ICFO - Institute of Photonic Sciences

Doctoral Thesis

A Variety of Optimization Techniques Applied In the Context of Quantum Information Theory

Author:
Luke Mortimer

Supervisor:
Prof. Antonio Acín

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

in the group of

Quantum Information Theory

July 28, 2025

Abstract

The thesis considers a number of optimisation techniques applied in the context of quantum information theory. After a pedagogical introduction of both quantum information theory and optimisation, it considers three main avenues of research. The first is the well-known foundational open problem of *mutually unbiased bases*, which consists of finding sets of orthonormal bases that are each unbiased with one another. More specifically, it remains unknown whether one can find a set of 4 mutually unbiased bases in dimension 6. A variety of optimisation techniques are applied, including non-linear semidefinite programming, see-saw optimisation, semidefinite programming relaxations, branch-and-cut, gradient descent methods and the method of Lagrange multipliers, each providing further insights into the problem. The second avenue is that of *Bell nonlocality*, more specifically attempting to simplify the hierarchy of semidefinite programs known as the NPA (Navascués-Pironio-Acín) hierarchy used to find bounds on the maximum quantum violation of Bell inequalities. For the case in which one has a large number of inputs per party, advantage in both memory and time versus state-of-the-art solvers is demonstrated using a combination several optimisation techniques. The third avenue is that of *many-body quantum physics*, which encompasses a wide range of topics. The thesis considers the problems of bounding expectation values of observables over the steady-states of open quantum systems, finding improved Fermion-to-qubit mappings and solving the graph colouring problem with a novel qudit-inspired optimisation algorithm. In each case, advantage versus comparable methods is demonstrated.

La tesis considera varias técnicas de optimización aplicadas en el contexto de la teoría de la información cuántica. Luego de una introducción pedagógica aăla teoría de la información cuántica yăa la optimización, se abordan tres líneas principales de investigación. La primera es el afamado problema abierto fundamental de *las bases mutuamente insesgadas*, que consiste en encontrar conjuntos de bases ortonormales que sean iinsesgadas entre sí. Más específicamente, aún se desconoce si es posible encontrar un conjunto de 4 bases mutuamente insesgadas en dimensión 6. Se aplican diversas técnicas de optimización, incluyendo programación semidefinida no lineal, optimización see-saw, relajaciones de programación semidefinida, branch-and-cut, métodos de descenso de gradiente y el método de los multiplicadores de Lagrange, que ofrecen nuevas perspectivas sobre el problema. La segunda línea de investigación se centra en *la no localidad de Bell*, y más concretamente, en la simplificación de la jerarquía de programas semidefinidos conocida como jerarquía NPA (Navascués-Pironio-Acín), utilizada para encontrar cotas sobre la máximaăviolación cuántica ăde las desigualdades de Bell. En el caso de un gran número de entradas por parte, se demuestra una ventaja tanto en memoria como en tiempo frente a los solucionadores más avanzados, combinando diversas técnicas de optimización. La tercera línea de investigación trata sobre *la física cuántica de muchos cuerpos*, un campo que abarca una amplia variedad de temas. La tesis considera los problemas de acotar los valores esperados de observables en los estados estacionarios de sistemas cuánticos abiertos, encontrar mejores mapeos de fermiones a cúbits y resolver el problema de coloreo de grafos mediante un novedoso algoritmo de optimización inspirado en qudits. En cada caso, se muestra una ventaja en comparación con métodos similares.

La tesi discorre sobre un nombre de tècniques doptimització que poden aplicar-se en el context de la teoria de la informació quàntica. Després duna introducció pedagògica tant de la teoria de la informació quàntica com de loptimització, té en compte tres línies principals de recerca. En primer lloc, una de prou coneguda, el problema obert i fonamental de les *bases mútuament imparcials*, que consisteix en trobar conjunts de bases ortonormals que siguin imparcials entre si. En concret, encara es desconeix si hom pot trobar un conjunt de 4 bases mútuament imparcials amb la dimensió 6. Shi apliquen diverses tècniques doptimització, entre les quals hi ha la programació semidefinida no lineal, loptimització cíclica, les relaxacions en programació semidefinida, el mètode Branch and cut, mètodes de gradient descendent i el mètode dels multiplicadors de Lagrange, i cadascuna aprofundeix amb noves perspectives al problema. La segona línia és la de la *no localitat de Bell*, en particular amb la intenció de simplificar la jerarquia de la programació semidefinida que rep el nom de jerarquia d'NPA (Navascués-Pironio-Acín), que sutilitzen per trobar fites als valors que maximitzen quànticament la violació de les desigualtats de Bell. Pel cas en què hi ha un gran nombre d'entrades per cada part, es demostra lavantatge tant en memòria com en temps envers els solucionadors més avançats utilitzant una combinació de diverses tècniques doptimització. La tercera línia és la de la *física quàntica de molts cossos*, que abraça un ampli ventall de temes. La tesi considera els problemes de fitar els valors esperats dels observables en els estats estacionaris de sistemes quàntics oberts, per trobar millors aplicacions de fermions a qubits i resoldre el problema de coloració de grafs mitjançant un nou algorisme doptimització inspirat en qudits. En cada cas, es demostra un avantatge respecte a mètodes comparables.

Acknowledgements

Thanks to everyone in the QIT group at ICFO for providing both academic support as well as a wonderful (and ever-changing) group of friends. I won't ever be able to look at a futbolin table without thinking of you all. In no particular order:

- to Toni for always giving ideas and help whenever I needed it
- to Máté for helping me a lot when I first started the PhD
- to Leo, Hippolyte, and everyone else who was ever in office 374 and had to listen to my ramblings about some silly new idea
- to Gaël, who had to keep listening to the ramblings even when we got home
- to Donato, Tim, Pere, David, Máté, Andreas, Nacho, Teo, Leo, Grazia, Korbinian, Maria, Irénée and Toni for being great co-authors
- to Pere and Leo for the help translating the thesis abstract
- to Cristian, Carlos and Miqueleta for the general thesis advice
- to Teo and Júlia for always being there for breakfast
- to Tim, Gaël, Cristian, Hippolyte, Leo, Mariana, Marcio, Ranieri, Carlos, Fionnuala, Máté, Vicky, Younes, Rafaelle, Nacho, Tomás, Anna, Aurora, Gustavo, Giacomo, Lorenzo, Ruth, Edward, Pedro, Luca, Tommaso, Marco and Raja for all of the events outside of the office

Thanks too to all my non-ICFO friends (both human and *animal*):

- to my various flatmates throughout the years (Eliza, Lena, Gaël, Yiannis, Ingrid, Ana, Pablo, *Merlin*) for making it a flat that I was always happy to come home to

- to my D&D group (Mattias, Matt, Callum, Martha, Rhys, *Nutmeg*) for keeping me busy every Wednesday evening for the past 5 years

- to the various friends of the flat: Steven, Fiorella, James, Gill, Michael, Ionah, Maria, Niki, Marianda, Elina, Sam, Christos, Odysseas, Nicola and many many others

- to Max and Max, respectively

Finally, thanks to my family (Mum, Jo, Ric, *Sam*, *Kenny*, *Benny*) for always being there for me - I know I don't call often, but that doesn't mean I don't enjoy when I do.

Contents

List of Publications and Preprints

Three numerical approaches to find mutually unbiased bases using Bell inequalities
MP Colomer, ***L Mortimer****, I Frérot, M Farkas, A Acín*
Quantum 6, 778, 2022

Mutually unbiased bases as a commuting polynomial optimisation problem
L Mortimer
arXiv:2308.01879, 2023

Reducing Entanglement with Physically Inspired Fermion-To-Qubit Mappings
T Parella-Dilmé, K Kottmann, L Zambrano, ***L Mortimer****, JS Kottmann, A Acín*
PRX Quantum 5 (3), 030333, 2024

Qudit inspired optimization for graph coloring
D Jansen, T Heightman, ***L Mortimer****, I Perito, A Acín*
Phys. Rev. Applied 22, 064002, 2024

Bounding Large-Scale Bell Inequalities
L Mortimer
Phys. Rev. A 111, 052442, 2025

Certifying steady-state properties of open quantum systems
L Mortimer*, D Farina, G Di Bello, D Jansen, A Leitherer, P Mujal, A Acín*
arXiv:2410.13646, 2024

List of Figures

List of Tables

List of Abbreviations

QIT	**Q**uantum **I**nformation **T**heory
SDP	**S**emi**D**efinite **P**rogram
PSD	**P**ositive **S**emi**D**efinite
MUB	**M**utually **U**nbiased **B**asis
MUSB	**M**utually **U**nbiased **S**ub**B**asis
NPA	**N**avascués, **P**ironio, **A**cín (referencing [1])
VQE	**V**ariational **Q**uantum **E**igensolver
NISQ	**N**oisy **I**ntermediate **S**cale **Q**uantum
QAOA	**Q**uantum **A**pproximate **O**ptimization **A**lgorithm

Chapter 1

Introduction to Quantum Information Theory

We begin with a chapter introducing Quantum Information Theory (QIT), covering broadly the history of the field, its applications, and the main research areas within it. We will also provide a brief overview of the mathematical background required to understand the quantum aspects of the rest of the thesis. If you come from any field with "quantum" in the name, you can probably safely skip this chapter.

1.1 Background

In science fiction, "quantum" is a word that gets thrown around a lot, often being used as a buzz-word simply to demonstrate that something is cutting-edge and futuristic. This reputation is not without some truth, as most quantum problems are significantly more complex than their classical (i.e. non-quantum) counterpart.

Before considering why, we first briefly discuss the origin of quantum mechanics. Even as early as 1670, scientists such as Isaac Newton were already describing the ability of light to behave as a particle. The ability of it to also act as a wave was later demonstrated by Thomas Young in 1801 with the double-slit experiment, which showed that light could demonstrate clear inference patterns and thus must also be able to behave in a wave-like manner. By the late 1800s there was clear disagreement in the expected thermal radiation of an object, often called the "ultraviolet catastrophe", which was eventually resolved by Max Planck in 1900 [2]. Planck proposed that the energy of light was quantized, meaning that it could only take on certain discrete values. This unit of quantized light later became known as a quantum (plural: quanta), and as such this theory became the first quantum theory in physics, winning Planck the Nobel Prize in 1918 [3].

Following Albert Einstein's subsequent explanation of the photoelectric effect using quanta in 1905, a number of quantum results quickly followed: spin quantization by Stern and Gerlach in 1922 [4], de Broglie's wave-particle duality in 1924 [5], and the Heisenberg uncertainty principle in 1927 [6]. Perhaps the most important result was Schrödinger's wave equation in 1926 [7], which provided a way to describe the wave-like behaviour of particles. This equation became the cornerstone of quantum mechanics, as it describes how a closed quantum system evolves over time.

Quantum mechanics was not without its critics, however. A "paradox" existed for many years regarding whether the universe could truly be local and deterministic, as classical physics suggested, whilst still reproducing the results demonstrated by the quantum experiments. This paradox was known as the Einstein-Podolsky-Rosen (EPR) paradox [8], and was eventually resolved by John Bell in 1964 who showed that a local hidden variable model could not reproduce the results of quantum mechanics and thus in order to reproduce the observed experimental results one must

consider a non-local model for our universe [9]. This result was later confirmed experimentally by Aspect et al. in 1982 [10], winning the Nobel prize in 2022 [11].

Since these early days of quantum theory, research has continued to accelerate. Just as we eventually learnt to control and utilise the flow of classical information with the field of information theory (leading to the development of the computer), so too are we learning to control and utilise the flow of quantum information with the field of quantum information theory (leading to the development of the quantum computer). Due to the various unique properties that quantum mechanics allows access to, it seems intuitive that being able to utilise such things as superposition and entanglement should be able to provide some advantage. Richard Feynman was once quoted as saying: "Nature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical" [12].

1.2 Main Research Areas and Applications

Since then, quantum information theory quickly spread into a number of different sub-fields, each of which with its own challenges and applications:

The most fundamental is the field of **Foundational Quantum Theory**, which seeks to understand the basic principles of quantum theory. This field is concerned with understanding the nature of quantum systems, how they differ from classical systems and how we might extract advantage from these differences. Whilst this often does not have direct applications, it is crucial for furthering our understanding and answering long-standing questions, involving topics such as causality [13] and contextuality [14]. Whilst many of the results are still quite analytical, there are still many problems in which one needs to perform very difficult optimisations.

Another broad field is that of **Many-Body Quantum Systems**, which seeks to understand the behaviour of large quantum systems. Applications include advances in quantum chemistry [15] (potentially improving the ability to test molecular interactions and thus drug discovery), the development of new materials with unique quantum properties [16], as well as improving our understanding of how quantum systems behave as the scale increases [17]. This field often involves heavy use of numerics, since solving such systems analytically becomes completely impractical very quickly, and in recent has years seen a flood of machine-learning [18, 19, 20] and tensor-network based approaches [21, 22, 23].

Perhaps the field that gathers the most media attention is that of **Quantum Computing**, which seeks to develop new computational paradigms based on the principles of quantum mechanics. The idea is that by using the unique properties of quantum systems, such as superposition and entanglement, we could solve certain problems much faster than we could with classical computers [24]. This field could have the potential to revolutionise the way we solve problems, with the most famous application being that of Shor's algorithm [25], capable of breaking much of today's encryption if a large enough quantum computer were to be built. Other applications include solving more general optimisation problems [26], quantum machine learning [27], and simulating quantum systems [28].

Quantum theory also provides many new ways to secure information, leading to the field of **Quantum Communication and Cryptography**. This field is concerned with developing new cryptographic protocols that are secure against quantum attacks, as well as developing new ways to secure the transfer of information using the principles of quantum mechanics. The most famous application is that of quantum key distribution, which allows two parties to securely share a key that is secure against any eavesdropper, even if they have access to a quantum computer [29]. Other

applications include secure multi-party computation [30], secure identification [31], and secure voting [32].

Whilst there do exist many other sub-branches of quantum theory, to the author it seems that almost all of them can be placed into one or more of the above categories, although probably to the disagreement of somebody. This thesis will focus on problems in the first two categories, joined by the utilisation of similar optimisation techniques for each.

1.3 Mathematical Background

The most fundamental unit in quantum information theory is that of the **qubit**: the quantum version of the bit. Whilst a classical bit is limited to being in a state of either 0 or 1, a qubit can be in a combination of both at once. We use here vector notation, such that the two possible "classical" states are $|0\rangle$ and $|1\rangle$, our **basis**:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{1.1}$$

A qubit can therefore be any combination of these two:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \tag{1.2}$$

where α and β are complex numbers such that $|\alpha|^2 + |\beta|^2 = 1$, keeping the state **normalized**.

This is the most common choice of basis for qubits, known as the **computational basis**. The notation here is known as **bra-ket** notation, such

that $\langle x|$ is called a bra and $|x\rangle$ is called a ket, which represent the conjugate transpose of each other. The product $\langle x|y\rangle$ is the inner product between vectors $|x\rangle$ and $|y\rangle$, meaning one has to multiply all the elements pair-wise and then take the sum. For example:

$$\begin{aligned} &\text{if} \quad |\psi\rangle = \begin{pmatrix}\psi_1\\ \psi_2\end{pmatrix} \quad \text{and} \quad |\phi\rangle = \begin{pmatrix}\phi_1\\ \phi_2\end{pmatrix} \\ &\text{then} \quad \langle\psi|\phi\rangle = \begin{pmatrix}\psi_1^* & \psi_2^*\end{pmatrix}\begin{pmatrix}\phi_1\\ \phi_2\end{pmatrix} = \psi_1^*\phi_1 + \psi_2^*\phi_2 \end{aligned} \tag{1.3}$$

When a qubit takes a value that isn't just $|0\rangle$ or $|1\rangle$, we say that it is in **superposition**. To visually represent all the possible states of a qubit, we can use something known as the **Bloch sphere**. This is a sphere where the poles represent the states $|0\rangle$ and $|1\rangle$, and all other points on the surface represent superpositions of these states. This is also a useful representation since it demonstrates that the state remains normalized (all points on the surface have the same magnitude) and also how performing operations on a qubit simply rotates it to another point on the Bloch sphere. More specifically, a state can be parameterized as $|\psi\rangle = \cos(\theta/2)\,|0\rangle + e^{i\phi}\sin(\theta/2)\,|1\rangle$, where θ and ϕ are the two angles used to represent the state on the sphere. An example of a Bloch sphere is shown as Figure 1.1, showing a few arbitrary states as examples. Note that here we ignore global phases.

Given a qubit, we can then perform **operations** on it to change its state. These operations are represented by **unitary matrices** which can apply to a state in vector form to get the new, transformed, state. A unitary matrix is a matrix U which when premultiplied by its conjugate transpose forms the identity matrix, such that $U^\dagger U = \mathbb{1}$. Some examples are the

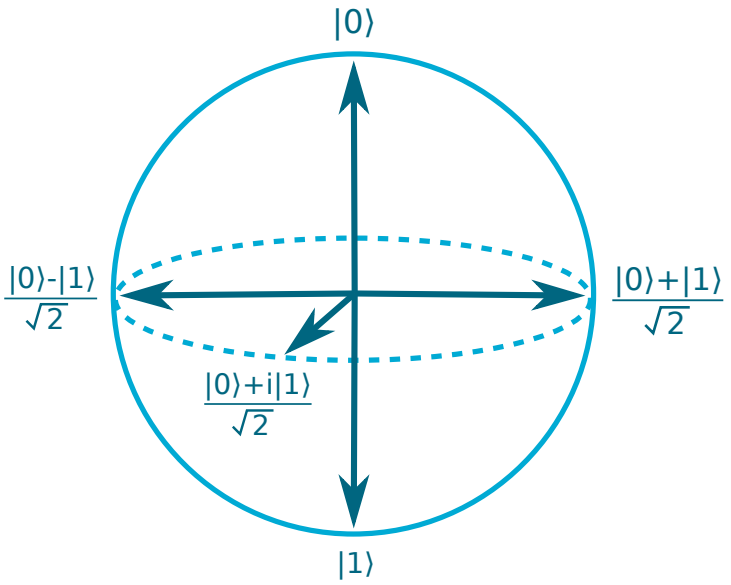

FIGURE 1.1: A Bloch sphere showing a few arbitrary states of a qubit.

Pauli matrices:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{1.4}$$

These can also be referred to as **gates** in the context of quantum computing, following the classical terminology of applying logical gates (e.g. AND, XOR). For example, applying the X gate to a qubit in the state $|0\rangle$ will transform it to the state $|1\rangle$, and vice versa, functioning like the

classical NOT gate:

$$X|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle \tag{1.5}$$

When we have multiple qubits, and assuming that the state is **separable**, we can represent the overall state as a **tensor product** of the individual qubits. For example, a simple two-qubit state $|\psi\rangle$ can be written as $|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$, where $|\psi_1\rangle$ and $|\psi_2\rangle$ are the states of the individual qubits. For instance, say we had a two qubit system in which the first qubit is in state $|1\rangle$ and the second qubit is in state $|0\rangle$, the overall system would be in state:

$$|\psi\rangle = |1\rangle \otimes |0\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ 1 \begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \tag{1.6}$$

For the sake of simplicity, we often write such **product states** without the tensor product symbol since it's implied, so the above might be written as $|1\rangle \otimes |0\rangle = |10\rangle$, as though using binary notation. We can then have superpositions of product states, for instance the state:

$$|\psi\rangle = \frac{1}{\sqrt{2}} (|00\rangle + |11\rangle) = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \tag{1.7}$$

This state has the quantum property of being **entangled**, which colloquially means that by measuring one qubit one could infer something about the other qubit. Here, for instance, if you measured the value of the first

qubit and saw that it was in state $|1\rangle$, you would know (without having to do any more measurements) that the second qubit also has to be in state $|1\rangle$. Put more technically, a pure state is entangled if it cannot be written as a product state, for instance if a state between two parties A and B cannot be decomposed into a tensor product of local states for each party, such that $|\psi_{AB}\rangle \neq |\psi_A\rangle \otimes |\psi_B\rangle$, then it is said that A and B are entangled.

Whilst here we are simply using the states $|1\rangle$ and $|0\rangle$ for simplicity, in reality the above vector could be any 4 complex numbers (plus the constraint that it remains normalized). This provides the first indication that the quantum world is going to be more difficult to handle than the classical, since with every extra qubit we need twice as many complex numbers to represent the full state. This is fine for small numbers, but when we have 20 qubits we would need on the order of $2^{20} = 1,048,576$ complex numbers to represent the full state, which is at the limit of practicality for modern computers.

Say now that we wanted to represent a mixture of two states, for instance if we prepare the above state with 50% probability and the state $|\phi\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$ with 50% probability. This would be represented by the **density matrix**, a more general quantum object:

$$\begin{aligned} \rho &= \frac{1}{2} |\psi\rangle \langle\psi| + \frac{1}{2} |\phi\rangle \langle\phi| \\ &= \frac{1}{4} \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix} \end{aligned} \tag{1.8}$$

Note that such matrices should always have a **trace** of 1 and should be **positive semidefinite** (being Hermitian and having only non-negative eigenvalues). These constraints will be quite common in optimization

problems throughout this thesis, written as $\mathrm{tr}(\rho) = 1$ and $\rho \succeq 0$, respectively.

Imagine now we want to determine the probability that when measuring the first qubit in the above state we get a 1. For this we use the **Born rule** combined with a **projective measurement**. A projective measurement is an operator that "projects" the state onto the state we want to measure. In the same way that you are a 3D object, but the sun causes you to leave a 2D shadow on the ground, the sun here is effectively doing a projective measurement.

Mathematically, the projection operator onto a state $|x\rangle$ is given by the operator $|x\rangle\langle x|$, and the probability $p(x)$ of observing outcome x is given by the Born rule, basically applying the projection to the state and then checking the trace:

$$p(x) = \mathrm{tr}(|x\rangle\langle x|\,\rho) \tag{1.9}$$

More general measurements can be performed using a **positive operator valued measure** (POVM), which is a set of operators $\{E_x\}$ which sum to the identity and are each positive semidefinite (i.e. $\sum_x E_x = \mathbb{1}$ and $E_x \succeq 0 \quad \forall x$). The probability of observing outcome x is then given by $p(x) = \mathrm{tr}(E_x\rho)$.

Now, say we have a density matrix pertaining to a large system, but we only care about a small subsystem. We can take the **partial trace** of the density matrix, which is the trace over all the states we don't care about. For instance, if we have a two qubit system with density matrix ρ_{12} and we only care about the first qubit. We can then take the partial trace over the second qubit (also called "tracing out" the second qubit) to get the reduced density matrix ρ_1. This is effectively summing over all the states of the second qubit. If we consider $c_{|i,j\rangle\langle k,l|}$ as generic coefficients

corresponding to projectors $|ij\rangle\langle kl|$:

$$\rho_{ab} = \begin{pmatrix} c_{|00\rangle\langle 00|} & c_{|00\rangle\langle 01|} & c_{|00\rangle\langle 10|} & c_{|00\rangle\langle 11|} \\ c_{|01\rangle\langle 00|} & c_{|01\rangle\langle 01|} & c_{|01\rangle\langle 10|} & c_{|01\rangle\langle 11|} \\ c_{|10\rangle\langle 00|} & c_{|10\rangle\langle 01|} & c_{|10\rangle\langle 10|} & c_{|10\rangle\langle 11|} \\ c_{|11\rangle\langle 00|} & c_{|11\rangle\langle 01|} & c_{|11\rangle\langle 10|} & c_{|11\rangle\langle 11|} \end{pmatrix} \tag{1.10}$$

Then, in the case where subsystem b takes the value 0, we consider only projectors in which the second qubit is 0, therefore the reduced density matrix is as follows, scaling by the probability of this occurring to keep it normalized:

$$\rho_{a0} = \frac{1}{p(b=0)} \begin{pmatrix} c_{|00\rangle\langle 00|} & c_{|00\rangle\langle 10|} \\ c_{|10\rangle\langle 00|} & c_{|10\rangle\langle 10|} \end{pmatrix} \tag{1.11}$$

Similarly when subsystem b takes the value 1, the reduced density matrix is:

$$\rho_{a1} = \frac{1}{p(b=1)} \begin{pmatrix} c_{|01\rangle\langle 01|} & c_{|01\rangle\langle 11|} \\ c_{|11\rangle\langle 01|} & c_{|11\rangle\langle 11|} \end{pmatrix} \tag{1.12}$$

Summing over both outcomes of b scaled by their probabilities we get the reduced density matrix, having traced out system b:

$$\rho_a = p(b=0)\rho_{a0} + p(b=1)\rho_{a1} \tag{1.13}$$

$$= \begin{pmatrix} c_{|00\rangle\langle 00|} + c_{|01\rangle\langle 01|} & c_{|00\rangle\langle 10|} + c_{|01\rangle\langle 11|} \\ c_{|10\rangle\langle 00|} + c_{|11\rangle\langle 01|} & c_{|10\rangle\langle 10|} + c_{|11\rangle\langle 11|} \end{pmatrix} \tag{1.14}$$

Another important concept in quantum theory is that of a **Hamiltonian**, which is an operator that describes the energy of a quantum system. Given

some Hamiltonian H, the expectation value of the energy of the system in state $|\psi\rangle$ is given by $\langle\psi| H |\psi\rangle$. In many quantum problems we want to minimize the energy by changing $|\psi\rangle$ to find the **ground state** of the system - the state with the lowest energy. A Hamiltonian can be represented as a square matrix with width and height equal to the size of the state vector, so for an n-qubit system the Hamiltonian would be a $2^n \times 2^n$ matrix. From this the ground state can be directly calculated by finding the lowest eigenvalue of the Hamiltonian, using an expensive technique known as **exact diagonalization**.

As mentioned earlier, it quickly becomes difficult to write the full state or Hamiltonian for even a small number of qubits, however there do exist ways of representing them in a more compact form if the system allows it. One common way is to represent them in terms of **Pauli strings**, which are tensor products of Pauli matrices, as well as the identity. We are then able to represent any Hamiltonian as a sum of Pauli strings, which is especially efficient if the system is somehow "sparse", for instance:

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix} = X \otimes \mathbb{1} + \mathbb{1} \otimes Z \tag{1.15}$$

For the sake of brevity, especially when dealing with much larger systems, we choose to use the notation in which X_i means the Pauli X matrix at site i and the identity everywhere else (e.g. for a four qubit system $X_3 = \mathbb{1} \otimes \mathbb{1} \otimes X \otimes \mathbb{1}$). Meanwhile $X_i Z_j$ means Pauli X at site i and Pauli Z at site j and identity everywhere else (e.g. for a four qubit system $X_2 Z_4 = \mathbb{1} \otimes X \otimes \mathbb{1} \otimes Z$).

We will also often describe objects in terms of their **expectation values**. For instance, the expectation value of a Hamiltonian H acting on some state $|\psi\rangle$ represents the expectation value of the energy of the system in that state, given by $\langle\psi| H |\psi\rangle$. We can also have the expectation value of a Pauli string, for instance the expectation value of $X_1 Z_2$ is given by $\langle\psi| X_1 Z_2 |\psi\rangle$, or if we are considering a non-specific state we will often write this as simply $\langle X_1 Z_2\rangle$. For a mixed state one can use the Born rule again, such that $\langle X_1 Z_2\rangle = \mathrm{tr}(X_1 Z_2 \rho)$.

We can also recover a partial or full density matrix even if we only have expectation values of Pauli strings, however to get the full state we need all 4^n terms:

$$\begin{aligned}\rho = \frac{1}{2^n}(\mathbb{1} + \langle X_1\rangle X_1 + \langle Y_1\rangle Y_1 + ... + \langle X_1 Z_2\rangle X_1 Z_2 \\ + ... + \langle Z_1 Z_2 ... Z_n\rangle Z_1 Z_2 ... Z_n)\end{aligned} \tag{1.16}$$

Chapter 2

Introduction to Optimisation

We now continue with a chapter discussing the other main topic of the thesis: optimisation. We will introduce a number of concepts relevant to optimisation, some mathematical descriptions of common optimisation problems, as well as the standard solving techniques. We will also discuss the relationship between optimisation and quantum information theory, and the difficulties that often arise. As with the previous chapter, if you have a background in computer science or are already familiar with optimisation, you can probably skip this chapter.

2.1 Background

As early as the 1600s, Newton and Leibniz independently developed calculus [33], which provided one of the first tools to find minima and maxima of arbitrary functions. This was later expanded by Euler and Lagrange [34], who developed the calculus of variations, which allowed for the optimisation of functionals. Whilst originally these were purely a set of analytical

tools for small problems, the field of optimization as we now know it began to properly appear with the development of linear programming by Dantzig in 1947 [35], followed by von Neumann and Tucker developing the concept of a dual program for linear programming [36], as well as the study of optimizing over convex sets [37], topics that are critical in modern optimisation and that will be in heavy use throughout this thesis.

In the coming decades there would then be an explosion of new optimisation problems and techniques: nonlinear programming results by Karush, Kuhn and Tucker [38], integer programming by Gomory [39], combinatorial optimisation [40], eventually leading to the interior point method by Karmarkar in 1984 [41], which provided a huge speed-up for linear programming problems. More modern inventions include quadratic [42] and semi-definite programming [43], and even more recently machine learning applied to optimisation [44].

The applications of all of the aforementioned techniques are vast. One can frame almost any problem as some kind of optimisation. There are, of course, the standard (read: capitalistic, profitable) ones that are quoted in every optimisation paper: portfolio optimisation, warehouse logistics, taxi booking and airline scheduling. In this thesis will we often deal with more theoretical problems, those which do not have immediate applications, but perhaps with deep study could provide insights into the workings of the universe. One can definitely argue that such problems are less immediately important than those of a medical/social/environmental nature. However, research is far from a zero-sum game, and having research into fundamental problems often leads to techniques that have a variety of cross-disciplinary applications.

2.2 Main Concepts

We will now cover some common concepts in optimisation, specifically the ones used frequently in this thesis. Perhaps one of the easiest ways to split all optimisation problems into two categories is by whether they are **convex** or **non-convex**. The easiest way to show the difference is by a diagram, as in Figure 2.1, but a more specific description is that in a convex set, the midpoint of any two of its points is also in the set.

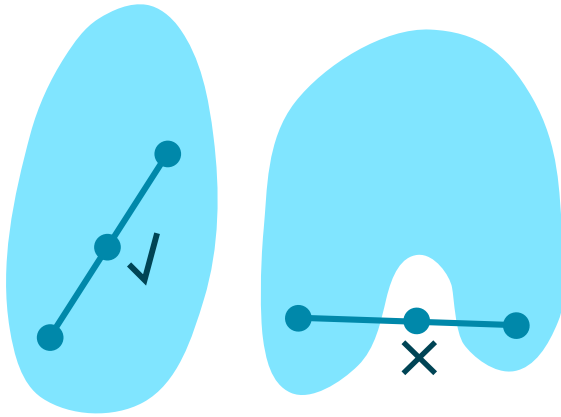

FIGURE 2.1: The left diagram shows a convex set, where the midpoint of any two of its points is also in the set. The right diagram shows a non-convex set, where this is not the case.

As one can perhaps imagine from simply viewing Figure 2.1, convex optimisation problems tend to be much easier than non-convex ones. In a convex optimisation problem, for instance finding the lowest point in the leftmost set in the figure, one simply can travel downwards, adjusting slightly if they hit a wall. Doing this, we have guarantees that we will eventually find the lowest point, and when we do finally settle into some

minimum we have a guarantee that we have found the best possible solution. This is often stated as saying that in a convex problem all **local minima** are equal to the **global minima** [45].

The same cannot be said for non-convex sets. As one can see from the rightmost diagram in Figure 2.1, if we start on the right side and then simply travel downwards, we will reach a local minimum, but it will not be the global minimum since there exist other points in the set that are lower. If we try to move around, the walls of the set will push us back into our local minima, thus it would seem as though we had found the optimum, when really we had not. We *could* get lucky when optimising a non-convex problem and still find the global optimum, but for larger problems the chance of this happening quickly decreases.

Another way to classify optimisation problem is that of **complexity classes**. This is a branch of computer science which seeks to classify problems into different classes based on the time or memory required to solve them as the problem becomes larger and larger. As this is a very broad topic, we won't go into so much detail here, but there are two classes that will be useful to know to understand the rest of this thesis: **P** and **NP**.

P represents the class of problems which can be **solved in polynomial time**: if such a problem has a size of n, then the worst-possible time and space required to solve it will be some polynomial of n. It could be that it needs $2n$ seconds. It could also be that it needs n^{100} years. The only requirement is that it scales polynomially with n. A simple example is finding a specific item in a list of n items - the longest this should take is n since in the worst case the item we want to find is at the end, in which case we would need to check all n items before we find the correct one. We often use **Big-O notation** to represent this, in this case that checking a list scales as $O(n)$ (i.e. linearly with n, but we don't care about units, factors or constants: it could be $n + 3$ seconds or $5n$ milliseconds).

NP, meanwhile, represents the class of problems that can be **verified in polynomial time**. This means that if someone gives you a solution to an NP problem, you can check if it's correct in polynomial time. For instance, if you are given the solution to a sudoku puzzle, you can check if it's correct by looking at each row, column and square to see if it contains all the numbers from 1 to 9. Easy to check that it's correct, but not necessarily easy to find. There are also some commonly used classes related to NP, such as **NP-complete** which means that it is equivalent in complexity to the hardest problems in NP, or **NP-hard** which means that it is at least as hard as the hardest problems in NP.

It should be noted that the class of NP entirely *contains* the class of P, meaning any problem in P is definitely in NP (if it's easy to solve, it must be easy to check). The reverse is not true, and it remains one of the most famous open problems to determine whether the sets of P and NP are equal: whether there really does exist any problem that can be checked easily but not solved easy. Of course, all evidence points to this being true, as there exist many problems for which we have never found any algorithm faster than exponential. Relating back to the idea of convexity, many convex problems are known to be in P (for instance any which can be reformulated as a polynomially-sized linear program), whilst many non-convex problems are known to be in NP (for instance the travelling salesman problem).

Another important concept is that of **duality**: for any given optimisation problem (which we will call the **primal**), there exists a **dual** problem. In the ideal case, the optimum of the dual problem gives the same value as the optimum of the primal, known as having no **duality gap**. This is the case if a number of conditions are met, for instance in the case of convex problems there is never a duality gap as long as a solution exists [46]. But why is this useful? In many cases, solving the dual problem at the same time as the primal can give better overall convergence [47], as well

as certification that we have found the optimum solution since the dual approaches the solution from a different direction: if the primal is trying to minimize a function, the dual will be maximizing a different function, hopefully meeting at the same point. A visual example of duality is shown in Figure 2.2.

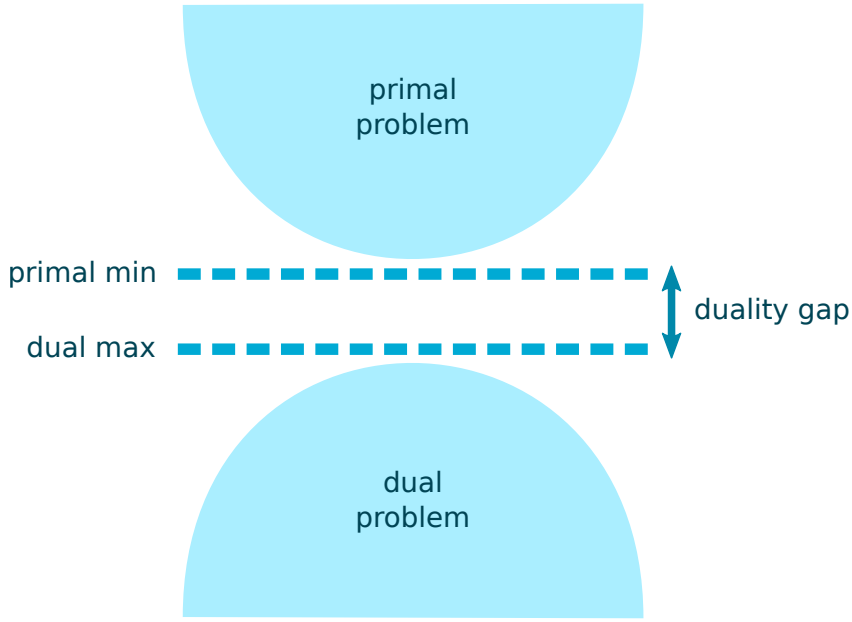

FIGURE 2.2: For some primal problem minimising a function, there exists its dual problem maximising a different function. The dual uses a different set of variables than the primal, so this diagram is an oversimplification, but the values of their objectives do indeed behave as in the diagram. Here there is a non-zero duality gap, a difference in their optimum values, but in some cases (e.g. feasible convex problems) this will instead be zero.

It's also important to mention the idea of a **relaxation** of a problem. Whilst optimising over a non-convex set might be difficult (due to local minima or weird properties), optimising over a slightly larger but convex set is generally easier. This larger set containing the original is known as a relaxation of the problem, and gives a bound on the optimum value. Say we are trying to minimize a function over a set, but then we consider a relaxation of said set. The value we get when minimizing the relaxation will be smaller than or equal to the minimum of the original problem. Or, put another way, we have found a lower bound for the true minimum. Similarly we can define a contraction as the opposite idea, when one finds a simpler subset of the original set. Many concepts like this are easier to show visually, for that see Figure 2.3.

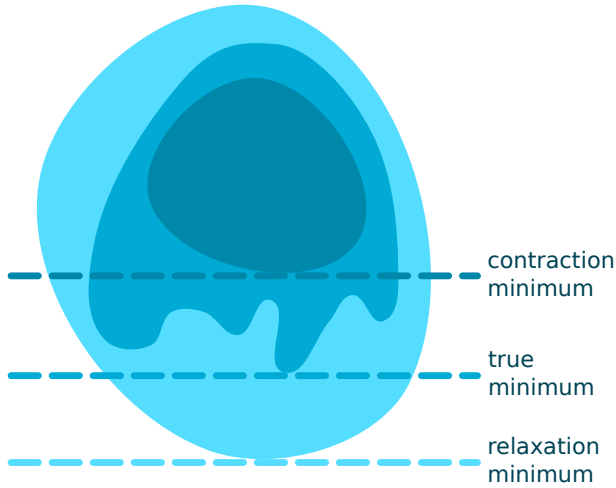

FIGURE 2.3: A non-convex set and examples of a convex relaxation (lighter) and convex contraction (darker). Assuming we are trying to find the minimum of the non-convex set, the relaxation gives a solution lower than the minimum and the contraction gives a solution higher than the minimum. We can therefore say with certainty that the true minimum lies between these two values.

With all of these possible classifications in mind, we can now begin to consider the most **general optimisation problem**, in which we try to minimize an objective function $f(x)$, subject to the constraint that the variable x remains within the set $\mathcal{S}$ we want to consider. Mathematically this is often written as follows:

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & x \in \mathcal{S} \end{aligned} \tag{2.1}$$

In this notation we don't require that the problem has any specific properties: the set $\mathcal{S}$ could be convex or non-convex, it could be easy or hard, x could be a giant matrix or just a single number. A more specific formulation of an NP problem is the **Ising model**, where we have a series of "spins" which can either be up or down, and we might assign a penalty to any two neighbouring spins that are both pointing in the same direction. For instance, we might say that if spin 1 and spin 2 are both pointing in the same direction, that gives some penalty. We then want to find the configuration of spins that minimizes the total penalty. The Ising model is known to be NP-complete [48], meaning that any problem in NP can be reformatted as an Ising model (although it will often be less efficient to do so). Mathematically, it looks as follows:

$$\begin{aligned} \text{minimize} \quad & \sum_{i,j} J_{ij} x_i x_j \\ \text{subject to} \quad & x_i \in \{-1, 1\} \end{aligned} \tag{2.2}$$

Here J_{ij} is the coefficient linking spin i and spin j, if it's negative it means that we want the spins want to be aligned (e.g. 1 and 1), if it's positive they want to be anti-aligned (e.g. 1 and -1). This is an example of **integer optimisation**, specifically **binary optimisation** since the spins can only be -1 or 1. See Figure 2.4 for a visual example of the problem.

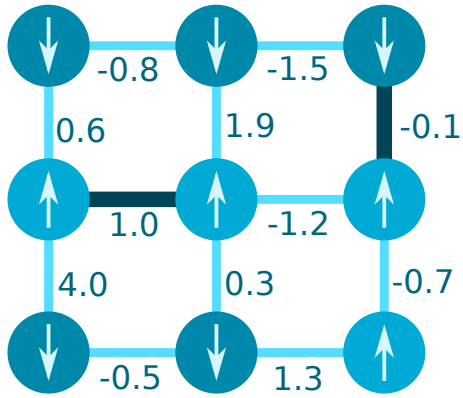

FIGURE 2.4: An example of an Ising model, where each spin is either up or down, and each link is given a coefficient. In this case, the configuration displayed is the global minimum, found by a brute force search, with a optimum value of -11.7. The two darker couplings are those that are those that are giving a penalty (frustrated) with the current configuration.

2.3 Optimisation Techniques

We now move on to consider the ways one can attempt to solve an optimisation problem. One format for representing any problem in P is that of linear programming [49]. Here one wants to find the minimum or maximum of some linear function, subject to some affine constraint:

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax \leq b \end{aligned} \tag{2.3}$$

Here x is a vector representing our variables, c represents our objective, then the matrix A and vector b represent our affine constraint. A linear function is one that doesn't move the origin (e.g. mapping $0 \to 0$), whilst an affine function can (e.g. mapping $0 \to 1$). When we talk about linear programming we tend to consider affine and linear functions to be interchangeable, since they generally don't change the hardness of the problem. The search space (or **feasible region**) for this kind of problem is always a **convex polytope**, meaning a convex shape formed of only straight edges, where the optimum of our problem will always be a vertex. We show a diagram of an example linear program in Figure 2.5.

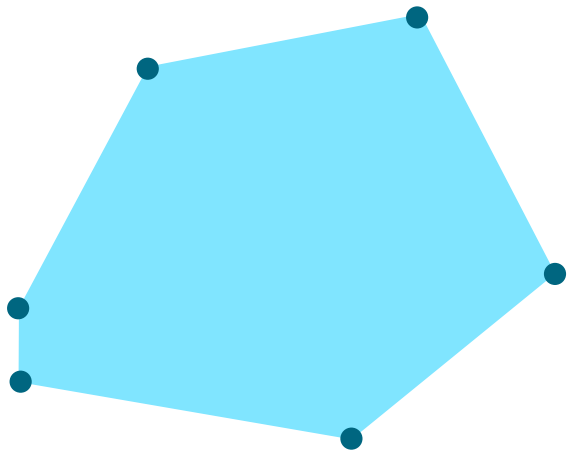

FIGURE 2.5: An example of the feasible region for a linear program: a polytope. No matter which direction you pick for the objective, the optimum will be a vertex. It could be that the objective is perpendicular to an edge, but in this case one can choose either vertex of that edge as the optimum.

The original method for solving linear programs is the **simplex method** [50]. The method works by starting at a vertex of the feasible set, then moving to an adjacent vertex that improves the objective function. This is repeated until no further improvement can be made, at which point the algorithm terminates. The simplex method is known to be very efficient in practice, but it can be exponentially slow in the worst case [51].

A major advancement was the development of the **interior point method**. The first example of this was the ellipsoid method [52], which was the first method known to be polynomial time in the worst case, thus demonstrating that linear programming lies in class P. The general idea of an interior point method is to start at a point inside the feasible set, then move towards the optimum whilst always remaining inside the set. This idea can also be expanded to most other optimisation problems, including non-convex ones.

Another optimisation problem also known to be in P is that of **semidefinite programming**. In this case we want to minimize or maximize a linear function subject to a constraint that a matrix is positive semidefinite, as well as the usual affine constraints as with linear programming. The constraint that a matrix is positive semidefinite is a convex constraint, thus when combined with the affine constraints (also convex) the feasible set remains convex. The difficulty comes from the fact that we don't have the simplicity of a polytope anymore - our set contains curved edges and thus we can't just check a finite amount of vertices. A general SDP looks like the following, where now C and B are matrices rather than vectors and we now have some index i iterating over the constraints:

$$\begin{aligned} \text{minimize} \quad & C \cdot X \\ \text{subject to} \quad & X \succeq 0 \\ & A_i X = B_i \quad \forall i \end{aligned} \tag{2.4}$$

Here the symbol "$\succeq$" means that the matrix X is positive semidefinite, the symbol "$\cdot$" represents the inner product between matrices (element-wise multiplication then sum), and the symbol "$\forall$" means "for all" and iterates over the constraints, however many we may have.

There exist a number of methods to solve semidefinite programs (SDPs), but the most common is, again, the interior point method. We begin with a point inside the feasible set then proceed in the direction of the objective. In many cases this means checking that the matrix remains positive at each step, something that will prove to be quite costly for many of the problems later in the thesis. To find the eigendecomposition of a matrix we normally require on the order of $O(n^3)$ operations where n is the width of the matrix. For small matrices this is trivial, but when we begin to talk about 1000×1000 matrices this can quickly become an issue.

There is also another distinction within the interior point methods: whether they are first-order or second-order. **First-order solvers** simply take the derivative of the objective (in this case just the matrix C) and travel in that direction, often with some adjustment based on how close they get to the edge of the set to prevent leaving the feasible region, known as a barrier function. This is known to have **linear convergence** [53], such that the error (distance from the optimum) at every step is some fraction of the previous error. For instance, we might half the distance away from the optimum at every step, meaning to go from 1 to 10^{-10} we would need 33 steps. Some examples of common first-order solvers for SDPs include SCS [54] and SDPA [55].

Meanwhile, **second-order solvers** take the second derivative of the objective (known as the **Hessian**) and use this to determine the direction to travel based on Newton's method for root finding. This is known to have **quadratic convergence** [56], meaning that the error at every step is some fraction of the square of the previous error. For instance, if at

each step we square and then half the error, we go from 1 to 10^{-10} in only 5 steps. The downside of second-order methods is that they are often more computationally expensive, since they require the calculation of the Hessian, which is expensive both in terms of time but also memory (for instance with a 1000×1000 matrix, we could have 1 million variables, thus the Hessian would be on the order of 1 million $\times$ 1 million). One common second-order solver for SDPs is MOSEK [57] and is the one we make most significant use of later in the thesis.

So far we've only discussed convex solvers. For non-convex problems, there exist a whole range of possible techniques, none of which have ever been proved to converge in polynomial time in the worst case (as this would prove P=NP). There are probabilistic methods like **simulated annealing** [58] or **genetic algorithms** [59], methods with slower but guaranteed convergence like **branch and cut** [60], or one can simply try a method better suited for convex optimisation like **gradient descent** or it's variations (such as **ADAM** [61] or **BFGS** [62]), which can be used to quickly find a local minimum (but probably not the optimum unlike the previous methods).

Regardless of the choice of solver, throughout this thesis we make regular use of **parallelisation**, a computational technique in which a number of tasks are performed at the same time. Most modern computers have at least 4 cores in their processor - 4 separate computational units that can each be performing a different set of instructions at the same time, whilst sharing certain parts of memory. When dealing with larger computers the number of cores grows significantly, with the largest supercomputer at the time of writing having over 8 million cores [63]. The challenge is then to decide how to separate an algorithm into many equally-sizes sets of instructions, the ideal being that each core works fully independently and very rarely has to slow down to talk to the others. In the case of iterative

solvers, the slowest part is often something related to matrix multiplication or eigenvalue calculation, for which parallelised algorithms are known [64].

2.4 Application to Quantum Information Theory

We can now begin to consider how all these various optimisation problems and solving techniques can be applied to quantum information theory. Whilst many problems in the quantum world can be quite directly mapped to a standard optimisation problem, the obvious mapping usually results in something exponentially large due to the fact that representing a full state or density matrix requires an exponential amount of memory. As such, in many quantum optimisations people often only consider the small cases (i.e. less than 10 qubits) when solving exactly, done via finding the eigendecomposition of the Hamiltonian to get the global optimum. Meanwhile for larger cases, one normally has to truncate/approximate/relax the quantum set to create a much easier optimisation problem.

This does, of course, make a lot of sense. If we could simulate all quantum systems exactly with a classical computer, then this would eliminate any idea of so called "quantum advantage", whereby quantum devices can do something that classical devices would struggle with. Assuming that quantum devices are somehow more powerful than classical devices, it then becomes a fundamental rule of nature that we won't be able to efficiently solve exactly large quantum problems with a classical computer. Nevertheless, we try, although on very different scales compared to classical optimisation. In modern classical optimisation one can find the global optimum for NP-complete problems of 10000 variables [65], whilst in the quantum world we will instead be happy if we can even get within 1% error for 30 qubits.

Perhaps an easier goal than finding the global optimum for a quantum problem is the idea of bounding the solution. Using something in the

style of Figure 2.3, we can often use two different classical methods to get a upper and lower bound on a quantum quantity of interest. This can then be used to benchmark versus a quantum device, to ensure that the quantum device (which is inherently somewhat probabilistic) gives a solution within the range, hopefully faster than the classical algorithms. For instance, one application of quantum computers could be to use QAOA (Quantum Approximate Optimization Algorithm) [26] to attempt find the ground state of a large-scale Ising model. By consider a classical relaxation and contraction of the problem we can obtain bounds that could then be used to certify the optimality of the output of such an algorithm, which due to the probabilistic nature of quantum algorithms, might not always give the correct answer [66].

We finish this section by noting that classical optimisation is a critical part of quantum information theory, and that the two fields are deeply intertwined. Being able to exactly solve small systems and approximate large ones is of great use to all sub-fields of quantum theory, and can help pave the way for the quantum devices of the future to (hopefully) out-perform these classical algorithms.

Chapter 3

Mutually Unbiased Bases

Having covered the basics of the two fields we intend to combine, we now move to the first of the specific topics. In this chapter we begin with a summary of mutually unbiased bases as well as the problem of finding 4 such bases in dimension 6 - a well-known open problem in quantum information since 1960 [67]. We will then discuss the work done for this thesis aimed at trying to tackle this problem, first in the context of a Bell inequality and then more generally as both commuting and non-commuting optimisation problems.

3.1 Background

For a given space, one can define a basis by considering the vectors that one could combine to get anywhere in said space. For a 2-dimensional space this simply requires two vectors of size two, for 3-dimension space three vectors of size three, and so on, with n-dimensional real space requiring n vectors each containing n elements. An example basis for 2D real space

could be the following:

$$a_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad a_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{3.1}$$

This allows one to reach any point in the space by combining the two vectors in some scaled linear sum, guaranteed due to their linear independence, a property implied by the fact that these vectors are normalized (each has a magnitude of 1) and orthogonal (their inner product is 0). Another equally valid basis is:

$$b_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \qquad b_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix} \tag{3.2}$$

However, these two bases also have an interesting property when we consider their inner products - they are known as **mutually unbiased bases** (MUBs). This is where the inner products of all vectors between the two bases have a magnitude of $\frac{1}{\sqrt{d}}$, where d is the dimension of the space. In the case of the above two bases, one can see:

$$|a_1 \cdot b_1| = |a_2 \cdot b_2| = |a_1 \cdot b_2| = |a_2 \cdot b_1| = \frac{1}{\sqrt{2}} \tag{3.3}$$

We demonstrate this visually in Figure 3.1, where it can be seen that this corresponds to one basis being equiangular versus the other. This property can also be extended to larger sets of bases, where we might have many bases such that we want every inner product between any two vectors of different bases to have a magnitude of $\frac{1}{\sqrt{d}}$.

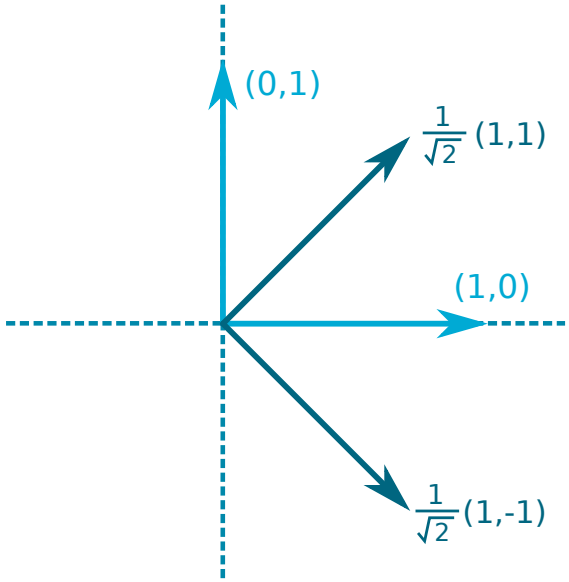

FIGURE 3.1: Two mutually unbiased bases in 2D real space, such that the inner product of vectors between orthonormal bases is equal to $\frac{1}{\sqrt{2}}$. Note that one could rotate all vectors by the same angle and it would still be valid, thus demonstrating a rotational symmetry.

Now, moving from real to complex, we change from the dot operator to using bra-ket notation. Assuming we have a series of n bases each with d vectors, $|v_{ij}\rangle$ where the first index is the basis (from $1 \to n$) and the second index is the vector in said basis (from $1 \to d$), we can then combine with the requirement for orthogonality and normalization and define the

general problem as:

$$\begin{cases} |\langle v_{ik}|v_{jl}\rangle| = \frac{1}{\sqrt{d}} & i \neq j \\ |\langle v_{ik}|v_{jl}\rangle| = 1 & i = j, k = l \\ |\langle v_{ik}|v_{jl}\rangle| = 0 & i = j, k \neq l \end{cases} \tag{3.4}$$

Such bases have significant use in quantum information theory, providing measurements maximising Bell inequalities [68, 69], their use in quantum state tomography [70, 71], or their use in secure quantum communication protocols [72]. This is due to the property that preparing in one basis and measuring in a mutually unbiased basis results in a uniform probability of any outcome, making it difficult for an attacker to gain information.

The question that one can then ask is: for some given dimension d, what is the maximum number of mutually unbiased bases that we can have? A general upper bound of $d+1$ MUBs is known, however this is only known to be achievable in the case of d being a prime power [73]. For example, when $d = 2$ we can easily find 3 MUBs, when $d = 3$ we can find 4 MUBs, and so on. There are even analytical constructions for finding said bases [74]. However, for the case of $d = 6$ (the first non prime-power) we do not know the maximum number of MUBs that can be found, although it is generally believed to be 3, known as Zauner's conjecture. This is the well-known and well-studied open problem of MUBs in dimension 6, which even has a monetary prize if it were to be solved [75].

However, one should not be deceived by the relative simplicity of stating the problem, as the resulting equation set is non-convex, non-linear and quite large in the open case. In dimension 6 the set whose existence we want to prove/disprove has 4 bases, each with 6 vectors, each with 6 complex elements, thus we would have a *minimum* of $6 * 6 * 6 = 216$ complex variables. For each of these we need to constrain the magnitude of

their inner product (and thus conjugates, which adds complexity), leading to a quite high-order set of equations of many variables. Trying to brute-force search this space to certify that one cannot satisfy all constraints is computationally infeasible, and simply searching for MUBs and then not finding any does not qualify as proof of their non-existence, thus smarter methods are needed.

There have been a number of attempts at a solution since the problem was first put forward in 1960, such as Weyl groups [76], unitary operators over finite fields [77] and Hadamard matrices [76]. These methods are each able to find MUBs in the prime-power cases, however cannot work in other dimensions. In terms of numerics, a large number of computational searches have been performed [78, 79, 80], with all results suggesting (but not proving) that one cannot find 4 MUBs in dimension 6. As such, here we document the various new attempts that we have tested, as it became somewhat of an obsession at the start of the PhD (despite advice from several post-docs to not spend time on it). Whilst we do not solve the open problem in this thesis, the applied optimisation techniques and insights gained are still valuable contributions to our understanding of MUBs as well as the application of optimisation techniques in quantum information theory.

3.2 As a Commuting Optimisation Problem

We first begin by reframing the MUB problem as a commuting optimisation problem, which we then try to optimize using several different methods. The following section is all adapted from a single-author paper [81] written by the author of this thesis, which as of the time of writing is available on the arXiv but is still in the publication process.

3.2.1 The MUB Problem as Polynomial Optimisation

We consider the general case of polynomial optimisation, which consists of trying to minimize a real polynomial $f(\vec{x})$ over a vector $\vec{x}$, subject to polynomial equality constraints $g(\vec{x}) = 0$ and polynomial positivity constraints $h(\vec{x}) > 0$. These polynomials can (and in this section will be) non-convex and non-linear, thus allowing all problems in NP to be expressed [82]. If the polynomials were all convex, such a problem could be solved easily using convex optimisation techniques, many of which are known to converge to finite precision with polynomial time complexity [41, 83]. The idea of converting the MUB existence problem into a polynomial optimisation problem has been studied before in the non-commuting complex case [84] or in the case of studying MUB constellations [85].

Here an objective function $f(\vec{x})$ is unnecessary, since we are only trying to find any valid set of MUBs, rather than the set of MUBs which minimizes some function. The same is true for the positivity constraints, since in the MUB problem we only need to constrain that various magnitudes of inner products equal either 0, 1, or $\frac{1}{\sqrt{d}}$. Thus it instead becomes an equality constraint satisfaction problem. For the problem of finding n MUBs in dimension d, the problem can be defined as that of finding complex vectors $\vec{v}_{ij}$, where here the first index indicates the basis (from $1 \to n$) and the second indicates the vector within the basis (from $1 \to d$), such that they obey Equation Set 3.4.

In order to simplify this set of equations we introduce two new real variables, α_t and β_t, for each of the pairwise inner products. We also replace each element of each vector with a combination of the real and complex parts, such that element m of vector $\vec{v}_{ij}$ becomes $v_{ijm} = a_{ijm} + ib_{ijm}$, giving the following polynomial equation set over the reals:

$$\begin{aligned}
\sum_{m=1}^{d}(a_{ikm} - ib_{ikm})(a_{ikm} + ib_{ikm}) &= 1 \quad \forall i, k \\
\sum_{m=1}^{d}(a_{ikm} - ib_{ikm})(a_{ilm} + ib_{ilm}) &= 0 \quad \forall i, k \neq l \\
\sum_{m=1}^{d}(a_{ikm} - ib_{ikm})(a_{jlm} + ib_{jlm}) &= \alpha_t + i\beta_t \quad \forall i, j, k, l \\
\alpha_t^2 + \beta_t^2 &= \frac{1}{d} \\
t = (i-1)nd^2 + (j-1)d^2 + (k-1)d + l &
\end{aligned} \tag{3.5}$$

The introduction of α_t and β_t allows us to keep the equations second-order, rather than removing the norm by squaring the constraint which would allow for less variables but a fourth-order equation set. Testing both versions we found it is significantly better regarding both the memory and the time required for both of our algorithms if the order is kept as low as possible.

One can then expand the products, equating the real and complex sides of each equation, as well as combining all variables into a single vector for computational simplicity. Now we have a set of real equations in which every feasible point is a set of MUBs, thus proving infeasibility of the equation set for dimension d and number of bases n proves the inability to generate MUBs for that d and n. The number of variables quickly becomes

large, as shown in Table 3.1, which becomes a problem when discussing the global optimisation required to prove infeasibility.

n/d	2	3	4	5	6
2	24	54	96	150	216
3	48	108	192	300	432
4	80	180	320	500	720
5	-	270	480	750	-
6	-	-	672	1050	-
7	-	-	-	1400	-

TABLE 3.1: Number of variables when converting the mutually unbiased basis problem into a real polynomial optimisation problem, without any reductions, for the problems of interest. The formula for this is $2nd^2 + d^2n(n-1)$. Note that due to the non-convexity of the problem the worst-case complexity of such general polynomial optimisation is exponential in the number of variables, thus here the open question of $d = 6$ and $n = 4$ remains highly intractable.

3.2.2 Reducing the Number of Variables Using Symmetries

Whilst one could use the algorithms presented in this work to optimise the polynomial problem in its current state, we were unable to perform global optimisation for even small sets in dimensions 2 and 3. Luckily for us, MUBs have a large number of rotational and reflectional symmetries that we can exploit to significantly reduce the number of variables and equations.

The first of these reductions can be derived from the fact that a unitary rotation applied globally to a set of MUBs preserves the mutually unbiasedness [86]. As such, one can always choose the first basis to be any

valid orthonormal basis without loss of generality. For the sake of simplicity, we choose the computational basis mentioned earlier, which as well as removing a large number of variables from the optimisation also allows for a further simplification: given that all inner products must have a certain magnitude and given that each vector in the computational basis only affects one element of each vector, one can now explicitly constraint the magnitude of every element of every other vector to be $\frac{1}{\sqrt{d}}$, for example in dimension 2:

$$\begin{aligned} &\text{if} \quad \vec{v}_{11} = (1 \;\; 0) \quad \text{and} \quad \vec{v}_{12} = (0 \;\; 1) \\ &\implies \vec{v}_{21} = \frac{1}{\sqrt{2}}(e^{i\theta_1} \;\; e^{i\theta_2}) \end{aligned} \tag{3.6}$$

One can also choose to fix some of the rotational symmetries by fixing the first vector of the second basis, for instance to the uniform vector of $\vec{v}_{21} = \frac{1}{\sqrt{d}}(1 \;\; 1 \;\; ... \;\; 1)$. A similar argument can be made for fixing the first element of every vector to be $\frac{1}{\sqrt{d}}$, further reducing the number of variables we need to optimise. These can both be done without loss of generality due to the inner product being unaffected by the phases of the bases.

Continuing with the idea of symmetry reductions, one can consider a number of other symmetries which preserve the various inner products of the problem, specifically certain variable permutations. For instance, given a set of MUBs, swapping the first and second bases is guaranteed to preserve the mutually unbiasedness. The same can be said about swapping two vectors within a basis, swapping the first and second elements of all vectors, as well as taking the conjugate of everything. All of these are transformations which preserve the inner product. In order to reduce the size of the search space we define positivity constraints for each of the aforementioned symmetries, such that one can always swap the sign of

the constraint function by performing the corresponding transformation on the equation set, whilst retaining mutual unbiasedness.

For example, since it is known that we can swap vectors 1 and 2 in the 3rd basis without loss of generality, once can define some function of the form $h_1(\vec{v}_{31}, \vec{v}_{32}) = \sum_{m=1}^{d}(a_{31m} + b_{31m}) - \sum_{m=1}^{d}(a_{32m} + b_{32m})$, the difference between the component-wise sums of each vector. This can always be made positive by MUBs, since if a set of MUBs was to make this negative one could simply swap the two vectors, resulting in the negative of the function. A similar function can be defined for the conjugation symmetry: $h_2(\vec{b_{ij}}) = \sum_m b_{ijm}$. Given MUBs making this function negative one can take the conjugate of the MUBs resulting in the negative of all of the imaginary components b_{ijm}, thus flipping the sign of this function whilst retaining the mutual unbiasedness. Adding each of these functions as positivity constraints to the polynomial problem reduces the size of the feasible set without loss of generality in the infeasibility case.

3.2.3 Reducing the Number of Variables Using Sub-bases

Whilst we have reduced the number of variables significantly with the reductions so far, we are still considering the full equation set. If we now shift the focus to not finding full sets of MUBs, but instead showing that MUBs are infeasible, the question can be asked, is any subset of the equations infeasible? If this were the case the infeasibility would extend to the full equation set, whilst being much simpler to prove due to the reduction in number of equations. Here we focus on the idea of optimising sub-bases, which we define as a sequence of orthonormal vectors which does not contain enough vectors to fully span the space, thus making it a subset of some basis. For example, in dimension 3 one could consider the sub-basis of the computational basis:

$$v_{11} = (1 \;\; 0 \;\; 0) \qquad v_{12} = (0 \;\; 1 \;\; 0) \tag{3.7}$$

Since finding a valid sub-basis is a requirement for finding a valid basis, if a given problem does not permit sub-bases, it certainly doesn't permit full bases. As such we extend the notation with the mutually unbiased property, turning our search from mutually unbiased bases to the sub-problem of searching for mutually unbiased sub-bases (MUSBs), which we denote by listing the number of vectors per set. For example in dimension 3, the MUSB problem with sizes $\{3, 2, 1\}$ would imply the search for a full basis (of 3 vectors), a sub-basis of 2 vectors, plus a sub-basis of 1 vector, each being mutually unbiased with each other.

The choice of such set sizes, in order to minimize the size of the problem whilst still remaining infeasible in the cases that should be infeasible, is something that was determined by starting with the smallest MUSB sizes for a given d and n, then increasing the sizes until the problem appears to become infeasible as decided by the upcoming methods. Using the apparent minimum set sizes along with the aforementioned reductions results in a significant drop in the number of variables, as given by Table 3.2. For the open question of 4 MUBs in dimension 6 this results in a 83% reduction ($720 \rightarrow 122$).

Just as with the idea of breaking symmetries and fixing bases, we do not claim that this is a novel technique, in the case of MUBs being previously referred to as "constellations" in the works by Brierley and Weigert [79, 87]. Nevertheless, our results agree with the cases given in their work, specifically in dimension 6, providing further numerical evidence of the nonexistence of such MUBs.

n/d	2	3	4	5	6
2	0	0	6	8	50
3	2	4	14	28	119
4	6	10	24	42	122
5	-	18	36	58	-
6	-	-	50	76	-
7	-	-	-	96	-

TABLE 3.2: Number of variables when converting the mutually unbiased basis problem into a real polynomial optimisation problem, with all reductions. Here the set sizes used are those with the lowest number of variables whilst still retaining infeasibility in the presumed infeasible cases. The specific set sizes used to generate these values are given later in the results subsection.

3.2.4 Method 1: Search Space Without Local Minima

Given this reduced set of equations, we first considered methods to ensure that this correctly conveys the problem, finding MUBs in the cases where they should exist and failing to find MUBs in those where they are not. To do this we use the idea of combining our many second-order equality constraints into one giant fourth-order objective function, which could then be minimized without constraints. Consider first some general equation set in which we want to find a common root:

$$h_i(\vec{x}) = 0 \qquad \forall i \tag{3.8}$$

which we can then combine by squaring each equation and summing:

$$f(\vec{x}) = \sum_i h_i^2(\vec{x}) = 0 \tag{3.9}$$

Since each term has to be non-negative, the only way such a function can reach zero is if all of the squared equations also reach zero, thus transforming the problem into an unconstrained optimisation. Here we just use the equality constraints of the reduced equation set and ignore the positivity constraints used for enforce some symmetries, since these only provide a slight improvement to this method but add quite a bit of complexity. Our aim with this optimisation is to converge rapidly to MUSBs if they exist, with a lack of convergence serving as a suggestion that such a set of MUSBs (and thus a corresponding set of MUBs) is unlikely to exist.

Based on the method of Lagrange multipliers, we begin by integrating our polynomial $f(\vec{x})$ by one of our variables, x_m, specifically one that is known to be able to reach zero without loss of generality. In the case of the MUB problem this is fine, since we have a number of variables already assumed to be zero due to rotational symmetries, however equally one could create a new variable and integrate by it to create something that resembles a traditional Lagrange multiplier. Regardless of the choice of variable, the result is then a new fifth-order polynomial $f_m(\vec{x})$ with a number of interesting properties. Considering the partial derivatives of this function with respect to every variable, we find that one of these derivatives is our original function, whilst every other derivative has our integrating variable x_m as a factor. As an arbitrary example, if our original function was:

$$f(\vec{x}) = x_0^2 + x_0 x_1 + x_2^4 \tag{3.10}$$

and we choose to integrate by x_1, ignoring the resulting constant term:

$$f_1(\vec{x}) = \int f(\vec{x})\ dx_1 = x_1 x_0^2 + \frac{1}{2} x_0 x_1^2 + x_1 x_2^4 \tag{3.11}$$

which has partial derivatives:

$$\begin{aligned}\frac{\partial}{\partial x_0} f_1(\vec{x}) &= 2x_1x_0 + \frac{1}{2}x_1^2 \\ \frac{\partial}{\partial x_1} f_1(\vec{x}) &= f(\vec{x}) \\ \frac{\partial}{\partial x_2} f_1(\vec{x}) &= 4x_1x_2^3\end{aligned} \tag{3.12}$$

From this it can be seen that the set of partial derivatives can only be all zero, a stationary point, if our original function is zero. The other derivatives being zero can be guaranteed to exist if the variable x_1 can also go to zero without loss of generality. As such, we have created a search space in which every single stationary point is a solution to our problem. In the case of MUBs this means a search space in which all stationary points are MUBs, with no stationary points existing in the cases in which MUBs do not exist.

Now the task is to find a stationary point, which we do using a Newton-style iterative approach. This is done by calculating the matrix of second-derivatives H as well as the vector of partial derivatives $\vec{g}$ of our function $f_m(\vec{x})$, then solving the system of linear equations $H\vec{p} = \vec{g}$ to get the update direction $\vec{p}$. We then follow this direction, scaled by some factor $0 < \alpha \leq 1$. The result is an algorithm which will eventually converge (given a small enough step size) if there are MUBs, whilst endlessly traversing the search space if MUBs do not exist. As such it does not constitute a formal proof of non-existence, simply serving as suggestion as to whether the given problem is feasible or not.

For the numerical computation of this method we use the library Eigen [88] for the linear system solving required for the Newton step. We tested

a number of solving methods but found that a partial-pivot LU decomposition works the best in our case, offering an effective blend of speed and accuracy, as well as offering a parallelized implementation. Whilst we did experiment with a line-search for choosing the parameter α, this seemed to require more time than it saved. We also tested a quasi-Newton update method in which we did not explicitly calculate the full Hessian, but the loss of accuracy resulted in very slow convergence. We also add a small (10^{-10}) term to the diagonals of the Hessian to improve the numerical stability.

3.2.5 Method 2: Branch-and-bound SDP hierarchy

For our second method, we use another optimization technique: a hierarchy of semidefinite programming relaxations similar to the Lasserre [89] or NPA [1] hierarchies for commuting and non-commuting polynomial optimization, respectively. The idea behind such techniques is to relax the problem and then add constraints of increasing complexity (referred to as levels in some hierarchy) to tighten the relaxation, often with guarantees of finite convergence. We begin by taking our non-linear equations and creating new variables to linearize them, turning each term $x_i x_j$ into a new variable x_{ij}, for example:

$$x_1 x_2 + x_2^2 + x_1 \to x_{12} + x_{22} + x_1 \tag{3.13}$$

We start by assuming that the new variables have the same bounds as the originals, resulting in a rather bad relaxation, for instance x_{ij} could be negative despite x_i and x_j being positive since for now there is no constraint linking them. It is, however, fast to compute, since we only have linear equations and thus it has become a convex optimisation problem. To then begin to constrain x_{ij} to be closer to $x_i x_j$ we add a hierarchy of positive-semidefinite (PSD) constraints, which are also convex and thus

the relaxation remains convex. The first level of such a hierarchy considers the moment matrix M_1 whose elements consist of products of all first-order monomials:

$$M_1 = \begin{pmatrix} 1 & x_1 & x_2 \\ x_1 & x_{11} & x_{12} \\ x_2 & x_{12} & x_{22} \end{pmatrix} \succcurlyeq 0 \tag{3.14}$$

One can show this is positive semidefinite by considering the product $c^T M_1 c$ where c is some arbitrary real vector of the same size as M_1. Expanding this product we get:

$$\begin{aligned} c^T M_1 c &= \begin{pmatrix} c_0 & c_1 & c_2 \end{pmatrix} \begin{pmatrix} 1 & x_1 & x_2 \\ x_1 & x_{11} & x_{12} \\ x_2 & x_{12} & x_{22} \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \end{pmatrix} \\ &= c_0^2 + 2c_0c_1x_1 + 2c_0c_2x_2 + c_1^2x_{11} + 2c_1c_2x_{12} + c_2^2x_{22} \\ &= (c_0 + c_1x_1 + c_2x_2)^2 \end{aligned} \tag{3.15}$$

As this will always be positive, for any vector c the product $c^T M_1 c$ is positive, thus M_1 is positive semidefinite. A similar proof can be done for the second level of the hierarchy, whose moment matrix look as follows, formed now with second order terms on the top row and then forming the corresponding products:

$$M_2 = \begin{pmatrix} 1 & x_1 & x_2 & x_{11} & x_{12} & x_{22} \\ x_1 & x_{11} & x_{12} & x_{111} & x_{112} & x_{122} \\ x_2 & x_{12} & x_{22} & x_{112} & x_{122} & x_{222} \\ x_{11} & x_{111} & x_{112} & x_{1111} & x_{1112} & x_{1122} \\ x_{12} & x_{112} & x_{122} & x_{1112} & x_{1122} & x_{1222} \\ x_{22} & x_{122} & x_{222} & x_{1122} & x_{1222} & x_{2222} \end{pmatrix} \succcurlyeq 0 \tag{3.16}$$

What we now have is an optimisation over a much larger set of variables with linear equality constraints corresponding to the MUB problem and linear inequality constraints to break symmetries, all to some chosen level of semidefinite hierarchy with the hope that at some level this convex relaxation of the problem will become infeasible in the cases of MUB non-existence. However, we find that even high levels (>5) of this hierarchy alone do not become infeasible, even for small cases. One work formatting this problem as a hierarchy of non-commuting variables suggest that one may need at least level 12 of their hierarchy [84]. If the level needed for the commuting case is similar (which is not at all guaranteed, as they are different formulations), it would require solving a semidefinite program of 110^{12} variables to solve the $d = 6$ $n = 4$ case, which is at least six orders of magnitude larger than what modern supercomputers can solve within a reasonable time. As such, we ask the question: what if we don't need to solve it in a single convex optimisation? What if instead, splitting it into a larger number of smaller convex regions would be sufficient?

Consider now the first level of the first-order relaxation of a single variable. Since it's a 2×2 matrix, there are only 2 eigenvalues. For both to be positive, the product of eigenvalues needs to be positive and thus the determinant needs to be positive:

$$\begin{pmatrix} 1 & x_1 \\ x_1 & x_{11} \end{pmatrix} \succcurlyeq 0 \qquad \implies \qquad x_{11} - x_1^2 \geq 0 \tag{3.17}$$

Assuming we also use box constraints of $-1 < x_1 < 1$, which we can assume in the MUB case since all of the magnitudes are bounded, the search space looks as given in Figure 3.2.

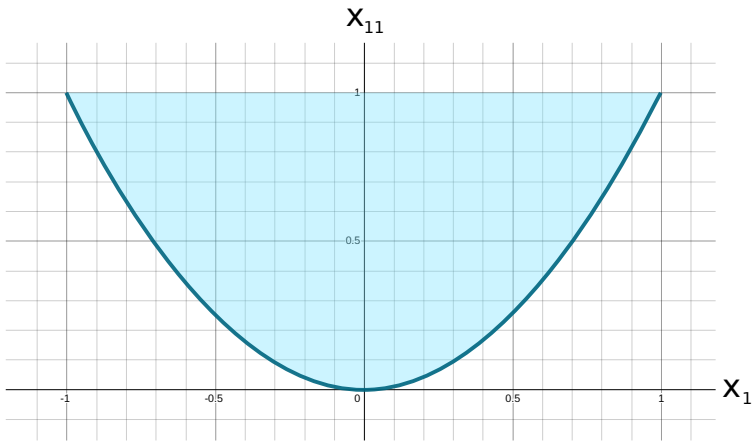

FIGURE 3.2: The search space created from the constraint $x_{11} - x_1^2 \geq 0$ as well as the box constraint $-1 \leq x_1 \leq 1$.

As one can see, this constraint still leaves many points such as $(0, 1)$ well within the search space whilst being very far from the $x_{11} = x_1^2$ curve onto which we want to constrain. This shows the issue with trying to convexify an equality constraint. Our approach to solve this is to stay in the first level of the hierarchy whilst splitting our convex region into two. Whilst for other types of constraint splitting a region into two would half the size of the region, in our case we can constrain further since we know the edge we are trying to converge towards, an idea shown more easily by a visual example as in Figure 3.3.

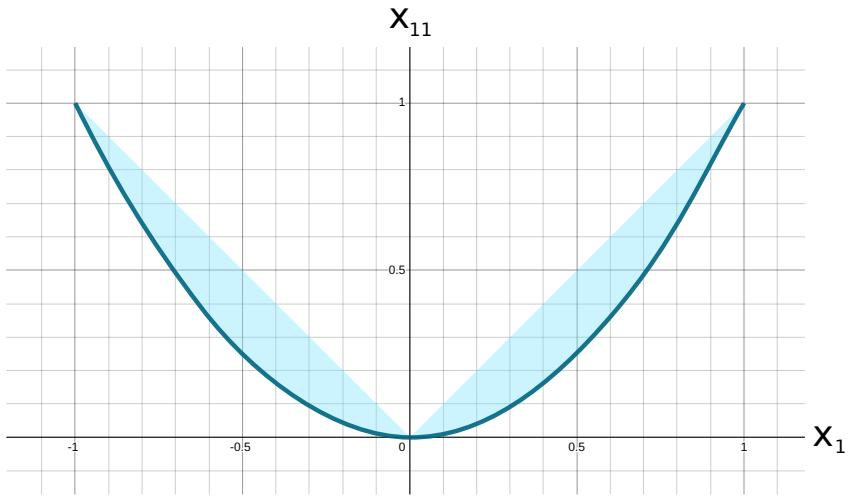

FIGURE 3.3: The two convex search spaces created from the constraint $x_{11} - x_1^2 > 0$ with the box constraints $-1 \leq x_1 \leq 0$ and $0 \leq x_1 \leq 1$.

By first performing the first-level relaxation without any splitting we obtain some point in the search space. For each first-order variable we define an error, $\text{err}(x_i) = (x_{ii} - x_i^2)^2$, giving us a heuristic for which variables are the least constrained. We found that a good method is to choose the variable with the highest error, splitting it in two (as in Figure 3.3) at the feasible point that was found by the relaxation. This is also referred to as "branching" at that variable. As such we have divided our whole search space into two, whilst also constraining tighter towards the set of valid first and second-order monomials. One can also choose to add an objective function to ensure that we find the "most-feasible" point (rather than just simply finding any point satisfying the constraints, trying to find the point deepest inside the feasible region), which we do by adding a new variable λ and asking that the moment matrix plus λ times the

identity is positive-semidefinite, then minimizing λ. In this way we find the moment matrix which needs the least "help" in order to be positive semidefinite, potentially even a matrix which allows the addition some negative λ whilst still being positive semidefinite.

By repeating this process one can see that we will eventually reach regions which are sufficiently small that the maximum error for any variable becomes less than some ϵ. In the MUB non-existence cases, under the assumption that there exists some ϵ such that one cannot find MUBs even allowing such error in the relations (e.g. that the various inner products are now constrained within $\frac{1}{\sqrt{d}} \pm \epsilon$), such small regions will result in convex infeasibility, which can be efficiently certified. By checking that a large number of these convex regions are all infeasible one can show that the original equation set is infeasible, thus proving the MUB nonexistence case.

Whilst in the above case we consider the first-level relaxation (i.e. the moment matrix given by Equation 3.15), moving to the second level (i.e. given by Equation 3.16) must be at least as constraining as the first, since the second-level PSD matrix contains the first as a submatrix. The procedure works the same as we move to higher-level relaxations, except now we have higher-order terms in the first row of our PSD matrix, making it much larger. For example, given 18 variables (as in the case of our reduction of the 5 MUBs in dimension 3 case), the first-level relaxation has a PSD matrix of 19×19, the second a matrix of 343×343, the third 6175×6175 and so on. Given that SDPs in the size of 300×300 are already difficult for desktop computers, doing a large number of 6000×6000 SDPs is near intractable. The hope, however, would be that by doing a tighter relaxation one does not need to perform so many branches to prove full infeasibility, giving a trade-off between the number of iterations needed and the level of the relaxation.

For the numerical computation of this method we have tested two SDP solvers: MOSEK [57] and SCS [90]. For the small cases and the first-level SDPs MOSEK works well, but has an incredibly high memory profile due to its need to store various second-order derivatives, however this allows it high accuracy and fast (logarithmic) convergence versus other methods. The alternative is SCS, which as a first order method has a very minimal memory usage, often using 100x less memory than MOSEK, at the cost of significantly longer convergence times. For all second-level optimisation we use SCS, since otherwise memory quickly becomes an issue, with MOSEK using around 80 GB for proving dimension 3 non-existence whilst SCS uses only 100 MB. With both libraries we exploit their sparsity and model parameterization features.

3.2.6 Results

We first begin by showing the amount of MUBs that seem to be feasible and infeasible using our first method, shown in Table 3.3, given as the number of iterations it required before we either found a satisfying vector or we stopped the optimization. With this we show how easily the algorithm finds solutions in contrast with the amount of time spend on the presumed non-existence cases.

The algorithm in general seems to find MUSBs with set sizes with less variance (i.e. $\{4, 4, 3, 3\}$) faster than those that have higher variance (i.e. $\{6, 5, 2, 1\}$), presumably because there are more MUSBs to find due to an increased number of symmetries. For highly uneven set sizes like $\{9, 7, 1, 1, 1, 1, 1, 1, 1, 1\}$ in dimension 9, the algorithm struggles, sometimes taken several thousand iterations. For high dimensions (>15) there is some instability that develops from the large linear systems, to combat this we lower the step size α and in some cases add a larger diagonal term to the Hessian to improve the condition number and thus numerical stability of the solver.

n / d	2	3	4	5	6	10
2	1	1	55	200	84	175
3	59	55	61	63	73	226
4	100000	113	75	104	100000	100000
5	100000	100000	87	96	100000	100000
6	100000	100000	100000	235	100000	100000
7	100000	100000	100000	100000	100000	100000

TABLE 3.3: The number of iterations required to either find the MUSB or until reaching the chosen iteration limit. These results, if we assume reaching the limit of 100000 iterations implies infeasibility, agree with known results and Zauner's conjecture. Here the criteria for convergence was reaching a value below 10^{-13} in the combined equation, roughly corresponding to a maximum error of 10^{-7} in any of the original equality constraints.

Now applying our second algorithm, we correctly convergence to the expected results in all cases in dimension 2 and 3. Whilst it was expected that we would not find MUSBs of sizes {2,1,1,1} in dimension 2, as far as the authors are aware it has not been shown that one cannot find MUSBs of sizes {2,1,1,1,1} in dimension 3. We show in Table 3.4 the number of SDPs that were required when branching with the first level of the hierarchy, whilst Table 3.5 shows the number using the second level. As expected, significantly less branches were needed as higher levels of the hierarchy were used, however using the third level or higher is intractable for anything higher than dimension 2.

n / d	2	3	4	5	6
2	0	0	11	43	?
3	2	40	81	19279	?
4	**11**	337	18004	?	**?**
5	-	**17153**	?	?	-
6	-	-	**?**	?	-
7	-	-	-	**?**	-

TABLE 3.4: Number of SDPs required to reach either convergence to an MUSB or to prove infeasibility when using the branch-and-bound algorithm with the first level of the hierarchy. Convergence to an MUSB is defined as an SDP being feasible with a largest monomial error of at most 10^{-8}. Bold text indicates the relevant problems for proving infeasibility. Here the 17153 iterations for dimension 3 takes around 3-4 minutes on a standard desktop.

n / d	2	3	4	5	6
2	0	0	13	13	?
3	1	18	35	?	?
4	**1**	22	?	?	**?**
5	-	**7**	?	?	-
6	-	-	**?**	?	-
7	-	-	-	**?**	-

TABLE 3.5: Number of SDPs required to reach either convergence to MUSB or proving infeasibility when using the branch-and-bound algorithm with the second level of the hierarchy. Bold text indicates the relevant problems for proving infeasibility. Here the 7 iterations for dimension 3 takes around a minute on a standard desktop with SCS.

As this algorithm runs, it regularly declares regions as infeasible, each of which has an associated area. Since in our problem the total search space is bounded (each variable between $\pm\frac{1}{\sqrt{d}}$) we can obtain the fraction of the search space that has been declared infeasible. If the entire problem is infeasible then this total value increases until reaching 100%, otherwise a solution is found before then. By considering the time taken to reach the current percentage of infeasibility and linearly extrapolating to 100%, the algorithm progressively outputs an estimate for the time it will take to prove infeasibility. For the $d = 3$ and $n = 5$ case (18 variables) this correctly predicts 5 minutes after only a few iterations. When running for $d = 4$ and $n = 6$ (50 variables) it estimates several months. For $d = 6$ and $n = 4$ (118 variables), the open problem, it estimates something on the order of hundreds of millions of years. Even assuming perfect parallelisation and running the algorithm on the worlds best supercomputer (at the time of writing the American "Frontier" [63]) it would still take hundreds of years to solve this open problem with this algorithm. As such, further reductions are needed, otherwise we may simply have to wait until classical computational power reaches a level such that this becomes tractable.

Considering the results from both methods, we notice there appears to exist some sort of pattern in the minimum sizes that are infeasible, for instance we find that for dimension 6 $\{6, 3, 3, 3\}$ does not seem to be feasible, whilst $\{6, 3, 3, 2\}$ is feasible and the algorithm finds a solution in very few iterations. Although our first thoughts were that perhaps it relates to the total number of vectors, it cannot be so simple due to the lack of freedom of the final vector in an orthonormal set, for instance that in dimension 3, $\{2, 2, 2, 2, 2\}$ and $\{3, 3, 3, 3, 3\}$ will both be equally infeasible despite having very different total vector counts. We list a selection of feasible and infeasible sets for dimensions 3, 4, 5 and 6 in Tables 3.6, 3.7, 3.8 and 3.9 respectively. Comparing with previous work on MUB "constellations" [79], our work agrees with the result of all known set sizes, apart from that of $\{6, 6, 3, 1\}$, which they declare as feasible within

their tolerance of 10^{-7}, whilst in our case we reached only 10^{-9}, thus not meeting our required tolerance of 10^{-13}, and thus our conclusion is that is indeed very close to being feasible, but does not appear to be able to reach a true zero (within numerical precision).

One possible conjecture is that the MUSB problem is infeasible if the number of independent constraints is at least the number of variables minus one. Whilst there are known theorems for the number of solutions of linear systems that are overdetermined and underdetermined [91], the results do not extend trivially to non-linear systems. Whilst there are some MUSBs such as $\{6, 6, 1, 1\}$ which have more constraints (95) than variables (92) despite being feasible, one can perform an analysis of the equations to show that some appear to imply others. We have found a set of 10 constraints for $\{6, 6, 1, 1\}$ that even when removed, all solutions we found satisfying this reduced equation set also satisfy the removed constraints, suggesting that the other 85 equations were sufficiently constraining. By extension this would also mean these constraints are redundant for $\{6, 6, 3, 1\}$, thus taking it down to 139 equations vs 136 variables, leaving it still infeasible under this conjecture and agreeing with the optimisation results.

set sizes	vecs	vars	eqns	result
{3,3,3,3}	12	74	101	proven feasible
{1,1,1,1,1}	5	18	15	proven feasible
{2,1,1,1,1}	6	18	18	**proven infeasible**
{3,1,1,1,1}	7	18	18	**proven infeasible**

TABLE 3.6: A list of some set sizes for dimension 3 as well as the conclusion made using our optimisation techniques. Here "proven feasible" means a valid MUSB was found to a tolerance of 10^{-13} using method 1 whilst "proven infeasible" means that we have branched the entire search space to infeasibility using method 2.

set sizes	vecs	vars	eqns	result
{4,4,4,4,4}	258	20	354	proven feasible
{1,1,1,1,1,1}	6	36	26	proven feasible
{2,1,1,1,1,1}	7	36	30	proven feasible
{2,2,1,1,1,1}	8	50	45	proven feasible
{2,2,2,1,1,1}	9	64	62	proven feasible
{2,2,2,2,1,1}	10	80	82	**seems infeasible**
{4,1,1,1,1,1}	9	36	34	proven feasible
{4,2,1,1,1,1}	10	50	50	**seems infeasible**

TABLE 3.7: A list of some set sizes for dimension 4 as well as the conclusion made using our optimisation techniques. Here "proven feasible" implies a valid MUSB was found to a tolerance of 10^{-13} using method 1, whilst "seems infeasible" implies our search for a feasible point using method 1 was inconclusive even after 100000 iterations.

set sizes	vecs	vars	eqns	result
{5,5,5,5,5,5}	30	652	902	proven feasible
{1,1,1,1,1,1,1}	7	60	40	proven feasible
{2,2,2,2,2,1,1}	12	138	132	proven feasible
{2,2,2,2,2,2,1}	13	162	161	**seems infeasible**
{3,2,2,1,1,1,1}	11	96	90	proven feasible
{3,3,2,1,1,1,1}	12	116	114	proven feasible
{3,3,3,1,1,1,1}	13	136	140	**seems infeasible**
{5,1,1,1,1,1,1}	11	60	55	proven feasible
{5,2,1,1,1,1,1}	12	78	75	proven feasible
{5,3,1,1,1,1,1}	13	96	97	**seems infeasible**

TABLE 3.8: A list of some set sizes for dimension 5 as well as the conclusion made using our optimisation techniques. Here "proven feasible" implies a valid MUSB was found to a tolerance of 10^{-13} using method 1, whilst "seems infeasible" implies our search for a feasible point using method 1 was inconclusive even after 100000 iterations.

set sizes	vecs	vars	eqns	result
{6,6,6}	18	170	206	proven feasible
{1,1,1,1}	4	22	7	proven feasible
{2,1,1,1}	5	22	9	proven feasible
{2,2,1,1}	6	36	18	proven feasible
{2,2,2,1}	7	50	29	proven feasible
{2,2,2,2}	8	66	43	proven feasible
{3,2,2,1}	8	50	33	proven feasible
{3,2,2,2}	9	66	48	proven feasible
{3,3,3,2}	11	102	86	proven feasible
{3,3,3,3}	12	122	109	proven feasible
{4,2,1,1}	8	36	24	proven feasible
{4,3,3,3}	13	122	117	proven feasible
{4,4,3,3}	14	144	144	**seems infeasible**
{6,1,1,1}	9	22	15	proven feasible
{6,6,1,1}	14	92	95	proven feasible
{6,6,2,1}	15	114	121	proven feasible
{6,6,3,1}	16	136	149	**seems infeasible**
{6,3,3,2}	14	102	100	proven feasible
{6,3,3,3}	15	122	125	**seems infeasible**
{6,3,2,2,1}	14	106	102	proven feasible
{6,3,2,2,2}	15	128	128	**seems infeasible**
{6,2,2,1,1,1,1,1}	15	134	131	proven feasible
{6,2,2,2,1,1,1,1}	16	158	160	**seems infeasible**

TABLE 3.9: A list of some set sizes for dimension 6 as well as the conclusion made using our optimisation techniques. Here "proven feasible" implies a valid MUSB was found to a tolerance of 10^{-13} using method 1, whilst "seems infeasible" implies our search for a feasible point using method 1 was inconclusive even after 100000 iterations.

The algorithms used here are general and can be applied to any commuting polynomial optimisation problem using our library, written as a single C++ header file. The library represents polynomials using hash tables, allowing for high performance even with large equations. We release this code as an open-source project [92], although it still needs work regarding user-friendliness and documentation, for now only serving as a reference for the methods used as well as containing the data files for each optimisation referenced.

3.2.7 Conclusion

We explored the polynomial representation of the open mutually unbiased bases existence problem, showing a number of reductions in order to take full advantage of the available symmetries. The result is then a second-order real non-linear set of equations. We then show two methods of approaching such a problem, a Lagrange-multiplier Newtonian-descent algorithm which we use for quickly checking which of our relaxations are feasible, as well as a global branch-and-bound algorithm. We explore the notation of mutually unbiased sub-bases and show that in some cases proving infeasibility can be done by proving infeasibility of sub-bases, reducing the number of variables drastically.

These reductions regarding the MUB problem may simplify future proofs and generate new questions about why sub-bases cannot be found in certain dimensions and why the minimum sub-basis size seems to follow a particular pattern. Unfortunately, solving the open question in this framework by searching the whole space still remains intractable, with our estimates suggesting it is still beyond the capacity of even the most advanced supercomputers at the time of writing.

3.3 As a Visual Game

Here we discuss a different formulation of the MUB problem, one that allows a human to visualize (and even find by hand) higher-dimensional MUBs, something that is normally somewhat tricky due to our inability to picture higher-dimensional spaces. Whilst this method has been implemented as an offline C++ code that offers better performance, for the purposes of this thesis an online version written in JavaScript has been created to offer the reader an **interactive MUB experience**.

Beginning with the general equation set for the MUB problem, we then choose to fix the first basis to that of the computational basis. This allows us to write each other basis as follows, in angular form as with Equation 3.6, here for the case of searching for 3 MUBs in dimension 2:

$$\begin{aligned} \text{if basis 1} = &\quad (1 \;\; 0) \quad \text{and} \quad (0 \;\; 1) \\ \implies \text{basis 2} = &\quad \frac{1}{\sqrt{2}}(e^{i\theta_1} \;\; e^{i\theta_2}) \quad \text{and} \quad \frac{1}{\sqrt{2}}(e^{i\theta_3} \;\; e^{i\theta_4}) \\ \implies \text{basis 3} = &\quad \frac{1}{\sqrt{2}}(e^{i\theta_5} \;\; e^{i\theta_6}) \quad \text{and} \quad \frac{1}{\sqrt{2}}(e^{i\theta_7} \;\; e^{i\theta_8}) \end{aligned} \tag{3.18}$$

One can see that the inner product constraint between basis 1 and basis 2 or 3 is automatically satisfied: each term in basis 2 and 3 is a complex number with magnitude $1/\sqrt{2}$, thus the squared inner product with any vector in basis 1 will be $1/2$. Normalization of each vector is also automatically satisfied. Thus, we are only left with two constraints: that the inner products within each basis should equal 0 (orthogonality) and that the squared inner products between basis 2 and 3 should have a magnitude of $1/2$. But, we have an extra trick we can play because of the exponential formational, if we now consider the inner product between the first vectors of basis 2 and 3, remembering to take the conjugate of the latter,

we get the constraint:

$$\left|\frac{1}{2}\left(e^{i(\theta_1-\theta_5)}+e^{i(\theta_2-\theta_6)}\right)\right|^2=\frac{1}{2} \tag{3.19}$$

We can then do a change of variables, take the square root and multiply by 2, to get the following:

$$\begin{aligned}\left|e^{i\phi_1}+e^{i\phi_2}\right|&=\sqrt{2}\\ \phi_1&=\theta_1-\theta_5\\ \phi_2&=\theta_2-\theta_6\end{aligned} \tag{3.20}$$

Doing a similar process for the orthogonality condition between both vectors of basis 2:

$$\begin{aligned}\left|e^{i\phi_3}+e^{i\phi_4}\right|&=0\\ \phi_3&=\theta_1-\theta_3\\ \phi_4&=\theta_2-\theta_4\end{aligned} \tag{3.21}$$

Whilst this might not seem like a simplification, it allows us to visualize the problem in a 2D space. Now, no matter the dimension, for each inner product we want to constrain that a series of vector additions has a certain magnitude, with linear constraints on the angles. Each vector sum can be seen as though trying to drag a "chain" consisting of a series of links, since each vector has fixed magnitude and thus only the angles can change. If we also use some of the other constraints mentioned in Section 3.2, we can even further reduce the number of angles. Simplifications can also be performed on the set of linear constraints to find the minimal set

of variable (linearly independent) angles which each affect a set of non-variable (linearly dependant) angles.

The full C++ suite exploring the formulation and optimisation (using various standard algorithms) can be found here [93]. In particular, the optimization algorithm known as L-BFGS seems find bases very quickly where they exist, given a few randomized starts. Due to the lack of overall novelty, this was never written as a publication, however the author feels that it is interesting and serves to show how easy finding bases can be: even humans can do it. To allow the reader of the thesis to explore this without having to compile anything, a JavaScript version is provided that can be played around with in any browser (mobile or otherwise), with a screenshot also available as Figure 3.4:

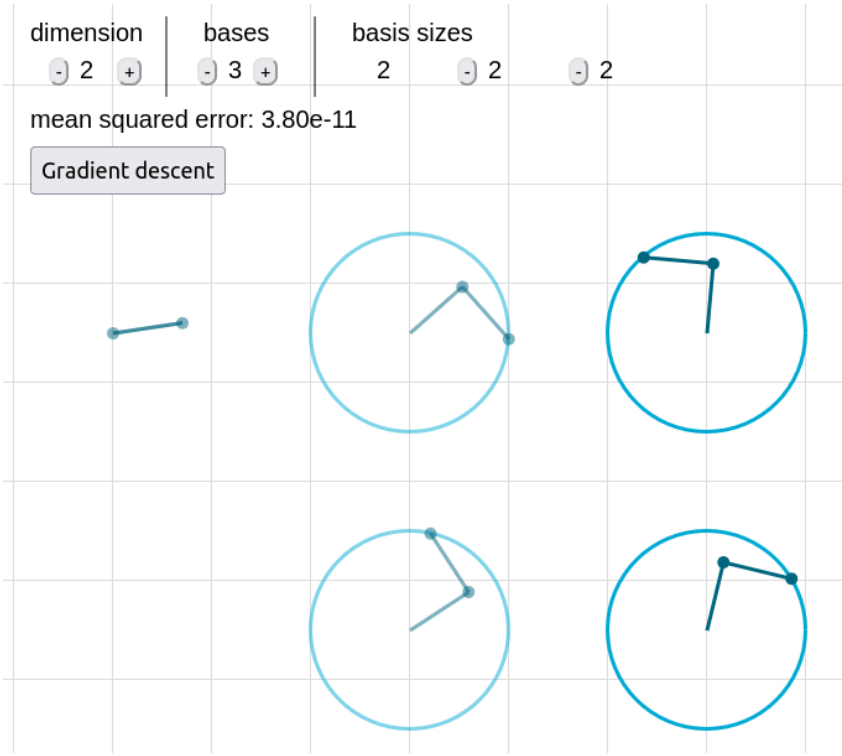

FIGURE 3.4: A screenshot of the JavaScript visual MUB game. The user can drag the darker blue vector chains around, with the goal of trying to reach the target radius for each chain. The lighter chains on the left are affected by the darker chains on the right, and cannot be manually dragged. The target radius of some of the chains is zero, hence why they wrap back on themselves. Shown is the ideal configuration (with an error of 3.8×10^{-11}) for 3 MUBs in dimension 2.

3.4 As a Non-Commuting Optimisation Problem

There have also been a number of attempts at converting the MUB problem into that of a non-commuting optimisation problem. In the view of the author, the "closest" anyone has gotten to solving the MUB problem so far has been the work of Gribling and Polak [94], who used the C^* formulation of the problem by Navascues, Pironio and Acin [95]. The core idea is that one can construct some C^* algebra if and only if a certain number of MUBs exist in a given dimension. The resulting problem can then be transformed into an SDP (one may begin to notice a theme in the thesis) and heavily reduced using similar symmetries as those mentioned in Section 3.2. A core difference is that now the symmetries are applied using the wreath product to actually reduce the number of variables, rather than simply restricting the search space. As such, they were able to reach level 5.5 in the hierarchy, able to disprove the existence of $d+2$ MUBs in dimension d for all $d \leq 8$, a fact that was already known but something that has proven difficult to do with numerical methods (e.g. in Section 3.2 it was difficult to even disprove 5 in dimension 3). But, despite the computationally efficient method, they were unable to solve the open problem.

Some time in the PhD was spent working with Gribling and Polak, along with Sebastian Designolle, to attempt to push the method further. The original Julia code has since been fully parallelised and heavily optimized and we have spent some time running much higher levels on a large cluster computer, with the larger SDPs requiring months to generate and further months to finish running. Yet, despite our efforts, the SDP always remained feasible, thus still failing to solve the open problem.

3.5 As a Bell Inequality

We now move on to consider the MUB problem in the context of Bell inequalities. We will show how the problem can be formulated as a Bell inequality, and how a number of optimisation techniques can be applied to prove the existence and non-existence of MUBs in certain cases. The work in this section is adapted from a publication [96] which has the author of the thesis as the joint first author. Said publication was split into three sections, where two of the sections were the contributions of the author, and thus these two sections are included here whilst the third has been removed.

3.5.1 Introduction

Given an orthonormal basis $\{|b_j\rangle\}_{j=1}^d$ in d-dimensional complex space ($\mathbb{C}^d$), one can define the corresponding POVM $\{B_j = |b_j\rangle\langle b_j|\}_{j=1}^d$, consisting of rank-1 projections onto the basis elements. We say that two measurements are MUBs if they correspond to a pair of orthonormal bases that are MUBs.

We look at MUBs in the context of Bell scenarios (see Ref. [97] for a review). Bell scenarios describe physical experiments performed by two distant parties, usually referred to as Alice and Bob. These parties share many copies of a bipartite quantum state, and perform local measurements on these copies. The experiment is described by the set of correlations $p(a, b|x, y)$, specifying the probability of Alice (Bob) observing locally the outcome a (b) upon choosing the measurement setting x (y). In quantum theory, the shared state is described by a density operator $\rho \geq 0$ with unit trace acting on a tensor product Hilbert space $\mathcal{H}_A \otimes \mathcal{H}_B$. The measurements are described by local POVMs $\{A_a^x\}$ and $\{B_b^y\}$ on the Hilbert spaces $\mathcal{H}_A$ and $\mathcal{H}_B$, respectively. For a fixed state and measurements, the

correlation is given by the Born rule,

$$p(a,b|x,y) = \mathrm{tr}[\rho(A_a^x \otimes B_b^y)] \tag{3.22}$$

Note that for a pure state, $\rho = |\psi\rangle\langle\psi|$, with $|\psi\rangle \in \mathcal{H}_A \otimes \mathcal{H}_B$ and $\langle\psi|\psi\rangle = 1$, the Born rule reduces to

$$p(a,b|x,y) = \langle\psi| A_a^x \otimes B_b^y |\psi\rangle \tag{3.23}$$

Bell functionals are linear functionals of correlations, i.e. functionals of the form

$$W(p) = \sum_{a,b,x,y} c_{abxy}\ p(a,b|x,y) \tag{3.24}$$

where c_{abxy} are real coefficients. Non-trivial *Bell inequalities* are Bell functionals for which $W(p) \leq \beta_L$ holds for every correlation of the form

$$p(a,b|x,y) = \int_\Lambda \mathrm{d}\mu(\lambda)\ p_A(a|x,\lambda)\ p_B(b|y,\lambda) \tag{3.25}$$

(also called a *local* correlation, considered as the notion of classicality in Bell scenarios), but for which there exists a quantum correlation p of the form as in Equation 3.22 such that $W(p) > \beta_L$. In Equation 3.25, Λ is a measurable set with a probability measure μ, $\lambda \in \Lambda$, and p_A and p_B are conditional probability distributions.

While the original interest in Bell inequalities was precisely this separation of local and quantum correlations, we will be interested in their quantum

maximum (or *maximal quantum violation*), i.e. the tight upper bound $W(p) \leq \beta_Q$ satisfied by all correlations of the form as in Equation 3.22.

3.5.2 Bell Inequalities for Mutually Unbiased Bases

In this work, we are interested in a family of Bell inequalities that was introduced in Ref. [69] and is parametrised by an integer $d \geq 2$. For a fixed d, Alice has d^2 measurement settings labelled as $x = x_1x_2$ with $x_1, x_2 \in \{1, \ldots, d\}$. Each of these measurements has three outcomes, $a \in \{1, 2, \perp\}$. Bob, on the other hand, has two measurement settings, $y \in \{1, 2\}$, with d outcomes each, $b \in [d]$. The Bell inequality then reads

$$\begin{aligned} W_d(p) = &\sum_{x_1,x_2,y} \left[p(y, x_y|x_1x_2, y) - p(\bar{y}, x_y|x_1x_2, y)\right] \\ &-\frac{1}{2}\sqrt{\frac{d-1}{d}} \sum_{x_1,x_2} \left[p_A(1|x_1x_2) + p_A(2|x_1x_2)\right] \end{aligned} \tag{3.26}$$

where $\bar{y}$ flips the value of $y \in \{1, 2\}$, and $p_A(a|x) = \sum_b p(a, b|x, y)$ is the marginal probability distribution of Alice (which is independent of y).

This is a non-trivial Bell inequality with maximal quantum violation $\beta_Q = \sqrt{d(d-1)}$ [69]. The maximal violation can be achieved with the maximally entangled d-dimensional state $|\phi_d^+\rangle \equiv \frac{1}{\sqrt{d}} \sum_{j=1}^{d} |j\rangle \otimes |j\rangle$, and *any* pair of MUB measurements on Bob's side. Moreover, if the dimension is fixed to be d, this is the only way in which the maximal quantum violation can be achieved, up to local unitary freedom [69]. This property of the Bell inequality as in Equation 3.26 forms the core of our numerical approaches to construct MUBs.

The above Bell inequality can be straightforwardly extended to a set of n measurements on Bob's side. The Bell inequality for n measurements is

a sum of Bell inequalities of the form as in Equation 3.26. For each pair $y, z \in [n]$ such that $y < z$ (denoted in the following as $(y, z) \in \text{Pairs}[n]$), we introduce d^2 settings for Alice, labelled as $x = (y, z)x_y x_z$ with $x_y, x_z \in [d]$, and take a copy of the Bell inequality in Equation 3.26, defined as

$$\begin{aligned} W_d^{(y,z)}(p) = \sum_{x_y,x_z,w} & \big[p(a_w, x_w|(y,z)x_y x_z, w) \\ & - p(\bar{a}_w, x_w|(y,z)x_y x_z, w)\big] \\ & - \frac{1}{2}\sqrt{\frac{d-1}{d}} \sum_{x_y,x_z} \big[p_A(1|(y,z)x_y x_z) \\ & + p_A(2|(y,z)x_y x_z)\big] \end{aligned} \tag{3.27}$$

where $w \in \{y, z\}$, $a_y = 1$, $a_z = 2$, and $\bar{a}_w$ flips the value of a_w. The final Bell inequality then reads

$$W_{d,n}(p) = \sum_{(y,z)\in\text{Pairs}[n]} W_d^{(y,z)}(p) \tag{3.28}$$

It is clear that $W_{d,n}(p) \leq \binom{n}{2}\sqrt{d(d-1)}$, by applying the known bound to each individual term in the above sum. Moreover, if the dimension is d, the only way to reach this maximum (up to local unitary freedom) is by using the maximally entangled state and the n measurements on Bob's side corresponding to MUBs. Hence, we can reformulate the MUB problem in terms of these Bell inequalities:

Proposition 1 *$W_{d,n}(p) = \binom{n}{2}\sqrt{d(d-1)}$ can be achieved in dimension d if and only if n MUBs exist in dimension d.*

3.5.3 The Optimisation Problem

According to the above proposition, finding n MUBs in dimension d can be cast as an optimisation problem, maximising $W_{d,n}(p)$ over d-dimensional quantum states and measurements. To see how to do this explicitly, let us first write out the Bell inequality W_d in Equation 3.26 in terms of a quantum state $|\psi\rangle$ and measurements $\{A_a^x\}$, $\{B_b^y\}$, using Born's rule in Equation 3.23:

$$\begin{aligned} W_d(|\psi\rangle, \{A_a^x\}, \{B_b^y\}) = & \sum_{x_1,x_2,y} \big(\langle\psi| A_y^{x_1x_2} \otimes B_{x_y}^y |\psi\rangle \\ & - \langle\psi| A_{\bar{y}}^{x_1x_2} \otimes B_{x_y}^y |\psi\rangle \big) \\ & - \frac{1}{2}\sqrt{\frac{d-1}{d}} \sum_{x_1,x_2} \big(\langle\psi| A_1^{x_1x_2} \otimes \mathbb{1} |\psi\rangle \\ & + \langle\psi| A_2^{x_1x_2} \otimes \mathbb{1} |\psi\rangle \big). \\ = & \sum_{j,k} \langle\psi| \big[(A_1^{jk} - A_2^{jk}) \otimes (B_j^1 - B_k^2) \\ & - \frac{1}{2}\sqrt{\frac{d-1}{d}} (A_1^{jk} + A_2^{jk}) \otimes \mathbb{1} \big] |\psi\rangle \end{aligned} \tag{3.29}$$

where in the second equality we wrote out the summation over y and switched to the notation $x_1x_2 \to jk$ with $j, k \in [d]$. Maximising the inequality given in Equation 3.29 in terms of the state and the measurements

can then be written as the optimisation problem

$$\begin{aligned} \max_{|\psi\rangle,\{A_a^x\},\{B_b^y\}} \quad & W_d(|\psi\rangle, \{A_a^x\}, \{B_b^y\}) \\ \text{s.t.} \quad & |\psi\rangle \in \mathbb{C}^d \otimes \mathbb{C}^d, \quad \langle\psi|\psi\rangle = 1 \\ & A_a^x, B_b^y \in \mathcal{L}_{\text{sa}}(\mathbb{C}^d) \quad \forall a, b, x, y \\ & A_a^x \geq 0 \quad \forall a, x, \quad B_b^y \geq 0 \quad \forall b, y \\ & \sum_a A_a^x = \mathbb{1} \quad \forall x, \quad \sum_b B_b^y = \mathbb{1} \quad \forall y \end{aligned} \tag{3.30}$$

where $\mathcal{L}_{\text{sa}}(\mathbb{C}^d)$ is the set of self-adjoint linear operators on $\mathbb{C}^d$. Optimising $W_{d,n}$ can be written in a similar fashion, with:

$$\begin{aligned} & W_{d,n}(|\psi\rangle, \{A_a^x\}, \{B_b^y\}) \\ & = \sum_{(y,z)\in \text{Pairs}[n]} \Bigg\{ \sum_{j,k} \langle\psi| \left[(A_1^{(y,z)jk} - A_2^{(y,z)jk}) \otimes (B_j^y - B_k^z) \right. \\ & \qquad \left. - \frac{1}{2}\sqrt{\frac{d-1}{d}} (A_1^{(y,z)jk} + A_2^{(y,z)jk}) \otimes \mathbb{1} \right] |\psi\rangle \Bigg\} \end{aligned} \tag{3.31}$$

The optimisation problem is then

$$\begin{aligned} \max_{|\psi\rangle,\{A_a^x\},\{B_b^y\}} \quad & W_{d,n}(|\psi\rangle, \{A_a^x\}, \{B_b^y\}) \\ \text{s.t.} \quad & |\psi\rangle \in \mathbb{C}^d \otimes \mathbb{C}^d, \quad \langle\psi|\psi\rangle = 1 \\ & A_a^x, B_b^y \in \mathcal{L}_{\text{sa}}(\mathbb{C}^d) \quad \forall a, b, x, y \\ & A_a^x \geq 0 \quad \forall a, x, \quad B_b^y \geq 0 \quad \forall b, y \\ & \sum_a A_a^x = \mathbb{1} \quad \forall x, \quad \sum_b B_b^y = \mathbb{1} \quad \forall y \end{aligned} \tag{3.32}$$

In this work, we consider various approaches to solve the optimisation problem given by Equation 3.32. In particular, from Proposition 1 it follows that n MUBs exist in dimension d if and only if the solution of the above optimisation problem is $\binom{n}{2}\sqrt{d(d-1)}$. We will facilitate the problem using knowledge about the optimal realisation of the Bell inequality from Ref. [69].

First of all, we notice that the value $\binom{n}{2}\sqrt{d(d-1)}$ can only be achieved in dimension d with the maximally entangled state [69]. Without loss of generality, we therefore impose that $|\psi\rangle = |\phi_d^+\rangle = \frac{1}{\sqrt{d}}\sum_{j=1}^{d}|j\rangle \otimes |j\rangle$. We can then use the fact that for any two operators A and B on $\mathbb{C}^d$ we have that $\langle\phi_d^+| A \otimes B |\phi_d^+\rangle = \frac{1}{d}\mathrm{tr}(A^T B)$, where $(.)^T$ is the transposition in the basis $\{|j\rangle\}$. As a second simplification, we notice that in order to saturate the bound $\binom{n}{2}\sqrt{d(d-1)}$ in dimension d, Alice's measurement operators A_1^x and A_2^x, and all of Bob's measurement operators must be trace-1 [69]. For such operators we have that $\langle\phi_d^+| A_1^x \otimes \mathbb{1} |\phi_d^+\rangle = \langle\phi_d^+| A_2^x \otimes \mathbb{1} |\phi_d^+\rangle = \frac{1}{d}$. The second term in Equation 3.31 is then a constant, $-\binom{n}{2}\sqrt{d(d-1)}$, and does not influence the optimisation problem. The simplified Bell expression finally reads

$$\begin{aligned} &W_{d,n}^{+}\left(\{A_a^x\},\{B_b^y\}\right) \\ &= \sum_{(y,z)\in \mathrm{Pairs}[n]} \left\{ \frac{1}{d}\sum_{j,k} \mathrm{tr}\big((A_1^{(y,z)jk} - A_2^{(y,z)jk})^T (B_j^y - B_k^z)\big) \right\} \end{aligned} \tag{3.33}$$

and its maximum quantum value $W_{d,n}^{+}$ satisfies:

$$W_{d,n}^{+} \leq W_{\mathrm{MUB}}(d,n) \equiv n(n-1)\sqrt{d(d-1)} \tag{3.34}$$

The simplified optimisation problem becomes

$$\begin{aligned} \max_{\{A_a^x\},\{B_b^y\}} \quad & W_{d,n}^{+}(\{A_a^x\},\{B_b^y\}) \\ \text{s.t.} \quad & A_a^x, B_b^y \in \mathcal{L}_{\text{sa}}(\mathbb{C}^d) \;\; \forall a,b,x,y \\ & A_a^x \geq 0 \;\; \forall a,x, \quad B_b^y \geq 0 \;\; \forall b,y \\ & \text{tr} A_a^x = 1 \;\; \forall a,x, \quad \text{tr} B_b^y = 1 \;\; \forall y,b \\ & \sum_a A_a^x = \mathbb{1} \;\; \forall x, \quad \sum_b B_b^y = \mathbb{1} \;\; \forall y \end{aligned} \tag{3.35}$$

The optimal value of this optimisation problem is $n(n-1)\sqrt{d(d-1)}$ if and only if n MUBs exist in dimension d.

We may further simplify the optimisation problem. From Ref. [69] we know that in the optimal realisation the B_b^y operators are rank-1 projections, $B_j^y = |b_j^y\rangle\langle b_j^y|$ (they are projections onto the basis elements of MUBs). For such operators, we have that

$$(B_j^y - B_k^z)^3 = [1 - \text{tr}(B_j^y B_k^z)](B_j^y - B_k^z) \tag{3.36}$$

which implies that the spectrum of $B_j^y - B_k^z$ is contained in $\{0, \pm\lambda_{jk}^{yz}\}$, where $\lambda_{jk}^{yz} \equiv \sqrt{1 - \text{tr}(B_j^y B_k^z)} = \sqrt{1 - |\,\langle b_j^y | b_k^z\rangle\,|^2}$. Moreover, we have that $\text{tr}[(B_j^y - B_k^z)^2] = 2(\lambda_{jk}^{yz})^2$ and $\text{tr}(B_j^y - B_k^z) = 0$, and therefore $B_j^y - B_k^z$ has one eigenvalue λ_{jk}^{yz}, one eigenvalue $-\lambda_{jk}^{yz}$ and the rest of the eigenvalues are 0. Furthermore, in the optimal realisation we have that $(A_1^{(y,z)jk})^T$ is the rank-1 projection onto the eigenspace of $B_j^y - B_k^z$ corresponding to λ_{jk}^{yz}, and $(A_2^{(y,z)jk})^T$ is the rank-1 projection onto the eigenspace of $B_j^y - B_k^z$ corresponding to $-\lambda_{jk}^{yz}$. With these final simplifications the Bell

expression reads

$$W_{d,n}^{+B}(\{B_b^y\}) = \frac{2}{d} \sum_{(y,z)\in \mathrm{Pairs}[n]} \sum_{j,k} \sqrt{1 - \mathrm{tr}(B_j^y B_k^z)} \tag{3.37}$$

$$= \frac{2}{d} \sum_{(y,z)\in \mathrm{Pairs}[n]} \sum_{j,k} \sqrt{1 - |\langle b_j^y | b_k^z \rangle|^2} \tag{3.38}$$

and the corresponding optimisation problem is

$$\begin{aligned} \max_{\{B_b^y\}} \quad & W_{d,n}^{+B}(\{B_b^y\}) \\ \text{s.t.} \quad & B_b^y \in \mathcal{L}_{\mathrm{sa}}(\mathbb{C}^d) \quad \forall x, y \\ & (B_b^y)^2 = B_b^y \quad \forall b, y \\ & \mathrm{tr} B_b^y = 1 \quad \forall b, y \\ & \sum_b B_b^y = \mathbb{1} \quad \forall y \end{aligned} \tag{3.39}$$

Note that projectivity and self-adjointness implies positive semidefiniteness, and therefore positive semidefiniteness does not need to be imposed.

The optimal value of the optimisation problem given by Equation 3.39, denoted as $W_{\max}(d, n)$, is $n(n-1)\sqrt{d(d-1)}$ if and only if n MUBs exist in dimension d. In fact, Alice's measurements have been completely removed from the problem, and one may regard Equation 3.39 as a purely geometrical problem: find n orthonormal bases, $\big\{\{|b_j^y\rangle \mid j \in [d]\} \mid y \in [n]\big\}$, maximising the "MUB-ness measure" of Equation 3.38.

In the following sections we apply different numerical methods to solve the optimisation problems given by Equations 3.35 and 3.39 in order to numerically tackle Zauner's conjecture.

3.5.4 Method - See-Saw SDP

One arrives at a relatively simple method of optimising Problem 3.35 by noticing that the objective function is bi-linear in the A_a^x and B_b^y matrices. That is, if every A_a^x is fixed, then the problem simplifies to optimising a linear functional of the B_b^y matrices with a series of linear and positive semidefinite constraints.

The *see-saw* optimisation technique starts with fixing the set of A_a^x matrices satisfying the constraints of Problem 3.35, either with random values or based on some prior knowledge. We then solve the problem for the B_b^y matrices, which is a standard SDP. Then, we fix the B_b^y matrices to the optimum found, and solve the resulting SDP for the A_a^x matrices, and so on. By repeating this process, the system eventually converges to a stable result, i.e. the value of the objective function does not change beyond a given threshold within a certain window of iterations (a change of less than 10^{-9} for 10 iterations in our implementation). Although the see-saw method is not guaranteed to converge to the global optimum for a non-convex problem, in this case it never failed to converge within the chosen precision if given sufficient time, and always to the value expected (i.e. to $W_{\text{MUB}}(d,n)$ whenever it is known that n MUBs exist in dimension d). We implemented the see-saw algorithm using MOSEK [57], with the code now publicly available [98].

3.5.5 Results - See-Saw SDP

The values obtained with the see-saw algorithm are shown in Table 3.10 (the values displayed are $1-W_{d,n}/W_{\text{MUB}}(d,n)$ for easier comparison across different n and d, where $W_{d,n}$ is the result of the optimisation). Notice that whenever n MUBs exist in dimension d, the see-saw method correctly converges to the MUB solution, and whenever it is known that n MUBs do not exist in dimension d, the method does indeed converge to a value less than $W_{\text{MUB}}(d,n)$. For the unknown case of four MUBs in dimension

six, the see-saw method could not find four MUBs, supporting Zauner's conjecture. Furthermore, the measurements found by the see-saw method are numerically very close to the "four most distant bases" of Ref. [99]. Note, however, that the results simply mean that the method could not find four MUBs in dimension six, but one cannot rule out the possibility that they exist. Given that these four most distant bases are already quite close to being mutually unbiased (on the order of 10^{-5}), it demonstrates the difficulty of the problem.

The method also has consistent convergence, for example, for $d = 2$ and $n = 2$, all of 10000 see-saw optimisations from random starts converged to the correct value of $W_{\text{MUB}}(2, 2) = 2.82843$ up to five decimal places, albeit finding a different set of optimum matrices. Similarly, for 10000 optimisations for $d = 2$ and $n = 4$, all optimisations converged to the same value of 16.72616, correctly suggesting non-existence.

All of the results for this method were obtained on a desktop PC with 8GB of RAM using 4 cores, with times varying between milliseconds for the smallest problem ($d = 2$, $n = 2$) and hours for the largest one ($d = 6$, $n = 4$). Since SDP solvers are generally quite efficiently parallelised, this method offers good parallel scaling, however, the memory requirement is quite high, since it requires explicit storage and optimisation of the A_a^x matrices.

3.5.6 Method - Non-Linear SDP

An alternative approach to the optimisation is to focus on Problem 3.39, which features only the B_b^y matrices and thus contains fewer variables for a reduction in the size of the search space as well as the memory required. The downside, however, is that the objective function is now non-linear (not even bi-linear) and thus many efficient solvers (i.e. for standard SDP systems) can no longer be applied. To optimise this problem we

n \ d	2	3	4	5	6
2	0.00000	0.00000	0.00000	0.00000	0.00000
3	0.00000	0.00000	0.00000	0.00000	0.00000
4	**0.01440**	0.00000	0.00000	0.00000	**0.00004**
5	-	**0.00391**	0.00000	0.00000	-
6	-	-	**0.00186**	0.00000	-
7	-	-	-	**0.00091**	-

TABLE 3.10: The values $1 - W_{d,n}/W_{\mathrm{MUB}}(d,n)$, where $W_{d,n}$ is the result of the optimisation, obtained with the see-saw method after convergence for various dimensions (d) and numbers of bases (n), to 5 decimal places. The values depicted are consistently obtained by multiple runs. The values in bold indicate that the value was at least 10^{-5} away from zero, meaning that the method could not find n MUBs in dimension d. A "-" indicates that we did not perform the optimisation. Note that in all the known cases, the algorithm predicts correctly existence/nonexistence, and it predicts that four MUBs do not exist in dimension six.

adapt a method based on the work by Yamashita et al. for optimising a non-linear SDP using a primal-dual interior point method [100]. This method, assuming a few basic conditions (discussed later), is guaranteed to converge to a Karush-Kuhn-Tucker (KKT) point, a point satisfying a series of constraints known as the KKT conditions, which are necessary for optimality [101]. These KKT conditions are only sufficient (i.e. imply a global minimum) in a subset of cases, the most common being that the problem is convex, which is unfortunately not true in our case, and as such we have no guarantee that the method will converge to the global optimum.

In order to implement the method of Yamashita et al., we parametrise the measurement operators $\{B_y^x\}$ by a real vector $\mathbf{x} = (x_i)_i$. To be able to deal with real numbers instead of complex ones, we note that every self-adjoint matrix $B = B_r + \mathrm{i}B_i$ (where B_r is real symmetric and B_i is real anti-symmetric) can be mapped to the real symmetric matrix $\hat{B}$ via

$$B \mapsto \hat{B} = \begin{bmatrix} B_r & B_i \\ -B_i & B_r \end{bmatrix} \tag{3.40}$$

It is easy to verify that $B \geq 0$ if and only if $\hat{B} \geq 0$, and $\mathrm{tr}B = \frac{1}{2}\mathrm{tr}\hat{B}$.

We therefore define - in line with the method of Yamashita et al. - a matrix $X(\mathbf{x}) = \sum_i C_i x_i + D$, which is a block diagonal matrix containing the $\hat{B}_b^y$ matrices on its diagonal in such a way that the linear constraints $\sum_b B_b^y = \mathbb{1}$ and $\mathrm{tr}B_b^y = 1$ of the optimisation Problem 3.39 are already enforced. The real parameters x_i correspond to those elements of the $\hat{B}_b^y$ matrices that are free after enforcing the linear constraints. While the constraint $B_b^y \geq 0$ is superfluous for the Problem 3.39, we chose to include this in our optimisation problem, as this constraint is heavily used in the method of Yamashita et al. With the parametrisation above, this constraint is equivalent to $X(\mathbf{x}) \geq 0$. The last remaining constraint is projectivity, $(B_b^y)^2 = B_b^y$ for all y and b, which is equivalent to $X^2(\mathbf{x}) = X(\mathbf{x})$. We enforce this constraint through $g(\mathbf{x}) \equiv \|X^2(\mathbf{x}) - X(\mathbf{x})\|_F^2 = 0$, where $\|.\|_F$ is the Frobenius norm. Further, we denote the objective function in terms of $\mathbf{x}$ by $W(\mathbf{x})$, suppressing the d, n indices whenever it does not lead to confusion.

The method requires introducing Lagrange multipliers (dual variables) for every constraint. In our case, there is a single inequality constraint $X(\mathbf{x}) \geq 0$, to which we assign the dual variable Z, which is a matrix with the same dimensions as $X(\mathbf{x})$. Furthermore, we have a single equality

constraint, $g(\mathbf{x}) = 0$, to which we assign the dual variable y, which is a scalar. The resulting Lagrangian reads

$$L(\mathbf{x}, y, Z) = W(\mathbf{x}) - yg(\mathbf{x}) - \mathrm{tr}[Z^T X(\mathbf{x})] \tag{3.41}$$

The algorithm for solving the optimisation problem is iterative, and each iteration begins with the calculation of G, the Hessian of the Lagrangian. In our case this can be quite expensive so we opt to use the alternative update method also proposed in Ref. [100] based on the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm, which approximates the Hessian without requiring the full calculation of the second derivatives. This G is then used to form a series of linear equations which we solve using a stabilised bi-conjugate gradient method in order to obtain the update directions for the primal ($\mathbf{x}$) and dual (y, Z) variables. Following this, a simple line search is performed to find the optimum step size, then the variables are updated. This process is then repeated until the following barrier KKT conditions are met for some barrier parameter μ:

$$r(\mathbf{x}, y, Z, \mu) \equiv \begin{pmatrix} \nabla L(\mathbf{x}, y, Z) \\ g(\mathbf{x}) \\ X(\mathbf{x})Z - \mu\mathbb{1} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{3.42}$$

Through repeated iterations of this method with values of μ converging towards zero, this algorithm has been proven to always converge to a KKT point of the system, assuming certain conditions. The first of these is that the functions $W(\mathbf{x})$ and $g(\mathbf{x})$ are both twice continuously differentiable, which in our case is true: $W(\mathbf{x})$ is simply a sum of square roots of polynomials of $\mathbf{x}$, whilst $g(\mathbf{x})$ is a vector of polynomials of $\mathbf{x}$. The second condition is that the vector $\mathbf{x}$ must remain within a finite set during the optimisation, which for us is true since infinite values are non-optimal for the MUB functional. The third condition is that the matrices C_i must

be linearly independent, which for us is true since they serve to place a single component of $\mathbf{x}$ to one (diagonal) or two (off-diagonal) positions in the matrix $X(\mathbf{x})$.

Similarly to the see-saw method, there exists some non-determinism to this method due to it starting from a randomised interior point. This is found by first beginning with a random vector, then performing gradient descent using the derivatives of the constraints until the vector satisfies all constraints within some precision. This vector is then used as the starting point for the iterative method. Although perhaps one could start with a "good guess" for the vector and optimise from there, we decided to begin from a random point to attempt to cover a wider search space. The code for this method is also publicly available [102].

3.5.7 Results - Non-Linear SDP

The values obtained through this algorithm are shown in Table 3.11. The optimal values, as well as the optimal bases found, agree with the result of the see-saw method up to high numerical precision. In particular, we find the same set of four bases to numerically optimise the six dimensional case.

Regarding performance, a large amount of time can be saved if certain parameters, such as the initial step size, are chosen correctly. Notably, there appears to be no universally optimal set of parameters, and many of the results are obtained through the manual tweaking of the parameters for the specific system. All of the results for this method were obtained on a standard desktop PC using 4 cores, with the times varying between milliseconds for the smallest problem and an hour for the largest one. This method has a significantly reduced memory cost compared to the see-saw method, since here the A_a^x matrices are not used, as well as in general offering faster optimisation.

n \ d	2	3	4	5	6
2	0.00000	0.00000	0.00000	0.00000	0.00000
3	0.00000	0.00000	0.00000	0.00000	0.00000
4	**0.01440**	0.00000	0.00000	0.00000	**0.00004**
5	-	**0.00391**	0.00000	0.00000	-
6	-	-	**0.00161**	0.00000	-
7	-	-	-	**0.00091**	-

TABLE 3.11: The values $1 - W_{d,n}/W_{\mathrm{MUB}}(d,n)$, where $W_{d,n}$ is the result of the optimisation, obtained at an approximate KKT point with $\|r(\mathbf{x}, y, Z, \mu)\| \equiv \sqrt{\|\nabla L(\mathbf{x}, y, Z)\|^2 + |g(\mathbf{x})|^2 + \|X(\mathbf{x})Z - \mu\mathbb{1}\|_F^2} \leq 10^{-5}$ for various dimensions (d) and numbers of bases (n), to 5 decimal places. The values depicted are consistently obtained by multiple runs. The values in bold indicate that the value was at least 10^{-5} away from zero, meaning that the method could not find n MUBs in dimension d. Note that in all the known cases, the algorithm predicts correctly existence/nonexistence, and it predicts that four MUBs do not exist in dimension six.

3.5.8 Conclusions

We reformulated the existence problem of MUBs as an optimisation problem, using a recently found family of Bell inequalities. We then applied several numerical methods suitable for optimising Bell inequalities in order to tackle the existence problem: see-saw SDP optimisation and non-linear SDP techniques. The results of all numerical optimisations is in full accordance with the known cases in dimensions $d = 2, 3, 4, 5$, where we find $d+1$ MUBs. Furthermore, whenever it is known that n MUBs do not exist in a given dimension d, all the different algorithms converge to the same set

of bases in all cases (with a slight difference between the see-saw method and the other method for $d = 4$, $n = 6$), and these bases are not MUBs.

We applied our numerical techniques to the open case of four MUBs in dimension six. Both algorithms suggest that there do not exist four MUBs in dimension six, by converging to a Bell value strictly smaller than the hypothetical MUB value. Moreover, the bases found by both algorithms are very close numerically with the "four most distant bases" in dimension six of Ref. [99]. Hence, our findings provide further numerical evidence for Zauner's conjecture.

Chapter 4

The NPA Hierarchy

Having briefly touched on nonlocality in the previous section, we now turn to a more general study of it. Specifically, we consider nonlocality problems formulated as Bell inequalities, which we then attempt to find the maximum quantum violation of using a hierarchy of semidefinite programs known as the Navascués-Pironio-Acín (NPA) hierarchy. We first begin with some background on nonlocality and Bell inequalities, followed by specific projects and their results.

4.1 Background

We begin with a refresher on the idea of nonlocality. The core idea is that some measurement statistics of certain quantum systems can be shown to be impossible to recreate under the assumption of **local realism**, generally described by a **hidden variable model** - the idea that the statistics can be reproduced simply by some secret knowledge shared by all the parties [103]. To better illustrate this, we consider a "game" (in the game-theory sense, not the fun sense).

Say we have two parties, Alice and Bob, who are separated by a large distance. They each receive an input value, x and y respectively, and must each output a value, a and b respectively. These values can be anything, but normally we assume the inputs/outputs are integers from some finite list. From this we can define a win condition for the game, such as "the game is won if Alice's output (a) is equal to Bob's input (y)", from which we would define the win probability as the probability that a is equal to y:

$$P_{\text{win}} = P(a = y) \tag{4.1}$$

Alice and Bob are allowed to use whatever strategy they desire, they could choose to always output a certain value, output random values, anything, as long as they don't communicate as soon as the game begins. We now turn our attention to a specific problem of this form, known as the Clauser-Horne-Shimony-Holt (CHSH) inequality [104]. Here we have the aforementioned scenario but now with each party having two inputs and two outputs, which for convenience we label the inputs as 0 and 1 and the outputs as -1 and 1. We also assume they that parties share some kind of object before the game starts, which we call ρ. For now, we make no assumptions about ρ, it could be some text, a number, an algorithm or a quantum object, anything. It is also common to represent such scenarios diagrammatically, as in Figure 4.1.

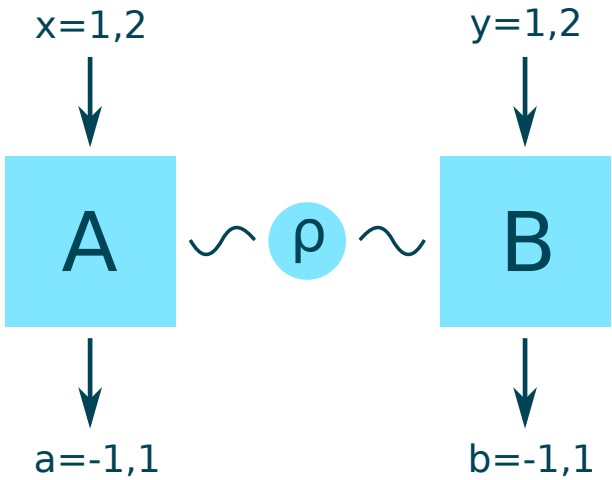

FIGURE 4.1: A diagram of the CHSH scenario. Alice and Bob each receive an input, x and y respectively, and output a value, a and b respectively. The generic object ρ is shared between them.

We then choose to change notation to expectation values rather than probabilities. To do this, we consider each input x leading to Alice performing some operation A_x, similarly for Bob (B_y). We then consider the expectation values of these operations, which represents the output value scaled by the probability of it occurring:

$$\begin{aligned}
\langle A_1 \rangle &= P(a=1|x=1) - P(a=-1|x=1) \\
\langle A_2 \rangle &= P(a=1|x=2) - P(a=-1|x=2) \\
\langle B_1 \rangle &= P(b=1|y=1) - P(b=-1|y=1) \\
\langle B_2 \rangle &= P(b=1|y=2) - P(b=-1|y=2)
\end{aligned} \tag{4.2}$$

We can also define expectation values for the combined operations, such as $\langle A_1 B_2 \rangle$, which represents the product of the expectation value of Alice's output when she has input 1 ($x = 1$) and Bob's output when he has input 2 ($y = 2$). This can be written as:

$$\begin{aligned} \langle A_1 B_2 \rangle = & P(a = 1, b = 1 | x = 1, y = 2) \\ & + P(a = -1, b = -1 | x = 1, y = 2) \\ & - P(a = 1, b = -1 | x = 1, y = 2) \\ & - P(a = -1, b = 1 | x = 1, y = 2) \end{aligned} \tag{4.3}$$

Given all of these expectation values, the function we want to maximize is the following quantity:

$$S = \langle A_1 B_1 \rangle + \langle A_1 B_2 \rangle + \langle A_2 B_1 \rangle - \langle A_2 B_2 \rangle \tag{4.4}$$

In the classical case, where Alice and Bob share only a classical object, the best they can possibly do is a value of 2. For example, imagine that Alice always chooses to output 1 and Bob also always outputs 1, no matter their input. Thus, since we're always expecting to get 1, we'd have:

$$\begin{aligned} \langle A_1 B_1 \rangle = 1 \qquad & \langle A_1 B_2 \rangle = 1 \\ \langle A_2 B_1 \rangle = 1 \qquad & \langle A_2 B_2 \rangle = 1 \end{aligned} \tag{4.5}$$

and thus get a value of $S = 2$. This is known to be the best one can do using only a classical strategy, thus the CHSH inequality is so called because it is normally written as an inequality for the classical case:

$$\langle A_1 B_1 \rangle + \langle A_1 B_2 \rangle + \langle A_2 B_1 \rangle - \langle A_2 B_2 \rangle \leq 2 \tag{4.6}$$

If we now consider the quantum case, we can show that we can reach up to a value of $2\sqrt{2}$, known as the Tsirelson bound [105]. This is done by considering the following quantum strategy: Alice and Bob share a maximally entangled state $\psi = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ and then perform measurements on their half of the state. Specifically, Alice measures in the Z or X basis, corresponding to $A_1 = Z$ and $A_2 = X$, whilst Bob measures in a combination of the two bases, corresponding to $B_1 = \frac{1}{\sqrt{2}}(X + Z)$ and $B_2 = \frac{1}{\sqrt{2}}(X - Z)$. The value of the inequality is therefore (since we're dealing with a pure state):

$$\begin{aligned} S &= \langle\psi|A_1B_1|\psi\rangle + \langle\psi|A_1B_2|\psi\rangle + \langle\psi|A_2B_1|\psi\rangle - \langle\psi|A_2B_2|\psi\rangle \\ &= 2\sqrt{2} \end{aligned} \tag{4.7}$$

Thus we have shown that the quantum strategy can reach a higher value than the classical one, and thus the CHSH inequality is violated. This is the core idea of nonlocality, that quantum systems can violate certain inequalities that classical systems cannot. If one were to see measurement statistics in a real experiment that violate such an inequality, it would demonstrate that we do not live in a local universe. Such violations have indeed been observed, notably by Aspect in 1982 [10], winning him and colleagues the Nobel prize in 2022 [11].

Whilst for this simple example we have an analytical solution, for larger Bell inequalities this is often not the case. As such, to determine the maximum possible quantum violation we move to using numerical methods and thus we return to the world of optimisation. Said maximum violation for a two-party system can be determined from the following optimisation

problem:

$$\begin{aligned} \max_{A_i,B_i,\rho} \quad & \sum_i c_{A_i} \langle A_i \rangle + \sum_i c_{B_i} \langle B_i \rangle + \sum_{i,j} c_{A_i B_j} \langle A_i B_j \rangle \\ \text{subject to} \quad & \langle A_i \rangle = \mathrm{tr}(A_i\ \rho) \qquad \forall i \\ & \langle B_i \rangle = \mathrm{tr}(B_i\ \rho) \qquad \forall i \\ & \langle A_i B_j \rangle = \mathrm{tr}(A_i B_j\ \rho) \qquad \forall i,j \\ & \rho \succcurlyeq 0 \\ & \mathrm{tr}(\rho) = 1 \end{aligned} \tag{4.8}$$

where here we want to maximize some linear function of the expectation values of the operators (defined by fixed coefficients c_{A_i}, c_{B_i} and $c_{A_i B_j}$), subject to the constraint that these expectation values should be valid quantum expectation values: obeying the Born rule coming from a certain ρ which is a valid density matrix. This is a difficult optimisation, not least because it is non-linear, can be non-convex, and if we want to explicitly optimise over the matrices we need to pick a dimension for them, which could be very large if a high-dimensional system is needed to reach the maximum. As such, actually optimising directly over the matrices A_i, B_i and ρ, each of which scales in size with the chosen dimension, is often intractable for all but the most trivial cases of low dimension and only a small number of operators/inputs.

But, assuming we are consider a small case, one way of getting a lower bound on the maximum violation (i.e. a valid set of quantum measurements which give some violation, but perhaps not the maximum) is to use a technique called **see-sawing**. This involves fixing some variables so the problem because linear in the objective, thus transforming it into an SDP which can be more efficiently solved. This was mentioned in Section

3.5.4, used in that section on the bi-linear Bell inequality. The more general problem is effectively tri-linear and thus to apply the seesaw method one needs to fix all B_i and ρ whilst optimising A_i, then fix A_i and ρ whilst optimising B_i, then fix A_i and B_i whilst optimising ρ and so on. At the end of each optimisation all constraints should be satisfied, with the hope that with successive iterations we move closer towards the optimum whilst staying within the feasible region, in a sort of interior-point gradient-descent method.

But now say that we want to approach this from the other direction. Rather than trying to find a lower bound for the max violation, what if we want to find an upper bound? To do so, we can no longer consider a state of a fixed or even finite dimension, since this would restrict us to a subspace of the set of possible quantum correlations. What we would rather do in this case is to consider a superset of the quantum correlations, a relaxation of the problem such that the optimum of the relaxation is an upper bound for our original problem, something that one can imagine visually from Figure 2.3.

Thus we arrive at the **NPA hierarchy** a technique first demonstrated by Navascués, Pironio and Acín in 2008 [1], based on similar relaxation of commuting optimisation problems such as that by Lasserre in [89]. The basic idea is to consider a **moment matrix**: in our case a matrix such that each entry is defined by $\Gamma_{ij} = \langle\psi|O_i^\dagger O_j|\psi\rangle$ where ψ is some quantum state and O_i are the operators of our system (in case of Bell scenarios, the measurement choices of Alice and Bob). If we then consider how this

matrix would multiply with some arbitrary complex vector v:

$$\begin{aligned} v^\dagger \Gamma v &= \sum_{ij} v_i^* \Gamma_{ij} v_j \\ &= \sum_{ij} v_i^* \langle\psi| O_i^\dagger O_j |\psi\rangle v_j \\ &= \langle\psi| \left(\sum_i v_i O_i\right)^\dagger \left(\sum_j v_j O_j\right) |\psi\rangle \\ &= \langle\psi| V^\dagger V |\psi\rangle \end{aligned} \tag{4.9}$$

The final object here should be positive, since it is the expectation value of a positive operator. Thus we can see that the moment matrix should be non-negative for any vector v and thus be positive semi-definite, thus allowing us to turn the problem into an SDP (yay!). Interestingly, the resulting SDP has no explicit mention of the state, only constraining that some moment matrix (of the expectation values) be positive, and that we should maximize some linear combination of said expectation values. As such, the SDP is state independent and finds the maximum over all quantum states, whether that be in dimension 2 or dimension 20.

For now, we consider the list of operators that we use for the moment matrix as the identity along with the various operators for Alice and Bob: e.g. $O = \{\mathbb{1}, A_1, A_2, B_1, B_2, ...\}$. As a simple example, we show here the SDP that finds the maximum violation of the CHSH inequality (Tsirelson's

bound $= 2\sqrt{2}$):

$$\begin{aligned} \max \quad & \langle A_1B_1\rangle + \langle A_1B_2\rangle + \langle A_2B_1\rangle - \langle A_2B_2\rangle \\ \text{subject to} \quad & \begin{pmatrix} 1 & \langle A_1\rangle & \langle A_2\rangle & \langle B_1\rangle & \langle B_2\rangle \\ \langle A_1\rangle & 1 & \langle A_1A_2\rangle & \langle A_1B_1\rangle & \langle A_1B_2\rangle \\ \langle A_2\rangle & \langle A_2A_1\rangle & 1 & \langle A_2B_1\rangle & \langle A_2B_2\rangle \\ \langle B_1\rangle & \langle B_1A_1\rangle & \langle B_1A_2\rangle & 1 & \langle B_1B_2\rangle \\ \langle B_2\rangle & \langle B_2A_1\rangle & \langle B_2A_2\rangle & \langle B_2B_1\rangle & 1 \end{pmatrix} \geq 0 \end{aligned} \tag{4.10}$$

Note that here we actually have some non-physical expectation values with $\langle A_1A_2\rangle$ and $\langle B_1B_2\rangle$ which cannot correspond to anything, however these values (annoyingly) serve a function as will be demonstrated in the next section. Just a reminder that all of the expectation values here are simply real numbers. The above SDP can be optimized quite easy with the following Python code, using the convex optimisation library cvxpy:

```
1  # Import the necessary library
2  import cvxpy as cp
3
4  # Moment matrix
5  M = cp.Variable((5,5), symmetric=True)
6
7  # Constrain the moment matrix
8  cons = []
9  cons.append(M >> 0)
10 cons.append(M[0,0] == 1)
11 cons.append(M[1,1] == 1)
12 cons.append(M[2,2] == 1)
13 cons.append(M[3,3] == 1)
14 cons.append(M[4,4] == 1)
15
```

```
16 # Define the objective
17 obj = M[3,1] + M[3,2] + M[4,1] - M[4,2]
18
19 # Solve the problem
20 prob = cp.Problem(cp.Maximize(obj), cons)
21 prob.solve()
22 print(prob.value)
23 # prints 2.8284271247459927 (approx 2sqrt(2))
```

Whilst this gives the known optimum for a simple 2 input 2 output case like the CHSH inequality, for more complicated inequalities it would more likely be a loose relaxation and thus would give a value higher than the true maximum violation, thus an upper bound. To attempt to converge closer to the true value we can expand the SDP to the second level of what we now define as a hierarchy of SDPs - a series of larger and larger SDPs with each giving a tighter and tighter upper bound.

In the case of the NPA hierarchy, the way we define our successive levels is to use more and more products of operators in the list of operators defining the moment matrix. For level L of the hierarchy we use products of operators up to order L. As an example, for a 2-input 2-output inequality like the CHSH the different levels are given by the following operator lists O_L, each of which is the previous set with extra higher-level operators added:

$$\begin{aligned} O_1 &= \{\mathbb{1}, A_1,\ A_2,\ B_1,\ B_2\} \\ O_2 &= O_1 \cup \{A_1A_2,\ A_1B_1,\ ...\} \\ O_3 &= O_2 \cup \{A_1A_2B_1,\ A_1A_2B_2,\ ...\} \end{aligned} \tag{4.11}$$

Note that in general there are an infinite number of possible operators we could add (e.g. $A_1A_2A_1A_2...$). Also note that these are just the operators creating the moment matrix, the actual matrix itself will have expectation

values of up to order $2L$ since it contains the product of everything in the operator list. When dealing with higher order operators, we can use a number of **replacement rules** to constraint moments that should be equal in an attempt to reduce the number of variables in our SDP. For instance, $\langle A_1 B_1 A_1 \rangle = \langle B_1 \rangle$ using the fact that A_1 and B_1 commute (since Alice and Bob cannot communicate), allowing us to swap those operators and therefore allowing $A_1 A_1$ to be reduced to $\mathbb{1}$. In the case of all operators commuting and squaring to the identity (as is the case in classical binary optimisation), there would no longer be an infinite number of possible operators and thus the size of the SDP would be finite.

4.2 Shadow of the Set

When performing optimisations of the type above, whereby you are maximizing some linear combination of a few terms, subject to the constraint that a matrix containing those terms be positive, there may be terms in such a matrix that appear "useless". They do not contribute to the objective in any way, simply serving as placeholders since our symmetric positive matrices have to be square. For high levels of the hierarchy these variables dominate the optimisation, we might have thousands of them and only a handful of variables that we actually care about the value of. Such values like $\langle A_1 A_2 B_1 \rangle$ also have no physical interpretation, since they are not valid measurements. Perhaps a natural question then is to ask, can we find a representation of the set that does not include extra variables? The test case for this will be simplest example possible: attempting to convert the CHSH moment-matrix positivity constrain mentioned in the previous section to some function f which contains only the variables in

the Bell inequality:

$$\begin{pmatrix} 1 & \langle A_1\rangle & \langle A_2\rangle & \langle B_1\rangle & \langle B_2\rangle \\ \langle A_1\rangle & 1 & \langle A_1A_2\rangle & \langle A_1B_1\rangle & \langle A_1B_2\rangle \\ \langle A_2\rangle & \langle A_2A_1\rangle & 1 & \langle A_2B_1\rangle & \langle A_2B_2\rangle \\ \langle B_1\rangle & \langle B_1A_1\rangle & \langle B_1A_2\rangle & 1 & \langle B_1B_2\rangle \\ \langle B_2\rangle & \langle B_2A_1\rangle & \langle B_2A_2\rangle & \langle B_2B_1\rangle & 1 \end{pmatrix} \geq 0 \tag{4.12}$$
$$\rightarrow \qquad f(\langle A_1B_1\rangle, \langle A_1B_2\rangle, \langle A_2B_1\rangle, \langle A_2B_2\rangle) \geq 0$$

We will refer to this as a "shadow" of the quantum set, since that is exactly what it appears to be, as shown by Figure 4.2. It is effectively the projection of every point in the set onto a plane, thus eliminating the variance of the unwanted variables. The resulting set should be convex, since convex objects always cast a convex shadow (as projections are linear functions and thus preserve convexity). There are, however, no guarantees about the complexity of representing the reduced set. It may be that the function f above has a very high degree and thus is just as computationally difficult to optimise over as the original set. Perhaps the advantage would only be in memory, since having a higher order polynomial of 4 variables may use less memory than a positive matrix of 10.

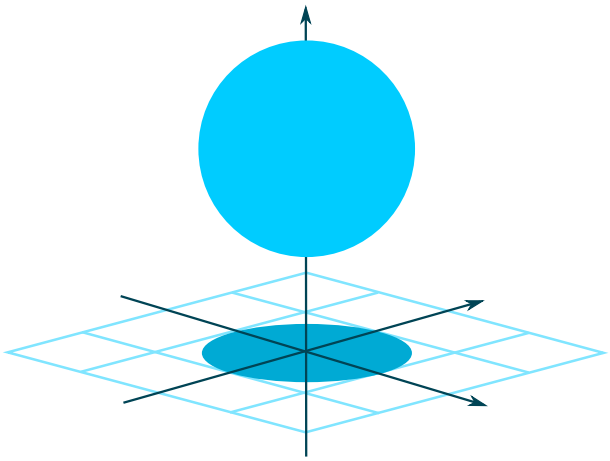

FIGURE 4.2: A diagram of the shadow of a set. The set is projected onto a plane, thus eliminating one of the unused variables. In this case, the original set is given by $x^2 + y^2 + (z-2)^2 \leq 1$ whilst the shadow is given by $x^2 + y^2 \leq 1$, thus the variable z has been eliminated. A link is also provided to an online version, allowing rotation and easier visualisation:

We now move specifically to our quantum set (we will refer to the quantum set specific to the first level of the NPA hierarchy solving the CHSH inequality as "the quantum set" or simply "the set" for brevity). One trivial reduction we can apply to our moment matrix is to use the fact that for a matrix to be positive, every principal submatrix should also be positive

semidefinite [106] and thus we can consider the bottom right 4×4 submatrix from our original 5×5 matrix, eliminating the single-order terms of $\langle A_1 \rangle$, $\langle A_2 \rangle$, $\langle B_1 \rangle$ and $\langle B_2 \rangle$. We are then left with trying to remove the two remaining terms that do not appear in our objective, $\langle A_1 A_2 \rangle$ and $\langle B_1 B_2 \rangle$, here also corresponding to the only non-physical terms.

In order to see how the remaining set looks, we can attempt to visualise it in 3D space by considering the case where one of our four main variables (say $\langle A_2 B_2 \rangle$) has some fixed value, then optimising to find 3D points (values for the other three main variables) that do not cause infeasibility in the resulting SDP. Doing this gives us a cloud of many points, to which we can then find a convex hull for simpler visualisation. Such a convex hull can then be compared to known sets.

In the case of $\langle A_2 B_2 \rangle = 1$, the points admitting a positive matrix form a convex hull resembling a simple three-variable semidefinite cone. Meanwhile, when $\langle A_2 B_2 \rangle = 0$, the resulting point cloud resembles a "sphube", a spherical cube, the 3D version of a squircle [107]. Examples of these shapes are shown in Figures 4.3 and 4.4.

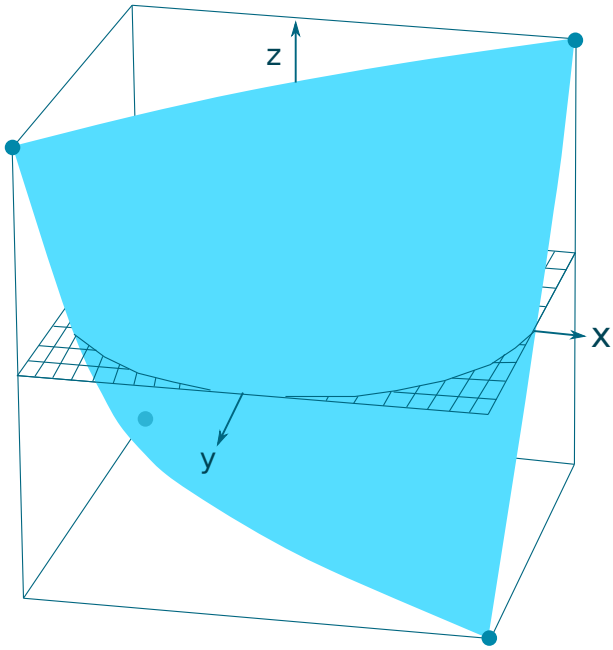

FIGURE 4.3: A visualisation of the convex hull of points that admit a positive moment matrix when $\langle A_2 B_2 \rangle = 1$. A link is also provided to an online version, allowing rotation and easier visualisation:

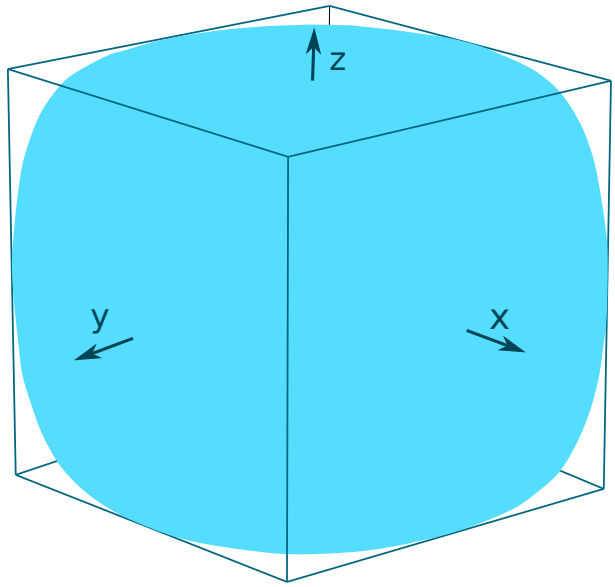

FIGURE 4.4: A visualisation of the convex hull of points that admit a positive moment matrix when $\langle A_2 B_2 \rangle = 0$. A link is also provided to an online version, allowing rotation and easier visualisation:

The question then remains, how do we get an analytical representation of this set? Whilst for this small case we can estimate the extent of the set by simply sampling, for larger cases this becomes quickly intractable. The aim is therefore to try to find an analytical form, with the hopes

that perhaps there might be some pattern that could be exploited when dealing with larger moment matrices which follow a similar structure. To begin to do so, we first transform the problem of positivity (an eigenvalues problem) into that of polynomial positivity, using the fact that for a matrix to be positive, so too do all of it's submatrix determinants. Exchanging the expectation values with new symbols for brevity (x, y, z, a for our main variables, b, c for the unwanted ones), our constraint becomes:

$$\begin{pmatrix} 1 & b & x & y \\ b & 1 & z & a \\ x & z & 1 & c \\ y & a & c & 1 \end{pmatrix} \geq 0 \tag{4.13}$$

Taking the determinant of the full matrix we get the polynomial positivity constraint:

$$\begin{aligned} f_1(x, y, z, a, b, c) &= a^2x^2 - a^2 - 2abcx + 2aby \\ &+ 2acz - 2axyz + b^2c^2 - b^2 - 2bcyz + 2bxz \\ &- c^2 + 2cxy - x^2 + y^2z^2 - y^2 - z^2 + 1 \geq 0 \end{aligned} \tag{4.14}$$

We can then take the determinant of the 3×3 upper-left submatrix to get another constraint:

$$f_2(x, z, b) = 2bxz - b^2 - x^2 - z^2 + 1 \geq 0 \tag{4.15}$$

And finally we can take the determinant of the 2×2 upper-left submatrix to get the constraint:

$$f_3(b) = 1 - b^2 \geq 0 \tag{4.16}$$

We now begin the process of trying to remove the b and c variables. Given that we are mostly focussed on the hull of our shape, we consider each individual equation being constrained to equal to zero rather than greater than zero. By setting an equation equal to zero, we can then use a symbolic solver like SymPy or Mathematica to solve for one of the variables we want to eliminate. We then substitute this into the remaining constraints and then repeat for the other variable. Doing so gives us a series of equations which include neither b nor c yet constrain at least one section of the original hull. We then need to repeat the process, as we do not know which equations should actually be zero, so we consider all branching choices, for all positive principal minors of the original matrix. In doing so, we eventually arrive at a set of equations which constrain to the edge of our original domain, for any value of a. First, the things that must always be true:

$$\begin{aligned} -1 \leq x \leq 1 \qquad & -1 \leq y \leq 1 \\ -1 \leq z \leq 1 \qquad & -1 \leq a \leq 1 \\ 1 + a^2y^2 - a^2 - y^2 \geq 0 & \end{aligned} \tag{4.17}$$

Then, at least one of the two following equations must be true (i.e. the full set is their union), which emerge due to the two $\pm$ solutions of solving

a quadratic equation:

$$a^2 - x^2 + y^2 - z^2 - 2a^2y^2 + 2ay\sqrt{1 - a^2 - y^2 + a^2y^2} \\ +2axyz - 2xz\sqrt{1 - a^2 - y^2 + a^2y^2} \geq 0$$

or (4.18)

$$a^2 - x^2 + y^2 - z^2 - 2a^2y^2 - 2ay\sqrt{1 - a^2 - y^2 + a^2y^2} \\ +2axyz + 2xz\sqrt{1 - a^2 - y^2 + a^2y^2} \geq 0$$

Plotting the points in this set gives a convex hull that appears equivalent to the original in all cases, however the issue is that each individual equation is not convex, as it represents only a section of the original convex hull. Only by taking the union of the sets generated by each of the different equations would we get a convex hull equivalent to the original set, that would allow us to actually optimize over the space without needing extra variables as was our original intent.

However, this is where the project stops, as the above method would not scale well to other problems due to the branching nature of having to set each equation to zero, as well as the difficulty of solving higher order polynomial equations to eliminate each variable. It is left here in case any reader might take inspiration from the idea, as finding the smallest possible representation of this type of set could result in large performance gains for SDP optimisations (nonlocality or otherwise).

4.3 Iterative Approaches

In this section we consider a number of projects related to the idea of being able to split an NPA-style optimization into a number of smaller iterations, rather than what often ends up being a giant SDP. This is similar to the ideas presented in Section 3.2, which also involved splitting a large problem into many smaller ones, although the main difference being that in that case we had a small but non-convex problem, whilst here we have a large convex one. Many of the projects in this section are still ongoing, as in general it is difficult to beat the state-of-the-art solvers due to their extensive optimisation.

4.3.1 The NPA Hierarchy As a Linear Program

The first method we consider is that of attempting to transform each level in the NPA hierarchy into a corresponding linear system. The idea of transforming SDPs into linear programs (LPs) is not novel [108], however the idea was that perhaps by specifically tuning it to the quantum set one could gain some performance or memory advantage. The method is as follows: starting from the SDP corresponding to a specific level, we first remove the positive semidefinite constraint on the moment matrix M. This gives us a linear program, which we can solve to get a very relaxed solution. To then enforce positivity, rather than trying to constrain positivity over all vectors (by enforcing all positive eigenvalues) we instead simply constrain positivity only over some of the eigenvectors, adding the constraint: $x^T M x \geq 0$ where x is an eigenvector of the relaxed solution with a negative corresponding eigenvalue. This gives us a new linear program, which we can solve to get a new solution and from there repeat this process. The idea is that by doing this we can get a series of linear programs that will converge to the solution of the original SDP, whilst still only solving linear programs. A diagram showing how this process looks visually is given as Figure 4.5.

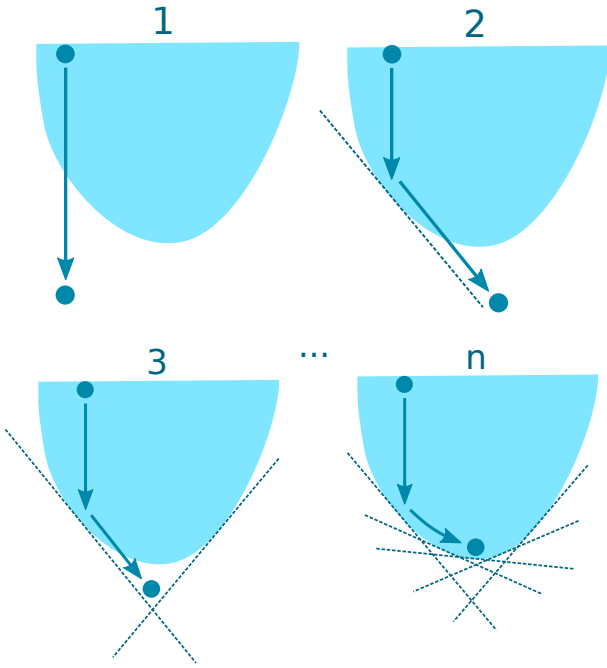

FIGURE 4.5: A diagram of the iterative linear program method. The idea is to start from a relaxed solution and then enforce positivity over the eigenvectors of the solution to get a new relaxed solution, repeating until convergence. We start with the fully relaxed problem (1), in which the SDP set is ignored, then add a constraint based on the result to give us optimisation (2) and so on, until we have eventually added $n-1$ constraints at iteration (n).

A small sample of results is provided as Table 4.1, showing the results of the method for both the CHSH inequality as well as the I3322 inequality [109]. As shown by the times relative to simply solving the SDP, the method is not so competitive, quickly requiring a very large linear system

as well as many iterations. But perhaps this is not so surprising, as one can see by Figure 4.5 that it is quite difficult to fully represent a curved surface with straight lines, needing theoretically an infinite number of constraints to do so. Whilst the hope was that linear constraints could be chosen in a smarter way, it seems that the method does not scale well. One might argue that the final linear system represents an interesting conversion of the problem, however one could much more quickly generate a linear system by first solving the SDP and then finding linear constraints based on the optimum solution.

Problem	Level	Iters	Lin Cons	Time (vs SDP)
CHSH	1	1	2	0.1s (1.6x)
I3322	1	132	373	2s (29x)
I3322	2	2083	15710	47m (22000x)

TABLE 4.1: Table of results when linearizing the NPA SDP for a number of problems and hierarchy levels. Each iterations adds a number of linear constraints to the linear program, which is solved again at each iteration. The time given is the total time for the whole method, with the relative difference versus the time to simply solve the original SDP (i.e. 29x meaning the linear version is 29 times slower than just solving the SDP). Convergence here is defined as when we get within 10^{-5} of the original SDP solution value. MOSEK is used both as the linear solver and the SDP solver, whilst Eigen's Self Adjoint Eigensolver is used to determine the eigendecomposition.

4.3.2 The DIGS Method

Another approach, which remains in development, is that of using an iterative algorithm to generate constraints of a higher level using only SDPs of lower levels. This idea is based on attempting to implement the Dynamic

Inequality Generation Scheme (DIGS) from [110], which they apply to a hierarchy of relaxations applied to classical optimisation problems. In that work, for some specific problems, they are able to reach the same bound given by the higher levels of the Lasserre hierarchy using only lower-level SDPs in an iterative fashion. More specifically, after first solving the SDP of some levels L, one then attempts to solve a new SDP at level $L + 1$ where now our aim is to find a dual solution such that it gives a negative inner product with the primal solution of the first SDP. This then acts as a type of witness, generating a linear constraint that will not be contained within the level L set, and hopefully may even be in the $L + 2$ or even $L + 3$ set. By repeating this process iteratively, the hope is that one ends up with constraints in higher-level sets without ever actually having to do any higher-level SDP or include any of the higher-level variables.

Despite the promising idea, we have yet to find a case where this works in the NPA case. For the I3322 inequality, running level 1 and level 2 SDPs (hoping to go past the level 2 bound) we simply get a series of constraints that eventually result in the level 2 bound, with further iterations not improving the bound further. We have done similar tests with a large number of randomized Bell inequalities, however none offered performance greater than the level 2 bound. As such, it may be that the method does not apply in our case, or it may simply be that it is only applicable to very specific problems.

4.4 Large-scale Problems

We now consider how we might find even a loose bound when the problems become large, as current SDP solvers generally cannot go above positive matrices more than a few hundred elements wide, thus as the size of the Bell scenario increases, solving with standard solvers becomes intractable.

The following section is all adapted from a single-author paper [111] written by the author of this thesis.

4.4.1 The Method

Consider the case where we have an SDP corresponding to a certain level of the NPA hierarchy. This SDP has an optimum which provides an upper bound for the quantum bound (sometimes referred to as "the value", for simplicity) of the Bell inequality. We first begin by taking the dual of this SDP, resulting in a new SDP whose optimum also provides an upper bound for the value, except that so too does every point in the feasible set (albeit worse bounds than the optimum). For a recap of duality, see Section 2, notably Figure 2.2. We therefore change to considering the relaxed problem of finding a point inside the feasible set as fast as possible, ideally as close to the optimum as possible.

The core of our idea is as follows: one travels as far away from the feasible set as possible in the direction of the objective function, ideally to the limit of infinity but in our case always truncating to some large (relative to the problem) distance. Then one projects back onto the set using the method of alternating projections, accelerated using a standard low-memory gradient descent algorithm known as L-BFGS. This rapidly gives a feasible point on the edge of the set, hopefully the nearest point to the far-away point but often simply a point nearby. A diagram showing this idea of "exile and projection" is given as Figure 4.6

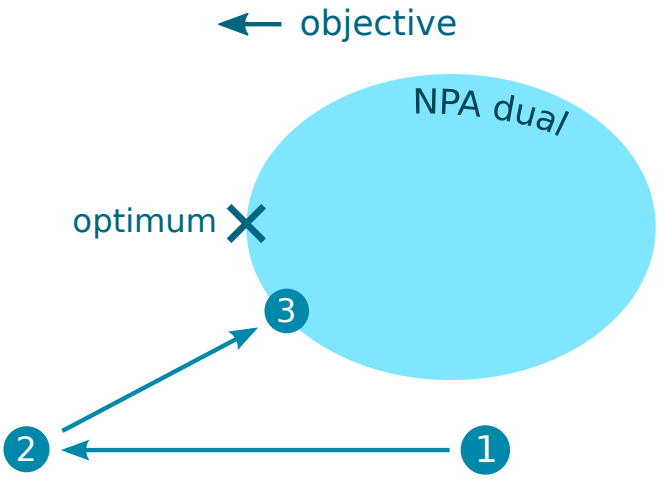

FIGURE 4.6: Diagram showing the outline of our method. Beginning with the dual problem of a given level of the NPA hierarchy, one starts at a point (point 1), generally from zero, then travel very far in the direction of the objective (point 2), before projecting back onto the set (point 3), giving an interior point near the objective and providing an upper bound for the Bell inequality.

Although this method might not give the best upper bound possible, it does so in a fraction of the time and memory than solving the SDP (primal or dual) with a state-of-the-art SDP solver. If one desires a better bound, one can then travel outside the set towards the objective function and repeat the projection, eventually converging (albeit slowly) to the optimum, however in general the focus of this work is to get a quick-and-dirty bound as fast as possible, rather than strictly the optimum.

4.4.2 Method of Alternating Projections

We now consider the problem of trying to find the closest point in a set A to a given point y ("projecting y onto A"). For many simple convex domains this can be done analytically, normally by first considering the problem:

$$\begin{aligned} \min_{x} \quad & ||x - y||^2 \\ s.t. \quad & x \in A \end{aligned} \tag{4.19}$$

One then forms the Lagrangian and solves for the KKT conditions [112]. In the case of the affine set $Ax = b$, the projection P is given by the pseudo-inverse [45]:

$$P_{\text{affine}}(y) = y - A^T(AA^T)^{-1}(Ay - b) \tag{4.20}$$

In the case of projecting a matrix onto the positive semidefinite (PSD) cone, the projection can be found by simply taking the eigenvalues of the original matrix, setting all negative eigenvalues to zero, then reconstructing the matrix [45], such that if the eigenvalues/vectors of y are λ_i and $|\phi_i\rangle$, one has

$$P_{\text{PSD}}(y) = \sum_i \max(\lambda_i, 0)\, |\phi_i\rangle \langle\phi_i| \tag{4.21}$$

Whilst projecting onto textbook-friendly sets like these is easy, projecting onto their intersection can be considerable harder (although still "easy" in a complexity sense, since SDPs are in P even if they often represent relaxations of problems seemingly not in P). The simplest method of doing this is the method of alternating projections, whereby one first projects onto one set, then the other, then repeat until one finds a point in the

set. Convergence to a point in the set is guaranteed if the sets are convex and their intersection is non-empty, although it won't necessarily be the closest point. An example of how this can occur is shown in Figure 4.7. Whilst there does exist a modification of the alternating projection method which promises to always converge to the closest point, known as Dykstra's projection algorithm [113], when the point is very far from the set this seems to always take a long time to converge and thus for our purposes offers very poor performance.

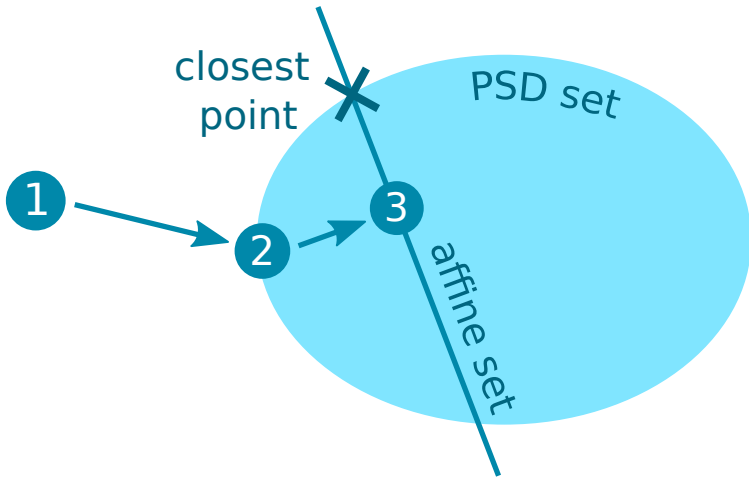

FIGURE 4.7: Diagram showing how the method of alternating projections might not always find the closest point to the point being projected (point 1) when projecting onto the intersection of a PSD and an affine set. By projecting first onto the PSD set (point 2) we move away from the true projection such that when we then project onto the affine set (point 3) we do not get the closest point. Whilst in this diagram it appears that one could simply project onto the affine set first to get the optimum, that is generally not the case.

4.4.3 L-BFGS

The alternating projection algorithm is known to converge linearly, meaning that the error at iteration i is proportional to the previous error, e.g. $\epsilon_i = 0.7\epsilon_{i-1}$. This is okay for many applications, however in our case it often required several hundred iterations to reach a tolerance of 10^{-8}. However, upon doing principal component analysis (PCA) on the sequence produced we noticed that the points follow a very smooth path, and thus may be susceptible to some kind of convergence acceleration. We first tested using Aitken's delta-squared process [114] which provided a minor speed-up (e.g. from 300 iterations to 250), however we found greater success instead applying a version of the L-BFGS algorithm.

In order to reach quadratic convergence (such that the error is proportional to the square of the previous error, e.g. $\epsilon_i = 0.7\epsilon_{i-1}^2$), one possible method is to use the full Hessian, in the style of Newton's method for root-finding [115]. This, however, requires a large number of derivative calculations, which often proves expensive in both time and memory. An important advancement was the use of the approximate Hessian by Broyden, Fletcher, Goldfarb and Shanno (BFGS), allowing a much faster iterate [62]. This was then further improved in the limited-memory version (L-BFGS) by instead finding an approximate Hessian using only a few (often less than 10) previous gradients, whilst still maintaining super-linear convergence [116].

One might note however, that we don't have an explicit nor smooth form for the gradient of our optimisation, since we are simply alternating projections. The method in which we approximate the gradient and thus a Hessian is by considering only the points after we project onto the affine set. That is, we begin at a point in the affine set, project onto the semidefinite cone and then back to the affine set. The "gradient" is then simply the difference between the two points in the affine set. A diagram demonstrating this idea is given as Figure 4.8. By feeding this gradient vector

into an existing implementation of L-BFGS as well as using the squared error as the "objective" for the L-BFGS line search, we get convergence in super-linear time at minimal memory usage. What previous required hundreds of iterations now requires only a few, at almost no extra computational cost.

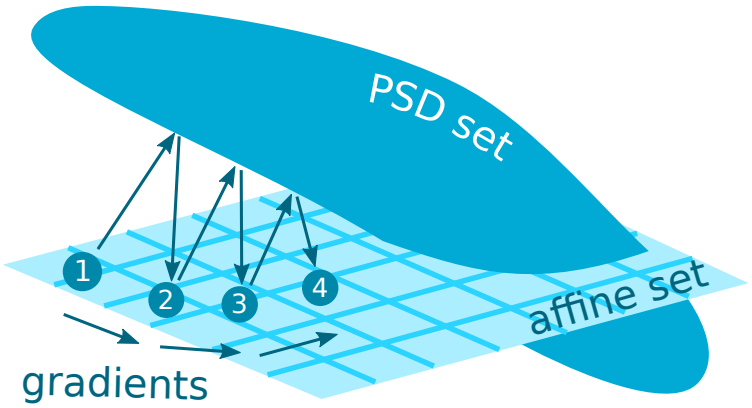

FIGURE 4.8: Diagram in 3D showing how we get the gradients for use with the L-BFGS algorithm. By projecting first onto the PSD set and then the affine set, we consider the difference between affine points as our gradients, which we then use to estimate the Hessian and thus accelerate convergence. As an aside, several proof-readers noted that this diagram may somewhat resemble a whale when viewed from afar.

4.4.4 Improving the Bound

Given the ability to find a projection at a point in the intersection very quickly as a black-box, it is possible to then get a point closer to the original point outside the set. The best method we found to do this is to simply travel from the projected point some fraction of the distance

towards the point outside the set. For instance, assuming we start at a point y and then have iterates x_i, it should be relatively clear that if we move an infinitesimal distance towards y and then project back onto the set, we get a resulting point also on the edge of the set that will be slightly closer to y. For faster convergence we take larger steps, reducing geometrically at a "slow enough" rate: $x_{i+1} = P(x_i + \alpha(y - x_i))$ such that each $\alpha_{i+1} = c\alpha_i$. Here c is some coefficient chosen to determine the rate of decrease of the step size, perhaps similar in spirit to methods such as simulated annealing which converge to the optimum if the system is "cooled" slow enough.

We have tested variations on this improvement procedure significantly, including using such steps to get an "outer" gradient and then feeding that into L-BFGS, using Aitken's delta-squared process, as well as testing various gradient descent algorithms, however we found no significant improvement. We also tested adding an extra affine constraint constraining that the objective equals a certain (lower) value, but found that this prevents L-BFGS from working efficiently and thus it seems that the superlinear convergence is restricted to just the standard moment-matrix set of constraints.

4.4.5 Implementation

Our implementation of this algorithm, as well as the test suite benchmarking against other algorithms, is all written in C++. For the implementation of L-BFGS we use Optim [117]. For all matrix/vector operations we use the C++ library Eigen [88]. In order to find the projection onto the affine set, whilst one could calculate and apply the pseudo-inverse, in our tests this often takes longer than using Eigen's implementation of BiCGSTAB [118] to attempt to solve the set of linear constraints whilst using the point to be projected as the warm start. To find the projection onto the PSD set, we use Eigen's Self Adjoint Eigensolver to get the

eigenvectors and eigenvalues, which we then use as described earlier to get the projection.

To further summarize the idea of the method, we provide some pseudocode, given as Algorithm 1. The code we use to generate all data in this work is open source and available on GitHub [119].

Algorithm 1: Summary

```
1  prepare the linear solver
2  initialize the distance
3  travel in the direction of the objective
4  project onto the affine space
5  repeat
6  |  project onto the positive cone
7  |  project onto the affine space
8  |  update the approx Hessian using this new point
9  |  use the approx Hessian to get the next point
10 until enough iterations performed
11 if better solution desired then
12 |  reduce the distance
13 |  go to line 4, keeping the current point
14 end
```

The following benchmarks were all performed with the same desktop computer: An Intel(R) Xeon(R) W-2295 CPU @ 3.00GHz (18 physical cores, 36 threads) with 128GB RAM. The method is fully parallelized due to the two slowest parts of the algorithm, the projections, already having parallelized implementations in Eigen.

4.4.6 Benchmarks - The I3322 Inequality

As a proof-of-principle, we begin with the smallest (in terms of number of inputs) inequality not reducible to the CHSH inequality, known as the I_{3322} inequality [109]. Here both Alice and Bob each can choose between three two-outcome measurements (hence the 3-3-2-2 part of the name), such that there exists a set of measurements acting on a shared quantum state that display nonlocality by violation of a classical bound, as with CHSH. A more interesting property of I_{3322} is its "hardness" despite only having a very slightly more complex description than CHSH: in order to fully characterize the quantum bound using NPA, at least level 4 of the hierarchy is required (only level 1 is required with CHSH) [1], and to find a lower bound using seesaw one is required to use a shared state more complex than a qubit [120]. In this work we consider the -1/1 formulation of this inequality, given as:

$$\begin{aligned} &\langle A_1\rangle - \langle A_2\rangle + \langle B_1\rangle - \langle B_2\rangle - \langle A_1B_1\rangle \\ &+ \langle A_1B_2\rangle + \langle A_2B_1\rangle - \langle A_2B_2\rangle + \langle A_1B_3\rangle \\ &+ \langle A_2B_3\rangle + \langle A_3B_1\rangle + \langle A_3B_2\rangle \ \leq \ 4 \end{aligned} \tag{4.22}$$

This has a maximum quantum violation of 5.5 at level 1, 5.00376 at level 2 and 5.0035 at level 3. We begin by showing in Table 4.2 that our method gives a valid upper bound with a single shot with a fraction of the resources of the other solvers, and then proceeds to converge to the optimum within precision (linear error less than 10^{-10}) using only a further 12 iterations of refinement. As will be demonstrated later, the initial bound being quite loose appears to be a feature notable of smaller problems, where perhaps one needs to have a very specific set of variable values in order to even get close to the solution.

Solver	Bound	Time	Memory
MOSEK	5.5	36 ms	26 MB
SCS	5.5	2 ms	26 MB
Ours (1)	6.34423 (+15%)	1 ms	14 MB
Ours (12)	5.5	5 ms	14 MB

TABLE 4.2: Table showing how our method performs against some common solvers when applied to level 1 of the NPA hierarchy for the I_{3322} inequality. The number given after "Ours" is the number of iterations performed, whereby 1 corresponds to just a single shot and higher numbers correspond to repeated iterations to improve the bound.

The main advantage of our method only begins to show as the size of the matrices grows, however. When now considering level 3 of the hierarchy, in which now the matrices are 88×88 rather than 7×7 as with level 1, we see that the time to do a single shot does not grow so much despite now giving a bound below that of level 1. Performing further iterations then brings the bound much closer at the expense of spending a similar amount of time as with the other solvers. These results are shown in Table 4.3.

4.4.7 Benchmarks - The RXX22 Inequality

To demonstrate the scalability of the method, we turn to bigger Bell inequalities. We consider a generalized inequality with x inputs either side, but still with two-outcomes for simplicity of generation, containing all one and two-body expectation values, whose coefficients are then randomized uniformly between -1 and 1. An example of such an inequality for $x = 2$

Solver	Bound	Time	Memory
MOSEK	5.0035	17790 ms	560 MB
SCS	5.0035	13070 ms	45 MB
Ours (1)	5.41829 (+8%)	217 ms	46 MB
Ours (100)	5.03148 (+0.5%)	12540 ms	46 MB

TABLE 4.3: Table showing how our method performs against some common solvers when applied to level 3 of the NPA hierarchy for the I_{3322} inequality. The number given after "Ours" is the number of iterations performed, whereby 1 corresponds to just a single shot and higher numbers correspond to repeated iterations to improve the bound.

inputs either side might be:

$$\begin{aligned} &0.23\langle A_1\rangle + 0.14\langle A_2\rangle - 0.9\langle B_1\rangle \\ &+ 0.82\langle B_2\rangle - 0.68\langle A_1B_1\rangle - 0.2\langle A_1B_2\rangle \\ &+ 0.11\langle A_2B_1\rangle + 0.72\langle A_2B_2\rangle \;\leq\; C \end{aligned} \tag{4.23}$$

where C is now some classical bound that can be found by optimizing over the local set, which can be phrased as a similar problem as in the quantum case except with commuting variables. Finding the value of C is not the objective of this work, but theoretically our method can also be applied in the case of any SDP, commuting or not.

Our main advantage is the time required to find an interior point. As shown by Figure 4.9, our method finishes orders of magnitudes faster than the other solvers for larger problems. This is due to the speed of projection combined with the super-linear convergence from L-BFGS.

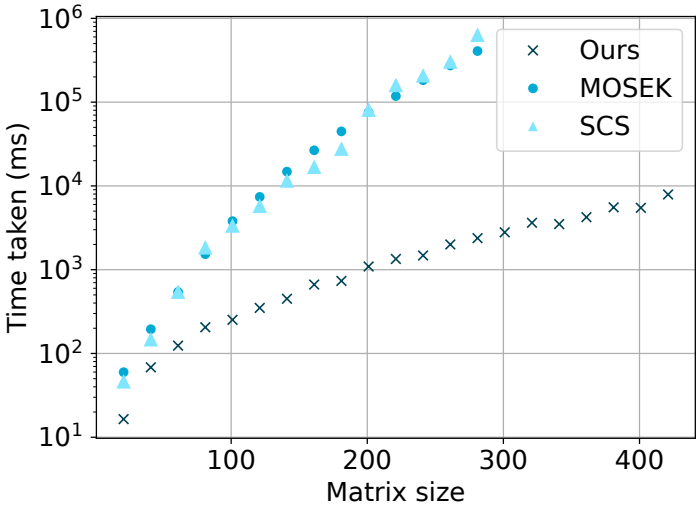

FIGURE 4.9: Graph showing how the runtime of our algorithm scales with the size of the moment matrix versus other solvers. Each point is averaged over 100 runs with unique seeds. Error bars not shown because they would be barely visible.

However, such a speed advantage is only useful if the resultant bound is not so far off the optimum. In Figure 4.10 we show that the percentage error in the solution (the upper bound given by our method versus that of MOSEK or SCS) is generally constant at around 2%, seemingly more consistent for larger problems. Thus, if one has a large Bell inequality in which they need a valid bound that needs not be tight, our method demonstrates a clear advantage.

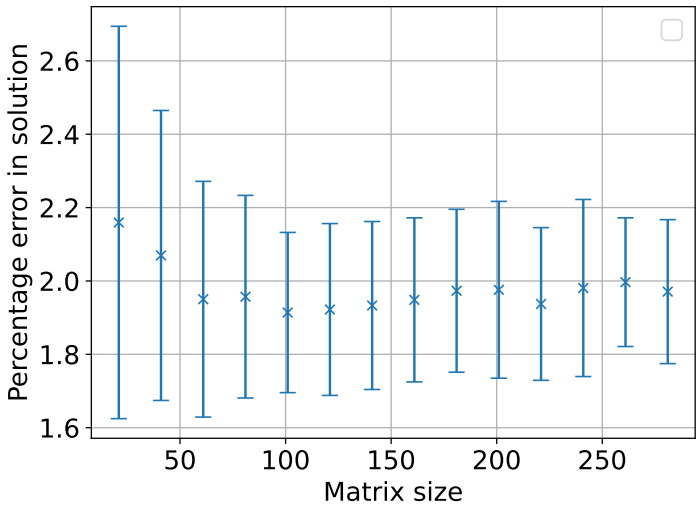

FIGURE 4.10: Graph showing how the percentage error of our algorithm scales with the size of the moment matrix. Other solvers can be said to effectively have zero relative error, since they always converge to the optimum (within precision). Each point is averaged over 100 runs with unique seeds, with the standard deviation shown as error bars.

We also analyse the peak memory usage of the algorithm, demonstrated in Figure 4.11, in which our method uses memory comparable to that of the other first order solver SCS. Second-order solvers like MOSEK require large amounts of memory to store the large matrices of second derivatives, whereas first-order methods have no such necessity. Our method combines the two, using L-BFGS to only maintain a small fraction of the information required to estimate the second-order gradients.

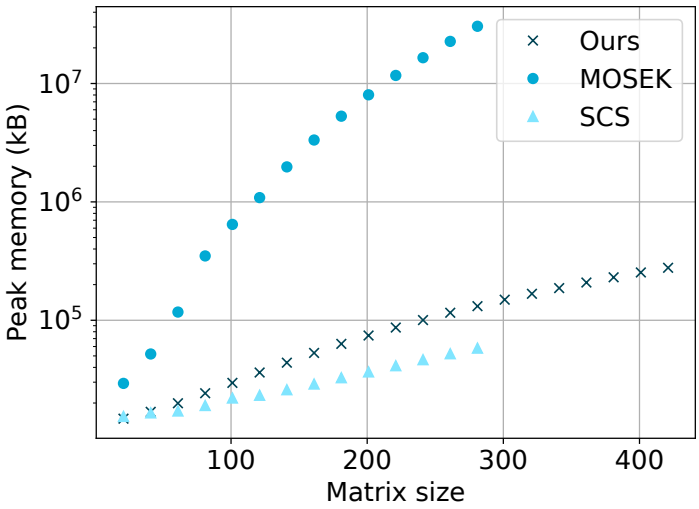

FIGURE 4.11: Graph showing how the memory usage of our algorithm scales with the size of the moment matrix versus other solvers. Each point is averaged over 100 runs with unique seeds. Error bars not shown because they would be barely visible.

4.4.8 Conclusion

In this project we describe and then demonstrate a novel algorithm for finding upper bounds of the maximum quantum violation of Bell inequalities. The method combines the method of alternating projection with the memory-efficient L-BFGS optimisation algorithm, allowing fast convergence to an interior point. Testing this method with the I_{3322} inequality we show that it gives results comparable with existing solvers. When applied to a large-scale randomized inequality that we dub R_{xx22}, it gives bounds generally within 2% error in a fraction of the time and memory

of other solvers. Such a method would prove useful in any case in which one needs a quick-and-dirty bound on any Bell inequality that one cannot otherwise optimize due to its size. The technique may also have applications when solving other SDP relaxations or when one wants to project onto large spectahedra.

Future work applications include applying it to specific Bell inequalities known to be hard, applying it to the local set to get the classical bounds, or applying the method to problems other than Bell inequalities. Given that this method is an approximation algorithm for an NP-hard problem, it has a wide range of applications. Further work also needs to be done to improve the ability of successive iterations to converge to the optimum, either with a smarter choice of step-size scaling or by using another method entirely.

Chapter 5

Many-body Quantum Systems

We now move on to the final topic of this thesis: many-body quantum systems, the study of how "large" quantum systems behave. When we say large here, we actually just mean more than a handful, since for some complicated systems even 10 qubits can be challenging due exponential scaling with the number of qubits. This is by far the most varied section, as we will touch on open-quantum systems, quantum chemistry and finally a quantum-inspired algorithm. This chapter does not have a common background section, as the problems and techniques used in each subsequent section are quite different, being linked only by the idea of trying to optimise as large a system as possible.

5.1 Bounding Steady-State Observables

5.1.1 Introduction

Open quantum systems, which interact continuously with their surrounding environment, are fundamental to understanding a vast array of physical phenomena and technological applications, including quantum optics, condensed matter physics, and quantum information processing [121, 122, 123]. These interactions often lead to dissipative dynamics that can significantly alter the system's behavior over time. Accurately determining the steady-state properties of such systems is crucial for predicting long-term behavior and for the design of quantum devices that can operate reliably under real-world conditions [124, 125].

From a general perspective, there has been increasing recent interest [126] in finding techniques to compute the stationary state and its properties [127, 128, 129], especially with variational methods [130, 131, 132, 133]. There is also attention on characterizing multiple steady states [134, 135] and designing Lindblad generators for target steady states [136, 137]. On the experimental side, an increasing effort is devoted to demonstrate stable quantum-correlated many-body states [138], steady-state superradiance in free space [139] and propose Dicke superradiant enhancement of heat current in circuit QED [140].

A common framework for modeling the dynamics of open quantum systems is through the Lindblad master equation [141, 142]:

$$\dot{\rho} = \mathcal{L}(\rho) \tag{5.1}$$

which provides a Markovian description of the time evolution of the system's density matrix [123, 143]. The Lindbladian superoperator has the

form

$$\mathcal{L}(\rho) = -i[H, \rho] + \sum_i \gamma_i \left(L_i \rho L_i^\dagger - \frac{1}{2} \left\{ L_i^\dagger L_i, \rho \right\} \right) \tag{5.2}$$

which encapsulates both the unitary evolution governed by the system's Hamiltonian and the dissipative processes arising from environmental interactions. Finding the steady state, ρ_{ss}, involves solving for the density matrix that remains static under the Lindbladian dynamics:

$$\mathcal{L}(\rho_{ss}) = 0 \tag{5.3}$$

The exact computation of the steady state, and therefore all its properties, is in principle possible [144], but soon becomes intractable when increasing the system size, as the number of parameters needed to specify it grows exponentially with the number of particles. In fact, the situation is harder than that observed for other relevant many-body problems, such as the computation of ground states, as here one has to deal with mixed states. To reach larger sizes, one usually employs variational approaches, such as tensor network methods [145], that approximate the steady state. Unfortunately, these methods do not provide any rigorous result about how close the estimated state is to the unknown steady state.

In many situations, however, having a full description of the steady state, that is, of all the parameters needed for its specification, is not needed, as one is interested in the determination of a few physical observables of relevance. Determining the complete state may in fact be a too demanding task if, in the end, one is only going to be able to compute a polynomially growing number of parameters. It may then be more practical to search for methods that target the observable(s) of interest without requiring the full reconstruction, or approximation, of the steady state.

We follow this approach and provide a scalable method to derive rigorous bounds on any observable of interest at the steady state of open-system dynamics described by a Lindblad master equation. To do so, we exploit relaxations for polynomial optimisation introduced in the context of quantum information theory [146, 1, 147] which make use of semi-definite programming (SDP). We apply our approach to a few paradigmatic models and compare them with tensor-network methods, seeing that they attain similar performance but with rigorous certificates, something difficult to obtain in variational methods [148]. Furthermore, while variational methods generally require the steady state to be unique [133], our approach also applies beyond this instance providing bounds over the whole convex set of steady states.

5.1.2 Bounding Observables as an Optimization Problem

The considered scenario consists of an open-system dynamics described by a Lindbladian superoperator $\mathcal{L}$ as in Eq. (5.2). We are then interested in bounding the expectation value of an observable O for the steady state of the system, i.e. minimizing/maximizing $\mathrm{tr}(O\rho)$ when $\mathcal{L}(\rho) = 0$. Since this is linear in ρ, combined with the fact that a density matrix should be positive, one can quite naturally formulate such a problem as an SDP. The full optimisation, however, would require an explicit form of ρ, which scales exponentially with the number of constituents, n, typically qubits. Modern SDP solvers such as MOSEK [149], SCS [54] and SDPA [55] already struggle with matrices of size 300×300, as such reconstructing the full moment matrix is not an option for anything larger than a small system.

To avoid the exponential growth, in what follows we restrict positivity constraints to moment matrices made of subsets of all possible n-qubit operators. While the operators to construct moment matrices are arbitrary, a very natural choice for n-qubit systems consists of the "Pauli strings", made of tensor products of Pauli matrices for each qubit. As a

reminder regarding the notation for Pauli strings: $\langle X_1 Y_2 \rangle$ for a 3-qubit system refers to $\mathrm{tr}((\sigma_x \otimes \sigma_y \otimes \mathbb{1})\rho)$, where σ_x and σ_y are the corresponding Pauli matrices acting on the first and second qubits, respectively.

5.1.3 Moment Matrix Positivity Constraint

For a given state ρ and set of operators $\{X_i\}$, we define A as the corresponding moment matrix (as in previous chapters) with elements

$$A_{ij} = \mathrm{tr}(\rho X_i^\dagger X_j) \tag{5.4}$$

which will be positive semidefinite for any choice of operators $\{X_i\}$. Note that in this case we know our operators (i.e. the Pauli operators) but not the state, meanwhile in device-independent implementations of this non-commuting optimisation (e.g. the NPA hierarchy) both the operators and the state are unknown. An example of our moment matrix constraint is

$$A = \begin{pmatrix} 1 & \langle X_1 \rangle & \langle Y_1 \rangle & \langle Z_1 \rangle & \langle X_2 \rangle & \langle Y_2 \rangle & \langle Z_2 \rangle \\ \langle X_1 \rangle & 1 & -i\langle Z_1 \rangle & i\langle Y_1 \rangle & \langle X_1 X_2 \rangle & \langle X_1 Y_2 \rangle & \langle X_1 Z_2 \rangle \\ \langle Y_1 \rangle & i\langle Z_1 \rangle & 1 & -i\langle X_1 \rangle & \langle Y_1 X_2 \rangle & \langle Y_1 Y_2 \rangle & \langle Y_1 Z_2 \rangle \\ \langle Z_1 \rangle & -i\langle Y_1 \rangle & i\langle X_1 \rangle & 1 & \langle Z_1 X_2 \rangle & \langle Z_1 Y_2 \rangle & \langle Z_1 Z_2 \rangle \\ \langle X_2 \rangle & \langle X_1 X_2 \rangle & \langle Y_1 X_2 \rangle & \langle Z_1 X_2 \rangle & 1 & -i\langle Z_2 \rangle & i\langle Y_2 \rangle \\ \langle Y_2 \rangle & \langle X_1 Y_2 \rangle & \langle Y_1 Y_2 \rangle & \langle Z_1 Y_2 \rangle & i\langle Z_2 \rangle & 1 & -i\langle X_2 \rangle \\ \langle Z_2 \rangle & \langle X_1 Z_2 \rangle & \langle Y_1 Z_2 \rangle & \langle Z_1 Z_2 \rangle & -i\langle Y_2 \rangle & i\langle X_2 \rangle & 1 \end{pmatrix} \succcurlyeq 0 \tag{5.5}$$

We refer to this as the level 1 moment matrix, as the top row (or leftmost column) contains only first-order expectation values. The level 2 moment matrix would instead contain all first- and second-order expectation values in the top row, and so on. Here we also use the commutation rules between different sites (i.e. $[X_i, X_j] = 0$ for $i \neq j$) as well as the Pauli reduction rules ($X_i X_i = \mathbb{1}$, $X_i Y_i = i Z_i$, etc.) to simplify the matrix. In general,

the size of moment matrix of level k grows like n^k, polynomially with the number of qubits.

5.1.4 Linear Lindbladian Constraints

Whilst moment matrix optimisation has been well documented in the literature [150, 151], including the case of bounding observables for many-body ground states [152], the case of steady-state optimisation offers an extra set of constraints. Based on the definition of the steady state, we have the extra constraint from Eq. (5.3) that the Lindbladian acting on our state should be zero. Converting this to expectation values, this implies that all time-independent observables must have constant expectation values at steady state.

We can also change to using the adjoint Lindbladian [123] in order to derive constraints in terms of moments of observables acting on ρ. Considering a generic operator A and the definition of adjoint Lindbladian $\mathcal{L}^\dagger$ we have that

$$\begin{aligned} &\mathrm{tr}(A\mathcal{L}(\rho_{ss})) = \mathrm{tr}(\mathcal{L}^\dagger(A)\rho_{ss}) = \langle \mathcal{L}^\dagger(A)\rangle_{ss} \\ &\mathcal{L}^\dagger(A) = i[H,A] + \sum_i \gamma_i \left(L_i^\dagger A L_i - \frac{1}{2}\left\{ L_i^\dagger L_i, A \right\} \right) \end{aligned} \tag{5.6}$$

Since for any operator A the expectation value $\langle \mathcal{L}^\dagger(A)\rangle_{ss}$ should be zero, we therefore have an equation in which we can place any operator to give us a linear constraint on the system. Placing all possible $4^n - 1$ Pauli strings together with the trace-one condition, $\langle \mathbb{1}\rangle_{ss} = 1$ provides a fully constrained system of 4^n independent linear equations, which can be used to retrieve the steady state from its Pauli decomposition. Note that there always is at least one steady state for a master equation in Lindblad form [123]. Such a problem is solvable at small values of n, say $n \lesssim 8$, using

a sparse linear solver, in our case the Least Squares Conjugate Gradient solver from the C++ library Eigen [88].

5.1.5 Automatic Constraint Generation

As it would be impractical to include all $4^n - 1$ constraints, we can instead choose a subset of the Pauli strings to include and corresponding linear constraints. The choice of these operators is arbitrary but, in what follows we apply the following recipe: we begin by placing the expectation values of a Pauli string appearing in the decomposition of the observable of interest O into the adjoint Lindbladian to generate a new linear constraint. We then take the moments that appear in the resulting constraint and plug them back into the adjoint Lindbladian. Repeating such a process we generate a series of linear constraints which appear to be much more relevant than those generated by simply inserting random operators or by inserting all Pauli strings of a certain order. In some cases we note that this eventually reaches a "cycle", in that no new operators are generated by the insertion of the previous operators. In this case, the resulting constraints appear to always form a fully constrained linear system, which can then be solved as before.

A similar method can also be employed to choosing which moments should be included in the top row of the moment matrix, rather than simply putting all level 1/2/3 moments. We refer to this idea, both in the linear and matrix constraints, as "automatic" constraint generation, which we do until we hit some limit or there are no new moments to be found.

5.1.6 State Reconstruction Positivity Constraints

Whilst it is completely infeasible to constrain the positivity of the whole density matrix for anything larger than a few qubits, what remains feasible is the idea of imposing positivity of all the reduced density matrices of a

given, small, number of qubits, $m \ll n$. This positivity constraints are again function of the Pauli strings acting on the corresponding m sites. By constructing all m-site density matrices and enforcing their positivity, we further constrain the set of feasible moments and improve the tightness of our bounds, often significantly, at minimal computational cost. As an example, for $m = 2$ we might constrain that the reduced matrix ρ_{12} of qubits 1 and 2 is positive:

$$\rho_{12} = \frac{1}{2^m}(\mathbb{1} \otimes \mathbb{1} + \langle X_1 \rangle \sigma_x \otimes \mathbb{1} + \langle X_1 Y_2 \rangle \sigma_x \otimes \sigma_y + ...) \succcurlyeq 0 \tag{5.7}$$

5.1.7 Symmetry Constraints

Until now we have considered the general case of a generic Lindbladian. However, if we know that the system takes some form, there can be additional constraints we can exploit. Here, let us assume the system has a single steady state (i.e. a non-zero Liouvillian gap). For example, if we know that the system is invariant under some swap of qubits, we can further constrain that their respective expectation values should be equal. For instance, if we have a simple two-by-two grid system, such that the two leftmost qubits are connected to the hot bath and the two rightmost qubits are connected to the cold bath, we can enforce that the expectation values of the leftmost qubits are equal, as well as their combined expectation values, with the rightmost qubits, following the symmetries. Such constraints are "for free" in that they add negligible computational cost, but can often improve the bounds by a non-negligible amount.

5.1.8 Moment Based Optimization Problem

With all this in mind, we can now formulate our optimization problem in a more general formalism.

To begin, let us define an index α iterating over a subset of possible 4^n Pauli strings. The set of Pauli strings is chosen heuristically based on the "most relevant moments", i.e. the ones that appear most in the problem. For scalability, either the maximum degree or maximum number of Pauli strings is bounded. We then define the corresponding operator for each Pauli string, P_α, and the corresponding expectation value $\langle P_\alpha \rangle$. The operator O that we want to bound is written as a linear combination with known coefficients c_α, such that $O = \sum_\alpha c_\alpha P_\alpha$. Thus, the complete problem can then be written as:

$$\begin{aligned} \min_{\langle P_\alpha \rangle}(\max) \quad & \sum_\alpha c_\alpha \langle P_\alpha \rangle \\ \text{s.t.} \quad & -1 \leq \langle P_\alpha \rangle \leq 1\,, \forall \alpha \\ & \langle \mathcal{L}^\dagger(P_\alpha) \rangle = 0\,, \forall \alpha \\ & \sum_\alpha A_{k,\alpha} \langle P_\alpha \rangle \succcurlyeq 0 \quad \forall k \in \{1...m_p\} \\ & \sum_\alpha b_{k,\alpha} \langle P_\alpha \rangle = 0 \quad \forall k \in \{1...m_s\} \end{aligned} \tag{5.8}$$

Here the m_p linear matrix inequality constraints defined by known matrices $A_{k,\alpha}$ can correspond either to moment matrix [as, e.g., in (5.5), obeying some Pauli replacement rules] or to reconstructed reduced density matrix positivity. The m_s linear equality constraints defined by coefficients $b_{k,\alpha}$ represent the various symmetries specific to the problem. Bounds (here ± 1) on the operators are not always necessary, but provide a "safety net" to prevent the problem being unbounded in the case that one of the moments is not sufficiently constrained.

It should be remarked that our construction provides rigorous bounds. They hold both in case the steady state is unique, a typical assumption in current estimation approaches [133], or not.

5.1.9 Example - Two-Qubit Test Case

We first consider the simple test case discussed by Refs. [153, 154]. The problem they consider is to find the steady-state heat current for a two-qubit chain, in which the leftmost qubit is connected to a hot bath and the rightmost qubit is connected to a cold bath. Choosing the local master equation, the open system dynamics is described by the following Lindbladian:

$$\begin{aligned}\mathcal{L}(\rho) &= -i\left[H_S + H_{\text{int}}, \rho\right] \\ &+ \sum_{j\in\{h,c\}} \left(\gamma_j^+ D\left[\sigma_+^{(j)}\right]\rho + \gamma_j^- D\left[\sigma_-^{(j)}\right]\rho\right)\end{aligned} \tag{5.9}$$

where

$$D[A]\ B = ABA^\dagger - \frac{1}{2}\{A^\dagger A, B\} \tag{5.10}$$

The Hamiltonian terms are set as

$$H_S = \sum_{j\in\{h,c\}} \varepsilon_j \sigma_+^{(j)}\sigma_-^{(j)} \tag{5.11}$$

$$H_{\text{int}} = g\left(\sigma_+^{(h)}\sigma_-^{(c)} + \sigma_-^{(h)}\sigma_+^{(c)}\right) \tag{5.12}$$

with ϵ_h, ϵ_c the energy gaps of the qubits, and g the coupling between the two qubits. To shorten the notation, whenever there is a j as a subscript from now on in this chapter it's either h or c. The kinetic coefficients

$$\gamma_j^+ = \gamma_j n_B^j \quad \text{and} \quad \gamma_j^- = \gamma_j\left(1 + n_B^j\right) \tag{5.13}$$

account, respectively, for absorption from and emission to bath j and $n_B^j = 1/(e^{\epsilon_j/T_j} - 1)$ are the Bose factors. Furthermore, γ_h, γ_c are the coupling of the qubits with their respective baths, T_h, T_c (the temperatures

of the baths). A diagrammatic representation of the system is given in Figure 5.1. In [154] the authors focus on the heat current defined as

$$\begin{aligned} J_{ss} &= Q_h^{ss} - Q_c^{ss}\,, \\ Q_j^{ss} &= \mathrm{tr}\left[H_S\left(\gamma_j^+ D\left[\sigma_+^{(j)}\right] + \gamma_j^- D\left[\sigma_-^{(j)}\right]\right)\rho_{ss}\right] \end{aligned} \tag{5.14}$$

To put this into our notation, we replace the plus and minus operators with the corresponding Pauli strings:

$$\sigma_h^+ = \frac{1}{2}(\sigma_h^x + i\sigma_h^y) \tag{5.15}$$

$$\sigma_h^- = \frac{1}{2}(\sigma_h^x - i\sigma_h^y) \tag{5.16}$$

$$\sigma_c^+ = \frac{1}{2}(\sigma_c^x + i\sigma_c^y) \tag{5.17}$$

$$\sigma_c^- = \frac{1}{2}(\sigma_c^x - i\sigma_c^y) \tag{5.18}$$

obtaining the objective function in terms of Pauli strings:

$$\begin{aligned} J_{ss} = &\frac{\epsilon_h}{2}(\gamma_h^+ - \gamma_h^-) - \frac{\epsilon_c}{2}(\gamma_c^+ - \gamma_c^-) \\ &- \frac{\epsilon_h}{2}(\gamma_h^+ + \gamma_h^-)\,\langle\sigma_h^z\rangle_{ss} + \frac{\epsilon_c}{2}(\gamma_c^+ + \gamma_c^-)\,\langle\sigma_c^z\rangle_{ss}\,. \end{aligned} \tag{5.19}$$

This system has a known analytical solution for the ρ_{ss}, given in Ref. [154]. This allows one to calculate the heat current (5.14), e.g., obtaining $J_{\mathrm{ss}} = 0.000208933$ for $\epsilon_h = 1$, $\epsilon_c = 1.0005$, $T_h = 1$, $T_c = 0.1$, $g = 1.6\times10^{-3}$, $\gamma_h = 10^{-3}$, and $\gamma_c = 1.1 \times 10^{-2}$.

In this case, due to the system being only two qubits, we can solve the full system of $4^2 = 16$ linear equations, giving upper and lower bounds which

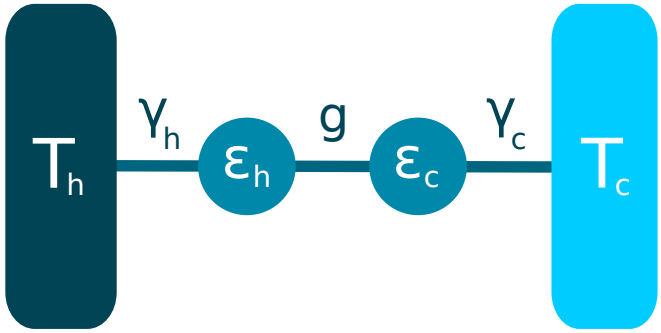

FIGURE 5.1: Two-qubit example. Here the leftmost qubit is connected to a hot bath and the rightmost qubit is connected to a cold bath. The system is defined by parameters γ_h, γ_c (coupling of the qubits with their respective baths), ϵ_h, ϵ_c (energy gaps of the qubits), T_c, T_h (the temperatures of the baths) and g (coupling between the two qubits).

only differ of values compatible with the numerical error of the solver (in this case 10^{-14}). A table of the various bounds given by different combinations of the aforementioned constraints is given as Table 5.1, and serves as the first example of how simply "brute-forcing" all possible insertions into the Lindbladian is often not necessary, as solving with 6 automatic linear constraints results in the exact solution, rather than needing to insert all 15 second-order Pauli strings. This is consistent with the analytical solution given in equations 6-8 in Ref. [154], in which there are also 6 non-zero variables and corresponding linear equations.

Extending this example to many qubits by simply adding extra qubits between the hot and cold qubits with the same coupling results in a system which appears to be very easy to solve. The automatic linear constraint generation always appears to reach a cycle after $2n^2 - n$ constraints, resulting in a full-rank linear system that can be solved exactly. This has been tested up to 100 sites (19900 linear constraints and the same number of variables). As such, it is assumed that this problem is therefore in P since it appears to be easily solved in polynomial time. It demonstrates a possible advantage of our constraint set, such that some systems may be shown themselves as being easily solvable.

5.1.10 Example - Periodic 1D Chain

The next system we consider is that of a periodic n-site 1D chain. The Hamiltonian used is the one used by Ref. [133], the transverse Ising model, specifically in which they consider a 1D chain such that the final qubit is also coupled to the first qubit and each qubit also coupled to a heat bath. The qubits are coupled to each other with some coefficient J, the system has some field h and the qubits are connected to the heat bath with coefficient γ_-. A diagram of this system is given as Figure 5.2. The

Hamiltonian is as follows,

$$H = J \sum_{\langle i,j \rangle}^{n} Z_i Z_j + h \sum_{i}^{n} X_i \tag{5.20}$$

where $\langle i, j \rangle$ denotes nearest neighbours. To form the full Lindbladian we then use the spin decay operator $\Gamma_k = \frac{\sqrt{\gamma_-}}{2}(X_k - iY_k)$ such that our full Lindbladian is:

$$\mathcal{L}(\rho) = -i[H, \rho] + \sum_{k}^{n} \left(\Gamma_k \rho \Gamma_k^\dagger - \frac{1}{2}\{\Gamma_k^\dagger \Gamma_k, \rho\} \right) \tag{5.21}$$

We then consider the case of $n = 12$, $J = 0.5$, $h = 0.5$ and $\gamma_- = 1$ as given in Ref. [133]. In their work they utilize a matrix-product ansatz combined with a Monte-Carlo style optimisation. For the 12-site chain, they obtain an estimate for the average magnetization ($\frac{1}{n}\sum_i^n \langle Z_i \rangle$) within 1% of the known optimum in approximately 22s (based on their Figure 5), whilst we obtain bounds of the optimum $\pm 1\%$ in approximately 50s. Whilst their approximation is faster, our method provides rigorous bounds on the quantity in a time of a similar order of magnitude. A table detailing the bounds for the various constraints is given as Table 5.2.

To demonstrate how the method scales with larger system sizes for this specific problem, we plot in Figure 5.3 the bound difference given using the same number of linear constraints and same moment matrix size for a number of system sizes, ranging from 12 to 100. Although the bounds become worse (from about 8% at 12 sites to 25% at 100 sites), it does not appear to be exponentially worse. Also, it should be noted that the sign of the observable is still certified even for the large system. It is worth mentioning that the computation of all these points require very modest

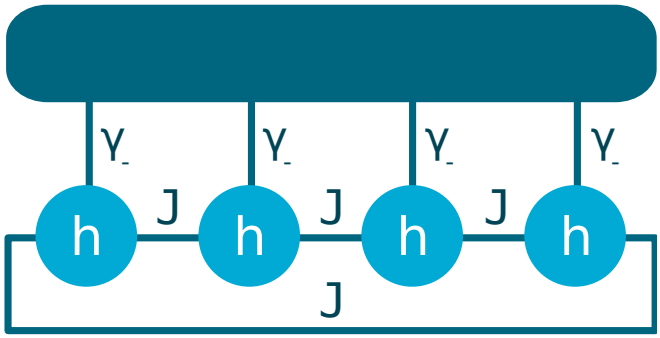

FIGURE 5.2: 1D periodic chain example. Here each qubit is coupled to the heat bath with coefficient γ_- and coupled to its nearest neighbours with coefficient J. The final qubit is also coupled to the first qubit. A field of coefficient h is also applied to each qubit in the X direction.

efforts and, in particular, the time to optimize is roughly constant within any given constraint set. For instance, for the largest constraint set each point takes around a minute on a desktop computer independently of the number of qubits.

5.1.11 Example - 2D Ladder

To explore how our method works in the 2D case, we now consider a simple 2D ladder system, as shown in Figure 5.4, effectively we have two 1D chains linked at each qubit, coupled to a hot and cold bath at either end of the chain. Here we use an extension of the Lindbladian and Hamiltonian from Eq.(5.9):

$$\mathcal{L}(\rho) = -i[H, \rho] + \sum_{j}^{n} (\gamma_j^+ D[\sigma_+^{(j)}]\rho + \gamma_j^- D[\sigma_-^{(j)}]\rho), \tag{5.22}$$

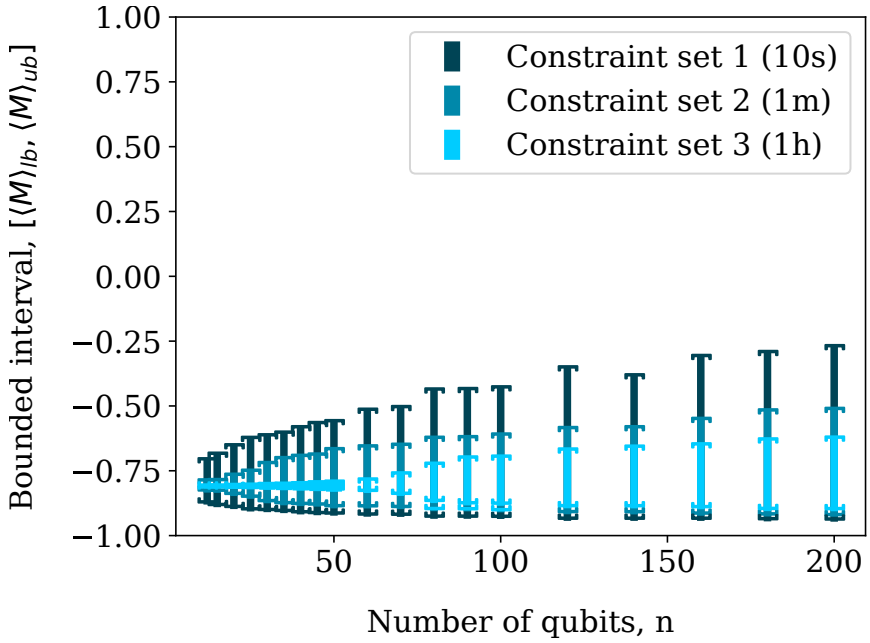

FIGURE 5.3: Upper and lower bounds on the magnetization $\langle M\rangle$ at the steady state as a function of the system size for the periodic linear chain. The plotted intervals represent the certified region in which the true value of the observable lies. The y-axis is scaled to $[-1, 1]$ to demonstrate the fraction of the trivial bounds that we restrict to. We apply method (5.8) imposing a limited number of linear constraints and moment matrix size (*set 1* - 10000 linear constraints and a 40×40 moment matrix, *set 2* - 30000 constraints and a 100×100 matrix, *set 3* - 70000 constraints and a 250×250 matrix). Symmetry between all qubits is also imposed. The time to optimize per point was roughly the same within each constraint set (*set 1* $\approx$ 10 seconds, *set 2* $\approx$ 1 minute, *set 3* $\approx$ 1 hour).

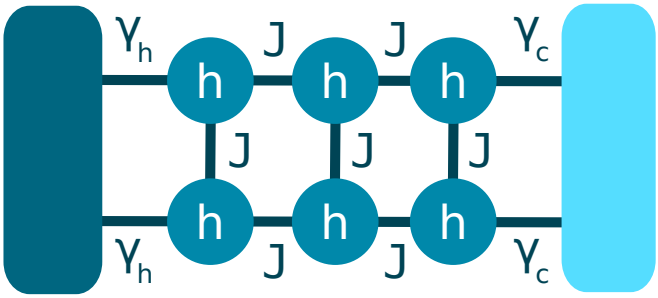

FIGURE 5.4: 2D ladder example. Here the end qubits are connected to the respective heat bath with coefficient γ_h/γ_c and coupled to its nearest neighbours with coefficient J. A field of coefficient h is also applied to each qubit in the Z direction.

$$H = \sum_{i}^{n} \varepsilon_i \sigma_+^{(i)} \sigma_-^{(i)} + g \sum_{\langle i,j \rangle}^{n} (\sigma_+^{(i)} \sigma_-^{(j)} + \sigma_-^{(i)} \sigma_+^{(j)}) \,, \tag{5.23}$$

Even adding this single extra row of qubits to the system appears to make the system significantly harder. For the case of $n = 10$, $J = 1$ and $h = 1$, for the same number of constraints used in the 12-qubit linear chain above gave bounds of 1%, we now get on the order of 36%. For specific results of different combinations of constraints for this system, see Table 5.3, with the largest constraint set reaching bounds of 36% after 2.5 hours. As the bounds do not improve even with larger constraint sets, it seems to suggest that the system has multiple steady states and thus these bounds may be tight.

Moment Matrix	**Linear**	**State Recon.**	**Sym.**	**Bounds**	**Time**
None	level 1 (6)	None	None	[-0.000001, 0.001167] (8.83%)	0.03s
level 1 (7×7)	level 1 (6)	None	None	[-0.000001, 0.001167] (8.83%)	0.04s
None	level 2 (15)	None	None	[0.000209, 0.000209] (0.0%)	0.05s
None	auto (6)	None	None	[0.000209, 0.000209] (0.0%)	0.03s

TABLE 5.1: Bounds on the average heat current for the two-qubit system for a variety of possible constraints (the first four columns). The bounds given are the strict lower and upper bounds for the heat current in the steady state, whilst the percentages represent the fraction of the full $[-1, 1]$ region that the bounds cover. The other values in parentheses are either the size of matrix or number of constraints added. The time taken for this case is almost entirely setup time, hence the lack of change with more/-less constraints.

Moment Matrix	**Linear**	**State Recon.**	**Sym.**	**Bounds**	**Time**
None	auto (1000)	None	None	[-1.00000, -0.27468] (36.27%)	0.38s
None	auto (10000)	None	None	[-0.92617, -0.67090] (12.76%)	22.87s
level 1 (37×37)	auto (10000)	None	None	[-0.88268, -0.69774] (9.25%)	9.97s
auto (51×51)	auto (10000)	None	None	[-0.88035, -0.70240] (8.90%)	11.70s
None	auto (10000)	all 4-site (794)	None	[-0.82115, -0.80172] (0.97%)	49.49s
auto (51×51)	auto (10000)	all 4-site (794)	None	[-0.82115, -0.80171] (0.97%)	1m13s
None	auto (10000)	all 4-site (794)	yes (66)	[-0.82089, -0.80215] (0.94%)	59.76s
None	auto (15000)	all 4-site (794)	yes (66)	[-0.81935, -0.80388] (0.77%)	1m26s
None	auto (30000)	all 4-site (794)	yes (66)	[-0.81740, -0.80551] (0.59%)	2m43s

TABLE 5.2: Bounds on the average magnetisation for the 12-site periodic 1D chain for the transverse Ising model coupled to a bath for a variety of possible constraints (the first four columns). The bounds given are the strict lower and upper bounds for the heat current in the steady state, whilst the percentages represent the fraction of the full $[-1, 1]$ region that the bounds cover. The other values in parentheses are either the size of matrix or number of constraints added.

Moment Matrix	**Linear**	**State Recon.**	**Sym.**	**Bounds**	**Time**
None	auto (1000)	None	None	[-0.9788, 0.3077] (64.32%)	0.96s
auto (101×101)	auto (10000)	None	None	[-0.8822, 0.1749] (52.86%)	37.48s
None	auto (10000)	all 4-site (386)	None	[-0.8531, -0.1285] (36.23%)	40.92s
None	auto (10000)	all 4-site (386)	yes (5)	[-0.8531, -0.1285] (36.23%)	41.09s
None	auto (50000)	all 4-site (386)	yes (5)	[-0.8524, -0.1327] (35.99%)	25m37s
None	auto (100000)	all 4-site (386)	yes (5)	[-0.8524, -0.1328] (35.98%)	147m35s

TABLE 5.3: Bounds on the average magnetisation for the 2×5 ladder chain for a variety of possible constraints (the first four columns). The bounds given are the strict lower and upper bounds for the heat current in the steady state, whilst the percentages represent the fraction of the full $[-1, 1]$ region that the bounds cover. The other values in parentheses are either the size of matrix or number of constraints added.

5.1.12 Conclusion

We have introduced a novel approach to bound the expectation values of observables in the steady state of open quantum systems. By formulating the problem as an SDP and incorporating various enhancements - such as linear constraints from the adjoint Lindbladian, moment matrix generation, partial state positivity, and the exploitation of system symmetries - we have significantly improved computational performance and accuracy. Our method has been rigorously tested on systems including a two-qubit chain, a one-dimensional chain, and a two-dimensional ladder. We achieved bounds within 1% error of the optimal value for a 12-site linear chain, demonstrating results comparable to state-of-the-art tensor-network approaches, as well as nontrivial bounds for systems composed of hundreds of qubits. These findings underscore the effectiveness of our method in providing tight bounds efficiently, making it a valuable tool for analyzing and certifying steady-state properties in open quantum systems.

Looking ahead, there are several promising directions for future research. A straightforward extension would be to apply these methods to study transport properties in quantum systems [155]. Here one could, for example, bound the current operator and try to extract the transport type or diffusion constant. Another promising application is to use the behavior of the difference between upper and lower bound in terms of a Liouvillian parameter to detect dissipative quantum phase transitions.

Furthermore, one could extend this SDP framework to study open quantum systems with more intricate interactions and higher-dimensional configurations, potentially including disorder and interactions beyond nearest neighbours. Another important direction is the integration of this method with machine learning techniques to optimize constraint selection and improve solver performance further [156]. Finally, investigating the combination of our SDP method with other numerical techniques, such

as quantum Monte Carlo or advanced tensor-network methods, may yield hybrid approaches that capitalize on the strengths of multiple computational strategies.

5.2 Optimising Fermion-to-Qubit Mappings

One promising and well-cited application of quantum theory is that of quantum chemistry - the idea that in order to fully understand many of the most complicated chemical processes one needs to treat molecules quantum-mechanically. The ability to do such fine modelling could theoretically allow improvements in medicine [157] and material design [158], as such it is a very active area of research. Yet, despite the attention, such problems are still inherently quantum and thus incredible difficult to simulate even for simple molecules. Whilst techniques utilising quantum hardware can help, designing such devices also remains a challenge, as does the mapping of the quantum chemistry problem to the quantum hardware. This section will briefly discuss the work done in optimising such mappings, adapted from the work in [159].

Quantum chemistry problems are normally first described by a Hamiltonian containing a number of Fermionic operators, often given as the ladder operators (e.g. $a_i^\dagger$, a_i, $a_i^\dagger a_j$). In order to consider using digital quantum hardware like a quantum computer to help to optimize such a Hamiltonian, one first needs to convert the Fermionic operators to qubit operators, often given as Pauli strings (e.g. X_i, Y_i, X_iY_j). The resulting transformation is known as a Fermion-to-qubit mapping, which is a non-unique object that heavily affects the performance of the optimisation [160]. Qubit systems, especially those based around near term (often called NISQ - Noisy Intermediate-Scale Quantum) hardware, are often

restricted to local interactions and thus care needs to be taken when mapping to this space to ensure that any non-local properties of the original system are preserved [161].

Perhaps the simplest Fermion-to-qubit mapping is that of the Jordan-Wigner transformation, which is a one-to-one mapping that preserves the anti-commutation relations of the Fermionic operators [162]. However, this mapping is not local and can lead to qubit Hamiltonians with long-range interactions, which can be difficult to simulate on quantum hardware. Other mappings, such as the Bravyi-Kitaev transformation, are more local but are not one-to-one and can lead to qubit Hamiltonians with a larger number of terms [163]. The choice of mapping is therefore often a trade-off between the locality of the qubit Hamiltonian and the number of terms.

One method of representing a ladder-to-Pauli mapping is by using a structure known as a **ternary tree** [161]. This is an extension of the concept of a binary tree - a branching structure in which each node (a parent node) has some nodes connected beneath it (child nodes), which each then have their own child nodes and so on. For a binary tree, each parent has two children, whilst in a ternary tree each parent has three children. In the case of mapping to Pauli strings, each branch corresponds to a specific Pauli operator, either X (left branch), Y (middle branch) or Z (right branch), whilst each node corresponds to a qubit. The tree is then traversed in order to generate the Pauli string corresponding to a given Fermionic operator. A simple diagram showing a ternary tree is given in Figure 5.5.

When given a Hamiltonian in the form of Pauli strings, one can then turn to quantum hardware (or in our case, a simulator) in order to try to minimize the energy and attempt to find the ground state of the molecule. To do so, one technique is that of the Variational Quantum Eigensolver (VQE), where one creates a parameterized quantum circuit (also called an **ansatz**) which spans a subset of the possible quantum states, which one

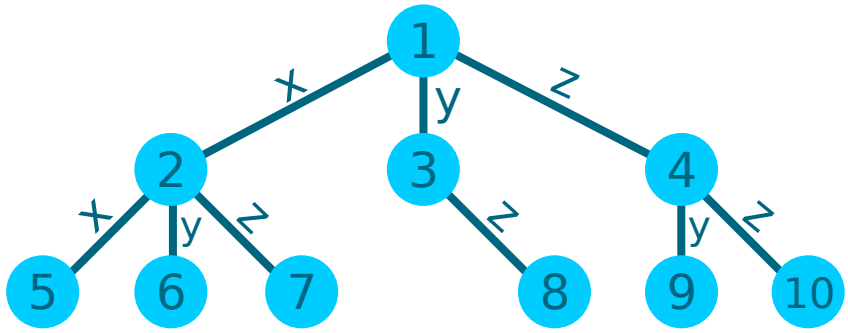

FIGURE 5.5: A simple example of a ternary tree. Each node corresponds to a qubit, whilst each branch corresponds to a Pauli operator. The tree is traversed in order to generate the Pauli string corresponding to a given Fermionic operator, which can then be used to convert from a Fermionic Hamiltonian to a qubit/Pauli Hamiltonian.

then tunes to get closer and closer to the eigenstate giving the minimal energy [164]. This tuning is performed classically using standard algorithms for gradient descent after each run of the circuit, thus the algorithm is a hybrid classical-quantum algorithm: quantum for exploring a subset of the exponentially large search space, then classical for the actual optimisation. Although VQEs still have many problems (notably: barren plateaus [165], convergence issues [166]), they represent a promising technology.

The overall approach of the work was as follows: we designed a scheme to generate a mapping in the form of a ternary tree, specific to a given molecule, that when applied and then optimized using methods such as VQE or Density Matrix Renormalisation Group (DMRG) [167] we obtain improved ground state energies for the molecule compared to using the more standard mappings. More specifically, in many cases the mapping reduced the amount of entanglement required to simulate the molecule, allowing for a circuit with less depth in the case of VQE and reduced bond dimension in the case of DMRG.

In order to generate our mapping, we first use the method of Unitary Coupled Cluster of Singlets and Doubles (UCCSD), or more specifically the pairwise version UpCCGSD [168], in order to get a series of parameterized angles which are each related to the various single and double excitations. The absolute values of these angles give us an indication of which excitations are most "important". We then take the most important excitations and encode them into the X (left) branch of our ternary tree mapping, whilst all other excitations are mapped into the Z (right) branch of the tree.

Given such a tree, and either true knowledge or an estimation of the ground state, we can then optimize over permutations of relabelled qubits to try to lower the distance of qubit pairs with high mutual information (MI), theoretically allowing simulation with shorter-range interactions. A

demonstration of our mapping for lithium hydride is given in Figure 5.6, as well as the MI matrix before and after optimisation.

After generating our mapping using the above method, we then move to testing how it performs when applying both VQE and DMRG to find the ground-state energy of the molecule. For all of the molecules tested in the work (LiH, H_2, H_4, $(H_2)_2$, N_2) and for any given number of VQE layers or bond dimension, our mapping finds a lower or equal energy ground state than Jordan-Wigner, even if we also optimize Jordan-Wigner's qubit ordering in the same way as our mapping for a fairer comparison. A example for both VQE and DMRG are shown in Figure 5.7.

One particular contribution by the author of this thesis was that of a brute-force search over all possible ternary trees, combined with a brute-force search over all qubit reorderings, in order to compare our mapping for H_2 to all other mappings. Said optimisation was computationally intensive due to the combinatorial problem of exploring the exponential number of ternary trees and qubit orders, requiring heavy optimisation and parallelism. In doing so, it was shown that our mapping had much lower mutual information than 81% of maps, with all maps with lower MI also displaying worst 2-layer VQE performance, hinting that our map for hydrogen maybe be optimal or almost optimal for low-layer VQE. A plot showing the results of this search is given in Figure 5.8.

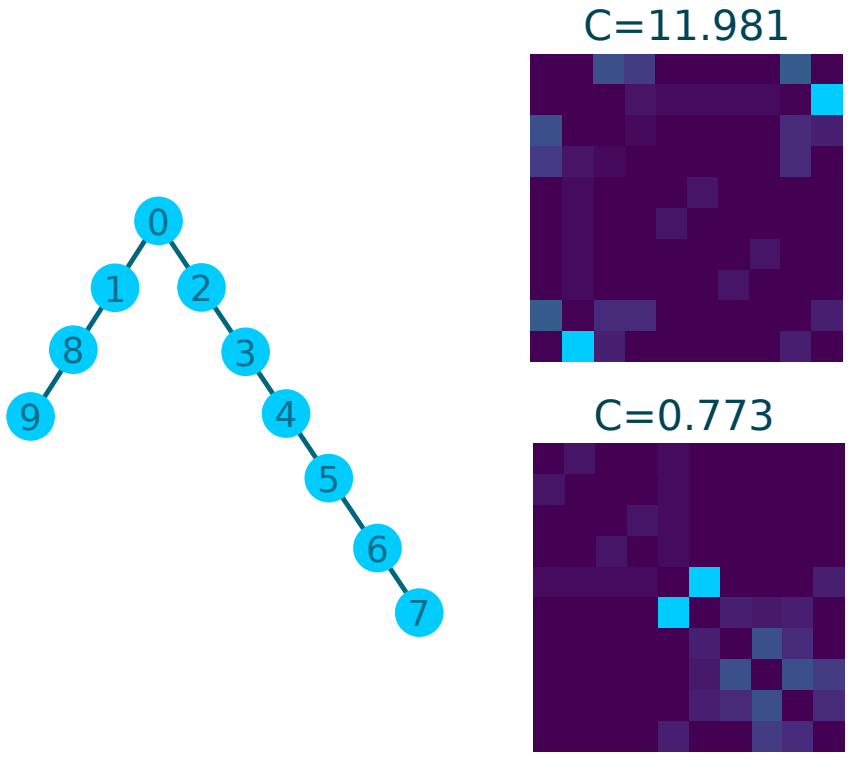

FIGURE 5.6: The mapping for lithium hydride and its mutual information matrices before and after optimization using a genetic algorithm. Here each element of the matrix is the mutual information I_{ij} between two sites i and j, and $C = \sum_{ij} I_{ij}(i-j)^2$ is the cost function being minimized which penalizes qubits with high MI being far apart. Lighter colors imply higher mutual information.

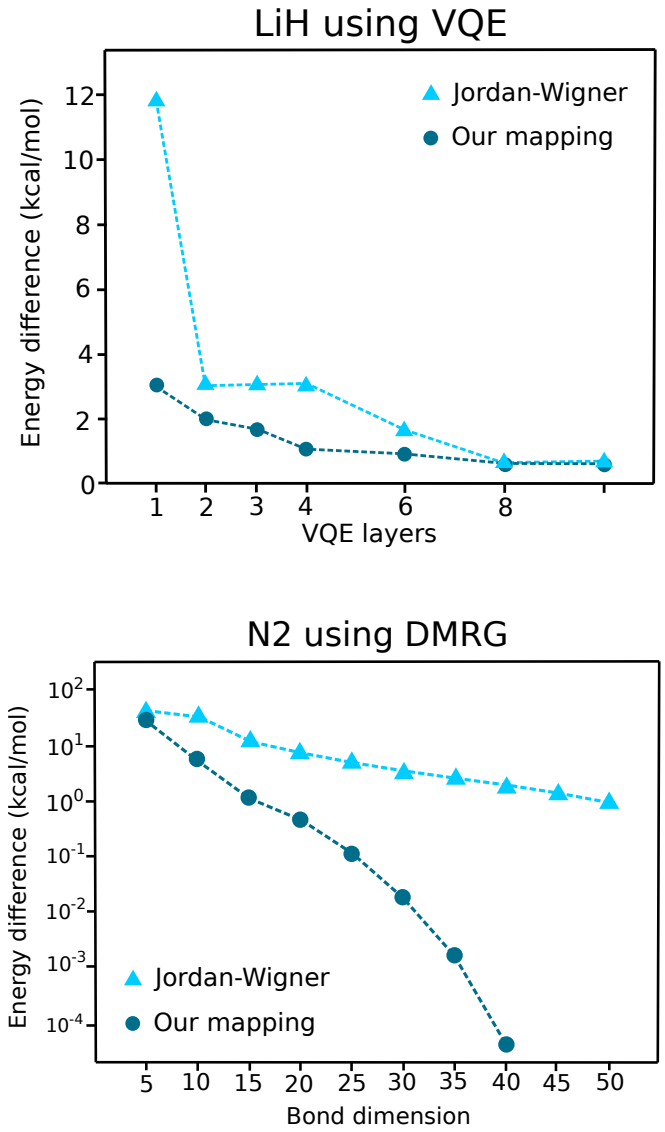

FIGURE 5.7: The VQE performance of our mapping compared to Jordan-Wigner for LiH and the DMRG performance versus Jordan-Wigner for N_2.

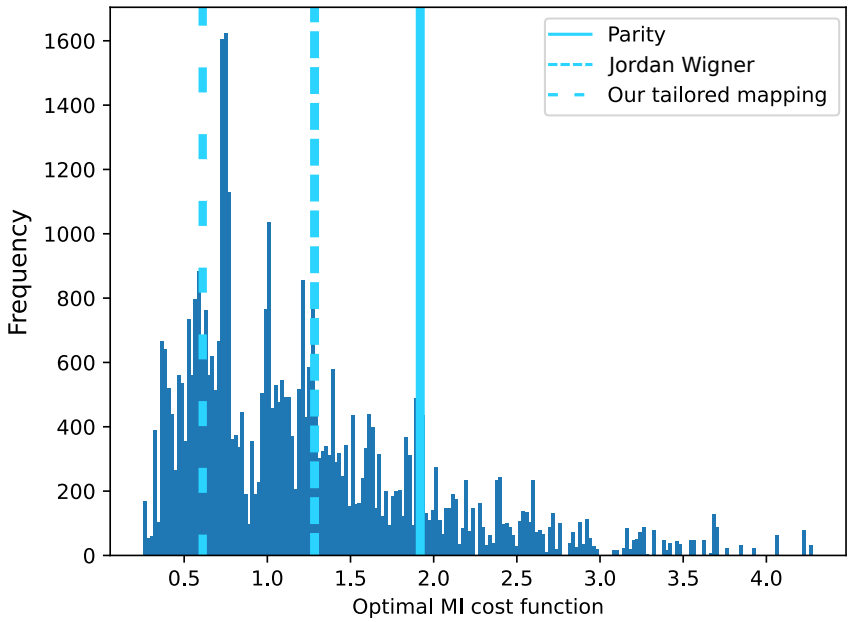

FIGURE 5.8: Results of the brute-force search over all ternary trees and qubit reorderings for hydrogen. The plot shows the mutual information of each mapping, with the some specific mappings marked.

5.3 Qudit Local Quantum Annealing

We now move on to briefly discuss a quantum-inspired algorithm - a classical algorithm which uses ideas and techniques based on quantum theory. Specifically, we consider an algorithm designed to solve the graph colouring problem, an NP-complete problem [169], thus having widespread applications. The work in this chapter is in the patent process, however the details shared in the chapter are based on the version published in Physical Review Applied [170].

Graph colouring is the problem in which one has a graph - nodes connected by edges - and needs to assign a colour to each node such that neighbouring nodes do not share a colour. Sometimes this is not possible for a given number of colours, so one needs to consider more colours, with the minimum number of colours required to colour the graph being known as the **chromatic number**. Given a graph, finding its chromatic number is NP-hard [171], whilst the decision problem of checking if a graph can be coloured with a given number of colours is NP-complete [169]. An example of a graph colouring problem is given in Figure 5.9.

Perhaps one can see that this formulation of problem is very similar to that of the Ising model or other graph problems. As with those problem, we can formulate our optimisation as a minimisation of an objective function, except now subject to the constraint the each variable is an integer bounded between 1 and the number of colours (m) we're testing. We want to penalise when two connected nodes have the same colour, thus one possible objective is the following, where n is the number of nodes, J_{ij} being non-zero mean we have a connection between nodes i and j, c_i is the colouring at site i, and δ is the Kronecker delta function (i.e. 1 if

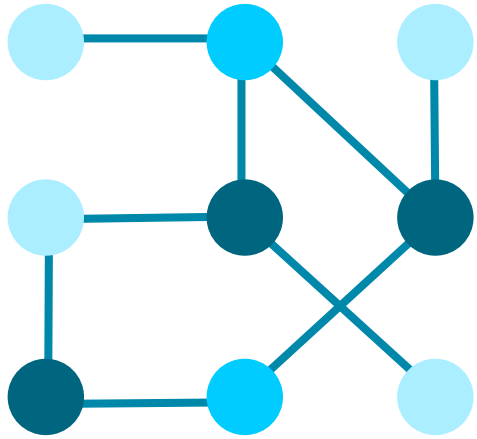

FIGURE 5.9: An example of a graph colouring problem. The goal is to assign a colour to each node such that no two connected nodes share a colour. Shown configuration is a valid solution for this graph, requiring three colours, the smallest number possible for this graph. The solution was obtained by brute-force, possible due to the small size since 9 nodes with 3 colours means only $3^9 = 19683$ possible combinations to test, solved in less than a second using Python.

the inputs are the same and 0 otherwise):

$$\begin{aligned} \min \quad & \sum_{i,j}^{n} J_{ij}(1-\delta(c_i,c_j)) \\ \text{subject to} \quad & c_i \in \{1,2,...,m\} \quad \forall i \end{aligned} \tag{5.24}$$

Whilst in general when discussing quantum concepts we often refer to the simplest quantum unit - the qubit - one can also discuss quantum bits of higher dimensions, known as qudits. Although qubits are known to be universal for quantum computation [172], in some cases qudits may allow for more natural representations. The graph colouring problem appears to be one such example, since if we were considering 3 colours per node we would need 2 qubits per node (giving us 4 choices, 1 of which would be unused), or we could simply use one qutrit (qudit where $d = 3$) per node allowing a much more elegant representation.

The method we consider is to imagine that for each node in our graph we have a qudit of dimension equal to the number of colours that we consider. Based on the method of quantum-inspired annealing by Bowles et. al. [173], we constrain that our solution will always be a product state of such qudits, which is an acceptable assumption since we known that the optimum state should be classical (and thus non-entangled). For the case of c colors, such qudits can be parameterized locally by $c-1$ angles,

such that for each node i we have qudit $|\psi\rangle_i$:

$$|\psi\rangle_i = \begin{bmatrix} \cos(\phi_0^i) \\ \sin(\phi_0^i)\cos(\phi_1^i) \\ \sin(\phi_0^i)\sin(\phi_1^i)\cos(\phi_2^i) \\ \vdots \\ \sin(\phi_0^i)\sin(\phi_1^i)\ldots\sin(\phi_{c-3}^i)\cos(\phi_{c-2}^i) \\ \sin(\phi_0^i)\sin(\phi_1^i)\ldots\sin(\phi_{c-3}^i)\sin(\phi_{c-2}^i) \end{bmatrix}. \tag{5.25}$$

As we consider the system to be in a product state, we can write $|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes \ldots |\psi_n\rangle$, as well as the local probability vectors $\vec{p_i} = \vec{\psi_i}^2$. In order to convert from a quantum state to a possible classical solution, we simply take the most probable projection onto the computational basis, such that we consider the probability p_{ij} for each colour j at node i. We find the highest probability for each node and then assign that colour, giving us a classical solution. This classical solution may not necessarily satisfy the requirement that neighbouring spins are all different, however we can simply replace any duplicate neighbours with a new colour, and then use the new number of colours as an upper bound for the chromatic number.

Although we have a way to convert from our quantum representation back to a valid classical solution, we need a way to optimize over our parameterized qudits. To do so, we use an annealing-style method, such that we begin with a simple Hamiltonian for which we know the ground state, before slowly changing to a more complicated Hamiltonian. One can prove that in the general quantum case if such a change is done slow enough we will remain in the ground state [174], thus solving the original problem at the cost of having to go exponentially slowly in the worst case. In our case, our target energy is not represented by a physical Hamiltonian and

thus the method is dubbed a quantum-inspired classical algorithm rather than something that should actually be performed on a quantum device.

In order to construct our full objective function, we first define our initial energy as:

$$E_{\mathrm{I}}(\psi) = -\sum_i \langle\psi|\hat{L}_i^x|\psi\rangle \tag{5.26}$$

Our target energy is split into two parts, firstly the term that penalises for having the same colour as a neighbour:

$$E_{\mathrm{F}}(\psi) = \sum_{i,j} J_{i,j}\vec{p}_i^T \cdot \vec{p}_j \tag{5.27}$$

Here J_{ij} is a random variable in the range $[1, 1+h]$ where h is a positive hyper-parameter, with the source of randomness helping the optimization to avoid local minima. Then we add an extra term that rewards a uniform spread of probabilities, also helping to avoid local minima, where γ is another positive hyper-parameter:

$$E_{\mathrm{W}}(\psi) = \gamma \sum_i \vec{p}_i^T \cdot \log(\vec{p}_i) \tag{5.28}$$

Our overall energy is therefore the following, with t varying from 0 to 1 as we perform the annealing:

$$E_{\mathrm{total}}(\psi, t) = (1-t)E_{\mathrm{I}}(\psi) + t\big(E_{\mathrm{F}}(\psi) + E_{\mathrm{W}}(\psi)\big) \tag{5.29}$$

By beginning the ground state of the initial objective function ($t = 0$), and then slowly changing to the final function ($t = 1$), rounding at each

step to get a classical solution, we get a solution to our original graph colouring problem that slowly increases, hopefully converging to something near to (if not equal to) the optimal solution. This final algorithm we dub "QdLQA", as it is the qudit extension of the original local quantum annealing algorithm, as well as introducing a number of new heuristics and hyper-parameters. Benchmarking our algorithm on a number of popular datasets we find that in most problems QdLQA matches or outperforms the current state-of-the-art methods, such as the neural network based algorithms PI-SAGE and GNN-1N.

Chapter 6

Conclusion

First of all, thank you for taking the time to read the thesis, I hope you found it interesting and informative. I have certainly learned a lot from working on these projects over the last few years, and I hope that I have been able to convey at least some of it to you.

To summarize the work presented: a variety of optimization techniques, spanning both convex and non-convex optimisation (and often in the form an SDP), have been applied to a range of problems in the field of quantum information theory:

- In the projects related to mutually unbiased bases, we demonstrated that such optimisation techniques are effective at finding bases, as well as providing some insights into the long-standing open problem.

- In the projects relating to the NPA hierarchy and non-locality, we explored many ideas of how one can improve this widely-used optimisation technique, demonstrating advantage over the state-of-the-art in some cases. Many of these ideas (and many more not mentioned

in the thesis) remain in-progress and are currently being explored by both myself and others in the group.

- In the projects related to many-body physics, we demonstrate advantage over several state-of-the-art methods in the fields of quantum chemistry, quantum-inspired algorithms and open-quantum systems. One such project (the quantum-inspired qudit annealing algorithm) is currently in the patent process due to its potential industry applications. Meanwhile, the code for bounding observables over steady-states is now being used for a number of other projects within the group, such as a project detecting Lindbladian phases and a project researching whether one can use measurement data to improve the bounds.

This thesis serves as an example of how, in the modern world of quantum theory, we are often reaching the limit of being able to apply purely-analytical techniques. Much as with many other industries in the last decade, we find ourselves dealing with datasets and problems that are intractable to solve by hand. Whilst analytical solutions will always be incredibly important in proving that an idea works for the small cases, these days much of the novelty of many results requires the use of highly optimized code in order to push the limits of system sizes. We often joke that if you try to publish a result for n qubits, the referee is always going to ask for $n + 1$.

Given that this boundary will most likely be pushed for the rest of time, with each new problem becoming larger and larger, it seems that most of the universe's remaining problems will not be solved with a pen - but with a supercomputer.

Bibliography

1 Miguel Navascués, Stefano Pironio, and Antonio Acín. “A convergent hierarchy of semidefinite programs characterizing the set of quantum correlations”. In: *New Journal of Physics* 10.7 (2008), p. 073013. DOI: 10.1088/1367-2630/10/7/073013.

2 Helge Kragh. “Quantum generations: A history of physics in the twentieth century”. In: (2020). DOI: 10.2307/j.ctv10crfmk.

3 Walter Meissner. “Max Planck, the man and his work”. In: *Science* 113.2926 (1951), pp. 75–81. DOI: 10.1126/science.113.2926.75.

4 Bretislav Friedrich and Dudley Herschbach. “Stern and Gerlach: How a bad cigar helped reorient atomic physics”. In: *Physics Today* 56.12 (2003), pp. 53–59. DOI: 10.1063/1.1650229.

5 Claude Elbaz et al. “Wave-particle duality in Einstein-de Broglie programs”. In: *Journal of Modern Physics* 5.18 (2014), p. 2192. DOI: 10.4236/jmp.2014.518213.

6 David C Cassidy. “Heisenberg, uncertainty and the quantum revolution”. In: *Scientific American* 266.5 (1992), pp. 106–113. DOI: 10.1038/scientificamerican0592-106.

7 Celso Luis Levada, Huemerson Maceti, and Ivan José Lautenschleguer. “Review of the Schrödinger Wave Equation”. In: *IOSR Journal of Applied Chemistry (IOSR-JAC, v. 11, n. 4, p. 1–07* (2018).

8 Margaret D Reid et al. “Colloquium: the Einstein-Podolsky-Rosen paradox: from concepts to applications”. In: *Reviews of Modern Physics* 81.4 (2009), pp. 1727–1751.

9 John S Bell. “On the einstein podolsky rosen paradox”. In: *Physics Physique Fizika* 1.3 (1964), p. 195. DOI: 10.1103/physicsphysiquefizika.1.195.

10 Alain Aspect, Jean Dalibard, and Gérard Roger. “Experimental test of Bell’s inequalities using time-varying analyzers”. In: *Physical review letters* 49.25 (1982), p. 1804. DOI: 10.1103/physrevlett.49.1804.

11 Alain Aspect, John F Clauser, and Anton Zeilinger. "The Nobel Prize in Physics 2022". In: *Nobel Prize Official Website* (2022).

12 Richard P Feynman. "Simulating physics with computers". In: *Feynman and computation*. cRc Press, 2018, pp. 133–153. DOI: 10.1201/9780429500459-11.

13 aslav Brukner. "Quantum causality". In: *Nature Physics* 10.4 (2014), pp. 259–263. DOI: 10.1038/nphys2930.

14 Costantino Budroni et al. "Kochen-specker contextuality". In: *Reviews of Modern Physics* 94.4 (2022), p. 045007. DOI: 10.1103/revmodphys.94.045007.

15 Yudong Cao et al. "Quantum chemistry in the age of quantum computing". In: *Chemical reviews* 119.19 (2019), pp. 10856–10915. DOI: 10.1021/acs.chemrev.8b00803.

16 James E Saal et al. "Materials design and discovery with high-throughput density functional theory: the open quantum materials database (OQMD)". In: *Jom* 65 (2013), pp. 1501–1509. DOI: 10.1007/s11837-013-0755-4.

17 Jens Eisert, Mathis Friesdorf, and Christian Gogolin. “Quantum many-body systems out of equilibrium”. In: *Nature Physics* 11.2 (2015), pp. 124–130. DOI: 10.1038/nphys3215.

18 Gyungmin Cho and Dohun Kim. “Machine learning on quantum experimental data toward solving quantum many-body problems”. In: *Nature Communications* 15.1 (2024), p. 7552. DOI: 10.1038/s41467-024-51932-3.

19 Artem Borin and Dmitry A Abanin. “Approximating power of machine-learning ansatz for quantum many-body states”. In: *Physical Review B* 101.19 (2020), p. 195141. DOI: 10.1103/physrevb.101.195141.

20 Hsin-Yuan Huang et al. “Provably efficient machine learning for quantum many-body problems”. In: *Science* 377.6613 (2022), eabk3333. DOI: 10.1126/science.abk3333.

21 Shi-Ju Ran et al. *Tensor network contractions: methods and applications to quantum many-body systems*. Springer Nature, 2020. DOI: 10.1007/978-3-030-34489-4.

22 Timo Felser, Simone Notarnicola, and Simone Montangero. "Efficient tensor network ansatz for high-dimensional quantum many-body problems". In: *Physical Review Letters* 126.17 (2021), p. 170603. DOI: 10.1103/physrevlett.126.170603.

23 Román Orús. "Tensor networks for complex quantum systems". In: *Nature Reviews Physics* 1.9 (2019), pp. 538–550. DOI: 10.1038/s42254-019-0086-7.

24 Andrew J Daley et al. "Practical quantum advantage in quantum simulation". In: *Nature* 607.7920 (2022), pp. 667–676. DOI: 10.1038/s41586-022-04940-6.

25 Peter W Shor. "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer". In: *SIAM review* 41.2 (1999), pp. 303–332. DOI: 10.1137/s0036144598347011.

26 Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. "A quantum approximate optimization algorithm". In: *arXiv preprint arXiv:1411.4028* (2014).

27 Jacob Biamonte et al. "Quantum machine learning". In: *Nature* 549.7671 (2017), pp. 195–202. DOI: 10.1038/nature23474.

28 Carlos Tavares et al. “Quantum simulation of the ground-state Stark effect in small molecules: a case study using IBM Q”. In: *Soft Computing* 25.9 (2021), pp. 6807–6830. DOI: 10.1007/s00500-020-05492-5.

29 Peter W Shor and John Preskill. “Simple proof of security of the BB84 quantum key distribution protocol”. In: *Physical review letters* 85.2 (2000), p. 441. DOI: 10.1103/physrevlett.85.441.

30 Claude Crépeau, Daniel Gottesman, and Adam Smith. “Secure multi-party quantum computation”. In: *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*. 2002, pp. 643–652. DOI: 10.1145/509907.510000.

31 Miloslav Duek et al. “Quantum identification system”. In: *Physical Review A* 60.1 (1999), p. 149. DOI: 10.1103/physreva.60.149.

32 Marianna Bonanome et al. “Toward protocols for quantum-ensured privacy and secure voting”. In: *Physical Review AAtomic, Molecular, and Optical Physics* 84.2 (2011), p. 022331. DOI: 10.1103/physreva.84.022331.

33 Jason Socrates Bardi. *The calculus wars: Newton, Leibniz, and the greatest mathematical clash of all time*. Hachette UK, 2009. DOI: 10.5860/choice.44-3264.

34 Izrail Moiseevitch Gelfand, Richard A Silverman, et al. *Calculus of variations*. Courier Corporation, 2000. DOI: 10.2307/3611748.

35 George B Dantzig. "Linear programming and extensions". In: *Linear programming and extensions*. Princeton university press, 2016. DOI: 10.7249/r366.

36 Evar D Nering and Albert W Tucker. *Linear programs and related problems*. Elsevier, 1992. DOI: 10.1137/1036161.

37 R Tyrrell Rockafellar. "Conjugate convex functions in optimal control and the calculus of variations". In: *Journal of Mathematical Analysis and Applications* 32.1 (1970), pp. 174–222. DOI: 10.1016/0022-247x(70)90324-0.

38 Harold W Kuhn and Albert W Tucker. "Nonlinear programming". In: *Traces and emergence of nonlinear programming*. Springer, 2013, pp. 247–258. DOI: 10.1007/978-3-0348-0439-4_11.

39 Ralph E Gomory. “Early integer programming”. In: *Operations Research* 50.1 (2002), pp. 78–81. DOI: 10.1287/opre.50.1.78.17793.

40 William J Cook et al. “Combinatorial optimization”. In: *Unpublished manuscript* 10 (1994), pp. 75–93. DOI: 10.1002/(sici)1099-1204(199809/10)11:5<273::aid-jnm304>3.0.co;2-d.

41 Narendra Karmarkar. “A new polynomial-time algorithm for linear programming”. In: *Proceedings of the sixteenth annual ACM symposium on Theory of computing*. 1984, pp. 302–311. DOI: 10.1007/bf02579150.

42 Marguerite Frank, Philip Wolfe, et al. “An algorithm for quadratic programming”. In: *Naval research logistics quarterly* 3.1-2 (1956), pp. 95–110. DOI: 10.1002/nav.3800030109.

43 Lieven Vandenberghe and Stephen Boyd. “Semidefinite programming”. In: *SIAM review* 38.1 (1996), pp. 49–95. DOI: 10.1137/1038003.

44 Dorina Weichert et al. "A review of machine learning for the optimization of production processes". In: *The International Journal of Advanced Manufacturing Technology* 104.5 (2019), pp. 1889–1902. DOI: 10.1007/s00170-019-03988-5.

45 Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004. DOI: 10.1017/cbo9780511804441.

46 Morton Slater. "Lagrange multipliers revisited". In: *Traces and emergence of nonlinear programming*. Springer, 2013, pp. 293–306. DOI: 10.1007/978-3-0348-0439-4_14.

47 Bingsheng He et al. "A generalized primal-dual algorithm with improved convergence condition for saddle point problems". In: *SIAM Journal on Imaging Sciences* 15.3 (2022), pp. 1157–1183. DOI: 10.1137/21m1453463.

48 Barry A Cipra. "The Ising model is NP-complete". In: *SIAM News* 33.6 (2000), pp. 1–3.

49 Maria Serna. “Approximating linear programming is log-space complete for P”. In: *Information Processing Letters* 37.4 (1991), pp. 233–236. DOI: 10.1016/0020-0190(91)90194-m.

50 John A Nelder and Roger Mead. “A simplex method for function minimization”. In: *The computer journal* 7.4 (1965), pp. 308–313. DOI: 10.1093/comjnl/7.4.308.

51 Ron Shamir. “The efficiency of the simplex method: a survey”. In: *Management science* 33.3 (1987), pp. 301–334. DOI: 10.1287/mnsc.33.3.301.

52 Martin Grötschel, László Lovász, and Alexander Schrijver. “The ellipsoid method and its consequences in combinatorial optimization”. In: *Combinatorica* 1 (1981), pp. 169–197. DOI: 10.1007/bf02579273.

53 Ion Necoara, Yu Nesterov, and Francois Glineur. “Linear convergence of first order methods for non-strongly convex optimization”. In: *Mathematical Programming* 175 (2019), pp. 69–107. DOI: 10.1007/s10107-018-1232-1.

54 Brendan Odonoghue et al. "Conic optimization via operator splitting and homogeneous self-dual embedding". In: *Journal of Optimization Theory and Applications* 169 (2016), pp. 1042–1068. DOI: 10.1007/s10957-016-0892-3.

55 Makoto Yamashita, Katsuki Fujisawa, and Masakazu Kojima. "Implementation and evaluation of SDPA 6.0 (semidefinite programming algorithm 6.0)". In: *Optimization Methods and Software* 18.4 (2003), pp. 491–505. DOI: 10.1080/1055678031000118482.

56 Gregorio Malajovich. "On generalized Newton algorithms: quadratic convergence, path-following and error analysis". In: *Theoretical Computer Science* 133.1 (1994), pp. 65–84. DOI: 10.1016/0304-3975(94)00065-4.

57 MOSEK ApS. *The MOSEK optimization toolbox for C++ manual. Version 10.0.*

58 Peter JM Van Laarhoven et al. *Simulated annealing*. Springer, 1987. DOI: 10.1007/978-94-015-7744-1_2.

59 Annu Lambora, Kunal Gupta, and Kriti Chopra. "Genetic algorithm-A literature review". In: *2019 international conference on machine learning, big data, cloud and parallel computing (COMITCon)*. IEEE. 2019, pp. 380–384. DOI: 10.1109/comitcon.2019.8862255.

60 John E Mitchell. "Branch-and-cut algorithms for combinatorial optimization problems". In: *Handbook of applied optimization* 1.1 (2002), pp. 65–77.

61 Diederik P Kingma. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

62 Roger Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2000. DOI: 10.1002/9781118723203.

63 V Rajaraman. "FrontierWorlds First ExaFLOPS Supercomputer". In: *Resonance* 28.4 (2023), pp. 567–576. DOI: 10.1007/s12045-023-1583-7.

64 Jack J Dongarra and Danny C Sorensen. "A fully parallel algorithm for the symmetric eigenvalue problem". In: *SIAM Journal on Scientific and Statistical Computing* 8.2 (1987), s139–s154. DOI: 10.1137/0908018.

65 Olga Ohrimenko, Peter J Stuckey, and Michael Codish. "Propagation= lazy clause generation". In: *Principles and Practice of Constraint Programming–CP 2007: 13th International Conference, CP 2007, Providence, RI, USA, September 23-27, 2007. Proceedings 13*. Springer. 2007, pp. 544–558. DOI: 10.1007/978-3-540-74970-7_39.

66 Flavio Baccari et al. "Verifying the output of quantum optimizers with ground-state energy lower bounds". In: *Physical Review Research* 2.4 (2020), p. 043163. DOI: 10.1103/physrevresearch.2.043163.

67 Julian Schwinger. "Unitary operator bases". In: *Proceedings of the National Academy of Sciences* 46.4 (1960), pp. 570–579.

68 Gelo Noel M Tabia et al. "Bell inequality violations with random mutually unbiased bases". In: *Physical Review A* 106.1 (2022), p. 012209. DOI: 10.1103/physreva.106.012209.

69 Armin Tavakoli et al. "Mutually unbiased bases and symmetric informationally complete measurements in Bell experiments". In: *Science advances* 7.7 (2021), eabc3847. DOI: 10.1126/sciadv.abc3847.

70 Lorena Rebón, C Iemmi, and S Ledesma. "Phase-measurement interferometry as a simulation of optimal quantum-state tomography". In: *Optik* 124.22 (2013), pp. 5548–5552. DOI: 10.1016/j.ijleo.2013.03.171.

71 Hao Yuan, Zheng-Wei Zhou, and Guang-Can Guo. "Quantum state tomography via mutually unbiased measurements in driven cavity QED systems". In: *New Journal of Physics* 18.4 (2016), p. 043013. DOI: 10.1088/1367-2630/18/4/043013.

72 Mhlambululi Mafu et al. "Higher-dimensional orbital-angular-momentum-based quantum key distribution with mutually unbiased bases". In: *Physical Review A* 88.3 (2013), p. 032305. DOI: 10.1103/physreva.88.032305.

73 Thomas Durt et al. "On mutually unbiased bases". In: *International journal of quantum information* 8.04 (2010), pp. 535–640. DOI: 10.1142/s0219749910006502.

74 Daniel McNulty and Stefan Weigert. "Mutually Unbiased Bases in Composite Dimensions–A Review". In: *arXiv preprint arXiv:2410.23997* (2024).

75 Pawe Horodecki, ukasz Rudnicki, and Karol yczkowski. “Five open problems in quantum information”. In: *arXiv preprint arXiv:2002.03233* (2020). DOI: 10.1103/prxquantum.3.010101.

76 Ingemar Bengtsson. “Three ways to look at mutually unbiased bases”. In: *arXiv preprint quant-ph/0610216* (2006). DOI: 10.1063/1.2713445.

77 Bandyopadhyay et al. “A new proof for the existence of mutually unbiased bases”. In: *Algorithmica* 34 (2002), pp. 512–528. DOI: 10.1007/s00453-002-0980-7.

78 Paul Butterley and William Hall. “Numerical evidence for the maximum number of mutually unbiased bases in dimension six”. In: *Physics Letters A* 369.1-2 (2007), pp. 5–8. DOI: 10.1016/j.physleta.2007.04.059.

79 Stephen Brierley and Stefan Weigert. “Maximal sets of mutually unbiased quantum states in dimension 6”. In: *Physical Review A* 78.4 (2008), p. 042312. DOI: 10.1103/physreva.78.042312.

80 Philippe Raynal, Xin Lü, and Berthold-Georg Englert. "Mutually unbiased bases in six dimensions: The four most distant bases". In: *Physical Review AAtomic, Molecular, and Optical Physics* 83.6 (2011), p. 062303. DOI: 10.1103/physreva.83.062303.

81 Luke Mortimer. "Mutually unbiased bases as a commuting polynomial optimisation problem". In: *arXiv preprint arXiv:2308.01879* (2023).

82 Stéphane Jacquet and Sylvain Hallé. "Reformulation of SAT into a polynomial box-constrained optimization problem". In: *Integrated Formal Methods: 16th International Conference, IFM 2020, Lugano, Switzerland, November 16–20, 2020, Proceedings 16*. Springer. 2020, pp. 387–394. DOI: 10.1007/978-3-030-63461-2_21.

83 Monique Laurent and Franz Rendl. "Semidefinite programming and integer programming". In: *Handbooks in Operations Research and Management Science* 12 (2005), pp. 393–514. DOI: 10.1016/s0927-0507(05)12008-8.

84 Sander Gribling and Sven Polak. "Mutually unbiased bases: polynomial optimization and symmetry". In: (2021). DOI: 10.48550/arXiv.2111.05698.

85 Stephen Brierley and Stefan Weigert. "Mutually unbiased bases and semi-definite programming". In: *Journal of Physics: Conference Series*. Vol. 254. IOP Publishing. 2010, p. 012008. DOI: 10.1088/1742-6596/254/1/012008.

86 JL Romero et al. "Structure of the sets of mutually unbiased bases for N qubits". In: *Physical Review A* 72.6 (2005), p. 062310. DOI: 10.1103/physreva.72.062310.

87 Stephen Brierley and Stefan Weigert. "Constructing mutually unbiased bases in dimension six". In: *Physical Review A* 79.5 (2009), p. 052316. DOI: 10.1103/physreva.79.052316.

88 Gaël Guennebaud, Benoît Jacob, et al. *Eigen v3*. 2010.

89 Jean B Lasserre. "Global optimization with polynomials and the problem of moments". In: *SIAM Journal on optimization* 11.3 (2001), pp. 796–817. DOI: 10.1137/s1052623400366802.

90 Brendan O'Donoghue et al. "Conic Optimization via Operator Splitting and Homogeneous Self-Dual Embedding". In: *Journal of Optimization Theory and Applications* 169.3 (June 2016), pp. 1042–1068. DOI: 10.1007/s10957-016-0892-3.

91 Gareth Williams. “Overdetermined systems of linear equations”. In: *The American Mathematical Monthly* 97.6 (1990), pp. 511–513. DOI: 10.2307/2323837.

92 Luke Mortimer. *poly*. 2023.

93 Luke Mortimer. *visual*. 2023.

94 Sander Gribling and Sven Polak. “Mutually unbiased bases: polynomial optimization and symmetry”. In: *Quantum* 8 (2024), p. 1318. DOI: 10.22331/q-2024-04-30-1318.

95 Miguel Navascués, Stefano Pironio, and Antonio Acín. “SDP relaxations for non-commutative polynomial optimization”. In: *Handbook on Semidefinite, Conic and Polynomial Optimization* (2012), pp. 601–634. DOI: 10.1007/978-1-4614-0769-0_21.

96 Maria Prat Colomer et al. “Three numerical approaches to find mutually unbiased bases using Bell inequalities”. In: *Quantum* 6 (2022), p. 778. DOI: 10.22331/q-2022-08-17-778.

97 Nicolas Brunner et al. “Bell nonlocality”. In: *Rev. Mod. Phys.* 86 (2 2014), pp. 419–478. DOI: 10.1103/RevModPhys.86.419.

98 Luke Mortimer. *seesaw*. 2021.

99 Philippe Raynal, Xin Lü, and Berthold-Georg Englert. “Mutually unbiased bases in six dimensions: The four most distant bases”. In: *Phys. Rev. A* 83 (6 2011), p. 062303. DOI: 10.1103/PhysRevA.83.062303.

100 Hiroshi Yamashita, Hiroshi Yabe, and Kouhei Harada. “A primal–dual interior point method for nonlinear semidefinite programming”. In: *Mathematical programming* 135.1 (2012), pp. 89–121. DOI: 10.1007/s10107-011-0449-z.

101 Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. DOI: 10.1017/CBO9780511804441.

102 Luke Mortimer. *nonlinear*. 2021.

Marek ukowski and aslav Brukner. "Quantum
103 non-localityit ain't necessarily so..." In: *Journal of Physics A: Mathematical and Theoretical* 47.42 (2014), p. 424009. DOI: 10.1088/1751-8113/47/42/424009.

John F Clauser et al. "Proposed experiment to test
104 local hidden-variable theories". In: *Physical review letters* 23.15 (1969), p. 880. DOI: 10.1103/physrevlett.23.880.

Boris S Cirel'son. "Quantum generalizations of
105 Bell's inequality". In: *Letters in Mathematical Physics* 4 (1980), pp. 93–100. DOI: 10.1007/bf00417500.

Charles R Johnson and Roger A Horn. *Matrix
106 analysis*. Cambridge university press Cambridge, 1985.

Chamberlain Fong. "Squircular calculations". In:
107 *arXiv preprint arXiv:1604.02174* (2016).

Kartik Krishnan. "Linear programming (LP)
108 approaches to semidefinite programming (SDP) problems". PhD thesis. Ph. D. thesis, Citeseer, 2002. DOI: 10.1090/fic/037/08.

Károly F Pál and Tamás Vértesi. "Maximal
109 violation of a bipartite three-setting, two-outcome Bell inequality using infinite-dimensional quantum systems". In: *Physical Review AAtomic, Molecular, and Optical Physics* 82.2 (2010), p. 022116. DOI: 10.1103/physreva.82.022116.

Davide Angioni. "Polynomial programming with
110 DIGS: a second order cone implementation". In: ().

Luke Mortimer. "Bounding large-scale Bell
111 inequalities". In: *Physical Review A* 111.5 (2025), p. 052442. DOI: 10.1103/PhysRevA.111.052442.

Geoff Gordon and Ryan Tibshirani.
112 "Karush-kuhn-tucker conditions". In: *Optimization* 10.725/36 (2012), p. 725.

Heinz H Bauschke and Jonathan M Borwein.
113 "Dykstra's alternating projection algorithm for two sets". In: *Journal of Approximation Theory* 79.3 (1994), pp. 418–443. DOI: 10.1006/jath.1994.1136.

Kjell Jørgen Overholt. "Extended Aitken
114 acceleration". In: *BIT Numerical Mathematics* 5 (1965), pp. 122–132. DOI: 10.1007/bf01939615.

Eldon Hansen and Merrell Patrick. "A family of
115 root finding methods". In: *Numerische mathematik* 27.3 (1976), pp. 257–269. DOI: 10.1007/bf01396176.

Richard H Byrd et al. "A limited memory
116 algorithm for bound constrained optimization". In: *SIAM Journal on scientific computing* 16.5 (1995), pp. 1190–1208. DOI: 10.2172/204262.

P Mogensen and A Riseth. "Optim: A
117 mathematical optimization package for Julia". In: *Journal of Open Source Software* 3.24 (2018). DOI: 10.21105/joss.00615.

Gerard LG Sleijpen, Henk A Van der Vorst, and
118 Diederik R Fokkema. "BiCGstab (l) and other hybrid Bi-CG methods". In: *Numerical Algorithms* 7 (1994), pp. 75–109. DOI: 10.1007/bf02141261.

Luke Mortimer. *iterativeNPA*. 2023.
119

120 Miguel Navascués et al. “Characterizing finite-dimensional quantum behavior”. In: *Physical Review A* 92.4 (2015), p. 042117. DOI: 10.1103/physreva.92.042117.

121 Howard Carmichael. *An open systems approach to quantum optics: lectures presented at the Université Libre de Bruxelles, October 28 to November 4, 1991*. Vol. 18. Springer Science & Business Media, 2009. DOI: 10.1007/978-3-540-47620-7.

122 Angel Rivas and Susana F Huelga. *Open quantum systems*. Vol. 10. Springer, 2012. DOI: 10.1007/978-3-642-23354-8.

123 Heinz-Peter Breuer and Francesco Petruccione. *The theory of open quantum systems*. Oxford University Press, USA, 2002.

124 Simon Lieu et al. “Symmetry breaking and error correction in open quantum systems”. In: *Physical Review Letters* 125.24 (2020), p. 240405. DOI: 10.1103/physrevlett.125.240405.

125 Gabriel T Landi, Andre L Fonseca de Oliveira, and Efrain Buksman. “Thermodynamic analysis of quantum error-correcting engines”. In: *Physical Review A* 101.4 (2020), p. 042106. DOI: 10.1103/physreva.101.042106.

126 Rosario Fazio et al. “Many-Body Open Quantum Systems”. In: *arXiv:2409.10300* (2024).

127 Riccardo Rota et al. “Critical behavior of dissipative two-dimensional spin lattices”. In: *Physical Review B* 95.13 (2017), p. 134431. ISSN: 2469-9950, 2469-9969. DOI: 10.1103/PhysRevB.95.134431.

128 Jonathan Wei Zhong Lau et al. “Convex Optimization for Nonequilibrium Steady States on a Hybrid Quantum Processor”. In: *Physical Review Letters* 130 (24 2023), p. 240601. DOI: 10.1103/PhysRevLett.130.240601.

129 Daniele Morrone et al. “Estimating Molecular Thermal Averages with the Quantum Equation of Motion and Informationally Complete Measurements”. In: *Entropy* 26.9 (2024), p. 722. ISSN: 1099-4300. DOI: 10.3390/e26090722.

130 Alexandra Nagy and Vincenzo Savona. “Driven-dissipative quantum Monte Carlo method for open quantum systems”. In: *Physical Review A* 97.5 (2018), p. 052129. ISSN: 2469-9926, 2469-9934. DOI: 10.1103/PhysRevA.97.052129.

131 Filippo Vicentini et al. "Variational Neural-Network Ansatz for Steady States in Open Quantum Systems". In: *Physical Review Letters* 122.25 (2019), p. 250503. ISSN: 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.122.250503.

132 Alexandra Nagy and Vincenzo Savona. "Variational Quantum Monte Carlo Method with a Neural-Network Ansatz for Open Quantum Systems". In: *Physical Review Letters* 122.25 (2019), p. 250501. ISSN: 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.122.250501.

133 Dawid A Hryniuk and Marzena H Szymaska. "Tensor-network-based variational Monte Carlo approach to the non-equilibrium steady state of open quantum systems". In: *arXiv:2405.12044* (2024). DOI: 10.22331/q-2024-09-17-1475.

134 Juzar Thingna and Daniel Manzano. "Degenerated Liouvillians and Steady-State Reduced Density Matrices". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 31.7 (2021). arXiv:2101.10236, p. 073114. ISSN: 1054-1500, 1089-7682. DOI: 10.1063/5.0045308.

135 Daniele Amato and Paolo Facchi. "Number of steady states of quantum evolutions". In: *Scientific Reports* 14.1 (2024), p. 14366. DOI: 10.1038/s41598-024-64040-5.

136 Jinkang Guo et al. *Designing open quantum systems with known steady states: Davies generators and beyond*. arXiv:2404.14538. 2024.

137 Leonardo da Silva Souza and Fernando Iemini. *Lindbladian reverse engineering for general non-equilibrium steady states: A scalable null-space approach*. arXiv:2408.05302. 2024.

138 Xiao Mi, et al. "Stable quantum-correlated many-body states through engineered dissipation". In: *Science* 383.6689 (2024), pp. 1332–1337. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.adh9932.

139 Giovanni Ferioli et al. "A non-equilibrium superradiant phase transition in free space". In: *Nature Physics* 19.9 (2023), pp. 1345–1349. ISSN: 1745-2473, 1745-2481. DOI: 10.1038/s41567-023-02064-w.

140 Gian Marcello Andolina et al. *Dicke superradiant enhancement of the heat current in circuit QED*. arXiv.2401.17469. 2024. DOI: 10.48550/ARXIV.2401.17469.

141 Vittorio Gorini, Andrzej Kossakowski, and Ennackal Chandy George Sudarshan. "Completely positive dynamical semigroups of N-level systems". In: *Journal of Mathematical Physics* 17.5 (1976), pp. 821–825. DOI: 10.1063/1.522979.

142 Goran Lindblad. "On the generators of quantum dynamical semigroups". In: *Communications in Mathematical Physics* 48 (1976), pp. 119–130. DOI: 10.1007/bf01608499.

143 Daniel Manzano. "A short introduction to the Lindblad master equation". In: *AIP advances* 10.2 (2020). DOI: 10.1063/1.5115323.

144 Simon B Jäger et al. "Lindblad master equations for quantum systems coupled to dissipative bosonic modes". In: *Physical Review Letters* 129.6 (2022), p. 063601. DOI: 10.1103/physrevlett.129.063601.

145 Daniel Jaschke, Simone Montangero, and Lincoln D Carr. "One-dimensional many-body entangled open quantum systems with tensor network methods". In: *Quantum Science and Technology* 4.1 (2018), p. 013001. DOI: 10.1088/2058-9565/aae724.

146 Miguel Navascués, Stefano Pironio, and Antonio Acín. "Bounding the set of quantum correlations". In: *Physical Review Letters* 98.1 (2007), p. 010401. DOI: 10.1103/physrevlett.98.010401.

147 Stefano Pironio, Miguel Navascués, and Antonio Acín. "Convergent Relaxations of Polynomial Optimization Problems with Noncommuting Variables". In: *SIAM Journal on Optimization* 20.5 (2010), pp. 2157–2180. DOI: 10.1137/090760155.

148 Dian Wu et al. "Variational benchmarks for quantum many-body problems". In: *arXiv:2302.04919* (2023). DOI: 10.1126/science.adg9774.

149 Erling D Andersen and Knud D Andersen. "The MOSEK interior point optimizer for linear programming: an implementation of the homogeneous algorithm". In: *High performance optimization*. Springer, 2000, pp. 197–232. DOI: 10.1007/978-1-4757-3216-0_8.

150 Ioannis Kogias et al. "Hierarchy of steering criteria based on moments for all bipartite quantum systems". In: *Physical Review Letters* 115.21 (2015), p. 210401. DOI: 10.1103/physrevlett.115.210401.

151 Alejandro Pozas-Kerstjens et al. “Bounding the sets of classical and quantum correlations in networks”. In: *Physical Review Letters* 123.14 (2019), p. 140503. DOI: 10.1103/physrevlett.123.140503.

152 Jie Wang et al. “Certifying ground-state properties of many-body systems”. In: *Physical Review X* 14.3 (2024), p. 031006. DOI: 10.1103/physrevx.14.031006.

153 Patrick P Hofer et al. “Markovian master equations for quantum thermal machines: local versus global approach”. In: *New Journal of Physics* 19.12 (2017), p. 123037. DOI: 10.1088/1367-2630/aa964f.

154 Shishir Khandelwal et al. “Critical heat current for operating an entanglement engine”. In: *New Journal of Physics* 22.7 (2020), p. 073039. DOI: 10.1088/1367-2630/ab9983.

155 Bruno Bertini et al. “Finite-temperature transport in one-dimensional quantum lattice models”. In: *Reviews of Modern Physics* 93 (2 2021), p. 025003. DOI: 10.1103/RevModPhys.93.025003.

156 Borja Requena et al. “Certificates of quantum many-body properties assisted by machine learning”. In: *Physical Review Research* 5 (1 2023), p. 013097. DOI: 10.1103/PhysRevResearch.5.013097.

157 Benjamin A Cordier et al. “Biology and medicine in the landscape of quantum advantages”. In: *Journal of the Royal Society Interface* 19.196 (2022), p. 20220541. DOI: 10.1098/rsif.2022.0541.

158 AJ Freeman. “Materials by design and the exciting role of quantum computation/simulation”. In: *Journal of computational and applied mathematics* 149.1 (2002), pp. 27–56. DOI: 10.1016/s0377-0427(02)00519-8.

159 Teodor Parella-Dilmé et al. “Reducing Entanglement with Physically Inspired Fermion-To-Qubit Mappings”. In: *PRX Quantum* 5.3 (2024), p. 030333. DOI: 10.1103/prxquantum.5.030333.

160 Charles Derby et al. “Compact fermion to qubit mappings”. In: *Physical Review B* 104.3 (2021), p. 035118. DOI: 10.1103/physrevb.104.035118.

161 Aaron Miller et al. “Bonsai algorithm: Grow your own fermion-to-qubit mappings”. In: *PRX Quantum* 4.3 (2023), p. 030314. DOI: 10.1103/prxquantum.4.030314.

162 Pascual Jordan and Eugene Paul Wigner. *Über das paulische äquivalenzverbot*. Springer, 1993. DOI: 10.1007/978-3-662-02781-3_9.

163 Sergey B Bravyi and Alexei Yu Kitaev. “Fermionic quantum computation”. In: *Annals of Physics* 298.1 (2002), pp. 210–226. DOI: 10.1006/aphy.2002.6254.

164 Jules Tilly et al. “The variational quantum eigensolver: a review of methods and best practices”. In: *Physics Reports* 986 (2022), pp. 1–128. DOI: 10.1016/j.physrep.2022.08.003.

165 Samson Wang et al. “Noise-induced barren plateaus in variational quantum algorithms”. In: *Nature communications* 12.1 (2021), p. 6961. DOI: 10.1038/s41467-021-27045-6.

166 Thinh Viet Le and Vassilis Kekatos. “Variational Quantum Eigensolver with Constraints (VQEC): Solving Constrained Optimization Problems via VQE”. In: *arXiv preprint arXiv:2311.08502* (2023). DOI: 10.1103/physreva.110.022430.

167 Ulrich Schollwöck. “The density-matrix renormalization group”. In: *Reviews of modern physics* 77.1 (2005), pp. 259–315.

168 Joonho Lee et al. “Generalized unitary coupled cluster wave functions for quantum computation”. In: *Journal of chemical theory and computation* 15.1 (2018), pp. 311–324.

169 Jason I Brown. “The complexity of generalized graph colorings”. In: *Discrete Applied Mathematics* 69.3 (1996), pp. 257–270. DOI: 10.1016/0166-218x(96)00096-0.

170 David Jansen et al. “Qudit-inspired optimization for graph coloring”. In: *Physical Review Applied* 22.6 (2024), p. 064002.

171 Adrian Kosowski and Krzysztof Manuszewski. “Classical coloring of graphs”. In: *Contemporary Mathematics* 352 (2004), pp. 1–20. DOI: 10.1090/conm/352/06369.

172 Sergey Bravyi and Alexei Kitaev. “Universal quantum computation with ideal Clifford gates and noisy ancillas”. In: *Physical Review AAtomic, Molecular, and Optical Physics* 71.2 (2005), p. 022316. DOI: 10.1103/physreva.71.022316.

173 Joseph Bowles et al. “Quadratic unconstrained binary optimization via quantum-inspired annealing”. In: *Physical Review Applied* 18.3 (2022), p. 034016. DOI: 10.1103/physrevapplied.18.034016.

174 Satoshi Morita and Hidetoshi Nishimori. “Mathematical foundation of quantum annealing”. In: *Journal of Mathematical Physics* 49.12 (2008). DOI: 10.1063/1.2995837.