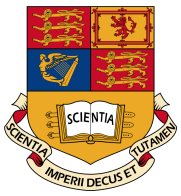

Imperial College London

Imperial College London

Department of Mathematics

Post-Quantum Cryptography: Algebraic Construction of Long Range Interactions and Spin Chains

Author:
Eleanor Kneip

Supervisor:
Prof. Florian Mintert

Co-Supervisor:
Dr. Ryan Barnett

Submitted in partial fulfillment of the requirements for the degree

MSci Mathematics with a Year Abroad

June 10, 2024

Abstract

In light of the last 30 years of groundbreaking developments in quantum technologies, the National Institute of Standards and Technology called researchers to propose potential cryptographic protocols that are quantum-resistant. Since then four protocols have been selected, three of which are lattice-based. Now it remains to verify whether these protocols are indeed resistant to quantum attacks. This dissertation provides an insight into the research in quantum-safe cryptography, from the foundations of quantum mechanics and classical cryptography, to the depths of dynamic invariants and long range spin chains. One unsolved problem in lattice theory and closely tied to cryptography, is the shortest vector problem. We illustrate how this problem can be mapped to a problem Hamiltonian with long range interactions. This gives us one method that solves the shortest vector problem: realise the problem Hamiltonian via continuous-time quantum computing and observe the first excited state. This has been successfully implemented using adiabatic quantum computing and in this dissertation we experiment with various boundary conditions on dynamic invariants, that can avoid the adiabatic constraint of slow time and provide a favourable scaling. While missing the desired target of creating any given long range interaction, we provide a set that, given a maximum range of interaction, scales linearly in the number of spins.

Acknowledgements

I would like to thank Florian Mintert for supervising this project and giving invaluable feedback and guidance throughout its development. I would also like to thank Ryan Barnett for teaching Quantum Mechanics 2 and introducing me to many of the concepts developed further throughout this project. Finally I would like to thank Modesto Ruiz for answering many of my questions and providing key resources to explore ideas further.

I acknowledge the use of IBM Quantum services for this work. The views expressed are my own, and do not reflect the official policy or position of IBM or the IBM Quantum team.

Plagiarism statement

The work contained in this thesis is my own work unless otherwise stated.

Signature: Eleanor Kneip
Date: June 10, 2024

Contents

Chapter 1

Introduction

Cryptography is an integral aspect of the digital world and in recent years, the need for secure, standardised post-quantum cryptography has increasingly become a greater concern [1]. As a result of the growing research in quantum computation, several of our widely-used cryptographic protocols are no longer secure against quantum attacks. Shor's algorithm is one such threat [2]. Thus in 2016, the National Institute of Standards and Technology (NIST) announced a competition to find new cryptographic protocols that are quantum-resistant [3]. This dissertation will provide an insight into both the worlds of quantum computing and cryptography, and illustrate how they can be used together effectively.

In Chapter 2, we begin this dissertation with an introduction to quantum computation. As such, we work from the foundations of quantum mechanics to build the Deutsch model of quantum computation [4]. Reflecting on the classical theory of computation, we draw several similarities that have inspired the field of quantum computation. This model has been the catalyst in the development of quantum algorithms. Indeed it has led to many ground-breaking algorithms [5, 6, 7], that have demonstrated the computational advantage a fault-resilient quantum computer could provide. In order to incentivise the need for post-quantum cryptography, we give a detailed analysis of Shor's infamous factoring algorithm [2]. While the current quantum computers of the NISQ (Noisy Intermediate-Scale Quantum) era are not yet capable of implementing Shor's algorithm effectively [8], this algorithm still poses the threat of "Store Now, Decrypt Later" attacks, in which encrypted data and public keys are harvested and stored with the intention of decrypting them once the means are available [9]. To complete this chapter, we introduce continuous-time quantum computing. This is an alternative model that has been shown to be equivalent to the Deutsch model [10], and has also given a new perspective to quantum computing, leading to the development of new algorithms solving several NP-complete problems [11, 12]. In particular, we will give two examples of continuous-time quantum computation: adiabatic quantum computation (AQC) and quantum computation by dynamic invariants (QCDI).

Before we make use of quantum computing in cryptography, in Chapter 3 we give a broad overview of current cryptographic methods, from hash functions to symmetric and asymmetric ciphers, providing for each its vulnerabilities to quantum attacks [13]. This will set us up well to introduce the learning with errors cryptosystem, which inspired two of the four finalist protocols selected by NIST following their competition [14]. We will find that the security of this cryptosystem relies on lattice problems. As a result we study how the aforementioned continuous-time quantum computation model can be employed for a given lattice problem, known as the shortest vector problem (SVP).

Both AQC and QCDI have been applied to solving SVP [15, 16, 17], and in Chapter 4, we study an approach using QCDI in more depth. First we will observe the effect of using dynamic

invariants to realise a given spin chain. Then we present an improvement to this construction, as suggested in [17]. Using computational means, we can experiment with various boundary conditions to construct invariants with desired properties [18]. In particular, the shortest vector problem can be mapped to a Hamiltonian with long range interactions, which thus incites the challenge of realising corresponding dynamic invariants with these long range interactions [15]. The greatest portion of my individual contribution is found here, where several examples are given, studying their properties in depth, as well as techniques to determine their constraints. Finding these examples required the development of a Python package that can manipulate operators and complete Lie algebras efficiently. We give some computational insight to the creation of this package in the dissertation and provide a demonstration of it in the repository itself [18].

Finally in Chapter 5 we give a summary of the results found, discussing limitations reached, from scaling to interaction range, as well as areas with potential for more research.

Chapter 2

Introduction to Quantum Computation

"Hilbert space is a big place."

- Carlton Caves [19]

Before we tackle how quantum computers can be used in cryptography, we define some essential concepts in quantum computation. We begin with a short recap of classical computation and then introduce the Deutsch model of quantum computation, including the analysis of a simple example to demonstrate the ideas presented. A more complex example is given in Section 2.2, where we study Shor's algorithm in depth. We complete this chapter with an introduction to continuous time quantum computing, which will be of particular use throughout the rest of the dissertation.

2.1 Definitions and Examples

In classical computation, we store information in an abstract unit known as a "bit", which takes the value 0 or 1. Classical computers can manipulate these bits to produce useful outcomes by applying boolean functions, $f : \mathcal{F}_2^n \to \mathcal{F}_2^m$. Some examples are found in Figure 2.1.

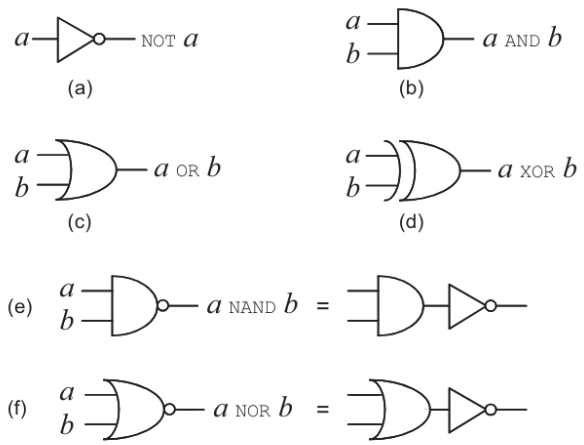

Figure 2.1: Selection of examples of classical logic gates [20].

In particular, we can define a universal set of logic gates, such that any boolean function, or logic gate, can be reduced to a sequence of these universal gates. An example of a classical

universal set is given in the following theorem [20].

Theorem 2.1. *Any classical logic gate can be decomposed into a sequence of NAND gates.*

This is a fundamental result in information theory and in fact we find has a quantum analogue. To formulate this, we introduce the Deutsch model of quantum computation [4].

A Hilbert space, $\mathcal{H}$, is a complex inner product space which is complete with respect to the norm induced by the inner product. Acting in this space, we can define 5 fundamental postulates of quantum mechanics [21]:

1. *State Postulate:* The state of a quantum system can be described by a non-zero vector in $\mathcal{H}$.

2. *Observable Postulate:* A measurable quantity is described by a Hermitian operator on $\mathcal{H}$.

3. *Born Rule Postulate:* The possible outcomes of a measurement of an observable, A, are the eigenvalues of the corresponding operator, $\hat{A}$, and the probabilities are given by

$$\mathbb{P}(a_j) = \langle\psi|\hat{P}_j|\psi\rangle, \tag{2.1}$$

   where $|\psi\rangle$ is the state immediately before measurement and $\hat{P}_j$ is the projection operator of the eigenspace of a_j.

4. *State-Collapse Postulate:* After measurement, the state collapses to the relevant eigenspace, or explicitly,

$$\frac{\hat{P}_j|\psi\rangle}{\sqrt{\langle\psi|\hat{P}_j|\psi\rangle}}. \tag{2.2}$$

5. *Time-Evolution Postulate:* A state subject to the Hamiltonian $\hat{\mathcal{H}}$, evolves in time according to the time-dependent Schrödinger equation,

$$i\hbar|\dot{\psi}\rangle = \hat{\mathcal{H}}|\psi\rangle. \tag{2.3}$$

We note that in Equation (2.1) and Equation (2.2), we have assumed the eigenspace of $\hat{A}$ is discrete, as is typical in quantum computation.

Thus a quantum computer can store information in a pure quantum state, known as a "qubit", with unit length [20]. Taking the Hilbert space $\mathcal{H}$ with computational basis $\{|0\rangle, |1\rangle\}$, we can write a general state in Dirac notation or standard vector notation equivalently:

$$\alpha|0\rangle + \beta|1\rangle \quad \text{or} \quad \begin{pmatrix}\alpha\\ \beta\end{pmatrix}.$$

This can be extended to a system of n qubits, using 2^n complex numbers to define the amplitudes of the corresponding computational basis states. In addition we use the Kronecker tensor product to define multiple qubit systems, conventionally denoted:

$$|0\rangle \otimes |0\rangle \equiv |00\rangle.$$

We can then manipulate the system according to Postulate 5, using unitary transformations, which we call quantum logic gates. A crucial consequence of this is that unitary transformations are norm preserving, so a state vector with unit normal will have unit norm throughout the

quantum circuit. For example, the quantum NOT gate on a single qubit can be represented as follows:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad \Longrightarrow \quad X(\alpha|0\rangle + \beta|1\rangle) = \alpha|1\rangle + \beta|0\rangle.$$

Finally, we can carry out measurements of the system using Postulate 3, where the projection used defines the measurement apparatus. We can, without loss of generality, take the possible projections to be the computational basis states, since any other projection could be achieved by first carrying out a change of basis using unitary transformations and then measuring with respect to the computational basis.

To consolidate these ideas, we construct the quantum circuit as shown in Figure 2.2, also known as the Bell circuit. The results are displayed in histograms in Figure 2.3.

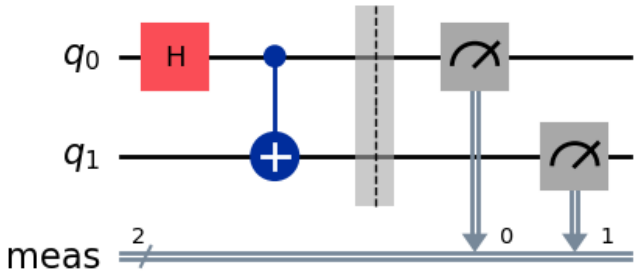

Figure 2.2: Example of a quantum circuit, creating a Bell state [18].

Conventionally, the initial state is taken to be $|00\rangle$, thus the state immediately before measurement is,

$$|00\rangle \mapsto \frac{|00\rangle + |11\rangle}{\sqrt{2}},$$

which is an example of an entangled Bell state. Next using the Born rule, Equation (2.1), we expect the following outcomes,

$$\mathbb{P}(|00\rangle) = \frac{1}{2} \quad \mathbb{P}(|01\rangle) = 0 \quad \mathbb{P}(|10\rangle) = 0 \quad \mathbb{P}(|11\rangle) = \frac{1}{2},$$

which can be seen in Figure 2.3.

This circuit was implemented and carried out on a quantum simulator (AerSimulator), as well as an actual quantum computer (IBM Osaka). When compared we see a small error in the probability amplitudes of the actual quantum computer, demonstrating the effects of noise.

Equipped with a general quantum logic gate represented by a unitary transformation, we can now generalise Theorem 2.1 in the quantum case in the following [20],

Theorem 2.2. *Any multiple qubit logic gate, U, can be represented to arbitrary accuracy by a concatenation of the $CNOT$ and single qubit gates.*

For example, a universal set of quantum logic gates is $\{CNOT, T, H\}$ where

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{pmatrix} \qquad H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{2.4}$$

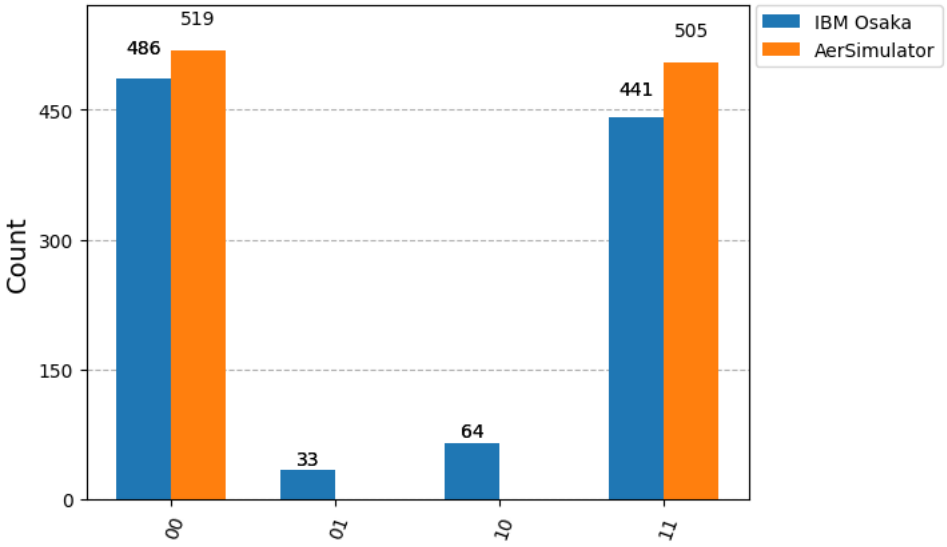

Figure 2.3: Results from the Bell circuit in Figure 2.2, carried out on AerSimulator and IBM Osaka [18].

It is also useful to introduce the following set of logic gates, which includes the NOT gate discussed earlier, known as the spin-1/2 Pauli gates,

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \qquad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{2.5}$$

These gates, along with the identity, I, form a basis for all single qubit logic gates, i.e. any logic gate can be written as linear combination of these gates.

This completes the discussion of the Deutsch model of computation, which will next be applied in Section 2.2 to Shor's infamous algorithm.

2.2 Shor's Algorithm

In this section we explicitly analyse the details of Shor's algorithm [2]. The goal of Shor's algorithm is in fact to find the period of a function, i.e. given a function f, such that $f(x+a) = f(x)$, find a. It can be shown that this is equivalent to factoring, i.e. given $N = pq$ for prime numbers p, q, find the values p, q.

2.2.1 Rabin-Miller Algorithm

The Rabin-Miller algorithm is a probabilistic algorithm for testing primality of an integer N, and can be summarised as follows [22]:

1. Choose $a \in \{2, 3, \ldots, N-1\}$ uniformly at random.
2. Compute $d := \gcd(a, N)$, for example using Euclid's algorithm.
3. (a) $d > 1$: d is a non-trivial divisor of N, success!
   (b) $d = 1$: compute smallest integer r such that

$$a^r \equiv 1 \bmod N. \tag{2.6}$$

4. (a) r is odd: declare failure, return to 1.
   (b) r is even: decompose as follows

$$a^r - 1 = (a^{\frac{r}{2}} - 1)(a^{\frac{r}{2}} + 1) \equiv 0 \bmod N.$$

   Then,
   i. $a^{\frac{r}{2}} + 1 \equiv 0 \bmod N$: declare failure and return to 1.
   ii. $a^{\frac{r}{2}} - 1 \equiv 0 \bmod N$: then $a^{\frac{r}{2}} \equiv 1 \bmod N$, which contradicts how we chose r, so this case is impossible.
   iii. Neither (i) nor (ii): must have $(a^{\frac{r}{2}} + 1)|p$ and $(a^{\frac{r}{2}} - 1)|q$, so can find $p = \gcd(a^{\frac{r}{2}} + 1, N)$, success!

We remark the following: there are several instances of success/failure before the end of the algorithm, for example at 3a, 4a and 4(b)i. However it can be shown for each case, that it is unlikely to happen. Overall then, we have the following theorem:

Theorem 2.3. *The probability of success in one run-through the the Rabin-Miller algorithm is greater than or equal to* $\frac{3}{4}$.

Thus if we wish to have $\mathbb{P}(success) \geq 1 - \epsilon$, we need $O(ln(\epsilon))$ executions of the algorithm. Indeed, after N runs, $\mathbb{P}(failure) = (\frac{1}{4})^N$, thus we need $\epsilon \geq (\frac{1}{4})^N$ which implies $N = O(ln(\epsilon))$. The proof of Theorem 2.3 was shown by M. Rabin [22].

The hardest step computationally in the Rabin-Miller algorithm is 3b. Indeed, the best classical algorithms to find the period r are superpolynomial in complexity. Thus this is the step Shor's algorithm sets out to replace. To do this we can define a function $f(x)$ such that

$$f(x) = a^x \bmod N \tag{2.7}$$

for a and N defined in the Rabin-Miller algorithm above. Then f must be periodic, with period r defined in Equation (2.6), since

$$f(x + r) = a^{x+r} \bmod N = a^x \bmod N = f(x).$$

Given f, it is the goal of Shor's algorithm to find the period r.

2.2.2 Period-Finding Algorithm

As suggested in Section 2.2.1, we are given a periodic function f, and wish to find the period r. Following the work of M. Nielsen and I. Chuang [23], first we note that we can represent integers as a string of qubits using binary, for example,

$$x = x_0 + 2x_1 + 4x_2 + \cdots + 2^{n-1}x_{n-1} \tag{2.8}$$

$$\implies |x\rangle = |x_0 x_1 x_2 \cdots x_{n-1}\rangle. \tag{2.9}$$

Now since our quantum computer cannot store infinite qubits, we must truncate at M qubits. We can choose M, such that $M > N^2$ and for simplicity take $M = 2^m$ for some $m \in \mathbb{N}$. This means we will be working with m qubits, making up the first quantum register, to represent any non-negative integer less than M. In addition we need m ancilla ("helper") qubits, making up the second quantum register, giving a total of $2m$ qubits.

Our next assumption is that f can be implemented on a quantum computer using the unitary transformation, U_f, defined on $2m$ qubits as

$$U_f|xy\rangle = |x\rangle \otimes |y + f(x)\rangle, \tag{2.10}$$

where the addition is taken modulo M and y is an integer also represented using Equation (2.9).

Finally we define the Quantum Fourier Transform (QFT), on m qubits as

$$QFT|x\rangle = \frac{1}{\sqrt{M}} \sum_{y=0}^{M-1} \exp\left(\frac{2\pi i x \cdot y}{M}\right)|y\rangle. \tag{2.11}$$

Now we are ready to construct and analyse the quantum circuit. Using the transformations introduced above, as well as the Hadamard gate, H, defined in Equation (2.4), the whole circuit can be depicted as in Figure 2.4 for $M = 2^3$.

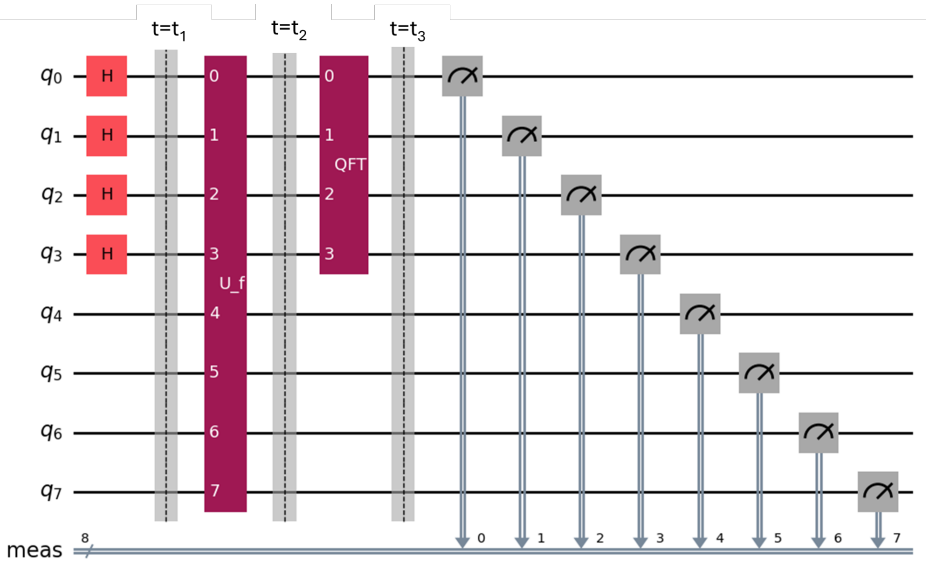

Figure 2.4: Shor's quantum circuit for the case $M = 2^3$.

We start at $t = t_0$ and let each barrier be carried out at $t = t_1, t_2, \ldots$ to aid analysis. Thus at $t = t_0$, the state of the system is:

$$|\psi_0\rangle = |00\cdots0\rangle = |0\rangle^{\otimes m} \otimes |0\rangle^{\otimes m}.$$

Next we apply the string of Hadamard gates, making the state at $t = t_1$,

$$\begin{aligned} |\psi_1\rangle &= (H^{\otimes m} \otimes I^{\otimes m})|0\rangle^{\otimes m} \otimes |0\rangle^{\otimes m} \\ &= \frac{1}{\sqrt{M}} \sum_{x=0}^{M-1} |x\rangle \otimes |0\rangle^{\otimes m}, \end{aligned}$$

where we have used the qubit representation of integers in binary, as in Equation (2.9), and I is the 2×2 identity. Now we can apply the unitary U_f given in Equation (2.10), so at $t = t_2$,

$$|\psi_2\rangle = \frac{1}{\sqrt{M}} \sum_{x=0}^{M-1} U_f\Big(|x\rangle \otimes |0\rangle^{\otimes m}\Big) \tag{2.12}$$

$$= \frac{1}{\sqrt{M}} \sum_{x=0}^{M-1} |x\rangle \otimes |f(x)\rangle \tag{2.13}$$

$$= \frac{1}{\sqrt{M}} \sum_{z_0=0}^{r-1} \sum_{j=0}^{j_{max}(z_0)} |z_0 + jr\rangle \otimes |f(z_0)\rangle, \tag{2.14}$$

where in Equation (2.14), we have made use of the periodicity of f and we take j_{max} such that the sums equate. In the simple case $r|M$, we take $j_{max} = \frac{M}{r} - 1$, however since this cannot be guaranteed, we instead take the z_0-dependent $j_{max}(z_0)$ with the upper bound $j_{max}(z_0) \leq \frac{M}{r} - 1$ for all z_0.

Next at $t = t_3$, we apply the Quantum Fourier Transform, given in Equation (2.11),

$$\begin{aligned}
|\psi_2\rangle &= \frac{1}{\sqrt{M}} \sum_{z_0=0}^{r-1} \sum_{j=0}^{j_{max}(z_0)} (QFT \otimes I^{\otimes m})|z_0 + jr\rangle \otimes |f(z_0)\rangle \\
&= \frac{1}{M} \sum_{z_0=0}^{r-1} \sum_{j=0}^{j_{max}(z_0)} \sum_{y=0}^{M-1} \exp\left(\frac{2\pi i(z_0 + jr)\cdot y}{M}\right)|y\rangle \otimes |f(z_0)\rangle \\
&= \sum_{y=0}^{M-1} \sum_{z_0=0}^{r-1} \frac{1}{M} e^{\frac{2\pi i}{M} z_0 y} \Big\{ \sum_{j=0}^{j_{max}(z_0)} e^{\frac{2\pi i}{M} jry} \Big\} |y\rangle \otimes |f(z_0)\rangle.
\end{aligned}$$

Now we are ready to measure the first register of qubits, i.e. the first m qubits. To find the probability of collapsing the first register to the state $|y\rangle$, we must apply the projector $P_y \otimes I^{\otimes m}$ where $P_y = |y\rangle\langle y|$, then

$$\mathbb{P}(|y\rangle) = \langle \psi_2 | P_y \otimes I^{\otimes m} | \psi_2 \rangle.$$

Thus,

$$P_y \otimes I^{\otimes m}|\psi_2\rangle = \sum_{z_0=0}^{r-1} \frac{1}{M} e^{\frac{2\pi i}{M} z_0 y} \Big\{ \sum_{j=0}^{j_{max}(z_0)} e^{\frac{2\pi i}{M} jry} \Big\} |y\rangle \otimes |f(z_0)\rangle \tag{2.15}$$

$$\implies \mathbb{P}(|y\rangle) = \frac{1}{M^2} \sum_{z_0, z_0'=0}^{r-1} e^{-\frac{2\pi i}{M} z_0 y} e^{\frac{2\pi i}{M} z_0' y} \sum_{j,j'=0}^{j_{max}(z_0)} e^{-\frac{2\pi i}{M} jry} e^{\frac{2\pi i}{M} j'ry} \langle f(z_0)|f(z_0')\rangle \tag{2.16}$$

$$= \frac{1}{M^2} \sum_{z_0=0}^{r-1} \sum_{j,j'=0}^{j_{max}(z_0)} e^{\frac{2\pi i}{M} ry(j'-j)} \tag{2.17}$$

$$= \frac{1}{M^2} \sum_{z_0=0}^{r-1} \Big| \sum_{j=0}^{j_{max}(z_0)} e^{\frac{2\pi i}{M} jry} \Big|^2 \tag{2.18}$$

where in Equation (2.17) we have used that $\langle f(z_0)|f(z_0')\rangle = \delta_{z_0, z_0'}$ since f is assumed different for each z_0.

Next we analyse this result classically. First it is useful to once again consider the special case where $r|M$. In this case, we can make the following conclusion, for $k = 0, 1, \ldots, r-1$:

$$\begin{aligned}
\mathbb{P}(y = k \cdot \frac{M}{r}) &= \frac{1}{M^2} \sum_{z_0=0}^{r-1} \Big| \sum_{j=0}^{\frac{M}{r}-1} e^{2\pi i jk} \Big|^2 \\
&= \frac{1}{M^2} \sum_{z_0=0}^{r-1} \Big| \sum_{j=0}^{\frac{M}{r}-1} 1 |^2 \\
&= \frac{1}{M^2} r \Big(\frac{M}{r}\Big)^2 \\
&= \frac{1}{r}.
\end{aligned}$$

Now since there are exactly r such ys, we must have $\mathbb{P}(y \neq k \cdot \frac{M}{r}) = 0$. Thus we will observe with certainty $y = k \cdot \frac{M}{r}$ for some $k = 0, 1, \ldots, r-1$. To find r, we can now divide y by M to give $\frac{y}{M} = \frac{k}{r}$ and simplify the fraction, giving the desired r on the denominator. This will give the period, since we want r to be the smallest such integer satisfying $f(x+r) = f(x)$. Overall this means we can find r with certainty from a single measurement!

Unfortunately, the case $r \nmid M$ is not quite as simple. However, we can show that a measurement will yield a result close to $k \cdot \frac{M}{r}$. The following theorem specifies this [23]:

Theorem 2.4. *Let* $I := \cup_{k=0}^{r-1} I_k$ *where* $I_k := [k \cdot \frac{M}{r} - \frac{1}{2}, k \cdot \frac{M}{r} + \frac{1}{2}$. *Then at measurement* $\mathbb{P}(|y\rangle \in I) \geq \frac{2}{5}$.

From this we have that with probability $\geq \frac{2}{5}$,

$$\left| y - k \cdot \frac{M}{r} \right| \leq \frac{1}{2} \tag{2.19}$$

$$\iff \left| \frac{y}{M} - \frac{k}{r} \right| \leq \frac{1}{2M} < \frac{1}{r^2}, \tag{2.20}$$

where in Equation (2.20), we have used that $M > N^2 > r^2$. Now we need one last theorem to put everything together [23],

Theorem 2.5. *If* $\gcd(k, r) = 1$ *and* $\left| \frac{y}{M} - \frac{k}{r} \right| \leq \frac{1}{2M} \leq \frac{1}{r^2}$, *then all solutions* $\frac{k}{r}$ *satisfying the inequality are convergents given by the continuous fraction expansion of* $\frac{y}{M}$.

We define the continuous fraction expansion and convergents as follows. For a fraction $\frac{y}{M}$, the continuous fraction expansion is,

$$\frac{y}{M} = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\ddots + \frac{1}{a_n}}}} =: [a_0; a_1; a_2; \ldots; a_n].$$

Then the set of convergents is given by $\{[a_0], [a_0; a_1], \ldots, [a_0; a_1; a_2; \ldots; a_n]\}$. The continuous fraction expansion can also be defined for irrationals, and provide an infinite set of convergents, however for our case we only need the finite, rational case.

In light of Theorem 2.4 and Theorem 2.5, we can analyse the result in the following steps.

1. Run Shor's circuit, as in Figure 2.4, and yield y.
2. Calculate the continuous fraction expansion of $\frac{y}{M}$.
3. Compute all convergents $\frac{s}{t} =: (s, t)$.
4. Test for each (s, t), whether $a^t \equiv 1 \mod N$ or equivalently whether $f(t) \equiv 1 \mod N$.
5. If there are several such (s, t), then $r = \min(s)$. If there are no such s, then declare failure and return to Step 1.

Thus we have an algorithm that determines the period r.

To find the probability of success of Shor's algorithm, we consider each step individually. Firstly, it can be shown that if k is chosen uniformly at random from $\{0, 1, \ldots, r-1\}$, then

$$\mathbb{P}(\gcd(k, r) = 1) \geq \frac{1}{4 \log(\log(r))} \tag{2.21}$$

$$> \frac{1}{4 \log(m)}, \tag{2.22}$$

where in Equation (2.22), we have used that $\log(r) < \frac{1}{2}\log(M) = \frac{1}{2}m$ [2]. This allows us to use Theorem 2.5. Then to use Theorem 2.4, we have $\mathbb{P}(|y\rangle \in I) \geq \frac{2}{5}$. All together then,

$$\mathbb{P}(success) \geq O\Big(\frac{2}{5} \cdot \frac{1}{4\log(m)}\Big) = O\Big(\frac{1}{\log(m)}\Big).$$

Now we can amplify this probability by running the algorithm several times. If we would like a success probability of $1-\epsilon$, we will need T runs, where

$$\Big(1 - O\Big(\frac{1}{\log(m)}\Big)\Big)^T \leq 1 - \epsilon$$
$$\implies T \geq |\log(\epsilon)| \log(m).$$

Noting that the hardest part of the classical side of the Rabin-Miller algorithm is using Euclid's algorithm to find the gcd, which is $O(m^3)$, we have that the total complexity of the Rabin-Miller algorithm, with probability amplification, is

$$\begin{aligned} O(m^3 T) &= O(m^3 \log(m) |\log(\epsilon)|) \\ &= O((\log(N))^3 \log(\log(N)) |\log(\epsilon)|). \end{aligned}$$

This concludes the analysis of Shor's algorithm, which has been successfully carried out on IBM's quantum processors to factorise $N = 21$ [24]. However for larger N, the algorithm requires a large number of qubits and in addition the effects of noise become a much greater issue. Error correction methods can be used to mitigate noise, but require further qubits.

2.3 Continuous-Time Quantum Computing

The Deutsch model of quantum computation described above has been widely successful and given rise to a surge of research in quantum algorithms since the 1980s. Some notable examples include the Deutsch-Jozsa algorithm for solving a particular oracle-problem with exponential speed-up in comparison to the classical solution [5]; Grover's algorithm for unstructured search giving quadratic speed-up [6] and Shor's algorithm for period-finding with applications to factorisation and the discrete logarithm also with exponential speed-up [2]. In particular, this model uses unitary transformations to act as quantum logic gates that determine how the state evolves in time. However an alternative model, known as continuous time quantum computation, allows the state to evolve in time continuously, according to the dynamics induced by a time-dependent Hamiltonian $H(t)$.

2.3.1 Adiabatic Quantum Computing

To study adiabatic quantum computing, we must first introduce some core concepts in adiabatic theory. This section will follow the analysis given by R. Barnett in his lecture series in quantum mechanics [25] and from which we will draw parallels in Section 2.3.2. Given a time-dependent Hamiltonian $\hat{\mathcal{H}}(t)$, the adiabatic limit assumes the time-dependence is "slow". We will define what we mean by "slow" in detail in the following analysis.

We wish to derive how a given state evolves in time, subject to a time-dependent Hamiltonian $\hat{\mathcal{H}}$. To do this, we need to understand how the eigenstates of $\hat{\mathcal{H}}$ evolve in time. Let the instantaneous eigenstates and eigenenergies be denoted

$$\hat{\mathcal{H}}(t)|\phi_k(t)\rangle = \epsilon_k(t)|\phi_k(t)\rangle$$

and suppose our initial state is in the non-degenerate eigenstate

$$|\psi(0)\rangle = |\phi_n(0)\rangle.$$

Then suppose $|\psi(t)\rangle$ solves the Schrödinger Equation (2.3), and since $\hat{\mathcal{H}}$ is Hermitian, we can expand,

$$|\psi(t)\rangle = \sum_k a_k(t) e^{i\gamma_k(t)} e^{-i\alpha_k(t)} |\phi_k(t)\rangle$$

with the initial condition $a_n(0) = 1$ and $a_k(0) = 0$ for $k \neq n$. We have also defined the dynamical phase, $\alpha_n(t)$, and the berry phase, $\gamma_n(t)$, for convenience, as follows,

$$\alpha_n(t) = \int_0^t \frac{\epsilon_n(t')}{\hbar} dt' \qquad \gamma_n(t) = \int_0^t i\langle\phi_n(t')|\dot{\phi}_n(t')\rangle dt'.$$

Next, we sub everything into the Schrödinger equation and apply $\langle\phi_m|$:

$$i\hbar\frac{\partial}{\partial t}\sum_k a_k(t) e^{i\gamma_k(t)} e^{-i\alpha_k(t)} |\phi_k(t)\rangle = \hat{\mathcal{H}}\sum_k a_k(t) e^{i\gamma_k(t)} e^{-i\alpha_k(t)} |\phi_k(t)\rangle$$

$$i\hbar e^{i\gamma_m(t)} e^{-i\alpha_m(t)} \Big(\dot{a}_m(t) + a_m(t) i\dot{\gamma}_m(t) - a_m(t) i\dot{\alpha}_m(t)\Big) + i\hbar \sum_k a_k(t) e^{i\gamma_k(t)} e^{-i\alpha_k(t)} \langle\phi_m(t)|\dot{\phi}_k(t)\rangle)$$
$$= \epsilon_m(t) a_m(t) e^{i\gamma_m(t)} e^{-i\alpha_m(t)}$$

$$i\hbar\Big(\dot{a}_m(t) - a_m(t)\langle\phi_m(t)|\dot{\phi}_m(t)\rangle - a_m(t) i\frac{\epsilon_m(t)}{\hbar}\Big) + \sum_k a_k(t) e^{i(\gamma_k(t)-\gamma_m(t))} e^{-i(\alpha_k(t)-\alpha_m(t))} \langle\phi_m(t)|\dot{\phi}_k(t)\rangle$$
$$= \epsilon_m(t) a_m(t)$$

$$\implies \dot{a}_m(t) = -\sum_{k\neq m} a_k(t) e^{i(\gamma_k(t)-\gamma_m(t))} e^{-i(\alpha_k(t)-\alpha_m(t))} \langle\phi_m(t)|\dot{\phi}_k(t)\rangle$$

From here we can approximate to first order,

$$\begin{aligned} a_m(t) &= a_m(0) - \int_0^t \sum_{k\neq m} a_k(t') e^{i(\gamma_k(t')-\gamma_m(t'))} e^{-i(\alpha_k(t')-\alpha_m(t'))} \langle\phi_m(t')|\dot{\phi}_k(t')\rangle dt' \\ &\approx a_m(0) - \int_0^t \sum_{k\neq m} a_k(0) e^{i(\gamma_k(t')-\gamma_m(t'))} e^{-i(\alpha_k(t')-\alpha_m(t'))} \langle\phi_m(t')|\dot{\phi}_k(t')\rangle dt'. \end{aligned}$$

Thus we recover $a_n(t) = a_0(t) = 1$ and for the remaining amplitudes,

$$a_m(t) - a_m(0) \approx -\int_0^t e^{i(\gamma_n(t')-\gamma_m(t'))} e^{-i(\alpha_n(t')-\alpha_m(t'))} \langle\phi_m(t')|\dot{\phi}_n(t')\rangle dt'. \tag{2.23}$$

This integral can be further approximated, but first we note,

$$\langle\phi_m(t)|\frac{\partial}{\partial t}\Big(\hat{\mathcal{H}}(t)|\phi_n(t)\rangle\Big) = \langle\phi_m(t)|\frac{\partial}{\partial t}\Big(\epsilon_n(t)|\phi_n(t)\rangle\Big)$$
$$\langle\phi_m(t)|\dot{\hat{\mathcal{H}}}(t)|\phi_n(t)\rangle + \langle\phi_m(t)|\hat{\mathcal{H}}(t)|\dot{\phi}_n(t)\rangle = \langle\phi_m(t)|\dot{\epsilon}_n(t)|\phi_n(t)\rangle + \langle\phi_m(t)|\epsilon_n(t)|\dot{\phi}_n(t)\rangle$$
$$\implies \langle\phi_m(t)|\dot{\phi}_n(t)\rangle = \frac{\langle\phi_m(t)|\dot{\hat{\mathcal{H}}}(t)|\phi_n(t)\rangle}{\epsilon_n(t) - \epsilon_m(t)}.$$

Returning to the integral Equation (2.23), we use asymptotic methods to make the following approximation, which is allowed since the term $e^{-i(\alpha_n(t')-\alpha_m(t'))}$ oscillates on a much faster time-scale than the remaining terms in the integrand,

$$\begin{aligned} a_m(t) &\approx -\Big[\frac{\hbar}{\epsilon_n(t)-\epsilon_m(t)} e^{i(\gamma_n(t')-\gamma_m(t'))} e^{-i(\alpha_n(t')-\alpha_m(t'))} \langle\phi_m(t')|\dot{\phi}_n(t')\rangle\Big]_{t'=0}^{t} \\ &= -\Big[\frac{\hbar}{(\epsilon_n(t)-\epsilon_m(t))^2} \langle\phi_m(t)|\dot{\hat{\mathcal{H}}}(t)|\phi_n(t)\rangle e^{i(\gamma_n(t')-\gamma_m(t'))} e^{-i(\alpha_n(t')-\alpha_m(t'))}\Big]_{t'=0}^{t}. \end{aligned}$$

Putting everything together, we can write the desired state $|\psi(t)\rangle$, with correction as,

$$|\psi(t)\rangle = e^{i\gamma_n(t)} e^{-i\alpha_n(t)} \Big(|\phi_n(t)\rangle - \sum_{k\neq n} a_m(t)|\phi_0\rangle\Big),$$

from which we conclude that starting in state $|\phi_n(0)\rangle$, we remain in this instantaneous eigenstate and obtain a phase, so long as the correction is small. This gives us the condition on t,

$$\Big|\frac{\hbar\langle\phi_m(t)|\dot{\hat{\mathcal{H}}}(t)|\phi_n(t)\rangle}{(\epsilon_n(t)-\epsilon_m(t))^2}\Big| \ll 1 \qquad \forall m \neq n$$

or alternatively, t must be slow compared to $(\epsilon_n - \epsilon_m)/\hbar$ for $m \neq n$. This concludes the study of adiabatic theory.

Overall, we have shown that if we start in an instantaneous eigenstate of $\hat{\mathcal{H}}$, then we will remain in this eigenstate so long as we propagate time slowly enough. When applied to quantum computing, this means we can use adiabatic theory to prepare the quantum state needed to solve a given problem. For example, suppose the solution to a problem is encoded in the ground state of the Hamiltonian $\hat{\mathcal{H}}_P$, which we call the problem Hamiltonian. Then initialising the system in the ground state of an initial Hamiltonian, $\hat{\mathcal{H}}_0$, we can define the time-dependent Hamiltonian,

$$\hat{\mathcal{H}}(t) = \lambda(t)\hat{\mathcal{H}}_P + (1-\lambda(t))\hat{\mathcal{H}}_0$$

for a suitable monotonically increasing $\lambda(t) \in [0,1]$. Now we can initialise the system in the ground state of $\hat{\mathcal{H}}_0$ and evolve the system according to $\hat{\mathcal{H}}(t)$. Then, using the adiabatic results from above, we know that at $t = T$, we will be in the ground state of the problem Hamiltonian, $\hat{\mathcal{H}}_P$. The intricacies of finding suitable $\hat{\mathcal{H}}_0$ and $\hat{\mathcal{H}}_P$, as well as designing $\lambda(t)$, is an expanding field that has shown much potential. Indeed, many algorithms using the Deutsch model of quantum computation have also been carried out using adiabatic quantum computation, such as Deutsch-Jozsa and Grover [11]. It has also been applied directly to problems in its own right, such as the boolean satisfiability problems (k-SAT) [11], which is NP-complete for $k \geq 3$, as well as randomly generated instances of an NP-complete problem [12].

2.3.2 Quantum Computing by Dynamic Invariants

Having discussed adiabatic quantum computing, we find the main drawback is the need to propagate time slowly. This can be quite a strong constraint, thus quantum computing by dynamic invariants is a similar model of continuous-time quantum computing that finds a way around this limitation.

Before we start, we will from here onwards be using a rescaled Hamiltonian, $\hat{\mathcal{H}}$, such that $\hbar$ is removed from Schrödinger's Equation (2.3).

To describe this model, we first define the dynamic invariant of a Hamiltonian, $\hat{\mathcal{H}}$. This is a Hermitian operator, $\hat{\mathcal{I}}(t)$, such that the following holds [26],

$$\frac{\partial}{\partial t}\hat{\mathcal{I}}(t) + i[\hat{\mathcal{H}}(t), \hat{\mathcal{I}}(t)] = 0. \tag{2.24}$$

This implies $\hat{\mathcal{I}}$ is a constant of motion, hence we have,

$$\frac{d}{dt}\langle\hat{\mathcal{I}}\rangle = 0.$$

Additionally, we can derive an important consequence of dynamic invariants: let $|\phi\rangle$ be a solution to the Schrödinger equation. Then we find using Equation (2.24),

$$\begin{aligned}
\dot{\hat{\mathcal{I}}}(t)|\phi\rangle &= -i[\hat{\mathcal{H}}(t), \hat{\mathcal{I}}(t)]|\phi\rangle \\
\dot{\hat{\mathcal{I}}}(t)|\phi\rangle - i\hat{\mathcal{I}}(t)\hat{\mathcal{H}}(t)|\phi\rangle &= -i\hat{\mathcal{H}}(t)\hat{\mathcal{I}}(t)|\phi\rangle \\
\dot{\hat{\mathcal{I}}}(t)|\phi\rangle + \hat{\mathcal{I}}(t)|\dot{\phi}\rangle &= -i\hat{\mathcal{H}}(t)\hat{\mathcal{I}}(t)|\phi\rangle \\
\implies i\frac{\partial}{\partial t}\Big(\hat{\mathcal{I}}(t)|\phi\rangle\Big) &= \hat{\mathcal{H}}(t)\Big(\hat{\mathcal{I}}(t)|\phi\rangle\Big).
\end{aligned}$$

Thus we have shown that if $|\phi\rangle$ solves the Schrödinger equation, then $\hat{\mathcal{I}}(t)|\phi\rangle$ does too. Then since solutions are unique up to a phase, we have

$$\hat{\mathcal{I}}(t)|\phi\rangle = \lambda|\phi\rangle.$$

In other words we can seek solutions to the Schrödinger equation, in eigenstates of invariants.

Following the work of M. Sarandy, E. Duzzioni and R. Serra [27], we now let $\{|\phi_k(t)\rangle\}_k$ be an orthonormal eigenbasis of $\hat{\mathcal{I}}(t)$ with corresponding eigenvalues $\lambda_m(t)$, which is guaranteed to exist by the Hermiticity of $\hat{\mathcal{I}}(t)$. We assume for simplicity that these are all non-degenerate. Then if $|\psi\rangle$ is a solution to the Schrödinger equation, we can expand it in this basis:

$$|\psi(t)\rangle = \sum_k c_k(t)|\phi_k(t)\rangle.$$

Using a similar method as in the adiabatic case, we insert this into the Schrödinger equation, and apply $\langle\phi_m|$,

$$i\sum_k \frac{\partial}{\partial t}\Big(c_k(t)|\phi_k(t)\rangle\Big) = \sum_k c_k(t)\hat{\mathcal{H}}(t)|\phi_k(t)\rangle \tag{2.25}$$

$$i\dot{c}_m(t) + i\sum_k c_k(t)\langle\phi_m(t)|\dot{\phi}_k(t)\rangle = \sum_k c_k(t)\langle\phi_m(t)|\hat{\mathcal{H}}(t)|\phi_k(t)\rangle \tag{2.26}$$

$$\dot{c}_m(t) = -\sum_k c_k(t)\Big(\langle\phi_m(t)|\dot{\phi}_k(t)\rangle + i\langle\phi_m(t)|\hat{\mathcal{H}}(t)|\phi_k(t)\rangle\Big). \tag{2.27}$$

Alternatively, we also have

$$\begin{aligned}
\frac{\partial}{\partial t}\Big(\hat{\mathcal{I}}(t)|\phi_k(t)\rangle\Big) &= \frac{\partial}{\partial t}\Big(\lambda_k(t)|\phi_k(t)\rangle\Big) \\
\dot{\hat{\mathcal{I}}}(t)|\phi_k(t)\rangle + \hat{\mathcal{I}}(t)|\dot{\phi}_k(t)\rangle &= \dot{\lambda}_k(t)|\phi_k(t)\rangle + \lambda_k(t)|\dot{\phi}_k(t)\rangle \\
-i(\lambda_k(t) - \lambda_m(t))\langle\phi_m(t)|\hat{\mathcal{H}}(t)|\phi_k(t)\rangle + \lambda_m\langle\phi_m|\dot{\phi})k\rangle &= \dot{\lambda}_k(t)\delta_{m,k} + \lambda_k(t)\langle\phi_m(t)|\dot{\phi}_k(t)\rangle \\
(\lambda_m(t) - \lambda_k(t))(i\langle\phi_m(t)|\hat{\mathcal{H}}(t)|\phi_k(t)\rangle + \langle\phi_m(t)|\dot{\phi}_k(t)\rangle) &= \dot{\lambda}_k(t)\delta_{m,k}.
\end{aligned}$$

Thus we can make the following conclusions,

$$m = k: \qquad \dot{\lambda}_k(t) = 0 \tag{2.28}$$

$$m \neq k: \qquad -i\langle\phi_m(t)|\hat{\mathcal{H}}(t)|\phi_k(t)\rangle = \langle\phi_m(t)|\dot{\phi}_k(t)\rangle. \tag{2.29}$$

Since $\hat{\mathcal{I}}$ is a constant of motion, having constant eigenvalues is no surprise. The $m \neq k$ case can be used to derive how $c_k(t)$ evolves by subbing back into Equation (2.27).

$$\dot{c}_m(t) = -c_m(t)\Big(\langle\phi_m(t)|\dot{\phi}_m(t)\rangle + i\langle\phi_m(t)|\hat{\mathcal{H}}(t)|\phi_m(t)\rangle\Big)$$
$$c_m(t) = c_m(0)\exp\Big(-\int_0^t \langle\phi_m(t')|\dot{\phi}_m(t')\rangle + i\langle\phi_m(t')|\hat{\mathcal{H}}(t')|\phi_m(t')\rangle dt'\Big),$$

where the exponential term is known as the Lewis-Riesenfeld phase [26]. Thus we have shown that if we initialise the system in an eigenstate of $\hat{\mathcal{I}}(0)$, such that $c_n(0) = 1$ and $c_k(0) = 0$ for $k \neq n$, then we must remain in a state proportional to the evolution of this eigenstate. It is crucial to note that there has been no time constraint assumed, as in the adiabatic case, so we can evolve time as fast as we wish and are guaranteed to remain in the instantaneous eigenstate of $\hat{\mathcal{I}}(t)$.

Before we relate these results back to quantum computing, we consider how $\hat{\mathcal{I}}(t)$ evolves in time. First we introduce the time-evolution operator, which is an operator, denoted $\hat{\mathcal{U}}$, satisfying the following evolution equation,

$$i\frac{\partial}{\partial t}\hat{\mathcal{U}}(t) = \hat{\mathcal{H}}(t)\hat{\mathcal{U}}(t) \qquad \hat{\mathcal{U}}(0) = I. \tag{2.30}$$

The key consequence of this is that we can multiply the above differential equation by a state $|\psi(0)\rangle$ to reproduce the Schrödinger equation:

$$i\frac{\partial}{\partial t}\Big(\hat{\mathcal{U}}(t)|\psi(0)\rangle\Big) = \hat{\mathcal{H}}(t)\Big(\hat{\mathcal{U}}(t)|\psi(0)\rangle\Big).$$

Now since the solution to Schrödinger's equation with initial condition $|\psi(0)\rangle$ is unique, we must have

$$|\psi(t)\rangle = \hat{\mathcal{U}}(t)|\psi(0)\rangle.$$

Thus we have a general equation for the evolution of a state with initial condition $|\psi(0)\rangle$. The explicit form of $\hat{\mathcal{U}}(t)$ can be expressed

$$\hat{\mathcal{U}}(t) = T_t e^{-i\hbar\int_0^t \hat{\mathcal{H}}(t')dt'}, \tag{2.31}$$

where T_t is the time-ordering operator, which orders a sequence of operators in reverse chronological order [25]. We can similarly use $\hat{\mathcal{U}}(t)$ to define the time evolution of $\hat{\mathcal{I}}(t)$,

$$\hat{\mathcal{I}}(t) = \hat{\mathcal{U}}(t)\hat{\mathcal{I}}(0)\hat{\mathcal{U}}(t)^\dagger. \tag{2.32}$$

Indeed, a quick calculation shows Equation (2.24) holds for all t:

$$\frac{\partial}{\partial t}\hat{\mathcal{I}}(t) + i[\hat{\mathcal{H}}(t), \hat{\mathcal{I}}(t)] = \dot{\hat{\mathcal{U}}}(t)\hat{\mathcal{I}}(0)\hat{\mathcal{U}}(t)^\dagger + \hat{\mathcal{U}}(t)\hat{\mathcal{I}}(0)\dot{\hat{\mathcal{U}}}(t)^\dagger + i(\hat{\mathcal{H}}(t)\hat{\mathcal{U}}(t)\hat{\mathcal{I}}(0)\hat{\mathcal{U}}(t)^\dagger - \hat{\mathcal{U}}(t)\hat{\mathcal{I}}(0)\hat{\mathcal{U}}(t)^\dagger\hat{\mathcal{H}}(t)) \tag{2.33}$$
$$= \dot{\hat{\mathcal{U}}}(t)\hat{\mathcal{I}}(0)\hat{\mathcal{U}}(t)^\dagger + \hat{\mathcal{U}}(t)\hat{\mathcal{I}}(0)\dot{\hat{\mathcal{U}}}(t)^\dagger - \dot{\hat{\mathcal{U}}}(t)\hat{\mathcal{I}}(0)\hat{\mathcal{U}}(t)^\dagger - \hat{\mathcal{U}}(t)\hat{\mathcal{I}}(0)\dot{\hat{\mathcal{U}}}(t)^\dagger \tag{2.34}$$
$$= 0, \tag{2.35}$$

where in Equation (2.34), we have made use of Equation (2.30).

Now that we have an understanding of the dynamic invariant, we can explore how to use it in quantum computation to solve a given problem. Originally invariants were used to solve for the eigenstate of $\hat{\mathcal{I}}$, given a Hamiltonian, which thus solves Schrödinger's equation [26]. Similarly to

the adiabatic case, we will instead reverse-engineer a suitable Hamiltonian, $\hat{\mathcal{H}}(t)$, given a desired final state.

In this case, we begin by choosing an initial invariant, $\hat{\mathcal{I}}_0$, such that the ground state is non-degenerate and easy to prepare experimentally. Then we define a suitable final invariant, $\hat{\mathcal{I}}_T$, such that the solution to the problem is encoded in the (non-degenerate) ground state. Next we can interpolate the invariant to find $\hat{\mathcal{I}}(t)$ with the boundary conditions $\hat{\mathcal{I}}(0) = \hat{\mathcal{I}}_0$, $\hat{\mathcal{I}}(T) = \hat{\mathcal{I}}_T$. This can be done using a suitable unitary operator, as in Equation (2.32), to ensure there is no level-crossing. Finally we can use $\hat{\mathcal{I}}(t)$ to determine the corresponding Hamiltonian, $\hat{\mathcal{H}}(t)$, from Equation (2.24). Using the above analysis, we know that the initial ground state, subject to $\hat{\mathcal{H}}(t)$, must evolve to the desired final ground state at $t = T$. Indeed this method can be used to perform optimal control and has been successfully used to overcome the adiabatic time constraint [28, 29].

It is interesting to note, as is indicated in [27], that if we take the above model of quantum computation and impose the adiabatic limit, i.e. that time evolves slowly, then Equation (2.24) becomes,

$$[\hat{\mathcal{H}}(t), \hat{\mathcal{I}}(t)] \approx 0$$

and thus $\hat{\mathcal{H}}(t)$ and $\hat{\mathcal{I}}(t)$ share a set of common eigenstates for all t, which will give exactly the dynamics in adiabatic quantum computation.

Alternatively, if we interpolate $\hat{\mathcal{I}}(t)$ as in Equation (2.32) and instead discretise the time-evolution operator, $\hat{\mathcal{U}}(t)$, we can form a sequence of unitary transformations, corresponding to quantum logic gates, as described in the Deutsch model of quantum computation.

This concludes the study of quantum computation models and will be applied in Chapter 4 to ensuring the cryptographic protocols presented in Chapter 3 are indeed quantum-safe.

Chapter 3

Quantum Safe Cryptography

"If computers that you build are quantum,
Then spies of all factions will want 'em.
Our codes will all fail,
And they'll read our email,
Till we've crypto that's quantum, and daunt 'em."

- Jennifer and Peter Shor [30]

"To read our E-mail, how mean
of the spies and their quantum machine;
Be comforted though,
they do not yet know
how to factorize twelve or fifteen."

- Volker Strassen [31]

While current quantum computers are not yet capable of implementing Shor's algorithm, as discussed in detail in Section 2.2, on a large enough scale to break current cryptographic protocols [8], our data is still at risk of Store Now, Decrypt Later (SNDL) attacks [9]. In such an attack, the encrypted data, as well as any public keys, are stored with the purpose of decrypting it in the future when a reliable quantum computer can break it. Thus it is crucial to develop quantum resistant technology and standardise it as quickly as possible. As a result, in 2016 the National Institute of Standards and Technology (NIST) released a call for proposals of quantum-resistant cryptographic protocols to replace the current quantum-vulnerable protocols [3]. Of those submitted, NIST then released four finalists in 2022, three of which are lattice-based [14].

This section begins with an introduction to classical cryptography, before tackling the intricacies of quantum-safe cryptography, studying in detail the shortest vector problem, which underpins two of the proposed protocols.

3.1 Foundations in Classical Cryptography

Cryptography, derived from the Greek words "kryptós" and "graphein" meaning "hidden writing", dates back to 2000 BC with the Egyptian practice of hieroglyphics, which were only understood by elite few [32]. Since then, the first classical ciphers were seen used by Julius Caesar (100-44 BC), who did not trust his messengers with information intended for his generals,

and has now become an integral part of the digital world.

In practice, cryptographic protocols take a message, known as the plaintext, and scramble it with the use of a key into ciphertext. This can then be sent to the intended recipient who has the means to decrypt the ciphertext back into plaintext, with the knowledge that any "middle-man" eavesdropping is unable to decrypt the message and equally has not tampered with it. The algorithm used to encrypt or decrypt the message is then known as a cipher.

Classical ciphers were developed before the innovation of modern computers. The classic example is the Caesar cipher, which replaces every letter in the message by one n after it in the alphabet, e.g. $e \mapsto h$ when $n = 3$. In this case, n is the key only known to those trusted with the message [33]. Classical ciphers such as these, are susceptible to brute-force attacks, and thus are insecure with the use of computers. As a result, modern ciphers were developed, which are based on the principles of mathematics and designed to be difficult to decrypt using classical computers. For example, as will be discussed in greater detail in the discussion of asymmetric keys, RSA depends on large number factorisation, a famously difficult problem from a computational standpoint.

Now with the fast-developing field of quantum computing, we once again find a threat to the security of our ciphers. As a result research has turned to quantum-safe cryptography, using more complex problems that are resistant to quantum attacks, and quantum cryptography, based on principles of quantum physics and instead transmitting information as qubits. We will be focusing our discussion on quantum-safe cryptography, however quantum cryptography is also an inspired field from aiding symmetric key cryptography with quantum key distribution [34] to fully quantum communication protocols [35].

The four key components underpinning classical, modern and quantum-safe cryptographic systems can be broken down into four main goals [32]:

1. *Authenticity:* verifies the source of the message, e.g. using digital certificates.
2. *Confidentiality:* guarantees the message can only be understood by those permitted to know it, e.g. using symmetric/asymmetric keys.
3. *Data Integrity:* ensures data has not been tampered with, e.g. using hashing.
4. *Non-Repudiation:* sender becomes accountable, the origin of the message must be undisputed, e.g. using digital signatures.

We will use the rest of this section to introduce three well-developed areas of cryptographic protocols, highlighting where relevant how they are vulnerable to quantum attacks. These short explorations will follow the content given by [13]. In particular it will be useful to introduce Alice and Bob, who are trying to communicate securely, as well as Eve, who is attempting to eavesdrop on them.

Cryptographic Hash Functions

Hash functions provide a cheap method to ensure authenticity and data integrity. They can be defined as a function $h : \Omega \to \{0,1\}^n$ for finite $n \in \mathbb{N}$, where Ω is the space of all finite-length strings. Thus any message can be mapped to a length-n binary string, known as a digest. We require that the function is deterministic, and difficult to reverse. Since $|\Omega| \gg 2^n$, h must be many-to-one, however we can construct h, with an "avalanche effect". This ensures that a small difference in the plaintext, leads to a large difference in the hash digest produced.

We illustrate the use of hash functions in the following scenario. Suppose Alice wishes to send Bob a message, having previously agreed on a hash function to use. Then Alice can hash her

message and send to Bob both the original message and the hash generated. Bob can then check whether the received message reproduces the same hash digest. If Eve managed to intercept the message and changed it, then Bob would find a different digest and can conclude the message has been tampered with. This ensures authenticity and data integrity. We could also ensure non-repudiation by having Alice send an encrypted digest, known as a digital signature, that Bob can decrypt before checking the equality of the digests.

The security of a hash function depends on two factors: it should be computationally intractable to find the pre-image of a digest, and equally intractable to find two messages giving the same digest. In both cases, brute force attacks using a quantum computer make use of Grover's algorithm or the BHT algorithm [7]. However we can effectively mitigate these threats by increasing the hash length.

Symmetric Key Cryptography

Symmetric key cryptography addresses concerns for authenticity, confidentiality and data integrity. In practice, this involves the sharing of a key prior to encryption, then messages can be encrypted using the key, transmitted, and finally decrypted using the same key. Thus there are 2 key areas of vulnerability: firstly in the distribution of the keys and secondly in the security of the cipher using the key. The Caesar cipher explained in the introduction to this section is a simple example of a symmetric cipher. Alternatively the Advanced Encryption Cipher (AES) is a widely used symmetric cipher, from secure communication and encrypting stored data to VPNs and authenticity verification. Symmetric key cryptography is favourable as it is fast, relatively simple to implement and scales well with larger amounts of plaintext and users. However the main drawbacks are the lack of non-repudiation (since every party uses the same key to encrypt messages) and the difficulty of secure key distribution.

In creating a secure symmetric cipher, the following factors must be incorporated [36].

- Resistance to brute force attacks - this requires having a large key space, i.e. the number of potential keys. A quantum computer can use Grover's algorithm to search the key space quadratically faster than a classical attack. However unlike the hashing case, doubling the length of a key to increase the key space can make the de-/encryption process computationally harder.
- Resistance to cryptanalytic attacks - this requires the ciphertext to have high entropy, which amounts to not being able to determine the plaintext or key from the ciphertext. For example, the enigma machine cipher has a large key space, however it is vulnerable to frequency analysis attacks and thus the use of a computer makes it insecure. Research on the effect of quantum computers is ongoing, but some classically-secure symmetric ciphers have been shown to be compromised [37].

Asymmetric Key Cryptography

Asymmetric key cryptography addresses authenticity, confidentiality, data integrity and non-repudiation. In general it uses the following scheme: Alice and Bob each have a private and public key, then Alice can use Bob's public key to encrypt her message and Bob can decrypt it using his private key. To avoid the use of hashing without losing non-repudiation and authenticity, Alice could encrypt her message first with her private key, then with Bob's public key. Bob can then decrypt the message with his private key, followed by Alice's public key. Thus the message is guaranteed to have been encrypted by Alice.

We illustrate asymmetric key cryptography with the RSA protocol [38]:

1. Generate the public key (e, n), and private key (d, n):
   (a) Choose two distinct, random primes, p and q.
   (b) Define $n = pq$.
   (c) Choose an integer e, such that $1 < e < \varphi(n)$ and $\gcd(e, \varphi(n)) = 1$, where φ is the Euler's totient function defined
   $$\varphi(n) := |m \in \mathbb{N} \text{ such that } 1 \leq m \leq n \text{ and } \gcd(m, n) = 1|$$
   or in this case $\varphi(n) = (p-1)(q-1)$.
   (d) Calculate d such that $de \equiv 1 \mod \varphi(n)$.
2. To encrypt a message, convert it to an integer m such that $0 \leq m < n$, and create the ciphertext using the public key (e, n): $m' = m^e \mod n$.
3. To decrypt a message, m', use the private key (d, n): $m = m'^d \mod n$.

From the above we notice that since n is public, for Eve to recreate the private keys p, q, she must factorise n. Any other method of breaking RSA has also shown to be as difficult as factorising n [38]. Thus the security of RSA depends on the computational complexity of factorising n.

As explained in-depth in Section 2.2, Shor's algorithm provides a computationally tractable way to break RSA. In his groundbreaking paper [2], P. Shor also designed an algorithm to carry out the discrete logarithm, which breaks another well-used asymmetric cipher used for key exchange known as the Diffie-Hellman algorithm [39].

As illustrated in the above, the concern for more secure cryptographic protocols is well-found. Next we discuss a potential solution and provide some attempts to use a quantum computer to break it.

3.2 Lattice-Based Cryptography

Having introduced a selection of classical algorithms, as well as their vulnerability to quantum attacks, we present in this section, potential cryptographic protocols that are quantum resistant. Several have been researched, from code-based [40] to isogeny-based [41], but given that in the final list of four protocols that NIST chose, there are three lattice-based protocols [14], we focus our efforts here. First we introduce some common lattice-based problems with their complexity. Then we will summarise the encryption protocol and how it relates to the aforementioned problems. With this we can then begin to lay out how quantum computing can be implemented to solve the problem and break the encryption.

3.2.1 Lattice Theory

A lattice, with rank n and dimension m, is constructed from n linearly independent vectors $\{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n\} =: B$, where $\mathbf{b}_k \in \mathbb{R}^m \ \forall k$, by discrete linear superpositions. Mathematically,

$$\mathcal{L} := \{a_1\mathbf{b}_1 + a_2\mathbf{b}_2 + \cdots + a_n\mathbf{b}_n \text{ such that } (a_1, a_2, \ldots a_n) \in \mathbb{Z}^n\}.$$

We will be working with full-rank lattices, such that $n = m$. A basis, such as B, is linearly independent and spans the lattice. A key remark to make is that the basis of a lattice is not unique. In particular we can construct "good" bases, with relatively short, perpendicular vectors, as well as "bad" bases, with large, close to parallel vectors, as depicted in Figure 3.1. More precisely, we have the following theorem [42],

Theorem 3.1. *Two different bases, with basis matrices B, B', construct the same lattice if and only if there exists a unimodular matrix, A (i.e $|A| = \pm 1$), such that $B = AB'$.*

This idea will be of importance when solving the following problems.

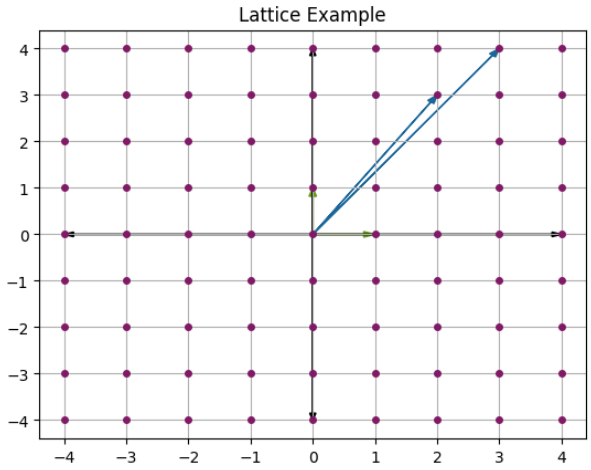

Figure 3.1: Example of good basis (green) and bad basis (blue) for a given integer lattice (red).

Lattice Problems

The shortest vector problem (SVP) - given a lattice, $\mathcal{L}$, find the length of the shortest non-zero vector between two lattice points, i.e. find $\lambda(\mathcal{L})$, where

$$\begin{aligned}\lambda(\mathcal{L}) &:= \min_{\mathbf{v}\in\mathcal{L}\setminus\{\mathbf{0}\}} ||\mathbf{v}||_A \\ &= \min_{\mathbf{v}_1,\mathbf{v}_2\in\mathcal{L},\mathbf{v}_1\neq\mathbf{v}_2} ||\mathbf{v}_1 - \mathbf{v}_2||_A,\end{aligned}$$

where $||\cdot||_A$ is the norm induced by the vector space.

It has been shown that for a random class of lattices in $\mathbb{Z}^n$, SVP is NP-hard [43]. Further research showed in addition, that in this case SVP is also NP-hard to approximate to within $2^{p/2}$, with respect to the ℓ_p norm [44].

The closest vector problem (CVP) - given an n-dimensional lattice and a point $\mathbf{x} \in \mathbb{R}^n$, find the lattice point closest to $\mathbf{x}$, i.e. find $\mathbf{u} \in \mathcal{L}$ such that

$$||\mathbf{u} - \mathbf{x}||_A = \min_{\mathbf{v}\in\mathcal{L}} ||\mathbf{v} - \mathbf{x}||_A.$$

Similarly to SVP, it has been shown that, even with pre-processing, where before the target vector $\mathbf{x}$ is revealed, an unlimited amount of processing of the given lattice, $\mathcal{L}$, can be carried out, CVP is NP-hard [45].

The bounded distance decoding problem (BDD) - special case of CVP, where we are given that the chosen point, $\mathbf{x} \in \mathbb{R}^n$, is within a given distance from the lattice.

We will next see the above problems underpin the mathematical foundations of the learning with errors cryptosystem presented in the following section.

3.2.2 Learning with Errors Cryptosystem

In the learning with errors cryptosystem, we are given four free parameters: the modulus, q, the number of samples, N, the dimension, n, and the probability distribution, χ. With these we outline the cryptosystem developed by O. Regev [46].

First we define the private key as a random vector of Alice's choice, $\mathbf{a} \in \mathbb{Z}_q^n$, where $\mathbb{Z}_q$ are the integers modulus q. Using this, we take N samples using the distribution χ, that form the public key as follows:

1. Choose $\mathbf{v} \in \mathbb{Z}_q^n$ uniformly at random, and $\epsilon \in \mathbb{Z}_q$ using the probability distribution χ.
2. Evaluate
$$w = \langle \mathbf{v}|\mathbf{a}\rangle + \epsilon, \tag{3.1}$$
where addition is taken modulus q.
3. Add the ordered pair $(\mathbf{v}, w)$ to the list of samples making up the public key and repeat until N samples are found.

We can encrypt a message, expressed as a string of binary numbers m, using the public key $\{(\mathbf{v}_k, w_k)\}_{k=1,\dots,N}$. For each bit m_k, choose a binary vector $\mathbf{r} \in \{0,1\}^n$ uniformly at random and encrypt it as follows:

$$m_k = 0 \mapsto \Big(\sum_{j=1}^{N} r_j \mathbf{v}_j, \sum_{j=1}^{N} r_j w_j\Big)$$
$$m_k = 1 \mapsto \Big(\sum_{j=1}^{N} r_j \mathbf{v}_j, \Big\lfloor \frac{q}{2} \Big\rfloor + \sum_{j=1}^{N} r_j w_j\Big).$$

To decrypt the ciphertext pair $(\mathbf{s}, t)$ using the private key $\mathbf{a}$, we note that in the case $m_k = 0$

$$t - \langle \mathbf{s}|\mathbf{a}\rangle = \sum_{j=1}^{N} r_j w_j - \sum_{j=1}^{N} r_j \langle \mathbf{v}_j|\mathbf{a}\rangle \tag{3.2}$$
$$= \sum_{j=1}^{N} r_j w_j - \sum_{j=1}^{N} r_j (w_j - \epsilon_j) \tag{3.3}$$
$$= \sum_{j=1}^{N} r_j \epsilon_j \approx 0, \tag{3.4}$$

where in Equation (3.4) we have made use of Equation (3.1). Similarly the $m_k = 1$ case becomes

$$t - \langle \mathbf{s}|\mathbf{a}\rangle = \Big\lfloor \frac{q}{2} \Big\rfloor + \sum_{j=1}^{N} r_j \epsilon_j \approx \Big\lfloor \frac{q}{2} \Big\rfloor.$$

To implement this we can thus simply round $2(t - \langle \mathbf{s}|\mathbf{a}\rangle)/q$ to the nearest integer and reproduce m.

Overall, breaking this cryptosystem amounts to using the public key alone to rederive the errors ϵ_k, to find the explicit values of $\langle \mathbf{v}_j|\mathbf{a}\rangle$, which then become an easily solved system of linear equations. However, finding these errors ϵ_k, was shown to be as hard as BDD [46], which was stated earlier in Section 3.2.1 as NP-hard.

The cryptographic protocol illustrated above has been extended to be more efficient using a module learning with errors framework, which replaces the additive group of the lattice with a modular lattice. This is what two of the three lattice-based cryptographic finalists in NIST's competition are based on (CRYSTALS-Kyber and CRYSTALS-Dilithium) [14].

3.2.3 The Problem Hamiltonian

In this and further sections, we consider the shortest vector problem defined in Section 3.2.1, and demonstrate how quantum computing can be applied to it.

Let our lattice have basis $\{b_1, b_2, \ldots, b_n\}$. Then any point, $\mathbf{r}$, on the lattice can be written

$$\mathbf{r} = \sum_{k=1}^{n} a_k \mathbf{b}_k$$

for $a_1, \ldots, a_n \in \mathbb{Z}^n$. This motivates the definition of the Gram matrix, G,

$$G_{jk} = \mathbf{b}_k \cdot \mathbf{b}_k,$$

from which we have

$$\begin{aligned} ||\mathbf{r}||^2 &= \sum_{j,k=1}^{n} (a_j \mathbf{b}_j) \cdot (a_k \mathbf{b}_k) \\ &= \sum_{j,k=1}^{n} G_{jk} a_j a_k. \end{aligned}$$

Next we wish to implement a continuous time quantum algorithm, as illustrated in Section 2.3, to solve SVP. To do this, we map SVP to a problem Hamiltonian, $\hat{\mathcal{H}}_P$, as follows [15]:

$$\hat{\mathcal{H}}_P := \sum_{j,k} G_{jk} \hat{N}_j \hat{N}_k, \tag{3.5}$$

where $\hat{N}_k$ has integer spectra and acts on the kth qubit and trivially elsewhere. Thus for integers $a_1, \ldots, a_n$,

$$\begin{aligned} \hat{N}_k |a_1 a_2 \cdots a_n\rangle &= a_k |a_1 a_2 \cdots a_n\rangle \\ \implies \hat{\mathcal{H}}_P |a_1 a_2 \cdots a_n\rangle &= \sum_{j,k} G_{jk} \hat{N}_j \hat{N}_k |a_1 a_2 \cdots a_n\rangle \\ &= \sum_{j,k} G_{jk} a_j a_k |a_1 a_2 \cdots a_n\rangle \\ &= ||\mathbf{r}||^2 |a_1 a_2 \cdots a_n\rangle. \end{aligned}$$

As a result we have shown that $\hat{\mathcal{H}}_P$ has eigenvalues corresponding to the squared length of arbitrary vectors on the lattice. In order to solve SVP, we simply need to simulate this Hamiltonian using continuous time quantum computing and measure the first excited state. The ground state is of course the trivial solution $\mathbf{r} = \mathbf{0}$.

Since we must work with finite operators to carry out quantum computation, we must truncate the spectrum range. One way of doing this is as follows [15]: we first introduce the qudit, which is represented by a system of qubits and expresses an integer in its binary decomposition. Explicitly, if $a_k = c_1 + 2c_2 + \cdots + 2^{M-1} c_M$, then the qudit $|a_k\rangle$ is

$$|a_k\rangle = |c_1 c_2 \cdots c_M\rangle.$$

Next we define the operator $\hat{N}_k$, acting non-trivially on the kth qudit,

$$\hat{N}_k := \frac{1}{2}\Big(\sum_{j=1}^{M} 2^{j-1} Z_{kj} + I\Big), \tag{3.6}$$

where Z_{jk} is the phase-flip gate defined in Equation (2.5), acting non-trivially on the jth qubit in the kth qudit. This operator has a spectrum of integers in the range $[-2^{M-1}+1, 2^{M-1}]$, for example in the case $M = 3$,

$$\hat{N}_1 = \frac{1}{2}\Big(\sum_{j=1}^{3} 2^{j-1} Z_{1j} + I\Big) = \frac{1}{2}\Big(Z_{11} + 2Z_{12} + 4Z_{13}\Big) + \frac{1}{2}I,$$

which applied to the integer eigenstate is

$$\hat{N}_1|a_1\rangle = \frac{1}{2}\Big((-1)^{c_1} + 2(-1)^{c_2} + 4(-1)^{c_3} + 1\Big)|c_1c_2c_3\rangle.$$

This gives a spectrum of integers in the range $[-3, 4]$, so we can associate $\hat{N}_k|a_k\rangle = a_k|a_k\rangle$ for the integer $a_k \in [-2^{M-1}+1, 2^{M-1}]$.

The method illustrated above to define $\hat{\mathcal{H}}_P$ can then be used to define a time dependent Hamiltonian that starts in an easy to prepare Hamiltonian $\hat{\mathcal{H}}_0$ and reaches $\hat{\mathcal{H}}_P$ at time T, as discussed in Section 2.3.

3.2.4 Previous Attempts to Solve SVP

There have been several methods solving SVP that use the problem Hamiltonian discussed in Section 3.2.3. In this section, we shed light on a few notable methods that inspired the work we present in Chapter 4.

Firstly, realising the problem Hamiltonian, using continuous time quantum computing to prepare the state, has been carried out successfully in numerical simulations and using quantum annealing [15] (in which states evolve adiabatically, but we start in an arbitrary state and converge upon the problem Hamiltonian), as well as using adiabatic quantum computation, with some optimisation techniques involved where we allow an approximate shortest vector as solution [16].

Unfortunately a large constraint in each of these cases, was the high demand of qubit space. With current quantum computers being highly susceptible to noise, increasing the number of qubits leads to a greater loss of information [8]. Alternatively, for the numerical simulation we must additionally deal with the exponential scaling of the Hilbert space that makes classical simulation computationally demanding. Moreover, it is interesting to note that it was found that if the simulation involved a slow sweep (i.e. slow time), as adiabaticity demands, the simulation primarily returned the zero vector, however in fast sweeps the returned vector would be random. Therefore a "Goldilocks zone" was established where we find the desired shortest vector [15]. This time-dependent constraint will be removed when we move to using quantum computing by dynamic invariants in Chapter 4.

The next method we review uses an iterative approach to find consecutively shorter vectors that converge on the SVP solution [47]. This method addresses the concern that in truncating the possible coefficients of lattice vectors, as in Equation (3.6), we risk finding the shortest vector within a subset of all possible vectors that does not contain the shortest vector itself. Explicitly, the iterative scheme presented uses a quantum approximate optimisation algorithm which gives, with high probability, a vector with a low energy. This will not necessarily be the desired first

excited state, but corresponds to a vector with a short squared length. Then the scheme uses this short vector to replace a longer vector in the basis, provided this preserves the basis, to create a new problem Hamiltonian with updated basis. Thus if the shortest vector could not be found in the original truncation, it may be found after several iterations of the scheme to give a basis of nicer, short vectors. One notable feature of this scheme is its flexibility. It has the potential to be used alongside classical schemes additionally simplifying the lattice, as well as other quantum algorithms that can find the shortest vector directly once the basis has been reduced far enough that truncation is no longer an issue.

Finally we note a method that, instead of considering general lattices, focuses on cyclic and nega-cyclic lattices and exploits their symmetries to reduce the problem [48]. More precisely, this method takes the problem Hamiltonian form, given in Equation (3.5), one step further and writes it in terms of the matrix U of eigenvectors of the Gram matrix, G, and eigenvectors, g_k:

$$\hat{\mathcal{H}}_P = \sum_{j,k=1}^{n} G_{jk}\hat{N}_j\hat{N}_k = \sum_{k=1}^{n} g_k \hat{S}_k^\dagger \hat{S}_k$$
$$\text{where } \hat{S}_k = \sum_{j=1}^{n} U_{kj}\hat{N}_j$$

Then using the properties of cyclic and nega-cyclic matrices, U_{kj} and g_k are known explicitly and depend only on the dimension n. It was then shown that low energy eigenvectors of $\hat{\mathcal{H}}$ coincide with high probability with the kernel of $\hat{S}_{k'}$, where k' corresponds to the largest eigenvalue of G. This allows for the reduction of $\hat{\mathcal{H}}$, such that it accounts for the constraint that short vectors are in the kernel of $\hat{S}_{k'}$. Overall this leads to a reduction in the required qubit space and was indeed carried out using the variational quantum eigensolver to show a greater likelihood of finding shorter vectors than without reduction. Making use of symmetries to reduce the Hamiltonian is a powerful technique, that will also be employed in Section 4.4.2, with the use of conserved quantities.

Chapter 4

Spin Chains with Long Range Interactions

"In natural science, Nature has given us a world and we're just to discover its laws. In computers, we can stuff laws into it and create a world."

- Alan Kay [49]

In this section, we experiment with the framework of dynamic invariants in quantum computation. In particular, the goal was inspired by the shortest vector problem, outlined in Section 3.2.1, which requires the simulation of a problem Hamiltonian, as derived in Equation (3.5), with notable long-range interaction terms, Z_jZ_k, connecting qubits in position j and k. We will begin with an outline of how Lie algebras have been used to help simulate the dynamics efficiently as well as how to improve the scalability further with reduced Lie algebras. To do this, we will make use of the Python package we developed that allows for the efficient manipulation of operators and construction of Lie algebras. An insight into how the package was developed can be found in Section 4.2 and a demonstration of its features can be found in the `demonstrations.ipynb` file of the GitHub repository [18].

4.1 Construction of Invariants

4.1.1 Lie Algebraic Construction

One of the main disadvantages of using invariants is the scaling: working with operators instead of states suggests a scaling that is quadratically worse. The general idea of using Lie algebras is to mitigate this problem, by ensuring the dynamics evolve within a Lie algebra with a nicer scaling. Then the scaling will follow the size of the Lie algebra, rather than the full Hilbert space squared. Using an algebraic formulation in dynamic invariant based quantum computing was first introduced by M. Sarandy et al. [27] and we discuss here the construction procedure developed by E. Torrontegui et al. [50].

In practice, we begin by defining the initial and problem Hamiltonians, $\hat{\mathcal{H}}_0$ and $\hat{\mathcal{H}}_T$ respectively, which form the boundary conditions of the time dependent Hamiltonian, $\hat{\mathcal{H}}(t)$. Then we seek a time-dependent invariant that starts and ends such that

$$[\hat{\mathcal{I}}(0), \hat{\mathcal{H}}_0] = 0 = [\hat{\mathcal{I}}(T), \hat{\mathcal{H}}_T].$$

Thus at $t = 0$ and $t = T$, the Hamiltonian and invariant share a common eigenbasis so we can start the simulation in an eigenstate of an easy-to-prepare Hamiltonian $\hat{\mathcal{H}}_0$ and, from the

discussion in Section 2.3.2, are guaranteed to end in an eigenstate of the problem Hamiltonian $\hat{\mathcal{H}}_T$. Now, we define a Lie algebra with time-independent operators $\{\hat{T}_k\}_{k=1,\dots,n}$, closed under commutations, such that we can decompose the Hamiltonian and invariant as follows,

$$\hat{\mathcal{H}}(t) = \sum_{k=1}^{n} a_k(t)\hat{T}_k \qquad \hat{\mathcal{I}}(t) = \sum_{k=1}^{n} b_k(t)\hat{T}_k,$$

with the boundary conditions at $t = 0$ and $t = T$ enforced on the coefficients $\{a_k(t)\}_{k=1,\dots,n}$ and $\{b_k(t)\}_{k=1,\dots,n}$. Now using Equation (2.24), we have

$$\frac{\partial}{\partial t}\hat{\mathcal{I}}(t) = i[\hat{\mathcal{I}}(t), \hat{\mathcal{H}}(t)] \tag{4.1}$$

$$\sum_{k=1}^{n} \dot{b}_k(t)\hat{T}_k = i \sum_{l,m=1}^{n} b_l(t)a_m(t)[\hat{T}_l, \hat{T}_m] \tag{4.2}$$

$$= \sum_{k,l,m=1}^{n} b_l(t)a_m(t)\gamma_{lm}^k \hat{T}_k, \tag{4.3}$$

where we have used in Equation (4.3) that the Lie algebra, $\{\hat{T}_k\}_k$, is closed under commutations, so we can write

$$[\hat{T}_l, \hat{T}_m] = -i\sum_{k=1}^{n} \gamma_{lm}^k \hat{T}_k.$$

Now since the Lie algebra forms a subset of a basis for the Hilbert space, we can match coefficients in Equation (4.3):

$$\begin{aligned}
\dot{b}_k(t) &= \sum_{l,m=1}^{n} b_l(t)a_m(t)\gamma_{lm}^k \\
&= \sum_{l=1}^{n}\Big(\sum_{m=1}^{n} a_m(t)\gamma_{lm}^k\Big)b_l(t) \\
&= \sum_{l=1}^{n} \Gamma_{kl}b_l(t) \quad \text{such that} \quad \Gamma_{kl} := \sum_{m=1}^{n} a_m(t)\gamma_{lm}^k, \quad \Gamma \in \mathbb{R}^{n\times n}.
\end{aligned}$$

Thus we have a first order linear differential equation for $\mathbf{b}^T(t) := (b_1(t), \dots, b_n(t))$ that can be readily solved,

$$\dot{\mathbf{b}}(t) = \Gamma\mathbf{b}(t),$$

with the boundary conditions

$$\hat{\mathcal{I}}(0) = \sum_{k=1}^{n} b_k(0)\hat{T}_k \qquad \hat{\mathcal{I}}(T) = \sum_{k=1}^{n} b_k(T)\hat{T}_k.$$

Since this is an over-constrained problem, we note that we cannot find an invariant $\hat{\mathcal{I}}$ for any specified $\hat{\mathcal{H}}(t)$ and boundary conditions on $\mathbf{b}(t)$, but must find suitable $\mathbf{a}(t)$ and $\hat{\mathcal{I}}(0), \hat{\mathcal{I}}(T)$ such that the dynamics are valid. Some techniques to ensure this are discussed in Section 4.4.3. Examples of using this formulation on the SU(2) algebra as well as two-level systems have been successfully carried out [50, 28].

Using this procedure to simulate dynamics thus has the potential to improve the scaling of a problem significantly. As suggested in the introduction of this section, we have now shown that instead of working with explicit states that scale exponentially with the number of spins, we can work with Lie algebras that scale with the size of the set, given in the case above by n. This is a crucial result in optimal control that was presented and exemplified in [17], where certain spin chains were used to demonstrate this favourable scaling with respect to the number of spins. This paper also suggests the further improvement of scaling discussed in the next section.

4.1.2 Reduced Lie Algebraic Construction

In the previous section, we ensured the initial Hamiltonian and invariant could be decomposed into operators of an algebra. This meant when we propagate $\hat{\mathcal{I}}$ in time, we remain within the algebra since the algebra is closed under commutations, which govern the equation of motion, given in Equation (2.24). However in doing this, we do not need the assumption that the Hamiltonian and invariant can be decomposed into the same set of operators. Indeed, we can express the Hamiltonian and invariant in terms of different sets and instead work with a subset of the full Lie algebra that contains the operators defining $\hat{\mathcal{I}}(0)$ and any commutations with the operators defining $\hat{\mathcal{H}}(0)$, but is not necessarily a complete Lie algebra. This opens up the possibility of working with even smaller sets of operators and allowing for greater computational efficiency.

Explicitly, we adapt the procedure of the previous section as follows. Let $\mathbf{R} := \{\hat{R}_k\}_{k=1,\dots,m}$ and $\mathbf{T} := \{\hat{T}_k\}_{k=1,\dots,n}$ be two sets of time-independent operators such that,

$$\hat{\mathcal{H}}(t) = \sum_{k=1}^{m} a_k(t)\hat{R}_k \qquad \hat{\mathcal{I}}(t) = \sum_{k=1}^{n} b_k(t)\hat{T}_k,$$

and additionally we require that the invariant set includes nested commutations with operators in the Hamiltonian set:

$$[\hat{T}_p, \hat{R}_q] = -i\sum_{k=1}^{n} \lambda_{pq}^k \hat{T}_k.$$

Then we can carry out an identical calculation as before, with adjusted sums. The resulting differential equation is then identified as,

$$\begin{aligned}
\frac{\partial}{\partial t}\hat{\mathcal{I}}(t) &= i[\hat{\mathcal{I}}(t), \hat{\mathcal{H}}(t)] \\
\sum_{k=1}^{n} \dot{b}_k(t)\hat{T}_k &= i\sum_{p=1}^{n}\sum_{q=1}^{m} b_p(t)a_q(t)[\hat{T}_p, \hat{R}_q] \\
&= \sum_{p,k=1}^{n}\sum_{q=1}^{m} b_p(t)a_q(t)\lambda_{pq}^k \hat{T}_k
\end{aligned}$$

$$\implies \dot{b}_k(t) = \sum_{p=1}^{n} \Lambda_{kp} b_p(t) \quad \text{such that} \quad \Lambda_{kp} := \sum_{q=1}^{m} a_q(t)\lambda_{pq}^k, \quad \Lambda \in \mathbb{R}^{n\times n},$$

or more concisely,

$$\dot{\mathbf{b}}(t) = \Lambda \mathbf{b}(t),$$

with the same boundary conditions as in the previous section,

$$\hat{\mathcal{I}}(0) = \sum_{k=1}^{n} b_k(0)\hat{T}_k \qquad \hat{\mathcal{I}}(T) = \sum_{k=1}^{n} b_k(T)\hat{T}_k.$$

Once again suitable $\mathbf{a}(t), \hat{\mathcal{I}}(0), \hat{\mathcal{I}}(T)$ must be determined in order to simulate the desired problem Hamiltonian. Some examples we will be working with are provided in Section 4.3.

4.2 Computational Insight

Operators on n spins are stored computationally as $2^n \times 2^n$ matrices. With such a scaling, carrying out several manipulations can be incredibly time and space consuming. Thus in this

section, we shed some light on the methods used to optimise the construction of Lie algebras and their reductions, as set out in Section 4.1.1 and Section 4.1.2. A demonstration of the package and its features can be found in the repository [18] under the file `demonstrations.ipynb`.

In particular, we focus in this section on completing Lie algebras. To carry out this construction, we must store the operators $\mathbf{T}$ as lists and perform the relevant commutations until the set is both complete and linearly independent. More precisely, we follow the pseudo code in Listing 4.1,

Listing 4.1: Pseudocode for completing Lie algebra

```
def complete_algebra(algebra):
    new_algebra = algebra
    while len(new_algebra) != len(algebra):
        algebra = new_algebra
        for operator1 in algebra:
            for operator2 in algebra:
                new_operator = commutation(operator1, operator2)
                new_algebra = algebra.append(new_operator)
                if new_algebra is not linearly independent:
                    remove new_operator
    return new_algebra
```

Thus in order to complete a Lie algebra, we must find efficient ways to carry out commutations and linear independence.

The obvious way of doing this is simply storing the operators as matrices and applying matrix multiplication to evaluate commutations and find the rank of the list of matrices for linear independence. Indeed, this is implemented in the function `complete_algebra(Ops: list, max: int, start: int = 0)`, which also allows for the algorithm to stop after a set maximum of operators is reached.

Now to improve this function, we can store and manipulate operators differently. As such, we observe that any operator acting on a single spin can be decomposed into the Pauli operators and the identity, $\{I, X, Y, Z\}$, as defined in Equation (2.5). Then we can create a basis for n spins from the set $\{I, X, Y, Z\}^{\otimes n}$. Now instead of storing operators as matrices, we can create a new object `Pauli(decomp:[])` that represents a single basis operator. For example we have the following relation,

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} = I \otimes X \iff \texttt{Pauli([I, X])}.$$

Then to represent any given operator, we create the object `SuperPauli(paulis: list = [])` that allows for any superposition of the Pauli class. For example,

$$\begin{pmatrix} 0 & 1 & 2 & 0 \\ 1 & 0 & 0 & 2 \\ 2 & 0 & 0 & 1 \\ 0 & 2 & 1 & 0 \end{pmatrix} = I \otimes X + 2X \otimes I$$

$$\iff \texttt{SuperPauli([(1, Pauli([I, X])), (2, Pauli([X, I]))])}.$$

Not only is this far more space efficient as we increase the number of spins, it is also a more natural representation for working with spin chains.

Now in order to maintain this representation, we also need an efficient way to evaluate the commutation that does not require using explicit matrices. To do this, we first note the following theorem,

Theorem 4.1. *Either the commutator or the anti-commutator of two basis Pauli operators must vanish.*

Proof. First we consider a single spin. Indeed we can derive tables of commutations and anti-commutations, given in Table 4.1 and Table 4.2. Thus the claim is evident in the case of one

$[\cdot,\cdot]$	X	Y	Z	I
X	0	$2iZ$	$-2iY$	0
Y	$-2iZ$	0	$2iX$	0
Z	$2iY$	$-2iX$	0	0
I	0	0	0	0

Table 4.1: Commutation evaluations.

$\{\cdot,\cdot\}$	X	Y	Z	I
X	$2I$	0	0	$2X$
Y	0	$2I$	0	$2Y$
Z	0	0	$2I$	$2Z$
I	$2X$	$2Y$	$2Z$	$2I$

Table 4.2: Anti-commutation evaluations.

spin and we can proceed by induction on the number of spins n. Suppose for a system with n spins, the theorem holds. Then for $a_1,\ldots,a_{n+1},b_1,\ldots,b_{n+1}\in\{I,X,Y,Z\}$,

$$[a_1\otimes\cdots\otimes a_{n+1},b_1\otimes\cdots\otimes b_{n+1}] = a_1b_1\otimes\cdots\otimes a_{n+1}b_{n+1} - b_1a_1\otimes\cdots\otimes b_{n+1}a_{n+1} \tag{4.4}$$

$$= a_1b_1\otimes\cdots\otimes a_nb_n\otimes\frac{1}{2}\Big([a_{n+1},b_{n+1}]+\{a_{n+1},b_{n+1}\}\Big) \tag{4.5}$$

$$- b_1a_1\otimes\cdots\otimes b_na_n\otimes\frac{1}{2}\Big(\{a_{n+1},b_{n+1}\}-[a_{n+1},b_{n+1}]\Big) \tag{4.6}$$

$$= \frac{1}{2}\Big(\{a_1\otimes\cdots\otimes a_n,b_1\otimes\cdots\otimes b_n\}\otimes[a_{n+1},b_{n+1}] \tag{4.7}$$

$$+ [a_1\otimes\cdots\otimes a_n,b_1\otimes\cdots\otimes b_n]\otimes\{a_{n+1},b_{n+1}\}\Big), \tag{4.8}$$

and a similar relation holds for the anti-commutation,

$$\{a_1\otimes\cdots\otimes a_{n+1},b_1\otimes\cdots\otimes b_{n+1}\} = \frac{1}{2}\Big([a_1\otimes\cdots\otimes a_n,b_1\otimes\cdots\otimes b_n]\otimes[a_{n+1},b_{n+1}] \tag{4.9}$$

$$+ \{a_1\otimes\cdots\otimes a_n,b_1\otimes\cdots\otimes b_n\}\otimes\{a_{n+1},b_{n+1}\}\Big). \tag{4.10}$$

Now since we have assumed the theorem for n spins, we have that either $[a_1\otimes\cdots\otimes a_n,b_1\otimes\cdots\otimes b_n]$ or $\{a_1\otimes\cdots\otimes a_n,b_1\otimes\cdots\otimes b_n\}$ is zero and from the single spin case either $[a_{n+1},b_{n+1}]$ or $\{a_{n+1},b_{n+1}\}$ is zero. Thus from the above calculation, we have verified the theorem for the $n+1$ case, which concludes the proof. □

In light of the above theorem we can iteratively apply the tables 4.1 and 4.2 to the equations 4.8 and 4.10 to find the commutator or anti-commutator of two Pauli basis operators. This will involve n iterations. However we can vectorise this operation to carry out all n iterations at once, since we know that there exists exactly one successful combination of commutators and anti-commutators for each spin pair, (a_k,b_k). The final hurdle is then to determine whether the successful, non-zero operator found is the commutator or anti-commutator. This can be found by noting that for a single spin, the commutator results in an imaginary prefactor, whereas the anti-commutator remains real. Thus if the resulting operator has an imaginary coefficient, then an odd number of commutators were carried out. This means there were an odd number of sign changes and we must have evaluated the commutator and the anti-commutator is zero. Conversely if the coefficient is real then we have evaluated the anti-commutator and the commutator is zero.

Finally, since the commutator/anti-commutator is distributative, we can use the above argument on linear combinations of basis Pauli operators by iterating through each basis operator. Indeed this has been implemented in the function `acomm_comm(A: Pauli, B: Pauli)` that returns the non-zero evaluation and whether this is the commutator or anti-commutator. To complete the Lie algebra construction, we note that linear independence can be carried out in an identical fashion to the matrix case, but using the basis expansion. Additional speed-up was also achieved in ensuring commutations are not repeated in the algorithm.

We can compare the time efficiency of the matrix and Pauli representation to complete simple, random algebras of increasing number of spins. This is illustrated in Figure 4.1, where for small n, the matrix manipulations are highly efficient but as the number of spins increases the matrix representation appears to grow exponentially whereas the Pauli representation remains fast. The algebras tested were ensured to be of size less than 20 to allow for faster testing, and exact results are given in the repository [18].

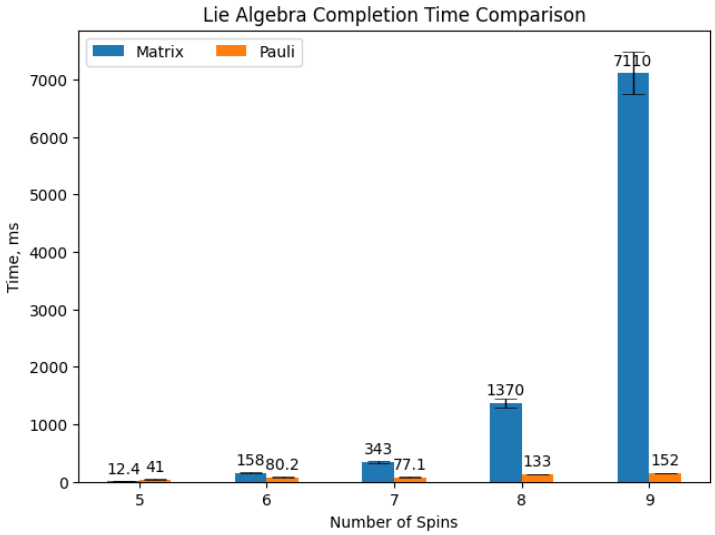

Figure 4.1: Demonstration of the time taken to complete Lie algebras of increasing size, using the matrix and Pauli representations, plotted with error bars.

Overall, we can clearly see that using the Pauli decomposition gives a substantial computational improvement to evaluating commutation relations and completing algebras.

4.3 Study of Examples

We consider four examples, illustrating the construction of invariants via the procedures discussed in Section 4.1.2 and Section 4.1.1. Their derivation, as well as several other examples, can be found in the repository [18]. Of the examples tried, we discuss these four in more depth, as they exemplify well how to build long range interactions and where the reduced algebraic formulation can provide a computational advantage.

4.3.1 Example 0: Lie Algebraic Procedure

We first introduce one of the Lie algebras studied in [17] that demonstrates the power of using Lie algebras on spin chains of size n and inspired the search for smaller sets using the reduced formulation. In this case the operators chosen to decompose the Hamiltonian and subsequently

the invariant are given by,

$$\mathbf{R} = \mathbf{T} = \left\{ \{Z_k\}_{k=1}^{n}, \{X_k X_{k+1}\}_{k=1}^{n-1}, X_1, X_n \right\}$$

and completed with respect to commutations. These Lie algebras were found to scale as $2n^2 + 3n + 1$ for n spins, which is clearly more favourable than the otherwise exponential scaling of states. In particular this Lie algebra can be used to generate a GHZ state, which will be briefly revisited in Section 4.4.3 and more discussion can be found in [17].

4.3.2 Example 1: Reverse-Engineering Long Range Interactions

Inspired by the above example, we adjust $\mathbf{R}$ and let $\mathbf{T}$ contain the sought after long range interactions. The intention is that, assuming a time-evolution path exists, we will be able to start with any other element generated in $\mathbf{T}$ and evolve to the desired long range interaction. As such, we fix $k \leq n$ and start with,

$$\mathbf{R} = \left\{ \sum_{j=1}^{n} Z_j, \sum_{j=1}^{n} X_j X_{j+1} \right\} \quad \mathbf{T}_k = \left\{ Z_1 Z_k \right\}. \tag{4.11}$$

Then we construct $\mathbf{T}'_k$ to include nested commutations with $\mathbf{R}$, as discussed in Section 4.1.2. Note also we have assumed periodic boundary conditions, such that $A_{n+1} = A_1$ for any operator A_j acting on spin j. We leave the value of k open, noting that if the scaling is favourable, then we can extend the invariant to

$$\mathbf{T} = \left\{ \{Z_1 Z_k\}_{k=1}^{n} \right\}$$

and find that the generated set is $\mathbf{T}' = \cup_{k=2}^{n} \mathbf{T}'_k$. The explicit sets generated, $\mathbf{T}'_k$, can be found in the repository for various k and n [18]. In particular, it was found that the set generated consisted of operators interacting spins around the first spin with spins around the kth spin. This means there are many trivial terms that lead to degeneracies. Indeed if the jth spin only has the identity acting on it for all terms in $\mathbf{T}'$, then this spin can be flipped, without affecting the eigenvalue. This is an undesirable feature, as we would like to start and end in an eigenstate of $\mathcal{I}$ that is non-degenerate. Additionally, while it is interesting to observe the locality of interactions that $Z_1 Z_k$ generates, this means that to evolve within this set we must have long range interactions at all times, which is not useful experimentally.

We can also make a note of the scaling: in Table 4.3 we present the sizes for $k = \lfloor n/2 \rfloor$, where it should be noted that since the set chosen for $\mathbf{R}$ is periodic, it has a symmetry with respect to permutations $\sigma_j \mapsto \sigma_{j+1}$ where σ_j is the jth spin. This means the set produced from the combination (n, k) will be identical to $(n, n-k)$. Thus choosing $k = \lfloor n/2 \rfloor$ is the longest possible interaction we can demand. We notice that the results indicate set sizes, that are not

Spins	Size
3	15
4	22
5	111
6	225

Table 4.3: Invariant set sizes with Hamiltonian defined in Equation (4.11), for increasing number of spins n and $k = \lfloor n/2 \rfloor$.

too large which indicates we can adjust our sets $\mathbf{T}$ and $\mathbf{R}$ to include more terms that hopefully include operators that are easier to simulate, as will be done in the following example.

4.3.3 Example 2: Engineering Long Range Interactions

In the previous example, we ensured the invariant contained long range interactions that would define $\hat{\mathcal{I}}(T)$ and tried to find operators in the invariant that could be implemented easily to define $\hat{\mathcal{I}}(0)$. This time we take the opposite approach, and ensure nearest-neighbour interactions are present in the invariant set to define $\hat{\mathcal{I}}(0)$, which will be easy to implement, and set up $\mathbf{R}$ to turn the nearest neighbour interactions into long range interactions in the generated operators of the invariant set. To make these long range interactions, we observe

$$[Z_1Z_2, X_2X_3] \propto Z_1Y_2X_3 \tag{4.12}$$
$$[Z_1Y_2X_3, Z_2] \propto Z_1X_2X_3 \tag{4.13}$$
$$[Z_1X_2X_3, Z_3] \propto Z_1X_2Y_3 \tag{4.14}$$
$$[Z_1X_2Y_3, X_2X_3] \propto Z_1Z_3. \tag{4.15}$$

Thus to generate the interaction Z_1Z_k via the above nested commutation sequence, we need the following starting sets,

$$\mathbf{R} = \left\{\{Z_j\}_{j=1}^n, \{X_jX_{j+1}\}_{j=1}^n\right\} \quad \mathbf{T} = \left\{Z_1Z_2\right\}. \tag{4.16}$$

This set generates the set sizes, $|\mathbf{T}'|$, given in Table 4.4. In studying the full sets, we see that

Spins	Size
3	30
4	70
5	420
6	990
7	2002

Table 4.4: Reduced Lie algebra size with Hamiltonian and invariant defined in Equation (4.16), for increasing number of spins n.

we indeed find all intended interactions Z_jZ_k, which are needed for the problem Hamiltonian in Equation (3.5). However since at each additional spin, the set size, $|\mathbf{T}'|$, almost doubles in size, this suggests a likely unfavourable scaling. Thus our next example finds a compromise between this example and the previous one.

4.3.4 Example 3: Compromised Interactions

In Example 2, we successfully found a set that had nearest neighbour interactions, desirable for $\hat{\mathcal{I}}(0)$, and long range interactions, desirable for $\hat{\mathcal{I}}(T)$. However the scaling was unfavourable, thus here we look for a smaller set. We observe that the chain of commutators we used to turn Z_kZ_{k+1} into Z_kZ_{k+2}, demonstrated in Equation (4.15), also causes an avalanche of new many-body interactions that make the set explode. Thus in this final approach we cut off this avalanche. Unfortunately, to do this we must sacrifice the goal of creating any long range interaction Z_1Z_k for any k. Instead we fix k and use the chain of commutations from Equation (4.15) to ensure our invariant can reach any Z_pZ_{p+q} for $q \leq k$. For example, if we take $k = 2$, so that we can turn nearest neighbour interactions into next-to-nearest neighbour interactions, we need the Hamiltonian to consist of the individual Z_j terms as well as $\{X_1X_2, X_3X_4, X_5X_6, \dots\}$, or more generally,

$$\mathbf{R} = \left\{\{X_{2j-1}X_{2j}\}_{j=1}^{\lfloor n/2 \rfloor}\right\} \quad \mathbf{T} = \left\{\{Z_jZ_{j+1}\}_{j=1}^{n-1}\right\}.$$

We can see that the avalanche-inducing X_jX_{j+1} is cut off at every other interaction. As will be shown in the subsequent scaling analysis, we can prove that this set guarantees a linear scaling

in n, while we can expect the scaling in k to be as bad as in the previous example. Indeed, the results reflect this, and present the following scaling for $k = 2$.

$$\begin{aligned} n = 2m : &\qquad 37m - 36 \\ n = 2m + 1 : &\qquad 37m - 30. \end{aligned}$$

We can easily extend this argument to $k = 3$, where we now allow one overlap in the X_jX_{j+1} chains before cutting it off. Then we have that the Hamiltonian must consist of the individual Z_j terms as well as $\{X_1X_2, X_2X_3, X_4X_5, X_5X_6, \dots\}$. This gives rise to the following scaling,

$$\begin{aligned} n = 3m : &\qquad 240m - 225 \\ n = 3m + 1 : &\qquad 240m - 210 \\ n = 3m + 2 : &\qquad 240m - 134. \end{aligned}$$

A general formula for this construction, also handling end cases, will be derived in the subsequent scaling analysis, and is given in Equation (4.18).

Scaling Analysis

In this subsection, we verify that the scaling of the suggested spin chain given in Section 4.3.4 is linear in n for a given required kth nearest-neighbour. More precisely, we must first generalise the operator set-up for a given number of spins, n, and length of desired interaction, k, for $k \geq 2$. We define,

$$\mathcal{J}_k(m) := \left\{ \{X_jX_{j+1}\}_{j=m}^{m+k-1} \right\} \qquad \mathcal{J}_0(m) = \emptyset.$$

Then we can define the Hamiltonian and invariant sets, $\mathbf{R}$ and $\mathbf{T}$ respectively, as follows

$$\mathbf{T} = \left\{ \{Z_jZ_{j+1}\}_{j=1}^{n-1} \right\} \tag{4.17}$$

$$\mathbf{R} = \left\{ \{Z_j\}_{j=1}^{n}, \{\mathcal{J}_k(1 + m(k+1))\}_{j=0}^{\lfloor \frac{n}{k+1} \rfloor - 1}, \mathcal{J}_{n \bmod (k+1)-1}\left(\left\lfloor \frac{n}{k+1} \right\rfloor \cdot (k+1) + 1 \right) \right\}. \tag{4.18}$$

So for example, the Hamiltonian set in the case $n = 10$, $k = 3$ is,

$$\begin{aligned} \mathbf{R} &= \left\{ \{Z_j\}_{j=1}^{10}, \mathcal{J}_3(1), \mathcal{J}_3(5), \mathcal{J}_1(9) \right\} \\ &= \left\{ \{Z_j\}_{j=1}^{10}, X_1X_2, X_2X_3, X_3X_4, X_5X_6, X_6X_7, X_7X_8, X_9X_{10} \right\}, \end{aligned}$$

with corresponding nearest neighbour interactions in the invariant set. Intuitively, we can see that there will be a chain of commutations, similar to Equation (4.15), that will cascade along the $\mathcal{J}_k(m)$ set of operators to produce Z_jZ_{j+m} for all $m \leq k$. However we have ensured the chain of X_jX_{j+1} in the Hamiltonian is cut off at the kth nearest neighbour interaction to avoid generating an avalanche of new terms. We see that this means it is impossible to connect the jth spin to spins beyond $j + 2(k+1)$. More mathematically, this idea is applied in the following proof.

Theorem 4.2. *Let $\boldsymbol{T}$ and $\boldsymbol{R}$ be defined as in Equation (4.17) and Equation (4.18), respectively, for fixed k, n. Then in building the reduced Lie algebra, derived in Section 4.1.2, we are guaranteed that $\boldsymbol{T}$ contains the terms $\{Z_jZ_{j+m}\}_{m=1}^{k}$. Additionally, for fixed k there is a linear scaling in n.*

Proof. First we show that we produce the desired mth nearest-neighbour interactions. We fix $m \leq k$ and wish to construct Z_jZ_{j+m}. Observe the following algorithm for interacting larger

distance spins:

$$\begin{aligned}[Z_jZ_{j+1}, X_{j+1}X_{j+2}] &\propto Z_jY_{j+1}X_{j+2}\\ [Z_jY_{j+1}X_{j+2}, Z_{j+1}] &\propto Z_jX_{j+1}X_{j+2}\\ [Z_jX_{j+1}X_{j+2}, Z_{j+2}] &\propto Z_jX_{j+1}Y_{j+2}\\ [Z_jX_{j+1}Y_{j+2}, X_{j+1}X_{j+2}] &\propto Z_jZ_{j+2}.\end{aligned}$$

Thus we can transform $Z_jZ_{j+1} \mapsto Z_jZ_{j+2}$ using $\{X_{j+1}X_{j+2}, Z_{j+1}, Z_{j+2}\}$. Now if the terms X_j and X_{j+m} both appear in a given $\mathcal{J}_k(l)$, then we can use the algorithm iteratively until we reach Z_jZ_{j+m} since the block $\mathcal{J}_k(l)$ contains all the required terms. Indeed, we can transform

$$Z_jZ_{j+1} \mapsto Z_jZ_{j+2} \mapsto \cdots \mapsto Z_jZ_{j+m}.$$

Alternatively, if X_j appears in $\mathcal{J}_k(l)$ and X_{j+m} appears in $\mathcal{J}_{k'}(l+mk)$ (where $k' = n \bmod (k+1) - 1$ if we have hit the end case where $\min(l+mk+k, n) = n$, and otherwise $k' = k$), then there exists p such that $j \leq p < j+m$ and $p \equiv 0 \bmod (k+1)$. Then we can apply the algorithm starting at the pivot point, p:

$$Z_pZ_{p+1} \mapsto Z_{p-1}Z_{p+1} \mapsto \cdots \mapsto Z_jZ_{p+1} \mapsto Z_jZ_{p+2} \mapsto \cdots \mapsto Z_jZ_{p+m}.$$

where we note that the algorithm can identically be applied to transform $Z_{j+1}Z_{j+2} \mapsto Z_jZ_{j+2}$ using $\{X_jX_{j+1}, Z_j, Z_{j+1}\}$. Thus we can make Z_jZ_{j+m} from nearest neighbour interactions Z_jZ_{j+1} and nested commutations with operators in $\mathbf{R}$.

Next we show the linear scaling. In repeatedly applying commutations with operators in $\mathbf{R}$, we differentiate between 2 cases:

- *Inner case:* starting from Z_jZ_{j+1} where $j \not\equiv 0 \bmod (k+1)$, we see that operators in $\mathbf{R}$ will only act non-trivially for interactions between spins in the range present in the given block $\mathcal{J}_{k'}(l)$ that contains X_jX_{j+1}, and commute with all other terms. Thus the block will never interact outside of its range and each block can be treated independently. Since the number of terms a given block generates is independent of n, we let all the operators generated from one block, as a result of all its inner cases, be $M(p)$. Here p refers to the block $\mathcal{J}_p(l)$ and its positioning l is irrelevant.

- *Edge case:* starting from Z_jZ_{j+1} where $j \equiv 0 \bmod (k+1)$, we can similarly see that non-trivial interactions will now be caused by two blocks, $\mathcal{J}_{k'}(j+1)$ and $\mathcal{J}_k(j-k)$. Thus using the same argument as for the inner case, we can let all operators generated from these blocks due to an edge case be $M'(p,q)$, where p and q refer to the two blocks on either side of the edge case.

Since $M(k)$ and $M'(k)$ are mutually exclusive, for fixed k, this then gives us a total number of operators,

$$\left\lfloor \frac{n}{k+1} \right\rfloor M(k) + M\Big(n \bmod (k+1) - 1\Big) + \Big(\left\lfloor \frac{n}{k} \right\rfloor - 1\Big) M'(k,k) + M'\Big(k, n \bmod (k+1) - 1\Big),$$

which is linear in n. □

Overall, while the long range interaction condition was lost, we can still apply this set to other problems as will be discussed in Chapter 5. Now that we have found a set with favourable scaling in n, we can discuss which evolutions are dynamically allowed.

4.4 Construction of Time-Evolution

As mentioned previously, determining a valid time-evolution for $\hat{\mathcal{I}}(t)$ is crucial. In this section, we cover some techniques for ensuring that the evolution is dynamically possible.

In particular, we will apply these techniques to Example 3, studied in Section 4.3.4, for the case $n = 4, k = 3$. From the set, $\mathbf{T}'$, found in this case, we have all $Z_k Z_l$ interactions available. Thus we propose an initial and final invariant,

$$\hat{\mathcal{I}}(0) = \sum_{j=1} \lambda_j Z_j Z_{j+1} \qquad \hat{\mathcal{I}}(T) = \sum_{j,l=1}^{4} = \mu_{jl} Z_j Z_l,$$

and try to find valid μ_{jl} such that it is possible to evolve to $\hat{\mathcal{I}}(T)$ for given λ_j.

4.4.1 Spectral Analysis

In Section 2.3.2, we made an important conclusion that the spectrum of an invariant must remain constant throughout time. Explicitly this was derived in Equation (2.28), and here we make use of this condition to give constraints on the possible μ_{jl}.

Since the operators used to construct $\hat{\mathcal{I}}(0)$ and $\hat{\mathcal{I}}(T)$ and diagonal, we can simply read off the diagonal elements to give the spectrum. This gives a system of 16 equations relating the 6 unknowns μ_{jl} to given λ_j. As expected from the construction, each eigenvalue of $\hat{\mathcal{I}}(0)$ and $\hat{\mathcal{I}}(T)$ is doubly degenerate. The next remark to make is that $[Z_1 Z_2, \hat{\mathcal{H}}(t)] = 0 = [Z_3 Z_4, \hat{\mathcal{H}}(t)]$ for all times. Thus the equation of motion for $\hat{\mathcal{I}}$ requires that the coefficients of $Z_1 Z_2$ and $Z_3 Z_4$ are constant in time. This gives the constraints $\mu_{12} = \lambda_1$ and $\mu_{34} = \lambda_3$. Rearranging the system of equations to account for these constraints gives us,

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & -1 \\ -1 & 1 & 1 & -1 \\ -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \begin{pmatrix} \mu_{23} \\ \mu_{13} \\ \mu_{14} \\ \mu_{24} \end{pmatrix} = P \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & -1 \\ 1 & -1 & -1 \\ 1 & -1 & 1 \\ -1 & -1 & 1 \\ -1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & 1 & 1 \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{pmatrix} - \lambda_1 \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \\ -1 \end{pmatrix} - \lambda_2 \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \\ 1 \\ -1 \\ -1 \\ 1 \end{pmatrix}, \quad (4.19)$$

where P is any permutation matrix, i.e. $P^T P = I = P P^T$. Now we can notice that on the LHS of this equation, row 1 is the negative of row 4, and similarly for rows $(2, 3), (5, 8), (6, 7)$. This gives us four new constraints on the possible permutation matrices. Given that these four constraints are imposed, we can reduce this sytem of equations to,

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ -1 & 1 & 1 & -1 \\ -1 & 1 & -1 & 1 \end{pmatrix} \begin{pmatrix} \mu_{23} \\ \mu_{13} \\ \mu_{14} \\ \mu_{24} \end{pmatrix} = G$$

$$\implies \begin{pmatrix} \mu_{23} \\ \mu_{13} \\ \mu_{14} \\ \mu_{24} \end{pmatrix} = \frac{1}{4} \begin{pmatrix} 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} G,$$

where G is a known 4×4 matrix made up of rows $1, 2, 5, 6$ of the RHS of Equation (4.19). This gives us several solutions for μ_{jl} and we note here three valid possibilities,

$$\hat{\mathcal{I}}(T) = \lambda_1 Z_1 Z_2 + \lambda_2 Z_1 Z_3 + \lambda_3 Z_3 Z_4 \tag{4.20}$$

$$\hat{\mathcal{I}}(T) = \lambda_1 Z_1 Z_2 + \frac{1}{2}\lambda_2 \Big(Z_2 Z_3 - Z_1 Z_3 - Z_1 Z_4 - Z_2 Z_4 \Big) + \lambda_3 Z_3 Z_4 \tag{4.21}$$

$$\hat{\mathcal{I}}(T) = \lambda_1 Z_1 Z_2 + \lambda_2 Z_2 Z_3 + \lambda_3 \Big(Z_3 Z_4 - Z_1 Z_4 - Z_2 Z_4 \Big). \tag{4.22}$$

In the following analysis, we will take the simplest evolution with $\lambda_1 = 0 = \lambda_3$, $\lambda_2 = 1$ and Equation (4.20). Thus, in this case we seek the transformation $Z_2 Z_3 \mapsto Z_1 Z_3$ from a nearest neighbour interaction to a next-to-nearest neighbour interaction.

4.4.2 Conserved Quantities

In this section we provide two important uses of conserved quantities. The first will reduce the differential equation in $\mathbf{b}(t)$, which defines the time-evolution of the invariant, to several differential equations with smaller dimension. The second will give an elegant constraint on the trace of the invariant at different times.

We recall our Hamiltonian and invariant expansions are,

$$\hat{\mathcal{H}}(t) = \sum_{k=1}^{m} a_k(t) \hat{R}_k \qquad \hat{\mathcal{I}}(t) = \sum_{k=1}^{n} b_k(t) \hat{T}_k.$$

Then we define a time-independent conserved quantity as an operator that commutes with the Hamiltonian. In our setup, this means an operator $\hat{A}$ such that

$$[\hat{A}, \hat{R}_k] = 0 \qquad \forall k = 1, 2, \ldots, m.$$

Now an important consequence of conserved quantities, is that their expectation values are time-independent. Indeed, suppose we are in the state $|\psi(t)\rangle$ at time t, then for the time-evolution operator $\hat{\mathcal{U}}(t)$,

$$\langle \hat{A} \rangle(t) = \langle \psi(t) | \hat{A} | \psi(t) \rangle \tag{4.23}$$

$$= \langle \psi(0) | \hat{\mathcal{U}}^\dagger(t) \hat{A} \hat{\mathcal{U}}(t) | \psi(0) \rangle \tag{4.24}$$

$$= \langle \psi(0) | \hat{A} | \psi(0) \rangle = \langle \hat{A} \rangle(0), \tag{4.25}$$

where in Equation (4.25), we have used that since $\hat{A}$ commutes with the Hamiltonian, it must do so with the time-evolution operator, as it can be explicitly written as given in Equation (2.31).

Before we make use of conserved quantities, we note that in Section 4.1.2, we found that the invariant equation of motion, given in Equation (2.24), implies

$$\dot{\mathbf{b}}(t) = \Lambda \mathbf{b}(t)$$

for a given, time-dependent, square matrix, Λ, specified by $\mathbf{a}(t), \mathbf{R}, \mathbf{T}$. Then for any matrix, P, we have

$$\begin{aligned} \dot{\mathbf{b}}(t) &= \Lambda \mathbf{b}(t) \\ &= P^{-1} \Big(P \Lambda P^{-1} \Big) P \mathbf{b}(t) \\ \implies \dot{\mathbf{s}} &= L \mathbf{s}, \end{aligned}$$

where we have defined $\mathbf{s}(t) := P\mathbf{b}(t)$ and $L := P\Lambda P^{-1}$. Thus if we choose P such that L is block diagonal, we can ensure the dynamics evolve in these blocks and reduce the problem. For example, in the case where Λ is time-independent and we impose the boundary condition at $t = 0$, we have the solution,

$$\mathbf{b}(t) = e^{\Lambda t}\mathbf{b}(0).$$

Now if we apply the transformation,

$$\begin{aligned} \mathbf{b}(t) &= e^{P^{-1}LPt}\mathbf{b}(0) \\ &= P^{-1}e^{Lt}P\mathbf{b}(0) \\ \implies \mathbf{s}(t) &= e^{Lt}\mathbf{s}(0) \end{aligned}$$

where, since L is block diagonal, we have

$$L = \begin{pmatrix} L_1 & 0 & 0 & \cdots \\ 0 & L_2 & 0 & \\ 0 & 0 & L_3 & \\ \vdots & & & \ddots \end{pmatrix} \implies e^{Lt} = \begin{pmatrix} e^{L_1 t} & 0 & 0 & \cdots \\ 0 & e^{L_2 t} & 0 & \\ 0 & 0 & e^{L_3 t} & \\ \vdots & & & \ddots \end{pmatrix}. \tag{4.26}$$

Thus the linear differential equation in $\mathbf{s}(t)$ can be solved in blocks and transformed back into $\mathbf{b}(t)$, reducing the computational effort needed to solve the problem.

In order to find this block diagonal structure, we can look at conserved quantities. Indeed, since conserved quantities commute with the Hamiltonian at all times, they will share an eigenspace. As a result, if we start the system in an eigenstate of both $\hat{A}$ and $\hat{\mathcal{H}}$, then at a later time the state must still be in the corresponding eigenspace, since the expectation of $\hat{A}$ must be preserved. Thus each block corresponds to a projection onto the eigenspaces of the conserved quantity, $\hat{A}$. If we can find several non-commuting conserved quantities, this can reduce the problem significantly by treating each block, and potential sub-blocks, independently.

This principle can be naturally applied to Example 1, from Section 4.3.2. As previously mentioned, we have the following symmetry

$$\Pi\hat{R}_k\Pi^\dagger = \hat{R}_k \qquad \forall k = 0, 1, \ldots, m$$

where n is the number of spins and Π shifts all the spins, such that $\Pi\hat{A}_j\Pi^\dagger = \hat{A}_{j+1}$ for operator $\hat{A}$ acting on the jth spin. Thus $[H, \Pi] = 0$ and we can treat eigenspaces of Π independently. Since $\Pi^n = I$, we have that

$$\begin{aligned} \Pi|\psi_k\rangle &= \lambda_k|\psi_k\rangle \\ |\psi_k\rangle = \Pi^n|\psi_k\rangle &= \lambda_k^n|\psi_k\rangle \\ \implies \lambda_k = e^{\frac{2\pi i}{n}k}, &\qquad k = 0, 1, \ldots, n-1. \end{aligned}$$

Since the Hilbert space has size 2^n, these n eigenvalues are highly degenerate, but reducing the problem to n smaller problems can still provide a computational advantage.

The next important use of conserved quantities, relies on the evaluation of the trace. Suppose we evaluate the trace of $\hat{\mathcal{I}}(t)\hat{A}$ at time t, then

$$\begin{aligned} \mathrm{tr}(\hat{\mathcal{I}}(t)\hat{A}) &= \mathrm{tr}(\hat{\mathcal{U}}(t)^\dagger\hat{\mathcal{I}}(0)\hat{\mathcal{U}}(t)\hat{A}) && (4.27) \\ &= \mathrm{tr}(\hat{\mathcal{U}}(t)^\dagger\hat{\mathcal{U}}(t)\hat{\mathcal{I}}(0)\hat{A}) && (4.28) \\ &= \mathrm{tr}(\hat{\mathcal{I}}(0)\hat{A}) && (4.29) \end{aligned}$$

where in Equation (4.27) we have used the evolution equation of $\hat{\mathcal{I}}(t)$, given in Equation (2.32). Thus the trace of $\hat{\mathcal{I}}(t)\hat{A}$ must be conserved in time.

Finally, we attempt to use these ideas on our proposed $\hat{\mathcal{I}}(0), \hat{\mathcal{I}}(T)$ with the correct spectral properties. However unfortunately we find that the only conserved quantities in this case are $\{Z_1Z_2, Z_3Z_4, Z_1Z_2Z_3Z_4\}$, which when imposing $\mathrm{tr}(\hat{\mathcal{I}}(0)\hat{A}) = \mathrm{tr}(\hat{\mathcal{I}}(T)\hat{A})$, given in Section 4.4.1, all give 0. Thus we do not gain any information in this case.

Constructing Conserved Quantities

Conserved quantities can be found simply by trial and error, or more conveniently by looking at the symmetries in the Hamiltonian. Here we present an alternative method, that makes use of the Choi-Jamiołkowsky isomorphism [51]. The Choi-Jamiołkowsky isomorphism treats operators as states, known as their dual state, by making the following transformation. Take the maximally entangled state,

$$|x\rangle = \sum_{j=0}^{n} |j\rangle \otimes |j\rangle$$

noting we have left this unnormalised for convenience. Then given an operator $\hat{B}$, we transform such that

$$\hat{B} \mapsto \Big(\hat{B} \otimes I\Big)|x\rangle$$

where the identity, I, has the same dimensions as $\hat{B}$. We illustrate this transformation in the case $n = 1$,

$$\begin{aligned}
\hat{B} &= \begin{pmatrix} a & b \\ c & d \end{pmatrix} \\
\implies \Big(\hat{B} \otimes I\Big)|x\rangle &= \sum_{j=0}^{1} \begin{pmatrix} a & b \\ c & d \end{pmatrix} |j\rangle \otimes |j\rangle \\
&= \begin{pmatrix} a \\ c \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \begin{pmatrix} b \\ d \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\
&= \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix}.
\end{aligned}$$

A useful consequence of this transformation can be derived: let $|b_1b_2\rangle$ be a basis state, where $|b_1\rangle, |b_2\rangle$ each represent n spins. First we have,

$$\begin{aligned}
\langle b_1b_2|\hat{B} \otimes I|x\rangle &= \sum_{j=0}^{n} \langle b_1b_2|\hat{B} \otimes I|jj\rangle \\
&= \sum_{j=0}^{n} \langle b_1|\hat{B}|j\rangle\langle b_2|j\rangle \\
&= \langle b_1|\hat{B}|b_2\rangle.
\end{aligned}$$

Similarly,

$$\begin{aligned}
\langle b_1b_2|I \otimes \hat{B}^T|x\rangle &= \sum_{j=0}^{n} \langle b_1b_2|I \otimes \hat{B}^T|jj\rangle \\
&= \sum_{j=0}^{n} \langle b_1|j\rangle\langle b_2|\hat{B}^T|j\rangle \\
&= \langle b_2|\hat{B}^T|b_1\rangle.
\end{aligned}$$

Thus,

$$\hat{B} \otimes I|x\rangle = I \otimes \hat{B}^T|x\rangle.$$

Now suppose we wish to find an operator that commutes with $\hat{B}$. Observe

$$\Big([\hat{B}, \hat{A}] \otimes I\Big)|x\rangle = \Big(\hat{B}\hat{A} \otimes I - \hat{A}\hat{B} \otimes I\Big)|x\rangle \tag{4.30}$$

$$= \Big((\hat{B} \otimes I)(\hat{A} \otimes I) - (\hat{A} \otimes I)(\hat{B} \otimes I)\Big)|x\rangle \tag{4.31}$$

$$= \Big((\hat{B} \otimes I)(\hat{A} \otimes I) - (\hat{A} \otimes I)(I \otimes \hat{B}^T)\Big)|x\rangle \tag{4.32}$$

$$= \Big((\hat{B} \otimes I) - (I \otimes \hat{B}^T)\Big)(\hat{A} \otimes I)|x\rangle, \tag{4.33}$$

where in Equation (4.33), we used that $\hat{A} \otimes I$ and $I \otimes \hat{B}^T$ commute. Now if we look at the kernel of $\hat{B} \otimes I - I \otimes \hat{B}^T$, we find a state $(\hat{A} \otimes I)|x\rangle$ such that

$$\Big((\hat{B} \otimes I) - (I \otimes \hat{B}^T)\Big)(\hat{A} \otimes I)|x\rangle = 0.$$

Clearly if the dual of an operator is 0, then the operator itself is 0. Thus

$$(\hat{A} \otimes I)|x\rangle \in \ker\left(\hat{B} \otimes I - I \otimes \hat{B}^T\right) \implies [\hat{B}, \hat{A}] = 0.$$

As a result we have found an operator $\hat{A}$ that commutes with $\hat{B}$. In our case a conserved quantity is found by finding an operator that commutes with R_k for $k = 1, \ldots, m$. Thus we would need to find the kernels of $\hat{R}_k \otimes I - I \otimes \hat{R}_k^T$ and find their intersection. For small systems this is fairly straight forward. For example in the system of two spins where

$$\mathbf{R} := \Big\{Z_1 + Z_2, X_1X_2\Big\},$$

then using the Choi-Jamiołkowsky procedure above gives an intersection

$$Sp\Big\{|0000\rangle + |1111\rangle, |0101\rangle + |1010\rangle, |0110\rangle + |1001\rangle\Big\}.$$

It is a good check to see that the identity is in this set.

While we have shown here an elegant method for finding conserved quantities, it is not computationally ideal. Indeed, dynamic invariants transitioned us from states to operators, which increased the dimensions quadratically. Now using Choi-Jamiołkowsky, we are returning to states, working with the kernel of an operator that is now again quadratically larger. In addition, this method loses the inherent Pauli decomposition we have used to make computation efficient, as presented in Section 4.2.

4.4.3 Explicit Time-Evolution of $\hat{\mathcal{I}}$

The final method for ensuring we have a valid time evolution for $\hat{\mathcal{I}}(t)$, is to simply solve the evolution equation directly. In Section 2.3.2, we discussed how the invariant evolves in detail and in some cases this can be determined analytically [27]. For the Lie algebraic case in Example 0, presented in Section 4.3.2, it was found in [17] that the time-evolution operator can be found explicitly such that the invariant end points are defined,

$$\hat{\mathcal{I}}(0) = \sum_{j=1}^{n} Z_j \qquad \hat{\mathcal{I}}(T) = \sum_{j=1}^{n-1} X_jX_{j+1} - \prod_{j=1}^{n} Z_j.$$

For this example, the corresponding $\hat{\mathcal{H}}_T$ can be chosen as $\hat{\mathcal{H}}_T = \hat{\mathcal{I}}(T)$, which has the GHZ state as eigenstate. Thus there exists a viable time evolution from the eigenstate of $I(0)$ to the GHZ state.

Alternatively, recognising that there exists a unitary operator such that $\hat{\mathcal{I}}(t) = \hat{\mathcal{U}}(t)\hat{\mathcal{I}}(0)\hat{\mathcal{U}}^\dagger(t)$ also allows for optimisation methods. For our example, given that in the Hamiltonian set, we have two commuting subsets of operators, we could suppose the unitary operator can be built from

$$\hat{U}(\alpha_1, \alpha_2) = \exp\Big(i\alpha_1 X_1 X_2 + i\alpha_2 X_3 X_4\Big) \qquad \hat{V}(\beta_1, \ldots, \beta_4) = \exp\Big(\sum_{j=1}^{4} i\beta_j Z_j\Big),$$

which can be written for computational ease as

$$\hat{U}(\alpha_1, \alpha_2) = \Big(cos(\alpha_1) + iX_1X_2 sin(\alpha_1)\Big)\Big(cos(\alpha_2) + iX_3X_4 sin(\alpha_2)\Big)$$
$$\hat{V}(\beta_1, \ldots, \beta_4) = \prod_{j=1}^{4}\Big(cos(\beta_j) + iZ_j sin(\beta_j)\Big).$$

In this form, optimisation techniques could readily be applied, for example to minimise the following with respect to the parameters $\boldsymbol{\alpha}, \boldsymbol{\beta}$

$$||V(\boldsymbol{\beta})U(\boldsymbol{\alpha})\hat{\mathcal{I}}(0)V(\boldsymbol{\beta})^\dagger U(\boldsymbol{\alpha})^\dagger - \hat{\mathcal{I}}(T)||_F$$

where the norm chosen is the Frobenius norm, $||A||_F := \sqrt{\text{tr}(AA^\dagger)}$. This will be discussed briefly in Chapter 5, where we outline possible future directions of research.

Chapter 5

Conclusion and Outlook

From the the analysis of Shor's algorithm to the discussion of vulnerable ciphers, this dissertation has given considerable incentive for the research into post-quantum cryptography. Working with dynamic invariants enabled us to tackle a lattice problem underpinning a proposed post-quantum cryptographic protocol. We worked with several examples, each with their own advantages and disadvantages, illustrating the flow of thought that led from one experiment to the next.

Starting with a recap of classical computation, we presented the necessary foundations to work with quantum computation models, illustrating their analogues with classical theory. This took us from Deutsch's model through adiabaticity to using dynamic invariants. Deutsch's model has had many successes and we presented Shor's algorithm in detail, showing that a quantum computer can factorise a composite number exponentially faster than a classical computer [2]. Replacing the discrete unitary transformations characterising the evolution of a state in the Deutsch model, with continuous evolution subject to a time-dependent Hamiltonian, gives rise to new computational capabilities. In particular, adiabatic quantum computation ensures we can realise an eigenstate of a problem Hamiltonian, at the cost of evolving time slowly enough. We were able to dismiss this limitation by using dynamic invariants, instead being faced with potentially catastrophic scaling [27]. Searching for the instances where the scaling of dynamic invariants is favourable, is the key to using dynamic invariants effectively.

We next turned to cryptography to find a suitable problem to employ dynamic invariants in. We found the history of cryptography relies heavily on foundational mathematical problems, so it was no surprise that an NP-complete problem helped devise the top candidates for cryptography in the post-quantum era [14]. Thus, working on the shortest vector problem, we derived a problem Hamiltonian that, if realised, could solve the problem [15]. They key feature of this Hamiltonian was the long range interactions, Z_jZ_k, between spins.

Inspired by the work of others, we worked with spin chains to try to derive these long range interactions. In particular, [17] proposed using a Lie algebraic approach to reduce the scaling of the problem from the exponentially growing Hilbert space to the size of the algebra. It was also suggested that this scaling could be further improved by only taking nested commutations with the Hamiltonian into account. From here we turned to experimenting with various boundary conditions. We began by ensuring the invariant at time T would have the desired long range interactions. However we found for our case, that this resulted in needing long range interactions to begin with. Thus we devised a procedure to create long range interactions from nearest neighbour interactions and adjusted the Hamiltonian and invariant sets to guarantee a more useful invariant at time 0. While the long range interactions were achieved, the set blew up such that any favourable scaling was lost. This then led to the search of a compromise. We reduced the goal of interacting any spins j, k, to fixing k and requiring interactions with the mth-nearest neighbour for $m \leq k$. This set was found to have a linear scaling in n.

The next task was to ensure there exists a viable time evolution from the initial invariant, i.e. nearest neighbour interactions, to the final invariant, i.e. mth-nearest neighbour interactions. This resulted in the derivation of several more properties of invariants found in the literature as well as making new use of the Choi-Jamiołkowsky isomorphism. Employing these on our examples found, we were able to define several constraints on the possible time evolutions of the invariant.

Looking ahead for our particular, final example, further research could use optimisation methods, as suggested in Section 4.4.3, to find explicit evolutions of potential invariants. Alternatively following the analytic constraints found, the available final invariants could be narrowed down further via analytic techniques, for example using pseudoinverses given the explicit evolution of the Hamiltonian expressed with $\mathbf{a}(t)$, as used in [50]. While the example found scales well in n, it does not scale well in k. Thus it is no longer applicable to the cryptographic problem Hamiltonian requiring long range interactions where n and k increase simultaneously. However, in k satisfiability problems, also determined as NP-complete [11], the relevant problem Hamiltonian is k-local. This means we would need to realise a Hamiltonian that is the sum of m operators acting on fixed k qubits each. With additional constraints on the operators, we could employ the set we found.

Looking at the wider picture, using a similar procedure to derive the long range interactions as in our examples, we could further adjust the boundary conditions to find new sets with new properties. Alternatively, we could turn to different lattice problems, and find other instances where a Lie algebraic or reduced Lie algebraic approach is advantageous.

Bibliography

[1] Daniel J. Bernstein and Tanja Lange. Post-quantum cryptography—dealing with the fallout of physics success. Cryptology ePrint Archive, Paper 2017/314, 2017. https://eprint.iacr.org/2017/314.

[2] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, October 1997.

[3] Lidong Chen, Stephen Jordan, Yi-Kai Liu, Dustin Moody, Rene Peralta, Ray Perlner, and Daniel Smith-Tone. Report on post-quantum cryptography, 2016-04-28 2016.

[4] David E. Deutsch and Roger Penrose. Quantum computational networks. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 425(1868):73–90, 1989.

[5] D Deutsch and R Jozsa. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 439:553 – 558, 1992.

[6] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 212–219, New York, NY, USA, 1996. Association for Computing Machinery.

[7] Gilles Brassard, Peter Hoyer, and Alain Tapp. *Quantum cryptanalysis of hash and claw-free functions: Invited paper*, page 163–169. Springer Berlin Heidelberg, 1998.

[8] John Preskill. Quantum computing in the NISQ era and beyond. *Quantum*, 2, 2018.

[9] David Joseph, Rafael Misoczki, Marcos Manzano, Joe Tricot, Fernando Dominguez Pinuaga, Olivier Lacombe, Stefan Leichenauer, Jack D. Hidary, Phil Venables, and Royal Hansen. Transitioning organizations to post-quantum cryptography. *Nature*, 605:237 – 243, 2022.

[10] Dorit Aharonov, Wim van Dam, Julia Kempe, Zeph Landau, Seth Lloyd, and Oded Regev. Adiabatic quantum computation is equivalent to standard quantum computation, 2005.

[11] Tameem Albash and Daniel A. Lidar. Adiabatic quantum computation. *Reviews of Modern Physics*, 90(1), 2018.

[12] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, Joshua Lapan, Andrew Lundgren, and Daniel Preda. A quantum adiabatic evolution algorithm applied to random instances of an np-complete problem. *Science*, 292(5516):472–475, April 2001.

[13] IBM Quantum Learning. Practical introduction to quantum-safe cryptography. https://learning.quantum.ibm.com/course/practical-introduction-to-quantum-safe-cryptography, 2024. Accessed: 09.06.2024.

[14] Gorjan Alagic, David Cooper, Quynh Dang, Thinh Dang, John M. Kelsey, Jacob Lichtinger, Yi-Kai Liu, Carl A. Miller, Dustin Moody, Rene Peralta, Ray Perlner, Angela Robinson, Daniel Smith-Tone, and Daniel Apon. Status report on the third round of the nist post-quantum cryptography standardization process, 2022-07-05 04:07:00 2022.

[15] David Joseph, Adam Callison, Cong Ling, and Florian Mintert. Two quantum ising algorithms for the shortest-vector problem. *Physical Review A*, 103(3), 2021.

[16] David Joseph, Alexandros Ghionis, Cong Ling, and Florian Mintert. Not-so-adiabatic quantum computation for the shortest vector problem. *Physical Review Research*, 2(1), 2020.

[17] Modesto Orozco-Ruiz, Nguyen H. Le, and Florian Mintert. A way around the exponential scaling in optimal quantum control, 2024.

[18] Eleanor Kneip. Lie Algebras. https://github.com/e-kneip/Lie-algebras, 2024.

[19] Carlton Caves. Carlton Caves' Slogan. http://info.phys.unm.edu/~caves/research.html. Accessed: 22.05.2024.

[20] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*, pages 17–24. Cambridge University Press, Cambridge, 2000.

[21] R. Shankar. *Principles of Quantum Mechanics*, pages 115–143. Springer, New York, 2nd edition, 1994.

[22] Michael O Rabin. Probabilistic algorithm for testing primality. *Journal of Number Theory*, 12(1):128–138, 1980.

[23] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*, pages 226–237. Cambridge University Press, Cambridge, 2000.

[24] Unathi Skosana and Mark Tame. Demonstration of shor's factoring algorithm for N=21 on ibm quantum processors. *Scientific Reports*, 11(1), 2021.

[25] Ryan Barnett. Quantum Mechanics 2 Lecture Notes, 2023-24. [Lecture Series] Imperial College London, Module Code: MATH70018.

[26] Harry R. Lewis and W. B. Riesenfeld. An exact quantum theory of the time dependent harmonic oscillator and of a charged particle in a time dependent electromagnetic field. *Journal of Mathematical Physics*, 10:1458–1473, 1969.

[27] M.S. Sarandy, E.I. Duzzioni, and R.M. Serra. Quantum computation in continuous time using dynamic invariants. *Physics Letters A*, 375(38):3343–3347, September 2011.

[28] Xi Chen, A. Ruschhaupt, S. Schmidt, A. del Campo, D. Guéry-Odelin, and J. G. Muga. Fast optimal frictionless atom cooling in harmonic traps: Shortcut to adiabaticity. *Physical Review Letters*, 104(6), 2010.

[29] A. Tobalina, M. Palmero, S. Martínez-Garaot, and J. G. Muga. Fast atom transport and launching in a nonrigid trap. *Scientific Reports*, 7(1):5753, July 2017.

[30] Jennifer Shor and Peter Shor. Science News Poetry Contest Submission. https://math.mit.edu/~shor/notapoet.html. Accessed: 24.04.2024.

[31] Volker Strassen. International Congress of Mathematician's Introduction. https://math.mit.edu/~shor/notapoet.html, 1998. Accessed: 24.04.2024.

[32] Kathleen Richards. What is Cryptography? https://www.techtarget.com/searchsecurity/definition/cryptography, 2024. Accessed: 24.04.2024.

[33] Simon Singh. *The Code Book: The Evolution of Secrecy from Mary, Queen of Scots, to Quantum Cryptography*. Doubleday, New York, NY, 1999.

[34] Charles H. Bennett and Gilles Brassard. Quantum cryptography: Public key distribution and coin tossing. *Theoretical Computer Science*, 560:7–11, 2014.

[35] Kishore Thapliyal and Anirban Pathak. Kak's three-stage protocol of secure quantum communication revisited: hitherto unknown strengths and weaknesses of the protocol. *Quantum Information Processing*, 17(9), 2018.

[36] Dennis Luciano and Gordon Prichett. Cryptology: From caesar ciphers to public-key cryptosystems. *The College Mathematics Journal*, 18(1):2–17, 1987.

[37] Gorjan Alagic and Alexander Russell. Quantum-secure symmetric-key cryptography based on hidden shifts. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017*, pages 65–93, Cham, 2017. Springer International Publishing.

[38] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 26:96–99, 1978.

[39] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[40] Raphael Overbeck and Nicolas Sendrier. *Code-based cryptography*, pages 95–145. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[41] C. Peng, J. Chen, S. Zeadally, and D. He. Isogeny-based cryptography: A promising post-quantum technique. *IT Professional*, 21(06):27–32, 2019.

[42] Yang Li, Kee Siong Ng, and Michael Purcell. A tutorial introduction to lattice-based cryptography and homomorphic encryption. *ArXiv*, abs/2208.08125, 2022.

[43] M. Ajtai. Generating hard instances of lattice problems (extended abstract). In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 99–108, New York, NY, USA, 1996. Association for Computing Machinery.

[44] Daniele Micciancio. The shortest vector in a lattice is hard to approximate to within some constant. *SIAM Journal on Computing*, 30(6):2008–2035, 2001.

[45] Daniele Micciancio. The hardness of the closest vector problem with preprocessing. *IEEE Transactions on Information Theory*, 47(3):1212–1215, 2001.

[46] Oded Regev. The learning with errors problem (invited survey). In *2010 IEEE 25th Annual Conference on Computational Complexity*, pages 191–204, 2010.

[47] Yifeng Rocky Zhu, David Joseph, Cong Ling, and Florian Mintert. Iterative quantum optimization with an adaptive problem hamiltonian for the shortest vector problem. *Physical Review A*, 106(2), 2022.

[48] Eden Schirman, Cong Ling, and Florian Mintert. Finding short vectors in structured lattices with reduced quantum resources, 2023.

[49] Alan Kay. Alan Kay Quote. https://www.goodreads.com/author/quotes/304595.Alan_Kay. Accessed: 06.06.2024.

[50] E. Torrontegui, S. Martínez-Garaot, and J. G. Muga. Hamiltonian engineering via invariants and dynamical algebra. *Phys. Rev. A*, 89:043408, 2014.

[51] Min Jiang, Shunlong Luo, and Shuangshuang Fu. Channel-state duality. *Phys. Rev. A*, 87:022310, Feb 2013.