# Density Matrix Simulation of Quantum Error Correction

Arvid Rolander

LUND UNIVERSITY

Master Thesis
Thesis advisors: Andreas Walther, Adam Kinos
Examiner: Cord Arnold
September 23, 2020

# Abstract

Quantum error correction will be integral in developing full scale quantum computers, but as of yet beneficial quantum error correction has not been demonstrated experimentally. An important question is therefore what prerequisites need to be met to achieve this. Results of density matrix simulations of the performance of the seven qubit Steane code in a quantum computing setting are presented. The full density matrix was kept throughout the whole simulations, which means that all errors can be accounted for. In particular, the importance of the *circuit depth*, i.e. the number of gates in series before error correction is applied, for the overall performance was investigated. It was found that the depth of the circuit has a large impact on the threshold error rate for which error correction becomes beneficial. A gain parameter was defined, which describes the largest constant factor by which errors can be suppressed. It was shown that there is a trade-off between the threshold error and the gain; The highest threshold value was found to be around $p_{th} \approx 10^{-4}$ which is in line with other estimates, but the maximum gain for this value was only 3. To achieve a gain of 100, an error rate of $p_{err} \approx 2 \cdot 10^{-9}$ is required. In addition, performance statistics such as run-time as a function of circuit depth and width for the matlab code used for simulations are presented.

# Acknowledgements

I would like to thank my supervisors Andreas Walther and Adam Kinos, without whose help this project would not have been possible. I would like to give a special thanks to Andreas for his extensive help with finalising this report. I would also like to thank Katarina Wilhelmsen for reading through countless drafts of this report and providing invaluable feedback. Finally, I would like to thank my family and friends for their general support.

# Contents

**Abstract** **iii**

**Acknowledgements** **v**

**1 Introduction** **1**
- 1.1 Background . . . 1
- 1.2 Aims . . . 2
- 1.3 Delimitations . . . 3
- 1.4 Structure of the Report . . . 3

**2 Background Theory** **5**
- 2.1 Quantum Mechanics . . . 5
  - 2.1.1 Quantum Operations . . . 7
- 2.2 Quantum Logic and Circuits . . . 8
  - 2.2.1 Quantum Gates . . . 8
  - 2.2.2 Quantum Circuits . . . 10
- 2.3 Quantum Error Correction . . . 12
  - 2.3.1 Fundamentals of Quantum Error Correction . . . 12
  - 2.3.2 The Steane Code . . . 15
  - 2.3.3 The Threshold Theorem . . . 22

**3 Methods** **23**
- 3.1 Density Matrix Simulation of Quantum Circuits . . . 23
  - 3.1.1 Partial Trace . . . 24
  - 3.1.2 Sparse Matrix Operations . . . 24
  - 3.1.3 Gate Tolerance . . . 24
  - 3.1.4 Error Models . . . 25

3.2 Simulating Error Correction with the Steane Code . . . . . . . 26

**4 Results & Discussion 29**

4.1 Code Performance . . . . . . . . . . . . . . . . . . . . . . . . 29

4.1.1 Performance as a Function of Circuit Width . . . . . . 30

4.1.2 Performance as a Function of Circuit Depth . . . . . . 32

4.1.3 Impact of Gate Tolerance . . . . . . . . . . . . . . . . 34

4.2 Error Correction Simulations . . . . . . . . . . . . . . . . . . 36

4.2.1 Impact of Different Types of Errors . . . . . . . . . . . 36

4.2.2 Finding Optimal Gate Numbers . . . . . . . . . . . . 39

4.2.3 Fidelity Error Curves at Different Gate Numbers . . . . 41

**5 Conclusions & Outlook 47**

5.1 Conclusions . . . . . . . . . . . . . . . . . . . . . . . . . . . 47

5.2 Outlook . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 48

**6 Appendix 51**

6.1 Raw Data for Code Performance . . . . . . . . . . . . . . . . . 52

6.2 Additional Plots from Simulations . . . . . . . . . . . . . . . . 56

**7 Bibliography 63**

# Chapter 1

# Introduction

## 1.1 Background

One of the first mentions of Quantum Computers was by Richard Feynman in 1981, when he proposed that the only way to efficiently simulate a quantum system would be to use a quantum computer [1]. There are now examples of algorithms that could only run on a quantum computer that would provide an exponential speed up compared to their classical counterparts for some tasks, such as factoring prime numbers. Since then, the field has come a long way, with the first case of quantum supremacy being demonstrated by Google's team in October 2019 [2].

Much like the fundamental unit of information in classical computing is the *bit*, which can assume the states 0 and 1, the fundamental unit of information in quantum computing is the *qubit*, which can be in the states $|0\rangle$ and $|1\rangle$. What separates the qubit from the classical bit is that the qubit can also exist in a so called superposition of the basis states. An arbitrary state of the qubit is written $|\psi\rangle$, and any state can be written as a linear combination by

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,,$$

where $\alpha$ and $\beta$ are complex numbers and $|\alpha|^2 + |\beta|^2 = 1$ [3, chap. 1.2]. One qubit is hardly enough for actual computations, so a way to describe a collection of qubits is also needed. A system of several qubits $|\psi_1\rangle$, $|\psi_2\rangle$ ... can be described as a *product state* [3, pp. 94] by

$$|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes ...,$$

where $\otimes$ denotes a *tensor product*, see [3, chap. 2.1.7] for a more thorough description. In this thesis, I will for convenience use the short notation $|xx...x\rangle$ to write a basis vector state of multiple qubits. This way, any product state can be written as a sum of the basis vectors.

One major obstacle in realising a full scale quantum computer is the proneness to errors exhibited by quantum systems, suffering from both environmental noise and faulty operations limiting the length of calculations that can be performed. This obstacle was long thought to be insurmountable, until the field of quantum error correction was invented in the early 1990's, when Peter Shor discovered the first error correcting code for quantum systems. One of the major achievements of quantum error correction is the threshold theorem, which states that given that the probability $p$ for error of individual components is below a threshold value $p_{th}$, arbitrary accuracy can be achieved by concatenating error correcting codes [3].

The quantum information group in Lund aims to build a quantum computer using rare-earth ions in crystals, and thus far, no one has investigated which methods of error correction would be most suitable for these systems.

## 1.2 Aims

The purpose of this work is twofold. The first part is to produce code that can be used to simulate quantum circuits, including ones using error correcting schemes. Here, it is worth mentioning that other efforts have been made to create software to simulate quantum circuits, two of them being *ProjectQ* which is introduced in [4] and *Quantumsim* introduced in [5]. The reason why these existing frameworks were not used merits some discussion. As the title of this thesis suggests, using the density matrix approach over the state vector approach was one of the central ideas in this work, and although ProjectQ comes with large emulation and simulation capabilities, density matrix simulation has not yet been implemented at the time of writing. Quantumsim on the other hand does provide high-performance density matrix simulation capabilities. However, a major obstacle in using Quantumsim is that at the time of writing, it lacks a comprehensive user manual. This meant that it was difficult to assess whether Quantumsim offered all the features required for this work. Therefore, writing new code for simulations was deemed a more efficient use of time.

The second and more important purpose is to investigate how the fidelity of a quantum circuit is affected by various parameters, with a specific focus on the parameters of the rare-earth systems used in Lund. These parameters include:

- Qubit gate fidelity and duration
- Readout fidelity and duration
- Ground state coherence time, or T2

In light of the *threshold theorem* described in section 2.3.3, one question was also if the threshold could be achieved in the special case of the Steane Code.

## 1.3 Delimitations

Since time and resources are limited, some delimitations had to be made. The first obvious delimitation is the choice to use the Steane seven qubit code as the sole example of error correction, even if this is hardly state of the art. The sophistication of the error model used had to be limited, especially for gates acting on several qubits at the same time to create entanglement. This is mainly due to the fact that the rare-earth ion system is a quite new technology, and the specific errors affecting it have not yet been established.

## 1.4 Structure of the Report

I will start the report by briefly introducing the background theory needed to understand the rest of the report. This will be done briefly in an extended list of definitions in chapter 2, with references to other work for more detailed descriptions. I will then continue by describing in more detail the methods used in chapter 3, and considerations that were taken when writing the code. I will move on to presenting and discussing the results in chapter 4. The results will be divided into two parts, namely code performance in section 4.1 and results of the simulations in section 4.2. I will then briefly summarise the conclusions that were made in section 5.1, and finally move on to ideas for future work in section 5.2.

# Chapter 2

# Background Theory

In this chapter I will briefly introduce the background theory needed to understand the rest of the report. This will be done in an extended list of definitions, with references to other work for a more in-depth discussion. I will start by going over some fundamental topics from quantum mechanics, then move on to more general quantum information topics and finally give a brief description of some concepts from quantum error correction.

## 2.1 Quantum Mechanics

Throughout this work, the density matrix or density operator formulation of quantum mechanics is used, which differs from the state vector formulation in some key aspects. The necessary definitions follow below.

### Representation

It will be crucial for us to have a regular vector representation of quantum states and a matrix representation for regular operators. A single qubit is a two-level system, and a general state for a single qubit can be written as $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$. This state can be parameterised by the complex numbers $\alpha$ and $\beta$, and we can represent this state with a column vector by

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

### Density Operator/Density Matrix

The *density operator* or *density matrix* of a quantum system is a representation of the state of that system. If the system is in a so called *pure state* $|\psi\rangle$, the density operator is given by $\rho = |\psi\rangle \langle\psi|$. If the general form of $|\psi\rangle$ introduced above is used, the density operator for the pure state has the matrix representation

$$\rho = |\psi\rangle \langle\psi| := \begin{pmatrix} |\alpha|^2 & \alpha\beta^* \\ \beta\alpha^* & |\beta|^2 \end{pmatrix}$$

The density matrix formulation is especially useful when one wants to describe *mixed states* [3, pp. 100]. The density operator of a *joint system* made up of several subsystems is given by a tensor product. For example, the density operator for $n$ qubits, where the state of each individual qubit is $\rho_i$, is given by

$$\rho = \rho_1 \otimes \rho_2 \otimes \ldots \otimes \rho_n$$

[3, pp. 102]. The matrix representation of the product state can be found taking the *Kronecker product* of the matrices of the individual qubits [3, pp. 74].

### Pure and Mixed States

A quantum system is said to be in a *pure state* when the state of the system is known exactly. Pure states can be conveniently described by the state vector $|\psi\rangle$. A system is said to be in a *mixed state* if it can be in any number of *pure states* $|\psi_i\rangle$. The state vector formalism fails to describe these states, but in the density matrix formalism such a state can be conveniently described by the density matrix $\rho = \sum_i p_i |\psi_i\rangle \langle\psi_i|$, where $p_i$ is the probability for the system to be found in the *pure state* $|\psi_i\rangle$ [3, pp. 99].

### Evolution of Quantum States

One of the postulates of quantum mechanics is that the time evolution of a *closed* quantum system is described by a *unitary transformation*, that is a transformation $U$ such that $UU^\dagger = U^\dagger U = I$. In other words, if the initial state of the system is $|\psi\rangle$ at some point, the final state after some time has passed is $U|\psi\rangle$. In the density matrix language, this may be written as $\rho \to U\rho U^\dagger$ [3, pp. 81, pp. 99].

### Projective Measurement

A *projective measurement* is described by a Hermitian operator $M$ that has the eigendecomposition

$$M = \sum_m m P_m$$

where $P_m$ is the projection operator onto the eigenspace of $M$ with eigenvalue $m$. The operators $P_m$ are orthogonal projectors, meaning that they satisfy the relation $P_m P_n = \delta_{mn}$ where $\delta_{mn}$ is the Kronecker delta. The probability of measuring outcome $m$ is given by $p(m) = \langle\psi| P_m |\psi\rangle$. The state immediately after measuring $m$ is $\frac{P_m|\psi\rangle}{\sqrt{m}}$ [3, chap. 2.2.5].

### The Reduced Density Operator

Sometimes it is convenient to study the individual parts of a system separately, for example in a situation when some qubits are used to represent data and others are used just for readout and are then discarded, so called *ancilla qubits*. This can be achieved within the density operator formalism by using the *reduced density matrix*. Suppose that you have a composite system $AB$ described by a density matrix $\rho^{AB}$ and you want to study subsystem $A$. The reduced density operator for system $A$ is defined by

$$\rho^A \equiv \mathrm{tr}_B(\rho^{AB}),$$

where the $\mathrm{tr}_B$ is called the *partial trace* over system $B$, and is defined by

$$\mathrm{tr}_B\left(|a_i\rangle\langle a_j| \otimes |b_k\rangle\langle b_l|\right) \equiv |a_i\rangle\langle a_j| \,\mathrm{tr}\left(|b_k\rangle\langle b_l|\right). \tag{2.1}$$

A more thorough description of the reduced density operator and the partial trace can be found in [3, chap. 2.4.3].

## 2.1.1 Quantum Operations

Because we want to be able to describe more general dynamics than just unitary evolution and projective measurements, such as stochastic noise and environmental decoherence, we need to turn to the *quantum operations* formalism. A quantum operation is defined as a map

$$\rho' = \mathcal{E}(\rho),$$

where $\mathcal{E}$ is a quantum operation. $\mathcal{E}$ can include both unitary transformations and projective measurements, but more generally it can include interactions with an environment. Quantum operations have a particularly useful representation, called the *operator-sum representation*, that will be used in this thesis. It turns out that any quantum operation $\mathcal{E}(\rho)$, which can include environmental interactions, can be represented as

$$\mathcal{E}(\rho) = \sum_k E_k \rho E_k^\dagger,$$

where the $E_k$:s are operators acting on the principal system. For an in-depth description of quantum operations and the operator-sum representation, the reader can refer to [3, chap. 8.2.1-8.2.2].

## 2.2 Quantum Logic and Circuits

Just like algorithms for a regular computer can be described using low level operations or gates on bits, such as the NOT, OR, and AND gates, quantum algorithms can be described in a similar way. Moreover, circuit diagrams can be used as a convenient means to represent algorithms graphically. In this section, I will briefly introduce the quantum gates that are used in this thesis together with an introduction to the quantum circuit diagrams. A more in-depth discussion of these topics can be found in [3, Chap. 1.3].

### 2.2.1 Quantum Gates

The most fundamental gates used in this work are the *Pauli Gates* that can be represented by the Pauli matrices. In this work, they together with the *Hadamard gate* and their respective *controlled* versions are the only gates used.

### Pauli Gates and Eigenstates

There are four Pauli matrices, often denoted by $\sigma_i$, but here we denote them by $I$, $X$, $Y$ and $Z$. They are defined as follows:

$$\begin{aligned} I &\equiv \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \\ X &\equiv \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \\ Y &\equiv \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \\ Z &\equiv \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \end{aligned}$$

The $I$ gate is simply the identity operation. The $X$ gate is analogous to a classical NOT gate, as it interchanges $|0\rangle$ and $|1\rangle$. The $Z$ gate is sometimes referred to as the *phase flip* operation, as it flips the phase of the $|1\rangle$ state. The $Y$ gate works as a combination of the $X$ and $Z$ gates, as it can be written as a product of the $X$ and $Z$ matrices. The *computational basis states* $|0\rangle$ and $|1\rangle$ can be identified as eigenstates of the $Z$ operator. A simple calculation will show that the states

$$\begin{aligned} |+\rangle &\equiv \frac{1}{\sqrt{2}} \left( |0\rangle + |1\rangle \right), \\ |-\rangle &\equiv \frac{1}{\sqrt{2}} \left( |0\rangle - |1\rangle \right) \end{aligned}$$

are eigenstates of the $X$ operator.

### The Hadamard Gate

The matrix representation of the *Hadamard gate* is given by

$$H \equiv \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

The action of the Hadamard gate on the computational basis states is to interchange $|0\rangle$ and $|+\rangle$, and $|1\rangle$ with $|-\rangle$.

**Controlled Gates**

A controlled gate takes a *control qubit* and a *target qubit* as input, and will do something to the target only if the control qubit is in the $|1\rangle$ state. There are two controlled gates we use in this work: the *controlled-NOT gate*, or CNOT gate, and the *controlled-Z* gate, or the CZ gate. The CNOT gate is a controlled version of the $X$ gate, and the CZ gate is a controlled version of the $Z$ gate. The matrix representations of these gates for a system of just a target and control bit are given by

$$\text{CNOT} \equiv \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix},$$

$$\text{CZ} \equiv \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix},$$

where the qubits are indexed from left to right. In the case of a system of three or more qubits where you want to perform a CNOT with for example qubit 1 as the control and qubit three as the target, the operation matrix can not be constructed by taking a tensor product between the operation matrix given above and another matrix. Instead, we have to consider the action of the CNOT: if the control bit is in state $|0\rangle$, do nothing, otherwise exchange $|0\rangle$ and $|1\rangle$. In the matrix representation, this can be seen as simply permuting the rows where the control qubit is in state $|1\rangle$. A similar rule can be used to construct the CZ matrix, with the difference being that the rows where the control and target are in state $|1\rangle$ get a sign change.

### 2.2.2 Quantum Circuits

We will need a good way to describe quantum algorithms in terms of gates and qubits, and for this we will use quantum circuit diagrams. Figure 2.1 below shows a simple example circuit where a $Z$ gate is applied to qubit $a$ which is then measured.

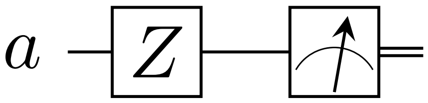

Figure 2.1: A simple example circuit.

In the diagram, qubits are represented by wires and gate operations have specific symbols. The symbols for the Pauli $Z$ gate and the measurement operation have already been shown, and the other Pauli gates will simply be represented by their corresponding letter in the same fashion as above. Similarly, the Hadamard gate will be represented by the letter $H$. Figure 2.2 shows a CNOT gate between two qubits.

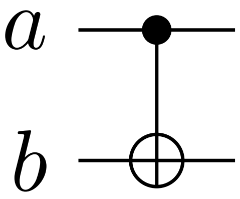

Figure 2.2: A CNOT gate with qubit *a* as control and qubit *b* as target

In figure 2.2, qubit *a* is the control qubit. Controlled gates are always denoted in the same way, with wires going from the control qubits to the target, which has a gate symbol. Figure 2.3 shows a generalised controlled $Z$ gate, with qubits *a* and *b* as controls.

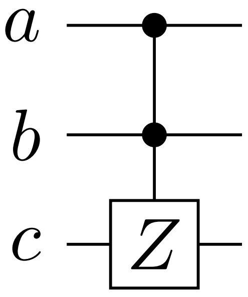

Figure 2.3: A generalised controlled $Z$ gate with qubits $a$ and $b$ as controls.

## 2.3 Quantum Error Correction

In this section, we will introduce the topic of *Quantum Error Correction* and the *Steane Code*, the error correcting code studied in this thesis, as well as the *Fidelity*, our main tool for checking how close two quantum states are.

**Fidelity**

The definition of fidelity varies between different works, but I will use the definition found in [3, Chap. 9.2.2]. The fidelity $\mathcal{F}$ between a pure state $|\psi\rangle$ and a density matrix $\rho$ is defined as

$$\mathcal{F}(|\psi\rangle, \rho) = \sqrt{\langle\psi| \rho |\psi\rangle}. \tag{2.2}$$

The Fidelity can take values in the interval $[0, 1]$, where a fidelity of 0 means $\psi$ and $\rho$ are orthogonal, and a fidelity of 1 means that they are the same state. Because we will work with fidelities very close to 1, a more practical measure is the *fidelity error*, $\epsilon_{fid} = 1 - \mathcal{F}(|\psi\rangle, \rho)$, which will be referred to as the *fidelity error*.

### 2.3.1 Fundamentals of Quantum Error Correction

There are three fundamental barriers to developing quantum error correcting codes which are not present for classical computers, namely

- **The No Cloning Theorem**, which states that it is impossible to make a copy of an arbitrary unknown quantum state [3, p. 532].

- The fact that quantum states exist in a continuum, which means that the errors affecting them are also continuous in nature.
- The fact that measurements destroy quantum superposition states.

Thankfully, these problems have neat workarounds. These workarounds will be described below, and a more detailed description can be found in [3, Chap. 10.1-10.2].

### Subspace Encodings & Syndrome Measurements

A basic idea in classical error correction is that of introducing redundancy in a system to protect against errors. We illustrate this with an example: Suppose we have one bit of information, that can be in state 0 or 1, that we want to transmit. Because of noise, there is a probability that the bit will flip to the other value during transmission, causing a 0 to become a 1 and vice versa. To protect against this, we make two copies of the bit and instead transmit either 000 or 111. Provided only one of the transmitted bits flip, we can deduce the original state and recover the lost information. This simple code can be adapted to the quantum case through what is called a *subspace encoding*, given by

$$\begin{aligned} |0\rangle &\to |0\rangle_L \equiv |000\rangle \\ |1\rangle &\to |1\rangle_L \equiv |111\rangle \,, \end{aligned}$$

where the subscript $L$ denotes a logical state An arbitrary superposition state can be encoded by

$$|\psi\rangle = a\,|0\rangle + b\,|1\rangle \to a\,|000\rangle + b\,|111\rangle \,.$$

A circuit that accomplishes this is shown in figure 2.4.

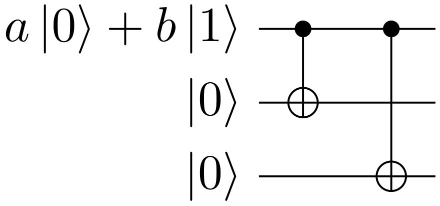

Figure 2.4: Circuit performing a basic subspace encoding.

Suppose that the Pauli $X$ operator is applied to the encoded qubits with a certain probability. We get four possible *error syndromes* that we can correct for:

$$\begin{aligned}
P_0 &= |000\rangle\langle 000| + |111\rangle\langle 111| \text{ No error}\\
P_1 &= |100\rangle\langle 100| + |011\rangle\langle 011| \text{ First qubit flipped}\\
P_2 &= |010\rangle\langle 010| + |010\rangle\langle 010| \text{ Second qubit flipped}\\
P_3 &= |001\rangle\langle 001| + |110\rangle\langle 110| \text{ Third qubit flipped.}
\end{aligned}$$

It is important to note that the measurement of the syndrome does not tell us anything about the underlying state i.e. the constants $a$ and $b$, only that an error has occurred. This means that the state isn't actually destroyed, which was another problem we had to deal with.

**Discretisation of Errors**

Suppose that we have a code that can correct both bitflip and phaseflip errors on one qubit, i.e. that can detect and correct both Pauli $X$ and $Z$ errors. It turns out that this is enough to correct an arbitrary error on one qubit. To see this, recall that any quantum operation has an *operator-sum representation*, that is

$$\mathcal{E}(\rho) = \sum_i E_i \rho E_i^\dagger.$$

The four Pauli matrices form a basis for the space of $2 \times 2$-matrices, so if only one bit is affected, each operation element $E_i$ can be written in the form

$$E_i = I \otimes I \otimes \ldots \otimes (a_i I + b_i X + c_i Y + d_i Z) \otimes \ldots \otimes I.$$

Noting that $Y = iXZ$ we can instead write this

$$E_i = I \otimes I \otimes \ldots \otimes (a_i I + b_i X + c_i XZ + d_i Z) \otimes \ldots \otimes I.$$

Suppose we have a collection of qubits in the total state $|\psi\rangle = |\psi'\rangle \otimes |\phi\rangle$ where we take $|\psi'\rangle$ to be one qubit that we want to operate on and $|\phi\rangle$ to be the rest of the system. We get

$$\begin{aligned}
\mathcal{E}(|\psi\rangle) &= \left[\sum_i (a_i I + b_i X + c_i XZ + d_i Z) \left|\psi'\right\rangle\right] \otimes |\phi\rangle\\
&= a\left|\psi'\right\rangle \otimes |\phi\rangle + bX\left|\psi'\right\rangle \otimes |\phi\rangle + cXZ\left|\psi'\right\rangle \otimes |\phi\rangle + dZ\left|\psi'\right\rangle \otimes |\phi\rangle .
\end{aligned}$$

We can now construct a syndrome measurement that collapses $\mathcal{E}(|\psi\rangle)$ into one of the components $|\psi'\rangle \otimes |\phi\rangle$, $X|\psi'\rangle \otimes |\phi\rangle$, $XZ|\psi'\rangle \otimes |\phi\rangle$ and $Z|\psi'\rangle \otimes |\phi\rangle$. Since all of these terms only contain $X$ and $Z$ errors, we can correct them.

### Error Models

Equipped with some basics of quantum logic, we can now start with some simple error models that are used here, following [3, Chap 8.3]. For this, we use the operator-sum representation of quantum operations. With error discretisation in mind, a very simple way to model an imperfect quantum gate is to say that the gate is applied perfectly, but is then followed by a bitflip, phaseflip or combination error with a certain probability. This way, the total operation for a gate $G$, with total error probability $p$ becomes

$$G^*(\rho) = (1-p)G\rho G^\dagger + \frac{p}{3}(XG\rho G^\dagger X + YG\rho G^\dagger Y + ZG\rho G^\dagger Z).$$

For gates involving several qubits, we can use combinations of the basic $X$, $Y$ and $Z$ errors on all qubits instead. Errors that can not be described in the state vector formalism are the phase damping and amplitude damping channels. The operation elements of the amplitude damping channel are

$$E_0 = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{1-\gamma} \end{pmatrix}, E_1 = \begin{pmatrix} 0 & \sqrt{\gamma} \\ 0 & 0 \end{pmatrix}.$$

The effect of this channel on a density matrix is to move the state towards the ground state $|0\rangle$, and it can be used to describe energy dissipation or spontaneous decay. The operation elements for the phase damping channel are given by

$$E_0 = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{1-\lambda} \end{pmatrix}, E_1 = \begin{pmatrix} 0 & 0 \\ 0 & \sqrt{\lambda}. \end{pmatrix}$$

This channel can model the loss of knowledge of the phases between the components in a superposition.

## 2.3.2 The Steane Code

The Steane Code is the main code studied in this thesis, and uses 7 physical qubits to encode a logical qubit. It can be used to correct one arbitrary error on a single qubit, or one phaseflip and one bitflip error on different qubits. Instead of defining the logical basis states directly, the Steane Code can be described completely by six operators, called stabilisers, acting on the 7 qubits that make up the logical state.

These are:

$$\begin{aligned}
K_1 &\equiv IIIXXXX,\\
K_2 &\equiv XIXIXIX,\\
K_3 &\equiv IXXIIXX,\\
K_4 &\equiv IIIZZZZ,\\
K_5 &\equiv ZIZIZIZ,\\
K_6 &\equiv IZZIIZZ.
\end{aligned}$$

The logical basis states $|0\rangle_L$ and $|1\rangle_L$ are defined as simultaneous $+1$ eigenstates of the stabilisers. To illustrate why it is practical to define the code in terms of these operators, the states $|0\rangle_L$ and $|1\rangle_L$ for the Steane code are given by

$$\begin{aligned}
|0\rangle_L = \frac{1}{\sqrt{8}}(\,&|0000000\rangle + |1010101\rangle + |0110011\rangle + |1100110\rangle\\
&+ |0001111\rangle + |1011010\rangle + |0111100\rangle + |1101001\rangle),\\
|1\rangle_L = \frac{1}{\sqrt{8}}(\,&|1111111\rangle + |0101010\rangle + |1001100\rangle + |0011001\rangle\\
&+ |1110000\rangle + |0100101\rangle + |1000011\rangle + |0010110\rangle).
\end{aligned}$$

These can be very cumbersome to work with. Moreover, as will be seen later, error correction and state preparation with the Steane code can be performed by measuring the stabilisers in a smart manner. Once the basis states $|0\rangle_L$ and $|1\rangle_L$ have been defined, one can also define logical equivalents of common operators, like the Pauli and Hadamard operators.

### Logical Operators

In the case of the Steane code, the logical equivalents of the operators that are used in this work, namely the Pauli and Hadamard operators have a very simple so called *transversal* form. To perform a logical $H$-operation, one simply applies the one bit $H$-operator to all the data qubits separately. The same holds for the Pauli operators. Thus, we can write e.g.

$$H_L = HHHHHHH.$$

### Parity Measurements

To perform state preparation and error correction with the Steane Code, we need a way to measure the stabilisers without destroying information. The type of measurement used for this is called a *parity measurement*. Parity measurements give us a way to measure operators without destroying information. Figure 2.5 shows a circuit to perform a parity measurement of an arbitrary operator $K$ on four qubits.

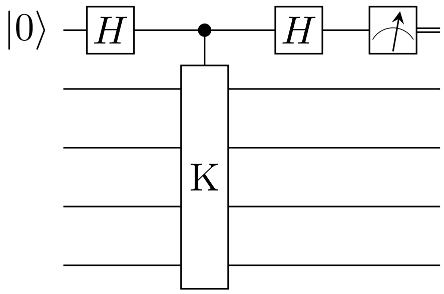

Figure 2.5: Circuit to perform a parity measurement of the $K$ operator.

This type of measurement does not destroy any information, since the measurement result only tells us if the state of the system is a $+1$ or $-1$ eigenstate of the operator, not what the eigenstate actually is. To see this, we write out the action of the circuit in figure 2.5: Let the data qubits be in the state $|\psi\rangle$, so that the total state of the system with ancilla and data qubits at the beginning of the circuit is

$$\left|\psi'\right\rangle = |0\rangle \otimes |\psi\rangle \, .$$

The first Hadamard gate acts on the ancilla, so that the total state is now

$$\left|\psi'\right\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |\psi\rangle \, .$$

Note that $|\psi'\rangle$ is not the same state, it is just used as a label. Applying the controlled $K$ operation we get

$$\left|\psi'\right\rangle = \frac{1}{\sqrt{2}}(|0\rangle \otimes |\psi\rangle + |1\rangle \otimes K\,|\psi\rangle),$$

and applying the second Hadamard gate gives

$$|\psi'\rangle = \frac{1}{2}|0\rangle \otimes (|\psi\rangle + K|\psi\rangle) + \frac{1}{2}|1\rangle \otimes (|\psi\rangle - K|\psi\rangle).$$

Noting now that $K$ is both unitary and Hermitian, we see that

$$\begin{aligned} K\frac{1}{\sqrt{2}}(|\psi\rangle + K|\psi\rangle) &= \frac{1}{\sqrt{2}}(|\psi\rangle + K|\psi\rangle, \\ K\frac{1}{\sqrt{2}}(|\psi\rangle - K|\psi\rangle) &= -\frac{1}{\sqrt{2}}(|\psi\rangle - K|\psi\rangle), \end{aligned}$$

so measuring the ancilla in $|0\rangle$ forces the system into a $+1$-eigenstate of $K$ and measuring the ancilla in $|1\rangle$ forces the system into a -1-eigenstate of $K$.

**Error Correction**

With the stabiliser operators and parity measurements in hand, we are ready to design a circuit to perform error correction with the Steane code. The strategy will be to measure all of the operators $K_i$, and then apply $X$ and $Z$ gates to the appropriate qubits to place the system in the correct eigenstate. Figure 2.6 shows a circuit that corrects one phaseflip error using the Steane code.

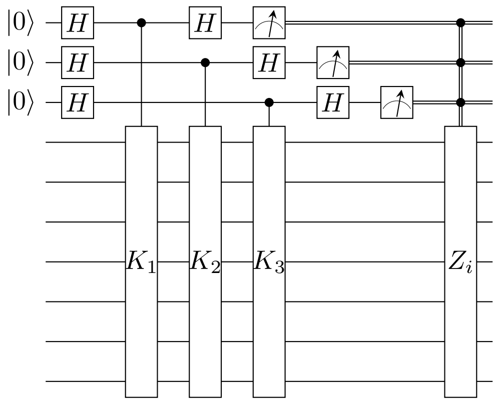

Figure 2.6: A circuit to correct one phaseflip error with the Steane code.

To see why $K_1 - K_3$ are used to find phaseflip errors, note that the $X$ and $Z$ operators anti-commute. When using the code for error correction, we assume that the state $|\psi\rangle$ of the data qubits is already a $+1$-eigenstate of $K_i$. Using $K_1$

as an example, let us say that there is a phaseflip error on the 7th qubit, so that we have the state $Z_7 |\psi\rangle$. Since $|\psi\rangle$ is a $+1$-eigenstate of $K_1$, the state $|\psi'\rangle$ after applying the second Hadamard gate in the parity measurement is

$$\begin{aligned} |\psi'\rangle &= \frac{1}{2}|0\rangle \otimes (Z_1|\psi\rangle - Z_1K_1|\psi\rangle) + \frac{1}{2}|1\rangle \otimes (Z_1|\psi\rangle + Z_1K_1|\psi\rangle) \\ &= |1\rangle \otimes Z_1|\psi\rangle, \end{aligned}$$

so performing the measurement will automatically place the system in a $-1$-eigenstate of $K_1$. The basic idea behind the circuit is then that three measurements are enough to determine which bit has a phaseflip error. To correct the error a $Z$ gate is then applied on that bit.

## Error Propagation & Fault Tolerance

The circuit in figure 2.6 would work if we had perfect gates apart from the error, and could prepare perfect ancilla qubits. However, in a real scenario we also want to limit how errors propagate. A circuit is called *fault tolerant* if it is designed in a way such that an error on one qubit can propagate to at most one other qubit. To achieve fault tolerance, we need to alter the circuit in figure 2.6. Figure 2.7 shows a circuit which applies a controlled stabiliser operation fault tolerantly and then measures it.

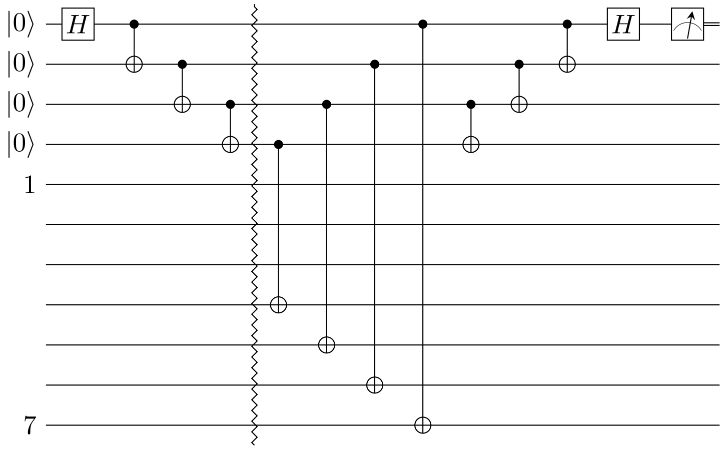

Figure 2.7: A circuit that applies a controlled stabiliser operator of the Steane code fault tolerantly and then measures it.

This is not enough for fault tolerance as the ancilla preparation, shown in the part of figure 2.7 to the left of the zig-zag line, is not fault tolerant. Figure 2.8 shows how to initialise the ancilla qubits fault tolerantly.

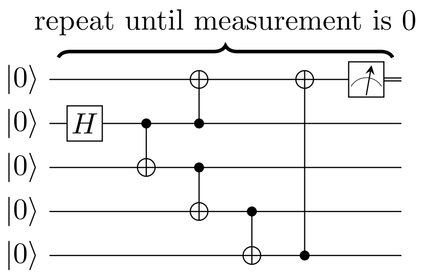

Figure 2.8: Fault tolerant ancilla preparation.

Here, the circuit is repeated until the top qubit is measured as $|0\rangle$. To make the circuit more clear, it is practical to introduce the equivalent operation shown in figure 2.9.

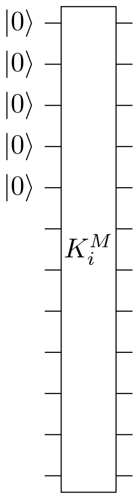

Figure 2.9: Circuit element representing a fault tolerant application of the operator $K_i$, with subsequent parity measurement. This includes fault tolerant ancilla preparation. This is the combination of the operations shown in figure 2.7 and figure 2.8

There is one more thing that requires consideration: The fact that the measurements themselves might fail. To combat this, majority voting is used to determine if the measurement result is correct. The complete fault tolerant circuit is shown in figure 2.10.

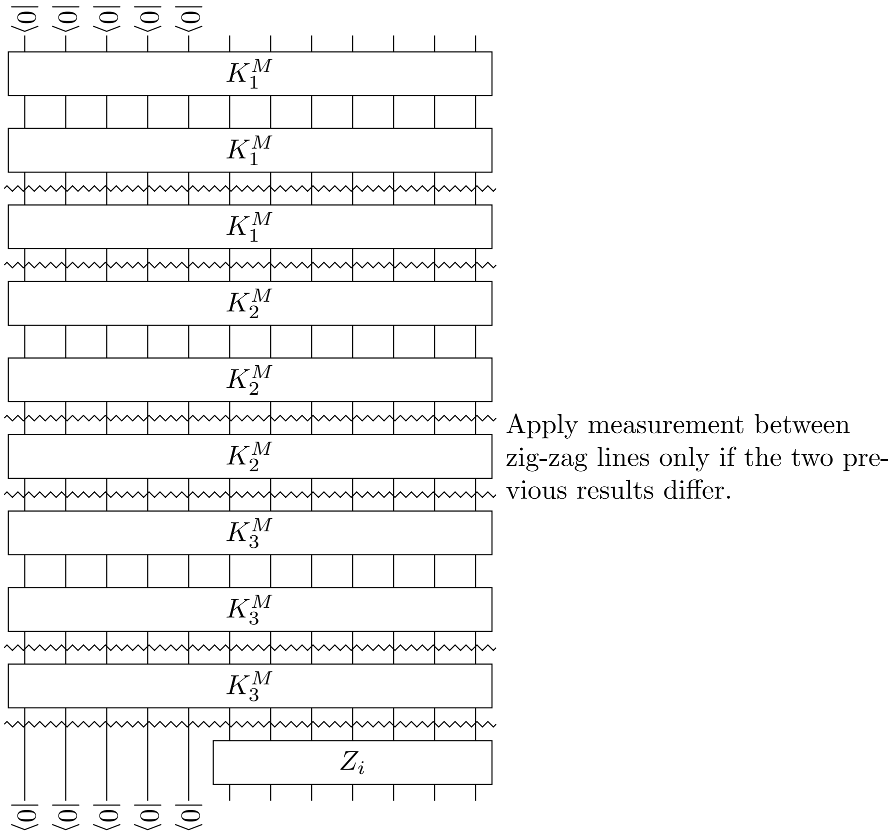
$\langle 0|$
$K_1^M$
$K_2^M$
$K_3^M$
$Z_i$
$|0\rangle$
Apply measurement between zig-zag lines only if the two previous results differ.

Figure 2.10: A circuit for fault tolerantly measuring the first three Steane stabiliser operators fault tolerantly. This circuit can be used for both state preparation and correcting phase flip errors. Image taken from [6].

This circuit uses the first three stabilisers, $K_1$-$K_3$, and can be used for both state preparation and correcting phase flip errors. To correct bit flip errors, $K_4$-$K_6$ can be used in an analogous circuit. For a more comprehensive explanation of the Steane code and error correction in general, see [3, Chap. 10] and [6].

### 2.3.3 The Threshold Theorem

The *Threshold theorem for quantum computation* is a very important result in quantum error correction. The theorem states that given some assumptions about the underlying hardware, a quantum circuit consisting of $p(n)$ gates, where $p(x)$ is a polynomial, may be simulated with error probability at most $\epsilon$ using

$$\mathcal{O}\left(\text{poly}(\log p(n)/\epsilon)p(n)\right)$$

gates where the probability of individual gate errors is $p$, given that $p$ is smaller than a constant threshold value $p_{th}$. This is important in the context of error correction, because it means that given a constant threshold value $p_{th}$ is achieved, error correction will provide a benefit. There are some important assumptions that are needed to be able to reach the threshold though, one of them being that parallel operations are possible. Otherwise, for example dephasing could become too strong for error correction to work [3, p. 481].

# Chapter 3

# Methods

In this chapter I will introduce the methods used for the simulations. This includes general considerations taken when writing the code, as well as what error models were used and how the circuits themselves were modelled.

## 3.1 Density Matrix Simulation of Quantum Circuits

Recall from section 2.1 that a qubit can be represented as a vector by

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

and that the density matrix for the qubit is given by

$$\rho = |\psi\rangle\,\langle\psi| := \begin{pmatrix}|\alpha|^2 & \alpha\beta^* \\ \alpha^*\beta & |\beta|^2\end{pmatrix}.$$

Recall also from section 2.1 that several qubits can be represented by a *product state*, and that in the matrix representation the product state can be found by taking the Kronecker product of the matrices for the individual qubits. This density matrix representation was used in the computer program to represent qubits. Thus, the state of the seven qubits of the Steane code was represented by a $2^7 \times 2^7$ matrix, and gate operations were implemented using the matrix representations given in section 2.2.1.

As mentioned before, the density matrix formalism provides some advantages over the state vector formalism. The ones relevant for this work are more specifically that it allows for mixed states, and system-environment interactions like the amplitude and phase damping channels described previously. It does however come with some disadvantages. An arbitrary system of $N$ qubits requires knowledge of $2^N$ constants in the state vector representation. In the density matrix language, this turns into $2^N \cdot 2^N = 2^{2N}$ constants. The Steane Code uses 7 data qubits and four ancilla qubits simultaneously, so simulation of one logical qubit seems to require performing hundreds of multiplications of $2^{11} \times 2^{11}$ matrices.

### 3.1.1 Partial Trace

Recall from section 2.1 that the *partial trace* can be used to remove parts of a system while keeping the correct measurement statistics about the system. This can be used to dynamically reduce the size of the system during simulations as ancilla qubits can be prepared separately and removed once no longer in use. This was used wherever possible, as a reduction in matrix size from $2^{11} \times 2^{11}$ to $2^7 \times 2^7$ is substantial enough to give a large performance boost.

### 3.1.2 Sparse Matrix Operations

The main way that was chosen to try to combat this is through using sparse matrix methods. Most of the operators used for error correction are either single qubit gates or tensor products of controlled versions of the Pauli operators. The Pauli operators and their controlled versions like the CNOT gate all have exactly one element per row, which means that using sparse matrix multiplication can eliminate a huge number of unnecessary multiplications by zero. It also seems reasonable to assume that the resulting density matrices will, even when errors are included, have a large number of zeros, although the exact amount is very difficult to estimate.

### 3.1.3 Gate Tolerance

Another method that was tried to achieve a speedup was to introduce a tolerance in the gates, so that after all operations have been performed elements of the density matrix smaller than a very small constant are set to zero. In all simulations, the tolerance was set to $10^{-16}$. This number was chosen because it should be reasonably close to the machine epsilon, and several orders of magnitude smaller than typical values of the fidelity error $\epsilon_{fid}$ which, by experience, are typically greater than $10^{-9}$.

### 3.1.4 Error Models

The question of which errors to consider is a difficult one, and it is possible to have very sophisticated models. However, since the exact errors affecting our system in particular are unknown, a different approach was taken.

#### Gate Errors

To begin with, it was assumed that gate operations can introduce errors on the control and target qubits, but that no additional correlated noise is present in the parts of the system at rest. Furthermore, it was assumed that operations can happen in parallel in the sense that if for example a Hadamard gate is applied to two separate qubits, the operation can be performed simultaneously on both. This means that amplitude and phase damping only has to be applied to qubits that are not affected by the multi qubit gate. Parallel operations are, as mentioned in section 2.3.3, a prerequisite for error correction to work in the first place. In light of the *discretisation of errors* mentioned in section 2.3.1, it seems reasonable to assume that single qubit gates introduce bit flip and phase flip errors each with probability $p$, and combined bit and phase flip errors, represented by the $Y$-operator, with probability $p^2$. For two qubit gates, it is assumed that errors can be introduced on both the target and control qubits. Therefore, one also needs to take into account errors on both qubits on the form $\sigma_i \otimes \sigma_j$, where $\sigma_i$ is one of the Pauli operators. For simplicity, since the exact shape of errors and error rates were unknown, it was assumed that the rate of single qubit errors were the same as for single qubit gates, and the rate for two qubit errors were the product of the rates for the single qubit errors involved resulting in e.g. a rate of $p^3$ for the error $X \otimes Y$. The constants in the phase and amplitude damping channels were again set to the single qubit gate error rate $p$.

It was also assumed that gate errors can be modelled by a perfect single or two qubit gates followed by an error operation. In the case of the Hadamard gate with errors, the total operation on one qubit would be

$$H'(\rho) = H\rho H[1-(2p+p^2)] + XH\rho HXp + YH\rho HYp^2 + ZH\rho HZp.$$

#### Initialisation & Readout Errors

In addition to gate errors and amplitude and phase damping, it was also assumed that there are errors associated with qubit initialisation and qubit readout. The initialisation error implementation is straightforward; one can simply assume that on initialisation to $|0\rangle$, there is a probability $p$ that the qubit is instead initialised to $|1\rangle$. Thus, the initialised state would be

$$|\psi\rangle = (1-p)\,|0\rangle + p\,|1\rangle\,.$$

Readout errors can also be implemented in a straightforward manner by defining a probability $p_r$ that the measurement result is incorrect. This way, measuring a qubit as $|1\rangle$ is wrong with probability $p_r$, so the state after measuring a qubit is

$$|\psi\rangle\langle\psi| = (1-p_r)\,|1\rangle\langle 1| + p_r\,|0\rangle\langle 0|\,.$$

The rare earth system in Lund has asymmetric readout errors, meaning that $|0\rangle$ has a relatively high probability to be measured as $|1\rangle$ but not the opposite [7]. With this in mind, it was assumed that the probability to measure $|1\rangle$ as $|0\rangle$ is negligible.

Using the sparse matrix representations of operators and density matrices together with the error models discussed, an object oriented framework for simulating quantum circuits was developed in Matlab, where arrays of qubits and quantum gates are represented by objects that can interact with each other.

## 3.2 Simulating Error Correction with the Steane Code

The final thing that deserves some discussion here is how to actually simulate error correcting circuits. The main merit of the density matrix approach is, as mentioned previously, that mixed states can be described. Recall from section 2.3.2 that the Steane code uses 7 qubits to encode a single logical qubit, and that it can be used to correct one bitflip and one phaseflip error provided that they happen on different qubits, or one arbitrary error. Every syndrome measurement on one logical qubit for a single type of error, that is bitflip or phaseflip error, therefore has 8 different outcomes: no error or e.g. a bitflip error on one of the qubits belonging to the block. Because of the fault tolerant measurement scheme for the stabiliser operators, the number of outcomes increases further. Studying figure 2.10, it is possible to count all of the possible paths when measuring $K_1$, $K_2$ and $K_3$. The individual stabiliser measurements have two outcomes, either $+1$ or $-1$. Each of these can be reached in three different ways, since if two consecutive measurements yield the same result that result is taken to be correct. For each stabiliser measurement, there are thus six different paths. After measuring three stabilisers there are thus $6^3 = 216$ possible paths to reach eight possible error syndromes. This is illustrated in table 3.1 below.

Table 3.1: Table illustrating the different outcomes when measuring $K_1$-$K_3$. Since majority voting is used, two consecutive equal measurement results, for example measuring $+1$, $+1$, automatically means the measurement is interpreted as $+1$. This means that there are six different outcomes of every stabiliser measurement. In total, there are therefore $6^3 = 216$ different paths that have to be accounted for.

| $K_1$ | $K_2$ | $K_3$ |
|---|---|---|
| +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, -1 | -1, +1, -1 | -1, +1, -1 |
| +1, -1, -1 | +1, -1, -1 | +1, -1, -1 |

In the case of measuring phase flip errors, there are eight possible outcomes: Either there is no measured error, or there is a phase flip error on one of the 7 data qubits. Therefore, after performing the measurement step one can take a sum over the 27 different paths leading to a specific syndrome, weighted by the correct probabilities for each outcome, and then perform the correct error correction operation. Finally, one can sum the total error corrected states weighted by the total probability to get them. In this way all calculations can be performed analytically, which would not be possible in the state vector representation.

Some additional assumptions were used during the simulations, and these will be listed here. Firstly, since the $|0\rangle$ and $|1\rangle$ used are different ground state levels, it was assumed that amplitude damping does not affect bits which aren't operated on. It was further assumed that both ancilla preparation and measurement can happen in parallel with operations on the data qubits. This means that the data qubits are not affected by phase damping during readout and ancilla preparation.

# Chapter 4

# Results & Discussion

In this chapter I will present and discuss the results. I will begin with presenting some run-time statistics about the code given different conditions, and then present the simulation results.

## 4.1 Code Performance

A prerequisite for simulating error correction is that the quality of the code used for simulation is high enough. A major concern at the beginning was that it would take too much time to simulate error correction, since it involves a large number of operations with large matrices. There are essentially two major factors that can be expected to have a large impact on the run-time of a simulation, namely the *depth* of the circuit and the *width of the circuit*. The width and depth of a circuit are illustrated in figure 4.1 below.

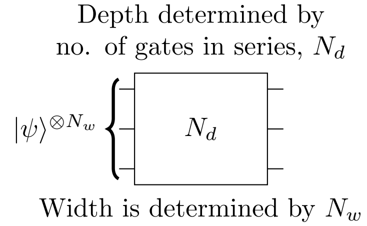
Depth determined by
no. of gates in series, $N_d$
$|\psi\rangle^{\otimes N_w}$
$N_d$
Width is determined by $N_w$

Figure 4.1: Diagram to illustrate the depth and width of a circuit. The width here is given by $N_w$, the number of qubits in the circuit. The depth is given by $N_d$, the number of gate operations in the circuit.

The depth of the circuit is simply the number of gates that are applied in series, and the width of the circuit is the number of qubits it uses.The run-time is expected to scale linearly with the circuit depth, since the time it takes to run a gate should be approximately constant. In contrast to this, one would expect the run-time to scale exponentially with the width of the circuit, since adding a qubit increases the size of the density matrix by a factor of 4. Therefore, one would expect the main limiting factor when it comes to code performance to be the width of the circuit. Also, this is the factor that should be affected by using sparse matrix operations, since the number of elements in the gate matrices scale as $2^{N_w}$ while the matrix size scales as $2^{2N_w}$.

### 4.1.1 Performance as a Function of Circuit Width

To test the impact of the circuit width on the run-time, a simple test was performed where one X-gate, with errors, was applied to a system of increasing size. For every size of the circuit, the run-time of applying a gate was measured a number of times and the final run-time was taken as the mean value. This was done for system sizes between 1 and 12 qubits. To compare with the case without sparse operations, the same test was also carried out for random full matrices of the correct size. To get the correct number of operations when compared with the gate with errors, the total operation $f(A)$ for the full matrix $A$ that was tested was $f(A) = A^3 + A^3 + A^3 + A^3$. This should be compared with the expression for a single qubit gate with errors given in section 3.1.4. The results of this test are shown in figure 4.2.

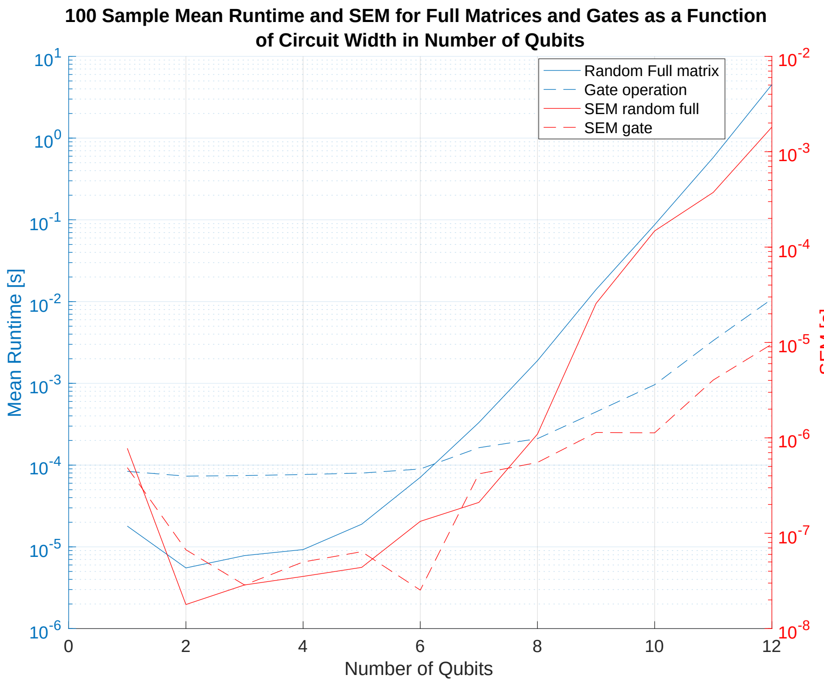

Figure 4.2: Mean run-time and Standard Error of the Mean (SEM) for 100 samples as a function of circuit width expressed in number of qubits. The solid lines show the results for full matrices, and the dashed lines show the results for the sparse matrices and gates used in actual simulations. The blue lines show the mean run-time, and the red lines show the SEM. Regarding the SEM, it is important to note that the scale is different from the scale for run-time, so the errors are quite small. Full matrices and sparse matrices have similar performance for narrower circuits, up to around 7 qubits, but for wider circuits the sparse operations can be seen to outperform full matrices by several orders of magnitude. The difference in run-time at 12 qubits is almost 3 orders of magnitude. The reason why the performance gain is so drastic is that the number of flops needed for operations with full matrices scale with $2^{2N_w}$, while for sparse gate operations the number of flops scales with $2^{N_w}$.

The blue lines in figure 4.2 show the mean run-time, and the red lines show the SEM, with full lines for full matrices and dashed lines for the gate operations used in my code. For narrow circuits, full gates actually outperform the gate operations.

This is to be expected, as sparse operations come with some overhead. However, for widths larger than 7 qubits the gate operations start outperforming full matrix operations quite drastically, with a difference of around 3 orders of magnitude for 12 qubits. This can also be expected due to the exponential scaling of the matrices. For full matrices, the number of flops needed to carry out one multiplication scales with $2^{2N_w}$ where $N_w$ is the number of qubits, whereas the number of flops for sparse matrix operations scale with $2^{N_w}$. It is important to note that the scale for the SEM in figure 4.2 is different from the scale for the run-time, so the errors are actually quite small. In the case of simulating circuits with the Steane code, 7 is the smallest number of qubits used at any point, with 12 being the highest if only one logical qubit is used. This means that sparse operations are better for the whole range of qubit numbers considered in this work. It is important to mention here that figure 4.2 shows a best case scenario, since a sparse density matrix was used for comparison. However, since the full matrix approach always needs more flops for matrix multiplication the performance of the implemented gates is still better.

### 4.1.2 Performance as a Function of Circuit Depth

To measure the run-time as a function of circuit depth, the time it takes to perform $N$ gate operations was measured for different integer values of $N$. The measurements were made separately for single qubit gates and two qubit gates, the single qubit gates being the Pauli gates and the Hadamard gate, and the two qubit gates being the CNOT and CZ gates. This was warranted by the fact that the single qubit gates need fewer operations to calculate errors than the two qubit gates. For the single qubit gates the run-time was measured as the average over 100 samples, and for the two qubit gates it was measured as the average over 25 samples. $N$ was chosen between 1 and 100 gates, and the starting state for all tests was the $|0\rangle_L$-state of the Steane code. The results are shown in figure 4.3.

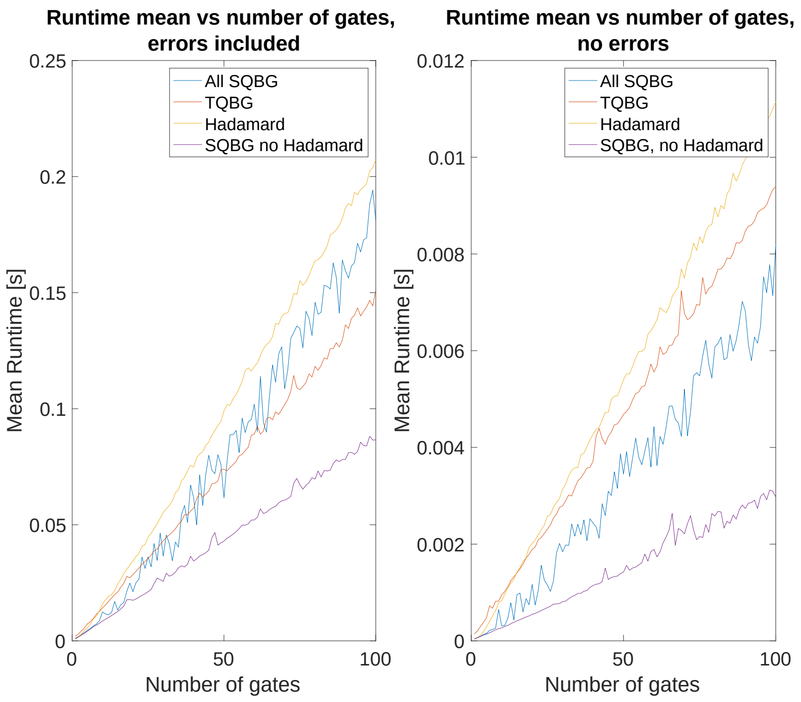

Figure 4.3: Average run-times for the different kinds of gates used in this work. The left hand plot shows run-times for gates with errors, and the right hand plot for gates without errors. Note in particular that the scales are different in the two plots, so the run-time for gates with errors are around an order of magnitude larger.

The left hand plot of the figure shows the average run-times for gates when errors are included, and the right hand plot shows results for gates when errors are not included. Note that the scale is not the same in the two plots, so the run-time for gates with errors is actually an order of magnitude longer than when errors are not included. Note also that the run-time for single qubit gates was measured for three different cases; the Pauli gates separately, the Hadamard gate separately, and all single qubit gates together. As expected, the run-time when errors are included is significantly longer than the case without errors. There are two main reasons for this; Firstly, including errors means having to perform more matrix multiplications and additions. Secondly, including errors often leads to the density matrix having more non zero elements, which would also reduce the efficiency of sparse matrix operations.

Something that is a little bit more surprising is that single qubit gates, when the Hadamard gate is included, perform substantially worse than the Pauli gates and even two qubit gates when errors are included. This is surprising because including the cross term errors for two qubit gates, that is, terms like XX, ZX etc. means that many more matrix multiplications and additions have to be performed. However, the Hadamard gate by itself being the worst performer in figure 4.3 could give us a hint for why this may be. Studying the right hand plot in figure 4.3 the difference between the Hadamard gate and the two qubit gates is not as drastic as in the plot where errors are included. The basic one qubit Hadamard matrix is full, whereas the CNOT, CZ and Pauli gates only have one non-zero element per row. This means that the Hadamard gate will always have twice as many elements in the operation matrix compared to the others when applied to a system of equal size. A likely reason for why the two qubit gates without errors do not perform as well as the Pauli gates is that building the matrices for the two qubit requires some more complicated calculations than for the Pauli gates, which can be constructed with just a Kronecker product.

The left hand plot of figure 4.3 shows that the run-time is about an order of magnitude larger than when no errors are present. This is to be expected as more matrix multiplications are needed per gate and the errors could possibly make the density matrix less sparse. It is interesting to note that the difference between the Hadamard gate and the other gates is further exaggerated in this case. This could maybe be explained by the fact that the Hadamard gate creates superposition states, which could potentially reduce the sparsity of the density matrices even further.

### 4.1.3 Impact of Gate Tolerance

The final method that was identified as potentially being able to reduce run-times in section 3.1.3 was that of introducing a tolerance in the gates to remove small elements from the density matrices, thereby introducing more sparsity. To test the impact of introducing a tolerance, the time to perform one error correction cycle was measured for tolerances between 0 and $10^{-8}$. A constant error rate of $10^{-8}$ was used, and the run-time was taken as the average of 10 samples. It is worth mentioning that the depth of the circuit performing error correction is around 200, and the width varies between 7 and 12 qubits. Figure 4.4 shows the results.

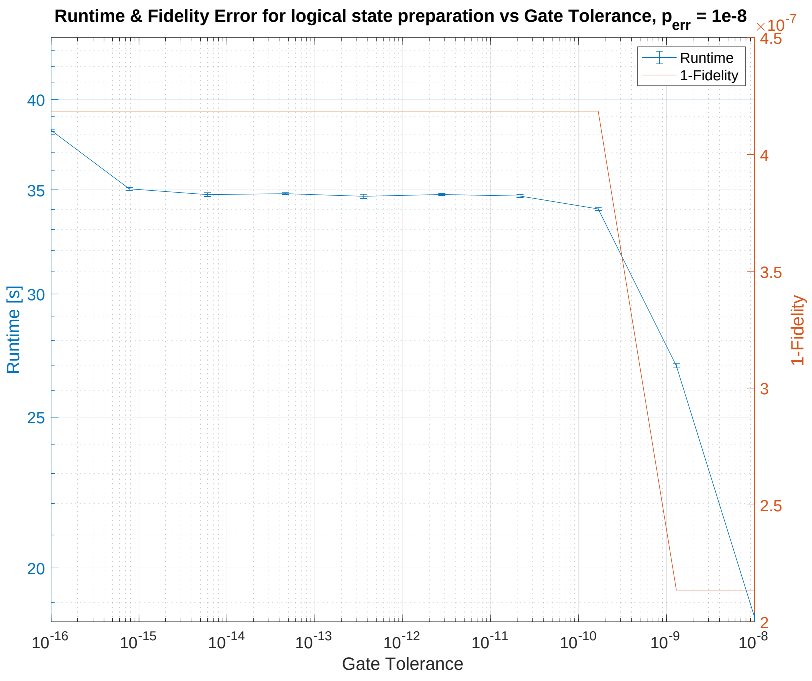

Figure 4.4: Average run-time for five samples and resulting fidelity for state preparation using the Steane code as a function of gate tolerance at an error rate of $10^{-8}$. Introducing a tolerance reduced the run-time in this case, and by choosing the tolerance small enough compared to the error rate the resulting fidelity error seems unaffected compared to when no tolerance is used.

It is interesting to note that the fidelity error plateaus around $4.2 \cdot 10^{-7}$ for all tolerances above $10^{-10}$, with an approximately constant run-time that is slightly lower than when no tolerance is used. Thus, it seems that some speedup could be achieved by choosing a favourable tolerance. It seems reasonable to assume that performance and fidelity error for a specific tolerance would depend on the error rate though, so choosing one tolerance for all error rates would in the best case give a small speedup and in the worst case render incorrect results. This speaks for not using a tolerance during simulations, as the potential gain in run-time is fairly small and it introduces a risk.

## 4.2 Error Correction Simulations

In this section, I will present and discuss the results of the actual simulations of circuits using the Steane code for error correction. I will begin by presenting some initial investigations into which types of errors have the most effect on the fidelity error, and some conclusions, and then move on to manifestations of the *threshold theorem*.

### 4.2.1 Impact of Different Types of Errors

At this point, it might be worthwhile to reiterate the purpose of error correction: To reduce the fidelity error per gate of a circuit beyond what is possible by just improving hardware. Essentially, when using an error correcting code one hopes to see a reduced fidelity error for a logical circuit compared to a physical one, provided that the error rate $p_{err}$ for the physical gates constituting the circuit is low enough. The typical circuit that is considered throughout this chapter is illustrated in figure 4.5.

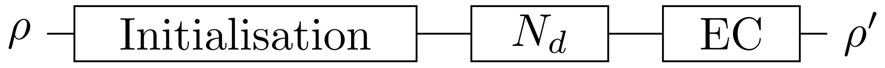

Figure 4.5: Illustration of the type of circuit considered in this section. The circuit is to be understood as a flowchart; A state $\rho$ is initialised, typically in the state $|0\rangle_L$, followed by an algorithm of depth $N_d$, which is then followed by error correction, denoted by EC. After error correction, the fidelity error of the final state $\rho'$ can be measured against a state $|\psi\rangle$, which is the resulting state of the circuit if errors are not present.

With the purpose as stated above in mind, one might expect that the fidelity error when using the Steane code for error correction as in figure 4.5 should be reduced for any circuit, provided that $p_{err}$ is small enough. This, however, turns out not to be true. Figure 4.6 below shows results for a few different scenarios. The circuit that was simulated is the one in figure 4.5 with $N_d = 1$ and where it is assumed that the initialisation stage is completely error free.

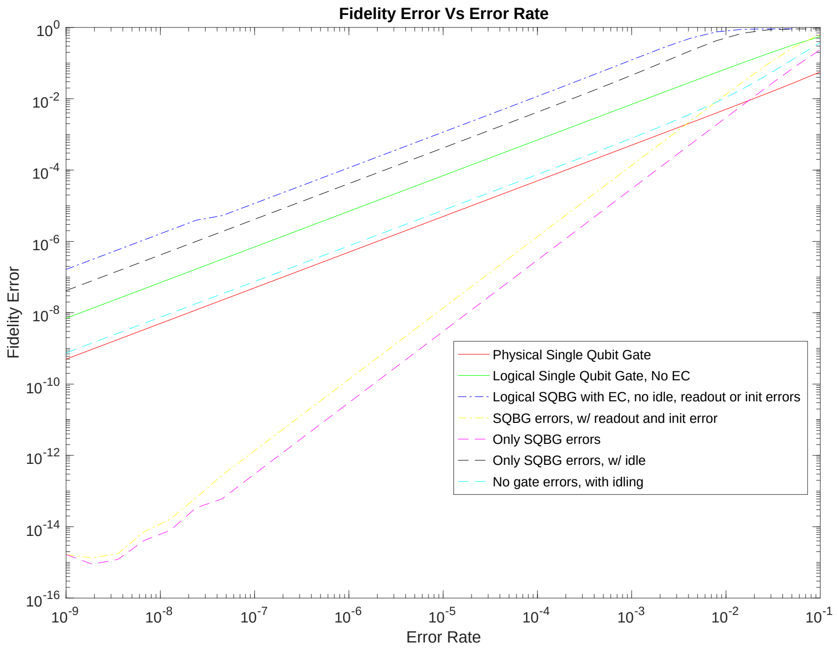

Figure 4.6: Fidelity error as a function of error rate $p_{th}$ for different errors and different scenarios. It is clear from studying the lines that the error for the schemes involving error correction is larger than for those without in realistic scenarios. This is because the circuit only uses 1 actual gate operation but has several EC-gates, so the error is limited by the error for EC.

Figure 4.6 shows 7 different lines. All of these show the fidelity error $\epsilon_{fid}$ as a function of $p_{err}$ for the circuit depicted in figure 4.5 with $N_d = 1$. Different assumptions about which error sources were present and different circuit widths were used for all cases.

1. The solid red line shows the results for a physical single qubit gate. The fidelity error is expected to scale linearly with $p_{err}$ and this seems to be the case.

2. The solid green line shows the results for a logical single qubit gate, without error correction. Since no error correction is used in this case, $\epsilon_{fid}$ is expected to be larger than $\epsilon_{fid}$ for the physical gate by a constant factor. One would however still expect it to scale linearly with $p_{err}$. This seems to be

the case, as this line has the same slope as the solid red line but is translated upwards in the plot.

3. The dash-dotted blue line shows results for a logical single qubit gate with error correction, but where both the readout and initialisation errors are set to 0 and where phase and amplitude damping is neglected. Since error correction is used and the Steane code can correct one error, one would expect $\epsilon_{fid}$ to be proportional to $p_{err}^2$ provided that $p_{err}$ is small enough, since $p_{err}^2$ is roughly the probability that 2 errors will occur. This is clearly not the case, as the slope is the same as for the two previous lines.

4. The dash-dotted yellow line shows results for the same circuit as the dash-dotted blue line. The difference is that all two qubit gate errors were set to 0, and the readout and initialisation errors were set to $p_{err}$. Amplitude and phase damping were also neglected. This line shows what one would expect from error correction, since the slope looks like it is twice that of the previous examples. However, this is not entirely helpful since the assumption of flawless two qubit gates is highly unrealistic.

5. The dashed magenta line shows the same circuit with the same assumptions as the dash-dotted yellow line, except that readout and initialisation errors are also neglected. It has the same slope as the dash-dotted yellow line, but differs from a constant factor. This seems reasonable: the slope should not increase since the Steane code can only correct one error, but removing further error sources should still result in further suppression of $\epsilon_{fid}$. This also suggests that readout and initialisation errors do not have a large impact on the final error.

6. The black dashed line shows results where all gate errors, and initialisation and readout errors have been removed, but amplitude and phase damping are allowed. This line has the same slope as the first few examples, which suggests that amplitude and phase damping errors accumulate too quickly for error correction to work in this case. Comparing this with the initialisation and readout errors used for the dashed magenta line shows that amplitude and phase damping are much more important sources of error.

7. Finally, the dashed cyan line shows results for a logical gate with error correction, but only the errors from amplitude and phase damping are included. This line again has the same slope as when error correction is not applied, and the fidelity error is a constant factor larger than for the physical qubit.

The results shown in figure 4.6 seem to suggest that error correction only works under unrealistic assumptions, at least when using the Steane code. Therefore, a different approach is needed. Gottesman writes in [8] that the number of gates applied before error correction has a large impact on the performance of the error correcting code. Going back to figure 4.5, this would mean increasing the depth $N_d$ of the circuit before error correction. On an intuitive level, it makes sense that this would be the case; Since the syndrome extraction circuit has a much larger depth and width than a physical gate performed on a physical qubit, the risk of introducing an error by performing error correction should be larger than in the physical case when $N_d$ is low.

### 4.2.2 Finding Optimal Gate Numbers

With the results from section 4.2.1 in mind, it seems reasonable to check what depth a physical circuit needs to have for the fidelity error of the circuit to be larger than for the error correction circuit. To do this, $\epsilon_{fid}$ was measured as a function of the circuit depth $N_d$, as illustrated in figure 4.5, for both physical and logical circuits for some values of $p_{err}$. Additionally, $\epsilon_{fid}$ for running an error correction cycle on $|0\rangle_L$ was measured for the same values of $p_{err}$. The results are shown in figure 4.7.

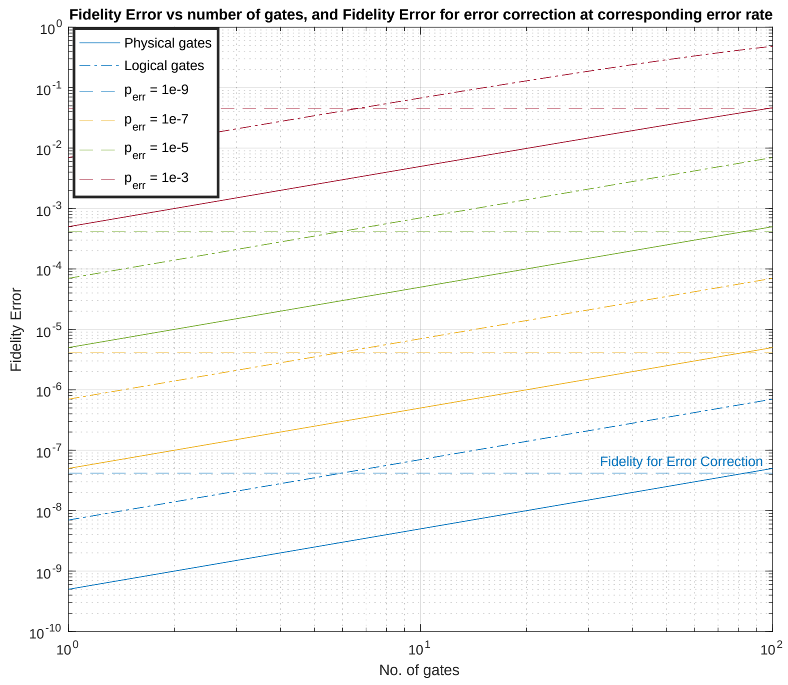

Figure 4.7: Fidelity error as a function of number of gates for physical and logical single qubit gates at fixed values of $p_{err}$, as well as fidelity error for error correction at corresponding values of $p_{err}$. The solid lines show results for physical gates, and the dash-dotted lines for logical gates. The horizontal dashed lines show $\epsilon_{fid}$ for running an error correction cycle. The lines are colour coded by $p_{err}$ so for example the group of blue lines show results for $p_{err} = 10^{-9}$. The effect of increasing $p_{err}$ is to move $\epsilon_{fid}$ vertically in the plot, but there also seems to be a minor change in the relative vertical positions between the lines. The interesting thing to look at is at what depth, given by number of gates in the figure, the line for error correction crosses the line for physical gates, since the intersection gives the circuit depth at which the accumulated error in the circuit is equal to the error created by error correction. This intersection is around $N_d = 85$ for all choices of $p_{err}$.

The lines in figure 4.7 are all colour coded after the value of $p_{err}$, so for example the blue lines at the bottom all show results for $p_{err} = 10^{-9}$. The solid lines in the figure show $\epsilon_{fid}$ for physical gates, the dash-dotted lines for logical gates and the horizontal dashed lines for one error correction cycle. What one expects to see from the logical gates compared to the physical ones is again a fidelity error increased by a constant factor. This is also the case in figure 4.7, as the lines for physical and logical gates have the same slope. What is more interesting to study however is the intersections between the lines for error correction and the lines for physical gates, as the point of intersection determines the depth $N_d$ at which error correction should start improving fidelity. The figure shows that for all chosen values of $p_{err}$ the required depth $N_d$ to see a gain from error correction is around 85.

### 4.2.3 Fidelity Error Curves at Different Gate Numbers

Now that the minimum depth that should needed to gain something from error correction is known, it is possible to study $\epsilon_{fid}$ as a function of $p_{err}$ for physical and logical gates with error correction, at some different depths $N_d$. Figure 4.8 contains two plots where the fidelity error for physical gates, logical gates without error correction and logical gates with error correction have been plotted against $p_{err}$. In the top plot, the depth was $N_d = 160$ and in the bottom plot it was set to $N_d = 10,000$. Additional figures at different values of $N_d$ can be found in Appendix 6.2. The reason why $N_d = 160$ is shown instead of $N_d = 85$ is that a depth of 85 is only enough for logical gates with error correction to break even with physical gates, so the figure becomes less clear. Both of the plots of figure 4.8 also show the fidelity error for error correction, that is, just the last step of figure 4.5 as a solid purple line. In both plots, the solid blue lines show $\epsilon_{fid}$ for physical gates, the solid orange line for logical gates and the solid yellow lines for logical gates with error correction.

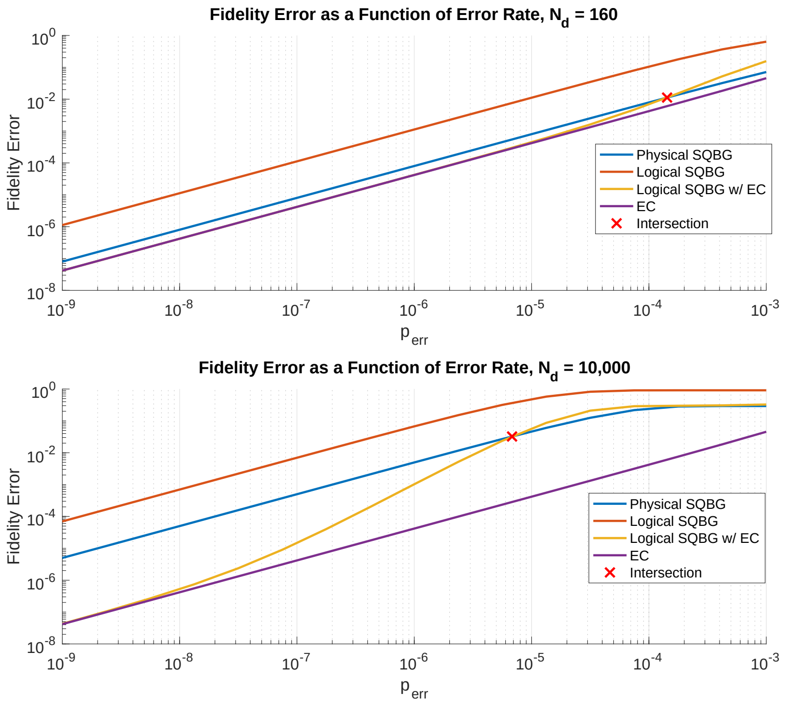

Figure 4.8: Fidelity error as a function of $p_{err}$. The top plot shows results for 160 gates, and the bottom for 10,000 gates. In both figures, the solid blue line shows $\epsilon_{fid}$ for physical gates, the solid orange line for logical gates without error correction and the solid yellow line for logical gates with error correction. Additionally, the solid purple line shows $\epsilon_{fid}$ for just the error correction operation, i.e. the last stage in figure 4.5, and the break even point for error correction is marked with a red cross.

Studying figure 4.8, it is clear that the fidelity error, when error correction is used, is bounded from below by the fidelity error for the error correction operation, shown as purple lines in the plots. This makes sense since the error correction operation itself can introduce errors. Therefore, the total fidelity error shouldn't be expected to be lower than this number. Moreover the fidelity error where error correction is used has approximately double the slope of the fidelity error for physical gates. This implies that the fidelity error scales with $p_{err}^2$ when error correction is used, which is what is expected from the literature (see e.g. [3, Chap. 10]).

Another interesting feature of figure 4.8 is that the threshold error $p_{th}$, which is given by the point of intersection between the curves for logical gates with error correction and the curves for physical gates, is not the same in the two plots. Table 4.1 below shows the threshold error and the ratio $\epsilon_{fid}^{gate}/\epsilon_{fid}^{EC}$ between the fidelity error for $N$ gates and the fidelity error for error correction with $p_{err} = 10^{-9}$. This ratio will also be referred to as the gain. The value of $p_{err}$ was chosen because it is small enough to be far away from the plateau for the fidelity error seen in figure 4.8.

Table 4.1: Threshold error and ratio between the fidelity error $\epsilon_{fid}^{gate}$ for $N_d$ gates and the fidelity error $\epsilon_{fid}^{EC}$ for error correction taken at $p_{err} = 10^{-9}$ for different values of $N_d$. $p_{err} = 10^{-9}$ was chosen because it is small enough for both $\epsilon_{fid}^{gate}$ and $\epsilon_{fid}^{EC}$ to scale appropriately with $p_{err}$.

| $N_d$ | $p_{th}$ | $\epsilon_{fid}^{gate}/\epsilon_{fid}^{EC}$ |
|---|---|---|
| 85 | $7.55 \cdot 10^{-6}$ | 1.00 |
| 100 | $6.58 \cdot 10^{-5}$ | 1.17 |
| 120 | $1.09 \cdot 10^{-4}$ | 1.42 |
| 160 | $1.43 \cdot 10^{-4}$ | 1.89 |
| 250 | $1.44 \cdot 10^{-4}$ | 2.97 |
| 500 | $1.00 \cdot 10^{-4}$ | 5.95 |
| $10^3$ | $5.87 \cdot 10^{-5}$ | 12.00 |
| $10^4$ | $6.82 \cdot 10^{-6}$ | 120 |
| $10^5$ | $6.80 \cdot 10^{-7}$ | 1207 |
| $10^6$ | $6.87 \cdot 10^{-8}$ | 11970 |

Table 4.1 confirms that the threshold error depends on the number of gates used before error correction. It is worth noting that the range of the values is in line with other estimates for the threshold error of the Steane code, see for example [6].

It is practical to define a gain parameter as $\epsilon_{fid}^{gate}/\epsilon_{fid}^{EC}$. The gain is a useful quantity as it describes how much can be gained by using error correction, in the sense that it shows how much larger $\epsilon_{fid}^{gate}$ is than $\epsilon_{fid}^{EC}$. In particular, the gain for $N_d = 85$ is exactly 1. This is in line with the results shown in figure 4.7.

The contents of table 4.1 are also shown in figure 4.9 below, where both the gain and $p_{th}$ have been plotted against the circuit depth $N_d$.

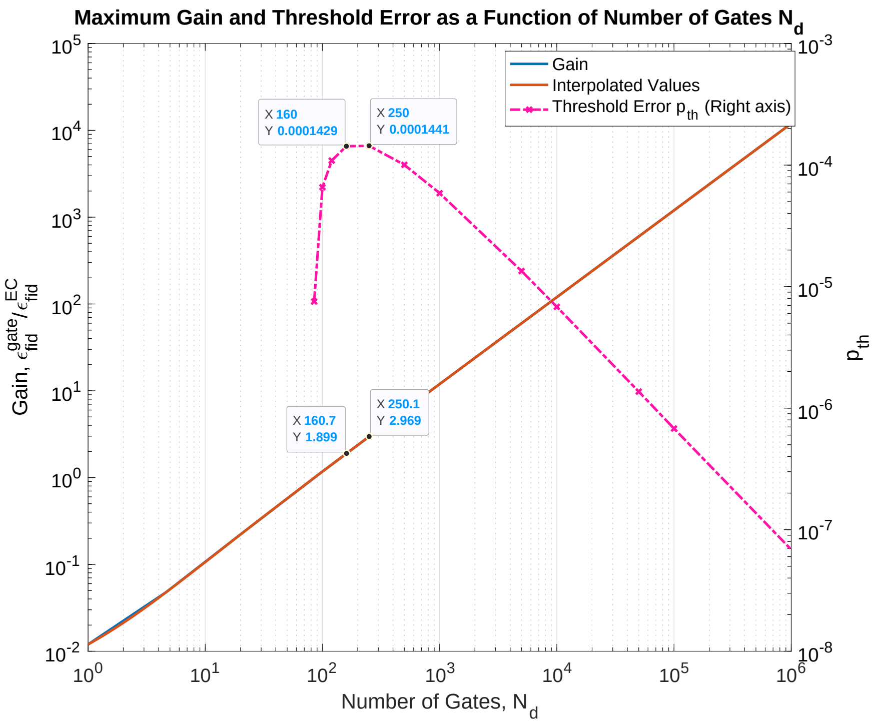

Figure 4.9: Ratio between fidelity error for error correction and fidelity error for physical gates as a function of number of gates on the left axis, and $p_{th}$ as a function of number of gates on the right hand axis. The two highest values for $p_{th}$ are marked, as well as the corresponding gain in fidelity error.

The left hand axis of figure 4.9 again shows the gain for $p_{err} = 10^{-9}$. The gain can be seen to scale linearly with the number of gates, and this has a fairly simple explanation: Figure 4.7 shows that $\epsilon_{fid}^{gate}$ has a linear dependence on $N_d$ for fixed values of $perr$. Thus, the gain given by $\epsilon_{fid}^{gate}/\epsilon_{fid}^{EC}$ will also depend linearly on $N_d$ since $\epsilon_{fid}^{EC}$ is constant for fixed $p_{err}$.

The right hand axis of figure 4.9 shows the threshold error $p_{th}$, that is the error rate $p_{err}$ at which the logical circuit with error correction has the same fidelity error as the physical circuit. The threshold error has a clear maximum around 160-250 gates, and decreases sharply for smaller numbers and smoothly for larger numbers. It is somewhat surprising that $p_{th}$ shows such a strong dependence on $N_d$, given the constant estimates for the threshold error in for example [3, Chap. 10] and [6].

To see why the threshold error is not constant, we have to go back to figure 4.8. It can also be useful at this point to study the figures in Appendix 6.2. In figure 4.8, $p_{th}$ is given by the intersection, marked by a red cross, between the fidelity error curves for the physical circuit and the logical circuit with error correction. The top plot of figure 4.8 is a very good illustration of what happens for small $N_d$. The potential gain is low, and the intersection is at a point where the slope of the curve for the logical circuit with error correction is somewhere between 1 and 2. The smaller $N_d$ is in this regime, the smaller the slope will be at the point of intersection, and this pushes the point of intersection to smaller values of $p_{err}$.

For values of $N_d$ larger than 250, table 4.1 shows that $p_{th}$ is inversely proportional to $N_d$. This can also be observed in the bottom plot of figure 4.8, as well as the plots in appendix 6.2. To see why this happens, one can argue that when $p_{err} \approx N_d^{-1}$ or larger, the probability for an error to occur should be so high that the original state is essentially scrambled, and it is not possible to reliably recover any information. This means that even when error correction is used, there is no way to recover the correct state after the circuit. Therefore, the result is no better than a random guess. This seems to be the case, as the bottom plot of figure 4.8 shows that $\epsilon_{fid}$ is constant for $p_{err} > N_d^{-1}$.

What is the significance of figure 4.9? According to [9], the lowest value for $p_{err}$ achievable with the Rare Earth system should be slightly higher than $10^{-4}$ for the best possible gate. This means that using the Steane code for beneficial error correction might be achievable. At the same time, the corresponding gain is only around 3, and to achieve this, a higher gate fidelity is required. It is also worth noting that actual experimental parameters have not been used in the error model. For example, the assumption that gate errors scale the same way as phase damping and amplitude damping is not necessarily completely valid. If for example phase damping has a larger effect than used in the calculations here, the threshold will occur for lower values of $p_{err}$. The results are however still interesting. For example, they imply that since high gate fidelities are needed to get a large gain with error correction, there is a clear trade-off to be made; While one might be able to improve the fidelity even at high error rates, the gain is so small that it may not be worth the trouble. To get a large gain, very low high gate fidelity is required to begin with. It is also worth mentioning that since only single qubit gates were considered here, results may change significantly when a circuit with more logical qubits is considered.

Other work on the topic of simulating error correcting codes has been made, for example [5] and [10]. The results of [10] are of particular interest here, since it was investigated if a code equivalent to the Steane code could be used in a quantum memory application. The results show that beneficial quantum error correction could be realised for a trapped-ion architecture, provided there are reasonable improvements in the hardware. It was also shown that optimising the syndrome extraction protocol can provide additional improvement. It should be noted that these results are not directly comparable to the ones presented in this work, since different but analogous parameters were studied and the underlying physical system is slightly different. The results still seem to agree with the ones presented in this work in that beneficial error correction should be possible. However, to reach high enough fidelities for running useful algorithms for example, either other codes need to be considered or some sort of code concatenation or other optimisation scheme must be employed.

# Chapter 5

# Conclusions & Outlook

The results presented in this work deal with two separate but connected topics: simulating circuits with quantum error correction and the performance of the code that was produced to simulate quantum circuits.

## 5.1 Conclusions

It was shown that logical state preparation could be performed in around 35 seconds on a normal desktop computer. It was also shown that the code optimisations employed, mainly using sparse matrices as often as possible, shortened run-times by several orders of magnitude compared to using full matrices. As expected, the run-time was found to increase exponentially with respect to circuit width, and linearly with respect to circuit depth. Overall, the performance was sufficient for simulating circuits with error correction was possible in a reasonable amount of time. A spontaneous test of running an error correction cycle on a system of two encoded qubits had an execution time of around 70 minutes on a normal desktop computer. Because of the way the error correction cycle was implemented, c.f. section 3.2, it can easily be modified to run in parallel on up to 8 cores of a normal CPU which could increase performance by around a factor 8. This means that it would be feasible to test fidelity for the Steane code on two logical qubits. Since the Steane Code uses 7 data qubits and up to five ancilla qubits, this means that a system of around 14 qubits could be possible to work with, but anything larger than this would probably start to become impractical.

Moving on, it was found that the circuit depth as defined in figures 4.1 and 4.5 has a significant impact on the performance of the Steane code when used for error correction in a circuit setting. It was found that a circuit depth of at least 85 was required to reach a break-even point where error correction is able to improve gate fidelity. Perhaps more interesting, it was found that the threshold error $p_{th}$ where the gate fidelity error for physical and logical circuits are equal has a strong dependence on circuit depth. The maximum value of $p_{th}$ found was slightly larger than $p_{th} = 10^{-4}$ at a circuit depth of between 160 and 250 gates. For depths $N_d$ larger than 250, it was found that $p_{th}$ scales as $N_d^{-1}$. The maximum value of $p_{th}$ is of particular interest since the estimated lowest gate fidelity error for the rare-earth-ion system used by the Quantum Information group at Lund University approaches $10^{-4}$. Moreover, it was found that the potential gain from using error correction, determined as $\epsilon_{fid}^{phys}/\epsilon_{fid}^{EC}$ has a linear dependence on $N_d$. At $N_d = 250$, it was found that the gain was around 3. This means that there is a clear trade-off when determining how often error correction should be performed, as a higher threshold error corresponds to lower gain and vice versa.

## 5.2 Outlook

There are a number of further investigations that could be pursued in relation to this work. Perhaps the most important, and a good starting point for further research would be to create a more sophisticated error model for rare-earth-ion systems. This would be highly beneficial, as it would enable a realistic assessment as to whether using the Steane code for error correction would be feasible with current or future hardware. Once such an error model is realised, it would also be possible to run similar simulations as the ones presented here using different error correcting codes. An important property of rare-earth-ions is their long coherence time, $T_2$. It was shown in [11] that coherence times of over 2 ms are achievable for excited states in $Eu^{3+}$: $Y_2SiO_5$. Furthermore, it was shown in [12] that coherence times for some hyperfine ground states of $Eu^{3+}$: $Y_2SiO_5$ of around 6 hours are possible. This is particularly promising for the rare-earth-ion quantum computing schemes introduced in [13], as hyperfine ground states are used for the $|0\rangle$ and $|1\rangle$ states, while gate operations make use of the excited state with $T_2 \approx 2$ ms. This means that a form of quasi-parallelism can be achieved in error correction protocols that make use of sequential operations, as idle qubits can have a very long lifetime. Some examples of codes using very few physical qubits and that use such sequential operations are introduced in [14]. These also have an advantage in that systems using few qubits are potentially easier to realise in an experimental setting. Further research could for example investigate if rare-earth systems would be good candidates for testing the codes and fault-tolerant schemes introduced in [14] experimentally. An example of other error correcting codes that could be interesting to study in a rare-earth-ion context would be the so called surface codes studied in [5]. These are less efficient than the Steane code in terms of how many physical qubits are needed to encode a logical qubit, but have the potential for higher threshold values $p_{th}$. Another more simple question to investigate is how the Steane code performs for systems larger than one logical qubit. It is not clear whether the performance would be the same as for a single logical qubit as, in spite of fault-tolerant gate design, errors in such a system are allowed to propagate between the two logical qubits.

As many of these further research topics require simulating wider circuits than in this work, the efficiency of the code would also most likely need to be improved in terms of memory usage and run-time. One way to do this would be to use a Monte Carlo approach for simulation instead of the analytical one used here. This has some drawbacks, for example requiring a large amount of runs to give good results. However, this may be mitigated by the fact that parallelisation would be straightforward. Taking the Monte Carlo approach would however not make profit from some of the benefits of the density matrix approach discussed in 3.2. Another interesting option would be to again take inspiration from [5], where GPU acceleration was used to achieve high performance even when large numbers of qubits were used.

# Chapter 6

# Appendix

## 6.1 Raw Data for Code Performance

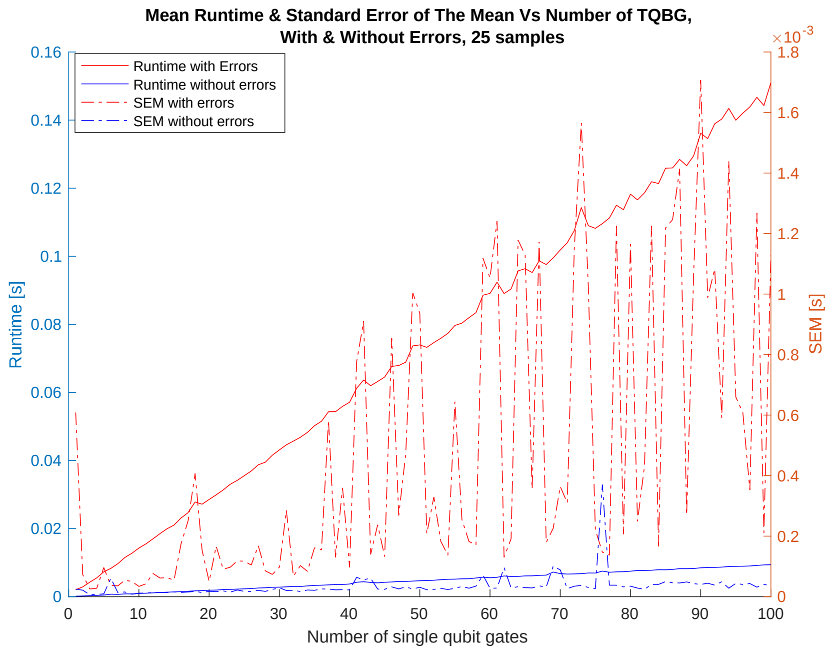

Figure 6.1: 25 sample mean runtime to the left and standard error of the mean to the right for CNOT and CZ gates with and without errors as a function of number of gates. The logical zero density matrix of the Steane code was used as the initial state for every run. The choice of gates was randomised, so that the plot would be representative for arbitrary combinations of two qubit gates.

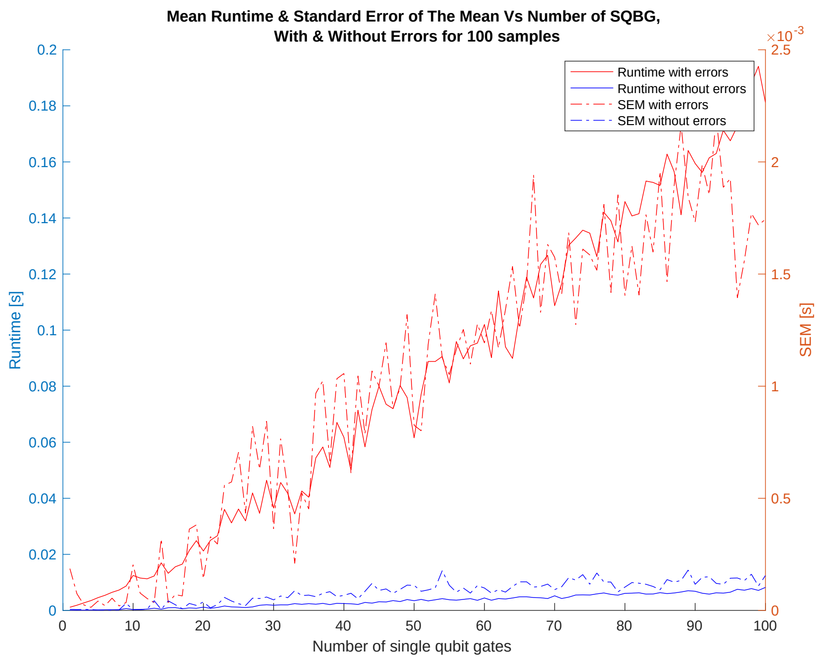

Figure 6.2: 100 sample mean runtime to the left and standard error of the mean to the right for the Hadamard and Pauli gates with and without errors as a function of number of gates. The logical zero density matrix of the Steane code was used as the initial state for every run, and the choice of gates was randomised for every run.

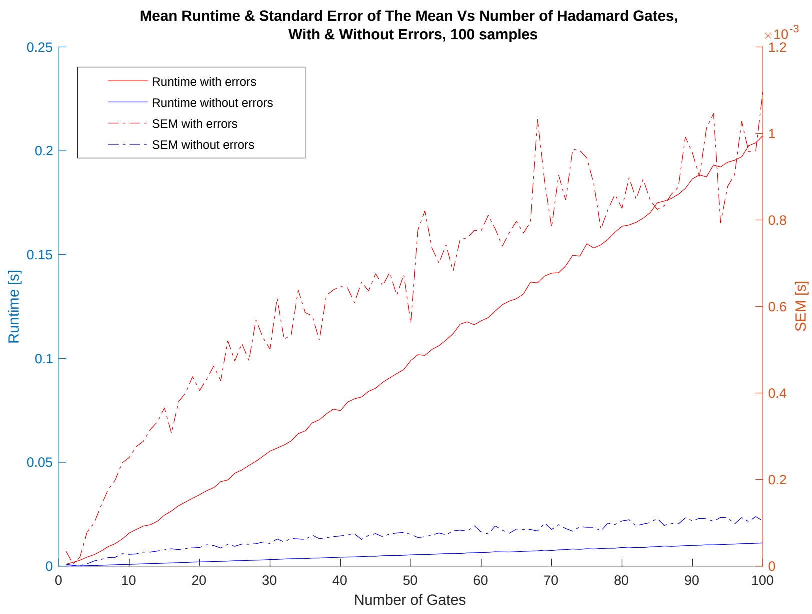

Figure 6.3: 100 sample mean runtime to the left and standard error of the mean to the right for the Hadamard gate with and without errors as a function of number of gates. The logical zero density matrix of the Steane code was used as the initial state for every run.

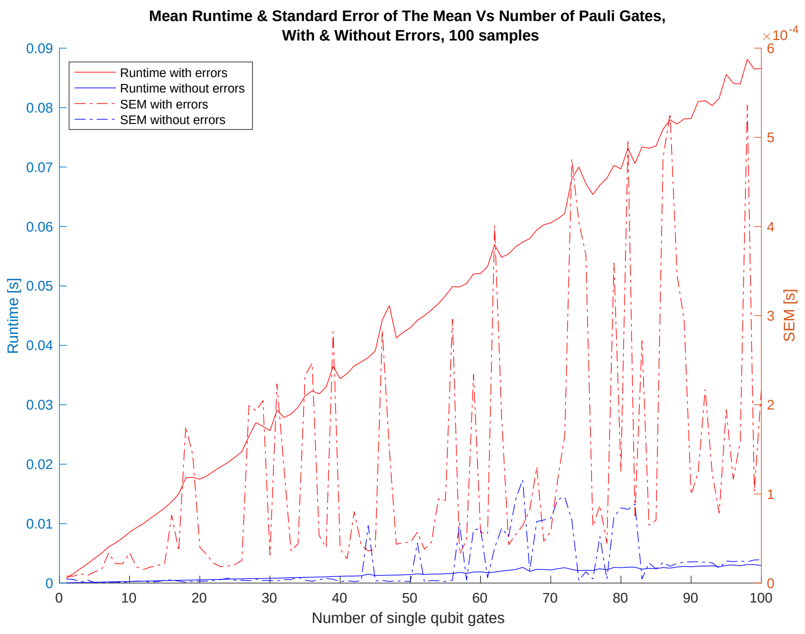

Figure 6.4: 100 sample mean runtime to the left and standard error of the mean to the right for the Pauli gates with and without errors as a function of number of gates. The logical zero density matrix of the Steane code was used as the initial state for every run, and the choice of gates was randomised for every run.

## 6.2 Additional Plots from Simulations

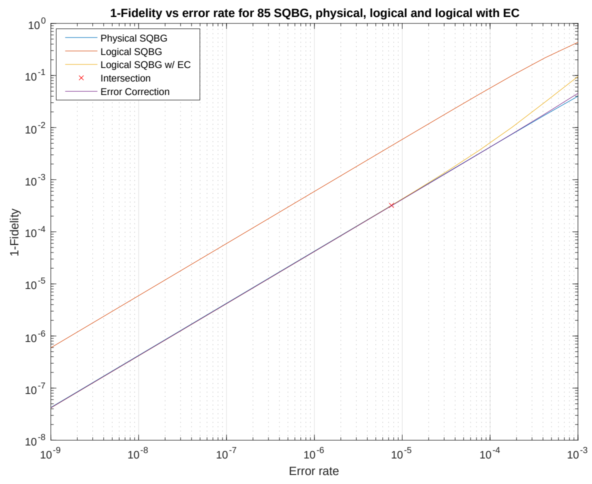

Figure 6.5: Fidelity error for 85 single qubit gates, 85 logical single qubit gates without error correction, and 85 logical single qubit gates with error correction as a function of error rate. The plot also shows the fidelity error for error correction by itself as a function of error rate.

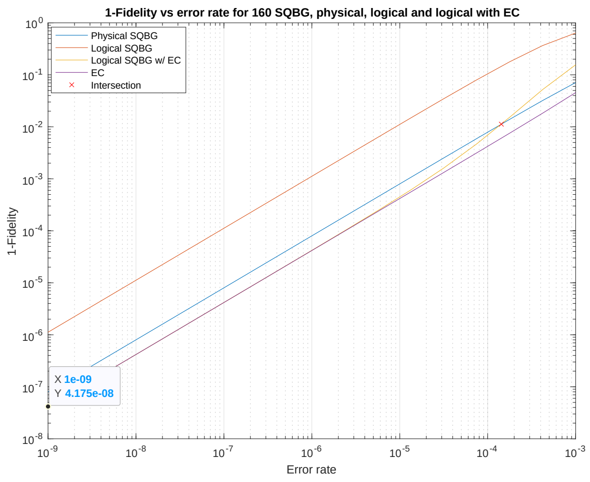

Figure 6.6: Fidelity error for 160 single qubit gates, 160 logical single qubit gates without error correction, and 160 logical single qubit gates with error correction as a function of error rate. The plot also shows the fidelity error for error correction by itself as a function of error rate.

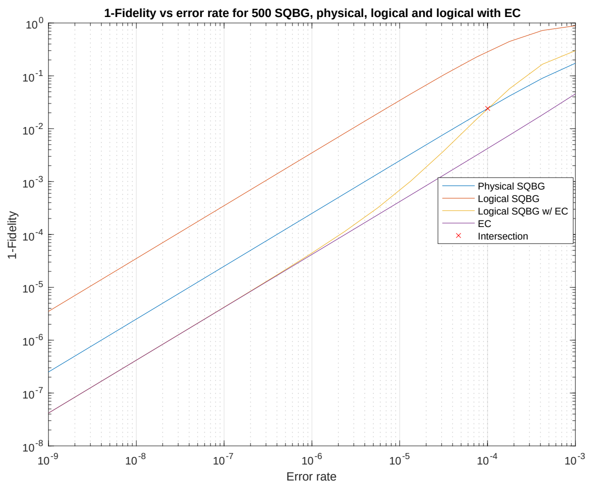

Figure 6.7: Fidelity error for 500 single qubit gates, 500 logical single qubit gates without error correction, and 500 logical single qubit gates with error correction as a function of error rate. The plot also shows the fidelity error for error correction by itself as a function of error rate.

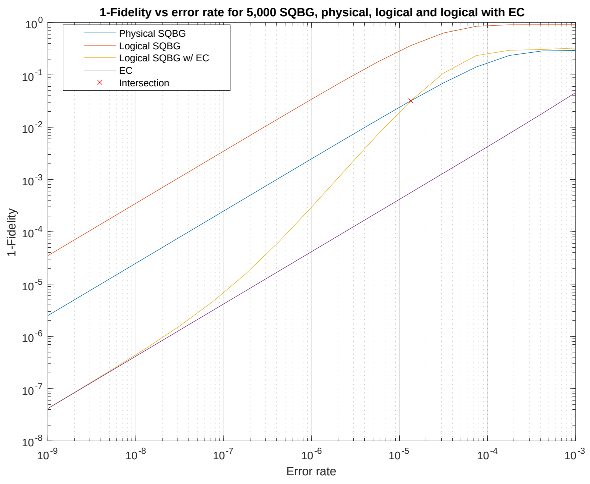

Figure 6.8: Fidelity error for 5000 single qubit gates, 5000 logical single qubit gates without error correction, and 5000 logical single qubit gates with error correction as a function of error rate. The plot also shows the fidelity error for error correction by itself as a function of error rate.

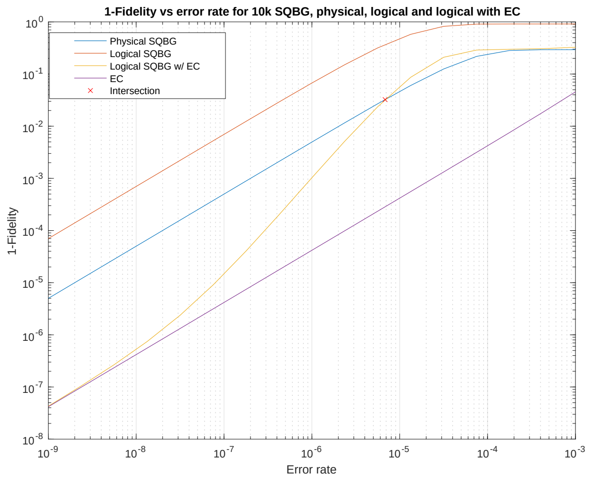

Figure 6.9: Fidelity error for 10,000 single qubit gates, 10,000 logical single qubit gates without error correction, and 10,000 logical single qubit gates with error correction as a function of error rate. The plot also shows the fidelity error for error correction by itself as a function of error rate.

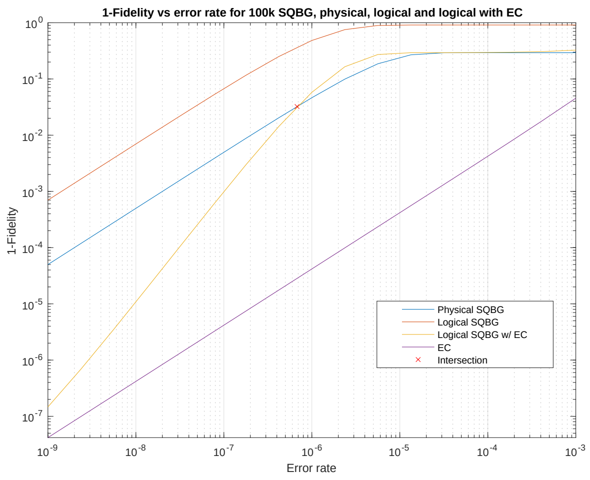

Figure 6.10: Fidelity error for 100,000 single qubit gates, 100,000 logical single qubit gates without error correction, and 100,000 logical single qubit gates with error correction as a function of error rate. The plot also shows the fidelity error for error correction by itself as a function of error rate.

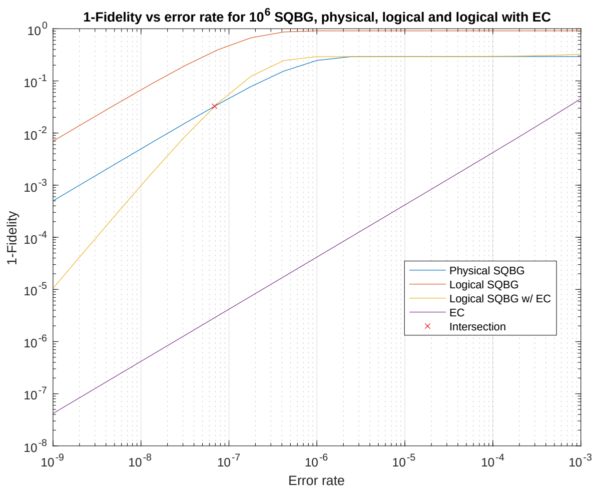

Figure 6.11: Fidelity error for $10^6$ single qubit gates, $10^6$ logical single qubit gates without error correction and $10^6$ logical single qubit gates with error correction as a function of error rate. The purple line shows the fidelity error of error correction by itself as a function of error rate.

# Chapter 7

# Bibliography

[1] R. Feynman, Int J Theor Phys **21**, 467 (1982).

[2] A. K. B. R. e. a. Arute, F., Nature **574**, 505 (2019).

[3] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press, 2010).

[4] D. S. Steiger, T. Häner, and M. Troyer, Quantum **2**, 49 (2018), ISSN 2521-327X, URL `https://doi.org/10.22331/q-2018-01-31-49`.

[5] T. O'Brien, B. Tarasinski, and L. DiCarlo, npj Quantum Inf **3** (2017), URL `https://rdcu.be/b6k3R`.

[6] S. J. Devitt, W. J. Munro, and K. Nemoto, Reports on Progress in Physics **76**, 076001 (2013), ISSN 1361-6633, URL `http://dx.doi.org/10.1088/0034-4885/76/7/076001`.

[7] A. Walther, L. Rippe, Y. Yan, J. Karlsson, D. Serrano, A. Nilsson, S. Bengtsson, and S. Kröll, Physical Review A **92** (2015), ISSN 2469-9926.

[8] D. Gottesman, *Stabilizer codes and quantum error correction* (1997), `quant-ph/9705052`.

[9] *As of yet unpublished results of the quantum information group at the faculty of engineering, lund university.*

[10] A. Bermudez, X. Xu, R. Nigmatullin, J. O'Gorman, V. Negnevitsky, P. Schindler, T. Monz, U. G. Poschinger, C. Hempel, J. Home, et al.,

Phys. Rev. X 7, 041061 (2017), URL https://link.aps.org/doi/10.1103/PhysRevX.7.041061.

[11] R. W. Equall, Y. Sun, R. L. Cone, and R. M. Macfarlane, Phys. Rev. Lett. **72**, 2179 (1994), URL https://link.aps.org/doi/10.1103/PhysRevLett.72.2179.

[12] M. Zhong, M. P. Hedges, R. L. Ahlefeldt, J. G. Bartholomew, S. E. Beavan, S. M. Wittig, J. J. Longdell, and M. J. Sellars, Nature **517**, 177 (2015), ISSN 1476-4687, URL https://doi.org/10.1038/nature14025.

[13] A. Walther, Ph.D. thesis, Atomic Physics (2009), defence details Date: 2009-03-20 Time: 10:15 Place: Lecture hall B, Department of Physics, Professorsgatan 1, Lund University Faculty of Engineering External reviewer(s) Name: Wunderlich, Christof Title: Prof. Affiliation: Universität Siegen —.

[14] R. Chao and B. W. Reichardt, Phys. Rev. Lett. **121**, 050502 (2018), URL https://link.aps.org/doi/10.1103/PhysRevLett.121.050502.