in many-body quantum systems. We will derive a quantum belief propagation algorithm by closely following the steps for the derivation of a classical belief propagation algorithm described in [27]. Then, we will build and evaluate a software package which is implemented based on the algorithm. We then use it to benchmark the quantum belief propagation algorithm against other algorithms in simulations of many-body quantum systems at finite temperature in the literature.

1.2 Overview

As background material, this project studies in detail the necessary components that are required to build the classical belief propagation algorithm as described in [27].

This starts from the definitions of graphical models. We show how a probabilistic graphical model can be represented as exponential families in terms of both canonical parameters and mean parameters. We introduce the log-partition function for the exponential family and its conjugate dual function. We show that the log-partition function is convex to build a variational optimisation problem. By establishing the connection between the conjugate dual and the Shannon entropy of the model, we can base the optimisation problem on the entropy. At this point, the optimisation problem is still difficult to solve, so we introduce relaxations to the problem by using the Bethe entropy approximation and dropping global consistency requirements for local consistency requirements. With these relaxations applied, we obtain a relatively straightforward Lagrangian for the problem. Differentiating the Lagrangian and applying some algebraic manipulation gives the classical belief propagation algorithm.

Additionally, the background material also covers some results used to develop the belief propagation algorithm, including cliques on a graph and the sum product algorithm, as well as some basic concepts useful for the quantum case, including von Neumann entropy and thermal equilibrium state representation.

With this background knowledge, we repeat the same steps on the quantum thermal equilibrium state representation, where the overall density matrix can also be interpreted as an exponential family. With assumption of short range interactions commonly used in studies of quantum systems, the systems can be interpreted as graphs, where edges correspond to particles that interact with each other. We identify the basis matrices corresponding to sufficient statistics in the classical case,

and define canonical parameters and mean parameters in their regard. Again, we showed the convexity of the log-partition function, which was much more involved than the classical case due to the non-commutativity of the matrices. We use the von Neumann entropy instead of the Shannon entropy to develop a connection with the conjugate dual function for the variational problem.

In order to get a straightforward Lagrangian, we use the ideas from the classical case to relax the variational problem. We start from the strong subadditivity theorem for von Neumann entropy and proceed to develop an entropy approximation that closely resembles the Bethe entropy approximation, albeit with less theoretical guarantee. We also drop global consistency conditions for local consistency conditions, which result in matrix Lagrangian multipliers.

This differs from the algorithms in literature where stronger requirements are made on the system in question, in that they need to be bi-factor networks [10]. The aim of our algorithm is to work on more general quantum systems using heuristics learned from the classical case.

With the Lagrangian, we start by restricting to the case where particles from a chain, which gives the 1D version of the algorithm. This is equivalant to the 1D algorithm described in [18], with a "Markov shield" of 2 and without looping. An algebraic interpretation was given for the ends of the chain.

This was then implemented in software, with options to specify the parameters of the Hamiltonian of the model and the belief propagator to run the algorithm and extract the beliefs.

The software was then used to benchmarking against exact solution and known results from Variational QMC. The benchmarking was more comprehensive than those presented in literature since it also covered computation time and convergence, in addition to the correctness of the results.

Our algorithm achieved very good performance in terms of running time compared to the exact solution. It also offers a very good approximation of the reduced density matrices, hence also other quantities of interest. It scales very well, giving state-of-the-art results for 100 particles.

Then, a general quantum belief propagation algorithm was developed based on the Lagrangian. The main difference between this and the classical algorithm is the directions of messages plays an explicit role in matrix representation, since reduced density matrices, like the overall density matrix, have an explicit ordering of the particles.

Instead of the general algorithm, a particular algorithm working on 2D lattices was implemented in software. This was chosen because the lattice model was a common model studied in literature, and had a straightforward implementation using rows and columns that deals very well with the directional nature of the algorithm. Again, there are options to specify the parameters of the Hamiltonian of the model and the belief propagator to run the algorithm and extract the beliefs.

We then studied 2D lattices using both the 1D version of the algorithm and the 2D version. For the 1D version, we used a snake-shaped approximation that treats all the particles as if they were on a chain. The 2D version achieved much better performance than the 1D version on a 3×3 lattice. Although the 2D algorithm performed well at low temperatures and in small systems, its scalability was not as good as it suffered from numerical stability issues. Since the algorithm required inverses of message matrices to be computed, numerical issues arose when the messages are close to singular. Regularisation methods were applied to some degree of success.

Chapter 2

Background

2.1 Graphical Model and Belief Propagation

2.1.1 Markov Random Field

The probablility distribution of random variables $(X_1 ... X_n)$ can be represented in a graphical model $G = (V, E)$ where each vertex represents a random variable, and each edge represents the dependence between two variables.

Figure 2.1 is an illustration of a simple graphic model with 6 random variables $X_1 ... X_6$, represented by vertices V_1 to V_6. Since this graph is undirected, it is known as an **undirected graphical model**, also known as a **Markov random field** or **Gibbs distribution** [27].

Graphical models can be very useful for analysing conditional independence. Let U_a, U_b and U_c be disjoint subsets of V, and X_a, X_b and X_c represent their respective joint probability. Suppose that when U_b is removed from V, there are no edges connecting U_a and U_c, then X_a and X_c are conditionally independent on X_b. [9].

For example, in Figure 2.1, the dependence between vertices $\{V_1, V_4\}$ and vertex V_6 is mediated through vertices V_2 and V_5, so if X_2 and X_5 are known, then (X_1, X_4) and X_6 are independent.

$$p(x_1, x_4, x_6 | x_2, x_5) = p(x_1, x_4 | x_2, x_5) p(x_6 | x_2, x_5) \tag{2.1}$$

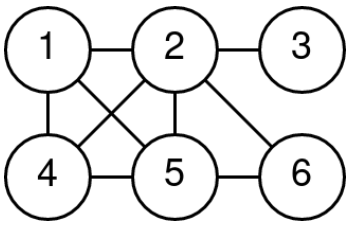

Figure 2.1: Example of an undirected graphical model for probablility distribution.

2.1.2 Cliques

Given a graph $G = (V, E)$, a **clique** C is defined as a subset of V, such that $\forall s, t \in C, (s, t) \in E$. By this definition, any set of a single vertex or any set of a pair of adjacent vertices is a clique. A clique that is not a subset of any other clique is a **maximal clique**.

The conditional independence in the Markov random field gives us a convenient way to factorise the probablility distribution according to cliques. Given the set of all cliques $\mathcal{C}$, the probablility distribution factorises as

$$p(x_1, ..., x_n) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(x_C) \tag{2.2}$$

where Z is a normalising factor [27]. The function ψ_C is called a **compatibility function**. For each C, $\psi_C : \otimes_{s \in C} \mathcal{X}_s \to \mathbb{R}^+$ is a function from the space of the random vector containing the elements of C to the positive real numbers.

Equation 2.2 in fact also holds for other choices of $\mathcal{C}$, since the compatibility functions of a non-mamixal clique can be absorbed into a larger clique that it is a subset of. Therefore, we can choose $\mathcal{C}$ to be the set of all maximal cliques, or the set of all maximal cliques and some non-mamixal cliques.

Referring back to Figure 2.1, we can identify $C_1 = \{V_1, V_2, V_4, V_5\}$, $C_2 = \{V_2, V_5, V_6\}$, and $C_3 = \{V_2, V_3\}$ as the maximal cliques of the graph. Selecting these as the target set of cliques, we can obtain the factorisation using the conditional dependence as follows:

$$\begin{aligned} p(x_1 ... x_6) &= p(x_1, x_4, x_5, x_6 | x_2) p(x_3 | x_2) p(x_2) \\ &= p(x_1, x_4 | x_2, x_5) p(x_6 | x_2, x_5) p(x_5 | x_2) p(x_3 | x_2) p(x_2) \end{aligned} \tag{2.3}$$

Letting

$$\psi_{C_1}(x_{C_1}) = p(x_1, x_4 | x_2, x_5) \tag{2.4}$$
$$\psi_{C_2}(x_{C_2}) = p(x_6 | x_2, x_5) p(x_5 | x_2) \tag{2.5}$$
$$\psi_{C_3}(x_{C_3}) = p(x_3 | x_2) p(x_2) \tag{2.6}$$

and $Z = 1$, we obtain a suitable set of compatibility functions for the set of maximal cliques.

This method of factorisation by conditioning on the nodes shared between cliques can be used to find a factorisation in the form of Equation 2.2 for probability distributions in general.

2.1.3 Sum-Product Algorithm

Consider a tree-structured graph with vertices V and edges E. Since there are no cycles, the cliques in the graph only consists of the vertices and edges. Hence, the probablility distribution associated with such a graphic model can be factorised over all cliques as

$$p(x) = \frac{1}{Z} \prod_{s \in V} \psi_s(x_s) \prod_{(s,t) \in E} \psi_{s,t}(x_s, x_t) \tag{2.7}$$

We are interested in the marginal distribution $\mu(x_s) = \sum_{\{x' | x'_s = x_s\}} p(x')$ for a particular node $s \in V$.

Let $N(s)$ denote the neighbours of s,

$$N(s) = \{t \in V | (s,t) \in E\} \tag{2.8}$$

For each $t \in N(s)$, let T_t denote the subtree (V_t, E_t), where V_t consists of the vertices that can be connected to t without going throught s, and E_t consists of the edges connecting vertices in V_t.

Let x_{V_t} denote the random vector associated to the subtree T_t, then

$$p(x_{V_t}) \propto \prod_{u \in V} \psi_u(x_u) \prod_{(u,v) \in E} \psi_{u,v}(x_u, x_v) \tag{2.9}$$

Now defining the message from $t \in N(s)$ to s

$$M^*_{ts}(x_s) := \sum_{x'_{V_t}} \psi_{st}(x_s, x'_t) p(x'_{V_t}) \tag{2.10}$$

we have

$$\mu(x_s) = \kappa \psi_s(x_s) \prod_{t \in N(u)} M^*_{ts}(x_s) \tag{2.11}$$

where κ is a normalising constant.

Proof. Since κ is a normalising constant, we only need to show $\mu(x_s) \propto \psi_s(x_s) \prod_{t \in N(u)} M^*_{ts}(x_s)$. Now,

$$\begin{aligned}
&\psi_s(x_s) \prod_{t \in N(u)} M^*_{ts}(x_s) \\
=&\psi_s(x_s) \prod_{t \in N(u)} \left(\sum_{x'_{V_t}} \psi_{st}(x_s, x'_t) p(x'_{V_t}) \right) \\
\propto&\psi_s(x_s) \prod_{t \in N(u)} \left(\sum_{x'_{V_t}} \left(\psi_{st}(x_s, x'_t) \prod_{u \in V_t} \psi_u(x'_u) \prod_{(u,v) \in E_t} \psi_{u,v}(x'_u, x'_v) \right) \right) \\
=&\psi_s(x_s) \sum_{x'_{V \setminus s}} \left(\prod_{u \in V \setminus \{s\}} \psi_u(x'_u) \prod_{(u,v) \in E} \psi_{u,v}(x'_u, x'_v) \right) \\
=& \sum_{\{x' | x'_s = x_s\}} \left(\prod_{u \in V} \psi_u(x'_u) \prod_{(u,v) \in E} \psi_{u,v}(x'_u, x'_v) \right) \\
=& \sum_{\{x' | x'_s = x_s\}} p(x') \\
=&\mu(x_s)
\end{aligned} \tag{2.12}$$

proves the statement, where we have used the property

$$\left(\sum_{a \in A} f(a) \right) \left(\sum_{b \in B} g(b) \right) = \sum_{(a,b) \in A \times B} f(a) g(b) \tag{2.13}$$

iteratively to simplify the product of sums. □

This provides us with a way to compute the marginal distribution for any particular random variable in a tree-structured graph. The sum-product algorithm is based on this idea but sends messages between all adjacent nodes simultaneously. Let $M_{ts}(x_s)$ denote the message sent from t to $s \in N(t)$, it is defined as

$$M_{ts}(x_s) \leftarrow \kappa \sum_{x'_t} \left(\psi_{st}(x_s, x'_t) \psi_t(x'_t) \prod_{u \in N(t) \setminus \{s\}} M_{ut}(x'_t) \right) \tag{2.14}$$

where κ is a normalisation term [27].

At each step, each node gathers all the messages it has received in the product term, which is analogous to the $p_{x'_{V_t}}$ term in the complete message M^*_{ts}. It can be shown [16] that after a finite number of iterations on trees, the messages converge to the fixed point $\{M^*_{ts}, M^*_{st} | (s,t) \in E\}$, as defined in Equation 2.10.

Although the sum-product algorithm is only exact for belief propagation on trees, it can still be quite useful in approximate inference on graphs with cycles. In order to see how it adapts to the general graphic model, we can reframe the problem of marginalisation to a problem of optimisation using variational methods. In order to see how variational methods can be applied, we view graphical models as exponential families.

2.2 Exponential Families and Examples

2.2.1 Exponential Families

Consider the random variables $X = (X_1 ... X_n) \in \mathcal{X}^m$ and sufficient statistics $\phi_1(x) ... \phi_d(x)$ for the random variables, which are $\mathcal{X}^m \to \mathbb{R}$. Let $\phi(x) = (\phi_1(x) ... \phi_d(x))$, then ϕ is a mapping from the

random variables to the d-dimensional vector representation of sufficient statistics. Let $\theta \in \mathbb{R}^d$ be the associated vector of canonical parameters. The **log-partition function** is defined as

$$A(\theta) = \log \left[\int_{\mathcal{X}^m} \exp\langle \theta, \phi(x) \rangle \nu(\mathrm{d}x) \right] \tag{2.15}$$

where $\langle , \rangle$ represents the Euclidean inner product, and ν is the underlying measure [27]. In the case of continuous variables, this is the Lebesgue measure for regular integration. In the case of discrete variables, this is the counting measure, so the integral is just the sum of the possible values as follows

$$A(\theta) = \log \left[\sum_{\mathcal{X}^m} \exp\langle \theta, \phi(x) \rangle \right] \tag{2.16}$$

The set of canonical parameters of interest is defined as

$$\Omega = \{\theta \in \mathbb{R}^d | A(\theta) < +\infty\} \tag{2.17}$$

Given $\theta \in \Omega$, the **exponential family** associated with ϕ is the set of functions for each θ value,

$$p_\theta(x) = \exp(\langle \theta, \phi(x) \rangle - A(\theta)) \tag{2.18}$$

We shall illustrate how some common distributions can be considered as exponential families via examples.

2.2.2 Bernoulli Distribution

Consider a Bernoulli distribution $X \sim \mathcal{B}(p)$, where p is the probablility that $x = 1$. A sufficient statistic for this distribution is simply the outcome x, and its corresponding canonical parameter can be denoted θ. The measure is the counting measure over $\mathcal{X} = \{0, 1\}$. Hence,

$$\begin{aligned} A(\theta) &= \log \left(\sum_{x \in \mathcal{X}} \exp(\theta x) \right) \\ &= \log \left(\exp(\theta) + 1 \right) \end{aligned} \tag{2.19}$$

Since $A(\theta)$ is finite for all real θ values, the set of interest is $\Omega = \mathbb{R}$. This gives the exponential family of

$$\begin{aligned} p_\theta(x) &= \exp\left(\theta x - \log(\exp(\theta) + 1)\right) \\ &= \frac{\exp(\theta x)}{\exp(\theta) + 1} \end{aligned} \tag{2.20}$$

for $x \in \mathcal{X}$. Regarding the original parameter p, we have $p = \frac{\exp(\theta)}{\exp(\theta)+1}$. It can be verified that this satisfies $p \in (0, 1)$. If we include the limits for $\theta \to \pm\infty$, then we can also get $p = 0$ or $p = 1$.

2.2.3 Ising Model

This example is taken from [27].

The Ising Model is a common model used in statistical physics. It is a graphical model on $G = (V, E)$, where the random variable x_s at each $s \in V$ is a Bernoulli distribution. Each pair of random variables x_s and x_t can only interact if $(s, t) \in E$. In this case, the set of sufficient statistics is the union of $\{x_s | s \in V\}$ and $\{x_s, x_t | (s, t) \in E\}$. This leads to an exponential family of probability distributions

$$p_\theta(x) = \exp\left(\sum_{s \in V} \theta_s x_s + \sum_{(s,t) \in E} \theta_{st} x_s x_t - A(\theta) \right) \tag{2.21}$$

where

$$A(\theta) = \log\left(\sum_{x\in\{0,1\}^m} \exp\left(\sum_{s\in V}\theta_s x_s + \sum_{(s,t)\in E}\theta_{st}x_s x_t\right)\right) \tag{2.22}$$

where $m = |V|$, since the measure in this model is the counting model involving $\{0, 1\}$ for each of the random variables in the graph.

The Ising Model has many generalisations, including for example, the model where triples of vertices can interact, and the model where each vertex represents a random variable that can take one of k values, where $k \in \mathbb{N}$ and $k > 2$.

2.2.4 Gaussian Distribution

Consider a Gaussian Distribution $X \sim \mathcal{N}(\mu, \sigma^2)$. A sufficient statistic $\phi(x) = (x, x^2)$, since the original parameters correspond to the first 2 moments of the distribution. The canonical parameters can be denoted as $\theta = (\theta_1, \theta_2)$. The measure is the Lebesgue measure over $\mathcal{X} = \mathbb{R}$. For the exponential family to be normalised properly, we introduce a modulating factor $h(x) = \frac{1}{\sqrt{2\pi}}$. Hence,

$$\begin{aligned} A(\theta) &= \log\left(\int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi}}\exp(\theta_1 x + \theta_2 x^2)\mathrm{d}x\right) \\ &= \log\left(\left(\frac{1}{-2\theta_2}\right)^{\frac{1}{2}}\exp\left(\frac{\theta_1^2}{4\theta_2}\right)\right) \\ &= -\frac{\theta_1^2}{4\theta_2} - \frac{1}{2}\log(-2\theta_2) \end{aligned} \tag{2.23}$$

Since the logarithm is only defined for $\theta_2 < 0$, the set of interest $\Omega = \{(\theta_1, \theta_2) \in \mathbb{R}^2 | \theta_2 < 0\}$. This gives the exponential family of

$$\begin{aligned} p_\theta(x) &= h(x)\exp(\langle\theta, \phi(x)\rangle - A(\theta)) \\ &= \frac{1}{\sqrt{2\pi}}\exp\left(\theta_1 x + \theta_2 x^2 + \frac{\theta_1^2}{4\theta_2} + \frac{1}{2}\log(-2\theta_2)\right) \\ &= \sqrt{\frac{-\theta_2}{\pi}}\exp\left(\theta_1 x + \theta_2 x^2 + \frac{\theta_1^2}{4\theta_2}\right) \end{aligned} \tag{2.24}$$

Comparing this to the standard equation for Gaussian distribution

$$p_{\mu,\sigma^2}(x) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left(\frac{(x-\mu)^2}{\sigma^2}\right) \tag{2.25}$$

we see the correspondence $\theta_1 = \frac{\mu}{\sigma^2}$ and $\theta_2 = -\frac{1}{2\sigma^2}$. Also, the requirement that $\theta_2 < 0$ corresponds to the requirement that $\sigma^2 > 0$.

2.2.5 Gaussian Markov Random Field

This example is taken from [27].

The Gaussian Markov random field is a Markov random field that represents a multivariate Gaussian distribution. Each vertex represents a single Gaussian random variable and random variables only interact if they are joined by an edge. The set of sufficient statistics is the union of $\{x_s, x_s^2 | s \in V\}$ and $\{x_s, x_t | (s,t) \in E\}$, where the x_s^2 term correspond to variances of each variable and the $x_s x_t$ terms correspond to covariances between terms that are adjacent to each other. For uniformity, we can consider the matrix xx^T, which corresponds to the overall covariance.

This leads to the definition of θ, a vector of size $m = |V|$, corresponding to x, and a symmetric matrix $\Theta \in \mathbb{R}^{m\times m}$ corresponding to xx^T, requiring that $\Theta_{st} = 0$ for (s,t) not connected by an edge. This leads to the exponential family of

$$p_\theta(x) = \exp\left(\langle\theta, x\rangle + \frac{1}{2}\mathrm{Tr}\left(\Theta xx^T\right) - A(\theta, \Theta)\right) \tag{2.26}$$

where

$$\mathrm{Tr}\left(\Theta x x^T\right) = \sum_{i=1}^{m}\sum_{j=1}^{m}\Theta_{ij}x_i x_j \tag{2.27}$$

indeed captures all the quadratic terms of the sufficient statistics. For the integral underlying the function A to be finite, we require additionally that $\Theta \prec 0$.

2.3 Convexity and Mean Parameter Space

2.3.1 Convexity

The following are some definitions related to convexity from [19].

A **convex set** is a set $S \subseteq \mathbb{R}^d$ such that $\forall x, y \in S$ and $0 \leq \alpha \leq 1$, $\alpha x + (1-\alpha)y \in S$. If a convex set contains two points, then it also contains all the points that lie between them on a line segment.

Given elements $x_1, ...x_k$ in a set S, a **convex combination** is a linear combination of the elements $\sum_{i=1}^{k}\alpha_i x_i$, where $\sum_{i=1}^{k}\alpha_i = 1$ and $\alpha_i \geq 0$ for all i.

A **convex hull** of a set S, $\mathrm{conv}(S)$, is the smallest set that contains all its convex combinations. This is a convex set by construction.

2.3.2 Mean Parameterisation and Examples

Mean parameters offer an alternative way of describing a probablility distribution from canonical parameters described earlier in Section 2.2.1.

Consider again the random variables $X = (X_1...X_n) \in \mathcal{X}^m$ and sufficient statistics $\phi_1(x)...\phi_d(x)$ for the random variables, which are $\mathcal{X}^m \to \mathbb{R}$. Let $p(x)$ be any valid probability distribution for X. For each $1 \leq i \leq d$, the **mean parameter** μ_i associated with a sufficient statistic $\phi_i(x)$ is defined as

$$\mu_i = \mathbb{E}_p[\phi_i(x)] = \int_{\mathcal{X}} \phi_i(x)p(x)\nu(\mathrm{d}x) \tag{2.28}$$

where ν is again the underlying measure.

Let $\mu = (\mu_1...\mu_d)$, a vector of mean parameters corresponding to each of the sufficient statistics, then the set of valid μ values as p varies is defined as

$$\mathcal{M} := \{\mu | \exists p \text{ s.t. } \mu_i = \mathbb{E}_p[\phi_i(x)]\ \forall i \in \{1...d\}\} \tag{2.29}$$

We known that $\mathcal{M}$ must be a convex set, since for any $\mu_1, \mu_2 \in \mathcal{M}$ with corresponding valid probability distribution p_{μ_1}, p_{μ_2}, the combination $\lambda p_{\mu_1} + (1-\lambda)p_{\mu_2}$ is also a valid probability distribution for $0 \leq \lambda \leq 1$, so $\lambda\mu_1 + (1-\lambda)\mu_2$ is also a member of $\mathcal{M}$. [27]

We shall illustrate the mean parameters using the same examples for exponential families. These examples are taken from [27], and correspond to Section 2.2.3 and Section 2.2.5.

Gaussian Markov Random Field

Consider a Gaussian Markove random field as described in Section 2.2.5. Since the sufficient statistics are x and xx^T, the mean parameters here are $\mu = \mathbb{E}[X] \in \mathbb{R}^m$ and $\Sigma = \mathbb{E}[XX^T] \in \mathbb{R}^{m\times m}$, where m is the number of vertices. The covariance of such a distribution is $\Sigma - \mu\mu^T$, which needs to be positive semi-definite, leading to the constraint that $\Sigma - \mu\mu^T \succeq 0$.

Ising Model

Consider an Ising Model as described in Section 2.2.3. The sufficient statistics are $\{x_s | s \in V\} \cup \{x_s x_t | (s,t) \in E\}$, the elements of which can be represented by the vector $\phi(x) \in \mathbb{R}^{|V|+|E|}$. The

mean parameters are

$$\mu_s = \mathbb{E}[X_s] = \mathbb{P}[X_s = 1] \text{ for } s \in V \tag{2.30}$$

$$\mu_{st} = \mathbb{E}[X_s X_t] = \mathbb{P}[X_s = 1 \wedge X_t = 1] \text{ for } (s,t) \in E \tag{2.31}$$

Hence, the set $\mathcal{M}$ is the convex hull of $\{\phi(x)|x \in \{0,1\}^m\}$, where $m = |V|$.

2.3.3 Convexity of the Log-partition Function

In this section, we demonstrate the convexity of the log-partition function as defined in Equation 2.1 by investigating the positive semi-definiteness of its Hessian $\nabla^2 A$.

The first derivative of A with respect to one of the canonical paramters is

$$\begin{aligned}
\frac{\partial A}{\partial \theta_i}(\theta) &= \frac{\partial}{\partial \theta_i}\left[\log\left(\int_{\mathcal{X}^m} \exp(\langle\theta, \phi(x)\rangle)\nu(\mathrm{d}x)\right)\right] \\
&= \frac{\int_{\mathcal{X}^m} \frac{\partial}{\partial \theta_i}\exp(\langle\theta, \phi(x)\rangle)\nu(\mathrm{d}x)}{\int_{\mathcal{X}^m} \exp(\langle\theta, \phi(x)\rangle)\nu(\mathrm{d}x)} \\
&= \int_{\mathcal{X}^m} \phi_i(x) \frac{\exp(\langle\theta, \phi(x)\rangle)}{\int_{\mathcal{X}^m} \exp(\langle\theta, \phi(u)\rangle)\nu(\mathrm{d}u)}\nu(\mathrm{d}x) \\
&= \int_{\mathcal{X}^m} \phi_i(x) p_\theta(x)\nu(\mathrm{d}x) \\
&= \mathbb{E}_\theta[\phi_i(X)]
\end{aligned} \tag{2.32}$$

where we have differentiated through the integral, which is valid as can be shown using the dominated convergence theorem [27]. The gradient $\nabla A(\theta)$ can be considered as a mapping to the mean parameters.

Hence, the second derivative is

$$\begin{aligned}
\frac{\partial^2 A}{\partial \theta_i \partial \theta_j}(\theta) &= \frac{\partial^2 A}{\partial \theta_i \partial \theta_j}\left[\log\left(\int_{\mathcal{X}^m} \exp(\langle\theta, \phi(x)\rangle)\nu(\mathrm{d}x)\right)\right] \\
&= \frac{\partial}{\partial \theta_i}\left[\frac{\int_{\mathcal{X}^m} \frac{\partial}{\partial \theta_j}\exp(\langle\theta, \phi(x)\rangle)\nu(\mathrm{d}x)}{\int_{\mathcal{X}^m} \exp(\langle\theta, \phi(x)\rangle)\nu(\mathrm{d}x)}\right] \\
&= \frac{\int_{\mathcal{X}^m} \frac{\partial^2}{\partial \theta_i \partial \theta_j}\exp(\langle\theta, \phi(x)\rangle)\nu(\mathrm{d}x)}{\int_{\mathcal{X}^m} \exp(\langle\theta, \phi(x)\rangle)\nu(\mathrm{d}x)} \\
&\quad - \frac{\int_{\mathcal{X}^m} \frac{\partial}{\partial \theta_i}\exp(\langle\theta, \phi(x)\rangle)\nu(\mathrm{d}x)}{\int_{\mathcal{X}^m} \exp(\langle\theta, \phi(x)\rangle)\nu(\mathrm{d}x)} \frac{\int_{\mathcal{X}^m} \frac{\partial}{\partial \theta_i}\exp(\langle\theta, \phi(x)\rangle)\nu(\mathrm{d}x)}{\int_{\mathcal{X}^m} \exp(\langle\theta, \phi(x)\rangle)\nu(\mathrm{d}x)} \\
&= \mathbb{E}_\theta[\phi_i(X)\phi_j(X)] - \mathbb{E}_\theta[\phi_i(X)]\mathbb{E}_\theta[\phi_j(X)] \\
&= \mathrm{cov}[\phi_i(X), \phi_j(X)]
\end{aligned} \tag{2.33}$$

Therefore, the full Hessian matrix $\nabla^2 A(\theta)$ is the covariance matrix of $\phi(x)$, which is by definition positive semi-definite on the open set Ω, which means the function A is convex [27].

2.3.4 Conjugate Dual Function

The **conjugate dual function** of a given log-partition function $A(\theta)$ is defined as

$$A^*(\mu) := \sup_{\theta \in \Omega}[\langle\mu, \theta\rangle - A(\theta)] \tag{2.34}$$

The conjugate dual functions satisfies some properties that enable us to convert the estimation problem for marginal probabilities to a variational optimisation problem [27].

Property A:

For $\mu \in \mathcal{M}^\circ$, let $\theta(\mu)$ be the unique vector of canonical parameters that satisfy the relation $\mathbb{E}_{\theta(\mu)}[\phi(X)] = \nabla A(\theta(\mu)) = \mu$, then

$$A^*(\mu) = \begin{cases} -H(p_{\theta(\mu)}) & \mu \in \mathcal{M}^\circ \\ +\infty & \mu \notin \overline{\mathcal{M}} \end{cases} \tag{2.35}$$

For boundary points $\mu \in \overline{\mathcal{M}} \backslash \mathcal{M}^\circ$, $A^*(\mu)$ is the limit of any sequence $\{\mu^n\} \subset \mathcal{M}^\circ$ that converges to μ.

Property B:

The log-partition function can be represented using the conjugate dual as

$$A(\theta) = \sup_{\mu \in \mathcal{M}} [\langle \theta, \mu \rangle - A^*(\mu)] \tag{2.36}$$

Property C:

For all $\theta \in \Omega$, the supremum in Equation 2.36 is attained uniquely at the vector $\mu \in \mathcal{M}^\circ$ that satisfies $\mathbb{E}_\theta(\phi(X)) = \mu$.

An important observation is that property B provides a way to find $A(\theta)$ by optimising over the μ values, and property A guarantees that any $\mu \notin \overline{\mathcal{M}}$ cannot be the optimal solution. This means we only need to optimise over $\mathcal{M}$ instead of $\mathbb{R}^d$. The fact that $A^*(\mu)$ is related to the entropy of the probablility distribution gives us a way to approach such an optimisation via the entropy.

We will illustrate the properties with an example taken from [27].

Bernoulli Distribution

Consider a scalar random variable following $X \sim \mathcal{B}(p)$ taking values from $\{0, 1\}$. Taking $\phi(x) = x$ as the sufficient statistic, we known from section 2.2.2 that $A(\theta) = \log(1 + \exp(\theta))$ and $\Omega = \mathbb{R}$. We also know that the mean parameter $\mu \in \mathcal{M} = [0, 1]$.

We have

$$A^*(\mu) = \sup_{\theta \in \mathbb{R}} [\theta\mu - \log(1 + \exp(\theta))] \tag{2.37}$$

Differentiating with respect to θ gives $\mu - \frac{\exp(\theta)}{1+\exp(\theta)}$. Setting this to 0, we obtain that $\theta = \log(\frac{\mu}{1-\mu})$. For $\mu \in (0, 1) = \mathcal{M}^\circ$, we have

$$\begin{aligned} A^*(\mu) =& \mu \log\left[\frac{\mu}{1-\mu}\right] - \log\left[1 + \frac{\mu}{1+\mu}\right] \\ =& \mu\log(\mu) - (1-\mu)\log(1-\mu) \end{aligned} \tag{2.38}$$

It can also be verified that for $\mu \notin [0, 1] = \overline{\mathcal{M}}$, the objective function in Equation 2.37 is unbounded, so $A^*(\mu) = +\infty$, and that for boundary points $\mu = 0$ and $\mu = 1$, $A^*(\mu) = 0$ from limits.

2.4 Variational Belief Propagation

This section is adapted from the description of the Bethe Variational Problem in [27], with some added details, new examples and a particular focus on Bernoulli random variables, since they are analogous to qubits in question in the quantum setting.

2.4.1 Model Setup

Bethe approximation can be applied to an undirected graphical model of Bernoulli random variables on a graph $G = (V, E)$ with compatibility functions involving at most pairs. This is called a **pairwise Markov random field**.

We will focus the discussion on a network of Bernoulli random variables.

In Section 2.2.2, we used the sufficient statistics x_s for $s \in V$ and $x_s x_t$ for $(s,t) \in E$. In order to build a more direct relationship with the marginal distributions at each vertex, we shall use a different set of sufficient statistics here.

For each $x_s \in V$, let $\mathbb{I}_{s;i}(x_s)$ represent the indicator function of x_s. That is, $\mathbb{I}_{s;0}(x_s) = 1$ if $x = 0$ and 0 otherwise, and $\mathbb{I}_{s;1}(x_s) = 1$ if $x = 1$ and 0 otherwise. Similarly, for each $(s,t) \in E$, let $\mathbb{I}_{st;ij}(x_s, x_t)$ be the indicator function for $x_s = i$ and $x_t = j$. We observe that this set of sufficient statistics is overcomplete, as in the sufficient statistics are not uniquely identifiable.

The corresponding exponential family can be described as

$$p_\theta(x) \propto \exp\left(\sum_{s\in V} \theta_s(x_s) + \sum_{(s,t)\in E} \theta_{st}(x_s, x_t)\right) \tag{2.39}$$

where we have defined the functions

$$\theta_(x_s) := \sum_{i\in\{0,1\}} \theta_{s;i}\mathbb{I}_{s;i}(x_s) \text{ for } s \in V \tag{2.40}$$

$$\theta_s(x_s, x_t) := \sum_{i,j\in\{0,1\}} \theta_{st;ij}\mathbb{I}_{st;ij}(x_s, x_t) \text{ for } (s,t) \in E \tag{2.41}$$

The mean parameters corresponding to the indicator functions are

$$\mu_{s;i} = \mathbb{E}[\mathbb{I}_{s;i}(x_s)] = \mathbb{P}(X_s = i) \text{ for } s \in V \tag{2.42}$$

$$\mu_{st;ij} = \mathbb{E}[\mathbb{I}_{st;ij}(x_s, x_t)] = \mathbb{P}(X_s = i \wedge X_t = j) \text{ for } (s,t) \in E \tag{2.43}$$

These are the marginal probability that a vertex or an edge takes a specific value. Hence, we can define the functions

$$\mu_s(x_s) := \sum_{i\in\{0,1\}} \mu_{s;i}\mathbb{I}_{s;i}(x_s) \text{ for } s \in V \tag{2.44}$$

$$\mu_s(x_s, x_t) := \sum_{i,j\in\{0,1\}} \mu_{st;ij}\mathbb{I}_{st;ij}(x_s, x_t) \text{ for } (s,t) \in E \tag{2.45}$$

which are exactly the marginal distributions for single vertices and pairs of vertices connected by an edge.

This corresponds to a marginal polytope, namely

$$\mathbb{M}(G) = \{\mu \in \mathbb{R}^d | \exists p \text{ s. t. the marginals are } \mu_s(x_s) \text{ and } \mu_{st}(x_s, x_t)\} \tag{2.46}$$

2.4.2 Outer Bound to the Marginal Polytope

We know that the marginal polytope $\mathbb{M}(G)$ can be written as the convex hull of a finite number of vectors corresponding to the sufficient statistics for each possible x. This grows exponentially as the number vertices in the graph grows. Hence, obtaining the exact constraints on $\mathbb{M}(G)$ is difficult in general. By only considering a subset of constraints, we can obtain an outer bound on $\mathbb{M}(G)$ that is easier to work with.

Consider a function τ over the graph such that $\tau_s(x_s)$ and $\tau_{st}(x_s, x_t)$ are the marginals over vertices and edges. For these to be valid marginals, we need τ_s to satisfy the normalisation condition

$$\sum_{x_s} \tau_s(x_s) = 1 \ \forall s \in V \tag{2.47}$$

and τ_{st} to satisfy the marginalisation condition

$$\sum_{x'_s} \tau_{st}(x'_s, x_t) = \tau_t(x_t) \text{ and } \sum_{x'_t} \tau_{st}(x_s, x'_t) = \tau_s(x_s) \ \forall (s,t) \in E \tag{2.48}$$

These constraints define the set of **locally consistent** marginal distributions

$$\mathbb{L}(G) = \{\tau \geq 0 | \text{the normalisation and marginalisation conditions hold for } \tau\} \tag{2.49}$$

We notice that $\mathbb{L}(G)$ is a lot easier to compute compare to $\mathbb{M}(G)$, since the number of constraints grows linearly as the number of vertices and edges grow.

Clearly, $\mathbb{M}(G) \subseteq \mathbb{L}(G)$, since any globally realisable distribution must satistfy the local normalisation and marginalisation constraints.

When the graph G is a tree, we have in fact that $\mathbb{M}(G) = \mathbb{L}(G)$. Given any $\tau \in \mathbb{L}(G)$, we can construct

$$p_\tau(x) := \prod_{s \in V} \tau_s(x_s) \prod_{s \in V} \frac{\tau_{st}(x_s, x_t)}{\tau_s(x_s)\tau_t(x_t)} \tag{2.50}$$

using the junction tree theorem in [27]. When G has cycles, however, we have $\mathbb{M}(G) \subset \mathbb{L}(G)$.

To provide a simple example, consider a fully connected graph $G = (V, E)$ with vertices $V = \{a, b, c\}$, and edges $E = \{(a, b), (a, c), (b, c)\}$. Let

$$\tau_s(x_s) = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \end{pmatrix} \ \forall s \in V \tag{2.51}$$

$$\tau_{ab}(x_a, x_b) = \tau_{ac}(x_a, x_c) = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix} \tag{2.52}$$

$$\tau_{bc}(x_b, x_c) = \begin{pmatrix} 0 & \frac{1}{2} \\ \frac{1}{2} & 0 \end{pmatrix} \tag{2.53}$$

then we can verify that all the local consistency constraints are held. However, we notice that a probability distribution with the given τ is impossible over this graph, since Equation 2.52 specifies $x_a = x_b = x_c$ while Equation 2.53 specifies $x_b \neq x_c$. Hence, this particular $\tau \in \mathcal{L}(G)$ is not a member of $\mathcal{M}(G)$.

2.4.3 Bethe Entropy Approximation

In general, the negative entropy $A^*(\mu)$ does not have a closed form expression in terms of μ. However, an exception to this is a tree-structured Markov random field.

Again by the junction tree theorem in [27], for a tree-structured graph, the probability distribution decomposes as follows

$$p_\mu(x) := \prod_{s \in V} \mu_s(x_s) \prod_{(s,t) \in E} \frac{\mu_{st}(x_s, x_t)}{\mu_s(x_s)\mu_t(x_t)} \tag{2.54}$$

Hence, the entropy $H(p_\mu)$ can be obtained directly.

$$\begin{aligned} H(p_\mu) &= \mathbb{E}_\mu[-\log(p_\mu(X))] \\ &= -\sum_{x \in \mathcal{X}} p_\mu(x) \log(p_\mu(x)) \\ &= -\sum_{s \in V} \sum_{x_s \in \mathcal{X}_s} \mu_s(x_s) \log(\mu_s(x_s)) - \sum_{(s,t) \in E} \sum_{(x_s, x_t) \in \mathcal{X}_s \times \mathcal{X}_t} \mu_s(x_s, x_t) \log\left(\frac{\mu_{st}(x_s, x_t)}{\mu_s(x_s)\mu_t(x_t)}\right) \\ &= \sum_{s \in V} H_s(\mu_s) - \sum_{(s,t) \in E} I_{st}(\mu_{st}) \end{aligned} \tag{2.55}$$

where H_s denotes the entropy at each vertex, and I_{st} denotes the mutual information at each edge according to μ.

This value is exactly equal to $A^*(\mu)$. For graphs with cycles, we may assume it is a valid approximation. Hence, the **Bethe entropy approximation** for a test function τ is defined as

$$-A^*_{\text{Bethe}}(\tau) \approx H_{\text{Bethe}}(\tau) := \sum_{s \in V} H_s(\tau_s) - \sum_{(s,t) \in E} I_{st}(\tau_{st}) \tag{2.56}$$

This value can be calculated for any $\tau \in \mathbb{L}(G)$.

Inexactness of Bethe Approximation

We use an example here to demonstrate the inexactness of the Bethe approximation.

Consider a fully connected graph $G = (V, E)$ with 3 vertices and 3 edges. Let

$$\mu_s(x_s) = \begin{pmatrix} \frac{1}{4} & \frac{3}{4} \end{pmatrix} \ \forall s \in V \tag{2.57}$$

$$\mu_{st}(x_s, x_t) = \begin{pmatrix} \frac{1}{4} & 0 \\ 0 & \frac{3}{4} \end{pmatrix} \ \forall (s,t) \in E \tag{2.58}$$

then we can verify that this specifies a valid probability distribution over the whole graph which places probability mass $\frac{1}{4}$ on the outcome $(0,0,0)$, and $\frac{3}{4}$ on the outcome $(1,1,1)$. Let $a = \frac{1}{4}\log(4) + \frac{3}{4}\log\left(\frac{4}{3}\right)$, then, the entropy $H_s(\mu_s) = h$ for all $s \in V$, and $I_{st}(\mu_{st}) = h$ for all $(s,t) \in E$. Also, the entropy of the overall distribution p_μ is also h.

Using the Bethe approximation, we calculate the Bethe entropy to be $H_{\text{Bethe}}(p_\mu) = 3H_s(\mu_s) - 3I_{st}(\mu_{st}) = 3a - 3a = 0$. Clearly, this is not equal to $H(p_\mu)$. In fact, the entropy of the distribution should not be 0 unless an outcome has probability mass 1.

2.4.4 Bethe Variation Problem

Recall that according to the original formulation, the variational problem for finding $A(\theta)$ is

$$\max_{\mu \in \mathcal{M}(G)} \left[\langle \theta, \mu \rangle + A^*(\mu)\right] \tag{2.59}$$

With the outer bound $\mathcal{L}(G)$ over the marginal polytope $\mathcal{M}(G)$, and the approximation $A^*_{\text{Bethe}}(\tau)$ for $A^*(\tau)$, we have the **Bethe variational problem** [27]

$$\max_{\tau \in \mathcal{L}(G)} \left[\langle \theta, \tau \rangle + \sum_{s \in V} H_s(\tau_s) - \sum_{(s,t) \in E} I_{st}(\tau_{st})\right] \tag{2.60}$$

It turns out that the sum-product algorithm from before is an algorithm that solves this problem.

We first formulate a Lagrangian corresponding to this problem. The normalisation constraint for each $s \in V$ can be expressed as

$$C_{ss}(\tau) := 1 - \sum_{x_s} \tau_s(x_s) = 0 \tag{2.61}$$

The marginalisation constraint for each direction $t \to s$ on each edge, and each x_s value for the s vertex can be expressed as

$$C_{ts}(x_s; \tau) := \tau_s(x_s) - \sum_{x_t} \tau_{st}(x_s, x_t) = 0 \tag{2.62}$$

Let λ_{ss} and $\lambda_{ts}(x_s)$ be the corresponding Lagrange multipliers, then the Lagrangian of the problem is as follows

$$\begin{aligned} \mathcal{L}(\tau, \lambda; \theta) =& \langle \theta, \tau \rangle + \sum_{s \in V} H_s(\tau_s) - \sum_{(s,t) \in E} I_{st}(\tau_{st}) + \sum_{s \in V} \lambda_{ss} C_{ss}(\tau) \\ &+ \sum_{(s,t) \in E} \left[\sum_{x_s} \lambda_{ts}(x_s) C_{ts}(x_s; \tau) + \sum_{x_t} \lambda_{ts}(x_t) C_{ts}(x_t; \tau)\right] \end{aligned} \tag{2.63}$$

2.4.5 Sum-Product Algorithm for Bethe Variation Problem

Given the Lagrangian in Equation 2.63, taking the derivative with respect λ and setting it to 0 gives the normalisation and marginalisation constraints given in Equations 2.61 and 2.62.

Now we shall take the derivative of the Lagrangian with respect to τ. First, consider a given s and given x_s, then

$$\begin{aligned}\frac{\partial}{\partial \tau_s(x_s)}\mathcal{L}(\tau,\lambda;\theta) =& \frac{\partial}{\partial \tau_s(x_s)}\Bigg(\theta_s(x_s)\tau_s(x_s) - \tau_s(x_s)\log(\tau_s(x_s)) \\ & - \lambda_{ss}\tau_s(x_s) + \sum_{t\in N(s)} \lambda_{ts}(x_s)\left(\tau_s(x_s)\right)\Bigg) \\ =& \theta_s(x_s) - \log(\tau_s(x_s)) - 1 \\ & - \lambda_{ss} + \sum_{t\in N(s)} \lambda_{ts}(x_s)\end{aligned} \tag{2.64}$$

$$\begin{aligned}\frac{\partial}{\partial \tau_{st}(x_s,x_t)}\mathcal{L}(\tau,\lambda;\theta) =& \frac{\partial}{\partial \tau_{st}(x_s,x_t)}\Bigg(\theta_{s,t}(x_s,x_t)\tau_{s_t}(x_s,x_t) \\ & - \tau_{st}(x_s,x_t)\log\left(\frac{\tau_{st}(x_s,x_t)}{\tilde{\tau}_s(x_s)\tilde{\tau}_t(x_t)}\right) \\ & - \lambda_{ts}(x_s)\tau_{st}(x_s,x_t) - \lambda_{st}(x_t)\tau_{st}(x_s,x_t)\Bigg) \\ =& \theta_{s,t}(x_s,x_t) \\ & - \log\left(\frac{\tau_{st}(x_s,x_t)}{\tilde{\tau}_s(x_s)\tilde{\tau}_t(x_t)}\right) - 1 \\ & - \lambda_{ts}(x_s) - \lambda_{st}(x_t)\end{aligned} \tag{2.65}$$

where $\tilde{\tau}_s(x_s) := \sum_{x_t} \tau_{st}(x_s,x_t)$.

Setting these to 0 gives some relations involving τ_s and τ_{st} terms. There are some descrepancies between the differentiation result here and the results given in [27], but since they only differ by constants, the descrepancies can be resolved by redefining some normalisation constant λ.

The given results are easier to work with, and they are as follows

$$\log(\tau_s(x_s)) = \lambda_{ss} + \theta_s(x_s) + \sum_{t\in N(s)} \lambda_{ts}(x_x) \tag{2.66}$$

$$\log\left(\frac{\tau_{st}(x_s,x_t)}{\tilde{\tau}_s(x_s)\tilde{\tau}_t(x_t)}\right) = \theta_{st}(x_s,x_t) - \lambda_{ts}(x_s) - \lambda_{st}(x_t) \tag{2.67}$$

where $\tilde{\tau}_s(x_s) := \sum_{x_t} \tau_{st}(x_s,x_t)$. Since the marginalisation constraint holds at the solution to the Lagrangian, we have $\tilde{\tau}_s(x_s) = \tau_s(x_s)$. Hence

$$\log(\tau_{st}(x_s,x_t) = \theta_{st}(x_s,x_t) - \lambda_{ts}(x_s) - \lambda_{st}(x_t) + \log(\tau_s(x_s)) + \log(\tau_t(x_t)) \tag{2.68}$$

We can substitute Equation 2.66 into this to obtain

$$\log(\tau_{st}(x_s,x_t)) = \lambda_{ss} + \lambda_{tt} + \theta_{st}(x_s,x_t) + \theta_s(x_s) + \theta_t(x_t) + \sum_{u\in N(s)\setminus\{t\}} \lambda_{us}(x_s) + \sum_{u\in N(t)\setminus\{s\}} \lambda_{ut}(x_t) \tag{2.69}$$

Defining $M_{ts}(x_s) = \exp(\lambda_{ts}(x_s))$ for each direction $t \to s$ on each edge, then we can rewrite Equations 2.66 and 2.69 as

$$\tau_s(x_s) = \kappa_s \exp(\theta_s(x_s)) \prod_{t\in N(s)} M_{ts}(x_s) \tag{2.70}$$

$$\tau_{st}(x_s, x_t) = \kappa_{st}\exp(\theta_{st}(x_s, x_t) + \theta_s(x_s) + \theta_t(x_t)) \prod_{u \in N(s)\setminus\{t\}} M_{us}(x_s) \prod_{u \in N(t)\setminus\{s\}} M_{ut}(x_t) \quad (2.71)$$

where κ_s and κ_{st} are related to λ_{ss} and λ_{tt} terms in the original equations. The appropriate values are chosen such that the normalisation and marginalisation constraints are satisfied.

Consider the marginalisation constraint $\sum_{x_t} \tau_{st}(x_s, x_t) = \tau_s(x_s)$. If we substitute Equations 2.70 and 2.71 into this constraint, we get

$$\begin{aligned}
&\sum_{x_t} \left[\kappa_{st}\exp(\theta_{st}(x_s, x_t) + \theta_s(x_s) + \theta_t(x_t)) \prod_{u \in N(s)\setminus\{t\}} M_{us}(x_s) \prod_{u \in N(t)\setminus\{s\}} M_{ut}(x_t) \right] \\
=&\kappa_{st}\exp(\theta_s(x_s)) \left(\prod_{u \in N(s)\setminus\{t\}} M_{us}(x_s) \right) \sum_{x_t} \left[\exp(\theta_{st}(x_s, x_t) + \theta_t(x_t)) \prod_{u \in N(t)\setminus\{s\}} M_{ut}(x_t) \right] \\
=&\kappa_s\exp(\theta_s(x_s)) \prod_{t \in N(s)} M_{ts}(x_s)
\end{aligned} \quad (2.72)$$

Cancelling out the terms that are present on both sides of this equation, we get

$$M_{ts}(x_s) \propto \sum_{x_t} \left[\exp(\theta_{st}(x_s, x_t) + \theta_t(x_t)) \prod_{u \in N(t)\setminus\{s\}} M_{ut}(x_t) \right] \quad (2.73)$$

This is the same sum-product algorithm as discussed before. Hence, by construction, we know any fixed point M^* would specify a pair (τ^*, λ^*) that satisfy the stationary conditions in the Lagrangian.

2.5 Quantum Information

The following are some definitions related to the reduced density operator from [14].

2.5.1 Density Operator

Suppose the possible states for a quantum state to be in is $|\psi_i\rangle$, where i is an index, and suppose that the probability that it is in state $|\psi_i\rangle$ is p_i. Then, the **density operator** is defined as

$$\rho := \sum_i p_i |\psi_i\rangle\langle\psi_i| \quad (2.74)$$

The diagonal values of ρ represents the probability that the system is in each corresponding state, which means we require the normalisation $\mathrm{Tr}(\rho) = 1$.

2.5.2 Reduced Density Operator

Consider two quantum systems A and B, which is described overall by density operator ρ^{AB}, then the **reduced density operator** over A is defined as

$$\rho^A := \mathrm{Tr}_B(\rho^{AB}) \quad (2.75)$$

where the **partial trace** Tr_B is defined by

$$\mathrm{Tr}_B(|a_1\rangle\langle a_2| \otimes |b_1\rangle\langle b_2|) = |a_1\rangle\langle a_2|\mathrm{Tr}(|b_1\rangle\langle b_2|) \quad (2.76)$$

For example, suppose the basis states are $|0\rangle$ and $|1\rangle$, and the matrix over A and B is M_{AB}, then the partial trace can be computed as

$$\mathrm{Tr}_A(M_{AB}) = (\langle 0| \otimes \mathbb{1})M_{AB}(|0\rangle \otimes \mathbb{1}) + (\langle 1| \otimes \mathbb{1})M_{AB}(|1\rangle \otimes \mathbb{1}) \quad (2.77)$$

2.5.3 Von Neumann Entropy

Similary to Shannon Entropy for discrete probability distributions, where

$$H(X) = \mathbb{E}[p(x)\log(p(x))] \tag{2.78}$$

the Von Neumann entropy is defined for quantum states, where the entropy of the state ρ is

$$S(\rho) = -\mathrm{Tr}(\rho\log(\rho)) \tag{2.79}$$

Suppose λ_x are the eigenvalues of ρ, then

$$S(\rho) = -\sum_x \lambda_x \log(\lambda_x) \tag{2.80}$$

2.5.4 Thermal Equilibrium State

The thermal equilibrium state [7] will be the starting point of the quantum belief propagation algorithm in this project. It is defined as

$$\rho = \frac{\exp(-\beta H)}{Z} \tag{2.81}$$

where H is the Hamiltonian, $\beta = \frac{1}{k_B T}$ and $Z = \mathrm{Tr}(\exp[-\beta H])$ is the normalising factor.

We notice that this formulation is very similar to the formulation in exponential families.

Chapter 3

Ethical Issues

3.1 Misuse

As is the case in any studies related to cutting-edge technology, there are people and organisations that are interested in using the technology for purposes that are harmful to the general public.

Studying the reduced density matrix does not have any direct dangers of misuse, but the wide array of fields that can benefit from the studies have practical applications with significant impacts. While the vast majority of such applications are beneficial, they can be misused in harmful ways.

3.2 Environmental Impact

During the benchmarking process of the project, I will be running resource-heavy computations, which may contribute to negative environmental impact.

Chapter 4

Derivation

4.1 Thermal Equilibrium State as Exponential Family

Recall that at thermal equilibrium, the density matrix ρ of a system is

$$\rho = \frac{\exp(-\beta H)}{Z} \tag{4.1}$$

where $Z = \mathrm{Tr}(\exp(-\beta H))$.

It is helpful to reinterpret this equation as an exponential family in the form of

$$\rho = \exp\left(\sum_k \theta_k \phi_k - A(\theta)\right) \tag{4.2}$$

similar to Equation 2.18 which describes the classical probability model, where ϕ_k are a series of "sufficient statistics", and θ_k are the canonical parameters associated with these statistics.

4.1.1 Single Qubit

First, we consider a system with only 1 qubit, where the Hamiltonian H is a 2×2 Hermitian matrix. The space of 2×2 Hermitian matrices is spanned by the identity matrix and the Pauli matrices [14], which are

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{4.3}$$

We will denote the identity matrix as $\sigma_0 = \mathbb{1}$ for notational uniformity. Then, the matrices $(\sigma_x, \sigma_y, \sigma_z, \sigma_0)$ make a suitable set of sufficient statistics. As H can be expanded in the basis written as

$$H = a\sigma_x + b\sigma_y + c\sigma_z + d\sigma_0 \tag{4.4}$$

we can rewrite $\exp(-\beta H)$ as

$$\exp(-\beta H) = \exp(\theta_x \sigma_x + \theta_y \sigma_y + \theta_z \sigma_z + \theta_0 \sigma_0) \tag{4.5}$$

where $\theta_x = -\beta a$, $\theta_y = -\beta b$, $\theta_z = -\beta c$, and $\theta_0 = -\beta d$. Hence, Z becomes

$$Z = \mathrm{Tr}(\exp(-\beta H)) = \mathrm{Tr}[\exp(\theta_x \sigma_x + \theta_y \sigma_y + \theta_z \sigma_z + \theta_0 \sigma_0)] \tag{4.6}$$

Due to the commutativity of the identity matrix with all other matrices, we can write

$$\exp(-\beta H) = e^{\theta_0} \exp(\theta_x \sigma_x + \theta_y \sigma_y + \theta_z \sigma_z) \tag{4.7}$$

and

$$Z = \mathrm{Tr}(\exp(-\beta H)) = e^{\theta_0} \mathrm{Tr}[\exp(\theta_x \sigma_x + \theta_y \sigma_y + \theta_z \sigma_z)] \tag{4.8}$$

which means the θ_0 contribution disappears due to the normalisation factor in

$$\rho = \frac{\exp(-\beta H)}{Z} = \frac{\exp(\theta_x\sigma_x + \theta_y\sigma_y + \theta_z\sigma_z)}{\mathrm{Tr}[\exp(\theta_x\sigma_x + \theta_y\sigma_y + \theta_z\sigma_z)]} \tag{4.9}$$

Hence, we can drop the identity matrix and only use the Pauli matrices $\{\sigma_x, \sigma_y, \sigma_z\}$ as the sufficient statistics.

Now we can write

$$\rho = \exp\left(\sum_{k\in\mathbb{P}} \theta_k\sigma_k - A(\theta)\right) \tag{4.10}$$

where $\mathbb{P} = \{x, y, z\}$ and

$$A(\theta) = \log\left(\mathrm{Tr}\left(\exp\left(\sum_{k\in\mathbb{P}} \theta_k\sigma_k\right)\right)\right) \tag{4.11}$$

4.1.2 Many Qubits

Consider a system with n qubits, then the Hamiltonian

$$H = H_1 \otimes H_2 \otimes ... \otimes H_n \tag{4.12}$$

where H_i is the Hamiltonian of the i-th qubit.

We denote $\sigma_{(k_1,k_2,...,k_n)} = \sigma_{k_1} \otimes \sigma_{k_2} \otimes ... \otimes \sigma_{k_n}$, the tensor products of n Pauli or identity matrices, which has dimensions $2^n \times 2^n$. We also denote $\mathbb{P}^* = \{x, y, z, 0\}$. Then $\{\sigma_k | k \in \mathbb{P}^{*n}\}$ forms a basis of an n-qubit system due to the properties of the tensor product.

Clearly, the $2^n \times 2^n$ identity matrix is in this basis. However, similar to the single qubit case, its contribution will disappear due to the normalisation factor. Therefore, we designate the sufficient statistics as the basis of the Hamiltonian excluding the identity matrix. For notational simplicity, we can define $\mathbb{P}_n$ as $\mathbb{P}^{*n}\backslash\{\mathbf{0}\}$. The identity matrix corresponds to the removed element $\mathbf{0} = (0, 0, ..., 0)$ where every component is 0.

Then we can write

$$\rho = \exp\left(\sum_{k\in\mathbb{P}_n} \theta_k\sigma_k - A(\theta)\right) \tag{4.13}$$

where

$$A(\theta) = \log\left(\mathrm{Tr}\left(\exp\left(\sum_{k\in\mathbb{P}_n} \theta_k\sigma_k\right)\right)\right) \tag{4.14}$$

This covers the single qubit case as well by recognising $\mathbb{P}_1 = \mathbb{P}$.

These are very similar to the Equation 2.18 and 2.15, which are the classical exponential family and log-partition function. Notably, the sum in the log partition function is replaced with the trace of the matrix. Since the diagonal values of the probability matrix represent the probability of the qubits being in different states, this makes sense as the trace is the sum of the diagonal values.

4.2 Mean Parameter and the Conjugate Dual

4.2.1 Mean Parameter

In Section 2.3.3 in the classical case, we established that differentiating the log-partition function leads to the mean parameters of the exponential family.

We can differentiate Equation 4.14 to get

$$\begin{aligned}\frac{\partial}{\partial\theta_j}A(\theta) &= \log\left(\mathrm{Tr}\left(\exp\left(\sum_{k\in\mathbb{P}_n}\theta_k\sigma_k\right)\right)\right)\\ &= \frac{\mathrm{Tr}\left(\frac{\partial}{\partial\theta_j}\exp\left(\sum_{k\in\mathbb{P}_n}\theta_k\sigma_k\right)\right)}{\mathrm{Tr}\left(\exp\left(\sum_{k\in\mathbb{P}_n}\theta_k\sigma_k\right)\right)}\\ &= \frac{1}{Z}\mathrm{Tr}\left(\frac{\partial}{\partial\theta_j}\exp\left(\sum_{k\in\mathbb{P}_n}\theta_k\sigma_k\right)\right)\end{aligned} \tag{4.15}$$

In order to find out the derivative

$$\frac{\partial}{\partial\theta_j}\exp\left(\sum_{k\in\mathbb{P}_n}\theta_k\sigma_k\right) \tag{4.16}$$

we quote the following formula from [29]

$$\frac{\partial}{\partial\lambda}e^{-\beta H} = -\int_0^\beta e^{-(\beta-u)H}\frac{\partial H}{\partial\lambda}e^{-uH}\,\mathrm{d}u \tag{4.17}$$

This can be adapted to our derivative by denoting $H' = \sum_{k\in\mathbb{P}_n}\theta_k\sigma_k$, noting that this is not the Hamiltonian H, but rather $-\beta H$, then

$$\frac{\partial}{\partial\theta_j}\exp(H') = \int_0^1 e^{(1-u)H'}\sigma_j e^{uH'}\,\mathrm{d}u \tag{4.18}$$

so

$$\begin{aligned}\frac{\partial}{\partial\theta_j}A(\theta) &= \frac{1}{Z}\mathrm{Tr}\left(\int_0^1 e^{(1-u)H'}\sigma_j e^{uH'}\,\mathrm{d}u\right)\\ &= \frac{1}{Z}\int_0^1\mathrm{Tr}\left(e^{(1-u)H'}\sigma_j e^{uH'}\right)\mathrm{d}u\\ &= \frac{1}{Z}\int_0^1\mathrm{Tr}\left(e^{uH'}e^{(1-u)H'}\sigma_j\right)\mathrm{d}u\\ &= \frac{1}{Z}\mathrm{Tr}\left(\exp\left[\sum_{k\in\mathbb{P}_n}\theta_k\sigma_k\right]\sigma_j\right)\\ &= \mathrm{Tr}\left(\frac{\exp\left[\sum_{k\in\mathbb{P}_n}\theta_k\sigma_k\right]}{Z}\sigma_j\right)\\ &= \mathrm{Tr}(\rho\sigma_j)\end{aligned} \tag{4.19}$$

where we have used the cyclic property of trace to rearrange the integrand inside the trace.

This is indeed the expectation value of the operator corresponding to the matrix σ_j, since

$$\mathrm{Tr}(\rho\sigma_j) = \mathrm{Tr}\left(\sum_i p_i|\psi_i\rangle\langle\psi_i|\sigma_j\right) = \sum_n\sum_i\langle n|p_i|\psi_i\rangle\langle\psi_i|\sigma_j|n\rangle = \sum_i p_i\langle\psi_i|\sigma_j|\psi_i\rangle = \langle\sigma_j\rangle \tag{4.20}$$

For each k, we can now define the mean parameter $\mu_k = \mathrm{Tr}(\rho\sigma_k)$.

4.3 Convexity

For a variational algorithm to work, we need to verify the convexity of the log-partition fuction $A(\theta)$. This can be done via verifying that the Hessian $\nabla^2 A$ is positive semi-definite.

We now compute second derivative of $A(\theta)$

$$\begin{aligned}\frac{\partial}{\partial\theta_i}\frac{\partial}{\partial\theta_j}A(\theta) &= \frac{\partial}{\partial\theta_i}\left[\frac{\mathrm{Tr}\left(\frac{\partial}{\partial\theta_j}\exp\left(\sum_{k\in\mathbb{P}_n}\theta_k\sigma_k\right)\right)}{\mathrm{Tr}\left(\exp\left(\sum_{k\in\mathbb{P}_n}\theta_k\sigma_k\right)\right)}\right] \\ &= \frac{1}{Z}\mathrm{Tr}\left(\frac{\partial}{\partial\theta_i}\frac{\partial}{\partial\theta_j}\exp\left(\sum_{k\in\mathbb{P}_n}\theta_k\sigma_k\right)\right) \\ &- \frac{1}{Z^2}\mathrm{Tr}\left(\frac{\partial}{\partial\theta_i}\exp\left(\sum_{k\in\mathbb{P}_n}\theta_k\sigma_k\right)\right)\mathrm{Tr}\left(\frac{\partial}{\partial\theta_j}\exp\left(\sum_{k\in\mathbb{P}_n}\theta_k\sigma_k\right)\right)\end{aligned} \tag{4.21}$$

Here, the second term is simply $-\mathrm{Tr}(\rho\sigma_i)\mathrm{Tr}(\rho\sigma_j)$ using the previous result from the first derivative in Equation 4.19. Again, letting $H' = \sum_{k\in\mathbb{P}_n}\theta_k\sigma_k$, the first term becomes

$$\begin{aligned}&\frac{1}{Z}\mathrm{Tr}\left(\frac{\partial}{\partial\theta_i}\int_0^1 e^{(1-u)H'}\sigma_j e^{uH'}\mathrm{d}u\right) \\ =&\frac{1}{Z}\frac{\partial}{\partial\theta_i}\int_0^1\mathrm{Tr}\left(\sigma_j e^{uH'}e^{(1-u)H'}\right)\mathrm{d}u \\ =&\frac{1}{Z}\frac{\partial}{\partial\theta_i}\int_0^1\mathrm{Tr}\left(\sigma_j e^{H'}\right)\mathrm{d}u \\ =&\frac{1}{Z}\frac{\partial}{\partial\theta_i}\mathrm{Tr}\left(\sigma_j e^{H'}\right) \\ =&\frac{1}{Z}\mathrm{Tr}\left(\int_0^1\sigma_j e^{(1-u)H'}\sigma_i e^{uH'}\mathrm{d}u\right)\end{aligned} \tag{4.22}$$

where we have used the cyclic property and the formula for differentiating matrix exponential again. So

$$\frac{\partial}{\partial\theta_i}\frac{\partial}{\partial\theta_j}A(\theta) = \frac{1}{Z}\mathrm{Tr}\left(\int_0^1\sigma_j e^{(1-u)H'}\sigma_i e^{uH'}\mathrm{d}u\right) - \mathrm{Tr}(\rho\sigma_i)\mathrm{Tr}(\rho\sigma_j) \tag{4.23}$$

Since the integral is symmetric, this means that $\nabla^2 A$ is real and symmetric. Now to show that it is positive semi-definite, we need to show that for any $v\in\mathbb{R}^N$, where N is the cardinality of $\mathbb{P}_n$, $v^T(\nabla^2 A)v \geq 0$, or equivalently

$$\sum_{ij} v_i v_j \frac{\partial}{\partial\theta_i}\frac{\partial}{\partial\theta_j}A(\theta) \geq 0 \tag{4.24}$$

We have

$$\begin{aligned}&\sum_{ij} v_i v_j \frac{\partial}{\partial\theta_i}\frac{\partial}{\partial\theta_j}A(\theta) \\ =&\sum_{ij} v_i v_j\left[\frac{1}{Z}\mathrm{Tr}\left(\int_0^1\sigma_j e^{(1-u)H'}\sigma_i e^{uH'}\mathrm{d}u\right) - \mathrm{Tr}(\rho\sigma_i)\mathrm{Tr}(\rho\sigma_j)\right] \\ =&\sum_{ij}\frac{1}{Z}\mathrm{Tr}\left(\int_0^1 v_j\sigma_j e^{(1-u)H'}v_i\sigma_i e^{uH'}\mathrm{d}u\right) \\ &-\sum_{ij}\frac{1}{Z^2}\mathrm{Tr}(e^{H'}v_i\sigma_i)\mathrm{Tr}(e^{H'}v_j\sigma_j)\end{aligned} \tag{4.25}$$

From here we notice the terms $\sum_k v_k\sigma_k$ are similar in construction to $H' = \sum_k\theta_k\sigma_k$, so we can denote them as $H'(v) = \sum_k v_k\sigma_k$, and the original H' as $H'(\theta)$ to increase clarity. Then, the above equation can be written as

$$\begin{aligned}&\frac{1}{Z}\mathrm{Tr}\left(\int_0^1 H'(v)e^{(1-u)H'(\theta)}H'(v)e^{uH'(\theta)}\mathrm{d}u\right) \\ &-\frac{1}{Z^2}\mathrm{Tr}(e^{H'(\theta)}H'(v))^2\end{aligned} \tag{4.26}$$

Let $|m\rangle$ denote the eigenbasis of $H'(\theta)$ with eigenvalues E_m. Then, let $r_m = \frac{e^{E_m}}{Z}$, $f_m = \langle m|H'(v)|m\rangle$. We can calculate

$$\begin{aligned}
&\mathrm{Tr}\Big(H'(v)e^{(1-u)H'(\theta)}H'(v)e^{H'(\theta)}\Big)\\
=&\sum_n \langle n|H'(v)e^{(1-u)H'(\theta)}H'(v)e^{uH'(\theta)}|n\rangle\\
=&\sum_{m,n} \langle n|H'(v)e^{(1-u)H'(\theta)}|m\rangle\langle m|H'(v)e^{uH'(\theta)}|n\rangle\\
=&\sum_{m,n} e^{E_m}e^{u(E_n-E_m)}\langle n|H'(v)|m\rangle\langle m|H'(v)|n\rangle\\
=&\sum_{m,n} e^{E_m}e^{u(E_n-E_m)}|\langle n|H'(v)|m\rangle|^2
\end{aligned} \tag{4.27}$$

and

$$\begin{aligned}
&\mathrm{Tr}\Big(e^{H'(\theta)}H'(v)\Big)\\
=&\sum_m \langle m|e^{H'(\theta)}H'(v)|m\rangle\\
=&\sum_m e^{E_m}\langle m|H'(v)|m\rangle
\end{aligned} \tag{4.28}$$

Then, Equation 4.26 can be rewritten as

$$\sum_{m,n} r_m|\langle n|H'(v)|m\rangle|^2\int_0^1 e^{u(E_n-E_m)}\mathrm{d}u - \left(\sum_m r_m f_m\right)^2 \tag{4.29}$$

Since $\int_0^1 e^{u(E_n-E_m)}\mathrm{d}u \geq 0$,

$$\begin{aligned}
&\sum_{m,n} r_m|\langle n|H'(v)|m\rangle|^2\int_0^1 e^{u(E_n-E_m)}\mathrm{d}u\\
\geq&\sum_m r_m|\langle m|H'(v)|m\rangle|^2\int_0^1 e^{u(E_m-E_m)}\mathrm{d}u\\
=&\sum_m r_m f_m^2
\end{aligned} \tag{4.30}$$

So continuing from Equation 4.29

$$\begin{aligned}
&\sum_{m,n} r_m|\langle n|H'(v)|m\rangle|^2\int_0^1 e^{u(E_n-E_m)}\mathrm{d}u - \left(\sum_m r_m f_m\right)^2\\
\geq&\sum_m r_m|\langle m|H'(v)|m\rangle|^2\int_0^1 e^{u(E_m-E_m)}\mathrm{d}u - \left(\sum_m r_m f_m\right)^2\\
=&\sum_m r_m f_m^2 - \left(\sum_m r_m f_m\right)^2 \geq 0
\end{aligned} \tag{4.31}$$

since this is equivalent to the variance of a random variable.

This shows that the sum is indeed positive hence $A(\theta)$ is indeed convex.

4.4 Conjugate Dual and von Neumann Entropy

Given the fact that $A(\theta)$ is convex and the mean parameters defined as $\mu_k = \mathrm{Tr}(\rho\sigma_k)$, now we can define the conjugate dual of the log partition function similar to Equation 2.34 in the classical case.

$$A^*(\mu) = \sup_\theta\left\{\sum_{k\in\mathbb{P}_n}\theta_k\mu_k - A(\theta)\right\} \tag{4.32}$$

Due to the property of the conjugate dual as discussed in Equation 2.36, this implies

$$A(\theta) = \sup_{\mu} \left\{ \sum_{k \in \mathbb{P}_n} \theta_k \mu_k - A^*(\mu) \right\} \tag{4.33}$$

In the classical graphical model, we discovered that conjugate dual function is equal to the negative Shannon entropy. In this case, it is equivalent to the negative von Neumann Entropy.

Proof. Suppose $\theta = \theta(\mu)$ satisfies the optimisation condition in Equation 4.32. Then, we can construct the density matrix ρ as $\rho = \rho(\theta) = \exp\left(\sum_{k \in \mathbb{P}_n} \theta_k \sigma_k - A(\theta)\right)$. Then the negative von Neumann entropy of ρ is

$$\begin{aligned} -S(\rho) &= \mathrm{Tr}(\rho \log(\rho)) \\ &= \mathrm{Tr}\left(\rho\left(\sum_{k \in \mathbb{P}_n} \theta_k \sigma_k - A(\theta)\right)\right) \\ &= \mathrm{Tr}\left(\rho\left(\sum_{k \in \mathbb{P}_n} \theta_k \sigma_k\right)\right) - A(\theta) \\ &= \sum_{k \in \mathbb{P}_n} \theta_k \mathrm{Tr}(\rho \sigma_k) - A(\theta) \\ &= \sum_{k \in \mathbb{P}_n} \theta_k \mu_k - A(\theta) \\ &= A^*(\mu) \end{aligned} \tag{4.34}$$

□

Inserting this into Equation 4.33, we get

$$A(\theta) = \sup_{\mu} \left\{ \sum_{k \in \mathbb{P}_n} \theta_k \mu_k + S(\rho) \right\} \tag{4.35}$$

4.5 Approximation Methods

4.5.1 Setup

Obtaining and processing information about overall quantum systems becomes difficult as the size of H and ρ grows exponentially. Like the classical belief propagation on graphical models, quantum belief propagation relies on solving a relaxed problem to get an approximate solution of the information about the systems. Also like the classical case, we can work with approximating the entropy and relaxing the global consistency conditions on the density matrix to local consistency conditions.

For this approximation to work well, the interactions should be short-ranged [18]. For the purpose of this particular project, we focus on systems involving only interactions between particles that are next to each other, e.g. on a chain or lattice. This can be compared to the pair-wise Markov random field in the context of classical probability.

What this means for the optimisation problem is that all θ_k values involving more than 2 particles will be 0. We can define a new set $\mathbb{P}'_n \subset \mathbb{P}_n$, which includes elements that have either 1 non-zero entry, or 2 entries at positions corresponding to 2 particles that have interactions with each other. Then we can state the problem as

$$A(\theta) = \sup_{\mu} \left\{ \sum_{k \in \mathbb{P}'_n} \theta_k \mu_k + S(\rho) \right\} \tag{4.36}$$

4.5.2 Entropy Approximation

The strong subadditivity theorem [14] for von Neumann entropy states that, for particles s_1, s_2, and s_3 in general,

$$S(s_1, s_2, s_3) + S(s_2) \leq S(s_1, s_2) + S(s_2, s_3) \tag{4.37}$$

The corresponding mutual information is defined [14] as

$$S(s_1 : s_2) = S(s_1) + S(s_2) - S(s_1, s_2) \tag{4.38}$$

So the strong subadditivity theorem implies

$$\begin{aligned} S(s_1, s_2, s_3) &\leq S(s_1, s_2) + S(s_2, s_3) - S(s_2) \\ &= S(s_1) + S(s_2) - S(s_1 : s_2) + S(s_2) + S(s_3) - S(s_2 : s_3) - S(s_2) \\ &= S(s_1) + S(s_2) + S(s_3) - S(s_1 : s_2) - S(s_2 : s_3) \end{aligned} \tag{4.39}$$

The intuitive way to think about it is consider these particles as connected in a chain like shown in Figure 4.1. The particles correspond to the single-particle entropy and the edges correspond to the pair-wise mutual information.

Figure 4.1: Diagram showing 3 particles connected in a chain.

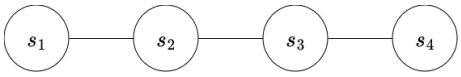

Figure 4.2: Diagram showing 4 particles connected in a chain.

Suppose we add another particle s_4 to the chain after s_3, like shown in Figure 4.2, then we can use the strong subadditivity theorem again

$$S(s_1, s_2, s_3, s_4) + S(s_3) \leq S(s_1, s_2, s_3) + S(s_3, s_4) \tag{4.40}$$

to show

$$S(s_1, s_2, s_3, s_4) \leq S(s_1) + S(s_2) + S(s_3) + S(s_4) - S(s_1 : s_2) - S(s_2 : s_3) - S(s_3 : s_4) \tag{4.41}$$

where again, the particles correspond to the single-particle entropy and the edges correspond to the pair-wise mutual information.

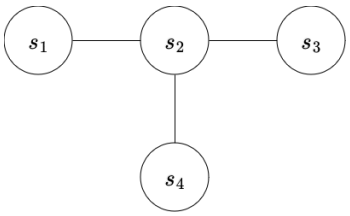

Figure 4.3: Diagram showing 4 particles connected in a tree shape.

Suppose instead that we add a particle that is connected to s_2 instead like Figure 4.3. Clearly, Equation 4.40 still holds, but since s_4 is no longer directly connected to s_3, there is no longer a direct match up between the figure and the approximation. Consequently, Equation 4.41 may not offer a good approximation, i.e. the difference between both sides of the inequality might be large.

Hence, we can apply the strong sub-additivity result differently in an attempt to get a better approximation.

$$S(s_1, s_2, s_3, s_4) + S(s_2) \leq S(s_1, s_2, s_3) + S(s_2, s_4) \tag{4.42}$$

which works out to give

$$S(s_1, s_2, s_3, s_4) \leq S(s_1) + S(s_2) + S(s_3) + S(s_4) - S(s_1 : s_2) - S(s_2 : s_3) - S(s_2 : s_4) \tag{4.43}$$

which we expect to be a better approximation is s_4 is directly connected to s_2.

We can consider this system as a graph $G = (V, E)$. Then $V = \{s_1, s_2, s_3, s_4\}$, and $E = \{(s_1, s_2), (s_2, s_3), (s_2, s_4)\}$. Then the inequality can be rewritten as

$$S(s_1, s_2, s_3, s_4) \leq \sum_{s \in V} S(s) - \sum_{(s,t) \in E} S(s : t) \tag{4.44}$$

In fact, given any general system represented by a tree-like graph $G = (V, E)$, Let $S(G)$ be the entropy of the system, we have

$$S(G) \leq \sum_{s \in V} S(s) - \sum_{(s,t) \in E} S(s : t) \tag{4.45}$$

Proof. This can be proven by induction. Suppose we have a system represented by $G_0 = (V_0, E_0)$, which satisfies

$$S(G_0) \leq \sum_{s \in V_0} S(s) - \sum_{(s,t) \in E_0} S(s : t) \tag{4.46}$$

Consider adding another particle represented by a new vertex s', which is connected to 1 particle in the system represented by $u \in V_0$. Let the resulting graph be $G' = (V', E')$ where $V' = V \cup \{s'\}$ and $E' = E \cup \{(u, s')\}$. Then by strong sub-additivity

$$S(G') + S(s') \leq S(G_0) + S(u, s') \tag{4.47}$$

which works out to give

$$S(G') \leq \sum_{s \in V'} S(s) - \sum_{(s,t) \in E'} S(s : t) \tag{4.48}$$

This means that starting from a chain of 3 particles, i.e. the base case shown in Equation 4.39, we can iteratively add particles that interact with exactly 1 particle within the system, and all resulting systems will be governed by the entropy approximation in Equation 4.48. In effect, this includes all tree like structures. □

This result is very similar to the Shannon entropy approximation in classical graphical algorithms. In the classical case the relationship is strictly equal when there are only pair-wise interactions represented by edges on the tree, and here we only have an upper bound provided by the inequality. This is due to the non-commutativity of the matrices involved in the entropy calculation.

Now consider the scenario based on the chain in Figure 4.1, but we add another edge connecting s_4 and s_1, which forms a loop. Clearly, the strong subadditivity still applies, hence Equation 4.41 still holds for this system. However, this ignores the edge (s_4, s_1), which means setting up an algorithm from this system with this inequality is difficult. So we can use

$$S(s_1, s_2, s_3, s_4) \approx S(s_1) + S(s_2) + S(s_3) + S(s_4) - S(s_1 : s_2) - S(s_2 : s_3) - S(s_3 : s_4) - S(s_4 : s_1) \tag{4.49}$$

instead.

We can choose to deal with loops in the graphs in general like this, and apply equation 4.48 to all graphs, except the inequality doesn't necessarily hold anymore.

$$S(G') \approx \sum_{s \in V'} S(s) - \sum_{(s,t) \in E'} S(s : t) \tag{4.50}$$

Like the classical case, this is a heuristic based on the structure of the system and doesn't have a lower bound or a theoretical guarantee that the approximation will be good. However, as will be shown later, this proves to be a useful heuristic in practice and provides some good results.

Recall in Equation 4.36, $A(\theta)$ depends on $S(\rho)$. Since ρ is the overall density matrix of the system represented by $G = (V, E)$, $S(G)$ and $S(\rho)$ represent the same value. With the entropy approximation, $A(\theta)$ becomes

$$A(\theta) = \sup_{\mu} \left\{ \sum_{k \in \mathbb{P}'_n} \theta_k \mu_k + \sum_{s \in V} S(s) - \sum_{(s,t) \in E} S(s:t) \right\} \tag{4.51}$$

4.5.3 Local Consistency

In order to perform belief propagation, we need to add some constraints on the beliefs on the reduced density matrices in the system, similarly to how we added constraints to the local probability distributions in the classical graphical model. This guarantees that the reduced density matrices can in fact be obtained from taking the partial trace of a global density matrices.

This is once again similar to the classical case, where global consistency conditions are difficult to obtain, as explained in Section 2.4.2. Global consistency conditions for density matrices is also difficult to obtain [18], so we will use local consistency conditions instead.

The most straight forward condition is normalisation. For each $s \in V$, let $\rho_s = \mathrm{Tr}_{V \setminus \{s\}}(\rho)$, i.e. tracing out all other vertices to get the reduced density matrix only for s. Then this matrix ρ_s needs to satisfy all the properties for density matrices. In particular, its trace needs to be 1.

$$\mathrm{Tr}(\rho_s) = 1 \tag{4.52}$$

Since we are working with systems with pair-wise interactions, we also need to consider pair-wise marginalisation conditions on the edges. For each $(s,t) \in E$, let $\rho_{s,t} = \mathrm{Tr}_{V \setminus \{s,t\}}(\rho)$, i.e. tracing out all vertices not belonging to the edge to get the reduced density matrix for s and t. Now, if we trace out vertex t from $\rho_{s,t}$, we would expect it to be the reduced density matrix for s, and vice versa.

$$\mathrm{Tr}_t(\rho_{s,t}) = \rho_s \tag{4.53}$$

$$\mathrm{Tr}_s(\rho_{s,t}) = \rho_t \tag{4.54}$$

4.6 Variational Formulation

4.6.1 Optimisation in terms of Density Matrices

We notice that the optimisation problem in Equation 4.51 is not expressed in terms of reduced density matrices. Since the constraints are in terms of these matrices, and we want the beliefs to be based on these matrices, we want to convert the optimisation problem to be based on them as well.

We can start with the 2nd term $\sum_{s \in V} S(s)$. Clearly, since ρ_s is the reduced density matrix for each s, $S(\rho_s)$ represents the same quantity as $S(s)$, so

$$\sum_{s \in V} S(s) = \sum_{s \in V} \mathrm{Tr}(\rho_s \log(\rho_s)) \tag{4.55}$$

Now consider the 3rd term $-\sum_{(s,t) \in E} S(s:t)$. This is the same as $S(s,t) - S(s) - S(t)$. Again, using the reduced density matrix of these corresponding subsystems, we get

$$\sum_{(s,t) \in E} S(s:t) = \sum_{(s,t) \in E} [\mathrm{Tr}(\rho_{s,t} \log(\rho_{s,t})) - \mathrm{Tr}(\tilde{\rho}_s \log(\tilde{\rho}_s)) - \mathrm{Tr}(\tilde{\rho}_t \log(\tilde{\rho}_t))] \tag{4.56}$$

where $\tilde{\rho}_s = \mathrm{Tr}_t(\rho_{s,t})$ and $\tilde{\rho}_t = \mathrm{Tr}_s(\rho_{s,t})$.

For the first term

$$\sum_{k\in\mathbb{P}'_n}\theta_k\mu_k = \sum_{k\in\mathbb{P}'_n}\theta_k \mathrm{Tr}(\rho\sigma_k) \tag{4.57}$$

since all the k values correspond to single particles or interacting pairs, the σ_k must be corresponding to one of the edges $(s,t)\in E$. If we trace out all the other particles (which σ_k doesn't act on), we get a 4×4 matrix $\mathrm{Tr}_{V\setminus\{s,t\}}(\rho)\in\mathbb{P}_2$. We denote this matrix $\sigma_{(s,t);k'}$, then $\mathrm{Tr}(\rho\sigma_k)=\mathrm{Tr}\big(\rho_{s,t}\sigma_{(s,t);k'}\big)$. This means we can rewrite the term as

$$\begin{aligned}\sum_{k\in\mathbb{P}'_n}\theta_k\mathrm{Tr}(\rho\sigma_k) &= \sum_{(s,t)\in E}\left[\sum_{k'\in\mathbb{P}_2}\theta_{(s,t);k'}\mathrm{Tr}(\rho_{s,t}\sigma_{(s,t);k'})\right]\\ &= \sum_{(s,t)\in E}\mathrm{Tr}\left[\left(\sum_{k'\in\mathbb{P}_2}\theta_{(s,t);k'}\sigma_{(s,t);k'})\right)\rho_{s,t}\right]\\ &= \sum_{(s,t)\in E}\mathrm{Tr}\Big[\hat{H}_{s,t}\rho_{s,t}\Big]\end{aligned} \tag{4.58}$$

where we have defined $\hat{H}_{s,t}=\big(\sum_{k'\in\mathbb{P}_2}\theta_{(s,t);k'}\sigma_{(s,t);k'})\big)$, which can be interpreted as the part of the $-\beta H$ acting over (s,t), where H is the Hamiltonian.

Now we need to clarify what the $\theta_{(s,t);k'}$ values are. They come from the canonical parameters θ_k. We notice that for each k that corresponds to a pair-wise interaction, there is a one-to-one correspondence in the above working, i.e. $\theta_{(s,t);k'}=\theta_k$.

However, for each k that corresponds to a single particle, we need to be more careful. For example, suppose vertex s has m neighbours, t_1 to t_m, and θ_{k_s} corresponds to a single particle action on s. Then θ_{k_s} will appear in $\theta_{(s,t_1);k'_s}$ through $\theta_{(s,t_m);k'_s}$, a total of m times. Thus, to avoid double counting and preserve the equality, we assign all these values to $\frac{1}{m}\theta_{k_s}$.

4.6.2 Lagrangian of the Variational Problem

Using results from Equation 4.55, 4.56, and 4.58, we reformulate the optimisation problem as finding the reduced density matrices that optimise

$$\begin{aligned}&\sum_{(s,t)\in E}\mathrm{Tr}\Big(\hat{H}_{s,t}\rho_{s,t}\Big)+\sum_{s\in V}\mathrm{Tr}(\rho_s\log(\rho_s))\\ &+\sum_{(s,t)\in E}[\mathrm{Tr}(\rho_{s,t}\log(\rho_{s,t}))-\mathrm{Tr}(\tilde{\rho}_s\log(\tilde{\rho}_s))-\mathrm{Tr}(\tilde{\rho}_t\log(\tilde{\rho}_t))]\end{aligned} \tag{4.59}$$

For local consistency, we can introduce a Lagrangian multiplier for each vertex for normalisation, and one for each edge for marginalisation.

For each vertex $s\in V$, we introduce λ_s to form the terms

$$\lambda_s(\mathrm{Tr}(\rho_s)-1) \tag{4.60}$$

For each edge $(s,t)\in E$ we introduce $\Lambda_{s(t)}$ and $\Lambda'_{t(s)}$ to form the terms

$$\Lambda_{s(t)}(\mathrm{Tr}_t(\rho_{s,t})-\rho_s) \tag{4.61}$$

$$\Lambda'_{t(s)}(\mathrm{Tr}_s(\rho_{s,t})-\rho_t) \tag{4.62}$$

We notices that Equations 4.61 and 4.62 are matrices rather than scalars. To turn them to scalars to work with the rest of the Lagrangian, we take the traces of these matrices to get

$$\mathrm{Tr}\big[\Lambda_{s(t)}(\mathrm{Tr}_t(\rho_{s,t})-\rho_s)\big]=\mathrm{Tr}\big[(\Lambda_{s(t)}\otimes\mathbb{1})\rho_{s,t}\big]-\mathrm{Tr}\big[\Lambda_{s(t)}\rho_s\big] \tag{4.63}$$

$$\mathrm{Tr}\Big[\Lambda'_{t(s)}(\mathrm{Tr}_s(\rho_{s,t})-\rho_t)\Big]=\mathrm{Tr}\Big[(\mathbb{1}\otimes\Lambda'_{t(s)})\rho_{s,t}\Big]-\mathrm{Tr}\Big[\Lambda'_{t(s)}\rho_t\Big] \tag{4.64}$$

This then allows us to write the full Lagrangian as

$$\begin{aligned}
&\mathcal{L}(\rho, \lambda, \Lambda, \Lambda') \\
&= \sum_{(s,t)\in E} \mathrm{Tr}\Big(\hat{H}_{s,t}\rho_{s,t}\Big) \\
&+ \sum_{s\in V} \mathrm{Tr}(\rho_s \log(\rho_s)) \\
&+ \sum_{(s,t)\in E} [\mathrm{Tr}(\rho_{s,t}\log(\rho_{s,t})) - \mathrm{Tr}(\tilde{\rho}_s\log(\tilde{\rho}_s)) - \mathrm{Tr}(\tilde{\rho}_t\log(\tilde{\rho}_t))] \\
&- \sum_{s\in V} \lambda_s(\mathrm{Tr}(\rho_s) - 1) \\
&- \sum_{(s,t)\in E} \big(\mathrm{Tr}\big[\Lambda_{s(t)} \otimes \mathbb{1})\rho_{s,t}\big] - \mathrm{Tr}\big[\Lambda_{s(t)}\rho_s\big]\big) \\
&- \sum_{(s,t)\in E} \Big(+\mathrm{Tr}\Big[(\mathbb{1} \otimes \Lambda'_{t(s)})\rho_{s,t}\Big] - \mathrm{Tr}\Big[\Lambda'_{t(s)}\rho_t\Big]\Big)
\end{aligned} \tag{4.65}$$

4.7 Belief Propagation Algorithms

4.7.1 Chain - 1D

Up until now, we have worked with a general graphic system. With Equation 4.65, we are ready to develop a belief propagation algorithm for a general graph of interactions. However, we will start by going through a special case of a chain of particles for various reasons: it is the simplest case and good for building intuitive understanding; it is a case where there are no loops, hence we can expect good convergence; and it is studied by various papers in the literature [11, 18].

Consider a chain with n particles. We can label them from 1 to n, and there will only be edges from i to $i+1$ for $1 \leq i \leq n-1$. We can also drop the target indicated in the brackets since each vertex only has one neighbour on each direction on the chain. With this relabelling, we get the chain version of Equation 4.65.

$$\begin{aligned}
\mathcal{L}(\rho, \lambda, \Lambda, \Lambda') = &\sum_{i=1}^{n-1} \mathrm{Tr}\Big(\hat{H}_{i,i+1}\rho_{i,i+1}\Big) \\
&+ \sum_{i=1}^{n} \mathrm{Tr}(\rho_i \log(\rho_i)) \\
&+ \sum_{i=1}^{n-1} [\mathrm{Tr}(\rho_{i,i+1}\log(\rho_{i,i+1})) - \mathrm{Tr}(\tilde{\rho}_i\log(\tilde{\rho}_i)) - \mathrm{Tr}(\tilde{\rho}_{i+1}\log(\tilde{\rho}_{i+1}))] \\
&- \sum_{i=1}^{n} \lambda_i(\mathrm{Tr}(\rho_i) - 1) \\
&- \sum_{i=1}^{n-1} (\mathrm{Tr}[(\Lambda_i \otimes \mathbb{1})\rho_{i,i+1}] - \mathrm{Tr}[\Lambda_i\rho_i]) \\
&- \sum_{i=1}^{n-1} \big(\mathrm{Tr}\big[(\mathbb{1} \otimes \Lambda'_{i+1})\rho_{i,i+1}\big] - \mathrm{Tr}\big[\Lambda'_{i+1}\rho_{i+1}\big]\big)
\end{aligned} \tag{4.66}$$

We want to work out what the derivative with respect to ρ_j is for a particular j, so we collect all the terms that are relevant to ρ_j. For now, we take j not to be 1 or n since those edge cases should be treated differently.

The 2nd term makes the contribution of $\mathrm{Tr}(\rho_j \log(\rho_j))$. The 3rd term makes the contribution of $-2\mathrm{Tr}(\rho_j \log(\rho_j))$, where there is a coefficient 2 because there's contributions at $i = j-1$ and $i = j$. The 4th term contributes $-\lambda_j \mathrm{Tr}(\rho_j)$. The 5th term contributes $\mathrm{Tr}(\Lambda_j\rho_j)$ at $i = j$ and the 6th term

contributes $\mathrm{Tr}(\Lambda'_j\rho_j)$ at $i = j-1$. Overall, this is

$$-\mathrm{Tr}(\rho_j \log(\rho_j)) - \lambda_j \mathrm{Tr}(\rho_j) + \mathrm{Tr}(\Lambda_j \rho_j) + \mathrm{Tr}\big(\Lambda'_j \rho_j\big) \tag{4.67}$$

Differentiating with respect to ρ_j and setting the result to 0 gives

$$\log(\rho_j) + \lambda'_j - \Lambda_j - \Lambda'_j = 0 \tag{4.68}$$

where $\lambda'_j = \lambda_j + 1$.

This can be rearranged to

$$\Lambda'_j = \log(\rho_j) + \lambda'_j - \Lambda_j \tag{4.69}$$

$$\Lambda_j = \log(\rho_j) + \lambda'_j - \Lambda'_j \tag{4.70}$$

Now consider the case where $j = 1$. The 2nd term makes the contribution of $\mathrm{Tr}(\rho_1 \log(\rho_1))$. The 3rd term makes the contribution of $-\mathrm{Tr}(\rho_1 \log(\rho_1))$, at $i = j = 1$. The 4th term contributes $-\lambda_1 \mathrm{Tr}(\rho_1)$. The 5th term contributes $\mathrm{Tr}(\Lambda_1 \rho_1)$ from $i = j = 1$. In contrast to the regular case, there is no contribution from the 6th term. Overall, this leads to the result

$$\lambda'_1 - \Lambda_1 = 0 \tag{4.71}$$

Similarly, for $j = n$, the equivalent result is

$$\lambda'_n - \Lambda'_n = 0 \tag{4.72}$$

Then, we want to work out the derivative with respect to $\rho_{j,j+1}$ for a particular j, so we collect all the terms that are relevant to $\rho_{j,j+1}$.

The 1st term makes the contribution $\mathrm{Tr}\Big(\hat{H}_{j,j+1}\rho_{j,j+1}\Big)$. The 3rd term contributes $\mathrm{Tr}(\rho_{j,j+1} \log(\rho_{j,j+1}))$. The 5th term contributes $-\mathrm{Tr}[(\Lambda_j \otimes \mathbb{1})\rho_{j,j+1}]$ and the 6th term contributes $-\mathrm{Tr}\big[(\mathbb{1} \otimes \Lambda'_j)\rho_{j,j+1}\big]$. Overall, this is

$$\mathrm{Tr}\Big(\hat{H}_{j,j+1}\rho_{j,j+1}\Big) + \mathrm{Tr}(\rho_{j,j+1} \log(\rho_{j,j+1})) - \mathrm{Tr}[(\Lambda_j \otimes \mathbb{1})\rho_{j,j+1}] - \mathrm{Tr}\big[(\mathbb{1} \otimes \Lambda'_j)\rho_{j,j+1}\big] \tag{4.73}$$

Differentiating with respect to $\rho_{j,j+1}$ and setting the result to 0 gives

$$\hat{H}_{j,j+1} + \log(\rho_{j,j+1}) + 1 - (\Lambda_j \otimes \mathbb{1}) - (\mathbb{1} \otimes \Lambda'_j) = 0 \tag{4.74}$$

This can be rearranged to

$$\rho_{j,j+1} = \exp\Big(-\hat{H}_{j,j+1} - 1 + (\Lambda_j \otimes \mathbb{1}) + (\mathbb{1} \otimes \Lambda'_j)\Big) \tag{4.75}$$

We are ready to build a belief propagation algorithm based on Equations 4.69, 4.70 and 4.75.

Let

$$m_{j \to j-1} = \exp(\Lambda'_j) \tag{4.76}$$

$$m_{j \to j+1} = \exp(\Lambda_j) \tag{4.77}$$

then Equation 4.75 can be written as

$$\rho_{j,j+1} \propto \exp(-\hat{H}_{j,j+1}) \odot (m_{j \to j+1} \otimes \mathbb{1}) \odot (\mathbb{1} \otimes m_{j \to j-1}) \tag{4.78}$$

noting that $\exp(\Lambda_j \otimes \mathbb{1}) = \exp(\Lambda_j) \otimes e\mathbb{1}$ because the identity matrix commutes with Λ_j, and the coefficient e can be absorbed into the proportionality. The case for $(\mathbb{1} \otimes \Lambda'_j)$ is similar.

Equations 4.69 and 4.70 can be written as

$$m_{j \to j-1} \propto \mathrm{Tr}_{j+1}(\rho_{j,j+1}) \odot m^{-1}_{j \to j+1} \tag{4.79}$$

$$m_{j \to j+1} \propto \mathrm{Tr}_{j-1}(\rho_{j-1,j}) \odot m^{-1}_{j \to j-1} \tag{4.80}$$

where we have used $\rho_j = \mathrm{Tr}_{j+1}(\rho_{j,j+1})$ and $\rho_j = \mathrm{Tr}_{j-1}(\rho_{j-1,j})$ in each of these equations consistent with the direction of the message, since the other part of the algorithm (Equation 4.78) results in pairwise reduced density matrices.

Note that $m_{n\to n-1}$ and $m_{1\to 2}$ would be needed in this algorithm, but clearly the above equations don't apply to these cases. Instead, we use the relevant Equations 4.71 and 4.72, and we notice that Λ_1 and Λ'_n do not in fact depend on any other messages or beliefs. This means that $m_{n\to n-1}$ and $m_{1\to 2}$ are always a scalar multiple of the identity matrix. This represents there being no updates coming from outside the ends of the chain of particles.

Now, we can start by initialising the messages. A choice that works well in practices in initialising all of them as identity matrices. A step in the algorithm then consists of (1) Using Expression 4.78 to compute the reduced density matrices for adjacent qubits; and (2) Using Expressions 4.79 and 4.80 to compute the next set of messages.

When the algorithm converges after a number of steps, we will have the beliefs for all the reduced density matrices for adjacent qubits, from which we can then compute the single qubit reduced density matrices by taking the partial trace.

This result is equivalent to the algorithm in [18] for a "Markov shield" of size 2.

We note that for a system with only Pauli Z matrices, there are no off-diagonal values in the Hamiltonian H or the density matrix ρ, making the algorithm effectively equivalent to a classical system on a chain.

4.7.2 General Case - 2D

In the classical graphical model, multiplication is commutative, so the edges can be treated as directionless. However, for the problem set up of the quantum case in 2D, it helps to label each qubit with a number, similar to the 1D case. This creates is a sense of direction when considering the partial traces, which is particularly useful for implementation.

For example, consider a $m \times n$ lattice, it would be sensible to label the particles as shown in Figure 4.4. With the numbering system, we can specify and ordering on the edges, requiring $s < t$ in $(s,t) \in E$. This also ensures that each edge is only counted once.

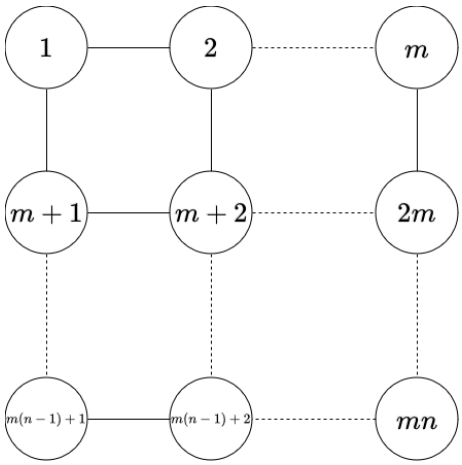

Figure 4.4: Diagram showing the numbering of particles in a lattice.

We can now proceed with the general case using the same method as the chain case.

$$\begin{aligned}
&\mathcal{L}(\rho, \lambda, \Lambda, \Lambda') \\
&= \sum_{(s,t)\in E} \mathrm{Tr}\Big(\hat{H}_{s,t}\rho_{s,t}\Big) \\
&+ \sum_{s\in V} \mathrm{Tr}(\rho_s \log(\rho_s)) \\
&+ \sum_{(s,t)\in E} [\mathrm{Tr}(\rho_{s,t}\log(\rho_{s,t})) - \mathrm{Tr}(\tilde{\rho}_s\log(\tilde{\rho}_s)) - \mathrm{Tr}(\tilde{\rho}_t\log(\tilde{\rho}_t))] \\
&- \sum_{s\in V} \lambda_s(\mathrm{Tr}(\rho_s) - 1) \\
&- \sum_{(s,t)\in E} \big(\mathrm{Tr}\big[\Lambda_{s(t)} \otimes \mathbb{1})\rho_{s,t}\big] - \mathrm{Tr}\big[\Lambda_{s(t)}\rho_s\big]\big) \\
&- \sum_{(s,t)\in E} \Big(+\mathrm{Tr}\Big[(\mathbb{1} \otimes \Lambda'_{t(s)})\rho_{s,t}\Big] - \mathrm{Tr}\Big[\Lambda'_{t(s)}\rho_t\Big]\Big)
\end{aligned} \tag{4.81}$$

For an arbitrary u in V, we want to work out what the derivative with respect to ρ_u is, so we collect all the terms that are relevant to ρ_u.

$$-\nu\mathrm{Tr}(\rho_u\log(\rho_u)) - \lambda_u\mathrm{Tr}(\rho_u) + \sum_{\{v\in N(u)|v<u\}} \mathrm{Tr}\Big[\Lambda'_{u(v)}\rho_u\Big] + \sum_{\{v\in N(u)|v>u\}} \mathrm{Tr}\big[\Lambda_{u(v)}\rho_u\big] \tag{4.82}$$

where $N(u)$ is the set of neighbours of u, and $\nu = |N(u)| - 1$. The coefficient $-\nu$ comes from 1 times $\mathrm{Tr}(\rho_u\log(\rho_u))$ from the 2nd term and $-|N(u)|$ times $\mathrm{Tr}(\rho_u\log(\rho_u))$ from the 3rd term, similar to the 1D case.

Differentiating with respect to ρ_u and setting the result to 0 gives

$$\nu\log(\rho_u) + \lambda'_u - \sum_{\{v\in N(u)|v<u\}} \Lambda'_{u(v)} - \sum_{\{v\in N(u)|v>u\}} \Lambda_{u(v)} = 0 \tag{4.83}$$

where $\lambda'_u = \lambda_u + 1$.

Taking a particular $v \in N(u)$, the previous equation can be rearranged to

$$\Lambda'_{u(v)} = \nu\log(\rho_u) + \lambda'_u - \sum_{\{v'\in N(u)\setminus\{v\}|v'<u\}} \Lambda'_{u(v')} - \sum_{\{v'\in N(u)|v'>u\}} \Lambda_{u(v')} \tag{4.84}$$

for $v < u$, and

$$\Lambda_{u(v)} = \nu\log(\rho_u) + \lambda'_u - \sum_{\{v'\in N(u)|v'<u\}} \Lambda'_{u(v')} - \sum_{\{v'\in N(u)\setminus\{v\}|v'>u\}} \Lambda_{u(v')} \tag{4.85}$$

for $v > u$.

For an arbitrary $(u, v) \in E$, we want to work out what the derivative with respect to $\rho_{u,v}$ is, so we collect all the terms that are relevant to $\rho_{u,v}$.

$$\mathrm{Tr}\Big(\hat{H}_{u,v}\rho_{u,v}\Big) + \mathrm{Tr}(\rho_{u,v}\log(\rho_{u,v})) - \mathrm{Tr}\big[(\Lambda_{u(v)} \otimes \mathbb{1})\rho_{u,v}\big] - \mathrm{Tr}\Big[(\mathbb{1} \otimes \Lambda'_{v(t)})\rho_{u,v}\Big] \tag{4.86}$$

Differentiating with respect to $\rho_{u,v}$ and setting the result to 0 gives

$$\hat{H}_{u,v} + \log(\rho_{u,v}) + 1 - \Lambda_{u(v)} \otimes \mathbb{1} - \mathbb{1} \otimes \Lambda'_{v(u)} = 0 \tag{4.87}$$

This can be rearranged to

$$\rho_{u,v} = \exp\Big[-\hat{H}_{u,v} - 1 + (\Lambda_{u(v)} \otimes \mathbb{1}) + (\mathbb{1} \otimes \Lambda'_{v(u)})\Big] \tag{4.88}$$

For each edge $(s,t) \in E$, let

$$m_{s\to t} = \exp(\Lambda_{s(t)}) \tag{4.89}$$

$$m_{t\to s} = \exp(\Lambda'_{t(s)}) \tag{4.90}$$

then Equation 4.88 can be written as

$$\rho_{u,v} \propto \exp\left(\hat{H}_{u,v}\right) \odot (m_{u\to v} \otimes \mathbb{1}) \odot (\mathbb{1} \otimes m_{v\to u}) \tag{4.91}$$

noting that $\exp(\Lambda_{s(t)} \otimes \mathbb{1}) = \exp(\Lambda_{s(t)}) \otimes e\mathbb{1}$ because the identity matrix commutes with $\Lambda_{s\to t}$, and the coefficient e can be absorbed into the proportionality. The case for $(\mathbb{1} \otimes \Lambda_{t\to s})$ is similar.

Equation 4.84 can be written as

$$m_{t\to s} = \nu\rho_s \odot \bigodot_{t'\in N(s)\setminus t} m^{-1}_{t'\to s} \tag{4.92}$$

However, like the 1D case, we need to decide which partial trace to take to get ρ_s since the other step also gives reduced density matrices for pairs of particles. Like the 1D case, we choose the traces of those matrices consistent with the direction of the message. This gives

$$m_{t\to s} = \bigodot_{t'\in N(s)\setminus t} \left[\mathrm{Tr}_{t'}(\rho_{s,t'}) \odot m^{-1}_{t'\to s}\right] \tag{4.93}$$

Similar to the 1D case, we can start by initialising the messages as identity matrices. A step in the algorithm then consists of (1) Using Expression 4.92 to compute the reduced density matrices for adjacent qubits; and (2) Using Expressions 4.93 and to compute the next set of messages. When the algorithm converges after a number of steps, we will have the belief for all the reduced density matrices for adjacent qubits, from which we can then compute the single qubit reduced density matrices by taking the partial trace.

We note that the message computation differs from the ones presented in literature [18], since it was derived under a more general assumption.

Chapter 5

Implementation

5.1 Repository Link

https://gitlab.doc.ic.ac.uk/jz4120/qbp

5.2 Tools

The software implementation for this project was made using the Python programming language. The library JAX was used to enable computation on the GPU. JAX offers its own implementation of numerical computation tools such as `jax.numpy` and `jax.scipy` which were used for the software implementation for this project, with almost identical syntax as that of the original libraries. JAX also offers varies ways to help optimise the computation.

Graphs used to visualise results of the evaluation were made using the `matplotlib.pyplot` library.

5.3 1D Implementation

This section describes the software implementation of the 1D chain belief propagation algorithm from Section 4.7.1 in the Derivation. The 2 major components of this implementation is the classes `Hamiltonian` and `BeliefPropagator`.

The class `Hamiltonian` is used to specify the Hamiltonian to be used for the algorithm. Its constructor `Hamiltonian(size, beta)` takes an integer argument `size` for the size of the system, i.e. how many particles are in the chain, and a float argument `beta` for the β value in $\exp(-\beta H)$. The θ values in $-\beta H = \sum_{k \in \mathbb{P}_n} \theta_k \sigma_k$ are set via the member functions `set_param_single` and `set_param_double` depending on how many particles the corresponding σ_k acts on.

The function `set_param_single(index, pauli, value)` takes the integer argument `index` of the single particle on which a particular σ_k acts, and `pauli`, which represent the type of the Pauli matrix (x, y, or z). It is a member of the `Enum` class `Pauli`, which represents the types of pauli matrices. The function also takes a float argument `value`, which is the value corresponding to the coefficient in models, such as $-h_x$ in the Ising model. Since in our algorithm the product $-\beta H$ is being processed as a unit, the assignment of $\theta_k = -\beta \times$ `value` is made.

Similarly, the function `set_param_double(index, pauli_0, pauli_1, value)` takes the integer argument `index` i, representing that a particular σ_k acts the pair of particles $(i, i+1)$. The `Pauli` arguments `pauli_0` and `pauli_1` specify the type of Pauli matrices regarding i and $i+1$, respectively. The float argument `value`, which is similarly the value corresponding to the coefficient in models such as J in the Ising model. So the assignment of $\theta_k = -\beta \times$ `value` is made.

Once the model is fully specified, the function `compute_partial_hamiltonians` computes the

partial hamiltonians $\hat{H}$ as specified in Equation 4.58. Then, the function `get_partial_hamiltonian(index)` can be used to get the partial Hamiltonian corresponding particles at the integer argument `index`$= i$ and $i+1$.

The class `BeliefPropagator` is the class that runs the belief propagation logic and stores the messages and beliefs. Its constructor `BeliefPropagator(hamiltonian)` takes a `Hamiltonian` and initialises the beliefs. It has a function `step()` which performs 1 step of the belief propagation algorithm, i.e. computes the messages based on the beliefs, and then computes the new beliefs based on the messages, as described in Section 4.7.1.

5.4 2D Implementation

This section describes the software implementation of the 2D belief propagation algorithm from Section 4.7.2. In particular, it works on a rectangular lattice. Similar to the 1D case, The 2 major components of this implementation is the classes `LatticeHamiltonian` and `LatticeBeliefPropagator`.

The class `LatticeHamiltonian` is used to specify the Hamiltonian to be used for the algorithm. Its constructor `LatticeHamiltonian(numrows, numcols, beta)` takes integer arguments `numrows` and `numcols` that specify the dimensions of the system, i.e. how many rows and columns are in the rectangular lattice. The float argument `beta` is again the β value in $\exp(-\beta H)$.

For setting single particle parameters, the function `set_param_single(rowindex, colindex, pauli, value)` works similarly to the 1D case, except it takes 2 integer arguments necessary for the specification of the particle location. For setting pair-wise parameters, since the model is organised in rows and columns in the rectangular lattice, we have functions `set_param_double_row(rowindex, edgeindex, pauli_0, pauli_1, value)` and `set_param_double_row(rowindex, edgeindex, pauli_0, pauli_1, value)` for the edges on the rows and columns respectively. For example, calling `set_param_double_row` with `rowindex`$= r$ and `edgeindex`$= e$ sets the coefficient of the Pauli matrix acting on the particle at (r, e) and $(r, e+1)$.

Once the model is fully specified, the function `compute_partial_hamiltonians` computes the partial hamiltonians $\hat{H}$ as specified in Equation 4.58. Then, the functions `get_partial_hamiltonian_row(rowindex, edgeindex)` `get_partial_hamiltonian_col(colindex, edgeindex)` can be used to get the partial Hamiltonian corresponding to the particles at the integer arguments `index`.

The class `LatticeBeliefPropagator` is the class that runs the belief propagation logic and stores the messages and beliefs. Its constructor `LatticeBeliefPropagator(lat_ham, reg_factor)` takes a `LatticeHamiltonian` and initialises the beliefs. It also has a function `step()`, which performs 1 step of the general belief propagation algorithm, as described in 4.7.2. It also has a float argument `reg_factor`, which is used for applying matrix regularisation to mitigate numerical instabilty,instabilty, as will be explained later. This class also has a function `mean_single_belief()` to get the beliefs on single particles, since the belief stored in the propagator is pair based.

5.5 Benchmarking Tools

Some relevant functions useful for benchmarking are in `qbp.example`. Although they do not form part of the belief propagation algorithm, they are used extensively in the benchmarking process.

These include the `rdm(rho, partial_dim, pos)`, which computes the reduced density matrix of matrix `rho` with number of particles specified by `partial_dim` at position specified by `pos`. This function implements the time and memory intensive exact diagonalisation method. They include functions to compute transverse magnetisation, correlation, and spin-spin correlation to study the property of the quantum system. The also include template functions to create, for example, a 3×3 Hamiltonian function with the specified parameters.

Chapter 6

Evaluation

6.1 1D Chain

6.1.1 Model

For the purpose of this evaluation, we choose the 1D Ising model to use for benchmarking the 1D algorithm described in Section 5.3. This is done because it is a widely used model in research [27, 11]. According to [11], the model is

$$H = J \sum_{\langle i,j \rangle} \sigma_i^z \sigma_j^z + h_z \sum_{i=1}^{N} \sigma_i^z - h_x \sum_{i=1}^{N} \sigma_i^x \tag{6.1}$$

where they have used slightly different notation to ours. In this notation, J, h_z and h_x are real scalar coefficients. The Pauli matrices are represented in a way where σ_i^z indicates σ_z acting on the ith particle without impacting other particles, and $\sigma_i^z \sigma_j^z$ indicates σ_z acting on the ith and jth particle without impacting other particles. The case for σ_i^x is similar.

What this means for the software implementation is, for example if $J = 1$, then we have to call `set_param_double(i, Pauli.Z, Pauli.Z, 1)` for all the `i` values. If $h_x = 1$ then we have to call `set_param_single(i, Pauli.X, -1)` for all the `i` values, noting the negative sign in front of h_x in the equation.

A good choice of the scalar coefficients for the 1D model according to [11] is $h_x = 1.05$, $h_z = 0.5$ and $J = 1$. These values will be used in this section unless otherwise specified.

6.1.2 Correctness

We used the specified values for h_x, h_z and J for the 1D belief propagation algorithm, and compare the results against the exact diagonalisation results. We find that the difference is very small, as shown in Figure 6.1, this indicates great performance of the algorithm in terms of correctness.

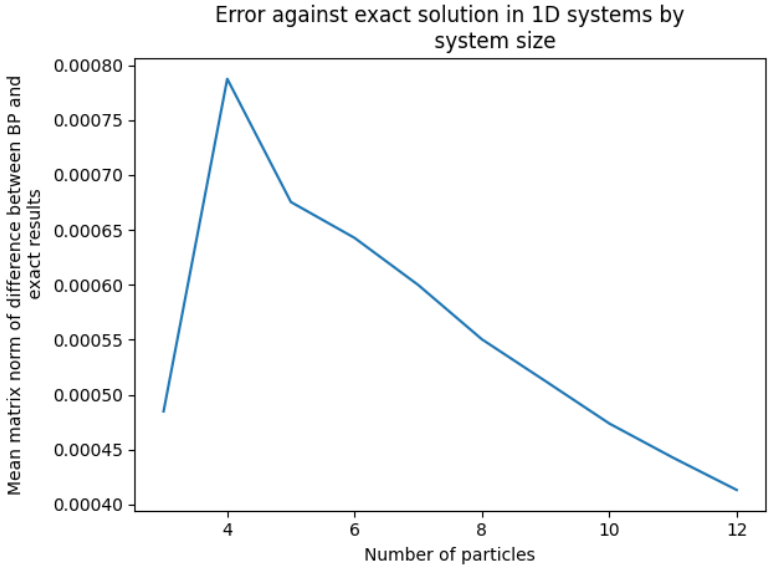

Figure 6.1: Error of Belief Propagation compared to the exact results. The values are mean matrix norm of the beliefs on single particles. Ising Model with $h_x = 1.05$, $h_z = 0.5$ and $J = 1$

We also investigate the behaviour when $J = 0$, i.e. there are no pair-wise interactions at all. This results in a much simpler system, and as expected, the algorithm achieves even better performance in terms of correctness, as shown in Figure 6.2.

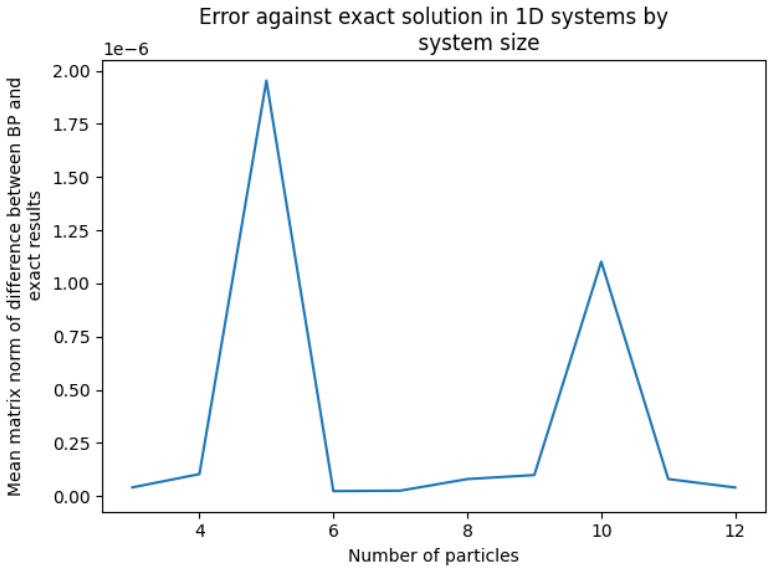

Figure 6.2: Error of Belief Propagation compared to the exact results. The values are mean matrix norm of the beliefs on single particles. Ising Model with $h_x = 1.05$, $h_z = 0.5$ and $J = 0$

6.1.3 Convergence

We investigate the convergence of the algorithm by checking the error against the exact result after each propagation step, using a system with $n = 10$ particles. We expected the algorithm to converge after at most n steps, since that is the number of steps necessary to send information from one end of the chain to the other. Figure 6.3 shows that the algorithm in fact converges much quicker, at only 4 steps for $n = 10$.

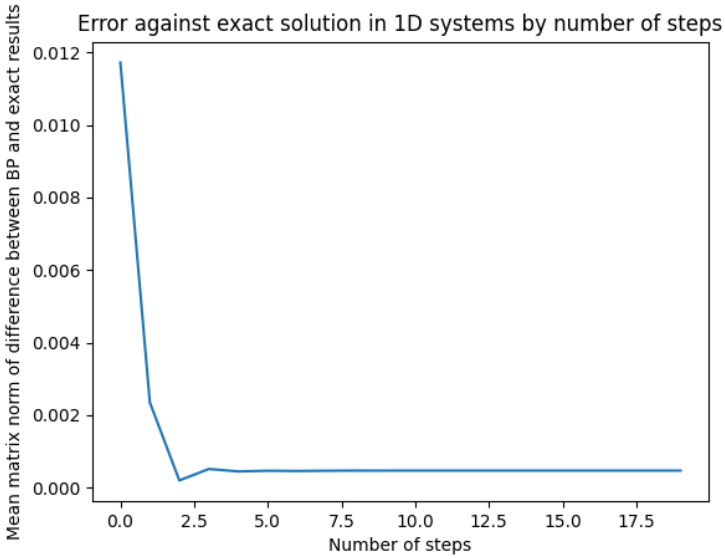

Figure 6.3: Error of Belief Propagation after each propagation step, compared to the exact results. The values are mean matrix norm of the beliefs on single particles. 10-particle Ising Model with $\beta = 1$, $h_x = 1.05$, $h_z = 0.5$ and $J = 1$

6.1.4 Time performance

Figure 6.4 shows the time it takes to run the algorithm for n steps for a system with n particles. As expected, it resembles a quadratic graph, since the number of steps and number of messages passed in each step both scales linearly with n, giving a quadratic overall relationship.

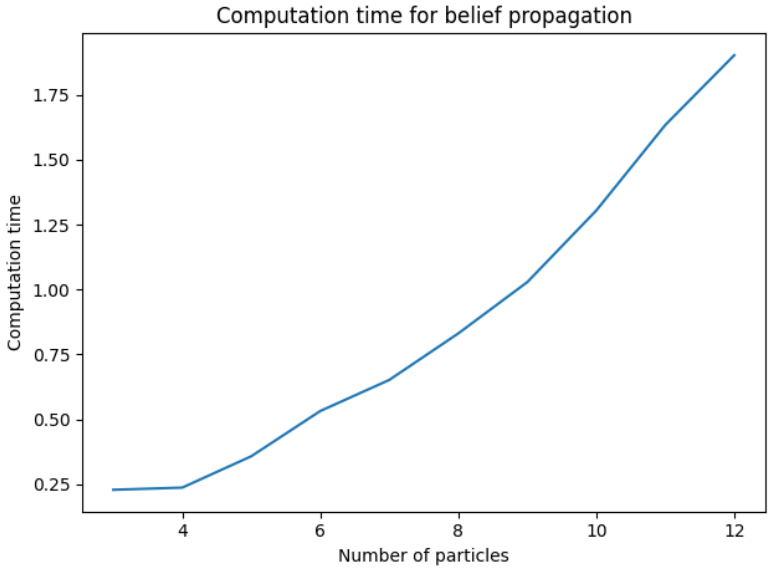

Figure 6.4: Computation time of Belief Propagation.

We can contrast this with the exact solution, which scales exponentially as the number of particles n increases, shown in Figure 6.5. This shows that the belief propagation algorithm does indeed fulfill its purpose of being applicable to much larger systems.

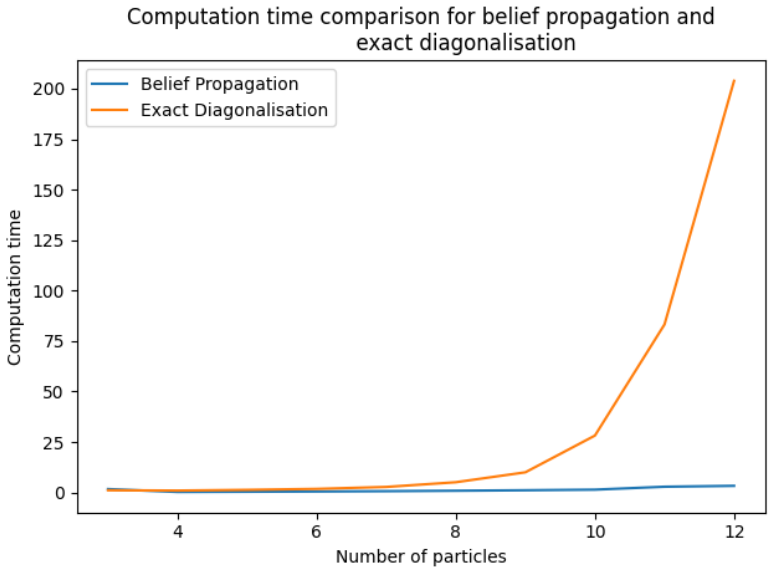

Figure 6.5: Computation time of Belief Propagation compared to the exact diagonalisation.

6.1.5 Scaling

We would like to know how the algorithm scales for larger systems. However, exact diagonalisation becomes infeasible as it takes exponentially more time and memory to compute the exact solution. The paper [11] provides a great reference point as they study the transverse magnetisation and correlation for different energy densities (obtained by different β values). Their results are summarised by the following plots in Figure 6.6.

The transverse magnetisation M_x and correlation C_{xx} are defined as [11]

$$M_x = \frac{1}{n}\sum_i \langle \sigma_i^x \rangle \tag{6.2}$$

$$C_{xx} = \frac{1}{n}\sum_i \langle \sigma_i^x \sigma_{i+1}^x \rangle \tag{6.3}$$

where the expectations are equivalent to the corresponding mean parameters. The energy density is $\frac{\langle H \rangle}{N}$, where we can compute $\langle H \rangle = \mathrm{Tr}(\rho H)$, which is difficult to compute directly but we can use paritial hamiltonian $\hat{H}$ values as defined in the derivation compute it.

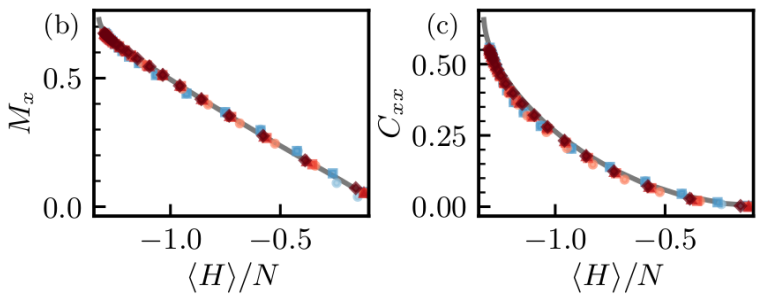

Figure 6.6: Results on transverse magnetisation M_x (left) and correlation C_{xx} for 100-particle Ising Model with $h_x = 1.05$, $h_z = 0.5$ and $J = 1$ from paper by [11]

Our results using the 1D belief propagation method is shown in Figure 6.7 for transverse magnetisation and Figure 6.8 for correlation.

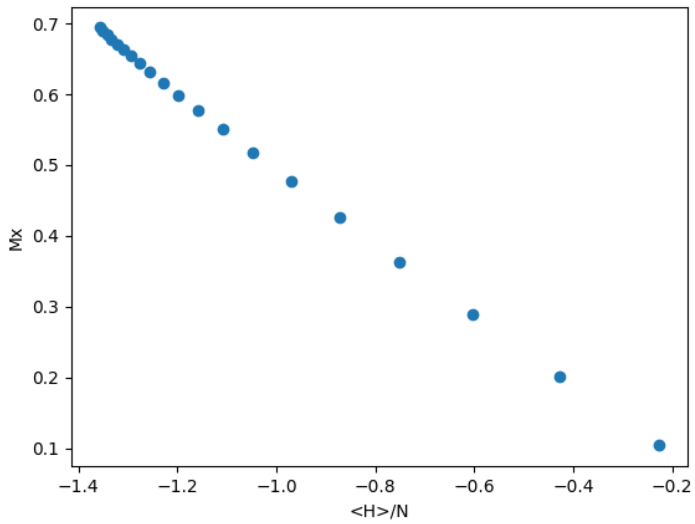

Figure 6.7: Results on transverse magnetisation M_x for 100-particle Ising Model with $h_x = 1.05$, $h_z = 0.5$ and $J = 1$ obtained using 1D belief propagation

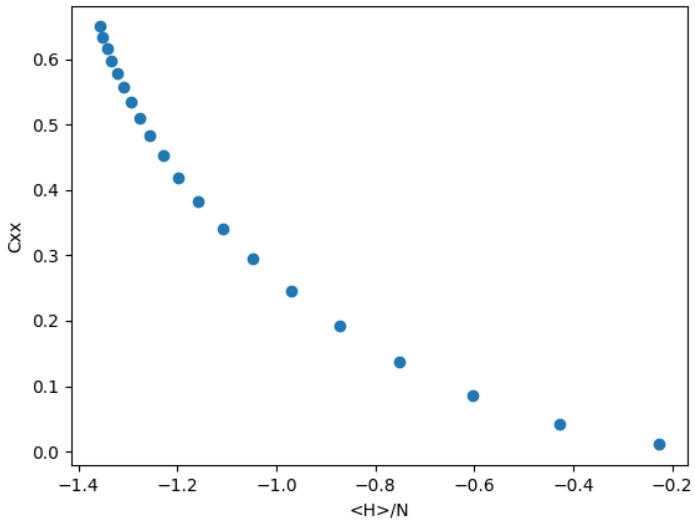

Figure 6.8: Results on correlation C_{xx} for 100-particle Ising Model with $h_x = 1.05$, $h_z = 0.5$ and $J = 1$ obtained using 1D belief propagation

6.2 2D Lattice

6.2.1 Model

After verifying the outstanding performance of our belief propagation algorithm on 1D chains, we now turn our attention to 2D lattices, which is more complicated and interesting, and where other methods have had worse performance [11].

For the 2D lattice, we will use the same Ising model as described in Equation 6.1. However, as opposed to the 1D case, here neighbouring particles include all directions on the lattice.

For example, in this case, suppose $J = 1$, then in the software implementation we need to call

`set_param_double_row(r, e, Pauli.Z, Pauli.Z, 1)` for all the `r` and `e` values for the edges in the row direction, and call `set_param_double_col(c, e, Pauli.Z, Pauli.Z, 1)` for all the `c` and `e` values for the edges in the column direction.

A good choice of the scalar coefficients for the 2D model according to [11] is $h_x = 2.5$, $h_z = 0$ and $J = -1$. These values will be used in this section unless otherwise specified.

6.2.2 Snake Formation of 1D Algorithm

Before we proceed to evaluating the 2D algorithm based on the general algorithm in Section 5.4, we will first demonstrate the need of such an algorithm by showing that the simpler 1D algorithm in Section 5.3 is inadequate at studying 2D lattices.

The way to study a 2D lattice using a 1D algorithm is putting all the particles in the lattice on a 1D chain, and the most intuitive way is arranging them in a snake formation as shown in Figure 6.9.

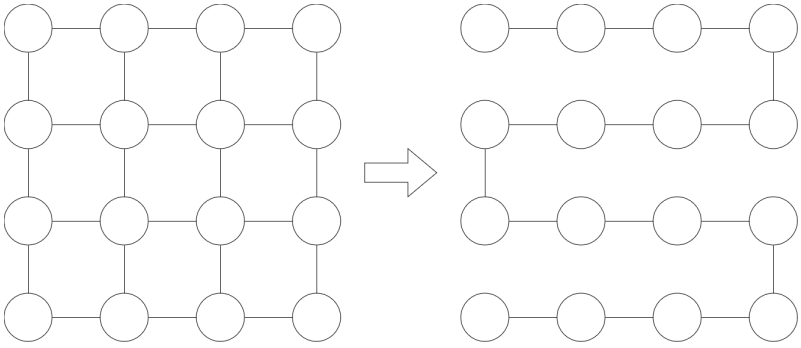

Figure 6.9: Snake-shaped approximation of a 2D lattice into a 1D chain

Since this approach ignores many pair-wise interactions in the lattice, we expect it to give a worse result than if the 1D algorithm had been applied to an actual chain. Figure 6.10 confirms by comparing the error of use the 1D algorithm against an actual 9 particle 1D chain and a 3×3 lattice approximation using the snake formation. Clearly, as expected, the 1D algorithm is less suited for studying a 2D lattice.

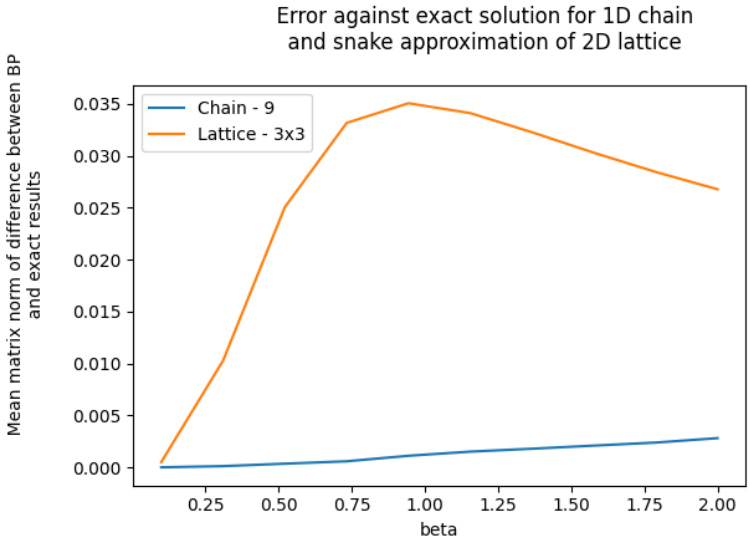

Figure 6.10: Error of Belief Propagation compared to the exact results for different values of β, comparing the performance of the 1D algorithm on an 9 particle 1D chain and a snake formation of a 3×3 2D lattice. The values are mean matrix norm of the beliefs on single particles. Ising Model with $h_x = 2.5$, $h_z = 0$ and $J = -1$.

Figure 6.11 shows how the 2D algorithm does better than the snake formation. Note that this figure only includes β values of up to 1. This is due to a numerical problem for larger β values which will be explored later.

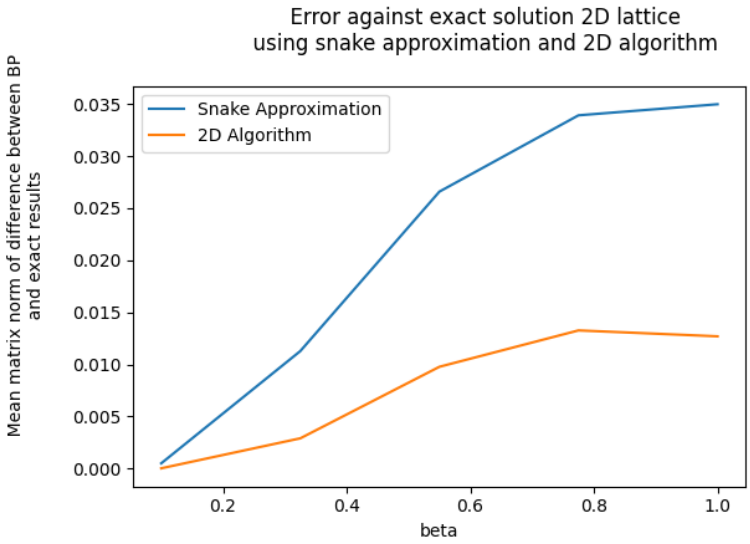

Figure 6.11: Error of Belief Propagation compared to the exact results for different values of β, comparing the performance of the 1D snake formation and the 2D algorithm. The values are mean matrix norm of the beliefs on single particles. 3×3 Ising Model with $h_x = 2.5$, $h_z = 0$ and $J = -1$.

6.2.3 Convergence

We have already demonstrated that the 2D algorithm gives good results regarding correctness in the previous comparison with the snake formation. Now we want to see how the algorithm converges. Figure 6.12 shows that similar to the 1D case, the algorithm converges rather quickly, in only 3

steps for the 3×3 system.

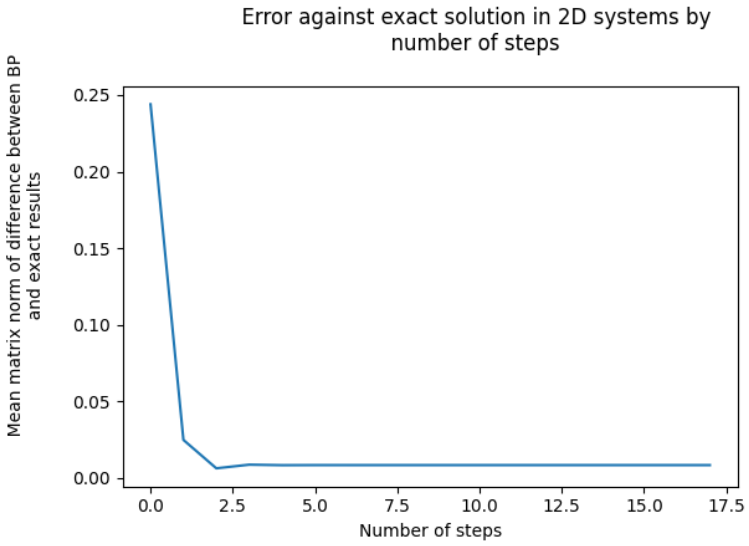

Figure 6.12: Error of Belief Propagation after each propagation step, compared to the exact results. The values are mean matrix norm of the beliefs on single particles. 3×3 Ising Model with $\beta = 1$, $h_x = 2.5$, $h_z = 0$ and $J = -1$.

6.2.4 Numerical Instability

Figure 6.14 shows the mean matrix norm difference between the belief propagation result and the exact result for different β values up to 7 for a 2×2 lattice. The error fluctuates for different betas, however, it generally remains low. However, past a threshold just above $\beta > 7$, the error blows up, and then the results from the belief propagation becomes NaN, as shown in Figure 6.13.

Figure 6.15 and 6.16 show that a similar problem happens with larger systems.

We investigated the messages in the belief propagation and found that the problem comes from some of the messages being too close to a singular matrix, i.e. one of their eigenvalues is too close to 0.

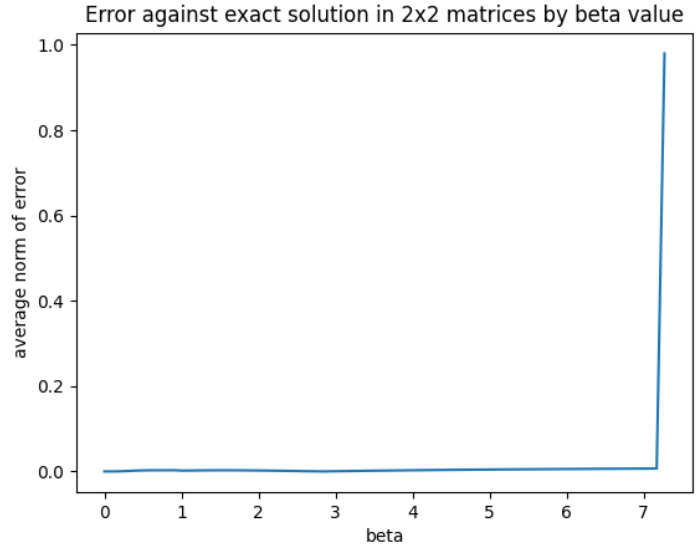

Figure 6.13: Error of Belief Propagation compared to the exact results for different values of β. The values are mean matrix norm of the beliefs on single particles. 2×2 Ising Model with $h_x = 2.5$, $h_z = 0$ and $J = -1$.

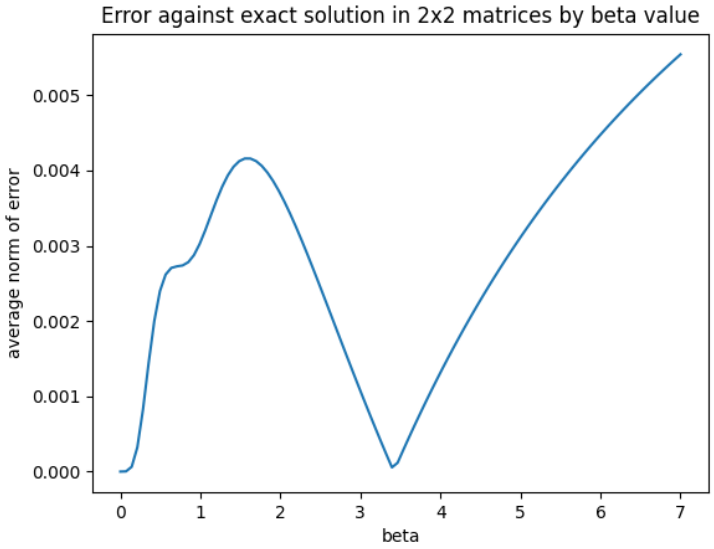

Figure 6.14: Error of Belief Propagation compared to the exact results for different values of β up to 7. The values are mean matrix norm of the beliefs on single particles. 2×2 Ising Model with $h_x = 2.5$, $h_z = 0$ and $J = -1$. 10-particle Ising Model with $\beta = 1$, $h_x = 1.05$, $h_z = 0.5$ and $J = 1$

6.2.5 2D Algorithm with Message Regularisation

In order to mitigate the problem of having to invert a matrix that is almost singular, we apply regularisation in the form of adding some value to its diagonal, inspired by ridge regularisation used in machine learning algorithms [20]. This inevitably leads to bias in the algorithm. However, we will see that in practice the reduction in numerical errors is worth the additional bias.

Let r be the regularisation factor, and the diagonal values of the message m be d_1 and d_2, then

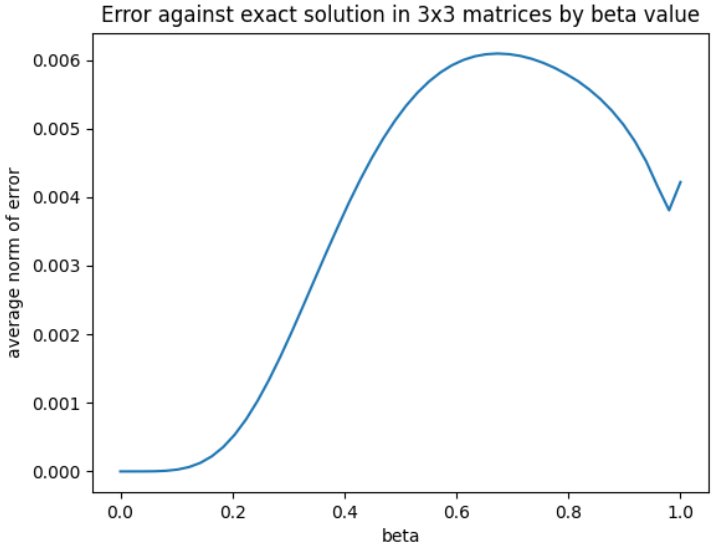

Figure 6.15: Error of Belief Propagation compared to the exact results for different values of β up to 1. The values are mean matrix norm of the beliefs on single particles. 3×3 Ising Model with $h_x = 2.5$, $h_z = 0$ and $J = -1$.

the adjusted message is

$$m + r \begin{pmatrix} d_1^{-1} & 0 \\ 0 & d_2^{-1} \end{pmatrix} \tag{6.4}$$

This adjustment is only made if the condition number of the message is large, as matrices with small condition numbers do not suffer from this numerical problem.

Initial exploration shows that this type of regularisation indeed works in mitigating the numerical instability reducing the error, as evidenced in Figure 6.17. We observe there is a sharp increase in error at around $\beta = 0.75$. This is where the threshold of regularisation is passed and messages start to be adjusted. Also, for $\beta < 1$, the error is larger than the algorithm without regularisation. This is expected due to the introduced bias. The errors here are significantly larger, hence they are difficult to work with. We now look to reduce the introduced bias in the form of r while still containing the numerical instability.

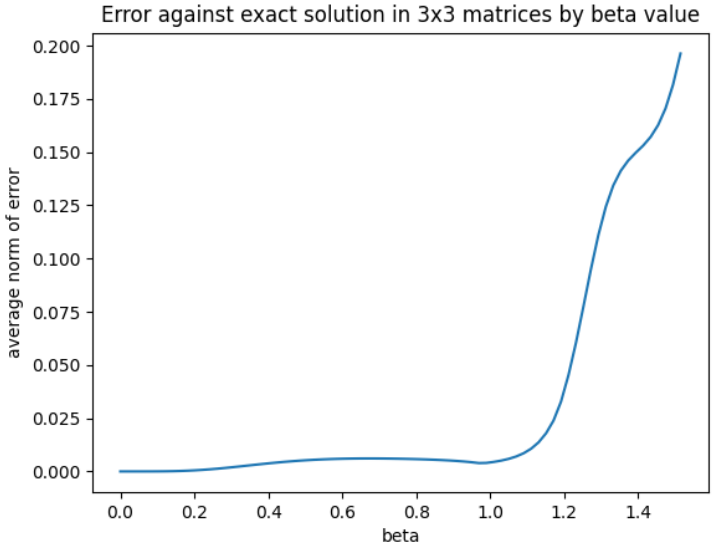

Figure 6.16: Error of Belief Propagation compared to the exact results for different values of β. The values are mean matrix norm of the beliefs on single particles. 3×3 Ising Model with $h_x = 2.5$, $h_z = 0$ and $J = -1$.

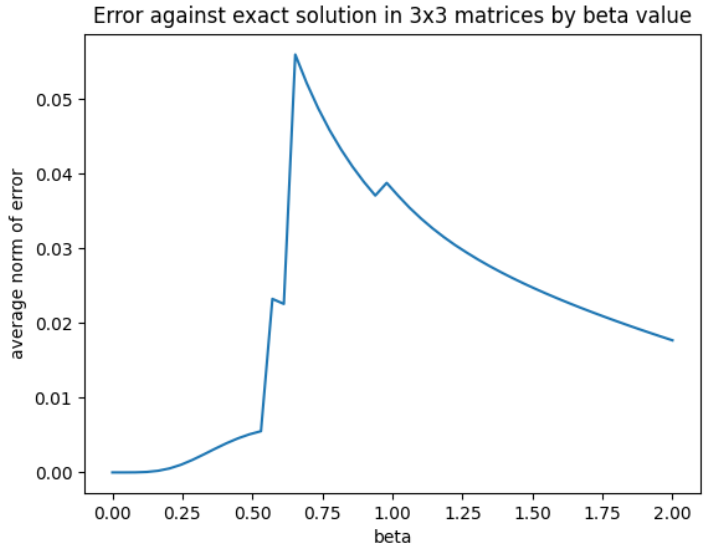

Figure 6.17: Error of Belief Propagation with Regularisation compared to the exact results for different values of β. The values are mean matrix norm of the beliefs on single particles. 3×3 Ising Model with $h_x = 2.5$, $h_z = 0$ and $J = -1$. Regularisation factor 0.1.

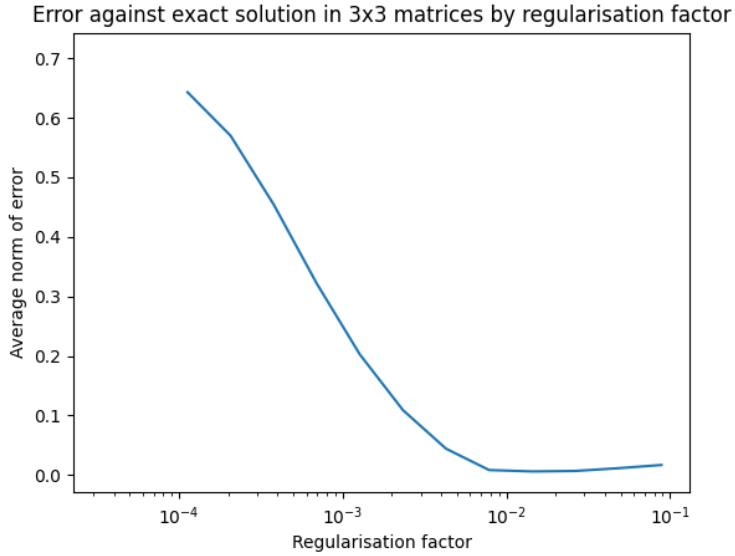

Figure 6.18: Error of Belief Propagation for different values of regularisation factor. The values are mean matrix norm of the beliefs on single particles. 3×3 Ising Model with $\beta = 2$, $h_x = 2.5$, $h_z = 0$ and $J = -1$.

Therefore, we performed a search for the optimal value for the regularisation factor, shown in Figure 6.18. It indicates that $r = 0.01$ is the optimal value among those searched. Hence, we repeat the algorithm on the 3×3 system with the optimal regularisation factor. The result is shown in Figure 6.19. These errors are more in line with the errors for small β without regularisation.

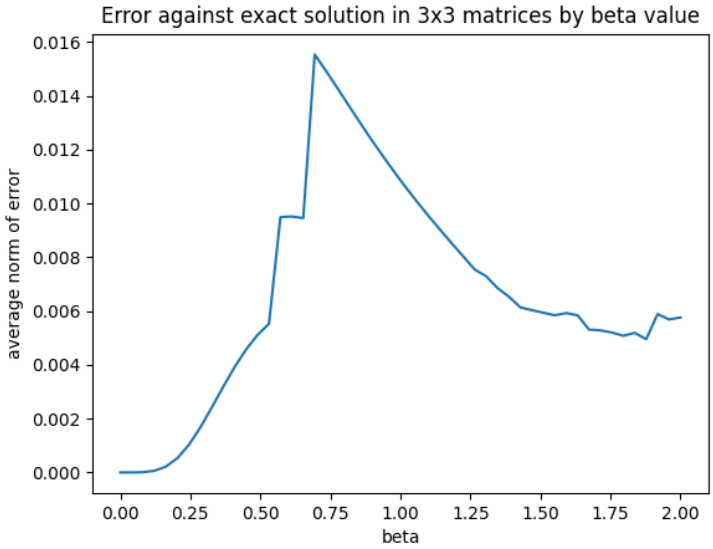

Figure 6.19: Error of Belief Propagation with Regularisation compared to the exact results for different values of β. The values are mean matrix norm of the beliefs on single particles. 3×3 Ising Model with $h_x = 2.5$, $h_z = 0$ and $J = -1$. Regularisation factor 0.01.

6.2.6 Scalability

We test the 2D belief propagation algorithm with message regularisation on larger systems. Since there is no exact result to compare against, we inspect the statistics such as transverse magnetisation and spin-spin correlation, used in [11].

The reference paper [11] does not give results for 4×4 systems. However, by comparing Figure 6.20 and 6.21 with their 10×10 results, we believe the algorithm gives good results for 4×4 systems.

The transverse magnetisation M_x is the same as in the 1D case, and the spin-spin correlation C_{zz} is defined as [11]

$$C_{xx} = \frac{1}{n} \sum_{\langle i,j \rangle} \langle \sigma_i^z \sigma_j^z \rangle \tag{6.5}$$

where the expectations are equivalent to the corresponding mean parameters. The elements of $\langle i, j \rangle$ are particle indices that share an edge. The energy density is also the same as in the 1D case.

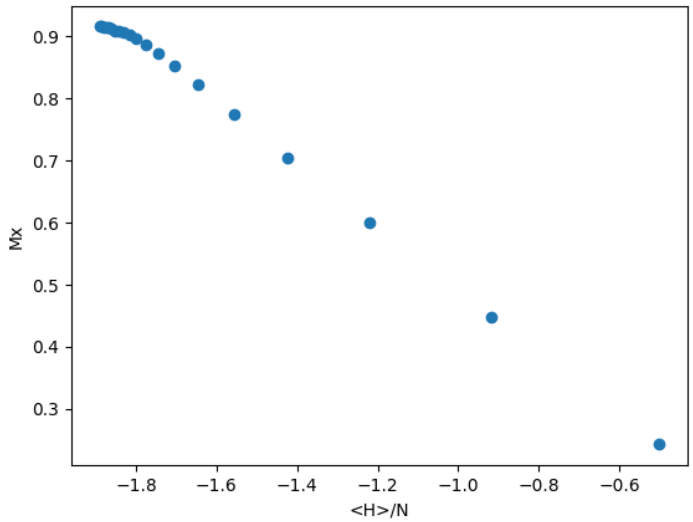

Figure 6.20: Results on transverse magnetisation M_x for 4×4 Ising Model with $h_x = 2.5$, $h_z = 0$ and $J = -1$. Regularisation factor 0.01.

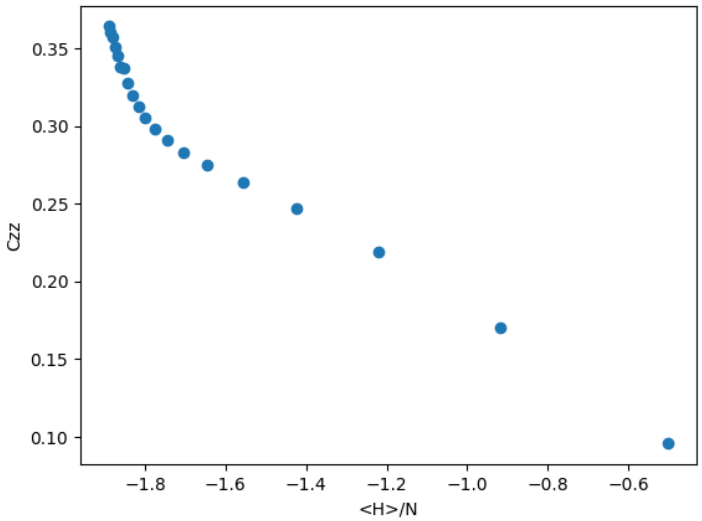

Figure 6.21: Results on correlation C_{zz} for 4×4 Ising Model with $h_x = 2.5$, $h_z = 0$ and $J = -1$. Regularisation factor 0.01.

Unfortunately, the scalability is limited. For larger systems such as the 7×7 one shown in

Figure 6.22, the result is less well-behaved. For 10×10 systems, the numerical problems are more prominent, where the results are still NaN for large values of β even with message regularisation. After attempting several different regularisation factors with poor results, we believe that more sophisticated regularisation methods need to be applied for better scalability.

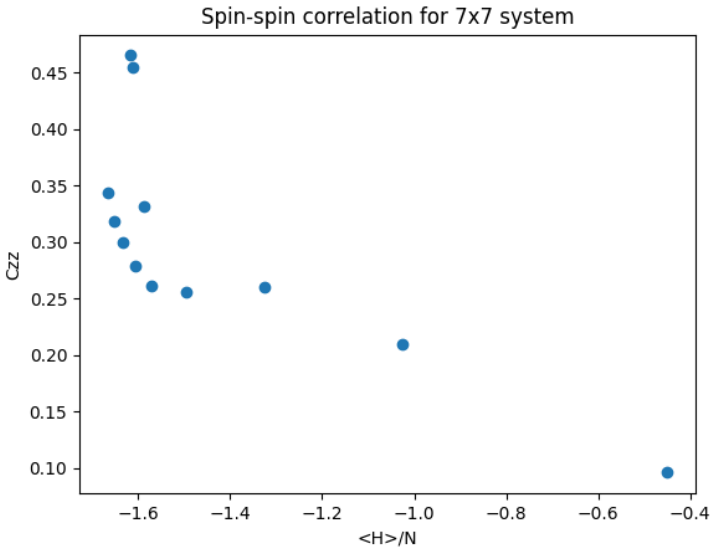

Figure 6.22: Results on correlation C_{zz} for 7×7 Ising Model with $h_x = 2.5$, $h_z = 0$ and $J = -1$. Regularisation factor 0.014.

Chapter 7

Conclusion

7.1 Summary

In this project, we followed the steps of the derivation of the classical belief propagation algorithm to derive a relatively general quantum belief propagation algorithm. This is because we observe that the thermal equilibrium state representation can be readily interpreted as an exponential family, and due to quantum interactions being short-ranged, they can be considered as graphical models. After recognising the equivalent relaxations to the variational problems, namely the entropy approximation and the local consistency constraints, we were able to follow the same method of building a message passing system using derivatives of the Lagrangian of the variational problem.

The resulting 1D belief propagation algorithm is effectively equivalent to the algorithms presented in literature. The 2D version is simpler in terms of computation since it is less dependent on the specific structure of the system.

We implemented the 1D algorithm and a particular version of the 2D algorithm that works on rectangular lattices in Python using JAX. The core part of the implementation involves a class to specify model parameters for the Hamiltonian of interest and a class to perform the belief propagation method, i.e. computing the messages and the beliefs. Various useful functions for benchmarking the belief propagation algorithm were also implemented.

In order to benchmark the implementations, scripts written in Python that run the implementations with various configurations were used to test key properties of the algorithms. We noticed that the algorithm converges much quicker than anticipated in the cases studied.

We found the 1D algorithm to have outstanding performance, achieving very low error compared to the exact solution while taking much less computation time. Where exact solutions are not available, we verified against results in literature that the algorithm achieves state-of-the-art results even in very large systems.

We found the 2D algorithm to achieve significantly better performance for 2D lattice models than if we were to approximate it as a 1D chain and then run the 1D algorithm on it, which is expected. The 2D algorithm achieves good results for small values of β in the cases studied. However, for large values of β, we identified a numerical problem in the algorithm where the messages are sometimes almost singular. We explored one way of mitigating the problem by regularising the messages if their condition numbers are too large. This achieved good results for smaller systems but limited success for larger systems.

7.2 Further Work

One area of further work possibility would be optimising the code by fully capitalising the parallelisation on the GPU. The algorithm is very easy to parallelise, as the computation of each message or belief is indenpendent from other messages or beliefs within the same step. One can also investigate the parts of the algorithm that can utilise the just-in-time compilation provided by JAX.

We have identified numerical challenges of the algorithm for large β values and provided ideas on how they can be overcome. Large β values correspond low temperatures since they are inversely proportional. More research could be done in a more detailed study of how the algorithm behaves as β gets very large (very low temperature), and how the algorithm can be adapted to 0 temperature, i.e. as $\beta \to \infty$. Then it would be natural to compare the results there to results of a classical system.

Another research direction is a more detailed study on how the belief propagation algorithm behaves on models where Monte Carlo algorithms suffer from the "sign problem", and compare the performance of both types of algorithms.

Bibliography

1. Alkabetz R and Arad I. Tensor networks contraction and the belief propagation algorithm. Phys. Rev. Res. 2021 Apr; 3(2):023073. DOI: 10.1103/PhysRevResearch.3.023073. Available from: https://link.aps.org/doi/10.1103/PhysRevResearch.3.023073
2. Azodi P and Rabitz HA. Dynamics and Geometry of Entanglement in Many-Body Quantum Systems. 2023. arXiv: 2308.09784 [quant-ph]
3. Benavides-Riveros CL, Wasak T, and Recati A. Extracting Many-Body Quantum Resources within One-Body Reduced Density Matrix Functional Theory. 2023. arXiv: 2311.12596 [quant-ph]
4. Chen T and Cheng YC. Numerical computation of the equilibrium-reduced density matrix for strongly coupled open quantum systems. The Journal of Chemical Physics 2022 Aug; 157:064106. DOI: 10.1063/5.0099761. Available from: http://dx.doi.org/10.1063/5.0099761
5. Childs AM, Su Y, Tran MC, Wiebe N, and Zhu S. Theory of Trotter Error with Commutator Scaling. Phys. Rev. X 2021 Feb; 11(1):011020. DOI: 10.1103/PhysRevX.11.011020. Available from: https://link.aps.org/doi/10.1103/PhysRevX.11.011020
6. Ferson S. What Monte Carlo methods cannot do. Human and Ecological Risk Assessment: An International Journal 1996; 2:990–1007. DOI: 10.1080/10807039609383659. eprint: https://doi.org/10.1080/10807039609383659. Available from: https://doi.org/10.1080/10807039609383659
7. Goldstein S, Lebowitz JL, Tumulka R, and Zanghì N. On the Distribution of the Wave Function for Systems in Thermal Equilibrium. Journal of Statistical Physics 2006 Dec; 125:1193–221. DOI: 10.1007/s10955-006-9210-z. Available from: http://dx.doi.org/10.1007/s10955-006-9210-z
8. Lanssens C, Ayers PW, Van Neck D, De Baerdemacker S, Gunst K, and Bultinck P. Method for making 2-electron response reduced density matrices approximately N-representable. The Journal of Chemical Physics 2018 Feb; 148. DOI: 10.1063/1.4994618. Available from: http://dx.doi.org/10.1063/1.4994618
9. Lauritzen SL. Graphical models. eng. Oxford statistical science series ; 17. Oxford: Clarendon Press, 1996 - 1996
10. Leifer M and Poulin D. Quantum Graphical Models and Belief Propagation. Annals of Physics 2008 Aug; 323:1899–946. DOI: 10.1016/j.aop.2007.10.001. Available from: http://dx.doi.org/10.1016/j.aop.2007.10.001
11. Lu S, Giudice G, and Cirac JI. Variational Neural and Tensor Network Approximations of Thermal States. 2024. arXiv: 2401.14243
12. Makur A, Mertzanidis M, Psomas A, and Terzoglou A. On the Robustness of Mechanism Design under Total Variation Distance. 2023. arXiv: 2310.07809 [cs.GT]
13. Mridula J, Kumar K, and Patra D. Combining GLCM Features and Markov Random Field Model for Colour Textured Image Segmentation. *2011 International Conference on Devices and Communications (ICDeCom)*. 2011 :1–5. DOI: 10.1109/ICDECOM.2011.5738494
14. Nielsen MA and Chuang IL. Quantum computation and quantum information. eng. 10th anniversary ed. Cambridge ; Cambridge University Press, 2010

15. Östlund S and Rommer S. Thermodynamic Limit of Density Matrix Renormalization. Phys. Rev. Lett. 1995 Nov; 75(19):3537–40. DOI: 10.1103/PhysRevLett.75.3537. Available from: https://link.aps.org/doi/10.1103/PhysRevLett.75.3537

16. Pearl J, Morgan MB, Jowell S, Genderen RV, Pearl D, and Medoff L. Probabilistic reasoning in intelligent systems : networks of plausible inference. eng. Revised second printing. Morgan Kaufmann Series in Representation and Reasoning. San Francisco, California: Morgan Kaufmann Publishers, Inc., 1988

17. Peng L, Zhang X, and Chan GKL. Fermionic reduced density low-rank matrix completion, noise filtering, and measurement reduction in quantum simulations. 2023. arXiv: 2306.05640 [quant-ph]

18. Poulin D and Hastings MB. Markov Entropy Decomposition: A Variational Dual for Quantum Belief Propagation. Physical Review Letters 2011 Feb; 106. DOI: 10.1103/physrevlett.106.080403. Available from: http://dx.doi.org/10.1103/PhysRevLett.106.080403

19. Rockafellar RT. Convex analysis. eng. Princeton mathematical series ; 28. Princeton, New Jersey: Princeton University Press, 1970

20. Saleh AKME, Kibria B. M. G. (MG, and Arashi M(. Theory of ridge regression estimators with applications. eng. 1st edition. Wiley series in probability and statistics. Hoboken, New Jersey: Wiley, 2019

21. Suzuki M, Miyashita S, and Kuroda A. Monte Carlo Simulation of Quantum Spin Systems. I. Progress of Theoretical Physics 1977 Nov; 58:1377–87. DOI: 10.1143/PTP.58.1377. eprint: https://academic.oup.com/ptp/article-pdf/58/5/1377/5389651/58-5-1377.pdf. Available from: https://doi.org/10.1143/PTP.58.1377

22. Takemori N, Teranishi Y, Mizukami W, and Yoshioka N. Balancing error budget for fermionic k-RDM estimation. 2023. arXiv: 2312.17452 [quant-ph]

23. Tindall J, Fishman M, Stoudenmire M, and Sels D. Efficient tensor network simulation of IBM's Eagle kicked Ising experiment. 2023. arXiv: 2306.14887 [quant-ph]

24. Trotter HF. On the product of semi-groups of operators. 1959. Available from: https://api.semanticscholar.org/CorpusID:35277671

25. Troyer M and Wiese UJ. Computational Complexity and Fundamental Limitations to Fermionic Quantum Monte Carlo Simulations. Physical Review Letters 2005 May; 94. DOI: 10.1103/physrevlett.94.170201. Available from: http://dx.doi.org/10.1103/PhysRevLett.94.170201

26. Vidal G. Entanglement Renormalization. Physical Review Letters 2007 Nov; 99. DOI: 10.1103/physrevlett.99.220405. Available from: http://dx.doi.org/10.1103/PhysRevLett.99.220405

27. Wainwright MJ and Jordan MI. Vol. 1. 2008. DOI: 10.1561/2200000001

28. Wang J, Li Y, and Sun H. Lie-Trotter Formula for the Hadamard Product. Acta Mathematica Scientia. 2020 May; 40:659–69. DOI: 10.1007/s10473-020-0305-4. Available from: https://doi.org/10.1007/s10473-020-0305-4

29. Wilcox RM. Exponential Operators and Parameter Differentiation in Quantum Physics. Journal of Mathematical Physics 1967 Apr; 8:962–82. DOI: 10.1063/1.1705306. eprint: https://pubs.aip.org/aip/jmp/article-pdf/8/4/962/19138643/962_1_online.pdf. Available from: https://doi.org/10.1063/1.1705306