Architectures for Fault-tolerant Quantum Computation

Joe O'Gorman

Wolfson College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Michaelmas 2017

Abstract

Quantum computing has enormous potential, but this can only be realised if quantum errors can be controlled sufficiently to allow quantum algorithms to be completed reliably. However, quantum-error-corrected logical quantum bits (qubits) which can be said to have achieved meaningful error suppression have not yet been demonstrated. This thesis reports research on several topics related to the challenge of designing fault-tolerant quantum computers. The first topic is a proposal for achieving large-scale error correction with the surface code in a silicon donor based quantum computing architecture. This proposal relaxes some of the stringent requirements in donor placement precision set by previous ideas from the single atom level to the order of 10 nm in some regimes. This is shown by means of numerical simulation of the surface code threshold. The second topic then follows, it is the development of a method for benchmarking and assessing the performance of small error correcting codes in few-qubit systems, introducing a metric called 'integrity' – closely linked to the trace distance – and a proposal for experiments to demonstrate various stepping stones on the way to 'strictly superior' quantum error correction.

Most quantum error correcting codes, including the surface code, do not allow for fault-tolerant universal computation without the addition of extra gadgets. One method of achieving universality is through a process of distilling and then consuming high quality 'magic states'. This process adds additional overhead to quantum computation over and above that incurred by the use of the base level quantum error correction. The latter parts of this thesis report an investigation into how many physical qubits are needed in a 'magic state factory' within a surface code quantum computer and introduce a number of techniques to reduce the overhead of leading magic state techniques. It is found that universal quantum computing is achievable with ~ 16 million qubits if error rates across a device are kept below 10^{-4}. In addition, the thesis introduces improved methods of achieving magic state distillation for unconventional magic states that allow for logical small angle rotations, and show that this can be more efficient than synthesising these operations from the gates provided by traditional magic states.

Acknowledgements

This thesis, the culmination of the last four and a bit years of my life, would not have been possible without the support, expertise and friendship of a number of great people, both inside and outside the office. This is by no means an exhaustive list, but in particular I must thank sincerely:

Simon Benjamin, for leaving me smarter after every conversation and whose issues with punctuality I learned were merely a manifestation of his incredible generosity with his time and expertise. I have him to thank for the rare combination of freedom and guidance that was afforded me in my studies at Oxford. Thanks in particular for your support in those times when I was unable to work or make the progress I wanted, your patience and understanding are forever appreciated.

Earl Campbell, who helped bestow just a fraction of his vast knowledge of magic states on me. For hosting me for some very fruitful discussions in Sheffield and for making a remote collaboration work painlessly. You taught me a great deal, even if I struggled to keep up at times, and I hope you enjoyed working together as much as I did. Both your beard and intellect are an inspiration. I could not have hoped for a better unofficial second supervisor.

Naomi Nickerson, for her help in getting me up to speed as I began my DPhil and for her insight and humour. I very much enjoyed working together and look forward to doing so again.

Philipp Ross and John Morton, for a number of great discussions and a crash course in donor-based qubits at the beginning of my PhD.

Xiao-si Xu for making my final project so much fun, for asking challenging questions and driving that research forward.

My office mate *Amir Fruchtman* for always having an interesting perspective on 'real' physics and real life, and for making coming into work a pleasure.

The rest of the QuNaT group: *George Knee, Kieran Higgins, Andrea Rocchetto, Erik Gauger, Ramil Nigmatulin and Stefan Zohren* for the pub trips, the film and

pizza nights and for the variety of personalities, interests and enormous brains that made the group such a pleasure to be a part of. And *Ying Li*, in particular, for having the answers to the all the hard questions.

Markus Muller and Alejandro Bermudez, for getting me into the 'small codes' project and for being such generous collaborators.

Andrea Morello and Guilherme Tosi, for hosting me for two fantastic weeks in Sydney and showing me how some real qubits work.

Two amazing cricket teams: *OUCC* for my (second) once-in-a-lifetime trip to Sri Lanka, for my Blue and a first-class 'hattrick' (one wicket, one catch and one run) and more importantly for giving me the chance to train and improve, and the opportunity to do something that I dearly love in beautiful places with talented people. The *Frogs*, for being terrific people and better friends, for their support and mockery in equal measure. There is still no better way to spend a sunny summer Sunday (or a wet one). We will always have Andover.

My friends, both in Oxford and at home, for showing (at times genuine) interest in qubits and magic states, and for enriching my life in every way. Mainly, of course my best friend, *Nina Klein*, for being at the centre of all the best bits about Oxford and for being there to help me through all the toughest parts. I can not possibly thank you enough, but I will continue to try. Thanks. Ppkk.

Finally my family *Deborah, Tom, Rosie, Conor, Anna, Eva and Maddy*, for the love and support not just these past four years, but my whole life. None of the opportunities that have come to me are my doing, they are yours, and I will continue to do my best to make the most of them. I am, despite appearances, very grateful.

Contents

List of publications

This thesis is based largely on work published in the following papers. Each research chapter closely follows the published papers as detailed below.

- Chapter 2: J O'Gorman, NH Nickerson, P Ross, JJL Morton and SC Benjamin "A silicon-based surface code quantum computer", npj Quantum Information **2**, 15019 (2016).
- Chapter 3: X. Xu, N de Beaudrap, J O'Gorman, SC Benjamin, "An integrity measure to benchmark quantum error correcting memories", New J. Phys. **20** 023009 (2018).
- Chapters 4 & 5: J O'Gorman and ET Campbell, "Quantum computation with realistic magic state factories", Phys. Rev. A **95**, 032338 (2017).
- Chapters 6: ET Campbell and J O'Gorman, "An efficient magic state approach to small angle rotations", Quantum Sci. Technol. **1** 015007 (2017).

Additional publications involving the author:

- A. Bermudez, X. Xu, R. Nigmatullin, J. O'Gorman, V. Negnevitsky, P. Schindler, T. Monz, U. G. Poschinger, C. Hempel, J. Home, F. Schmidt-Kaler, M. Biercuk, R. Blatt, S. Benjamin, and M. Muller, "Assessing the progress of trapped-ion processors towards fault-tolerant quantum computation", Phys. Rev. X 7, 041061(2017).

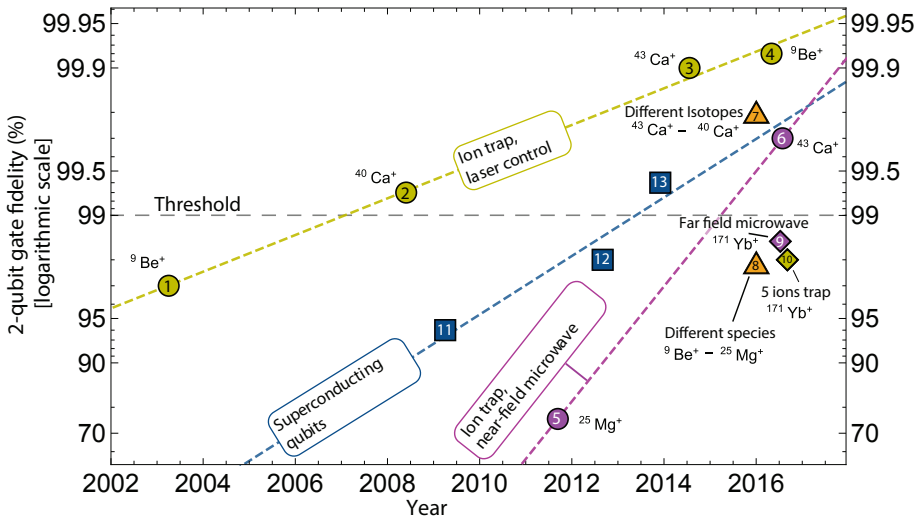
Introduction

The idea of a quantum computer was discussed in 1982 by Feynman [1] who identified the difficulty of simulating quantum systems on classical computers and suggested that computers made from quantum systems might be able to outperform their classical counterparts in this area. The field of quantum computation was put on firm footing in a landmark paper formalising the idea [2] by Deutsch, and later Deutsch and Jozsa who proposed an algorithm which could solve a certain class of problems exponentially faster than a classical algorithm [3]. The discovery of a polynomial-time factoring algorithm by Peter Shor in 1994 [4] led to an explosion of interest in quantum computing. Classically, factoring is a computationally hard problem, meaning that for any known classical algorithm the computational resources required to factor a number scale exponentially according to how many digits the number has. By contrast, multiplication is a computationally easy problem, the resources required scale polynomially with the size of the inputs to the computation. It is this difference in computational complexity between multiplication and factoring that forms the basis of RSA encryption, the world's most widely used cryptosystem. Shor showed that for a quantum computer both multiplication and factoring are computationally easy, the implication being that a quantum computer could break RSA. Shor's algorithm is one example, of which there are now a number [5], of a quantum algorithm which allows a computational speed up over the best known classical algorithms run on a classical computer. Not all of these provide the exponential speed up of Shor's algorithm, for example Grover's algorithm for searching an unstructured database provides a quadratic speed up over classical search algorithms [6]. Perhaps the most exciting potential application of the quantum computer is the one originally envisaged by Feynman: the ability to simulate the behaviour of quantum systems. Currently in many areas of quantum chemistry or materials science the inability to efficiently simulate many-body quantum systems is something of a bottleneck.

Quantum algorithms are able to achieve this speed up for certain problems by exploiting the fact that quantum bits, called "qubits", are more complex objects than classical ones. A classical bit can be in one of two stable states, 0 and 1, and a probabilistic mixture of the two. Probabilities of course are simply positive numbers between 0 and 1. A quantum object can not only live in either of the stable states, now called $|0\rangle$ and $|1\rangle$ and a probabilistic mixture of the two, *but also* in a superposition state. This is a combination of the two states where each is given a complex amplitude α and β such that $|\alpha|^2 + |\beta|^2 = 1$. The modulus squared of the amplitude associated

with a state is the probability of finding the system in that state when it is measured. In higher dimension systems, i.e. those with multiple qubits, the vast Hilbert space in which these quantum states live allows access to a great many superposition states and entangled states, which are not available in classical computation (or in classical physics at all). A quantum algorithm may make use of this larger available set of states or amplitudes in the following way: an algorithm constructed in the correct manner will cause a state representing a 'wrong' answer to have complex amplitudes such that these amplitudes cancel out with those from other 'wrong' answers. For example, a negative amplitude can completely cancel a positive amplitude of the same size. Classical probability theory, of course, does not allow for negative amplitudes. An algorithm can be designed such that wrong answers interfere destructively while the correct answer(s) interfere constructively. At the end of a computation when the final measurement of the qubits is made, a suitably constructed quantum algorithm will return only a right answer (with very high probability) and return one of the many wrong answers with a vanishing probability. It is this ability to make use of negative and complex amplitudes that can allow a quantum algorithm to function as an interferometer which allows the determination of right answers while using interference to remove the possibility of observing incorrect ones.

One of the reasons that such devices have been hard to realise in practice is that quantum effects do not survive at the macroscopic level. The ability to create large quantum states, where superpositions and entanglement persists across a large array of qubits, is necessary. Once the complex amplitudes decay the quantum computation resembles a classical one and any extra power one might extract from quantum physics is lost. This process of decoherence, where the complex parts of the amplitude associated with quantum states are killed off, is due to unwanted interactions with an environment outside the knowledge or control of the experimentalist. There is obviously a tension between the need to isolate quantum systems to a degree that allows them to act in the manner described by quantum physics and the need to gain access to the qubits, to read and manipulate their state, and to control their interaction with other qubits. Currently physicists are able to leave isolated qubits for extremely long periods of time without them suffering a loss of quantumness. For example ion traps have demonstrated coherence times of $\sim$1-10 minutes [7, 8], superconducting qubits have managed 10-100 μs [9, 10], nitrogen vacancy (NV) centres in diamond 600 ms [11] and phosphorus donors in silicon have achieved 2 s [12]. In each of these cases the coherence time is orders of magnitude greater than the single qubit operations that can be performed on the qubits. What currently appears to



	Ion trap: laser				Ion trap: near field						Superconducting		
Data point	1	2	3	4	5	6	7	8	9	10	11	12	13
Reference	[13]	[14]	[15]	[16]	[17]	[18]	[19]	[20]	[21]	[22]	[23]	[24]	[25]

Figure 1: Progress of trapped ion and superconducting qubit technology over time. y-axis: 2-qubit gate-fidelity, in logarithmic scale. Above the threshold of 99% it is possible to reduce errors using the surface code error-correcting scheme. [Taken from NQIT's Technical Roadmap for Fault-Tolerant Quantum Computing, Amir Fruchtman and Iris Choi, October 2017 [26]. Used with permission.]

be the hardest thing to do is to interact two qubits together in a controlled manner. Progress is being made in a range of systems. The fidelity of two-qubit gates in leading ion trap and superconducting systems, whose development is summarised in Figure 1, now exceeds the surface code threshold. This means that error rates in a quantum memory or computation could be suppressed arbitrarily given sufficient qubits interacting with each other with that level of precision.

The design of a quantum computing *architecture* must take into account the various questions that must be answered in bridging the gap between these low level implementations of qubits and qubit gates, and the final goal of a performing a quantum algorithm which offers a speedup over the best classical one for the task. The questions that lie in between these two ends of the quantum computing spectrum will concern us in this thesis. How do we protect the computation from the errors on qubits? How do we link up qubits to provide the right set of high-level operations? How can we design a scalable device: one suitable not just for demonstrations of a few qubits but large numbers of cheap, easily reproducible qubits? How many qubits are in fact required, and how fast can the computer run? And knowing this, where are we now on the road to universal fault-tolerant quantum computation?

Chapter 1

Quantum Error Correction

This chapter introduces the ideas behind quantum error correction and universal quantum computation that will underpin the work in this thesis. Section 1 provides a general introduction to quantum error correcting codes. In Section 2 the key idea of fault-tolerance is described. Section 3 presents some issues surrounding the universality of quantum computation. Section 4 is a more detailed introduction to perhaps the most promising method of realistically performing large scale error correction: the surface code.

The greatest challenge that separates quantum computation from classical computation is that of error correction. The presence of noise within a quantum system under experimental control is unavoidable. In a classical device this problem is far less prevalent. In a classical device there are many physical states which represent the '0' in the computer and many which represent '1'. For example, a common way of storing classical bits is in a region of a magnetic disk. Magnetisation in one direction indicates '0' and magnetisation in the opposite direction represents '1'. One can almost always say with confidence whether a system handed to you was initially encoded in a 0 or 1. One might say on reading the memory that if at least 90% magnetisation in one direction is present then that almost surely represents an attempt to encode in that direction. The two states representing 0 and 1 are separated by such a large energy gap that any flip from one state to another due to, for example, thermal fluctuations causing individual spins to flip, is exceedingly rare and in practice the probability of this occurring is negligible. Perhaps most importantly, we can always measure a classical bit. If the state of bit is degrading, in this example for instance the magnetisation has reduced to 80%, then we can check this and rewrite it with 100% magnetisation. Clearly the option of constantly measuring single qubits does

not exist if we wish to exploit superpositions and entanglement to process information quantum mechanically.

A quantum computer, on the other hand, is inherently more vulnerable to error. Where a classical bit could only suffer the bit-flip error, a qubit can undergo a similar bit flip, or Pauli X error but also the phaseflip, or Pauli Z error. Perhaps even more troubling one might imagine that a number of small continuous errors could afflict a quantum computation, building up over time and causing it to fail. The mechanisms by which such errors can occur are hard to prevent, if a qubit is encoded in the spin state of an electron for example a small thermal fluctuation or an incident photon could corrupt the desired state. Although recent advance in a number of system such as superconducting qubit, ion traps, donors in silicon and NV centres have reduced the error in operations on single qubits to 0.0001% [7] and on two qubits to 0.09% [27], and prototype devices have demonstrated un-error-corrected versions of certain algorithms, such fidelities these numbers are not good enough to allow an 'useful' unencoded computation to occur. For example a 'post-classical' factoring task might require many thousands of logical qubits and potentially over 10^{11} gates as we shall revisit in Chapter 5

Any serious scheme for quantum computing needs to include some method for quantum error correction such that errors can be identified and fixed as they arise, allowing the computation to proceed without accumulation of errors. To this end quantum error correcting codes have been developed [28–30]. These are methods of encoding 'logical' qubits in the state of multiple physical qubits in an error correcting code, on which operations can be performed to detect and correct errors, thereby protecting the logical information. Shor introduced the first quantum error correcting code in 1995 which stored one logical qubit in the state of nine physical qubits [28]. This code is capable of correcting one phase-flip error and one bit-flip error. As long as the rate of error occurring on the physical qubits is low enough, then it is possible to arbitrarily suppress the rate of logical error by increasing the redundancy of the code, for example by concatenation, and moreover this is efficient. That is the number of physical qubits required to encode a logical qubit with error ϵ scales as polylog$(1/\epsilon)$, so the need to encode our quantum information does not have an exponential overhead that would destroy the fundamental speed-up that we expect from our quantum algorithm.

1.1 Error correcting codes

Quantum error correcting codes use many physical qubits and to describe them using a full quantum state picture can be impractical. A more practical method to describe quantum codes is provided by the stabiliser formalism [31], which can be used to describe certain codes by operators rather than the states of the qubits. In this section we introduce this formalism and demonstrate how it is used to describe error correction with a stabiliser code.

1.1.1 Pauli group

To begin we define the Pauli group on a single qubit, $\mathcal{P}_1$ which comprises, up to phases, the four Pauli operators

$$\mathbb{1} = \sigma_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, X = \sigma_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, Y = \sigma_2 = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, Z = \sigma_3 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{1.1}$$

The index of an operator indicates the qubit on which that operator acts. For instance Z_i indicates a Z operation on qubit i, with identity, $\mathbb{1}$, acting on all other qubits. The Pauli group on n-qubits, $\mathcal{P}_n = \mathcal{P}_1^{\otimes n}$, is simply a tensor product of single-qubit Pauli operators on n qubits, such that $P \in \mathcal{P}_n$ can be written

$$P = O_1 \otimes O_2 \otimes \cdots \otimes O_n, \text{ where } O_i \in P_1 \ \ \forall i \in 1, \cdots, n.$$

We will use the term weight-x operator to describe a Pauli operator containing x non-identity terms. Please note that the term 'weight' will also be used to describe the number of 1s in a bit string or row of a matrix.

1.1.2 Stabiliser formalism

Quantum error correcting codes consist of a number of physical qubits n which encode a smaller number k of logical qubits. A subspace of the total Hilbert space of the n physical qubits has some protection against errors. We call this subspace the *codespace*. Stabiliser codes are subspaces described by an abelian group called the stabiliser $\mathcal{S}$, which is a subgroup of the Pauli group. The projector on to the codespace is $\Pi \propto \sum_{s \in \mathcal{S}} s$, so that $s\Pi = \Pi$ for all $s \in \mathcal{S}$. There always exists a minimal set of operators $\{S_1, S_2, ...S_m\}$ that generate the group.

To process information using the protected subspace of the code it is necessary to define logical operators for the encoded qubits. If $\left|\bar{\psi}\right\rangle_i$ is the logical state on the i^{th} logical qubit then the logical operators are defined,

$$\bar{X}_i|\bar{0}\rangle_i \quad = |\bar{1}\rangle_i \tag{1.2}$$

$$\bar{X}_i|\bar{1}\rangle_i \quad = |\bar{0}\rangle_i, \tag{1.3}$$

$$\bar{Z}_i|\bar{0}\rangle_i \quad = |\bar{0}\rangle_i \tag{1.4}$$

$$\bar{Z}_i|\bar{1}\rangle_i \quad = -|\bar{1}\rangle_i. \tag{1.5}$$

These are valid logical operators so long as they commute with the entire stabiliser group and are not contained within it. X_i and Z_i must also anti-commute with each other.

Quantum error correcting codes are often described in short-hand by three numbers: $[[n, k, d]]$, where n is the number of physical qubits required to encode k logical qubits. The parameter d is called the distance of the code and is the weight of the smallest logical operator in the code or, to put it another way, it is the weight of the lowest weight error which can cause an uncorrectable logical error to occur.

1.1.3 Correcting errors

To identify and correct an error one can measure all the generators of the stabiliser group. Consider a code prepared in an error free state in the codespace $\left|\bar{\psi}\right\rangle$ that suffers from some error in the Pauli group $E \in \mathcal{P}_n$. Before the error has occurred then each measurement of a stabiliser will return a '+1' result. However if a '-1' is recorded then this indicates the presence of an error. Since both the error E and the stabilisers $\mathcal{S}$ are elements of the Pauli group then E either commutes or anticommutes with the stabilisers.

In the case of the E commuting with the stabiliser $S_i \in \mathcal{S}$ the state is still a '+1' eigenstate of S_i. However if E and S_i anticommute then

$$S_i E\left|\bar{\psi}\right\rangle = -ES_i\left|\bar{\psi}\right\rangle = -E\left|\bar{\psi}\right\rangle$$

so the error has transformed the state to a '-1' eigenstate of S_i indicating the presence of an error. Measuring all the stabiliser generators returns a set of +1 and -1 outcomes that together make up a *syndrome*. This syndrome contains the information which allows the determination of a correction operator C which can return the state to the codespace, such that

$$CE\left|\bar{\psi}\right\rangle = S\left|\bar{\psi}\right\rangle$$

where $S \in \mathcal{S}$. Finding a correct or 'good' choice of a correction operator for a given syndrome in some code is not necessarily an easy task.

We can see from the above that should E commute with the stabiliser group then we have one of two situations: the error is itself a member of the stabiliser group and is therefore does not affect the encoded information; or the error E is not correctable by the code as it cannot be identified as having occurred.

1.1.4 The five-qubit code

The smallest posssible code that is capable of protecting against all single qubit errors is the five qubit code. This is a [[5,1,3]] code, meaning that it encodes one logical qubit in the state of five physical qubits. A distance of 3 means that the smallest possible logical operators are weight-3 Pauli operations, two qubit errors can be detected, and any one qubit error can be corrected. The stabiliser group of the code is generated by the operators:

$$S_1 = \mathbb{1}XZZX$$

$$S_2 = X\mathbb{1}XZZ$$

$$S_3 = ZX\mathbb{1}XZ$$

$$S_4 = ZZX\mathbb{1}X.$$

The logical operators of the code can be chosen as

$$\bar{X} = XXXXX$$

$$\bar{Z} = ZZZZZ.$$

It should be easy to identify that these satisfy the necessary (anti)commutation relations. Note that here we have chosen the logical operators to be weight-5 as they have nice 'transversal' form, but we know that the distance of the code tells us it is possible to choose logical operators of weight-3. Any logical operator can be transformed to another possible logical operator simply by multiplication by one (or more) of the stabilisers without changing their action on the logical information. For example an weight-3 logical operator $\bar{X}'$ can be formed by $\bar{X}' = S_1\bar{X} = X\mathbb{1}Y\mathbb{1}Y$ and similarly $\bar{Z}' = S_1\bar{Z} = ZY\mathbb{1}\mathbb{1}Y$.

Any single qubit error on the five qubits can be identified by the syndrome obtained by measuring the stabiliser generators. Let us consider $E = Z_3$, a Pauli Z error on the 3rd qubit. The operator commutes with S_1, S_3 and anticommutes with S_2, S_4 giving us the syndrome $\{S_1, S_2, S_3, S_4\} = \{1, -1, 1, -1\}$. For this code there

are $2^4 = 16$ possible syndromes, one of which, namely {1,1,1,1}, corresponds to the case of no error and the other uniquely identify each of the 15 possible single qubit errors. It is in this sense that the five qubit code is 'perfect'. Thus any single qubit error produces a unique syndrome and can be fixed by applying the appropriate correction operation. In our example the syndrome $\{1, -1, 1, -1\}$ indicates: 'apply Z_3 to fix'.

In addition to this example we note that if we simply wish to detect the errors and discard a faulty logical qubit, rather than correcting it, then each two-qubit Pauli error on the five qubit code will also lead to non-trivial syndrome. Clearly there is no way to distinguish the two qubit errors from single qubit errors, with only 16 unique syndromes available. Given that errors on qubits occur independently with probability p for small p it is always more likely that a single qubit error has resulted in a given syndrome than a higher weight error. To return to the example of the syndrome $\{1, -1, 1, -1\}$: this could be caused by an error $E = Z_3$ with probability p but also could be due to an error $E' = X_4 Z_5$ with probability p^2. Clearly if one is to apply a correction it is always preferable to apply the correction $C = Z_3$, but, if the two-qubit error is the one which has occurred then the overall result of the error followed by the most likely correction operator is $CE' = Z_3 X_4 Z_5$ which does not commute with all the stabilisers so the correction. Thus the correction has not returned us to the codespace. Had we chosen to merely detect rather than correct the error, we would have discarded the run with probability $O(p)$ and been left with a logical state with error $O(p^3)$ given that the trivial syndrome was returned. If we chose to correct the error the logical state after correction (now with no runs discarded) would have been left with a logical error $O(p^2)$.

1.2 Fault-tolerance

The paradigm of error correction we have explored thus far is somewhat removed from the realities of dealing with real errors in physical systems. In particular, we assume that errors have occurred on qubits by some unknown process and the method by which we have extracted error syndrome information and applied correction operators is perfectly reliable. This issue must be dealt with immediately. Is it still possible to correct errors in a quantum system, given not only that errors occur on the physical qubits of the code, but also: in the ancillary systems we couple to these qubits to extract syndrome information, in each and every attempt at an operation on those qubits, every operation between the qubits, and in the measurement and preparation

of the qubits? The answer is perhaps surprisingly (and thankfully!): yes. This is the theory not just of error correction, but fault-tolerance. If error correction suggests the ability to put out a fire, then fault-tolerant error correction suggests that this can still be done if the fire extinguisher is also on fire.

1.2.1 Circuit level errors

First let us explore how quantum errors are modelled in practice. In Chapters 2 & 3 we will delve into the performance of the surface code and small quantum codes in the presence of errors on all the operations on the qubits, endeavouring to model them in a way that accurately reflects the capabilities of the systems in question and therefore to assess the merits of the codes and physical implementations employed in near- and further-future experiments and quantum computers. Here the generic model of noise used is introduced. Where appropriate later in the thesis other sources or distributions of errors that might be unique to the specific system under investigation will be added.

Environmental decoherence

Environmental decoherence is modelled as a depolarising process that occurs independently for each physical qubit. Specifically, when our physical qubits are exposed to the environment for some time t then the probabilities of an error is given by

$$p = \frac{3}{4}\left(1 - \exp(-t/T)\right).$$

Given that an error occurs, it is assigned as one of the three Pauli operators X, Y, Z selected uniformly at random. This occurs independently and in parallel for each physical qubit. In this model the probability of an error occurring in any interval $t + dt$ is constant and Markovian and as $t \to \infty$ a quantum state in this environment approaches the maximally mixed state.

Single qubit gate errors

A noisy single-qubit gate is modelled by the ideal gate followed, with probability p_e, by one of the three Pauli operators X, Y, Z selected uniformly at random.

State preparation error

Noisy state preparation is modelled by ideal preparation followed by a possible error in the same fashion as above.

Measurement error

Noisy measurement is modelled by inverting the state to be measured in the relevant measurement basis, with probability p_e. So for example, prior to a measurement in the z-basis a X operation will be applied to the qubit with probability p_e.

We select a measurement error rate p_m and then a particular outcome of the measurement, $q \in \{0, 1\}$ corresponds to the intended projection P_q applied to the state with probability $(1-p_m)$ and the opposite projection $P_{\bar{q}}$ applied with probability p_m. This noisy projector can be written:

$$\mathcal{P}_q\left(p_m\right) = \left(1 - p_m\right)|q\rangle\langle q| + p_m|\bar{q}\rangle\langle\bar{q}| \tag{1.6}$$

In a refinement of this model, can designate two different values of p_m, one for the case that $q = 0$ and one for $q = 1$. This reflects the reality of many experimental realisations of measurement where, e.g., $|1\rangle$ is associated with an active detection event and $|0\rangle$ is associated with that event not occurring (in optical measurement, the event is seeing a photon that is characteristic of $|1\rangle$). Because of the asymmetry of the process, once imperfections such as photon loss are allowed for then the fidelity of measurement becomes dependent on the state that is measured, $|0\rangle$ or $|1\rangle$.

Two qubit gate errors

A noisy two-qubit gate, such as the entangling gates: controlled-Z (CPHASE) and controlled-X (CNOT), is modelled by the ideal gate followed, with probability p_G, with one of the fifteen non-trivial Pauli operators products $\mathbb{1} \otimes X$, $\mathbb{1} \otimes Y$,..., $Z \otimes Z$ selected uniformly at random.

Discretisation of errors

If two errors E_1 and E_2 are correctable with a certain code then every linear combination of these $\alpha E_1 + \beta E_2$ with $\alpha, \beta \in \mathbb{C}$ is also correctable. The act of making a syndrome measurement will project into the state where either E_1 or E_2 occurred, and these errors are by definition correctable. As the Pauli operators form the basis of operators for an n-qubit Hilbert space, the ability to correct Pauli errors of a certain weight allows us to correct all errors up to that weight.

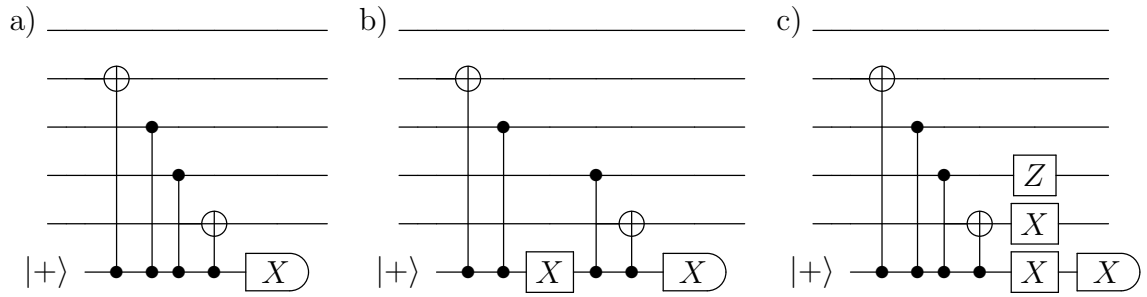


Figure 1.1: (a) The circuit for implementing S_1 of the five qubit code. (b) An X error occurs on the ancilla qubit in the middle of the stabiliser measurement, before the final two entangling gates. (c) The X propagates an uncorrectable two-qubit error onto the code block.

1.2.2 Stabiliser measurements

So how can we ensure that our syndrome extraction process is 'fault-tolerant' and what precisely do we mean by this? A quick example from the five qubit code is instructive. As discussed the correction of a single error on a physical qubit is possible given perfect stabiliser measurements. How is such a measurement made in practice? The quantum circuit realising the measurement of S_1 is shown in Figure 1.1(a). To provide a realistic appraisal of how this circuit works, we need to consider the possibility that every operation we observe (including the identity) can inject some noise onto the circuit, recognising that a two-qubit gate can result in two-qubit errors. Before this though, consider the situation in Figure 1.1(b), here all gate operations are perfect and the 'environment' causes an X error to occur on the ancillary qubit after the second CNOT gate. This single X error will propagate through the remaining gates onto leave a $\mathbb{1}\mathbb{1}\mathbb{1}ZX$ error on the data qubits. Thus a single physical error has lead to an uncorrectable two qubit error on the code qubits! Clearly our attempt at error correction can fail even at $O(p)$, many single error events will be detected and corrected as before, but the circuit will no longer take an error rate of p to $O(p^2)$ but from p to cp where we cannot even necessarily guarantee that $cp < p$. We will thus define fault-tolerance in the following intuitive way: a circuit will be deemed non-fault-tolerant if any single error event leads to an uncorrectable error on the codespace. A fault-tolerant circuit will thus be one that does not allow this to happen. Syndrome extraction for topological codes, such as the surface code, can rely on alternative methods to achieve fault-tolerance. By ordering operations carefully and repeating measurements to account for measurement error is possible to prevent correctable errors spreading and forming chains of error that lead to uncorrectable corruption of the topological code [32, 33].

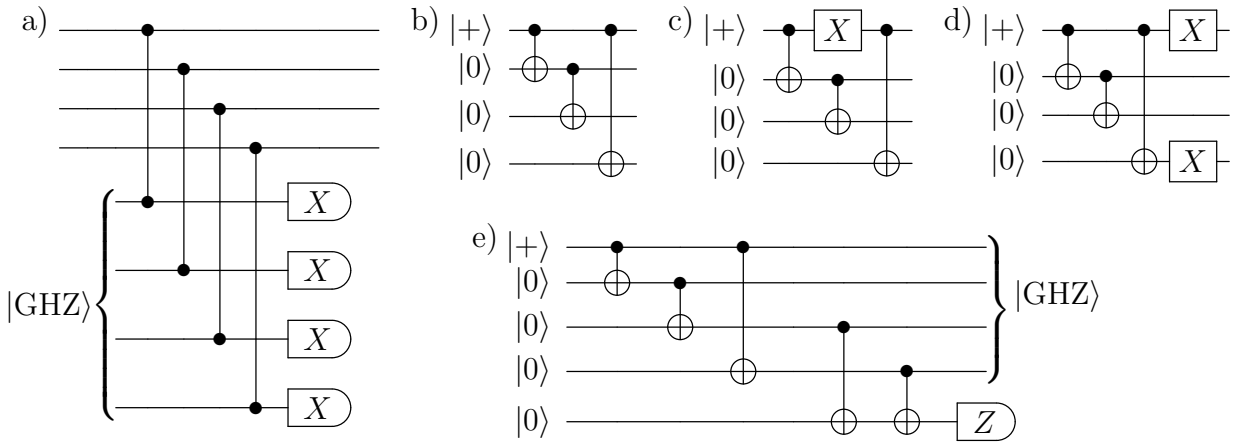


Figure 1.2: (a) A fault-tolerant construction of stabiliser measurement S_1 of the five-qubit code. With all operations perfect this has the same effect on the code block as the circuit in Fig. 1.1(a). (b) The circuit that constructs a $|GHZ\rangle$ assuming no errors. (c) An error occurs on the uppermost qubit after the first CNOT gate. (d) This error propagates to the lowermost wire. It can be seen that this would create an uncorrectable two qubit Z error on the code block in (a). (e) With an additional ancilla and two more CNOT the ancilla construction of (b) can be verified. Any -1 outcomes on the extra qubit indicate a faulty GHZ that could corrupt the data, this run must then be discarded.

It is possible to adapt the simple circuit in Figure 1.1 to prevent this problem. We can stop errors from spreading in the manner described by encoding the ancilla. For example, consider what happens if the ancilla is encoded as a cat state by replacing $|0\rangle + |1\rangle$ with $|0000\rangle + |1111\rangle$. Now each qubit of the ancilla interacts with just one physical qubit of the code block. If there are no faults then the circuit then this implements the desired stabiliser measurement, now however a single error event will not spread to an uncorrectable error on the code block. Note that we need a cat state of w qubits to measure a weight-w stabiliser. There is one outstanding problem, namely that if we want a cat state we have to prepare it and if we prepare it it will be potentially faulty. A Z error in the cat state ancilla will not propagate down to the code block but will lead to the equivalent of a 'measurement error'. An X error such as the one shown in Figure 1.2(c), however, will propagate onto the code block in a damaging way, see Figure 1.2(c). To prevent this we 'verify' the ancilla before we use it by introducing one more single qubit ancilla, see Figure 1.2(e), and discard the ancilla if the measurement returns a '-1'.

We have not yet discussed one particular class of error, namely measurement error. This occurs when a qubit measurement should return an answer of '+1', for example a Z measurement on $|0\rangle$, but instead returns an answer of '-1' while still projecting

the state onto $|0\rangle$. One of these errors would cause one to make an incorrect inference of the correction operator and could potentially lead to an uncorrectable error on the codeblock. However it should be clear that this fault can be tolerated by making repeated measurements of the stabiliser. If a stabiliser measurement is repeated $2n-1$ times then n measurement errors would have to occur for a 'majority vote' of the measurement outcomes to lead an incorrect syndrome measurement. For a distance-3 code, with fault-tolerant syndrome extraction, taking a majority vote of 3 repeated measurements of each stabiliser will allow syndrome information extracted to be robust against any one error that occurs during the whole process.

This particular fault-tolerant construction of a stabiliser measurement is known as 'Shor-style extraction' [34], summarised as a five step procedure:

1. Prepare 'cat state' ancilla
2. Verify ancilla
3. Measure syndrome
4. Repeat as necessary
5. Infer correction operation.

There are alternative constructions such as that due to Steane [35], where the ancilla is encoded in the same code at the logical qubit(s), or a recent suggestion by Chao and Reichardt [36] which uses just two ancilla qubits. Some codes, such as the surface code can achieve fault-tolerance though other interesting tricks such as ordering operations in particular ways to force errors to propagate in a controllable manner even with single ancillas used for an individual stabiliser measurement.

1.2.3 Logical operations

If we wish to manipulate and process quantum information with our noisy qubits we also need a method of performing logical operations on the logical state of the encoded qubits. We of course also desire that these operations are fault-tolerant as well. We want a fault-tolerant operation on a code block to not increase the number of errors within that code block, so that a correctable number of errors does not increase to an uncorrectable number. If the circuit implementing the logical operation is faulty at r locations then the number of errors at the output of that circuit should not exceed

$r + s$ where s was the number of errors already present in the block at the input to the circuit.

One way to guarantee that a logical operation has this property is to perform it transversally. Transversal gates are gates that act bitwise, such that they may be represented by tensor product operators in which the jth term acts only on the jth qubit from each block. This is true for both single and multi-qubit gates. For example a two-qubit gate on two logical qubits, whose physical qubits are labelled by i and j respectively, is performed by applying a two qubit gate $U_{i,j}$ where $i = j$ so each physical qubit of a code block interacts only with its opposite number in the other block. It is of course crucial to perform such gates within a quantum computer, and as we have seen in the previous section we cannot expect a CNOT, for example, not to increase the weight of an error. What we can expect though, again, is that the number of errors in a code block does not increase by a number greater than the number of faulty operations. If CNOT (or any two qubit gate) can be applied transversally and effect a logical version of the gate then although an error can be propagated from one block to another, crucially it does not spread the errors within the block. Thus the error now introduced into a previously pristine block should be correctable, provided the error which propagated from the other block was itself correctable.

For all codes in an important class of codes called the Calderbank-Shor-Steane (CSS) codes the transversal CNOT performs a logical CNOT. The five-qubit code is not a CSS code, but still possesses certain transversal gates, such as the logical X and Z operations as we have already seen. One might wonder whether all the gates one needs for quantum computation can be performed transversally or fault-tolerantly in a given code, or indeed in any code? The short answer is no.

1.3 Universality, magic states and gate synthesis

Given that we want a *universal* quantum computer to be capable of simulating *any* quantum evolution, it is interesting to note that this is possible using a finite set of gates. For a fixed number of qubits n a finite set of gates will generate a finite group. However the unitary group on n qubits $U(2^n)$ is uncountable, not finite, so we know that we cannot exactly implement all the unitaries. As a result of the Solovay-Kitaev theorem [37] we know to get within a distance ϵ of the desired unitary we only need to use polylog$(1/\epsilon)$ gates from a finite 'universal set'. We call the process of decomposing an arbitrary unitary into a series of gates from our chosen universal set *gate synthesis*.

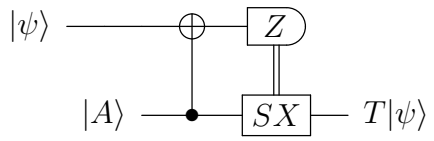


Figure 1.3: Using only Clifford operations and a pure non-stabiliser state $|A\rangle$ the non-Clifford gate T can be effected on the state $|\psi\rangle$.

There are a number of possible choices for the set of universal gates one might opt to use. For example, a CNOT gate and an arbitrary single qubit rotation (which does generate an infinite set) is a valid choice, as is the set of H, CNOT and the $\pi/8$ phase rotation $T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$. Another (overcomplete) group that we will return to frequently is the set of Clifford operations (those that transform Pauli group operators to Pauli group operators under conjugation) and the T gate. In fact it is this group that inspires the model of quantum computation that we will explore in this thesis. Namely one based on magic state distillation.

Based on the discussion of the previous section, ideally it would seem that we would want to be able to perform a universal set of gates transversally within a single code. However it has been shown that this is not possible [38, 39]. While different codes have different sets of transversal gates, none will have a set that can efficiently approximate an arbitrary unitary. For many codes, such as the five qubit code that we have met, and the surface codes which we are about to consider, the possible operations are in the Clifford group. The Steane code [40], for example, has the ability to perform transversally a set of operations that generates the Clifford group.

Given the ability to generate the Clifford group, one just needs any rotation from outside the Clifford group to achieve a universal gate set. Consider the circuit shown in Figure 1.3, which shows one way of performing a non-Clifford rotation in a slightly roundabout way using an ancilla qubit to 'teleport' the gate in. Note that in this circuit each of the operations are Clifford group operations. The only additional ingredient is the pure state ancilla in a non-stabiliser state $|A\rangle$. Now, we know that it will not be possible to prepare this state in a fault-tolerant fashion within a code that does not allow a non-Clifford operation, however Bravyi and Kitaev in Ref. [41] showed that given a supply of noisy copies of some such ancillas it was possible to distill them into a smaller number of higher fidelity copies and, crucially, to do so using only Clifford operations! These ancillas became known as magic states and it was shown that they could be distilled to within a distance ϵ of the desired pure magic state with polylog$(1/\epsilon)$ overhead.

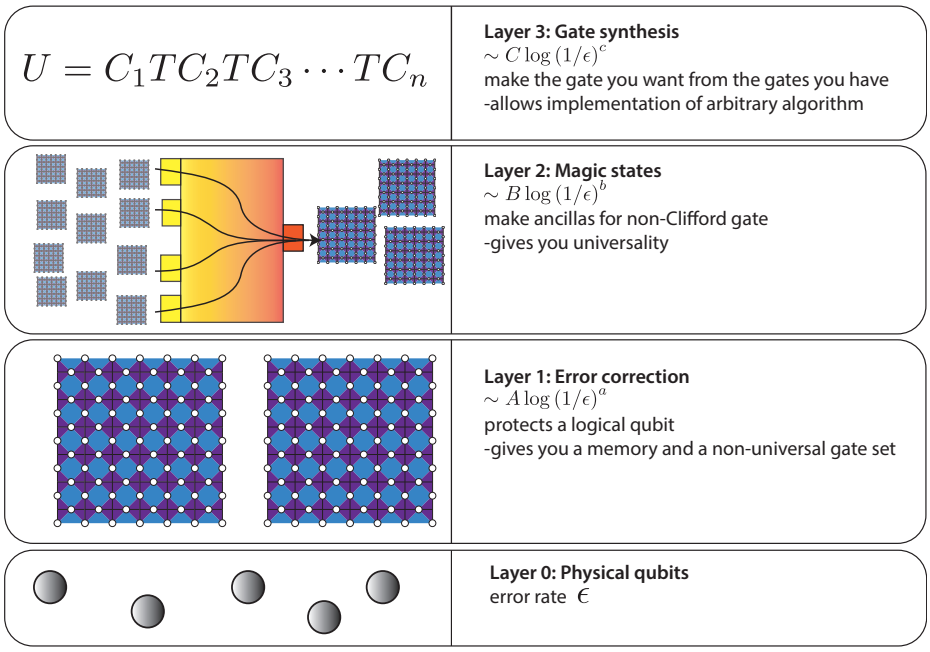


Figure 1.4: The layers of quantum computation. To achieve universal fault-tolerant quantum computation several different ingredients are required, each of which has its own associated overhead. All of these overheads are proven to scale efficiently, but it must be demonstrated that in practical terms these overheads can be feasibly achieved. This means investigation of the prefactors, not just the scaling, is important.

In Figure 1.4 we represent the three layers of universal fault-tolerant quantum computing, which shows that despite each of these layers having their own overhead universal quantum computation remains theoretically efficient. The practicality of quantum computation then rests on finding suitable error correction, magic state and gate synthesis techniques whose total resources and complexity are practically feasible. In Chapter 2 of this thesis we investigate a novel proposal for a quantum computer based on the surface code approach to error correction and we look at how one can quantify and assess progress towards useful error correction in small systems in Chapter 3. In the latter half of the thesis we return to magic state distillation, with a more formal introduction, showing particular interest in analysing and quantifying the resources requirements of quantum computer based on a surface code + magic state approach to fault-tolerant, universal quantum computation.

1.4 Surface code

A particularly important class of stabiliser codes are the topological codes, of which the Kitaev surface code [42, 43] is a particularly important example. The surface code has a number of properties which make it one of the leading candidates for use in practical fault-tolerant quantum computation. It has the highest error tolerance of the stabiliser codes and requires only local stabiliser operations. There are two main versions of the surface code, named for their particular topologies: the toric code and planar code, which differ only in their boundary conditions. The toric code has periodic boundary conditions, allowing the qubits to be arranged on the surface of a torus while maintaining the geometrically local stabiliser generators. The planar code dispenses with these boundary conditions allowing the physical qubits of the code to live on a flat 2D surface. In this thesis, where we consider a particular physical system of qubits (as in Chapter 2) we will be restricted to working within a flat architecture so from here out when we refer to the 'surface code', unless specifically stated otherwise, we will be referrring to the planar variant.

The surface code is most often described by arranging the qubits on the edges of a square lattice as in Figure 1.5, although we will meet a more efficient qubit layout later.

The stabiliser group of the planar surface code is generated by four types of operators. In fact it is easiest to think of them as two types, each of which is modified slightly when appearing at the edge of the lattice. The two main types of operator are plaquette operators and star operators. A plaquette operator exists for each face of the lattice, f and is composed of Pauli Z operators acting on each qubit touching that face,

$$P_f = \bigotimes_{i \in f} Z_i.$$

A star operator exists on each vertex of the lattice v and is composed of a Pauli X acting on each qubit touching that vertex,

$$P_v = \bigotimes_{i \in v} X_i.$$

Since the eigenvalues of X and Z are ± 1, then so are those of P_f and P_v.

In the bulk (meaning far from the boundaries) of the surface code the plaquette and star operators are weight-4 operators, but those at the boundary are weight-3. As shown in Figure 1.5 the boundaries are defined by these three body stabilisers, where in this case we have a so-called 'smooth' boundary of weight-3 star operators

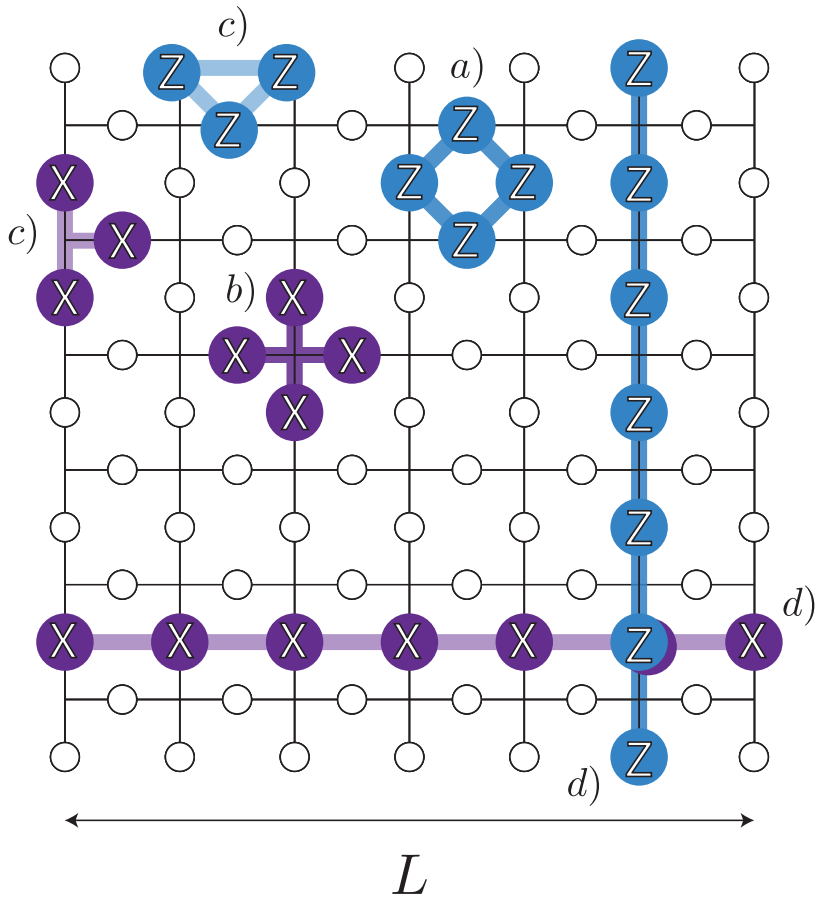


Figure 1.5: The surface code. Qubits (white circles) are arranged on the edges of a square lattice. The stabilisers of the code are divided into two types, (a) plaquette operators that live on the faces of the lattice and (b) star operators that live on the vertices. The edges of the lattice are defined by weight-3 operators (c) of each type. The logical operators (d) are formed by chains of X or Z operators that span the surface between 'like' boundaries.

at the east and west edges of lattice and the weight-3 plaquette operations defining the 'rough' edges to the north and south. The X logical operator is any string of Pauli-X operators which span the code from one rough edge to another and the Z logical operator is any string connecting one smooth edge to the other. The distance of the code is given by the lattice dimension L, the length of the shortest string that can connect two like boundaries, again as indicated in Figure 1.5.

A (planar) surface code of dimension L, as defined in Figure 1.5 has $L^2+(L-1)^2 = 2L^2-2L+1$ physical qubits and $2L(L-1)$ independent stabilisers. This leaves one degree of freedom spare, so the surface code encodes one logical qubit.

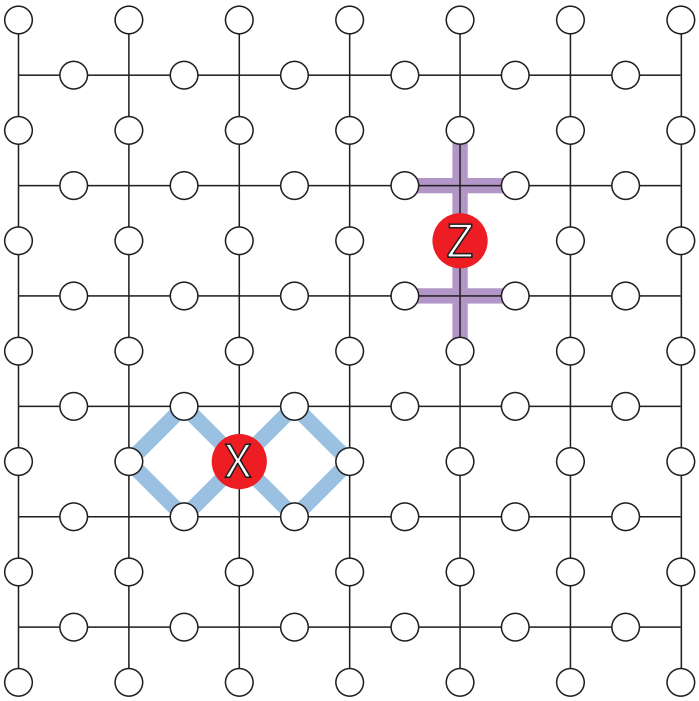


Figure 1.6: Syndrome given a single X error or Z error in the bulk of lattice. X errors are detected by the plaquettes, and Z errors by the stars.

1.4.1 Error correction in the surface code

In the bulk a single qubit Pauli error will anticommute with two stabilisers and flip their measurement outcomes as shown in Figure 1.6. An X error will flip the value of the two plaquette operators that it touches and a Z error will flip the two star operators that include it. A Pauli Y is of course a product of X and Z errors and so a Y error on a single qubit will cause all four neighbouring stabiliser operators to flip. It is however sufficient to consider only correcting X and Z errors separately since all errors can be written in terms of these two. At the boundaries of the surface code one must consider also that errors can occur that will flip only one stabiliser, we will return to this presently.

Implementing error correction will require repeated measurements of all the stabilisers. We will refer to the measurement of all the plaquette and all the star operators once as a stabiliser cycle. To make these measurements in a non-destructive fashion it is usual to introduce an 'ancilla' qubit whose job it is to interact with four neighbouring data qubits before being measured out and thus effecting the desired parity projection. The circuits which achieve this are shown in Figure 1.7. This approach in effect increases the number of physical qubits required to implement the code by up to a factor of two. Each stabiliser operator could have a unique ancilla

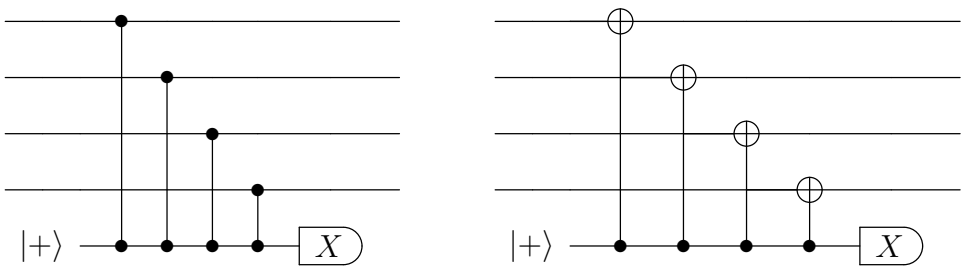


Figure 1.7: Measuring the stabilisers of the surface code. (a) The plaquette ($ZZZZ$) operator. (b) The star operator($XXXX$). For the weight-3 boundary stabiliser only three entangling operations are required.

assigned to it, which would live on the vertices of the lattice (for the stars) and the faces of the lattice (for the plaquettes). However, if one wanted to be very frugal with one's qubit resources (and had a lot of spare time) one could use a single mobile ancilla that measured the stabilisers in sequence! Regardless of these details, it is repeated measurement of stabiliser cycles that reveals errors in the *changes* of the syndrome from round to round.

1.4.2 Decoding

Clearly the task that remains is determining our correction operator C from the given syndrome that arises from a stabiliser cycle. Application of this correction operator to the data qubits of the surface code should return us to the codespace without logical error. We call this task *decoding*.

Syndrome information in the surface code does not map to a unique error pattern, since all error configurations that are equivalent up to a product of stabilisers must produce the same syndrome information. However the task of the decoder is not to perfectly determine the pattern of errors that has occurred, but to determine the correction operator that is most likely to return the code to the codespace without logical error. For a given physical error rate p we wish to find a decoder that will minimise the corresponding rate of logical error P_L.

The surface code exhibits threshold behaviour [44, 45]. That is, when p is below some threshold error rate p_{th}, the probability of logical error decreases exponentially with the size of the lattice. In other words we can arbitrarily suppress error by making the surface code larger, and because of the exponential nature of this suppression this can be achieved 'efficiently' i.e. without the number of qubits scaling exponentially with the desired logical error rate.

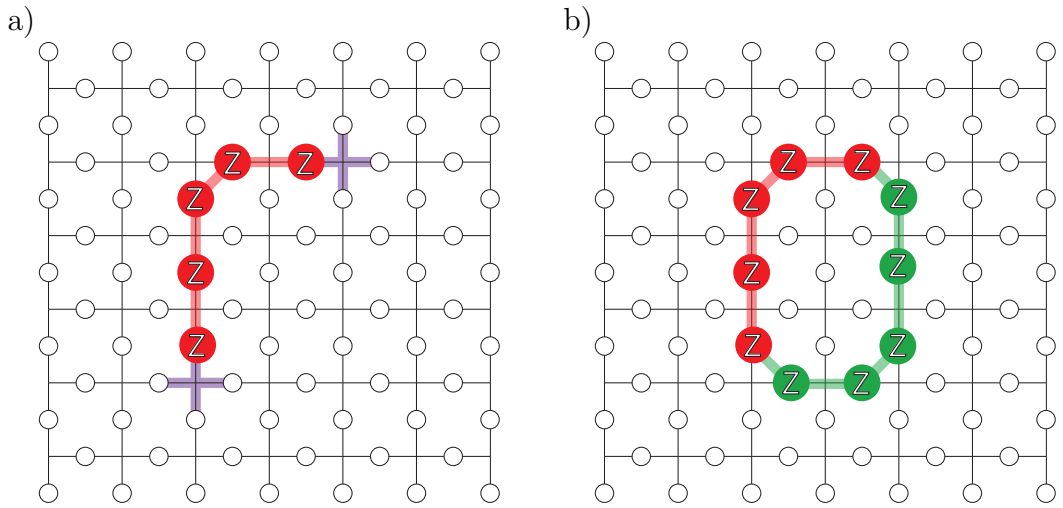


Figure 1.8: Chains of errors in the surface code. (a) Syndrome generated by a chain of errors on the surface code. The chain results in two stabiliser violations at the ends of the chain. In the rest of the chain each star is acted on by two errors, so they commute and the stabilisers' values remain the same. (b) The resulting state after applying a correction operator, which in this case is not equal to the error chain. The residual loop of errors is left on the code, but there are no longer any stabiliser violations, nor chains that are logical operators. The residual loop acts trivially on the encoded information.

1.4.3 Diagnosing error chains

While we have seen in Figure 1.6 the effect of a one qubit error on a surface code, the decoder will of course have to deal with the more complex task of determining a correction operator on a surface which has suffered multiple errors and generated a more complex syndrome.

Let us consider the example shown in Figure 1.8 which shows a chain of five Z errors occurring somewhere in the bulk of our surface code. We note that it is only at the ends of the error chain that the stabilisers report a '-1' outcome, in the middle of the chain the star operators intersect with the chain in two locations so commute with the error. As one can see, if only the end points of an error chain can be identified then there are any number of possible error chains that could have lead to this syndrome. If the decoder determines the correction operator which matches the error chain in Figure 1.8(a), so its application should directly fix the error that occurred, leaving no residual operations on the codespace. However in Figure 1.8(b) an equally likely correction operator is identified, which has a different path to the original error chain. After application of this correction operator there would seem to be an even larger number of physical errors left on the code! However, if we look

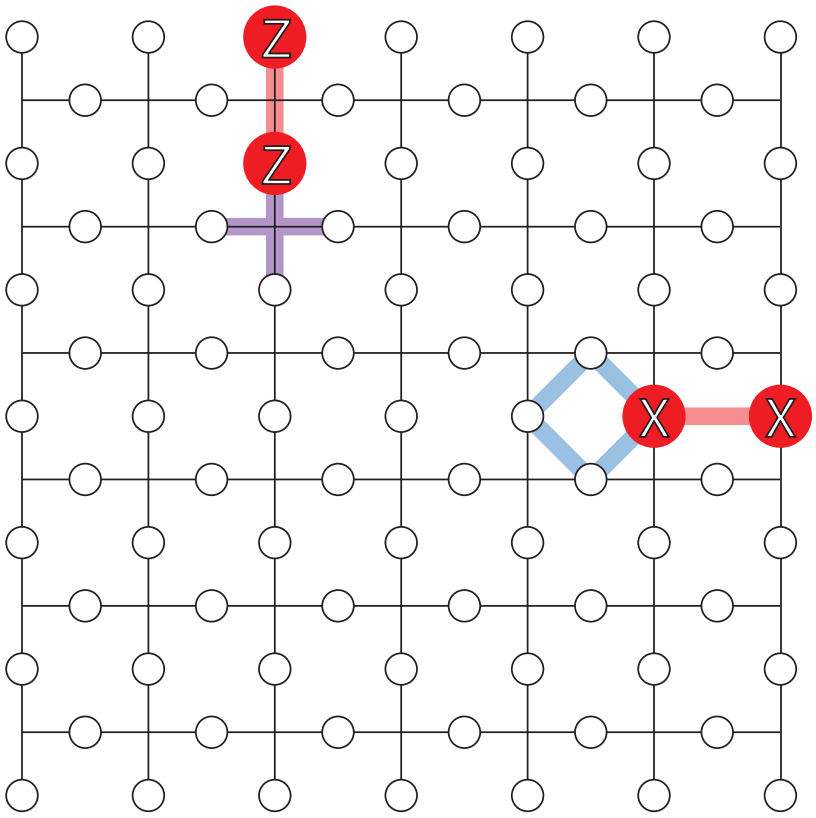


Figure 1.9: Error chains that terminate on the boundary violate only one stabiliser, where the chain terminates in the bulk.

closer we can see that this final error configuration is equivalent to the product of six plaquette stabilisers enclosed by the loop. Our leftover operation is a member of the stabiliser group and therefore has no effect on the logical state of the code! We find that this is true in general, we do not need to identify the 'correct' error configuration but merely identify a pairing of the stabiliser violations that combined with the physical error chain, forms a closed or 'trivial' loop. We refer to pairing up of the stabiliser violations as 'matching'.

When an error chain terminates at a boundary, we see something different, as in Figure 1.9. In this case only one stabiliser violation is recorded, at the point in the bulk where the chain terminates. We can see that this may lead to confusion down the line. What if we have multiple error chains, some terminating at the boundary? This complicates our matching problem slightly, as we are no longer guaranteed to have an even number of stabiliser violations to pair up. Notice that with the periodic boundaries of the toric code this issue does not arise. We will return to this problem in a subsequent section, but for now will describe a procedure for decoding the toric code.

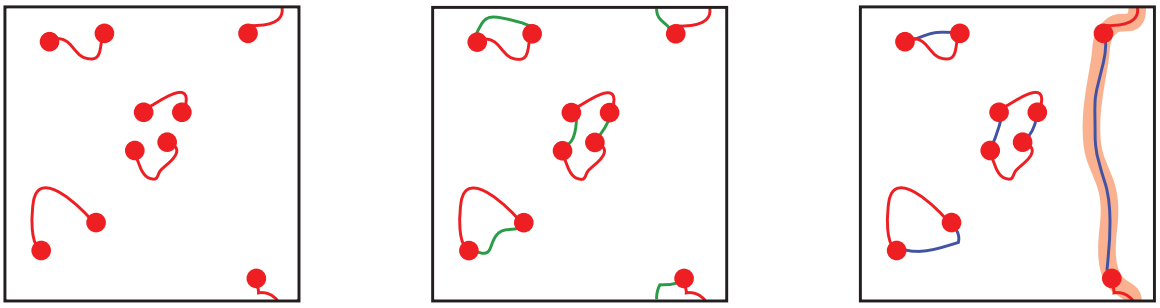


Figure 1.10: a) Cartoon of a possible error configuration (red lines) and syndrome. Stabiliser violations are indicated by the red circles. (b) A successful matching. The stabilisers are matched in such a way that the correction operators and error chains form trivial loops on the code. These are products of stabilisers and so the logical information is preserved. (c) An unsuccessful matching. One pairing has resulted in a chain of errors that spans the lattice. This is a logical operation on the qubits that we are unaware of and, as such, is a logical error.

1.4.4 Pairing syndrome violations

We have seen that loops of errors in the bulk lattice are equivalent to products of stabilisers and therefore act trivially on the logical state. For now ignoring error chains terminating at the boundary, the challenge is to 'match' or pair up the '-1' syndrome measurements such that we form trivial loops in the code and choose these pairings in a way that minimises the chance of logical error.

Figure 1.10(a) shows an example toric code (remember the periodic boundary conditions) which has suffered from chains of errors (red lines) leading to the stabiliser violations at their endpoints (red circles). The correction operators (green lines) return the state of the qubits to the codespace. Figure 1.10(b) shows a successful decoding: all the loops formed by the combination of errors and correction operators are trivial, they can be closed to a point and do not span the code. The logical information has not been corrupted. In Figure 1.10(c) we see an example of a failed correction, where the -1 stabiliser outcomes have been paired in a way that leads to a chain that spans the lattice, a non-trivial loop that does not reduce to a point. The state has now been returned to the codespace, and commutes with all the stabilisers so no errors will now be identified, but the encoded qubit has undergone an unintended and unknown logical operation: so a logical error has occured.

1.4.5 Minimum weight perfect matching

Simply by pairing the syndrome violations we can return the toric code to the codespace. What is now needed is a method of choosing a set of pairings that will

minimise the risk of logical error. The method we use is a minimum weight perfect matching (MWPM) decoder.

The MWPM algorithm aims to identify the the lowest weight error configuration that can produce the observed syndrome information. If an error occurs on a qubit with probability p then an error configuration E with weight N_E on N_Q qubits has a probability $P(E) = p^{N_E}(1-p)^{N_Q - N_E}$, and therefore,

$$P(E) \propto \left(\frac{p}{1-p}\right)^{N_E}$$

which is maximised when N_E is minimised, so the error confirguration with the smallest possible weight N_E is the most probable one. Remember that the task of our decoder is not necessarily to find the most likely, or even the 'correct', error configuration, but to find the operation most likely to fix the code. Here we are simplifying the task somewhat to consider the most likely way a syndrome could be produced rather than the more complex (in fact computationally hard) calculation that would consider all possible error configurations that could have led to the syndrome and weighting them accordingly.

For a particular pair of syndrome violations the smallest number of errors that are needed to produce it is given by the shortest possible path between them. On the surface code lattice the shortest distance between two stabilisers is given by sum of their vertical and horizontal separations, also called the Manhattan distance. The weight of a matching of all the pairs of syndrome violations is the sum of the Manhattan distances between each of the pairs of matched stabiliser violations.

Finding the set of pairings which has the overall lowest weight is a problem with a computationally efficient solution. First we map our syndrome information to a completely connected graph, with stabiliser violations as the nodes and the edges joining them assigned a weight equal to their Manhattan distance. A toric code example is shown in Figure 1.10. Finding the MWPM of this graph is then equivalent to the finding the lowest total weight error configuration that could have led to the observed syndrome. This graph matching can be solved in polynomial time using Edmonds' Blossom V algorithm [46]

1.4.6 Boundary matching

The complication introduced by the planar surface code's open boundary conditions requires a slight work-around to enable us to map our problem to an appropriate graph for which a matching can be found. In the toric code we were guaranteed an

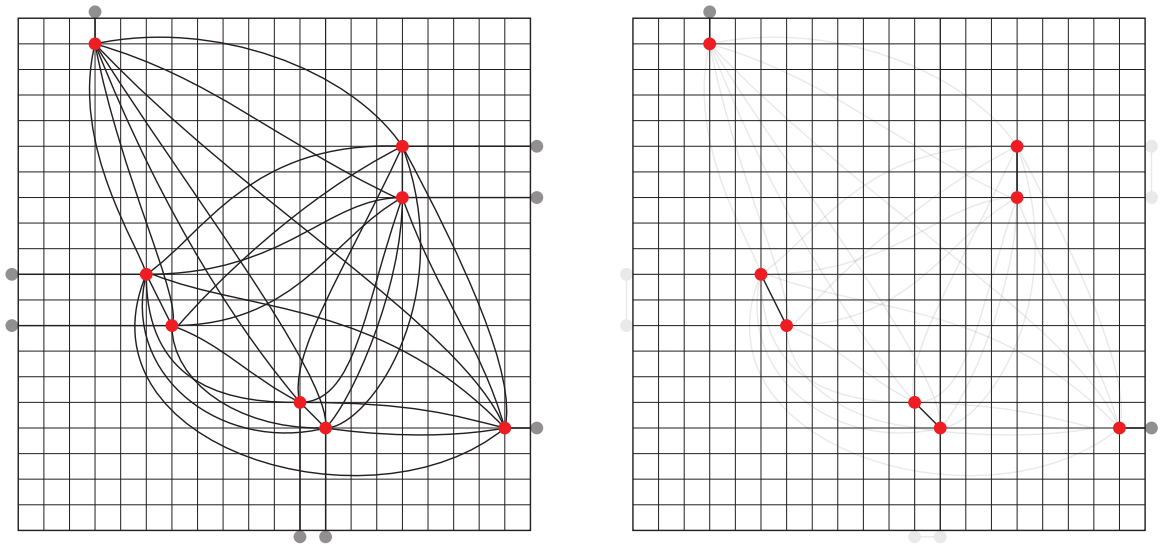


Figure 1.11: Translating the task of diagnosing the syndrome to a minimum weight perfect matching graph problem. (a) Measured syndrome violations (red) represents the nodes of a graph, which is then completely connected. Each measured syndrome violation is given a 'virtual' node to pair with at the boundary, and an edge is made between them. The 'virtual' nodes are completely connected to each other with edge weight 0. Each edge (black line) is given a weight corresponding to the Manhattan distance between them. (b) The result of finding the minimum weight configuration of edges that pairs all the nodes.

even number of syndrome violations, but this is not the case in the planar variant where an error chain can terminate on the boundary and result in only one stabiliser violation. Thus we must include a way for the node of our graph to 'pair' to the code boundary.

To do this we give every node of graph formed from the syndrome information a 'virtual node' at its nearest boundary that we will allow it to possibly pair to, assigning an edge from each node to its virtual node. This is illustrated in Figure 1.11. As such we now have an even number of nodes for our graph matching problem. This method was originally described by Wang et el. in [47]. We then proceed to find a matching as in the case of the toric code, however now if a pairing between a node and its virtual node is selected then this represents the case of an error chain terminating at the boundary. We must ensure that we assign an edge between every pair of virtual nodes with weight 0, which will force them to be matched if they are not required. On this graph we are the able to generate a pairing of syndrome violations and boundaries as shown in Figure 1.11(b).

1.4.7 Faulty Measurements

So far we have only considered a situation where physical errors occur on the data qubits and measurements are perfect. We have made no reference to what the source of these errors might be nor whether error correction is still possible if we cannot necessarily trust the information given by the stabiliser measurement. In a real experiment both of these are relevant. The specifics of how the errors occur in the system are best left to later when we consider specific implementations of the surface code. But the need to deal with faulty measurements: when a stabiliser that should say '+1' says '-1" or vice versa, is generic.

It turns out that to deal with measurement errors we must add some redundancy in time, that is making multiple rounds of stabiliser measurements in order to deal with these measurement errors. We adapt our decoder in the following way. The syndrome information now comes in the form of a three dimensional array. Rather than assign a node in our graph to each stabiliser violation we make multiple rounds of stabiliser cycles and assign a node to each location where the stabiliser value changes from round to round, see Figure 1.12. A physical error causes a pair of nodes in the spatial dimension, whereas a measurement error creates a pair of nodes in the time direction.

Having built up this 3D array of syndrome information we can then apply the matching algorithm as before 1.13. The pairs can now be both spatially and temporary separated so represent chains of error of both physical Pauli errors and measurement errors. Again a correction operator can then be applied to return the qubits to the codespace.

1.4.8 Determining thresholds

Calculating a threshold then consists of performing Monte Carlo simulations of the noisy stabiliser measurements. At each time step errors are randomly injected into the lattice according to the superoperator derived for the stabiliser measurement,

When the error rate is below threshold increasing the lattice size will increase the number of steps over which the logical information is protected in the code. The opposite is true above threshold where the noisy stabilisers are injecting errors in at a faster rate than the code can remove them. Thus to determine the threshold under a certain error model, the point at which curves of logical error rate (per fault-tolerant block of stabiliser cycles) against physical error rate for different lattice sizes cross

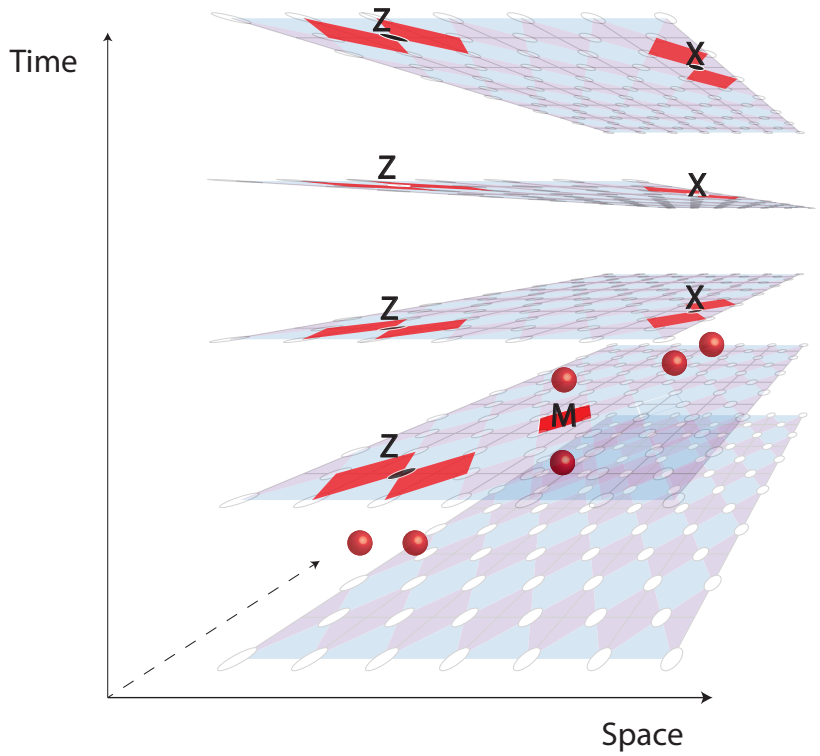


Figure 1.12: Syndrome produced by a physical X error, a physical Z error and a measurement error. The physical errors remain in the subesquent rounds of measurement so after changing value the stabilisers then remain unchanged. The nodes of the graph to be matched (red spheres) are identified as the places where stabiliser measurements have changed from one round to the next. For physical errors they occur next to each other in the spatial dimensions. When a measurement error occurs a single stabiliser changes value and returns to its previous value when the correct measurement is made in the next round. This produces the characteristic signal of a measurement error, two nodes next to each other in the time direction.

must be determined. Wang et al. [47] examined the behaviour of the surface code close to threshold and determined that the probability of logical failure is given by

$$P_{\text{fail}} = (p - p_{th})L^{1/\nu_0} \tag{1.7}$$

for sufficiently large lattice parameter L. The three-qubit 'edge' stabiliser of the planar code variant introduce some finite size effects which can be accounted for by fitting to a quadratic function

$$P_{\text{fail}} = a + b(p - p_{th})L^{1/\nu_0} + c(p - p_{th})L^{2/\nu_0} \tag{1.8}$$

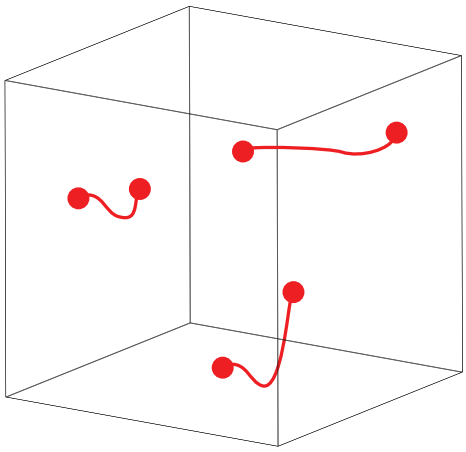


Figure 1.13: Example of a 3D syndrome cube after L rounds of measurement in an $L \times L$ surface code. Error chains (red lines) are made up of physical errors, which cause space-like separation of the nodes (red spheres) and measurement errors result in the time-like separation.

1.4.9 Stabilisers as superoperators

We may want to investigate a number of protocols to implement stabiliser operations on the surface code, with different kinds of error occurring and at different rates. We wish to know what actual error prone operation of the stabiliser looks like on the state of the surface code. These operations may work better on some states than others but calculating a state fidelity would require that we introduce a specific input state. We would prefer to understand the fidelity of the stabiliser (or indeed a generic gate or operation) as an operation without reference to a particular instance of an input state. We want to describe the action of a noisy stabiliser as a *superoperator*.

If we have a desired unitary interaction on a set of a qubits $\mathcal{Q}$ in a state $\rho_{\mathcal{Q}}$, in the ideal case the transformation is simply described by a single unitary operator U which transforms the state to $\rho_{\mathcal{Q}} \rightarrow U\rho_{\mathcal{Q}}U^{\dagger}$. However we want to model the non-ideal scenario when such a gate is implemented in reality and is inevitably afflicted by noise, e.g. a depolarising channel. Because the noise is probabilistically implemented, the true map that describes the operation $\mathcal{E}$, where $\rho_{\mathcal{Q}} \rightarrow \mathcal{E}(\rho_{\mathcal{Q}})$, cannot be written as a single unitary operator.

In addition to noise making the operation non-unitary, the process of making a stabiliser measurement, or some other operation, may require the use of an ancilla space $\mathcal{A}$, as well as the qubits $\mathcal{Q}$ on which we wish to perform our gate. This ancilla space is measured out and discarded in the process of implementing the gate. In general such an operation can clearly not be described by a single unitary operation

on the subspace $\mathcal{Q}$. In the reduced space the operation must be described in a superoperator of the form

$$\mathcal{E}(\rho) = \sum_i p_i K_i \rho K_i^\dagger \tag{1.9}$$

which corresponds to a series of operators $\{K_i\}$ being performed on the state ρ with probability p_i. This superoperator fully describes the operation of the gate.

Such a decomposition is a natural way to describe the fidelity of the gate's operation. If we aim to perform some operation $\mathcal{P}$, but do so with some noisy process, then the true superoperator $\mathcal{E}$ would be expected to have a dominant Kraus operator $K_0 = \mathcal{P}$ and a corresponding probability p_0 hopefully somewhere close to 1, with a series of other Kraus operators $K_i \neq 0$ which each represent a faulty/undesired operation. The value of p_0 then gives the gate's fidelity.

Consider our realistic protocol $\mathcal{P}$ that we would like to implement on a Hilbert space $\mathcal{H}_\mathcal{Q}$. This Hilbert space is made up of the data qubits $\mathcal{Q}$ on which we want to perform the gate. An ancillary Hilbert space $\mathcal{H}_A$ which is measured out over the computation is also introduced. Operation of $\mathcal{P}$ takes a state in the larger space to one in the reduced space $\mathcal{P}(\rho_\mathcal{Q} \otimes \rho_A) = \rho'_\mathcal{Q}$.

The Choi-Jamiolkowski isomorphism

In order to describe the map we use the Choi-Jamiolkowski isomorphism [48, 49] which allows us to study the effect of linear maps of operators. To find the elements of the map $\mathcal{E}$ we can consider how the same protocol would act on the state where each data qubit in $\mathcal{Q}$ is replaced by one half of a maximally entangled bell pair, $|\phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ to create an operational state

$$|\Phi\rangle = \left|\phi^+\right\rangle \otimes \left|\phi^+\right\rangle \otimes ... \otimes \left|\phi^+\right\rangle = \frac{1}{\sqrt{2^d}} \sum_{x=0}^{2^d-1} |x, x\rangle \tag{1.10}$$

Intuitively we are 'tagging' each qubit in $\mathcal{Q}$ with a reference $\mathcal{Q}_R$. The maps $\mathcal{E}$ alters the state of the qubits in $\mathcal{Q}$, leaving the 'references' unchanged. When the protocol is completed, the reference qubits are used to determine how each component was transformed during the process. The introduction of these reference qubits is a purely theoretical tool. We follow the same same procedure used in deriving the superoperator for noisy stabilisers in Ref [50].

Applied to a state Φ the total map is described by

$$(\mathcal{E} \otimes \mathcal{I})\Phi = \sum_i p_i (K_i \otimes \mathcal{I})\Phi(K_i \otimes \mathcal{I})^\dagger \tag{1.11}$$

$$= \sum_i p_i \Phi_{\mathcal{E}}^{(i)} = \Phi_{\mathcal{E}} \tag{1.12}$$

Each Kraus operator can be decomposed in the computational basis as,

$$K_i = \sum_{a,b \in \{0,1\}^d} = K_{ab}|a\rangle\langle b| \tag{1.13}$$

The states $\Phi_{\mathcal{E}}^{(i)}$ are found by operation with a single Kraus operator on the initial state. Applying operators of the form of (1.13) to the state Φ each element can be written:

$$\left|\Phi_{\mathcal{E}}^{(i)}\right\rangle = K_i|\Phi\rangle = \frac{1}{\sqrt{2^d}} \sum_{a,b} K_{ab}|a\rangle\langle b| \sum_{x=0} |x, x\rangle \tag{1.14}$$

$$= \frac{1}{\sqrt{2^d}} \sum_{a,b} \sum_{x=0} K_{ab}|a\rangle\delta_{bx}|x\rangle \tag{1.15}$$

$$= \frac{1}{\sqrt{2^d}} \sum_{a,b} K_{ab}|a\rangle|b\rangle \tag{1.16}$$

We can then see a direct correspondence between the form of the Kraus operation K_i and the state $\left|\Phi_{\mathcal{E}}^{(i)}\right\rangle$. The output state of our noisy map applied to the Bell pairs, $\Phi_{\mathcal{E}}$, can be diagonalised to find its eigenvectors $\frac{1}{\sqrt{2^d}} \sum_{a,b} K_{ab}|a\rangle|b\rangle$, which by simple rearrangement can be used to form the Kraus operators $K_{ab}|a\rangle\langle b|$. The corresponding eigenvalues represent the relevant probabilities p_i in the Kraus decomposition.

The Kraus operators and their probabilities are all that are required to fully describe the map in the reduced subspace $\mathcal{Q}$. We have been able to determine this in a single 'run' by the action of $\mathcal{P}$ on Φ, rather than simulation of a range of different input states. The Kraus decomposition, however, is not unique and often there will exist a particular set of operators, $\{\bar{K}_i\}$, which make physical sense to represent the noise that has occurred. For example, we would often expect that that resulting operations can be decomposed in to products of the intended operation (say a four-qubit even parity projection) and a combination of Pauli errors acting on one or more of the qubits, $\bar{K}_i = K_0 \cdot (\sigma_j \otimes \sigma_k \otimes ... \otimes \sigma_n)$ where $j, k, ...n = 0, 1, 2, 3$.

The utility of this method of determining the superoperator is clear. The protocols used to implement a gate or a series of gates to effect a stabiliser measurement may

be lengthy and require a large ancilla space, which can make the computation slow to perform numerically. With this method the full simulation (1.11) need only be performed once to determine the elements of the superoperator, after which it can be applied directly to the state.

1.4.10 Computing with the surface code

With the ability to perform the stabiliser measurements, combined with a technique such as magic state distillation (of which more in Chapters 4-6), all the operations required of a universal quantum computer can be performed simply by selecting when and where to enforce stabiliser measurements. Originally the proposed method of performing the multi-qubit gates between the encoded qubits of the surface code was to do so transversally. In this picture a CNOT gate between logical qubits, each stored in its own surface code, would be performed by performing CNOT gates between each individual physical qubit in the lattice and its opposite number in the second lattice. This method presents something of a challenge in most experimental qubit systems, which are often restricted to a two dimensional architecture with only nearest neighbour interactions. Happily two methods exist which provide the ability to perform two-qubits gates while still needing only nearest-neighbour interactions. The first of these proposed in Ref. [51] by Kitaev in 2003 used defect-based codes and the second, known as lattice surgery was introduced by Horsman et al. in Ref. [52] in 2012.

Braiding defects

Defect-based use of the surface code defines logical qubits as defects on a single piece of surface code. Previously, the ability to encode a logical qubit in a planar code was described as resulting from the degree of freedom remaining after the evaluation of all the stabilisers. In the defect-based picture more degrees of freedom are introduced by choosing not to enforce certain stabilisers - creating defects in the code. Deforming and braiding of these defects then allows the implementation of two-qubit gates, all the while still only requiring nearest-neighbour interactions between physical qubits in a two dimensional surface. The cost of maintaining these desirable properties, which would have to be sacrificed to perform the gates transversally, is that each logical qubit you wish to encode requires over three times as many physical qubits [52] as a transversal implementation. The cost is greater at lower distances of code: for example the smallest planar surface code which can correct an error (a distance 3

code) has lattice dimension L = 3 which requires 13 physical qubit. A canonical defect based surface code for a single qubit of the same distance requires 72 physical qubits. An excellent and comprehensive review of implementing a full universal quantum computer using a defect-based surface code is outlined in [32] by Fowler et al..

Lattice surgery

Lattice surgery is another method of implementing the multi-qubit gates within the surface code without resorting to transversal implementations. As opposed to the defect-based paradigm, lattice surgery cuts and stitches together planar surface codes of the form described in Section 1.5 to produce other planar surfaces in states corresponding to a parity projection of the initial surfaces, without any transversal interactions. These operations essentially boil down to enforcing or not enforcing stabilisers across the boundaries of neighbouring pieces of surface code. Ref [52] outlines the method of performing CNOT gates, Hadamards and magic state injection; all the components necessary for universal quantum computation. The resource use of the lattice surgery technique is shown to be less than that of defect-based codes. Although this saving is modest, for the first likely demonstrations of two-qubit gates in the surface these would be significant. The smallest (distance 3) lattice surgery CNOT requires 53 physical qubits, whereas the smallest known CNOT operation between two defect qubits in a surface code requires 104 qubits. Another advantage of the lattice surgery approach is that it makes it easy to see the impact of defective qubits. When using the lattice survey approach, since logical qubits can live a specific blocks, if a high number of defective qubits occur within a small region this block can be declared 'dead' and not used. Provided these are not too frequent information can be routed around the 'dead pixels' of the device. Improvements to the lattice surgery picture have been made in work extending the methods to the colour codes [53]. The methods outlined in this paper allow modest overhead savings in the merging and splitting of different blocks of surface code.

Chapter 2

A silicon-based surface code quantum computer

In this chapter a novel method for implementing the surface code using donor qubits in silicon is presented. In Section 1 the physics of the proposed 'orbital probe' scheme is described. In Section 2 an assessment of the surface code threshold is made with reference to an error model tailored to the proposal. In Section 3 a brief comment is made on how a surface code machine can perform universal quantum computation In Section 4 we comment on the feasibility of the scheme in light of those results - this section is informed heavily by the expertise of John Morton and Phillip Ross who are coauthors of the published paper on which the chapter is based [54]. In Section 5 a preliminary analysis of an alternative realisation of the scheme is presented. All calculations and threshold simulations described in this chapter were performed by the author. Note that the text of this chapter is adapted from the text the author wrote for Ref. [54] and the code written for the numerical simulations is openly available online (please see '/naominickerson/fault_tolerance_simulations/releases' on GitHub).

Classical computers were once built from vacuum tubes and weighed tonnes. Their use as personal machines, that would fit in a person's home or pocket was once unimaginable. However, the invention of the silicon transistor and the subsequent semiconductor revolution enabled vast scaling and miniaturisation. It is reasonable to hope that some of this vast expertise could be brought to bear on quantum systems. An influential early paper exploring this possibility was written by Kane [55] in 1998. According to this proposal, impurity atoms implanted in a purified silicon substrate constitute the means of storing qubits. Operations between qubits would occur through *direct* contact interactions between such spins, which necessitated an inter-qubit spacing of at most nanometers (and therefore a precision considerably

beyond than this) to allow the wavefunctions of the bound electrons to overlap and mediate the interaction. This also required exquisitely small and precisely aligned electrode gates above and between donors. This proposal proved highly influential and progress toward realising it has been made at both the theoretical [56] and experimental level, including impurity positioning via STM techniques that have achieved nanometer precision [57, 58]. However it remains extremely challenging as a path to practical quantum computing.

In later chapters we will explore the overhead of magic state distillation which will underline the importance of qubit technologies which can support control and addressability of many millions of qubits in order to achieve useful fault-tolerant quantum information processing. With this point in mind it is important not only to focus on technologies which can demonstrate high quality operations on very small quantum systems in the short-term, but also to explore technologies for which the route from 'proof-of-principle' experiments to full-scale commercial devices might be (relatively speaking) more straightforward.

This chapter introduces a novel concept for universal quantum computation based in a silicon-donor based architecture. In view of the power and elegance of the surface code, ideas of the Kane proposal are revisited and an engine designed 'from the bottom up' to efficiently perform stabilizer measurements is proposed. We find that one can abandon the need for direct gating between physical qubits, and thus the need for extreme precision in the location of impurities and the equally challenging gating of qubit-qubit interactions. Instead we exploit long-range dipole fields rather than contact interactions, and we are thus able to set the scale of the device according to our technological abilities.

The scheme - which we often refer to as the 'orbital probe scheme' - comprises two physically separated silicon wafers each of which contains a two-dimensional array of donor qubits which interact with each other via dipolar spin-spin coupling. The repeating motion of one of these platforms controls the interaction strength between the qubits of the two wafers. This motion allows the four-qubit stabiliser generators of the surface code to be measured and from this repeated process a universal quantum computer can be constructed.

The work in this chapter was the result of a collaboration with Philipp Ross and Prof. John Morton of UCL and the London Centre for Nanotechnology. They contributed toward the original conception of the scheme and analysis of the fabricational and mechanical feasibility detailed in Sec. 2.4. All simulations and calculations pre-

sented were performed by the author. The results of the work have been published in Nature Partner Journal: Quantum Information [54].

2.1 Principles of device operation

2.1.1 The 'orbital probe' parity measurement.

The basic principle underlying the scheme is shown in Fig. 2.1(a). In the figure, four spin-$\frac{1}{2}$ particles referred to as 'data qubits' are embedded in a static lattice. In practice these are likely to be electrons bound to isolated donor impurities in silicon, which we describe in more detail later. Meanwhile another spin-$\frac{1}{2}$ particle is associated with a mechanically separate element which can move with respect to the static lattice. We assume this 'probe spin' [59, 60] is also electronic, for example, either a different species of donor in silicon or an NV defect centre in diamond. It will be necessary to prepare and measure the state of the probe; this might be achieved via spin-to-charge conversion for donors in silicon or, alternatively, by optical means for the NV centre. There are two key dimensions: the vertical distance between a probe qubit and a data qubit at closest approach, d, and the separation between qubits in the horizontal plane, D. It is important that $d \ll D$, in order that any interactions between the in-plane spins are relatively weak. As we will discuss, the optimal choice of dimensions varies with several factors including the nature of the mechanical movement; but as an example, for one realisation of the system $d = 40\,\text{nm}$ and $D = 400\,\text{nm}$ will prove to be appropriate. For comparison, note that commercial disk drives can achieve a 3 nm 'flying height' between read/write head and platter. Given this setup, our goal is to measure the parity of the four data qubits i.e. to make a measurement which reports 'even' and leaves the data qubits in the subspace $\{|0000\rangle, |1100\rangle, |0011\rangle, |0110\rangle, |1001\rangle, |0101\rangle, |1010\rangle, |1111\rangle\}$, or which reports 'odd' and leaves the four data qubits in the complementary subspace.

In section 1.2.2 we saw that a stabiliser measurements could be simply constructed from a set of commonly used abstract quantum gates and preparation and measurement of single qubits. However we do not necessarily have to use, for example, a CNOT or CPHASE gate as our two-qubit entangling operation. The physics of a given system may give rise more naturally to a different type of gate which can be used directly to achieve the same overall operation or circuit. In the system we investigate here we can perform an operation that is essentially identical to a CPHASE by exploiting the dipole-dipole interaction between the probe and the nearby data qubit. In our scheme the separation between data qubits is at least 10 times greater than

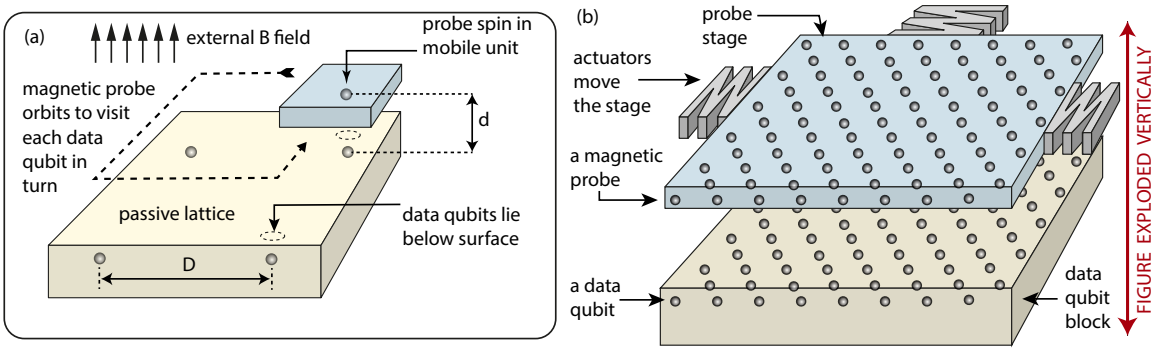


Figure 2.1: (a) The principle of the orbital probe parity measurement: a probe spin comes into proximity with 4 data qubits in turn during one cycle. (b) Simplified schematic of a scalable device showing that the probe layer and the data qubit layer both contain extended spin arrays (details of the their relative positions are shown in Fig. 2.3). We depict the probe stage as mobile while the data qubit stage is static; but in fact either may move, it is their relative motion that is key. [Figure credit: Simon Benjamin].

the probe-data separation, thus the interaction of the probe and the three data qubits to which it is not immediately proximal is three orders of magnitude weaker and can be treated as negligible, to an excellent approximation. Therefore the Hamiltonian of interest that of two $S = \frac{1}{2}$ spins, each in a static B field in the Z direction and experiencing a dipole-dipole interaction with one another, which is

$$H_{2S} = \mu_B B(g_1\sigma_1^z + g_2\sigma_2^z) + \frac{J}{r^3}\big(\boldsymbol{\sigma}_1 \cdot \boldsymbol{\sigma}_2 - 3(\hat{\boldsymbol{r}} \cdot \boldsymbol{\sigma}_1)(\hat{\boldsymbol{r}} \cdot \boldsymbol{\sigma}_2)\big).$$

Here $\boldsymbol{r}$ is the vector between the two spins, and $\hat{\boldsymbol{r}}$ is the unit vector in this direction and $J = \frac{\mu_0 g_e^2 \mu_B^2}{4\pi}$. In the present analysis we assume that the Zeeman energy of the probe spin *differs* from the Zeeman energy of the data qubit by an amount $\Delta = \mu_B B(g_1 - g_2)$ and this *detuning* is orders of magnitude greater than the dipolar interaction strength J/r^3, a condition that prevents the spins from 'flip-flopping', as shown in [61]. They thus can only mutually acquire phase through their interaction. Then in a reference frame that subsumes the continuous Zeeman evolution of the spins their interaction is simply of the form

$$S(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \exp(i\theta) & 0 & 0 \\ 0 & 0 & \exp(i\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \left(\cos\frac{\theta}{2}\right)\mathbb{1} - i\left(\sin\frac{\theta}{2}\right)Z_1Z_2. \qquad (2.1)$$

where the expressions discard irrelevant global phases, $\mathbb{1}$ is the identity and Z_1, Z_2 are Pauli matrices acting on the two spins respectively.

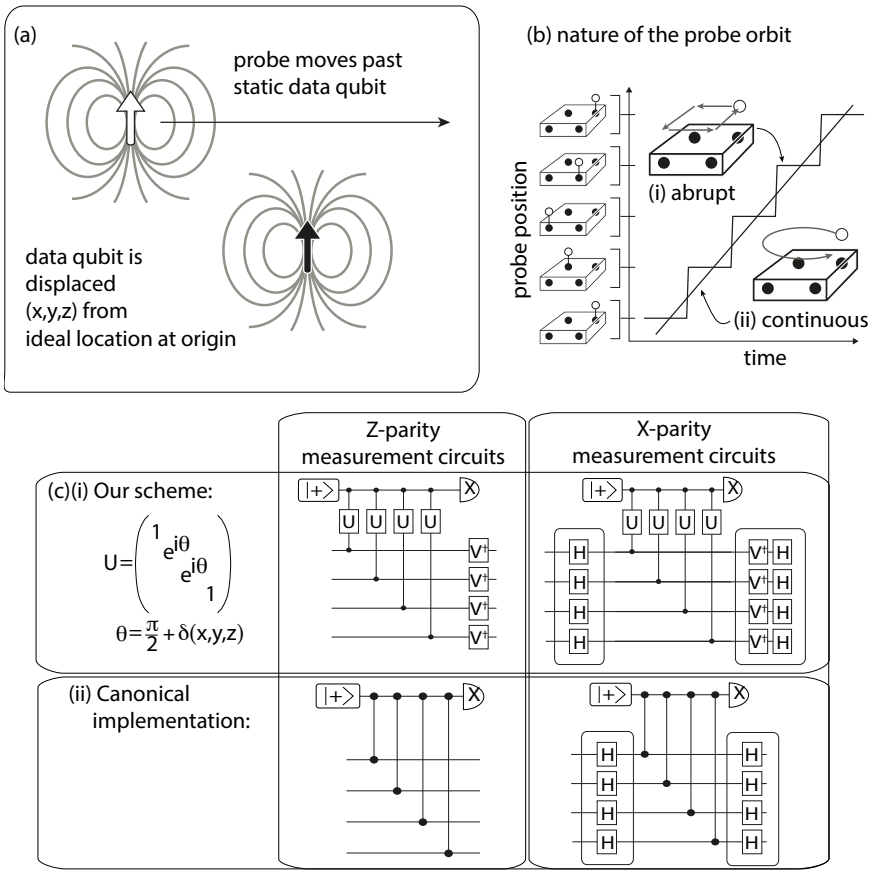


Figure 2.2: The physical process of parity measurement. (a) The probe and a data qubit move past one another and in doing so a state dependent phase shift occurs. (b) We consider two ways in which the probe may move: abruptly, site to site, or in a continuous circular motion. (c) The net phase acquired by a probe as it transits the cycle of four data qubits reveals their parity, but nothing else. (i) Two equivalent circuits for measuring the Z-parity of four data qubits. Using $U = S(\pi/2)$, as we propose and fixing the unconditional phases V on the data qubits is equivalent to (ii) the canonical circuit composed of controlled-phase gates. Note that only global (boxed) operations are required on the data qubits and the X- and Z-parity measurements differ only by global Hadamard operations. A full description of the noisy circuit is deferred to Appendix 2.2.2.

The condition that $\Delta \gg J/r^3$ will certainly be met if, as we suggest, the probe and data qubits are of different species. Suppose that the data qubits are phosphorus donors in silicon, while the probes are NV centres in diamond. The zero-field splitting of an NV centre is of order 3 GHz, while there is no equivalent splitting for the phosphorus donor qubit; this discrepancy implies Δ is more than six orders of magnitude greater than J/r^3 (the latter being of order 0.8 kHz at $r = 40$ nm). A similar conclusion can be reached even if the probe and data qubits are both silicon-based, for example if each probe is a bismuth donor and each data qubit is a phosphorus donor. Given the hyperfine interactions strengths for phosphorus and bismuth donors of 118 MHz and 1475 MHz respectively, the typical minimum detuning between the two species is nearly six orders of magnitude greater than J/r^3. We performed exact numerical simulation of the spin-spin dynamics with Mathematica using these values, finding as expected that deviations from the form of $S(\theta)$ given above are extremely small, of order 10^{-4} or lower.

For our purposes $U = S(\pi/2)$ is an ideal interaction: it is equivalent to the canonical two-qubit phase gate CPHASE (up to an irrelevant global phase) if one additionally applies local single-qubit gates $V^\dagger = diag\{1, -i\}$ to each qubit. Thus the desired four-qubit parity measurement is achieved when the probe experiences an $S(\pi/2)$ with each qubit in turn, followed by measurement of the probe and application of $V^\dagger$ to all four data qubits (see Fig 2.2(d)).

Our goal is therefore to acquire this maximum entangling value of $\theta = \pi/2$ during the time that the two spins interact. Consider two basic possibilities for the way in which the mobile probe spin moves past each static data qubit, shown in Fig. 2.2(b). The first possibility is that the probe moves abruptly from site to site, remaining stationary in close proximity to each data qubit in turn. In this case, we simply have $\theta = \alpha t$ where $\alpha = \frac{\mu_0 g_e^2 \mu_B^2}{4\pi d^3}$ i.e. the phase acquired increases linearly until the probe jumps away. The motion of the probe between sites is assumed to be on a timescale that is very short compared to the dwell time at each site; in practice this motion might be in-plane or it might involve lifting and dropping the probe.

An alternative which might be easier to realise is that the probe moves continuously with a circular motion. Because the data qubits are widely spaced, from the point of view of a data qubit the probe will come in from a great distance, pass close by and then retreat to a great distance. The interaction strength then varies with time; but by choosing the speed of the probe we can select the desired total phase shift, i.e. we again achieve $S(\pi/2)$. The nature of the circular orbit has positive consequences in terms of tolerating implantation errors, as we presently discuss (see

Fig. 2.5 upper panels versus lower panels). However, our simulations indicate the continuous circular motion does slow the operation of the device by approximately a factor of ten as compared to abrupt motion; therefore there is more time for unwanted in-plane spin-spin interactions to occur (see Sec. 2.2.4). To compensate we may adjust the dimensions of the device, for example choosing $d = 33$ nm with $D = 700$ nm will negate the increase.

2.1.2 Performing rounds of parity measurements

In the interest of preserving our quantum information and simplifying the task of manipulating it, it might be desirable to perform the fewest possible operations on the qubits. Even better would be restrict a large amount of the control operations to a subset of the qubits, perhaps particularly if these are shorter-lived qubits, such as the probes. To this end, we establish that it suffices for our device to have local control *only* of the probe qubits, in order to prepare and to measure in the X, Y or Z basis. To achieve a probe state initialisation in the X or Y basis, we rely on initialisation in the Z eigenstate basis and subsequent spin rotations using the local probe control. Global control pulses suffice to manipulate the probe qubits during their cycles, and moreover the data qubits can be controlled entirely through global pulses. The surface code approach to fault tolerance requires one to measure parity in both the Z and the X basis. Crucially, both types of measurement can be achieved with the same probe cycle. The X-basis measurements differ from the Z-basis simply through the application of global Hadamard rotations to the data qubits before, and after, the probe cycle (see Fig. 2.2 rightmost). Note that the additional phases $V^\dagger$ required in our protocol are easily accounted for in the scheme e.g. by adjusting the next series of Hadamard rotations to absorb the phase.

However the surface code protocol does require that the data qubits involved in X- and Z-basis stabilizers are grouped differently. This can be achieved in a number of ways; generally there is a tradeoff between the number of probe qubits required, the time taken to complete one full round of stabilizer measurements, and the complexity of the motion of the moving stage. Three possible approaches are detailed in Fig. 2.3. The fastest protocol involves manufacturing identical probe and data grids, and corresponds to the simplest mechanical motion of the moving stage: it can be performed with continuous circular movement. In this approach there must be a method of 'deactivating' probes which are not presently involved in the parity measurements. One can achieve this by preparing probes in the $|0\rangle$ states so that they do not entangle with the data qubits, instead only imparting an unconditional

phase shift (the correction of which can be subsumed into the next global Hadamard cycle). Our simulations assume solution (b) from Fig. 2.3; this solution divides a full cycle into four stages, but has the considerable merit that it requires the fewest probes and therefore the lowest density for the measurement/initialisation systems.

The same 'deactivation' of probes also allows us the flexibility to perform either three- or two-qubit stabilizers should we wish to, or indeed one-qubit stabilisers i.e. measurement of a specific data qubit. This can be achieved without altering the regular mechanical motion by appropriately timing the preparation and measurement of a given probe during its cycle: it should be in the deactivated state while passing any qubits that are not to be part of the stabiliser, but prepared in the $|+\rangle$ state prior to interacting with the first data qubit of interest. Probe measurement to determine the required stabiliser value should be performed in the Y-basis after interacting with one or with three data qubits, or in the X-basis after interacting with two data qubits. Note that the simple phase shifts induced by the 'deactivated' probe can either be tracked in the classical control software, or negated at the hardware level by repeating a cycle twice: once using $|0\rangle$ for the deactivated probe and once using state $|1\rangle$.

2.2 Errors, thresholds and results

2.2.1 Random error

The description above is in terms of ideal behaviour. To demonstrate the feasibility of the scheme we must analyse the behaviour of the scheme in the presence of error. Simulation of the surface code allow us to determine the threshold behaviour under the action of an error model appropriate to the system. In our simulations we combined the outlined 'perfect' procedure with a comprehensive set of error sources as specified in Fig. 2.4. Our error model is the standard one in which, with some probability p, an ideal operation is followed by an error event: a randomly selected Pauli error, or simple inversion of the recorded outcome in the case of measurement. Further details of this 'discretisation' process and how the numbers enter the simulations are given in Chapter 1.2.1. Figure 2.4 shows that the numbers used in our simulation are compatible with the values found in the literature for phosphorus in ^{28}Si, as discussed presently. The relative importance of the different errors is explored in additional simulations presented in Section 2.2.5.

In the spirit of tailoring an error model specific to the system, we modify the form of two-qubit gate error that we met in Chapter 1. Let us introduce 'jitter' – random variations in the interaction between the probe and a data qubit. We assume that the

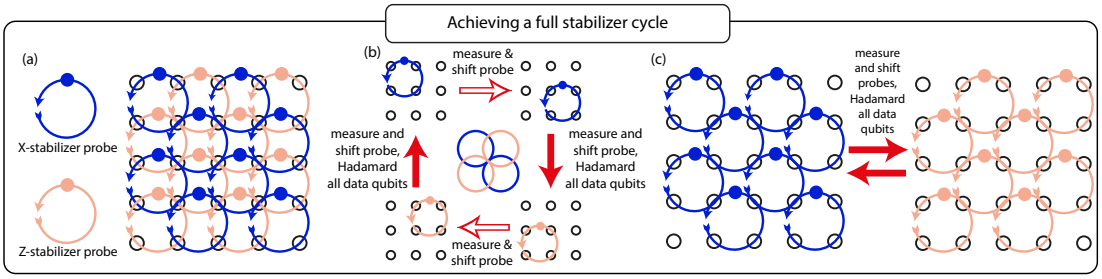


Figure 2.3: Three approaches to implementing the full surface code. (a) The probe stage is manufactured with an identical lattice to the data qubits. In this approach all the X-stabilizer operations are performed in parallel, with the probes for the Z-stabilizers made "inactive" by preparation in the $|0\rangle$ state. A global Hadamard is then performed on the data qubits. Finally, the Z-stabilizers are all measured with the X-probes made inactive. The correction of the extra phase acquired by interaction with inactive probes can be subsumed into the global Hadamard operations. If the time for a probe to complete one orbit is τ then this approach takes 2τ to complete a full round of stabilizer measurements. (b) The probe stage has 1/4 the qubit density of the data lattice. All probes are "active" (prepared in $|+\rangle$) throughout. A more complex probe orbit is required to achieve this approach: here a "four leaf clover" motion. This protocol has time cost $\sim 4\tau$ per round. This is the approach simulated to produce our threshold results. (c) The probe stage is manufactured with 1/2 the qubit density of the data stage. An abrupt shift of the probe stage is required between the rounds of X- and Z-stabilizers. All probes are "active" throughout and this method requires time $\sim 2\tau$ per round.

actual phase acquired in the two-qubit operation S varies randomly and uniformly between limits $\frac{\pi}{2} \pm \phi_e$, and in our simulations we found we could set $\phi_e = (0.044)\frac{\pi}{2}$, i.e. 4.4% of the ideal phase, before there was any appreciable impact on the threshold. We therefore selected this level of jitter, as reported in the table within Fig. 2.4. The reason for this remarkable level of tolerance is a quadratic relation between the unwanted phase shift (which is proportional to physical imperfections such as, e.g., a timing error) and the actual probability of a discretised error entering the model. The following text summaries the argument.

Consider first the simple bimodal case where phase $\frac{\pi}{2} + \phi_b$ occurs with probability one half, and $\frac{\pi}{2} - \phi_b$ occurs with probability one half. The evolution of the quantum state can therefore be written as the ideal phase gate $S(\frac{\pi}{2})$ followed by an error mapping ρ of the form

$$\begin{aligned} \rho &\rightarrow \frac{1}{2} S(\phi_b)\, \rho\, S^\dagger(\phi_b) + \frac{1}{2} S(-\phi_b)\, \rho\, S^\dagger(-\phi_b) \\ &= \left(\cos^2 \frac{\phi_b}{2}\right) \rho + \left(\sin^2 \frac{\phi_b}{2}\right) Z_1 Z_2\, \rho\, Z_1 Z_2 \end{aligned} \tag{2.2}$$

where Z_1 and Z_2 are Pauli operators on the probe and data qubit and $S(\phi)$ was defined in Eqn. 2.1. This is therefore equivalent to discrete error event $Z_1 Z_2$ occurring with probability $\sin^2(\phi_b/2)$ or approximately $\phi_b^2/4$ for small ϕ_b.

If we now consider any symmetric distribution of possible phase errors, i.e. a probability density $p(\phi)$ of an unwanted phase shift between ϕ and $\phi + d\phi$ where $p(-\phi) = p(\phi)$, then can can simply match positive and negative shifts as above and integrate, so that our mapping is $\rho \rightarrow (1-\epsilon)\rho + \epsilon Z_1 Z_2\, \rho\, Z_1 Z_2$ with

$$\epsilon = 2 \int_0^\infty p(x) \sin^2(\frac{x}{2})\, dx.$$

Taking our uniform distribution of phase error from $+\phi_e$ to $-\phi_e$ one finds that $\epsilon \simeq \phi_e^2/12$ for small ϕ_e. With our choice of a 4.4% jitter, i.e. $\phi_e = (0.044)\frac{\pi}{2}$, this corresponds to $\epsilon = 4 \times 10^{-4}$ i.e. a very small 0.04% probability of the $Z_1 Z_2$ error event.

Errors associated with our active manipulations (initialisation, periods of interaction, control pulses, measurements) are modelled as occurring at the time of that manipulation. Meanwhile it is convenient to model 'environmental decoherence' on our data qubits, i.e. errors that are not associated with our active manipulations, by applying Pauli errors at the end of each round of stabilizer measurements. In reality such effects can occur at any time; however the consequences of any change to the state of a data qubit arise only once that qubit interacts with a probe. Introducing

errors discretely at a fixed time effectively accounts for the accumulation of error probability over the time the data qubit is isolated (and this is the majority of the time: $\sim 98\%$ for the circular orbit model). There will be a some small decoherence effect during the probe-data qubit interaction, but even this is approximately captured by our model. To see this we need only consider X-type errors since Z errors have no immediate effect – they commute with the interaction S, see Eqn. 2.1. Then note that such an error gives rise to a superposition of two states equivalent to "flip suffered immediately before the interaction" and "flip suffered immediately after the interaction"; subsequent measurement of the probe collapses this superposition since the terms lead to different parity measurements.

2.2.2 Donor placement and systematic error

In addition to the 'usual' sources of random error, including our phase only two-qubit error 'jitter', it is a critically important point that we must suppress the systematic errors that arise from fixed imperfections in the locations of the spins. Each data qubit is permanently displaced from its ideal location by a certain distance in some specific direction, and these details may be unknown to us. Our analytic treatment (see Appendix 2.2.2) reveals that the general result is to weight certain elements of the parity projection irregularly. Specifically, whereas the ideal even parity projector is

$$\hat{P}_{\text{even}} = |0000\rangle\langle 0000| + |1100\rangle\langle 1100| + ... + |1111\rangle\langle 1111|$$

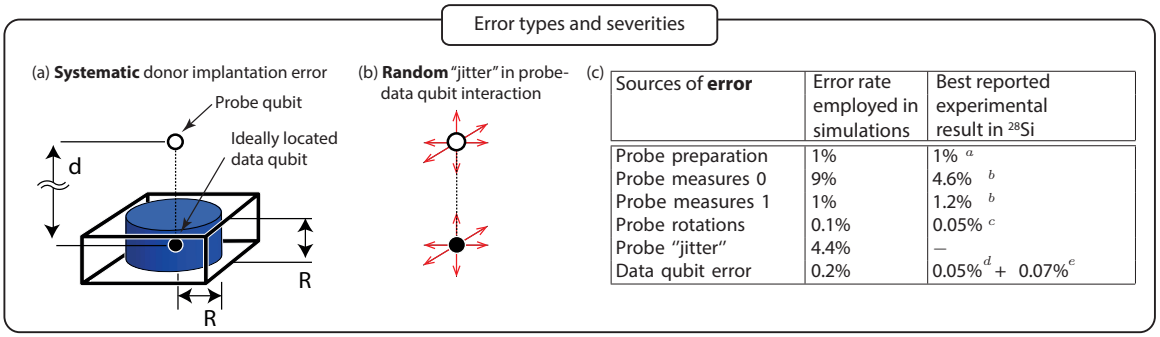
when the spins involved are misaligned then one finds that different terms in the projector acquire different weights, so that the projector has the form,

$$\begin{aligned}\hat{P}'_{\text{even}} = A\big(|0000\rangle\langle 0000| &+ |1111\rangle\langle 1111|\big) + \\ B\big(|0011\rangle\langle 0011| &+ |1100\rangle\langle 1100|\big) + \\ C\big(|0110\rangle\langle 0110| &+ |1001\rangle\langle 1001|\big) + \\ D\big(|0101\rangle\langle 0101| &+ |1010\rangle\langle 1010|\big) + \hat{W}\end{aligned}$$

where $\hat{W}$ is a set of lower weighted projectors on odd states. Meanwhile, the odd parity projector, $\hat{P}_{\text{odd}}$ becomes $\hat{P}'_{\text{odd}}$ which is similarly formed of a sum of pairs; each pair such as $(|0001\rangle\langle 0001| + |1110\rangle\langle 1110|)$ has its own weighting, differing from that of the other permutations. Where a qubit is too strongly coupled to the probe, by being positioned nearer the surface for example, a probe passing the data qubit will accumulate too much phase in the clockwise direction when passing over $|0\rangle$ or too much in the anticlockwise direction when passing $|1\rangle$, with an under accumulation

occurring if the qubits are implanted so their separation is greater than expected. Using these projectors to measure the stabilizers of the surface code presents the problem that the error is systematic: for a particular set of four spins, the constants A, B, C and D will be the same each time we measure an 'even' outcome. Each successive parity projection would enhance the asymmetry. In order to combat this effect, and effectively 'smooth out' the irregularities in the superoperator, we introduce a simple protocol that is analogous to the 'twirling' technique used in the literature on entanglement purification. Essentially we deliberately introduce some classical uncertainty into the process, as we now explain.

Suppose that one were to apply the imperfect $\hat{P}'_{\text{even}}$ projector to four data qubits, but immediately prior to the projection and immediately after it we flip two of the qubits. For example, we apply $XX\mathbb{1}\mathbb{1}$ before and after, where X is the Pauli x operator and $\mathbb{1}$ is the identity. Notice there would be no net effect if $\hat{P}'_{\text{even}}$ were the ideal parity projector. In practice the net effect would still be to introduce (unwanted) weightings corresponding to A, B, C and D, however these weights would be associated with different terms than for $\hat{P}'_{\text{even}}$ alone; for example the A weight will be associated with $|1100\rangle$ and $|0011\rangle$. Therefore, consider the following generalisation: we randomly select a set of unitary single qubit flips to apply both before and after the $\hat{P}'_{\text{even}}$ projector, from a list of four choices such as $U_1 = \mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1}$, $U_2 = \mathbb{1}\mathbb{1}XX$, $U_3 = \mathbb{1}X\mathbb{1}X$, $U_4 = \mathbb{1}XX\mathbb{1}$. That is, we choose to perform our parity projection as $U_i\hat{P}'_{\text{even}}U_i$ where i is chosen at random. We then note the parity outcome, 'odd' or 'even', and *forget* the i. The operators representing the net effect of this protocol, $\hat{P}^{\text{smooth}}_{\text{even}}$ and $\hat{P}^{\text{smooth}}_{\text{odd}}$ are specified in Appendix 2.2.2. Essentially we replace the weightings A, B, C and D with a common weight that is their average, but at the cost of introducing Pauli errors as well as retaining the problem that $\hat{P}^{\text{smooth}}_{\text{even}}$ has a finite probability of projecting onto the odd subspace. Analogously $\hat{P}^{\text{smooth}}_{\text{odd}}$ involves smoothed out odd projectors, newly introduced Pauli error terms, and a retained risk of projection onto the even subspace. However these imperfections are tolerable – indeed they will occur in any case once we allow for the possibility of imperfect preparation, rotation, and measurement. Crucially, the 'twirling' protocol allows us to describe the process in terms of a superoperator that we can classically simulate. It is formed from a probabilistically weighted sum of simple operators, each of which is either $\hat{P}_{\text{even}}$ or $\hat{P}_{\text{odd}}$, together with some set of single qubit Pauli operations, i.e. $S_1S_2S_3S_4$ where S_i belongs to the set $\{\mathbb{1}, X, Y, Z\}$. In our simulation we can keep track of the state of the many-qubit system by describing it as the initial state together with the accumulated Pauli errors.



Sources of **error**	Error rate employed in simulations	Best reported experimental result in ^{28}Si
Probe preparation	1%	1% [a]
Probe measures 0	9%	4.6% [b]
Probe measures 1	1%	1.2% [b]
Probe rotations	0.1%	0.05% [c]
Probe "jitter"	4.4%	–
Data qubit error	0.2%	0.05% [d] + 0.07% [e]

Figure 2.4: Our error model. (a) The imprecision in donor implantation within the silicon substrate is a systematic error; it is the same on each pass of the probe. We model this as each data qubit being randomly misplaced according to some distribution which will depend on the method used to manufacture the qubit arrays. Pictured here, a data qubit at a random fixed position with uniform probability inside the blue pillbox. (b) We also include a random fluctuation in the field strength of the dipole-dipole coupling, which we call "jitter". This would correspond to random spatial vibration of the probe qubit, or a random error in the timing of the orbit. This error occurs at each probe-data interaction independently. (c) Full table of the additional sources of error that are considered in our simulations and the experimental state-of-the-art for each in doped ^{28}Si. In each case note that we select fidelities for our simulations which are comparable with those which have been experimentally demonstrated. See Chapter 1.2.1 for more details of the error model. Note the experimental numbers for data qubit error are computed from reported rates applied over 1.2 ms, the cycle time given an abruptly moving probe and a 40 nm separation; achieving the same rates for the circular orbit would require improved materials and/or a smaller separation. References: [a][62], [b]Private communication with authors of[63], [c][64], [d][12], [e][65].

An equivalent effect to the U_i twirling operators can be achieved *without* actually applying operations to the data qubits. This is possible since flipping the probe spin before-and-after it passes over a given data qubit is equivalent to flipping that data qubit, i.e. it is only the question of whether there is a net flip between the probe and the data qubit which affects the acquired phase. Therefore we can replace the protocol above with one in which the probe is subjected to a series of flips as it circumnavigates its four data qubits, while those data qubits themselves are not subjected to any flips. Since we are free to choose the same $i = 1 \ldots 4$ for all parity measurements occurring at a given time, these probe-flipping operations can be global over the device. In this approach the only operations that target the data qubits are the Hadamard rotations at the end of each complete parity measurement. This is an appealing picture given that we wish to minimise noise on data qubits, and it is this variant of the protocol which we use in our numerical threshold-finding simulations, the results of which are shown in Fig. 2.5.

An extension relevant to many real systems would be to use a spin echo technique to prevent the probe and data qubits from interacting with environmental spins. In this case we would apply at least one flip to the spins (both the data and probe families simultaneously) during a parity measurement cycle; fortunately it is very natural to combine such echo flips with the flips required for twirling. The time for the parity measurement is thus not limited by the dephasing time T_2^*, but by the more generous coherence time T_2 [1].

2.2.3 Threshold results

In addition to these sources of error we investigate three particular models of the systematic misalignment of the data qubits versus the two different forms of probe orbit (circular versus abrupt, as discussed earlier). The resulting six variants are shown in Fig. 2.5. Our threshold-finding simulation generates a virtual device complete with a specific set of misalignments in the spin locations (according to one of the distributions), a specific probe orbit and the additional sources of error detailed in Fig. 2.4(c). Using this protocol, the simulation tests the virtual device to see whether it successfully protects a logical qubit for a given period, and then repeats this process over a large number of virtual devices generated with the same average severity

[1] Using similar techniques, it is also possible to decouple certain parts of the dipole-dipole interaction between the probe and the data qubits, which would reduce phase gate errors from probe qubits that are too strongly coupled to their data qubits. We note that the results detailed here do not make use of this performance improving technique.

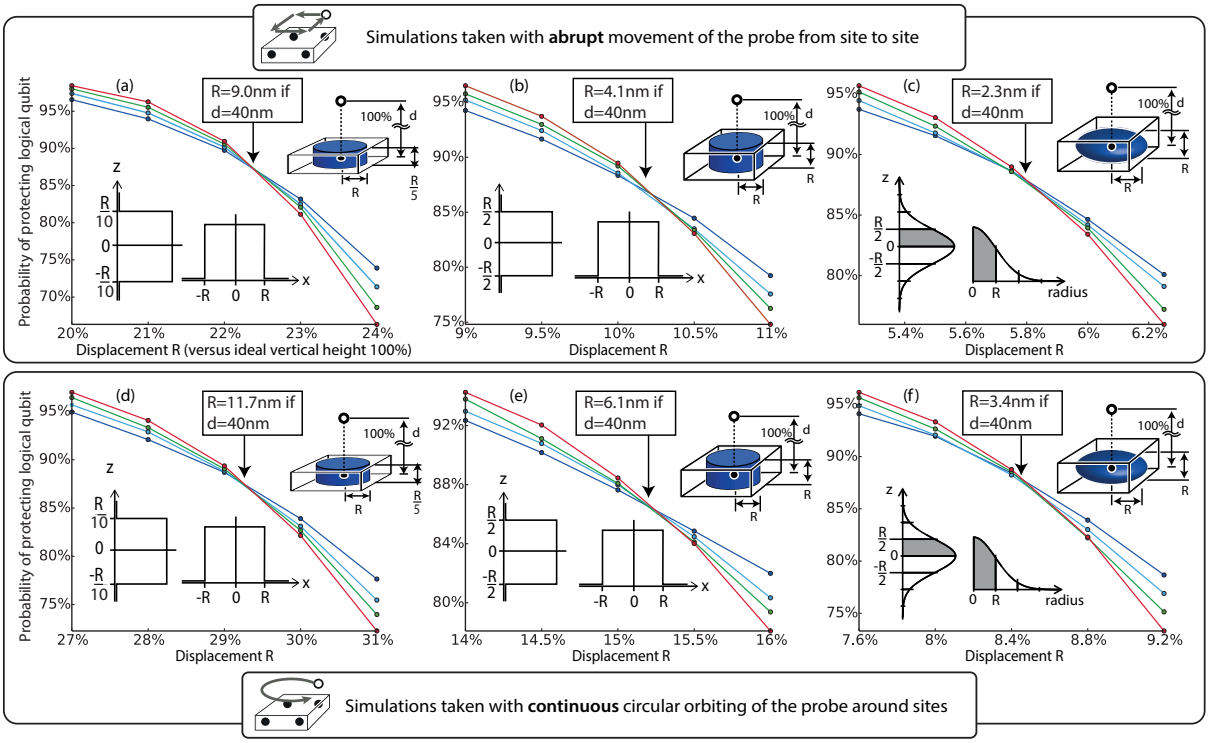


Figure 2.5: Results of threshold-finding numerical simulations. A system has surpassed the threshold for fault tolerant representation of a logical qubit if, when the system size is increased, we increase the probability of storing that logical qubit without corruption. Thus the crossing point of the lines reveals the threshold point. Lines from blue to red correspond to increasing array sizes of 221, 313, 421 and 545 physical qubits (corresponding to codes of distance 11, 13, 15 and 17). Figure 2.4(c) specifies the assumed error levels in preparation, control and measurement of the probe qubits. The data in (a) & (d) are for the disk-shaped distribution shown in the inset – data qubits are located with a circle of radius R in the x-y plane, and with a z displacement $\pm R/10$. The data in (b) & (e) are for a pillbox distribution, with a ratio of 2 : 1 between lateral and vertical displacement. In (c) & (f) the same 2 : 1 ratio is used, but with a normal distribution where R is now the standard deviation. Each data point in the figures corresponds to at least $50,000$ numerical experiments. Decoding is performed using the Blossom V implementation of Edmond's minimum weight perfect matching algorithm [66, 67].

of misalignments. Thus the simulation determines the probability that the logical qubit is indeed protected in these circumstances. By performing such an analysis for devices of different size (i.e. different numbers of data qubits) we determine whether this particular set of noise parameters is within the threshold for fault tolerance – if so, then larger devices will provide superior protection to logical qubits. Repeating this entire analysis for different noise parameters allows us to determine the threshold precisely. The results are shown in Fig. 2.5, and are derived from over six million individual numerical experiments.

The threshold results are shown in terms of qubit misplacement error as a percentage of the ideal probe-data separation d. The use of the long range dipole-dipole interaction means one can choose the scale of the device. On each plot we indicate the error tolerated for the specific case of $d = 40\,\text{nm}$. These show an extremely generous threshold in the deviation in the positioning of the implanted qubits, with displacements of up to 11.7 nm being tolerable in the best case scenario Fig. 2.5(d). Thus if donor qubits can be implanted with better accuracy than these values over a whole device, which otherwise operates with the errors detailed in Figure 2.4(c), we can arbitrarily suppress the logical qubit error by increasing the size of the qubit grid. We note that the continuous motion mode leads to a higher tolerance than the abrupt motion mode, as the smooth trajectory means a lower sensitivity to positional deviations in the x-y plane.

Concerning the three models of distributing the donors we note that the uniform disk-shaped distribution, where error in the z-position of the donor is five times smaller than the lateral positional error (Fig. 2.5 leftmost panels), may be the distribution expected of a more sophisticated fabrication technique involving precise placement of donor in a surface via an STM tip with layers of silicon then grown over. Meanwhile the pillbox and normal distributions (centre and right panels) might be more representative of the results of using an ion implantation technique with very low or very high levels of straggle respectively. These possibilities are discussed in more detail below.

2.2.4 Considering the 'dipolar background'

The simple of model of decoherence that we employ in our simulations can be improved to take in to account correlated errors arising from the magnetic dipole-dipole interactions of the data qubits with each other. These will lead to correlated pairs of errors occurring between the qubits, the probability of which decreases with the distance between them.

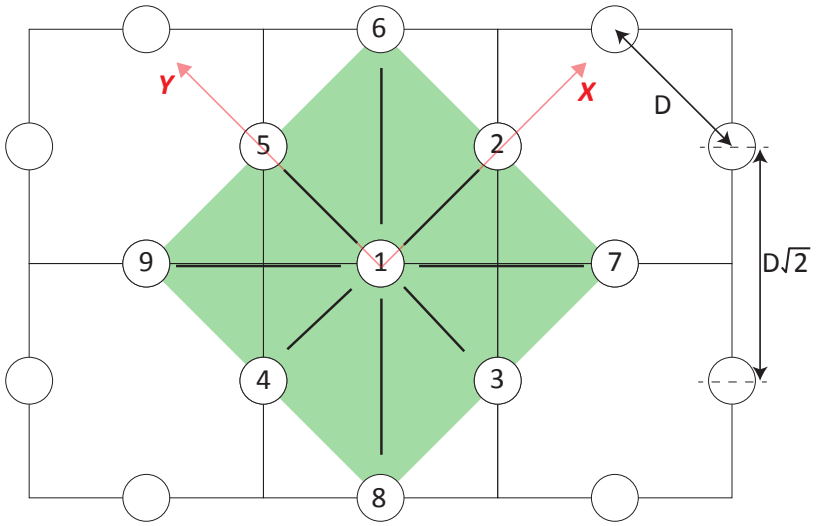


Figure 2.6: A portion of a larger surface code, with nearest and next-nearest interactions labelled as used in our simulated model of the 'dipolar background'. To a good approximation the evolution of the data qubits due to their mutual dipolar interactions is modelled by the interaction of nearest and next-nearest neighbour pairs. The distance between nearest neighbours is D, and the axes are chosen such that nearest neighbours lie either on the X- or Y-axis.

In this section we investigate the more nuanced model of decoherence that incorporates errors of this kind. It has already been shown in [68] that, despite the provable lack of threshold for these kind of errors in the surface code, practically speaking they are tolerable in that they can be suppressed to a desired degree. In fact that paper demonstrates that correlated pairs of errors whose probability decays with the *square* of the distance separating the two qubits are well-handled, this is an even longer range interaction than the one considered here.

In Fig. 2.7. we show threshold plots which demonstrate that the position of the threshold in terms of the qubit displacement varies slowly as we turn on correlated errors on nearest- and next-nearest-neighbours in the data qubit array. Here we set data qubit decoherence as modelled previously, that is single qubit Pauli errors, to occur with a small but non-zero level $p = 0.001$, so that the new correlated errors are the dominant effect for the data qubits. We leave other sources of error at the same level as in the other reported simulations. We then apply correlated errors according to the following procedure.

The Hamiltonians of the dipole-dipole interactions between the data qubits is:

$$H_{ij} = \frac{J}{r^3}\big(\boldsymbol{\sigma}_i \cdot \boldsymbol{\sigma}_j - 3(\hat{\boldsymbol{r}} \cdot \boldsymbol{\sigma}_i)(\hat{\boldsymbol{r}} \cdot \boldsymbol{\sigma}_j)\big).$$

We take an approximation that considers only one- and two-nearest neighbours, which comprise a data qubit and its nearest eight counterparts. Depending on the orientation of the pairs and distance separating them their evolution is governed by slightly different Hamiltonians. For example, the Hamiltonians describing a qubit 1 and its eight pairwise interactions with its nearest and next nearest neighbour neighbours are:

$$H_{12} = H_{14} = \frac{J}{D^3}\big(-2XX + YY + ZZ\big),$$

$$H_{13} = H_{15} = \frac{J}{D^3}\big(XX - 2YY + ZZ\big),$$

and

$$H_{16} = H_{17} = H_{18} = H_{19} = \frac{J}{2\sqrt{2}D^3}\big(-\frac{1}{2}XX - \frac{1}{2}YY + ZZ - \frac{3}{2}(XY + YX)\big),$$

where D is the distance between two nearest neighbour data qubits and the axes and qubit numberings are defined in Fig. 2.6.

The period of time for which this coupling runs can be estimated as the time to complete a round of stabilizers

$$t_{stabilizer} = \kappa\ 2\pi\frac{d^3}{J}$$

where $\kappa \in [2, \sim 80]$ is a parameter which reflects that in a slow orbit – such as the smooth circular motion, or a protocol which requires a larger number of orbits per round of stabilizers – there is a longer time between the completion of full stabilizer rounds. For example, an abrupt motion with 2 orbits required for a full round (i.e. one for X parity measurements and one for Z parity measurtments) corresponds to $\kappa = 2$ and slow circular motion with the same number of rounds corresponds to $\kappa = 20$.

The evolution of the data qubits and its neighbours in a short period of time according to such a Hamiltonian will have the form

$$U = \exp(-i\left(\sum_{\text{1NN,2NN}} H_{ij}\right)t)|\psi\rangle \approx \left(\prod_{\text{1NN,2NN}} \exp(-iH_{ij}t)\right)|\psi\rangle.$$

Taylor expanding and retaining the leading order terms, i.e. the errors on pairs of qubits, one obtains

$$|\psi(t_{stabilizer})\rangle \approx \left(|\psi\rangle + \kappa\ 2\pi\left(\frac{d}{D}\right)^3(aX_iX_j|\psi\rangle + bY_iY_j|\psi\rangle + cZ_iZ_j|\psi\rangle + \ ...)\right)$$

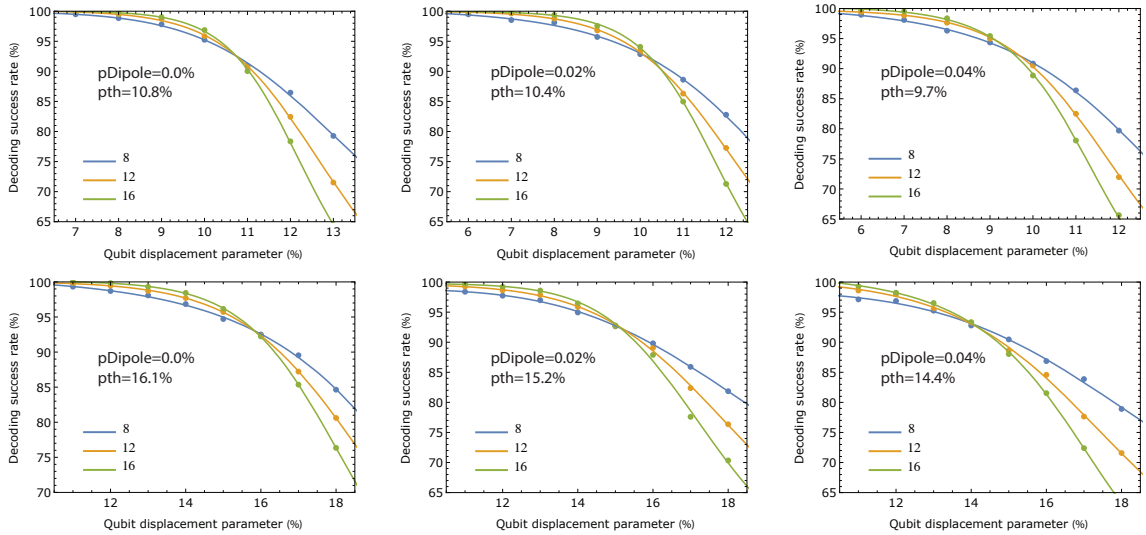


Figure 2.7: Variation in the qubit displacement threshold as the dipolar coupling of the data qubits is turned on. All data in this figure are for the case of 'pillbox' distributed qubit positional errors, as in Fig. 2.5(b) and (e). The upper three panels are for an abrupt probe orbit, the lower three for a smooth circular orbit. The other sources of error are fixed to the same values as in Fig. 2.4, with the exception of data qubit decoherence which we now set to 0.1 % to reflect than in earlier simulations we had modelled this being due in part to these dipolar couplings.

where $a, b, c...$ are determined by the relative positions of the data qubits with relation to one another.

At each stabilizer measurement the state of the qubit pairs will be projected into one of the states corresponding to either "no error" or one of the possible two-qubit errors with probability given by the square of its amplitude. We thus model the dipolar background in a similar way to our other sources of error: we now inject correlated two-qubit errors at the end of each round with the relative probabilities determined by the forms of H_{ij} above. The new parameter in our model is therefore $p_{dip} = (\kappa\ 2\pi)^2 \left(\frac{d}{D}\right)^6$.

We find that our threshold plots show the familiar behaviour, i.e a well defined crossing point, for the range of p_{dip} considered. To relate the values of of p_{dip} investigated in Fig. 2.7 to our proposed device dimensions, consider an abrupt orbit at the dimensions $d = 40\,\text{nm}$ and $D = 400\,\text{nm}$. Then $p_{dip} \sim 0.016\,\%$ which is comparable to the rates investigated in the central plots for which we see only small variation in the qubit displacement threshold. In the main text we suggest that for the slower circular orbits (i.e. increasing κ by a factor of 10) using the dimensions $d = 33\,\text{nm}$ and $D = 700\,\text{nm}$ which will achieve the same p_{dip} at the cost of a clock cycle approximately

10 times slower. We note that in both the circular and abrupt cases our simulations show that when setting $p_{dip} = 0.04\,\%$, larger than the values relevant to our proposed dimensions, the threshold in terms of qubit misplacement reduces by approximately one tenth, still allowing generous tolerance of the scheme to fabrication error.

The probability of a two-qubit error decreases with $\frac{1}{r^6}$ in this approximation, the results of [68] would suggest that practical surface code quantum computation is certainly possible with such a class of error. Although the dipole background can lead to correlated pairs errors occurring in distant parts of a logical qubit, as far as the code is concerned these just appear as two single qubit errors which it tries to correct. The free-running dipolar coupling of the data qubits does not cause the very damaging classes of error such as long chains or large areas suffering error, which could corrupt the logical qubit more easily [68].

2.2.5 Effect of other parameters on the threshold

The main result was to demonstrate a reasonable threshold value for the required donor implantation accuracy, as this is the crucial parameter for fabricating a large scale device. In doing so we fixed the errors associated with manipulations and measurement of the qubits to values currently achievable in the experimental state-of-the-art. The threshold is in fact a 'team effort': if we are able to reduce the measurement error, for example, then greater error in the other parameters can be tolerated. The 'thresholds' we determined are thus single points in a vast parameter space. Here we investigate further the effect that changing some of these parameters will have on the 'donor implantation error' threshold values determined in Fig. 2.5. Having made our code openly available we hope that interested researchers will find it easy to make further investigations based on their own favoured parameter regimes.

Fig. 2.8 shows how the threshold value for qubit misplacement depends on other key error parameters. In each graph one error parameter is varied while the others remain fixed to the values stated in Fig. 2.4(c). The simulations and the resulting fits show that data qubit decoherence is a key source of error to minimise in comparison with measurement and 'jitter' errors, at least in this region of the parameter space. Doubling the data qubit decoherence in this regime will have a more deleterious effect on the tolerance of implantation errors than a similar doubling of jitter or measurement error. However, the low rate of decoherence in donor qubits in silicon is one of the great strengths of the system.

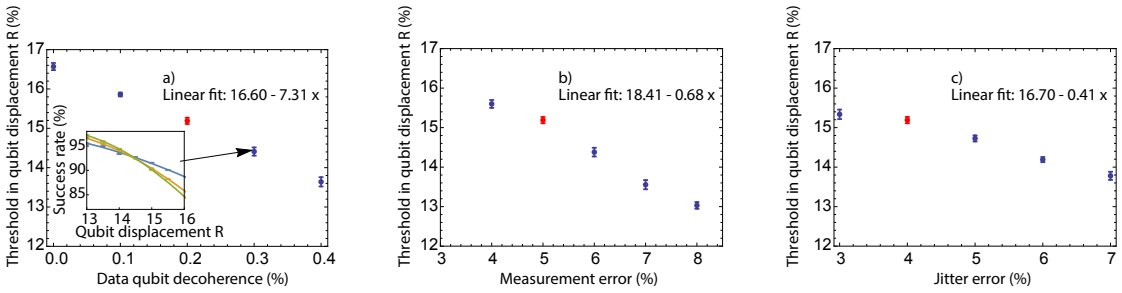


Figure 2.8: Variation in the qubit displacement threshold depending on a) data qubit decoherence; b) measurement fidelity and c) jitter error. Each data point corresponds to a a full threshold simulation as shown in the insert to a). All data in this figure are for the case of a "circular probe" orbit with "pillbox" distributed qubit positional errors. As such the red data points correspond to the threshold result shown in Fig. 2.5(e).

Interestingly, jitter errors (which corresponds to random fluctuations in the strength of the dipole interaction, which one might envisage being due to random timing error) seem to be even more well tolerated than measurement errors, which are generally seen as one of the less crucial sources of error for surface code thresholds. One might imagine that - in a future where donor placement is very far below threshold - characterisation of the device and individual control of probe orbits might mean that the fixed misplacement errors can become random errors in how well we calibrate our device to the misalignments. This would make 'jitter' the main source of error, as such it is encouraging to see it is well tolerated.

2.3 Generalisation to quantum computation.

The only fundamentally new element required to create a machine that performs universal computing in addition to the surface code parity measurements the creation of magic states [41]. In later chapters we will explore this issue in some detail. For our purposes there are two issues to confirm: Can we make such states within our architecture, and, does the need to do so revise our threshold(s)? Magic state generation involves injection (mapping a single physical qubit to an encoded qubit) and distillation (improving the fidelity of such encoded states by sacrificially measuring some out). The latter involves only Clifford operations (which may include a previously distilled S gate), and therefore falls under the discussion in our previous paragraph. Injection requires operations on individual data qubits rather than the groups of four, but this is possible within our constraint of sending only global pulses

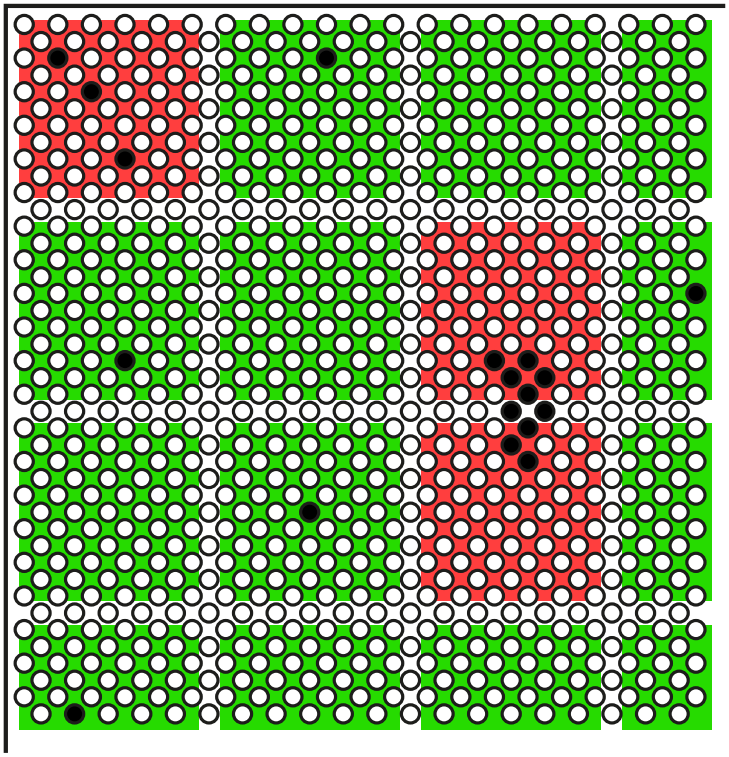


Figure 2.9: Example of how one could encode and process multiple logical qubits into a flawed array. Each white circle is a data qubit; each green patch is a subarray representing a single logical qubit. When we wish to perform a gate operation between two logical qubits then we begin making parity measurements at their mutual boundary, according to the lattice surgery approach [69]. If a region of the overall array contains multiple damaged or missing data qubits, we simply opt not to use it (red patches). Note that in a real device each patch would be several times larger in order to achieve high levels of error suppression. [Figure credit: Simon Benjamin].

to our data qubits: control of individual probe spins implies the ability to control individual data qubits and indeed to inject a magic state. For example, suppose we set probes to $|1\rangle$ where we wish to have a net effect on the adjacent data qubits, and $|0\rangle$ elsewhere. Consider the sequence $Y(-\pi/2)Z(-\pi/8)S(\pi/8)Y(\pi/2)$, where Y and Z are Pauli rotations both performed on all data qubits and S is the probe-data qubit interaction, Eqn. 2.1. The net effect on a data qubit is the identity when the probe is in $|0\rangle$, but when the probe is $|1\rangle$ it constitutes a rotation of $X(\pi/4)$ on the data qubit, where is X is the Pauli rotation. This operation would take a data qubit from $|0\rangle$ to a magic state $\cos(\pi/8)|0\rangle - i\sin(\pi/8)|1\rangle$ if-and-only-if the proximal probe is in state $|1\rangle$. Crucially, the purification threshold for a noisy magic state is much larger than that for a surface code memory. Ultimately, therefore, the overall threshold for quantum computation is indeed set by the memory threshold, as discussed by the relatively early literature [70]. More on this later.

More generally, one can ask about how multiple qubits should be encoded into a large array of data qubits, and what the impact of flaws such as missing data qubits would be on a computation. While we do not undertake a detailed analysis, approaches such as lattice surgery [69] can offer one simple solution that is manifestly tolerant of a finite density of flaws. The approach is illustrated in Fig. 2.9. Square patches of the overall array are assigned to hold specific logical qubits; stabilizers are not enforced (i.e. parity measurements are not made) along the boundaries except when we wish to perform an operation between adjacent logical qubits. Importantly, if a given patch is seriously flawed (because of multiple missing data qubits during device synthesis, or for other reasons) then we can simply opt not to use it – it becomes analogous to a 'dead pixel' in a screen or CCD. As long as such dead pixels are sufficiently sparse, then we will always be able to route information flow around them.

2.4 Feasibility

We now discuss the practicality of the proposed device, in light of the simulation results. A more detailed version of this section can be found in Ref. [54].

2.4.1 Timescales and decoherence.

For a probe - data qubit separation of $d = 40\,\text{nm}$, the total interaction time for the four $S(\pi/2)$ phase gates with the four data qubits is $t_{\text{int}} = 2\pi d^3/J \approx 1.2\,\text{ms}$. This stabilizer cycle time is short enough to comply with the coherence times of donors in

silicon or the NV centre and long enough to avoid a significant operational lag due to the finite operation speed of the stage. In the abrupt motion scenario Fig. 2.2(b)(i) with negligible transfer times, this would allow an operation of the device at about 1 kHz. In the continuous circular motion picture, a significant time is required for the probe's transfer between data qubits which slows down the device by roughly a factor of $D/d = 10$. As noted earlier the consequent increased accumulation of unwanted in-plane spin-spin interactions can be negated by varying the dimensions, for example to $d = 33\,\text{nm}$, $D = 700\,\text{nm}$. We further note that – due to the $1/d^3$ dependence of the dipolar interaction – every reduction of d by a factor of two allows eight times faster operation frequencies. So, if manufacturing precision improves, this device should be readily scalable to faster operations. In practice selecting dimensions d and D will be a trade off between the fabrication capabilities, the achievable translation velocities and the decoherence time of the qubits.

2.4.2 Mechanics and device design.

The tip of an atomic force microscope cantilever can achieve mechanical motion with sub-nanometer positional accuracy and thus could serve as a prototype device for implementing a single parity measurement. In principle, an array of tips on a single cantilever could incorporate the probe qubits and a cyclic motion of the cantilever would allow the four qubit phase gates.

A more viable mechanical system could instead be x-y translation stages realised by micro electromechanical systems (MEMS). Various designs for MEMS x-y translation stages have been put forward with travel ranges in the 10µm range or higher [71–73] and positional accuracies in the nm regime [74, 75], both meeting essential requirements of our proposal. Designs with frequencies $> 10\,\text{kHz}$ [76] would permit stabilizer cycle translation times on the order of $\sim 100\,\mu\text{s}$.

Since the probe qubits (but not the data qubits) need to be individually controlled and measured, local electric gates are required. There are two basic strategies depending on whether it is the probe array or the data qubit array that is in motion. If the probe stage is mobile, then it is necessary to deliver the electrical contacts over the suspension beams at the side of the probe stage. Wide beams would result in a large in-plane spring constant for the stage motion, so we suggest using many thin beams. The spring constant of the beams for the in-plane motion of the stage scales with the width cubed. N thin beams of width w thus only increase the spring constant by a factor of Nw^3 compared to $(Nw)^3$ in the single wide beam case. If however the data qubit grid forms the movable stage, then there is no need for such bridging since the

data qubits are controlled purely through global pulses. The control for the probe qubits in the static silicon stage below could then be written in the same process as the probe spins themselves, relying on atomic-precise fabrication of phosphorous impurity SETs and gates (see below, [58, 77]).

2.4.3 Material systems.

Finally we direct our discussion to the properties of the proposed solid state qubit systems for this orbital probe architecture, namely donor impurities in silicon and the NV centres in diamond.

To achieve the aforementioned individual addressability of the probe qubits, either probe spins could be individually Stark shifted into resonance with a globally applied microwave source [78, 79], or, alternatively, magnetic field gradients could be applied to detune the individual resonance frequencies within the probe qubit grid. Most of these systems also exhibit hyperfine structure, meaning that the transition energy between the $|0\rangle$ and $|1\rangle$ state of the electron spin depends on the nuclear spin state. This effectively results in two or more possible 'species' of qubits, each of which must be manipulated by a microwave pulse of a different frequency. To perform the required qubit operations regardless of the nuclear spin state, we propose to use multi-tone microwave pulses composed of all resonance frequencies of the different species. With this we can ensure that nuclear spin flips - so long as they occur less frequently than the time for a stabilizer cycle - do not affect our ability to implement the proposed protocol.

Donors in silicon Silicon can be isotopically purified to a high degree, which reduces the concentration of ^{29}Si nuclear spins and creates an almost ideal, spin-free host system. Consequently, electron spins of donor impurities, such as phosphorus, show extraordinarily long coherence times of up to 2 s [12], thus enabling a very low data qubit memory error probability (sub 0.1%) over the timescale of a single parity measurement of 1.2 ms. If donors in silicon are employed as the probe qubits, then initialisation and read-out of the electron spin of single dopants can be achieved using SETs [62, 63, 80] with average measurement fidelity of 97 % with read-out timescales on the order of milliseconds. Read-out fidelity can be further increased if the electron spin state is transferred to the nuclear spin [81] from which it can then be measured with higher fidelity up to values of 99.99 % [63]. However this is a much slower process with measurement times of order 100 ms which are two orders of magnitude slower than other timescales described in this proposal. The single qubit control fidelity

for an electron spin of a single P donor in ^{28}Si in these devices has been reported as 99.95 % [64]. Furthermore, it was shown in [63] that the decoherence time of the qubit is not significantly affected by its proximity to the interface and can reach values up to 0.56 s with dynamical decoupling sequences.

The footprint of the required electronic components to measure a single donor spin in silicon is typically on the order of $200 \times 200\,\mathrm{nm}^2$ and is thus small enough to achieve qubit grid separations of $D = 400\,\mathrm{nm}$.

As shown by our threshold calculations, a key figure for this scheme is the implantation accuracy required for the probe and data qubit arrays. Ion implantation methods with resolutions approaching 10 nm can be achieved using either e-beam lithography directly on the substrate [82, 83] or nanostencil masks drilled into AFM cantilevers [84]. For donors deterministic single qubit implantation is also possible [85]. Another technique for silicon is the STM tip patterning of a hydrogen mask and the subsequent exposure to phosphene gas, which enables atomically-precise (± 3.8 Å) phosphorus donor incorporation in all three dimensions [57, 58]. This accuracy is more than an order of magnitude below our calculated thresholds of Fig. 2.5 and the challenge remaining is to maintain this precision over larger qubit arrays.

Diamond nitrogen-vacancy centres The electron spin qubit associated with the nitrogen-vacancy (NV) defect centre of diamond features optically addressable spin states, which could be manipulated even at room temperature. By using resonant laser excitation and detection of luminescence photons, fast ($\sim$40 µs [86]) and reliable (measurement fidelity of 96.3 % [87]) read-out of single NV centre spins could be employed for the probe spin measurement and initialisation. The nuclear spin of ^{14}N or of adjacent ^{13}C may again be exploited to enhance the measurement fidelity (99.6 % [88]). The coherence times in isotopically purified diamond samples ($T_2 = 600\,\mathrm{ms}$ at 77 K using strong dynamical decoupling [89]) are long enough to allow millisecond long stabilizer cycles and individual probe control using a global microwave field is possible using electric gates and the Stark effect [90], similarly to donors in silicon.

While the qubit operations possible in NV centres are advanced, so far very few micro-electromechanical devices have been realised using diamond. In principle though, diamond possesses promising material properties for MEMS applications [91].

The implantation accuracy for the NV centre is determined by ion beam techniques. The most accurate method known to the author achieves lateral accuracies of $\sim 25\,\mathrm{nm}$ at implantation depths of $8 \pm 3\,\mathrm{nm}$ [92]. This precision is slightly below

threshold of our scheme and it is reasonable to hope that new implantation methods could meet the requirements in the near future. Furthermore, we note that the proposed grid spacing of $D = 400\,\text{nm}$ is greater than the diffraction limit for optical read-out (250 nm [93]). Recent advances in 'laser writing' NV centres [94] promise a method that will lead to less damage the lattice and potentially writing of large scale 2D arrays in diamond. At this early stage however, implantation accuracy and yield do not yet meet the requirement of our threshold simulations.

In fact the low yield of active NV centres per implanted nitrogen atom, which is typically well below 30 % [95] is a barrier to scalability. Such a low yield would result in too great a number of 'dead pixels' within the layout specified in Fig. 2.9 to allow for the construction of a useful device at this time.

While there are still significant challenges remaining to an integrated diamond MEMs probe array, it is encouraging that the basic requirement of our proposed scheme, i.e. the control of the dipolar interaction of two electron spins by means of changing their separation mechanically, has already been achieved. Grinolds *et al.* were able to sense the position and the dipolar field of a single NV centre by scanning a second NV centre in a diamond pillar attached to an AFM cantilever across it — at a NV centre separation of 50 nm [96].

2.5 Proposed all electronic architecture

This section contains a preliminary investigation into a variant of the 'orbital probe architecture' to one which does not require the use of micro-mechanical devices to move the probe qubits in and out of proximity with the data qubits. Instead we propose a system of probe electron spins defined in an array of lateral electrode-defined quantum dots, which are shuttled from site to site by control of electrode voltages, allowing them to interact with the donor-based data qubits.

Two regular arrays of qubits are fashioned as shown in Figure 2.10. The data qubits are a regular array of phosphorus donors in silicon. The spin state of electron encodes the qubit. These do not have to be individually addressed. The probe qubits are a regular array of quantum dots positioned vertically above each donor. Single electrons, or single-triplet qubits, are thus defined and moved ballistically within a structured 2DEG at the interface of, for example Si/SiGe. The parity measurements of the surface code proceed as follows: A probe qubit is prepared in the $|+\rangle$ state in the 2DEG. It is then moved around a square, stopping at four dots to interact with

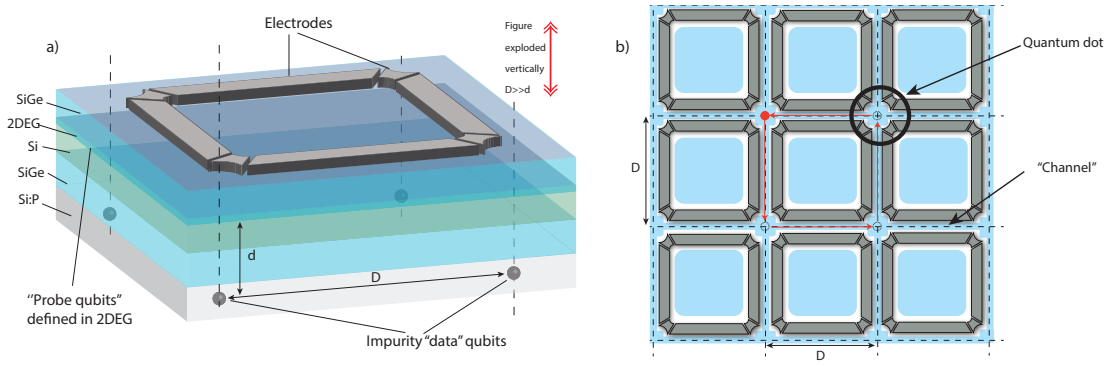


Figure 2.10: The architecture. (a) The unit cell of the device. A 2DEG is formed at the interface of SiGe/Si or some other materials. The probe qubit is an electron controlled in this 2DEG by the electrodes above. Donors in silicon are positioned below. A buffer layer prevents the donor wavefunction leaking into 2DEG. A single electron visits a site above each donor qubit in turn and acquires phase via dipole-dipole interactions. The probe qubit is then measured. (b) The layout of a large scale device. Blue shaded regions indicate areas in which electrons can exist in 2DEG layer. In red, the path of a single probe performing a parity measurement is shown. This occurs in parallel in each cell across the device. Measurement of each probe electron must occur after every cycle of four interactions (not pictured).

the donor qubits below. After the four interactions, the spin state of the probe qubit will need to be measured. The process is continuously repeated.

2.5.1 Thresholds

As before we simulate virtual devices of the kind described in Figure 2.10 including the various potential sources of error previously described for the mechanical orbital probe scheme. Here however, due to the differing dimensions of the device we focus on two different parameters and their effect on the threshold: those being (a) probe dephasing is the dominant error or (b) timing errors in the orbit dominate. More details of our simulation can be found in Figure 2.11. Our previous work demonstrated the robustness of the mechanical orbital probe scheme. The differences between these results and those in 2.2.3 are as follows. It is assumed that all the operations, such as single-qubit Paulis, are performed far better than threshold values that we previously determined: we set them to be an order of magnitude better than the values used in the previous analysis. The exceptions are then the parameters of interest here, namely the dephasing of the probe and 'jitter' errors.

Another difference is that the simulations shown here take into account the non-point-like nature of the qubits, which becomes more relevant as we decrease the

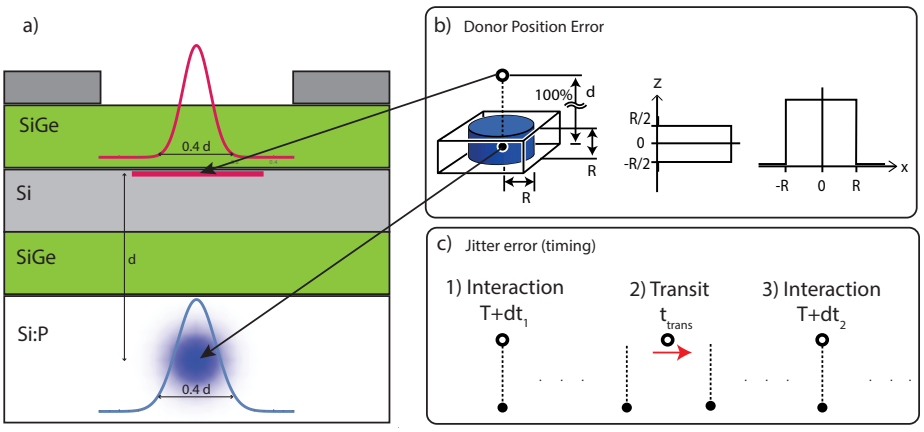


Figure 2.11: Errors and length scales in simulation (a) The wavefunction profiles (Gaussians) used for simulation with probe-data separation d set to unity w.l.o.g.. 2D wave function with circular symmetry for the probe (red). 3D wave function with spherical symmetry for the donor qubit (blue). (b) The donor position error profile used. Each donor in the simulated lattices is randomly positioned within a pillbox of radius and height R. (c) One interpretation of jitter error. Each probe-data interaction is scheduled to last for time T to allow $\pi/2$ phase to be acquired by the probe, at each interaction this time T can err. In our simulations the transit time is taken to 0, so the probe instantaneously jumps from dot to dot.

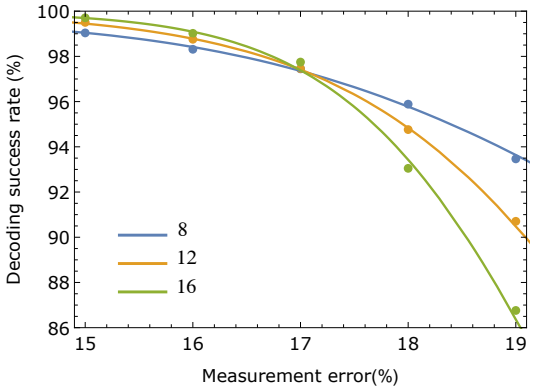
distance between probe and data qubits. Figure 2.11 contains more information about our simulations. We have investigated two 'limiting' cases, very bad probe dephasing and very bad jitter error.

Probe dephasing

The surface codes tend to be very tolerant to 'measurement error', i.e. with some probability our probe measurement outcome indicates $|0\rangle$ when the state is in fact $|1\rangle$. Essentially to combat measurement error, repeated measurement is required to obtain confidence in the result of those measurements. In the limit then one only needs to be $(50+\epsilon)\%$ confident of a making a measurement correctly to infer a true result, albeit after many repeated measurements [32]. It can be shown that dephasing of the probe maps to a final measurement error. This perhaps indicates that the T_2 of the probe qubits need not be very long to realise the architecture. We performed simulations to investigate this.

Following the standard definition of the T_2 time, the mapping to measurement error p_{meas} is (assuming that the source of measurement error is dephasing of the probe qubit),

$$\frac{T_2}{\tau} = \frac{-1}{2\ln(1-2p_{\text{meas}})}, \tag{2.3}$$



Key operations	Simulated infidelity
Donor position error R/d	5%
Probe preparation	0.1%
Probe measurement	–
Jitter	0.4%
Probe rotations	0.01%
Data qubit error	0.02%

Figure 2.12: Measurement error threshold. Each line represents a virtual device of a different size. Below threshold larger devices protect information better, above threshold smaller devices are superior. Thus the threshold is the crossing point of these lines. The threshold for measurement error is 16.9%, when other errors are fixed to the levels indicated in the adjacent table. This would correspond to a minimum T_2 of 1.2τ, which for a probe-data separation of 10 nm corresponds to $\sim 86\mu s$. Here we fix donor position error to 5% of probe-donor separation.

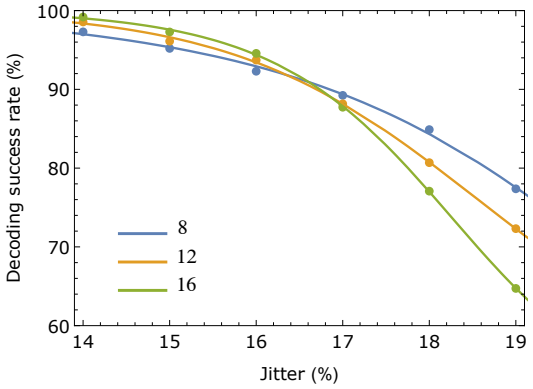
where τ is the time taken for the process of one stabilizer measurement, i.e. $4t_{transit} + 4t_{interaction} + t_{meas}$. We can therefore determine a minimum phase coherence required of our probe qubit that would allow this scheme to operate below threshold.

We determine the threshold of the device in terms of the T_2 by numerical simulation of virtual devices. We find that the threshold in measurement error is 16.9%, see Figure 2.12. This corresponds to a minimum T_2 time for the probe qubit of 1.2τ; which at a probe-data separation D of 10 nm corresponds to roughly $1.2(4*(18\ \mu s)) \approx 86\ \mu s$. Reducing the probe-data separation D to 5 nm would reduce the required T_2 by roughly a factor of 8.

These numbers assume $t_{transit}, t_{meas} \to 0$. Finite values for these will will increase τ. Evidently the faster a stabilizer cycle can be completed then the less the required T_2 of the probe. This would seem promising in light of this result [97] which demonstrated a T_2 on the order hundreds of milliseconds. For a device to be practical it would have to operate at 3-5 times below threshold, i.e with $T_2 \sim 5\tau$. A good T_2 will relax the requirements on the other sources of error.

"Jitter"/Timing errors

The parameter "jitter" is essentially a random variation in the interaction strength at each of the four probe-data qubit interactions. This could be from various sources, for example random vibration in the position of the qubits. An error source that



Key operations	Simulated infidelity
Donor position error R/d	5%
Probe preparation	0.1%
Probe measurement	0.5%
Jitter	–
Probe rotations	0.01%
Data qubit error	0.02%

Figure 2.13: Jitter threshold. The threshold value for jitter, here given explicitly by the crossing, is 16.4%. The values for the other forms of error are indicated in the adjacent table. Note here we take measurement error/probe T_2 to be very small. As such this plot corresponds to a variation on the above device in which the probe is likely not the 'active' or 'mobile' element.

closely corresponds to our simulation is random error in the interaction time between the qubits, i.e. the time the probe spin is held at each dot. We find that the tolerable threshold of this error is high for our chosen values of other error parameters, see Figure 2.13, a percentage timing error of $\sim 16.4\%$. This may be a significant error in some possible implementations of the scheme which rely on more complex motion of the probes or donor qubits.

2.6 Conclusion

This chapter has sought to demonstrate that if the current capabilities of donor-based qubits could be replicated at large scale, then a surface code quantum computer could be constructed that would be tolerant to fabrication error and achieve very high qubit densities. Silicon is a highly promising substrate, not just for the 'spin vacuum' properties of isotopically purified silicon which allow such impressive coherence times, but also given the large numbers of qubits required to fault-tolerantly implement a post-classical computation (as we investigate in Chapter 5) working within a system that could make use of an industry and range of techniques that have demonstrated huge scalability in classical computing should prove hugely beneficial. Particularly given that we demonstrate the incredibly challenging exchange gates which have been the goal of much research into donor-based qubits are not necessarily a prerequisite for universal quantum computation with those qubits.

Having thought about a blueprint for long-term quantum information processing, we now consider a framework to help give direction and motivation to present-day device experiments containing on the order of 10 qubits.

Chapter 3

Small Codes

Chapter 3 presents a method for assessing the performance of error correcting codes in small quantum systems. In Section 1 the concept of 'integrity' is introduced. In Section 2 four milestones for experimentalists are described. In Section 3 we present simulations of three small codes, the five qubit, seven qubit (Steane) and nine qubit (surface) codes, providing a comparison of their characteristics as well as demonstrating some properties of integrity as a metric. Section 4 compares the performance of our simulated 'Bob' to the trace distance. Section 5 explores the case where a more complex but fault-tolerant circuit is superior to a simpler, non-fault-tolerant, method of syndrome extraction. The work in this chapter is based on the paper [98]. The author contributed to the conception and simulations of the protocols, writing code to confirm and verify results generated for the published paper by Xiaosi Xu (XX). Where indicated the results were generated by XX, otherwise simulation results were generated with the author's independent code. Note that the text of this chapter is adapted from the text the author co-wrote for Ref. [98]

The previous chapter presented the results of a theoretical analysis of a large scale computer architecture that may take years or perhaps a decade to realise experimentally. In this chapter the topic instead concerns theory aimed at supporting efforts occurring now, and in the immediate future, to demonstrate the first real encoded logical qubits. The scale of the devices is therefore more modest – fewer than twenty physical qubits, and often fewer than ten – and the concept of a fault tolerance threshold does not necessarily translate directly to such systems. There is a need for a measure of success that is relevant to systems of only one, or at most two, logical qubits and such a measure should be experimentally attainable while having a suitable degree of rigour.

Recently there has been rapid progress in the implementation of quantum codes, across platforms as diverse as ion traps [99–101], superconducting qubits [102–105], and crystal defect systems [106]. There is a need to benchmark the achievements in these diverse systems, and to compare the inherent power of the codes they rely upon. To this end, this chapter will introduce a performance measure called *integrity*, which relates to the probability that an ideal agent will successfully 'guess' the state of a logical qubit after a period of storage in the memory. Integrity is straightforward to evaluate experimentally without state tomography and it can be related to various established metrics such as the logical fidelity and the pseudo-threshold.

This chapter will introduce the notion of integrity through its intuitive meaning as "the probability that Bob, receiving a logical qubit from the memory system, can infer its state". In simple cases integrity also corresponds to "the fidelity of a logical qubit after storage in the memory", but that the former meaning based on state inference remains meaningful even when the latter notion of a memory's fidelity becomes ill defined. This chapter will set out four milestones based on comparing the integrity of an encoded and actively corrected quantum memory versus either un-corrected variant or with a single physical qubit. The milestones are increasingly challenging with the fourth being a demonstration of 'Strictly superior encoded memory' . Memories based on the five-qubit code, the seven-qubit Steane code, and the nine-qubit small surface code are assessed through intensive numerical simulations. Estimates are derived for the performance levels required in the error correcting process (performed by an agent we label 'Igor') so that our milestones can be met.

A comprehensively successful quantum code will have been achieved when one can demonstrate a full set of quantum operations on encoded qubits with a fidelity that exceeds that of the best possible unencoded physical qubits [107]. However this criterion is very challenging to achieve; it means 'beating' the superb fidelities exceeding 99.9% that can now be achieved with single physical qubits [16, 27] – in fact 99.9999% can be achieved in a system dedicated specifically to single qubit gates [108]. Even the task of achieving a superior coherence time with a memory based on an encoded qubit, versus a single physical qubit, is not trivial. Individually controlled physical qubits can persist for the order of a minute when not actively manipulated [108], or 10 minutes using dynamical decoupling [8].

3.1 Introducting integrity

A memory is a channel for communicating information from the present ($t = 0$) to a specified future time ($t = \tau$). The simplest notion of an ideal memory would be one where no change whatsoever happens to the stored information. It is useful to begin by asking how we would benchmark performance against this basic standard: we could compare the state at $t = 0$ with the state at $t = \tau$, using either the fidelity or the trace distance. Let us briefly review these two quantities.

In this work we will take the definition of fidelity to be

$$\mathcal{F}(\rho_0, \rho_1) = \left\| \sqrt{\rho_0}\sqrt{\rho_1} \right\|_{\text{tr}}^2. \tag{3.1}$$

This definition uses the trace norm, itself defined as

$$\|\sigma\|_{\text{tr}} = \text{Tr}\left(\sqrt{\sigma^\dagger \sigma}\right). \tag{3.2}$$

In the case that ρ_0 is a pure state $|\psi_0\rangle\langle\psi_0|$, the fidelity then has a simple physical interpretation: if we measure state ρ_1 in a basis where one of the possible outcomes is $|\psi_0\rangle$, the fidelity is precisely the probability of this outcome. When both states are pure, we have simply

$$\mathcal{F}(\psi_0, \psi_1) = |\langle\psi_0|\psi_1\rangle|^2.$$

While the fidelity measures the similarity of two states, the trace distance measures the degree to which two states differ. It is defined as

$$\mathcal{D}(\rho_0, \rho_1) = \tfrac{1}{2}\left\|\rho_1 - \rho_0\right\|_{\text{tr}}. \tag{3.3}$$

Ranging from 0 to 1, the trace distance has a clear intuitive meaning: it tells us the probability that two states ρ_0 and ρ_1 could be told apart by an ideal experimentalist. Suppose that we present to an experimentalist, Bob, a theoretical description of both ρ_0 and ρ_1, and we also prepare a physical system in one of these two states (with a 50/50 prior probability) and present this to the Bob. He must guess whether the physical state is ρ_0 or ρ_1. Using his optimal strategy, his probability p_g of guessing correctly is simply

$$p_g = \tfrac{1}{2} + \tfrac{1}{2}\mathcal{D}(\rho_0, \rho_1). \tag{3.4}$$

We will make extensive use of this idea presently.

The functions $\mathcal{D}(\rho_0, \rho_1)$ and $1 - \mathcal{F}(\rho_0, \rho_1)$ can both be regarded as measures of how distinct two states are. However it is important to note that these quantities are fundamentally different, and can give very different 'scores' in experimentally

relevant cases. We opt to employ the trace distance, for various reasons described later but most particularly because Eqn. (3.4) leads to straightforward experimental realisations.

The idea of a perfect memory being one that simply allows no change to occur does not take into account some key facts. Certain changes are in fact harmless and do not practically reduce the quality of a memory. Any deterministic, known and anticipated change to the stored information is harmless if we can easily compensate: an example is the continuous phase evolution which occurs within any physical qubit if the states $|0\rangle$ and $|1\rangle$ are non-degenerate eigenstates. In fact for a memory that employs a quantum code to protect logical qubit(s) we can go further: Any *correctable* error is also relatively harmless in the sense that a ideal agent can recover the logical qubit with certainty. We would like our measure of the quality of a memory to incorporate these principles; additionally, we have a notion of a 'useless' memory, one that should score zero, as a memory that fails to preserve any recoverable information whatsoever.

Suppose that some qubit with density matrix ρ is to be stored in a code-based memory channel for a specified period of time. We will use the symbol Φ to denote the memory channel itself. The initial state ρ maps to the final state $\tilde{\rho}$ through this process:

1 At $t = 0$ Alice (taken to be perfect) encodes the single qubit ρ into an n-qubit logical code: $\rho_n = \mathsf{E}(\rho)$ where E is the encoding map.

2 Evolution and degradation of the logical qubit occurs while it is stored. This may include the effects of actively applied error correction (if so, we label the non-ideal agent responsible 'Igor'). We have $\rho'_n = \mathsf{N}(\rho_n)$ where N is the noise map.

3 At $t = \tau$, Bob (like Alice, taken to be perfect) performs an error correction cycle, and then reverses Alice's encoding process to obtain a single physical qubit: $\tilde{\rho} = \mathsf{D}(\rho'_n)$ where D is the decoding map.

It is the second step that we are interested in; steps one and three (Alice and Bob) merely frame the process. We can write the entire channel as $\Phi = \mathsf{D} \circ \mathsf{N} \circ \mathsf{E}$, thus incorporating Alice's encoding E, the noise N, and Bob's decoding D. This overall map Φ takes as input a single qubit state (Alice's initial choice ρ) and ultimately returns another single qubit state $\tilde{\rho} = \Phi(\rho)$, i.e. Bob's single qubit after decoding.

For our naive concept of an ideal memory as one causing no change at all, we would desire $\Phi(\rho) = \rho$, and so (for example) $1 - \mathcal{D}(\rho, \Phi(\rho))$ could suffice as a good metric for the performance of our memory. However we have noted that a much broader notion of 'ideal' is needed. Fortunately, there is a natural way to proceed: instead of focusing on the changes suffered by a single logical qubit between $t = 0$ and $t = \tau$, we can instead focus on the idea that a memory should preserve the *distinguishability* of different states. This notion can incorporate both fixed, known evolutions and random-but-correctable errors. Conversely it will properly recognise that all forms of memory which leave us with no recoverable information, are equally and entirely useless.

Consider two pure states $\psi = |\psi\rangle\langle\psi|$ and $\psi_\perp = |\psi_\perp\rangle\langle\psi_\perp|$ which are orthogonal to one another. Orthogonal states have trace distance of unity, since they can certainly be told apart. Let Alice choose ψ at random, uniformly from all possible single-qubit states, and then opt to encode *either* ψ or instead the antipodal state $\psi_\perp$. Then $\Phi(\psi)$ or $\Phi(\psi_\perp)$ will describe the state after it has passed through the memory channel and been decoded by Bob. If the channel has caused the same fixed evolution to occur to each (logical) state, or indeed if it has introduced errors but they are correctable, then these states will still be completely distinguishable – they will still have trace distance equal to unity. Therefore we define the integrity of the memory as

$$\mathcal{R}(\Phi) = \min_{\psi} \mathcal{D}\big(\Phi(\psi), \Phi(\psi_\perp)\big). \tag{3.5}$$

Note that we take the minimum over all possible choices of ψ made by Alice. We do this to account for the fact that certain memory channels may have no detrimental effort on special choices of the state, as for example a dephasing channel leaves $|0\rangle$ and $|1\rangle$ unchanged. To avoid falsely overrating the quality of such a channel we impose the minimum by finding the worst case. Note that for many environmental noise models, including pure depolarising noise, Bob's performance does not vary with Alice's choice.

In the case that Φ is a channel which is well-approximated by a Pauli channel (so that in effect it is subject to only decoherent Pauli noise), we may make a stronger statement [109]. For such a channel, we have $\Phi(\sigma^{(i)}) = \alpha_i \sigma^{(i)}$ for some scalar α_i. (If the noise process is weak enough that the error-free mode dominates, we have $\alpha_i \geq 0$ for each i.) Depending on the relative importance of the Pauli errors in the channel, there is one Pauli spin operator $\sigma^{(m)}$ which is most suppressed by the memory channel Φ, in that $\alpha_m = \min_i \alpha_i$. Then it is easy to show that $\mathcal{R}(\Phi) = \alpha_m^2$. This implies that the integrity of Φ may be experimentally determined by a procedure in which

we prepare eigenstates $|\varphi_{\pm}^{(i)}\rangle$ of the Pauli operators $\sigma^{(i)}$ as input, store them in the memory, and then determine the probability with which we obtain the ± 1 outcome when we perform a $\sigma^{(i)}$ measurement on the retrieved state. In the simulations performed in this chapter the simulations are all of a depolarising error model, as such no minimisation is required as there is no preferred basis for error.

It is worth emphasising that $R(\Phi)$ is a function on the memory channel Φ itself, thus one should speak of the *integrity of the memory*. It is understood that the memory channel is used for some defined time τ, and that if the same memory system were used for a longer time then its integrity would be lower; typical channels will have zero integrity as $\tau \to \infty$.

The integrity of the memory has a highly intuitive and natural meaning through the following scenario: We suppose that Bob initially knows nothing about Alice's choice of qubit to encode, but after Bob has completed his decode process to obtain the single qubit we then describe to him two choices: either Alice's initial qubit was ψ or it was $\psi_{\perp}$. Bob then makes a measurement of his choice to try to determine whether it is $\Phi(\psi)$ or $\Phi(\psi_{\perp})$ that he has received. The integrity $\mathcal{R}(\Phi)$ tells us Bob's probability p_g of guessing successfully according to

$$p_{g,\text{worst}} = \tfrac{1}{2} + \tfrac{1}{2}\mathcal{R}(\Phi) \quad \text{and so} \quad \mathcal{R}(\Phi) = 2p_{g,\text{worst}} - 1. \tag{3.6}$$

Here the label 'worst' reminds us that this is the lowest success probability, i.e. when the options ψ, $\psi_{\perp}$ are the ones least well preserved by the memory (if indeed there is any variation). The integrity is therefore a natural measure of how well a memory preserves the distinctiveness of different states.

Notice that we constrain Bob to use a specific method to identify the received state. He must map the logical qubit back to a single physical qubit by first applying a standard round of error correction for the code in question, then applying the inverse of Alice's encoding circuit. Bob's sole freedom is that he can choose how to measure that final physical qubit. As we explain in Section 3.5, by constraining Bob this way we ensure that his performance is associated with the code structure and its capacity to protect information. In that section we discuss the performance of a more powerful agent who is given full information about the error channel and complete license to perform any operations on all n received qubits; this agent generally has very similar (sometimes identical) performance to our constrained Bob.

For a typical memory Φ (although not for all conceivable memories) Bob's correct strategy for his final step is the obvious one: just measure in the basis $\{|\psi\rangle, |\psi_{\perp}\rangle\}$ and make the guess correspondingly. Then Bob's success probability is simply the

fidelity of the state $\Phi(\psi)$ with respect to Alice's initial state ψ, since the fidelity of any state with respect to a pure state is the probability of obtaining that outcome in a measurement (as was remarked following Eqn. (3.1)). So in this case Bob's probability of guessing correctly is simply $p_g = \mathcal{F}(\psi, \Phi(\psi))$. Moreover his worst performance is

$$p_{g,\text{worst}} = \min_{\psi} \mathcal{F}(\psi, \Phi(\psi)).$$

But given Eqn. (3.6) we can now offer a precise meaning to the idea of "the (worst case) fidelity of a logical qubit stored in the memory" for any channel where Bob would opt to measure in the basis $\{|\psi\rangle, |\psi_\perp\rangle\}$. For such a channel,

$$F_{\text{logic}} = \tfrac{1}{2} + \tfrac{1}{2}\mathcal{R}(\Phi).$$

Loosely, F_{logic} is the fidelity after we project into the logical subspace of the code with a perfect round of error correction. For memory channels with sufficiently complex noise maps $\mathcal{N}$ that Bob's choice of measurement basis would *not* be $\{|\psi\rangle, |\psi_\perp\rangle\}$, the very idea of the "fidelity of a logical qubit stored in the memory" becomes ill defined. Thus, integrity is a general measure which relates to the notion of logical fidelity *when the latter notion makes sense*. However integrity remains well-defined and meaningful even when the logical fidelity does not: it is the more general and robust concept.

Integrity is useful as it lends itself naturally to an experimental setting. The is no need for state tomography or methods of building up density matrices to calculate fidelities. Notice that although the definition Eqn. (3.5) refers to two different states, we would evaluate the integrity $\mathcal{R}$ through a series of single uses of the memory – we simply follow our scenario described above involving Bob guessing between options and employ Eqn. (3.6). Consequently the costly process of performing full state tomography is not required.

Importantly, while the definition describes the encoding and decoding as occurring perfectly (conceptualised by saying that Alice and Bob are perfect), one can show that in practice they can be made imperfect and yet the experiment can gauge the integrity with good accuracy – this feature is discussed in more detail in the published paper [98]. Also, under certain circumstances (discussed further in Appendix C.3), Alice and Bob would obtain enough information to accurately assess integrity by only testing Pauli eigenstates. Theses states can often be prepared fault-tolerantly, which proves particularly important when considering noisy Alice and Bob.

	Theoretical protocol for measuring integrity
1a 1b	Alice (perfect) prepares a single qubit state $\|\psi\rangle$ or $\|\psi_\perp\rangle$. Alice perfectly encodes it into the n physical qubits.
2	From $t = 0$ to τ the n qubits are in the memory; noise occurs from environment and possibly error correction.
3a	Bob (perfect) performs error correction on the n qubits, then decodes (inverse of 1b) the state to a single qubit.
3b	Bob is told Alice's qubit was *either* $\|\psi\rangle$ *or* $\|\psi_\perp\rangle$. He measures his qubit and guesses, success probability p_g.

Table 3.1: Evaluating the integrity of a memory system. See also Fig. (3.1).

3.2 Relation and comparison to existing measures

We choose to use the integrity as a measure of how well the channel Φ preserves the distinguishability of pairs of states. Ideally when modelling the effect of storing a state in a memory for some period of time and retrieving it, we would want Φ to be a good approximation of the identity map.

Given a memory system uncoupled to any ancillary system, we could then have choosen to use the induced trace distance between Φ and the ideal memory,

$$\mathcal{D}_1(\Phi, \mathbb{1}) = \max_\rho \mathcal{D}(\Phi(\rho), \rho), \tag{3.7}$$

to measure the quality of the memory. The smaller the induced trace distance the higher quality the memory. More generally we would like to take into account that the state ρ most sensitive to the error channel might be one entangled with an error-free ancillary system. This means making reference to the diamond norm [110],

$$\left\|\Phi - \mathbb{1}\right\|_\diamond = \left\|(\Phi - \mathbb{1}) \otimes \mathbb{1}\right\|_{\mathrm{tr}}. \tag{3.8}$$

which would quantify the probability of distinguishing Φ from $\mathbb{1}$ in a single use of the channel [111]. The induced trace distance measures this but without access to an ancillary system not acted on by Φ. For our purposes however these metrics do have weaknesses. We might think that for a high quality channel which will be used repeatedly, the single use distinguishability is not the most important metric by which to judge it. After all if Φ can be distinguished from $\mathbb{1}$ in one use, the memory is pretty useless. A more precise objection to use of the induced trace distance and/or the diamond norm is that they would both make a quantitative distinction between between a channel producing a constant pure state $\Phi_0(\rho) = |\alpha\rangle\langle\alpha|$ and the completely

dephasing channel $\Phi_1(\rho) = \frac{1}{d}\mathbb{1}$, however we would want to quantify these as equally useless memories as neither preserves any information.

The integrity $\mathcal{R}$ and the induced trace distance $\mathcal{D}_1$ are both defined in terms of the trace distance on states. There is thus a simple relationship between them. For any pair of pure orthogonal state we have

$$\begin{aligned}
&\tfrac{1}{2}\Big\|\Phi(\psi_0) - \Phi(\psi_1)\Big\|_{\mathrm{tr}} \\
&\geq \tfrac{1}{2}\Big|\, \Big\|\psi_0 - \psi_1\Big\|_{\mathrm{tr}} - \Big\|\Phi(\psi_0) - \psi_0\Big\|_{\mathrm{tr}} - \Big\|\Phi(\psi_1) - \psi_1\Big\|_{\mathrm{tr}}\Big| \\
&= \Big|1 - \mathcal{D}\big(\Phi(\psi_0), \psi_0\big) - \mathcal{D}\big(\Phi(\psi_1), \psi_1\big)\Big|. \qquad (3.9)
\end{aligned}$$

In the case that $\mathcal{D}_1(\Phi, \mathbb{1}) \leq \frac{1}{2}$, we may remove the absolute value bars, so that when ψ_0 and ψ_1 minimise the first line of Eqn. (3.9) we obtain

$$\mathcal{R}(\Phi) \geq 1 - 2\,\mathcal{D}_1(\Phi, \mathbb{1}). \qquad (3.10)$$

It is easy to see that this holds with equality when the minimising pure states $\psi_0 \perp \psi_1$ are both mapped to mixtures of ψ_0 and ψ_1, as for example occurs with dephasing and depolarising channels Φ.

To take a specific example of two memories we would consider equally useless: In the case $\psi_0 = |0\rangle\langle 0|$ and $\psi_1 = |1\rangle\langle 1|$ if we have two equally memories we would like to find equally 'useless' such as $\Phi_0(\rho) = \psi_0$ and $\Phi_1(\rho) = \frac{1}{2}\mathbb{1}$ for all qubit states ρ, we would have

$$1 - \mathcal{D}_1(\Phi_0) = 1 - \mathcal{D}(\psi_0, \psi_1) = 0, \qquad (3.11a)$$

$$1 - \mathcal{D}_1(\Phi_1) = 1 - \mathcal{D}(\tfrac{1}{2}\mathbb{1}, \psi_1) = \tfrac{1}{2}, \qquad (3.11b)$$

so use of the induced trace distance would erroneously imply Φ_1 is superior. Whereas the fact that neither of these channels Φ_i preserve any information about their input implies that we should score them equally, which we would using the integrity:

$$\mathcal{R}(\Phi_0) = \mathcal{D}(\psi_0, \psi_0) = 0 = \mathcal{D}(\tfrac{1}{2}\mathbb{1}, \tfrac{1}{2}\mathbb{1}) = \mathcal{R}(\Phi_1). \qquad (3.11c)$$

Similar arguments would apply to the use of the diamond norm as ultimate measure of the memory's utility, or to be more precise the quantities $\mathcal{D}_1(\Phi \otimes \mathbb{1}, \mathbb{1} \otimes \mathbb{1})$ and the integrity of the channel $\mathcal{R}(\Phi \otimes \mathbb{1})$

3.3 Milestones toward successfully protected memories

Described simply, the integrity is the worst-case probability that the state of a stored qubit can be inferred by Bob, and this gives us a simple experimental procedure to assess it. This section identifies some milestones towards the goal of superior code-based quantum memories based on this metric.

For convenience of exposition we may imagine that a third party, besides Alice and Bob, is responsible for the cycle(s) of error correction performed during the memory period: since this individual is effectively a flawed assistant for Bob, we use the name Igor after the famous fictional lab assistant.

In a similar spirit [33] has described 'phases' of development for surface code realisations. The authors emphasise the crucial task of scaling so that phases beyond the first correspond to 'at least tens of qubits'. In this chapter we take the complimentary approach of stressing tipping points in performance, while remaining agnostic as to the code type and the number of physical qubits.

It is worth noting that it is already a non-trivial accomplishment to achieve logical encoding with good fidelity, and this can be established by verifying the logical qubit immediately after its creation – this is the scenario explored in [33]. In our approach the first milestone is the one that would naturally follow such an accomplishment, i.e. introducing Igor to perform an additional round of error correction mid-way. The logical encoding and decoding of the information in our protocol merely frame the process. In the published work upon which this chapter is based (Ref. [98]) the effect of having a noisy encoding and decoding process is explored further.

In similar work Gottesman [107] describes a minimal family of fault-tolerant circuits based on the four-qubit error detecting code for which 'fault-tolerance' can be demonstrated with only 5 qubits. This would demonstrate rigorous fault-tolerance, although only in post-selection, for at least ancilla preparation. This provides a simple milestone of "fault-tolerance demonstrated in a small quantum system" rather than a set of increasingly challenging benchmarks that we will outline. More generally rather than referring to measures on the quantum channel only complete quantum circuits are considered and the statistical distance of the classical output of the noisy circuit from that output by the 'ideal circuit'.

In Ref [107] Gottesman states that "It seems plausible that the best way to predict how well fault tolerance will work on a large system is to see how well it works on a small system", however there are other ways to address the power and performance of

small quantum systems, rather than simply assessing their performance in performing error correcting codes. For example, in Ref [112] researchers at IBM introduced a 'hard to game' measure of a small system's utility as a quantum information processor which they called quantum volume. In essence this takes the square of the either the number of qubits or the achievable circuit depth, whichever is the smaller number, as the ultimate measure of how 'powerful' a small controlled quantum system is. This makes no reference whatsoever to the ability of the system to correct its own errors, the problem on which we are focussing and to which Gottesman refers, but is a more general, if slightly arbitrary, measure of the ability of a small quantum information processor to implement algorithms. We choose to investigate specifically milestones towards performance of a strictly superior quantum memory that uses quantum error correction, with the view that long-term quantum computation will need fault-tolerance to achieve circuit depths needed for interesting problems.

3.3.1 M1: Beneficial error correction

We initially focus on the case where at most one error correction cycle is used during the entire period τ where memory operates, i.e. in between Alice ($t = 0$) and Bob ($t = \tau$). We therefore now specify Step 2 of Table 3.1 in more detail, setting it out in Table 3.2. The key idea will be to compare the integrity of the memory channel without error correction (no Igor) to the case with EC (Igor participates) and determine whether the latter is superior.

Let us use the symbol Φ^m to label the memory channel when Igor performs m rounds of error correction, so that Φ^0 labels the channel when no QEC is performed (i.e. noise sources are purely environmental). Then we say that a round of error correction is beneficial if Bob's probability of subsequently discriminating the state correctly is higher when Igor indeed performs that round, i.e. when

$$\mathcal{R}(\Phi^1) > \mathcal{R}(\Phi^0). \tag{3.12}$$

This criterion for successful error correction can be summarised as, "Is Igor a help or a hinderance to Bob?".

This seems entirely straightforward but there is a subtlety: the question of whether Eqn. (3.12) is satisfied will depend on the duration τ of the memory channels (here we would naturally set the same τ for both Φ^1 and Φ^0 for a fair comparison). Since we are interested in defining a first milestone for experimental efforts, we say that error correction *can be* beneficial if

$$\mathcal{R}(\Phi^1) > \mathcal{R}(\Phi^0) \quad \text{for some value of } \tau. \tag{3.13}$$

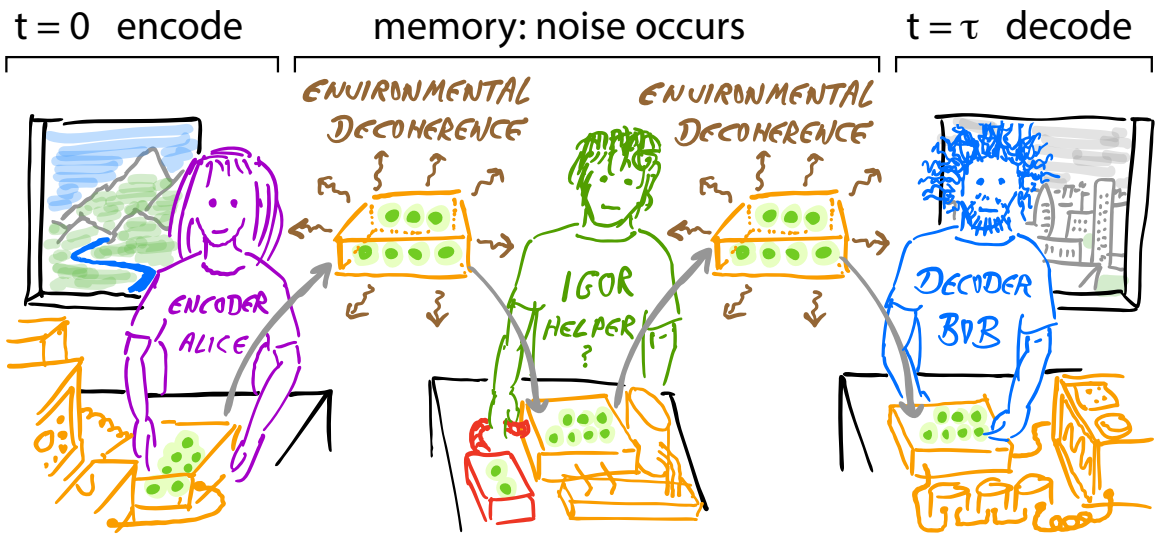


Figure 3.1: **Adapted from [113]: Cartoon of the protocol for assessing the integrity of an error-corrected memory**. Alice and Bob perfectly perform the encoding and measurement, respectively, of a logical qubit. Meanwhile Igor is an imperfect agent attempting error correction to fight noise. [Figure credit: Simon Benjamin].

We will refer to the challenge of satisfying Eqn. (3.13) as milestone **M1: Beneficial error correction**.

One might wonder if we should consider a stronger condition: $\mathcal{R}(\Phi^1) > \mathcal{R}(\Phi^0)$ for *all values* of the common duration τ. It is interesting and important to note that this condition will be impossible to satisfy in physical devices where the process of error correction is very fast compared to the rate of environmental decoherence. This applies, for example, to ion trap devices with clock-transition qubits where the environmental decoherence time can be on the order of minutes but gate operations are sub-millisecond. We need only assume that environmental decoherence is a continuous process to see that $\mathcal{R}(\Phi^0) \to 1$ as $\tau \to 0$, i.e. the integrity of the simple memory channel is arbitrarily close to unity for a sufficiently short value of the memory duration τ. In other words, finite environmental noise needs finite time to occur. We can inspect the equivalent limit for Φ^1 if we assume that Igor's error correction cycle is instantaneous (whereas if it takes finite time δ then we cannot employ memory channel Φ^1 for time durations less than $\tau < \delta$). But given instantaneous error correction, we find $\mathcal{R}(\Phi^1) \to \epsilon$ as $\tau \to 0$, where ϵ is non-zero and is related to the inevitable imperfections in the cycle of error correction, i.e. the circuit elements such as state initialisations, one- and two-qubit gates, and measurements will all have finite infidelity. What we are noticing is that it is not desirable to perform error correction

	Without Error Correction: Memory Φ^0
2a	The n physical qubits are subjected to environmental noise for a time τ.

	With Error Correction: Memory Φ^1
2a	The n physical qubits are subjected to environmental noise for a time $(\tau - \delta)/2$.
2b	Optionally, Igor is asked to apply a full round of *imperfect* error correction, taking time δ.
2c	The n physical qubits are subjected to environmental noise for a further time $(\tau - \delta)/2$.

Table 3.2: Expanding on Step 2 of Table 3.1 when we wish to assess the benefits of error correction.

'as frequently as possible' – we should wait for a finite time before applying an error correction cycle, so that its negative impact on the memory is justified by the positive gain. This is made very apparent by Fig. 3.5 in the next section. Of course, if the time required for error correction is comparable to the environmental decoherence rate, as may be the case for superconducting qubits, then one never has the luxury of waiting until the optimum time to perform error correction; cycles should indeed be performed 'back to back'.

3.3.2 M2: Beneficial multi-round error correction

While Eqn. (3.13) is an important first milestone for an error-correcting quantum memory, further milestones can also be identified. For a sufficiently high performing Igor, and a long memory duration τ, it will be beneficial to have multiple rounds of correction. This will be a signature of further progress toward a practical quantum memory. We would then find that

$$\mathcal{R}(\Phi^m) > \mathcal{R}(\Phi^{m-1}) \quad \text{for some value of } \tau. \tag{3.14}$$

Here we understand this need only be satisfied for some particular $m > 1$ (it seems likely that it would be achieved first for $m = 2$ but we do not insist on this). We will refer to the challenge of meeting the condition in Eqn. (3.14) as milestone **M2: Beneficial multi-round error correction**.

3.3.3 M3: Beneficial encoded memory

Equations (3.13) and (3.14) involve comparisons between memories which both employ encoded qubits. There is of course another type of comparison we can make, one

which directly addresses the question of whether it is 'worth' using encoded memories at all: we should contrast such a memory channel to a simple, single qubit memory. Let us use the symbol Θ for that memory channel. We can consider its integrity $\mathcal{R}(\Theta)$ easily enough. Alice prepares a single qubit, again choosing between ψ and $\psi_\perp$, but does not encode it into multiple qubits. It exists as a memory from $t = 0$ to $t = \tau$, and finally Bob receives it but of course he has no decoding to do. The qubit he receives, $\Theta(\psi)$ or $\Theta(\psi_\perp)$, differs from Alice's qubit only because of environmental noise. But as before Bob must measure it to guess between the two possible states, and as before his probability of success is simply $\frac{1}{2} + \frac{1}{2}\mathcal{R}(\Theta)$. For our actively-corrected encoded memory to 'beat' the simple single-qubit memory, we require

$$\mathcal{R}(\Phi^m) > \mathcal{R}(\Theta) \text{ for } \textbf{some } \tau_\Theta, \text{ while using } \tau_\Phi = \alpha\tau_\Theta. \tag{3.15}$$

Here we require only that this is true for some specific value of $m > 0$ (it seems likely that $m = 1$ would be the first demonstration). Note the more complex condition on the channel durations. In Eqn. (3.13) and Eqn. (3.14) it was clear that the duration τ of the two memory channels should be the same for a fair comparison. This is not necessarily true of the Eqn. (3.15) since one can argue that the meaningful time scale for a quantum memory is not 'wall clock' time but rather the time required to perform an active gate operation (perhaps the average time, given that circuit operations will differ in their time requirements, or perhaps the slowest time to be strict). Depending on the hardware platform and architecture, the time required to perform a gate operation on an encoded qubit may be longer than the time to perform the equivalent operation on an unencoded qubit. This would then suggest that τ_Φ should be longer than τ_Θ, and the factor $\alpha \geq 1$ is included in Eqn. (3.15) to reflect this. We will refer to the challenge of satisfying Eqn. (3.15) as milestone **M3: Beneficial encoded memory**.

Here we can make contact with the concept of a 'pseudo-threshold' (see e.g. Ref. [114]). This concept is typically used in the context of concatenated codes, where there may be several levels of concatenation required before error rates fall sufficiently. In the present context, we restrict our interest to the first level of concatenation where a process involving unencoded qubit(s) is compared to a process with a single level of encoding. The pseudo-threshold has been surpassed if a circuit performs to a higher standard with the encoded qubits, i.e. logical qubits, versus using the physical qubits directly. One might demand that for a complete universal set of operations, each operation at the encoded level outperforms the analogous

operation using unencoded qubits.In essence Eqn. (3.15) represents the (lowest tier) pseudo-threshold for memory, i.e. for the identity operation in a circuit.

3.3.4 M4: Strictly superior encoded memory

Assuming that Eqn. (3.15) can be satisfied, there is a higher goal which might be achieved namely

$$\max_m \mathcal{R}(\Phi_{QEC}^m) > \mathcal{R}(\Theta) \text{ for } \textbf{all } \tau_\Theta, \text{ and using } \tau_\Phi = \alpha\tau_\Theta. \tag{3.16}$$

Here the maximum is over a family of memory channels having the same duration τ_Φ but with differing numbers of error correction cycles m. Importantly, we permit $m = 0$. If this condition is satisfied, it means that for any desired duration we can sustain our encoded quantum memory at a higher integrity than a single physical qubit memory. We do so by applying a suitable number of error correction cycles. Moreover this is true even allowing for the factor α discussed above. This is therefore the 'gold standard' for demonstrating a quantum memory and it is the most challenging of the criteria we have presented in this section. We will refer to the task of satisfying Eqn. (3.16) as milestone **M4: Strictly superior encoded memory**.

This section has presented four milestones in an order which may represent an increasing degree of challenge. It is not necessarily the case that each is more difficult than the last – for example, conceivably M3 may be achieved before M2 in a given physical device. However the fourth milestone is clearly the most demanding and we should expect that the inequalities in Eqns. (3.13), (3.14) and (3.15) must all be satisfied before Eqn. (3.16) can be achievable.

3.4 Numerical studies

This section presents simulation results under the Alice-Igor-Bob framework described in Tables 3.1 and 3.2. The simulation technique is based on the Monte Carlo method, pure states are simulated many times with noise randomly applied in each run. It is worth emphasising that the integrity benchmark is appropriate for any and all error models, including coherent noise, non-Markovian noise and so on. The simple canonical Pauli depolarising noise model (on all elements including state preparations, gates, measurements, and environmental noise) is employed in this section since it is a standard model to use in a first investigation. We aggregate a large number of individual runs, in each of which a pure state undergoes a specific trajectory: after every circuit element is applied, a classical random number is generated and

compared with the error rate for that circuit element in order to decide whether an error is applied and if so its type.

Environmental decoherence is modelled as a depolarising process that occurs independently for each physical qubit as defined in Chapter 1.2.1. Noise also occurs when gate operations are applied by 'Igor' while performing error correction cycle(s) during the memory channel in order to actively protect the stored information. Recall that Alice and Bob, whose actions at $t = 0$ and $t = \tau$ frame the memory channel, are considered ideal for the purpose of the definition of integrity. In all these cases our error model for circuit operations are as described in Chapter 1. The same error probability p_e is used for all types of circuit element; this is the number that is specified as 'Igor's error rate' and typically expressed as e.g. 0.3%.

All the simulations presented use the Alice-Igor-Bob scenario that has been discussed in Tables 3.1 and 3.2. The circuit level description is shown in Figure C.5, where the five-qubit code is used as an example. Igor performs his error correction cycle halfway through the duration of the memory channel (or for a channel with n correction cycles, at points $t = m/(n+1)$ with $m = 1, 2, \cdots, n$). Igor measures a complete set of stabiliser measurements and applies error correction based on the error syndrome. Igor's action is taken to be instantaneous although it is trivial to assign it a finite time δ as indicated in Table 3.2. If Igor's analysis indicates that a correction is necessary, then the appropriate correction is applied *perfectly* – in reality one can simply note the need for correction and update the Pauli frame, thus never needing to apply an imperfect physical operation the qubits. Note that altering this principle to apply a noisy fix would make negligible difference to the observed integrity, since it is merely one additional operation for Igor's circuit which at minimum involves over a dozen gates.

Three well-known codes are evaluated: the five-qubit code which is the smallest possible error correcting code [115], the seven-qubit Steane code [30] and the nine-qubit surface code [116]. Each of which we have met in Chapter 1, the nine-qubit surface code being the 'rotated' variant of a distance 3 planar code [52]. We will compare the inherent properties of these codes, both in their simple and fault-tolerant variants, and we will show examples where the various milestones described in the previous section are (or are not) met.

3.4.1 Investigating integrity with the five qubit code

To start we explore some examples with the five qubit code. Typically the graphs in this section are of the general form exemplified by Fig. 3.2(a). On the vertical

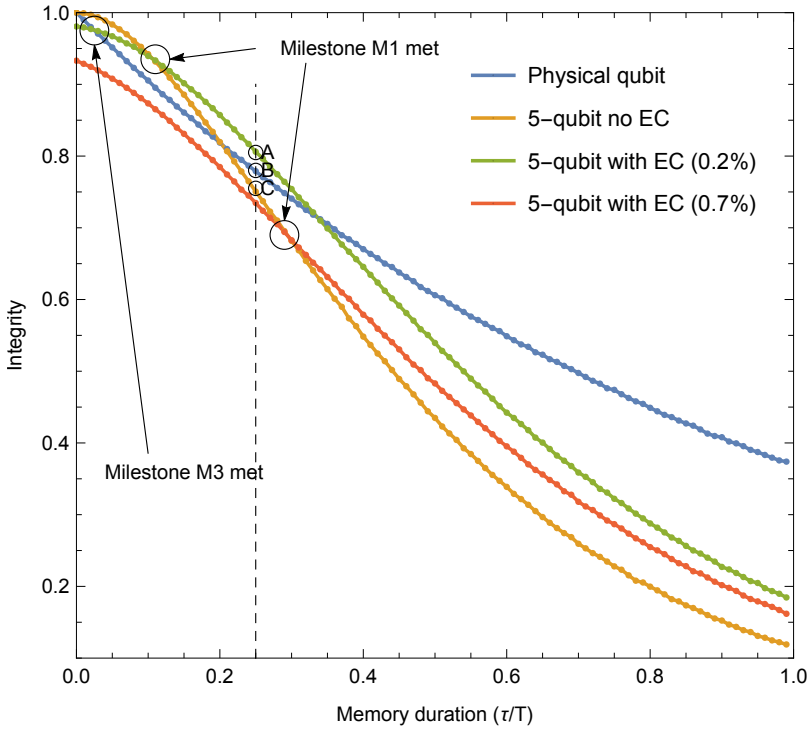


Figure 3.2: **The integrity of different types of memory channel.** Memory integrity assessed over many different durations τ ranging from zero to T, the single-qubit decoherence time. Blue line: A single physical qubit, i.e. no encoding. Orange: a memory using the five-qubit code for the stored qubit, but without active correction during the channel. Green: the same five-qubit code, but now with a round of error correction performed mid-way through the memory duration, i.e. at time $t = \tau/2$. Error rate in operations during the correction cycle is 0.2%. Red: As for green, but error rate 0.7%.

axis we show the integrity, as defined earlier, which of course is equal to unity for an ideal memory. The horizontal axis shows the duration τ of the memory channel(s) in question; the duration is shown as a ratio with respect to the environmental decoherence rate T which is the decoherence time of an isolated single physical qubit. Each point along a curve in the figure is thus the integrity of a specific memory channel of duration indicated by the horizontal axis.

There are four types of memory channels shown in Fig. 3.2(a): The simple one-qubit memories Θ (shown in blue), encoded channels without Igor's error correction, Φ^0 (orange), and two sets of channels where Igor does perform one round of correction Φ^1 (green, red). The last two differ only in that Igor's error correction circuits have error rates 0.2% and 0.7%, respectively. In all cases the encoded channels are using the five-qubit code. Encoding and decoding tasks, performed by Alice and Bob, are perfect as per the definition of integrity.

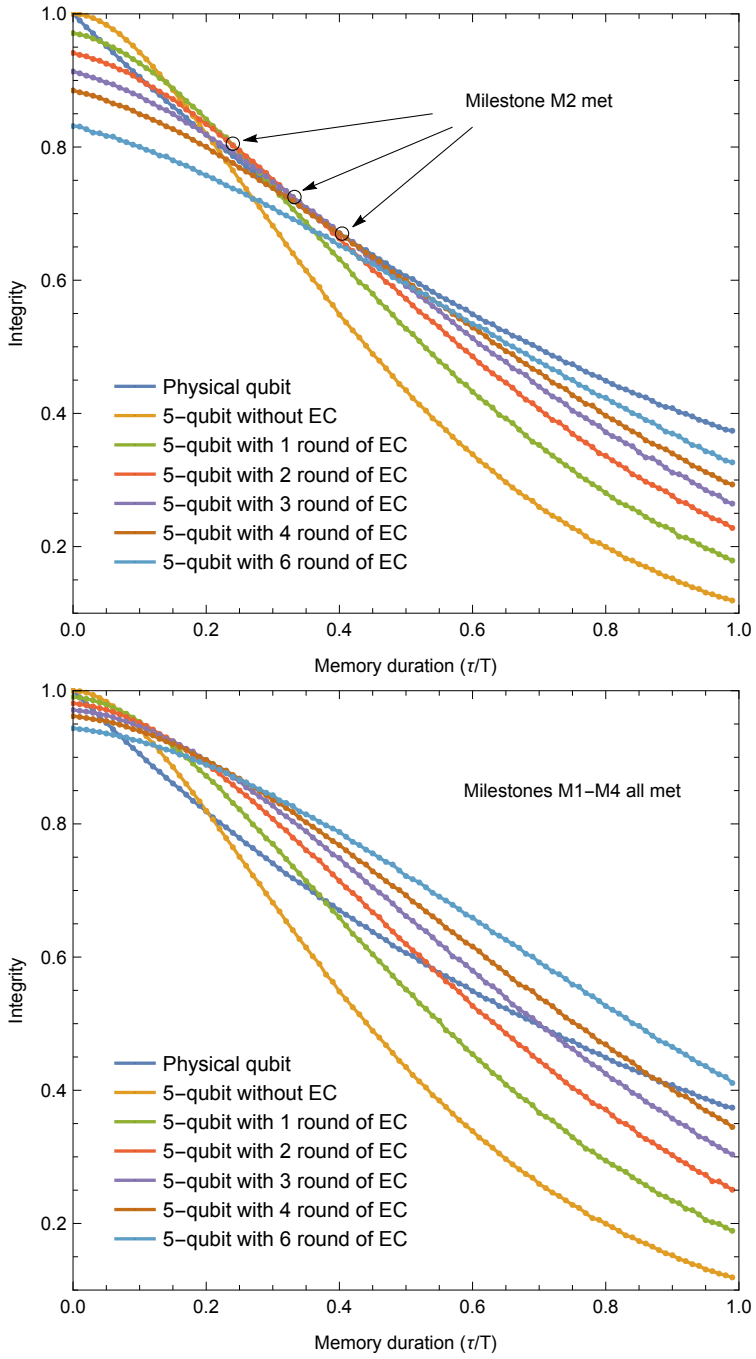


Figure 3.3: **Multiple rounds of quantum error correction (EC).** The integrity of a family of memory channels all employing the five-qubit code but differing in the number n of rounds of error-correction performed during the memory, where $n = 0, 1, 2, 3, 4$ or 6. Our imperfect agent Igor performs error correction cycles at times $t = m\tau/(n+1)$ for $m = 1..n$. In the upper panel (a) Igor's gate-level error rate is 0.3%. As explained in the main text, the system meets milestones M1, M2 and M3 but fails to meet M4. In the lower panel (b) Igor's error rate is now 0.1% and we see that by choosing a suitable n we can select a five-qubit encoded memory that will beat the single-qubit memory for any desired duration τ, so meeting milestone M4: Strictly superior encoded memory.

Igor's error rate of 0.2% is low enough for him to perform well and consequently we observe two desirable line crossings in the figure. For all durations $\tau > 0.11\,T$ we see that the error corrected memory Φ^1 (green) is superior to the un-corrected memory Φ^0 (orange). Thus for any $\tau > 0.11\,T$ we meet the *M1: beneficial error correction* milestone specified earlier in Eqn. (3.13). Furthermore, for all durations τ between $\tau \simeq 0.025\,T$ and $\tau \simeq 0.34\,T$ the corrected memory Φ^1 (green) has superior integrity to the single-qubit memory Θ (blue). Note however that in comparing the single-qubit channel Θ to the encoded channels, we have not introduced any scaling factor to adjust their relative durations (i.e. we have set $\alpha = 1$ in Eqn. (3.15)). This might be considered unreasonable unless the physical platform embodying the memory system is capable, in principle, of performing transversal gates in one step so that operations on logical qubits are on the same timescale as operations on physical qubits. With this important caveat, we can say that for any duration in the range $0.025\,T < \tau < 0.34\,T$ we can meet the *M3: beneficial encoded memory* milestone, Eqn. (3.15).

The red line, corresponding to the higher per-gate error rate of 0.7% during Igor's error correction, never surpasses the integrity of the single physical qubit memory; therefore this channel does not meet milestone *M3: beneficial encoded memory*. However when the duration $\tau > 0.29\,T$ it does slightly surpass the integrity of the Φ^0 channel. Therefore milestone *M1: beneficial error correction* is met.

3.4.2 All milestones with five qubit code

In preceding figures we have focused on cases where a single round of error correction is applied during a memory channel, and we have identified points where our milestones M1 and M3, would be satisfied. In Figure 3.3 we show an example of when the use of multiple rounds of error correction (equispaced within the duration of the memory channel) with the five qubit code may allow us to meet milestone *M2: Beneficial multi-round error correction* associated with Eqn. (3.14), or even milestone *M4: Strictly superior encoded memory* associated with Eqn. (3.16). In the upper panel, Igor's error rate suffices for the former but not the latter; in the lower panel Igor's error rate is set to 0.1% which proves to be sufficient to achieve the fourth milestone.

3.4.3 Integrity at interruption

Integrity is defined as a property of an entire channel; but we can ask what would happen if Bob were to 'step in early' at any time between $t = 0$ and $t = \tau$. We

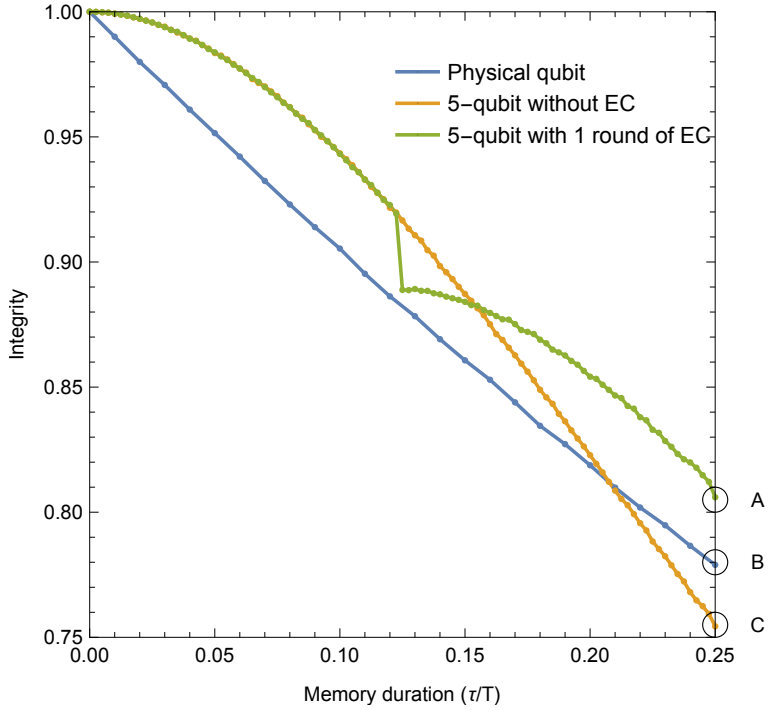


Figure 3.4: **Integrity change during a given memory channel (lower).** We plot the 'integrity at interruption' in order to look inside three specific memory channels during their operation. The three channels all have duration $\tau = 0.25\,T$. and are taken from the points labelled A, B, and C in Fig. 3.2(a). The key feature is the step-like decline occurring at $t = \tau/2 = 0.125\,T$ when Igor performs imperfect error correction.

suppose that Bob would perform his usual decoding, measurement and guess using the state of the memory system at that premature point. From his performance we can infer an 'integrity so far', which we call the 'integrity at interruption'. This is an interesting number, as it will reveal more precisely what the effect of introducing Igor has been.

Three points labelled A, B, and C have been highlighted in Fig. 3.2. They lie at a value of the duration, $\tau = 0.25\,T$ for which the high-fidelity corrected channel is superior to the single physical qubit memory Θ, which in turn is superior to the encoded-but-uncorrected channel Φ^0. One might wish to understand how the integrity varies *over the course* of the duration of those memory channels. Fig. 3.4 shows this quantity. The blue and orange lines, which correspond to the single-qubit memory Θ and the encoded memory without correction Φ^0, do not reveal anything interesting. Indeed they precisely correspond to the same lines in the region $0 < \tau < 0.25\,T$ in

the Fig. 3.2 (in effect, we have just 'zoomed in'). For these two cases the noise on the memory is simply a continuous process; when Bob interrupts our channel that should have had duration $\tau = 0.25\,T$, it is exactly equivalent to having a memory of the shorter duration.

The green line in Fig. 3.4 corresponding to the error-corrected channel Φ^1 is far more interesting. It is exactly coincident with the Φ^0 line until $t = 0.125\,T$ because in these cases Bob interrupts before Igor performs his error correction cycle. But then the 'integrity at interruption' falls sharply, i.e. there is a significant difference between Bob interrupting immediately before Igor's effort, versus doing so immediately afterwards. The reason is that Bob's process of decoding the memory begins with a round of error correction and *his* error correction is perfect; thus it can only be worse to have Igor apply his own flawed effort at correction immediately beforehand. However despite the sharp step down, the eventual integrity after full duration time $0.25T$ is higher. This is because Igor's efforts reset the accumulation of errors, lowering the overall chance of an uncorrectable set of errors (i.e. 2 or more errors, for the five-qubit code) over the course of the complete memory channel. We see this evidenced by the inverted parabolic curve immediately after Igor's action: in effect the environment must 'start again' to build up significant probability of weight-2 errors.

We have observed that error correction cannot increase the quantity 'integrity at interruption', assuming we have no knowledge of the initial encoded qubit (given such knowledge we can trivially increase integrity by erasing the memory and reinitialising it). Any form of error correction, with whatever code and however well performed, is a process that merely 'delays the inevitable' in the sense that integrity must fall; we can only alter the rate at which it falls. For the ultimate goal of fault tolerant quantum computing, we must slow the decay of integrity to such an extent that the entire calculation can take place before an error becomes likely. As will be shown in later chapters on magic state distillation the code size (or resource cost) must be chosen such that there probably will not be a logical error in a whole computation. The 'losing battle' that is being fought against quantum errors just has to ensure we do not lose during the implementation of the algorithm. The fact that integrity is a non-increasing function of time is a merit versus over other measures (such as the simple fidelity with respect to an ideal state) which can both fall and rise, and could thus create the impression that quantum information is being regenerated.

A related observation is the following: The rate at which we should apply error correction cycles has some optimum which depends on the relative severity of

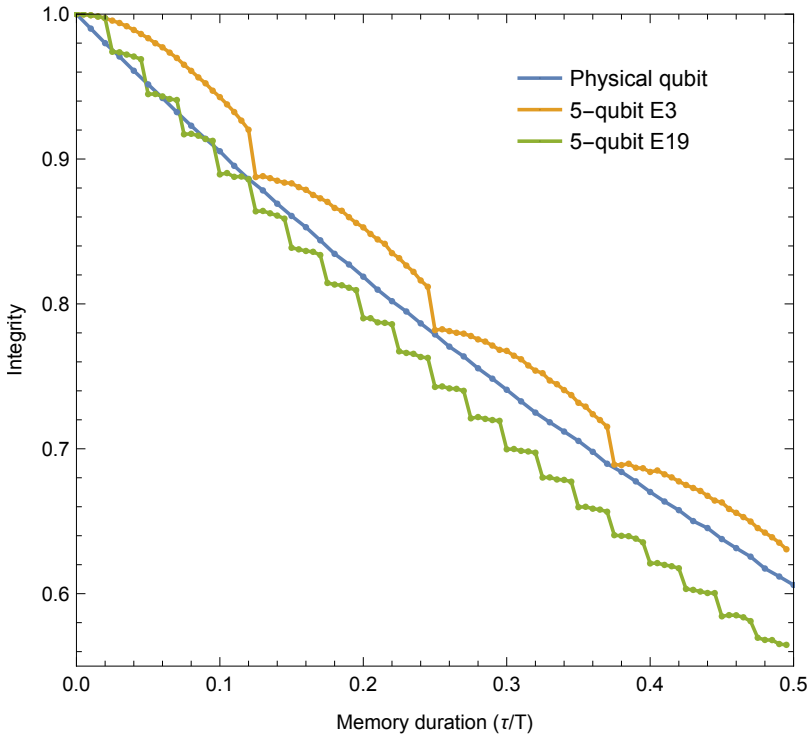


Figure 3.5: **Integrity change during three different memories, each of duration $\tau = 0.5T$.** This figure is equivalent to Fig. 3.2(b), but with three rounds (orange) or nineteen rounds (green) of error correction (gates error rate 0.2%) applied during the period of memory storage. Clearly three rounds of error correction sustains the logical qubit while too many rounds corrupt the logical qubit.

environmental decoherence per unit time versus the error rate within our error correction process (the noise in Igor's circuits). We should not apply error correction more frequently than this rate, or else the loss of the integrity will be dominated by the noise we introduce in our error correction cycles. This is made apparent by the simulation results shown in Fig. 3.5 which again shows the 'integrity at interruption' as in Fig. 3.4, but now for three different channels of common duration $\tau = 0.5\,T$, the channels being the single qubit memory Θ, and memories using three or nineteen error correction cycles (Φ^3 and Φ^{19}). From the right hand side of the graph we find the integrities of the three memory channels: they are approximately 0.61, 0.63 and 0.56 respectively, i.e. the memory channel featuring nineteen correction cycles is by far the worst, while three cycles provide a superior integrity versus the single qubit memory. The reason is clear from inspecting the curves: the 'integrity at interruption' reveals that the decay of the over-corrected channel is indeed dominated by the step-like drops associated with noise from Igor.

3.4.4 Code comparison

This section now presents a series of simulations which compare different codes, and also compare fault-tolerant versus non-fault-tolerant implementations of error correction circuits. We use the standard error model of homogeneous Pauli noise occurring without correlation, and for Igor's circuits the noise occurs on all circuit operations with equal probability. It is worth stressing that the relative performance of the codes may differ greatly when this error model is substantially varied.

As a first step toward comparing the efficacy of different codes, it is useful to begin by reporting a special case which is achieved by setting the memory duration to zero, and simply investigating the impact of the error correction process itself. Thus, we take a perfectly encoded qubit prepared by Alice and present it directly to Igor who performs an (unnecessary) error correction cycle before passing the encoded qubit directly to Bob for his analysis. The reduction in integrity is thus purely due to Igor's action. The results are shown in Fig. 3.6. Notice that in contrast to all other figures in this section, the horizontal axis here is not time (since the duration is zero) but rather Igor's error rate.

Figure 3.6 includes eight different options for the encoding and correction of a logical qubit. Three different codes are considered: the five-qubit code, the seven-qubit Steane code, and the nine-qubit surface code. For each of these, the performance of a non-fault-tolerant (non-FT) Igor is plotted. For the nine-qubit code, a second curve shows the performance when Igor employs a specific FT error correction circuit

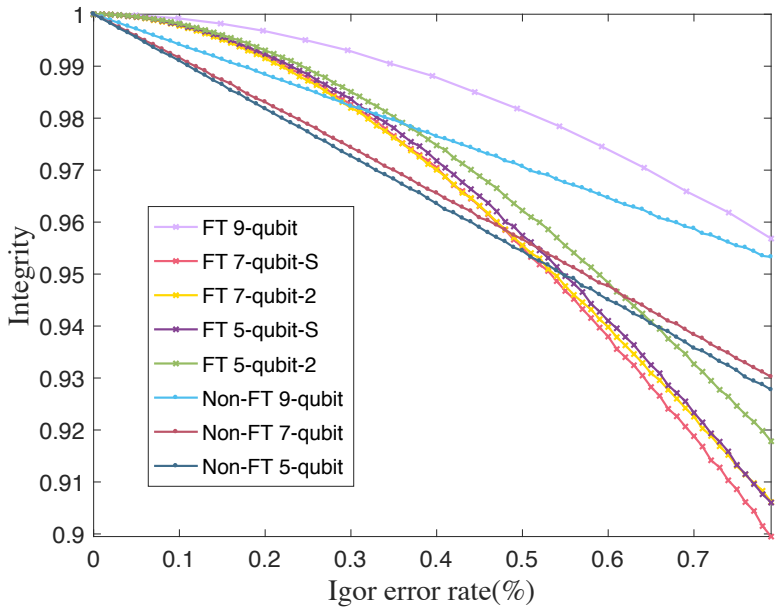


Figure 3.6: **Integrity change with increasing gate error rate.** Here the duration of our memory is set to zero, in order to directly inspect the negative impact of an imperfect error correction performed by Igor. The horizontal axis shows the level of noise associated with each circuit element of Igor's circuits. We analyse memories based on the five-qubit, the Steane, and the nine-qubit codes. Igor's error correction is performed either in a simple, non-fault tolerant fashion or with full fault tolerance. As explained in the text, the various line shapes and the relative levels of performance are straightforward to understand qualitatively. [Figure credit: Xiao-si Xu.]

(see Fig. C.6(e)). For each of the other two codes, the figure shows the performance of two different FT circuits: The standard 'Shor' approach using four ancilla qubits, and an alternative method proposed by Chao and Reichardt [36] which requires only two ancillas, see Fig. C.6 panels (c) and (d).

There are several interesting general observations to be made from Fig. 3.6. Firstly, two logically-necessary features are indeed present: One observes that all the cases which employ non-fault-tolerant (non-FT) error correction for Igor have the expected linear decay as Igor's error probability p increases from zero: integrity goes as $1 - cp$ for some constant c because non-FT circuits are vulnerable to single errors. Meanwhile the scenarios featuring FT error correction all have the expected inverted-parabolic shape: the integrity goes approximately as $1 - kp^2$ when Igor's error probability p is small. The fault-tolerant circuits for these codes are indeed 'immune' to single errors and vulnerable only to weight two (or higher) errors. Note that for higher (but still sub-1%) error rates for Igor, the fault tolerant circuits be-

come inferior to the simpler non-FT circuits. The reason is essentially combinatorial scaling: the FT-circuits are generally considerably more complex with far more gates, thus as gate failure probability p increases the risk of a double error in these complex circuits eventually outweighs the risk of a single error in the simple non-FT circuits. Thus one should not suppose that 'fault tolerant circuits are always better' – for small codes and appreciable rates of gate error, they may not be.

The different gradients in the various linear and parabolic curves can be qualitatively understood by considering two aspects of the codes and their implementation. The first is the proportion of all possible weight-2 errors that actually prove to be correctable. For example, the five-qubit code is corrupted by all weight-2 errors, but the seven-qubit Steane code can correct any pair of errors if (and only if) one is of type X and one of type Z. The nine-qubit surface code has the highest portion of 'harmless' weight-2 errors in this sense. The relative ordering of the non-FT codes can be explained by this feature alone. However for the FT codes, there is second and competing feature: as noted above the complexity of the FT error correction circuits is what 'kills' their performance, so simpler circuits are superior. Consistent with this principle, we see wherever an appreciable performance gap exists between the 'two-ancilla' variant of a FT code versus the 'Shor' variant of the same code, the former is always superior. Moreover the total size of the FT circuits for the five-qubit code (which has fewer stabilisers) is smaller than that of the seven-qubit Steane, thus among the FT curves the five-qubit outperforms the Steane. Remarkably FT circuits for the nine-qubit code exist which are actually very simple (as previous authors have noted [33, 117]), and thus the FT surface code benefits from both desirable features described here, and is unconditionally superior to all other codes in the plotted error range. However, it does require the largest number of qubits: the 9 data-qubits themselves, and Igor also requires 6 ancillas in order to perform stabiliser evaluation without error propagation.

It is important to remember that the comparison made in Fig. 3.6 is for zero environmental error. The relative performance of different codes will change once we deploy them properly into a memory channel where environmental noise is degrading the encoded qubit. Figure 3.7 shows the integrity change of the memory under our standard memory channel scenario Φ^1 i.e. 'one use of Igor's error correction midway' where the code employed is either the five-qubit, Steane, or nine-qubit code (all with non-FT correction). See Fig. C.5 for the explicit circuit used in the five-qubit code case; circuits for the other cases differ simply by substituting the appropriate stabiliser checks. All curves in this figure correspond to an internal error rate for

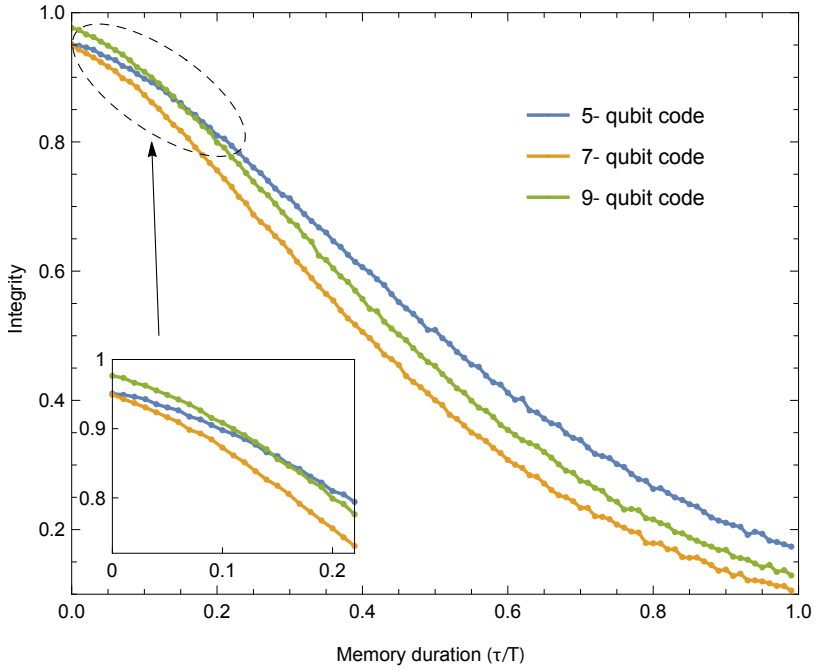


Figure 3.7: **Comparison between the 5/7/9 qubit codes.** The data shown are for our canonical Alice-Igor-Bob scenario where total memory duration τ during which pure environmental decoherence occurs continuously and a single (imperfect) round of error correction occurs midway at $t = \tau/2$. See e.g. Fig. C.5. The error rates used for all the gate operations during the error correction procedure are 0.5%.

Igor's operations of 0.5%. Thus the far left of the figure, with $\tau = 0$, gives us the same set of three data points as can be read from Fig. 3.6 when the $x-$axis, the error rate, is 0.5%. We see that the Steane code is marginally superior to the five-qubit code, but both are markedly inferior to the nine-qubit code. However, as we move away from the hard left of Fig. 3.7 to consider increasing duration of the memory, we find that the five-qubit code surpasses first the Steane and then even the nine-qubit code. The reason is that as more environmental error accrues, a code with a larger number of physical qubits will reach the point where two-or-more errors are present, i.e. the situation where the logical qubit may be corrupted, at an earlier time.

The code 'performance' here is of course only an investigation of one aspect of quantum codes. There are other merits beyond the question of how well a code preserves channel integrity – for example, the Steane code has the significant merit versus the smaller five-qubit that all Clifford operations can be applied transversally. The Steane code and small surface code are also instances of small topological codes whose size can be scaled while maintaining local interactions between the qubits, something not available when concatenating a five qubit code with itself.

3.5 Comparison with a more powerful Bob

The integrity measure contains within its definition the notion that the agent Bob, who receives the memory state at the end of its duration, will perform a round of (perfect) error correction as the first step of his analysis.

However it is an interesting exercise to to make a comparison between Bob's ability to correctly guess the received state, as captured by the integrity, versus Bob's performance if he were given carte blanche to make his guess by performing *any* physically allowed process on all of the encoded physical qubits. The performance of such a Bob would correspond to the trace distance

$$p_B = \tfrac{1}{2} + \tfrac{1}{2} D\big(\rho'_{n,0}, \rho'_{n,1}\big),$$

where $\rho'_{n,0}$ and $\rho'_{n,1}$ are the two possible n-qubit post-channel states received by Bob, which were sent in the two orthogonal states (here labelled 0 and 1). In Figure 3.8 we show a comparison between the performance of this more powerful Bob, and the Bob as we have defined for the integrity measure. For both the five-qubit code and the Steane code there is a negligible difference when Bob is given this extra freedom. For the nine-qubit code there is a small difference. This indicates that the error correction process itself is not quite optimal: some measurements differing from the

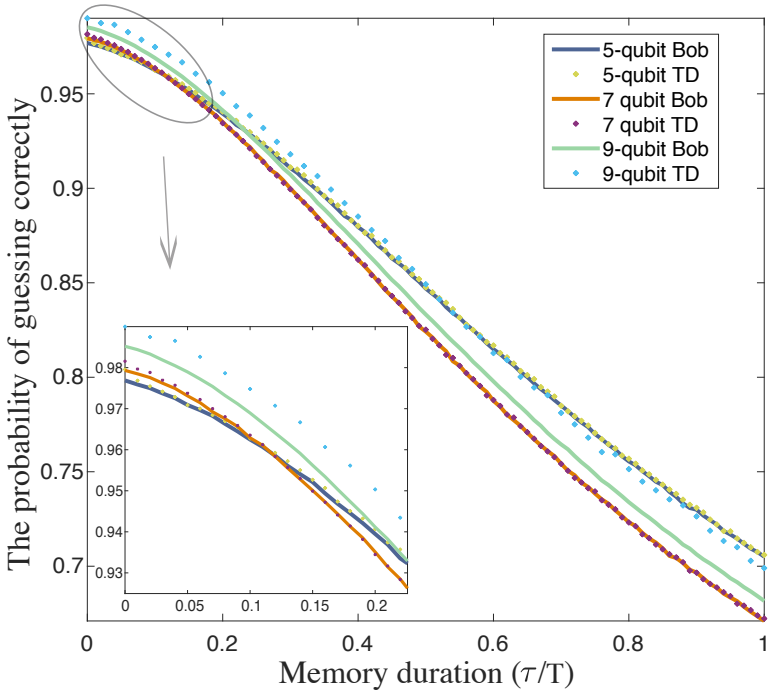


Figure 3.8: **Effect of a 'more powerful' Bob.** The solid lines here correspond to the performance of Bob as we have specified him within our definition of integrity. The vertical axis here is Bob's probability of making a successful guess, and the solid lines correspond to memory channels with a single round of Igor's error correction with error rate 0.5%. The dotted lines are the performance of a more powerful Bob as described in the text; the dotted and solid lines are essential identical except for the nine-qubit code. [Figure credit: Xiao-si Xu]

canonical nine-qubit code stabiliser measurements would permit a superior guess, and indeed the simple decoder we used could be replaced by a perfect 'look-up' decoder to improve performance of the nine-qubit code itself.

In order to achieve the higher level of performance that doesn't rely on the nine-qubit stabiliser measurements, Bob would require not only the freedom to make any measurements he sees fit on the received n encoded qubits, but also a complete understanding of the noise processes in the memory channel. In short, he would require an accurate theoretical description of the memory channel itself, so that he can derive both $\Phi(\psi)$ and $\Phi(\psi_\perp)$ once the two options for the original qubit, ψ and $\psi_\perp$, are revealed to him. Only then can he determine what measurements to make in order to achieve maximum probability of a distinguishing between them.

For these reasons it is preferable to constrain Bob as described in the Sec. 3.1. Doing so gives us a more 'operational' meaning to integrity, and allows us to make direct links to other related concepts in the field.

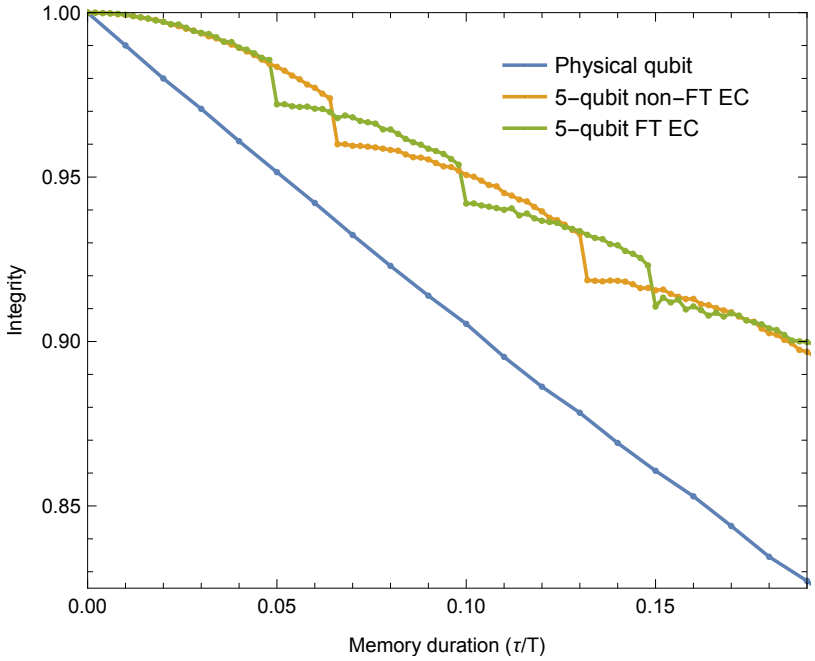


Figure 3.9: **Comparing memories employing FT verus a non-FT error correction.** In this figure we plot the 'integrity at interruption' to look inside a memory process, as discussed earlier for Figs. 3.2 and 3.5. We compare three memory channels all of the same duration $\tau = 0.2\,T$. The blue line is our standard reference, the single-qubit memory. The other two are based on the five-qubit code, with the error rate of all the gates involved in the error correction process to be 0.1%. The data shown in orange are for the memory channel protected by Igor using a non-FT error correction, and the optimal number of such cycles is 2. The data shown in green is for a more sophisticated Igor using a Shor-style fault tolerant circuit shown in Fig. C.6(a). It is interesting to note that the optimal number of error correction cycles is now 3. However the overall performance is near-identical (i.e. lines are very close on the far right).

3.6 Case where fault tolerance is beneficial

Fig. 3.3(b) shows the performance of a high quality memory channel using the five-qubit code and performing n cycles of error correction, equispaced over the duration, with a gate error rate of 0.1%. With this level of fidelity we find that the memory channel meets the most demanding of our milestones, *M4: Strictly superior encoded memory*. In this case analysed previously Igor used a non-fault tolerant error correction cycle; however from the earlier Fig. 3.6 one would expect that a fault tolerant Igor using a Shor-style syndrome extraction might lead to an *even* more highly performing memory channel.

In fact one finds that the memory using fault-tolerant correction is indeed supe-

rior, albeit just slightly. An interesting point is that the optimal number of error correction cycles is *higher*, for a given channel duration, when one employs fault tolerant correction versus the naive circuit. This makes intuitive sense: when gate errors are as low as 0.1% the fault tolerant error correction circuits work well and introduce less noise than the naive circuits (c.f. Fig. 3.6), so that we will see smaller step-like deteriorations in the quantity we call 'integrity at interruption' implying that they can be used more frequently. This is shown in Fig. 3.9 where we contrast a fault-tolerant and non-fault-tolerant channels of duration $\tau = 0.2\,T$. In the published paper [98], a similar figure demonstrates that using the 2-qubit fault-tolerant syndrome extraction of Chao and Reichardt [36] has a higher number of optimal error correction cycles (4 as opposed to 3) and outperforms non-fault-tolerant error correction by a greater degree (albeit still a very small amount). Again this is to be expected as the circuit is even simpler than the Shor-style fault-tolerant circuit so the step-like deterioration can be smaller.

3.7 Conclusion

This chapter (and the previous discussion of the orbital probe computer) have focussed on the storage of qubits in a fault-tolerant quantum memory and the steps and milestones towards achieving this at both large and then small scale. But as discussed earlier, in order to actually compute in a fashion that is beyond the reach of classical computer then we will need to face the problem that our most well-understood and high-threshold codes cannot directly support a universal set of operations. To that end, the following chapters investigate the use of magic states to supplement error correction to achieve universal quantum computation. Particular emphasis is placed on the overhead required to convert a Clifford quantum computer, or simple quantum memory such as the silicon based surface code described in Chapter 2, into a universal machine capable of approximating any quantum algorithm.

Chapter 4

Magic state distillation protocols

Sections 1-2 of this Chapter review and introduce some of the key concepts in magic state distillation and set out the protocols that will be further investigated in a common formalism. In Sections 3-4 we set out the construction of a magic state factory explicitly and introduce some jargon and notation. Sections 5-7 set out original analytic and numerical work on a more accurate assessment of the global error rate in the output of magic state factories, extending previous results. The author performed all numerical simulations presented. The original research in this chapter was published in Physical Review A [118]. Note that the text of Sections 3-7 of this chapter is adapted from the text the author wrote for Ref. [118]

As we have seen fault-tolerant quantum computing involves a host of resource overheads, entering at different stages of the process. The most widely known cost is that of encoding a logical qubit into many physical qubits, which provides safe storage of quantum information. We have already explored in some detail a specific example of how this might work in practice. However, once encoded, a logical qubit only natively supports a limited set of fault-tolerant operations [38]. For high-threshold codes, such as the surface code, the native operations belong within the Clifford group (and may not even generate the full Clifford group) and a method of achieving non-Clifford operations is required. In the following chapters we investigate the methods and associated overheads involved in one particular method of achieving non-Clifford operations efficiently: magic state distillation.

To briefly recap the idea that we first met in Sec. 1.3: We can know as a result of the Solovay-Kitaev theorem [37] that if the Clifford group is augmented by a non-Clifford 'T' gate then this set of operations can efficiently approximate any required quantum algorithm. Thus a machine which can perform both the operations in the Clifford group and the T gate can be said to be universal. These T gates can

in turn be fault-tolerantly performed by using state injection of high-fidelity magic states, prepared by distillation [41]. However, the Solovay-Kitaev theorem requires a huge number of T gates to approximate an arbitrary gate, and early magic state distillation protocols put a high price on each one. The last decade has seen substantial advances in both these areas. Magic state distillation is now possible at better rates [41, 119–122], reducing the expected number of noisy magic states per high quality T-gate. The number of T-gates needed to approximate some unitary, the so-called T-count, has also been significantly reduced after the discovery of new gate synthesis techniques [123–127]. Nevertheless, fault-tolerant quantum computing remains a monumental challenge, so resource savings, wherever they can be found, are essential. One suggested route is to circumvent magic states altogether, for instance by using gauge-fixing [128, 129] of subsystem codes like the 3D colour code [130, 131]. All current indications are that colour codes have a much lower error correction threshold [132] than the toric code [133–136]. For now, such low-noise levels appear beyond technological reach, ruling out gauge-fixing in the near-term.

In this chapter we explore some of the abstract characteristics of magic state distillation techniques, showing in particular one way in which the error rates associated with a certain class of protocols have been systematically overstated. This chapter treats the process of magic state distillation largely as a black box, in particular we do not discuss the underlying mechanism used for error correction, instead focussing on the magic state distillation in isolation. In the following chapter we take a more practical look at the implementation of these magic state distillation procedures, quantifying and improving the resources associated with the realisation of some of the most promising protocols. We end with an estimate of the size and operation speed one could expect of a 'useful' universal quantum computer given our current best guess at how these could be constructed: that is as a combination of surface code and MSD.

4.1 Universal quantum computation with magic states

For now we explore a little further the need for magic states and some of the key results upon which the construction of magic state factories will rely. A brief introduction was already given in Chapter 1.3 and here we will discuss the original conception of magic state distillation, motivated with particular reference to three key theorems.

4.1.1 The Clifford hierarchy and the magic state model

Let us introduce the Clifford hierarchy and the magic state model. The well known Pauli group $\mathcal{P}$ (which we met in Chapter 1) is the group composed of tensor products of the single qubit Pauli operators. Unitaries mapping the Pauli group to itself are called Clifford unitaries, which again form a group $\mathcal{C} := \{U|UPU^\dagger \in \mathcal{P}, \forall P \in \mathcal{P}\}$. Clifford operations are the physical operations composed of Clifford unitaries, measurement of Pauli operators and preparation of their eigenstates (the stabiliser states), and classical feedforward. On a single qubit, the Clifford operations allow one to reach any state within the octahedron bounded by the Pauli eigenstates, see Figure 4.1(a). The magic state model of quantum computation [41] assumes Clifford operations are free resources that can be implemented perfectly, and proceeds to evaluate the cost of non-Clifford operations. Such a model is suitable for logical qubits where the Clifford operations are fault-tolerantly protected, as is common.

Unitaries outside the Clifford group can fall into other levels of the Clifford hierarchy, defined recursively as

$$\mathcal{C}_\ell := \{U|UPU^\dagger \in \mathcal{P}, \forall P \in \mathcal{C}_{\ell-1}\}, \tag{4.1}$$

where $\mathcal{C}_1 := \mathcal{P}$. It follows that the Clifford group is the second level, and all higher levels include non-Clifford gates. We will soon see the Clifford hierarchy plays an important operational role in a teleportation process known as state injection [137].

The **Gottesman-Knill theorem** proves that one can efficiently simulate any quantum circuit which only uses operations from the Clifford group. This is possible by tracking the generators of the stabiliser group. Every operation in the Clifford group maps the stabiliser generators to other stabiliser generators, which are Pauli operators, and this means such a quantum circuit has an efficient representation using stabilisers. This is even the case in the presence of Pauli errors which can be similarly accounted for by updating the stabiliser generators in the classical tracking software. Clearly if a family of quantum circuits can be efficiently classical simulated they are not providing the ability to perform universal quantum computation – unless, of course, it proves to be the case that all quantum computing can be classically simulated!

To enable universal quantum computation the addition of some non-Clifford gate must therefore be necessary, in fact any non-Clifford unitary will be sufficient. One common gate to consider is the T gate, which when combined with the Clifford group

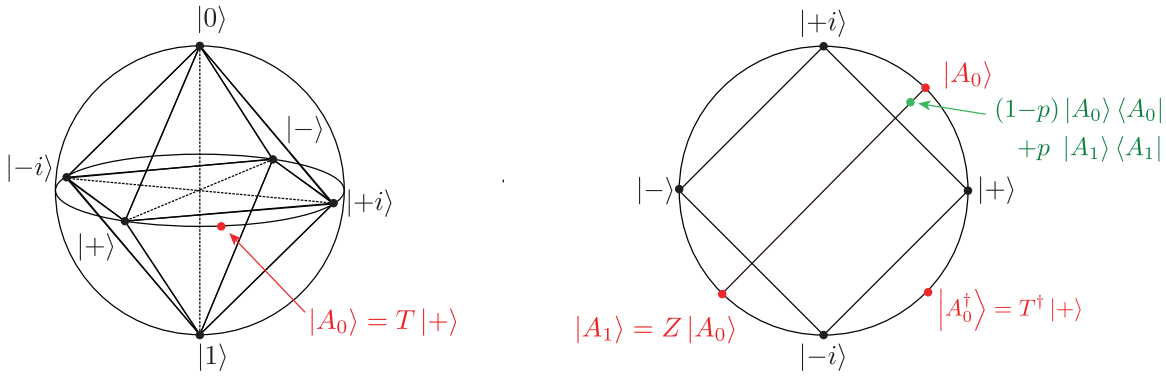


Figure 4.1: (a) The Bloch sphere and Clifford octahedron. The ability to utilise the Clifford operations allows the preparation of any state with the octahedron bounded by the eigenstates of the Pauli operators. Any operations which only map states within this octahedron to other states within the octahedron can be simulated classically. The addition of any prepared *pure* state outside this octahedron can allow universal quantum computation. Magic state distillation allows for some mixed states outside this octahedron to be distilled to higher fidelity using only the Clifford operations. (b) The A magic states. This x-y plane of the Bloch sphere is labelled with some key states for our description of magic state distillation.

provides an overcomplete universal set of gates. The T gate

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix} \tag{4.2}$$

is a member of the third level of the Clifford hierarchy, $\mathcal{C}(3)$. It is often also called the $\pi/8$ rotation because $T \sim \mathrm{diag}\{\exp(-i\pi/8), \exp(i\pi/8)\}$ up to a global phase.

It was been shown by **Eastin and Knill** [38] that that if a quantum code can detect at least any error on a single qubit then it will not have a transversal universal set of gates. A similar result from Zeng *et al* [39] is that a qubit stabiliser code does not allow for a universal set of transversal unitary gates. One often finds that for interesting codes, notably including the surface code, the protected set of gates fall within the Clifford group. The fact that the protected set of gates for the surface code fall within the Clifford group is a manifestation of a third result that will motivate the need for magic states distillation: that of **Bravyi and Koenig**[138]. They found that for any 2D stabiliser code the logical gates that can be performed using only local gates are members of the Clifford group. More generally for an n-dimensional code, the possible local gates fall in the $(n-1)^{\text{th}}$ level of the Clifford hierarchy. This statement does not refer solely to transversal gates but any constant depth circuit – which is a natural extension of the notion of a transversal gate.

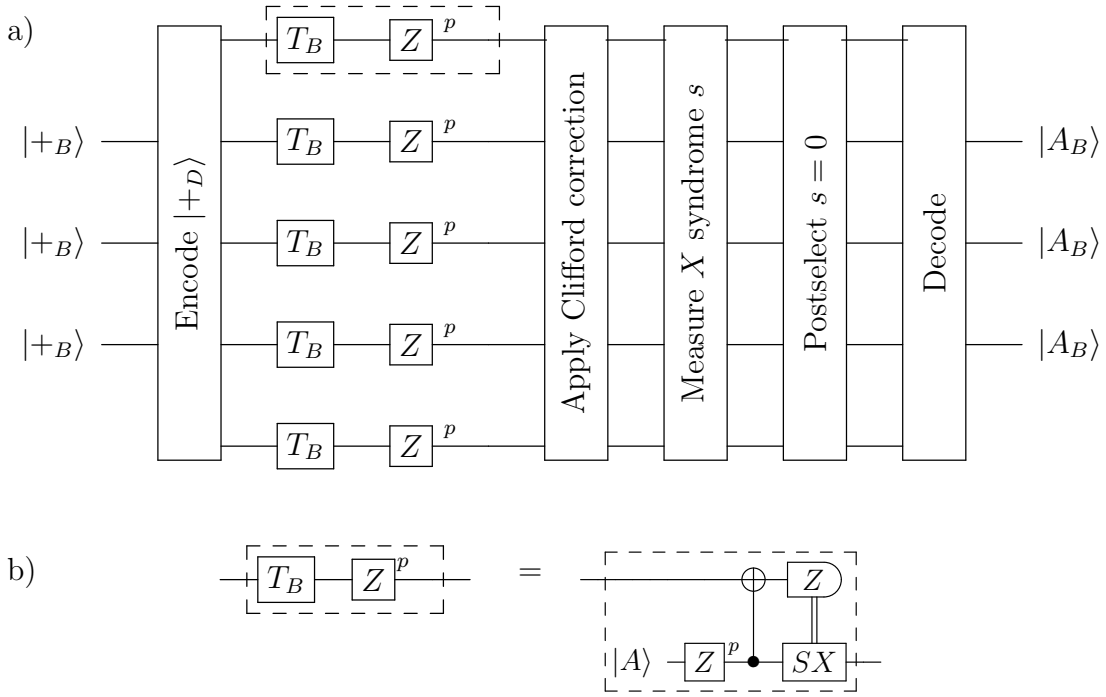


Figure 4.2: The distillation subroutine (a) for a magic state $|A\rangle$ based on a distillation code $[[n_D, d_D, k_D]]$. The encoder prepares k_D copies of the state $|+_B\rangle$. Implementation of each T gate consumes one $|A\rangle$ as in (b). If the ancilla has error rate p, each T gate is followed by a Z error with probability p. The Clifford correction is particular to the choice of code. A non-destructive measurement of the X checks is made and if the trivial syndrome reported the decoder is applied.

Given then that we need a non-Clifford gate to supplement our protected operations the question remains how we can do this is in a fault-tolerant manner. We know that the circuit given in Figure 4.2(b) (or Fig. 1.3) would allow us to teleport a T gate into the computation using Clifford operations, if only we were given a supply of pure states such as $|A\rangle = T|+\rangle \propto |0\rangle + \exp(i\pi/4)|1\rangle$. As Figure 4.1(b) shows, this state lies outside the Clifford octahedron. Given a Clifford computer we need to add only the supply of a non-Clifford pure states to complete a universal machine. However we know we cannot prepare such states in many stabiliser codes directly in a fault-tolerant manner given that we have only the Clifford operations. It turns out the $|A\rangle$ state is magic state: a pure non-stabiliser states which can be distilled from certain (but not all [139]) mixed non-stabiliser states using only Clifford operations. Therefore it is possible to achieve universal quantum computation with perfect Clifford operations and noisy (mixed) magic states!

4.1.2 Magic state distillation

Magic state distillation is the technique that allows us to turn a supply of noisy magic states into a smaller number of high quality of magic states, thus allowing us to teleport gates from higher in the Clifford hierarchy into our Clifford computer. In this and the following chapter we will further investigate the resources and practical implementation of 'magic states' from $\mathcal{C}(3)$, that is in particular $|A\rangle$ and $|\text{Tof}\rangle$, the state which allow the injection of T and Toffoli gates respectively. Gates from higher levels of the Clifford hierarchy will be discussed further in Chapter 6.

A number of protocols for magic state distillation have been proposed, but all follow the same basic format [140]. They entail projecting several copies of the initial state onto a stabilizer codespace, that of a code which has the desired gate as a transversal operation, and then decoding from the codespace onto a smaller number of qubits.

We will describe, in broad strokes, a distillation subroutine for a single qubit $|A\rangle$ magic state. The protocol takes n copies of a *mixed* state ρ such that $\langle A|\rho|A\rangle = 1-p$ where p is the input error rate. We note that we can always prepare the noisy initial magic state such that it is diagonal in the A basis by applying the operators I and $A \equiv TXT^\dagger$ with probability $1/2$ to each copy of the the input state (which we are free to do as these are both Clifford operations). This leaves a state with only a Z error on it, an error which will anti-commute with at least one of the all X stabilisers, allowing detection. This is form of twirling, a concept which we met earlier in Chapter 2.

We assume the code $[[n_B, k_B, d_B]]$ or *base code* is used for the low level error correction and fault-tolerance (i.e. the perfect Clifford operations) and that a *distillation code* $[[n_D, k_D, d_D]]$ is used to provide the magic state distillation. In the following $|\psi_B\rangle$ is used to specify a logical state in the base code, and $|\psi_D\rangle$ the logical states of the distillation code (remembering that the 'physical qubits' of the code are logical qubits in the base code, i.e. this is a concatenation of the code). The circuit describing this process is shown in Fig.4.2.

1a Prepare k_D copies of $|+_B\rangle$. This process is fault-tolerant.

1b Encode them into $|+_B\rangle^{\otimes k_D}$ of the distillation code, this is also assumed to be fault-tolerant due to the error suppression provided by the base code.

2 Prepare n_D copies of the $|A_B\rangle$ state. These will be noisy at a rate comparable to the noise on the worst physical gate operation.

3 Use the $|A_B\rangle$ states to inject the T_B gate onto every qubit of the distillation code to yield $|A_D\rangle^{\otimes k}$. This is achieved because the distillation code supports transversal T gates and so $T_B^{\otimes n_D}|+_D\rangle^{\otimes k_D} = T_D^{\otimes k_D}|+_D\rangle^{\otimes k_D} = |A_D\rangle^{\otimes k}$.

4 Measure the stabilisers of the distillation code to check for errors caused by an injection of a noisy T_B gate. Discard if error detected.

5 Decode distillation code back onto base code. To leave $|A_B\rangle^{\otimes k_D}$

Thus we have taken n_D noisy magic states (encoded in the base code), used these to perform a transversal T gate in the distillation code, detected errors and postselected on success and decoded back onto the base code. Thus we are left with k_D magic states in the base code, with reduced error.

For the remainder of the chapter we will investigate the black box workings of a distillation procedure, deferring questions of their resource requirements to the following chapter. This level of abstraction suffices to allow us to determine the output error of magic states from a factory and to investigate the effect of different levels of post-selection on the global error rate of the final states. As such we will not yet delve into the circuits and other details needed to fully implement the process.

4.2 Formal tools

The notion of a G-matrix to describe a quantum code was introduced by Bravyi and Haah in [120]. They showed that if this matrix had certain properties then one could construct codes with a transversal T gate. The T and Toffoli gates are in the third level of the Clifford hierachy, which we call $C(3)$. The structure of the Clifford hierarchy will be described in Chaper 6 when we investigates magic states for smaller angle rotations than the T gates. While previously distillation procedures were described in a variety of ways, throughout the remainder of this thesis we will present all the $C(3)$ distillation codes that we investigate in this G-matrix formalism. Apart from allowing greater clarity this allows us to more easily determine the relationships between rounds of distillation that use different codes and to construct the circuit-level descriptions of each protocol.

4.2.1 G-matrices

The G matrix is a binary matrix composed of two submatrices G_1 and G_0.

$$G = \left(\frac{G_1}{G_0}\right) = \left(\begin{array}{c} g_m \\ \vdots \\ \hline g_{b+1} \\ g_b \\ \hline \vdots \\ g_1 \end{array}\right) \tag{4.3}$$

We label the rows of G as $\{g_1, \ldots, g_b, g_{b+1}, \ldots g_m\}$ where the rows $\{g_{b+1}, \ldots, g_m\}$ belong to G_1 and the rows $\{g_1, \ldots, g_b\}$ belong to G_0. We will also introduce some further notation here. If g is a binary vector of length n, we will use $Z[g] = \otimes^n_{j:[g]_j=1} Z_j$ and similarly $X[g] = \otimes^n_{j:[g]_j=1} X_j$. For example, $Z[\{0,1,0,0,1\}] = \mathbb{1} \otimes Z \otimes \mathbb{1} \otimes \mathbb{1} \otimes Z$.

The G-matrices can define a CSS code as follows. The rows of G_0 are some set $\{g_1, \ldots, g_b\}$ that specify the X-type stabiliser generators $\{X[g_1], \ldots, X[g_b]\}$. The Z-type generators can then be chosen in the following way. Denote $\mathcal{G}^\perp$ to be the binary vector space orthogonal to both G_0 and G_1, that is $\mathcal{G}^\perp := \{g : (f,g) = 0; \forall f \in G_0, G_1\}$ where we use the inner product $(f,g) := f \cdot g \pmod 2$. Note that Bravyi-Haah required that $G_0 \subset \mathcal{G}^\perp$. Any operator $Z[f]$ will then be guaranteed to commute with the previously defined X-type stabilisers. We define $G^\perp$ to be some (minimal) matrix with rows $\{f_1, \ldots f_a\}$ that generate the group $\mathcal{G}^\perp$ under row-wise modular addition. Thus this defines the Z-type generators $\{Z[f_1], \ldots, Z[f_b]\}$. Note that there exist

many different choices for $G^{\perp}$, which all result in the same CSS code with Z-type generators $\{Z[f_1], \ldots, Z[f_a]\}$.

That completes the description of the stabiliser codespace, though we also need to know how information is stored within the subspace. We have that the X operator for the k^{th} logical qubit is $X[g_k]$ where g_k is a row of G_1, and so $b+1 \leq k \leq m$. These are representatives of the logical operators, with equivalent logical operators differing by only a stabiliser. For Bravyi-Haah protocols all rows in G_1 have odd weight and all the G_0 rows have even weight. This allows us to take that the Z operator for the k^{th} logical qubit as $Z[g_k]$ where g_k is the k^{th} row of G_1. This achieves the correct commutation relations between logical X and Z operators and these logical operators all commute with stabilisers of the code given that $G_0 \subset \mathcal{G}^{\perp}$.

So a G matrix represents a CSS code.

$$G = \left(\begin{array}{c} G_1 \\ \hline G_0 \end{array}\right) \begin{array}{l} \}\text{logical } X \text{ operators} \\ \}X\text{-stabilisers} \end{array} \tag{4.4}$$

with $G^{\perp}$ representing the Z-stabilisers generators.

For example, the seven qubit Steane code expressed in this manner would be:

$$G_{\text{Steane}} = \left(\begin{array}{ccccccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{array}\right). \tag{4.5}$$

4.2.2 Triorthogonality and transversal T

Now let us introduce the idea of biorthogonal and triorthogonal matrices, again as outlined by Bravyi and Haah. We will make extensive use of the the symbol $u \circ v$ to denote the element-wise products of vectors u and v such that it has the elements

$$[u \circ v]_j = u_j v_j \tag{4.6}$$

which generalises to an arbitrary number of vectors, e.g.

$$[u \circ v \circ w]_j = u_j v_j w_j.$$

Furthermore we use $|\ldots|$ to denote the weight of a vector so that

$$|u| = \sum_j u_j$$

Now let us define bi- and tri-orthogonality.

Definition 1 *A binary matrix G is biorthogonal if and only if the set of row vectors $v \in G$ are linearly independent and for all distinct pairs $u \neq v$*

$$|u \circ v| = 0 \pmod 2.$$

Furthermore,

Definition 2 *A binary matrix G is triorthogonal if and only if it is both biorthogonal and for all $u \neq v \neq w \neq u$ we have*

$$|u \circ v \circ w| = 0 \pmod 2$$

Let us take G to represent a CSS code, as described before. The submatrix G_0 defines the code's X stabilisers and G_1 identifies the k logical X operators. The whole codespace is spanned by vectors $|v\rangle$ where $v \in \text{span}\{G\}$. The row span of G generates a vector space $\mathcal{G}$ and similarly we use $\mathcal{G}_0$ for the vector space spanned by G_0. The elementary "all-zero" logical state is

$$|\{0,\ldots,0\}_L\rangle = \frac{1}{\sqrt{|G_0|}} \sum_{f \in \mathcal{G}_0} |f\rangle \tag{4.7}$$

If the rows of G_0 are linearly independent we can also write this as

$$|\{0,\ldots,0\}_L\rangle = \frac{1}{\sqrt{|G_0|}} \sum_{y \in \mathbb{Z}_2^k} \left|G_0^T y\right\rangle \tag{4.8}$$

For a logical state $|x_L\rangle$ where $x = \{x_1, x_2, \ldots, x_k\} \in \mathbb{Z}_2^k$ (the set of binary vectors of length k) we have

$$\begin{aligned} |x_L\rangle &= \frac{1}{\sqrt{|\mathcal{G}_0|}} \sum_{y \in \mathbb{Z}_2^s} \left|G_0^T y + G_1^T x\right\rangle \\ &= \frac{1}{\sqrt{|\mathcal{G}_0|}} \sum_{y \in \mathbb{Z}_2^s} \left|G^T(x,y)\right\rangle \end{aligned} \tag{4.9}$$

where (x, y) is the composite vector $\{x_1, \ldots, x_k, y_1, \ldots, y_s\}$. Observe that $G^T(x,y) \in \mathcal{G}$ and that all logical states are orthogonal if the rows of G are linearly independent. A biorthogonal code will possess a transversal gate

$$S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \tag{4.10}$$

though we will not give the details here, rather we will comment on the triorthogonality and transversality of the non-Clifford gate

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}. \tag{4.11}$$

Consider $T^{\otimes n}$ applied to a term in the codespace. It is useful to define $z = (x, y)$ so

$$T^{\otimes n}\big|G^T z\big\rangle = \exp\left(i\frac{\pi}{4}|G^T z|\right)\big|G^T z\big\rangle. \tag{4.12}$$

To simplify the exponent we note that $G^T z$ is computed in mod 2 arithmetic and has the form

$$|G^T z| = \sum_a g_j z_j \pmod 2 \tag{4.13}$$

where g_a denotes the row vectors of G. Whilst $G^T z$ is computed modulo 2, the exponent is only modulo 8. We can now make use of a nice trick. Notice that $a_1 \oplus a_2 = a_1 + a_2 - 2a_1a_2$, where we use $\oplus$ to show modular arithmetic in the left hand side. For a trio of bits $a_1 \oplus a_2 \oplus a_3 = a_1 + a_2 + a_3 - 2a_1a_2 - 2a_1a_3 - 2a_2a_3 + 4a_1a_2a_3$. In general we have

$$\bigoplus a_j = \sum_j a_j - 2\sum_{j,k<j} a_j a_k + 4\sum_{j,k<j,m<m,k} a_j a_k a_m \ldots, \tag{4.14}$$

where dots show there are further terms with higher order products of bits. For the products of r bits the term is always multiplied by $(-2)^{r-1}$ and so for $r > 3$ they are a multiple of 8. Thus, when we apply this relation to the exponent of Eq. 4.12, which can be evaluated modulo 8 the higher products disappear. If we extend this insight to vectors we find we can rewrite

$$|G^T z| = \sum_j |g_j| - 2\sum_{j,k<j} |g_j \circ g_k| + 4\sum_{j,k<j,m<k} |g_j \circ g_k \circ g_m| \tag{4.15}$$

Now the significance of triorthogonality becomes apparent. For a triorthogonal code the triple terms are always even weight and so when multiplied by the prefactor of 4 will always vanish in modulo 8 arithmetic leaving

$$|G^T z| = \sum_j |g_j| - 2\sum_{j,k<j} |g_j \circ g_k| \tag{4.16}$$

Even though in a triorthogonal code the $|g_j \circ g_k|$ are even weight the factor of 2 means they will not necessarily vanish in modulo 8 arithmetic. However, Bravyi and Haah [120] show that a Clifford unitary can counteract these terms, leaving only

$$|G^T z| = \sum_j |g_j|. \tag{4.17}$$

For the first k rows of G these weights are odd and the rest are even and so we can write $|g_j| = x_j + 2\Gamma_j$ for $j < k$ and $|g_j| = 2\Gamma_j$ otherwise, and where Γ_j are integers. Hence

$$|G^T z| = \sum_{1 \leq j \leq k} x_j + \sum_j 2\Gamma_j = |x| + 2\sum_j \Gamma_j. \tag{4.18}$$

Again the second term can be cancelled by an appropriate Clifford unitary. Therefore there exists a Clifford unitary such that

$$CT^{\otimes n}\big|G^T z\big\rangle = \exp\left(i\frac{\pi}{4}|x|\right)\big|G^T z\big\rangle \tag{4.19}$$

4.2.3 The Bravyi-Haah codes

Triorthogonality and the concepts introduced above give us sufficient criteria for generating codes with transversal T gates which can then be used in magic state distillation. In the paper introducing triorthogonality and G-matrices Bravyi and Haah introduced a family of codes, which we will now refer to as the Bravyi-Haah codes or *block codes* to achieve magic state distillation. We choose to investigate these codes as they are among the most efficient [120], that is the average number of noisy magic states required to produce a distilled one is low according to a 'cost' metric that we will revisit in 4.6. We now outline the construction of the G-matrices representing these codes, for which it can be verified that the triorthogonality condition is satisfied.

The k qubit Bravyi-Haah codes are given by the matrices $G_{\mathrm{BH}}(k)$. For each even number $k \geq 2$ define the matrix as

$$G_{\mathrm{BH}}(k) = \left(\begin{array}{ccccccccc} 0 & 1 & 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \ddots & \vdots & \vdots \\ 0 & 1 & 0 & 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & \cdots & 0 & 1 \\ \hline L & L & R & R & R & R & \cdots & R & R \end{array}\right) \tag{4.20}$$

where here 0 and 1 are constant vectors $0 = (0, 0, 0, \cdots, 0)$ of the necessary width and

$$L = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}, \; R = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}. \tag{4.21}$$

R appears k times in the matrix $G^{\mathrm{BH}}(k)$ which has $3k+8$ columns and $k+3$ rows. It can be verified that this matrix describes a distance 2 code. Thus the Bravyi-Haah codes take $3k+8$ noisy $|A\rangle$ states with errors ϵ and output k states with error $O(\epsilon^2)$.

4.2.4 The 15 qubit Reed-Muller code

The original magic state distillation code was based on a 15 qubit punctured Reed-Muller code and is described in [41]. The G matrix describing this code is

$$G_{\mathrm{RM}} = \left(\begin{array}{cccccccccccccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{array}\right) \tag{4.22}$$

where it is easy to verify that this satisfies the triothogonality condition. This code is distance 3 so take 15 noisy $|A\rangle$ state to 1 distilled state with error $O(\epsilon^3)$. We will refer to this simply as the Reed-Muller code from now.

4.2.5 'T to Toffoli' code

We also consider distillation of 3-qubit Toffoli states, which act as resources for implementing Toffoli gates. One suitable definition of Toffoli states is $\frac{1}{\sqrt{8}}\sum_{a,b,c}(-1)^{abc}|a,b,c\rangle$, though for technical reasons it more elegant to use the Clifford equivalent state

$$|\mathrm{Tof}\rangle = \frac{1}{\sqrt{8}}\sum_{a,b,c}(-1)^{(a+1)(b+1)(c+1)}|a,b,c\rangle. \tag{4.23}$$

A basis of orthogonal Toffoli states is again formed using Z noise, so that

$$|\mathrm{Tof}_{x,x',x''}\rangle = (Z^x \otimes Z^{x'} \otimes Z^{x''})|\mathrm{Tof}\rangle \tag{4.24}$$

which we again simply represent by $x = \{x, x', x''\}$. We use subscripts to label different Toffoli states with the ordering $x = \{x_1, x_2, \ldots x_n, x'_1, x'_2, \ldots x'_n, x''_1, x''_2, \ldots x''_n\}$. There are two types of distillation protocol available to Toffoli states.

First, one can distill a Toffoli state from more noisy Toffoli states. Paetznick and Reichardt [141] showed that by taking a Bravyi-Haah protocol for $|A\rangle$ distillation, one can use 3 parallel copies so that $G_0^{\mathrm{PR}} = (G_0 \otimes \mathbb{1}_3)$ and $G_1^{\mathrm{PR}} = (G_1 \otimes \mathbb{1}_3)$, where $\mathbb{1}_3$

is a 3-by-3 identity matrix. We have used the superscript PR to denote that these are the relevant matrices for the PR (Paetznick and Reichardt) protocol, and using the PR matrices in equations (4.32) and (4.31) again describes the performance of the block. Since the protocol is essentially Brayvi-Haah applied to Toffoli states, which will often refer to this protocol as Brayvi-Haah.

There also exist protocols that convert between species of magic states, providing some error suppression in the process. Specifically, one can take 8 $|A\rangle$ magic states and outputs 1 $|\text{Tof}\rangle$ state, detecting all single qubit Z errors [142, 143]. We call this the Toff protocol and observe that these protocols can also be described in the same formalism by using

$$G_{\text{Toff}} = \left(\begin{array}{cccccccc} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ \hline 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right). \tag{4.25}$$

The G_{Toff} matrices do not satisfy the usual triorthogonality conditions, since they do not distill T states into T states. Rather they satisfy a very different set of conditions reflecting the fact that the protocols convert species of magic states as we now discuss. Also notice that the rows of G_1 are even weight so only describe the X logical operators, whereas for Bravyi-Haah and Reed-Muller the rows $g \in G_0$ were odd weight so could be used to describe logical X and Z operators $X[g]$ and $Z[g]$.

To provide a deeper understanding of why this matrix describes a code that converts between magic states, we deduce the weights of the codewords from properties of the G matrix. Labelling the rows of G_1 as g_1, g_2, g_3 and the row of G_0 as g_0, we have that each term is of the form

$$|(a,b,c)_L\rangle \propto |ag_1 \oplus bg_2 \oplus cg_3\rangle + |ag_1 \oplus bg_2 \oplus cg_3 \oplus g_0\rangle \tag{4.26}$$

Looking at the first term and converting $\oplus$ to $+$ we have

$$\begin{aligned} ag_1 \oplus bg_2 \oplus cg_3 = {} & ag_1 + bg_2 + cg_3 \\ & - 2[ab(g_1 \circ g_2) + bc(g_2 \circ g_3) + ac(g_1 \circ g_3)] \\ & + 4abc(g_1 \circ g_2 \circ g_3) \end{aligned}$$

where $\circ$ denotes elementwise multiplication. Taking the weight and using that

$$|g_j| = 4, j \neq 0 \tag{4.27}$$

$$|g_j \circ g_k| = 2, j \neq 0, k \neq 0, k \neq j \tag{4.28}$$

$$|g_j \circ g_k \circ g_q| = 2, j \neq 0, k \neq 0, q \neq 0, k \neq j \neq q. \tag{4.29}$$

we have

$$|ag_1 \oplus bg_2 \oplus cg_3| = 4(a+b+c) \\ - 4(ab+bc+ac) + 4abc$$

Clearly this weight is always a multiple of 4. However, one can also see that it only vanishes modulo 8 whenever $a = b = c = 0$. Therefore the code has a transversal U_{Tof} gate for the following reason. Applying $T^{\otimes 8}$ to a codeword means it will pick up a phase $e^{iW2\pi/8}$ where W is the weight of the codeword. For the $|(0,0,0)_L\rangle$ this phase is +1, but for all other codewords the phase is -1 up to a global phase. This is exactly the action we want of a Toffoli gate!

To complete the description explicitly, the Z logical operators are represented by $Z[f]$ where $f \in F$

$$F = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}. \tag{4.30}$$

4.3 Overview of magic state factories

Building on the literature outlined in the previous sections, we now turn to the original research of the author into the output error rates of the MSD protocols. Here we focus on the the global output error rate ϵ_{g} of the protocols: that is the probability that there is any error on the qubits output by a round of magic state distillation. In other words the completely error-free output is achieved with probability $1 - \epsilon_{\text{g}}$. As we will see, by taking an estimate of ϵ_{g} from the average error rate on single qubits of the multi-qubit output state, ϵ_{g} has been previously overestimated.

We will often refer to 'magic state factories' in our conception of the likely design of a quantum computer. By this we are acknowledging that it is likely that our computer will be separated into distinct layers with different functions, much like the design of classical machines. As we will see a large proportion of a quantum computer's resources will be given over to a section of the machine whose sole job is produce high fidelity magic states to be used by the information processing unit of the quantum computer. Under a magic state model of quantum computation, the need for such a factory is independent of the problem which is to be solved. So the design and optimisation of this portion of the computer is crucial to any future device.

We show that magic state factories can make more use of the block protocols than previously thought by using new techniques to reduce and calculate the global output error. Consider a protocol outputting K qubits with multi-qubit global error rate ϵ_g. All previous studies have used the union bound, also called Boole's inequality, to assert $\epsilon_g \leq K\epsilon$ where K is the number of magic states and ϵ is the average error rate of single-qubit outputs, obtained by tracing the qubit from the multi-qubit state. For an uncorrelated product state, $\rho_g = \rho^{\otimes K}$, we have $\epsilon_g = 1-(1-\epsilon)^K$ and so to leading order $\epsilon_g = K\epsilon - O(\epsilon^2)$, so the union bound is a good estimate for small ϵ. However, the output of block protocols can be highly correlated. Boole's inequality holds for correlated states, but ϵ_g can be much smaller than $K\epsilon$. In distillation protocols such as Bravyi-Haah, correlations are produced because block protocols reduce the probability of single errors, but pairs or clusters of errors can go undetected and lead to a correlated error pair. If one error is present, it almost certainly has a partner. As such, use of the union bound has lead to systematic over estimation of noise in Bravyi-Haah and other distillation protocols. Here we track these correlations through multiple rounds of a magic state factory. Once we begin tracking correlations, it becomes apparent that quality control of the factory can be improved. Detecting an error in one part of the factory can, due to correlations, indicate a likely undetected partner error elsewhere in the factory. We introduce the notion of module checking whereby we discard magic states brought into disrepute by their correlated partners. The enhanced fidelity of module checking is both analytically estimated and found by numerical Monte Carlo simulations, with excellent agreement between these methods.

4.3.1 Blocks, branches and modules

We begin by introducing some helpful vocabulary for describing magic states factories. Efficient distillation uses $n \to k$ block protocols, that take n noisy $|A\rangle \propto |0\rangle + \exp(i\pi/4)|1\rangle$ and with some probability output k states of a higher fidelity. Such a process we call a block, and the previous section described the details of the inner working of such a block. Now we treat each block as a black box with known relations between input and output, and consider how these blocks are composed together.

Distillation protocols have many levels forming a tree-like structure with many branches that merge at points we call modules, shown in Fig. 4.3. Branches contain many qubits, which are potentially correlated. However, the inputs to block protocols must not be correlated, so each qubit in a branch must be fed into a different block. Therefore, as we enter a module, a branch of B_l qubits is split up so that each qubit enters a different block. If each block implements a $n_l \to k_l$ protocol, then the

Protocol	Input	Output	Block Failure Probability	Output error on single magic state
BH	$(3k+8)$ $\|A\rangle$ states	k $\|A\rangle$ states	$(3k+8)\epsilon + O(\epsilon^2)$	$(3k+1)\epsilon^2 + O(\epsilon^3)$
RM	15 $\|A\rangle$ states	1 $\|A\rangle$ state	$15\epsilon + O(\epsilon^2)$	$35\epsilon^3 + O(\epsilon^4)$
Toff	8 $\|A\rangle$ states	1 $\|\text{Toff}\rangle$ state	$8\epsilon + O(\epsilon^2)$	$28\epsilon^2 + O(\epsilon^3)$

Table 4.1: Summary of the performance of our distillation protocols. The prefactor which determines the leading order block failure rate is simply the number of detectable single qubit errors, which is the number of qubits used by the distillation code. The leading order output error rate is given by counting the number of undetectable of errors of a weight equal to the code distance.

whole module can be thought of taking $B_l n_l$ inputs to $B_l k_l$ outputs. This entails that $B_{l+1} = B_l k_l$ and that each module has n_l branches feeding into it. Thus a module is a collection of blocks for which the input to the module may have correlations, but the blocks which themselves compose the module have uncorrelated inputs. Initially, branches are single qubits, $B_1 = 1$, and so $B_l = \prod_{1 \leq j < l} k_j$. This module-branch structure is common to all proposals to date. Such explicit terminology has not previously been introduced but rather been left as an implicit consequence of statements about correlation avoidance. In fact such considerations are required even when building classical computers from unreliable components whose outputs are correlated, as was noted in early work by von Neumann [144].

Establishing clear vocabulary about this structure is important as we delve into the effect of postselecting at different levels on this structure. Previous protocols have considered whether individual blocks succeed or fail, we call this *block checking*. Below, we outline why it can be advantageous to postselect on the level of the modules, which are collections of blocks. We propose an additional quality check, so that the whole module is discarded whenever any of its blocks fail. We call this *module checking*.

A block will always detect a single incoming error, but might fail to detect a pair of errors. When a block detects an error, it indicates the presence of damaged branches, and since errors cluster together within branches, this increases the likelihood of errors in other blocks throughout the module. Consider when two branches fail each with a correlated error pair, the first branch sends damaged qubits to blocks 1 and 2, and the second branch sends damaged qubits to blocks 2 and 3. Since blocks 1 and 3 each received a single erroneous qubit, they will detect them. But block 2 receives a pair of errors, so they may go undetected. A simplified illustration of this process is shown in Fig. 4.3. Module checks improve fidelity by preventing these processes from

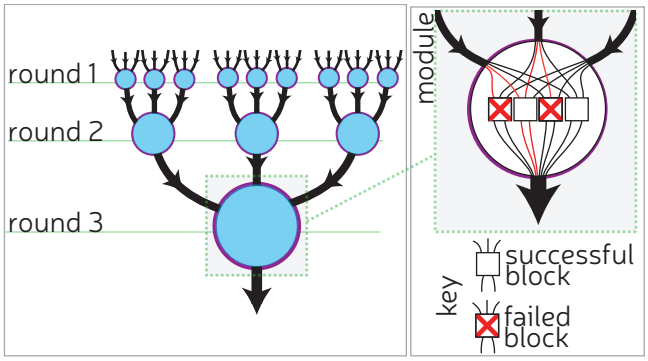


Figure 4.3: The a tree structure of many rounds of distillation, with branches (directed black lines) that merge at branching points that we call modules. The thickness of the branch increases with each round. The figure shows a fictitious scheme where $n_l = 3$ and $k_l = 2$ for all rounds. (inset) The structure within a module. Incoming branches contain many qubits, here this is shown to be 4. These qubits undergo a permutation σ and are feed into an instance of a block of a distillation protocol shown as a square. Here the 3 incoming branches carry 4 qubits and so we need 4 instances of a $3 \to 2$ protocol. A fictitious protocol has been used to keep the numbers low enough to illustrate clearly. Each of the 4 blocks output 2 qubits and these are merged into a branch of 8 qubits which feeds into a later module. A pernicious error pattern is shown in red, which is detected in 2 of distillation blocks, marked with crosses, but goes undetected in a third block.

degrading the output fidelity. Even with module checks, it is possible for a pair of corrupted branches to go undetected, but both branches must carry exactly the same pattern of errors, which is a very rare occurrence.

4.4 A black box description of block code performance

To give a pragmatic account of the performance of block codes we can focus on the properties of G_0 and G_1. As $G^\perp$ is only necessary to determine the Z stabilisers, in the abstract setting where we are only interested in the error out and success probability of the blocks, we can get by without explicit knowledge of the form of $G^\perp$. Due to the twirling of the input magic states there are only Z errors to correct which will commute with the Z-stabilisers. Of course to implement magic state distillation in reality, as outlined already, we will need to know $G^\perp$, but we defer explicit calculation of these matrices until Chapter 5 where we look at realistic implementation of magic state distillation.

We use $|A_0\rangle \propto |0\rangle + e^{i\pi/4}|1\rangle$ for a magic state and $|A_1\rangle = Z|A_0\rangle$ for the orthogonal

state with a Z error. A pure multi-qubit state $|A_{x_1}\rangle|A_{x_2}\rangle\dots|A_{x_n}\rangle$, we concisely represent with the vector $x = \{x_1, x_2, \dots x_n\}$. If we apply a block protocol to state x, the block succeeds (detecting no errors) if $G_0 x = 0 \pmod 2$ where the arithmetic is performed modulo 2. When successful, the block outputs a state $y = G_1 x$. Noisy magic states will be some probabilistic ensemble over x, with probability $\Pr(x)$. The protocol will detect no errors and output state $|A_{y_1}\rangle|A_{y_2}\rangle\dots|A_{y_k}\rangle$ with (unnormalised) probability

$$\Pr_{\text{unnorm}}(y) = \sum_{\{x:G_0x=0,G_1x=y\}} \Pr(x). \tag{4.31}$$

The total success probability is captured by the sum over all possible output states, so

$$\begin{aligned} P_{\text{suc}} &= \sum_y \Pr_{\text{unnorm}}(y) \\ &= \sum_y \sum_{\{x:G_0x=0,G_1x=y\}} \Pr(x) \\ &= \sum_{\{x:G_0x=0\}} \Pr(x). \end{aligned} \tag{4.32}$$

Conditioned on success, the normalised distribution on output states is $\Pr_{\text{out}}(y) = \Pr_{\text{unnorm}}(y)/P_{\text{suc}}$. Given an explicit form for G_0 and G_1, this completes the black box picture of block protocol performance. These formulae form the basis upon which we build both our analytic and numerical analysis in following sections.

The G-matrix formalism of Bravyi and Haah has been significantly generalised [145, 146]. This extension provides protocols that convert noisy T magic states into another species capable of injecting complex multi-qubit circuits. Included in this framework are protocols, based on G-matrices, which provide resources for implementing Toffoli gates. Protocols independently proposed by Jones [143] and Eastin [142] realised error suppressed Toffoli gates, and here we consider a variant based on G-matrices that we discuss further in Sec. 5.3. All three variants perform identically when we use block checking. However, with the G-matrix formalism we can again use module checking to track correlations and achieve superior error suppression. This is just one additional application of module checking, the technique can also be deployed in conjunction with the general class of protocols introduced in Ref. [145, 146]

4.5 Module checking

We present a method of tracking the leading order errors, accounting for correlations, through many rounds of module checked protocols. At each level of distillation the

level 1	level 2	C_l	$\lim_{k\to\infty} C_l$	$\lim_{k\to\infty} k_1 k_2 \epsilon_{\mathrm{BH}}/\epsilon^4$
BH_{k_1}	BH_{k_2}	$\left(16+\frac{9}{2}k_1(k_1-1)\right)\left(4+\frac{3}{2}k_2(k_2-1)\right)$	$\frac{27}{4}k_1^2k_2^2$	$27k_1^3k_2^2$
Tof	BH_k	$112\left(4+\frac{3}{2}k(k-1)\right)$	$168\cdot k^2$	$2352\cdot k^2$
BH_k	Tof	$\left(16+\frac{9}{2}k(k-1)\right)28$	$126\cdot k^2$	$252\cdot k^3$

level 1	level 2	level 3	C_l	$\lim_{k\to\infty} C_l$	$\lim_{k\to\infty} k_1 k_2 k_3 \epsilon_{\mathrm{BH}}/\epsilon^8$
BH_{k_1}	BH_{k_2}	BH_{k_3}	$\left(256+\frac{81}{2}k_1(k_1-1)\right)\left(16+\frac{9}{2}k_2(k_2-1)\right)\left(4+\frac{3}{2}k_3(k_3-1)\right)$	$273.375\cdot k_1^2k_2^2k_3^2$	$2187\cdot k_1^5k_2^3k_3^2$
Tof	BH_{k_1}	BH_{k_2}	$1792\left(16+\frac{9}{2}k_1(k_1-1)\right)\left(4+\frac{3}{2}k_2(k_2-1)\right)$	$12096\cdot k_1^2k_2^2$	$28^4\cdot 3^3\cdot k_1^3k_2^2$
BH_{k_1}	BH_{k_2}	Tof	$\left(256+\frac{81}{2}k_1(k_1-1)\right)\left(16+\frac{9}{2}k_2(k_2-1)\right)28$	$5013\cdot k_1^2k_2^2$	$20412\cdot k_1^5k_2^3$

Table 4.2: The leading coefficient C_l for a variety of protocols with 2 and 3 levels of distillation. For clarity we also show C_l in the large block limit ($k\to\infty$). When we write BH_k, we implicitly assume $k>2$, as the results differ slightly for the $k=2$ case. The final column shows the ratio between the union bound estimate made by utilising the reduced error rate on a single qubit ϵ_{BH} made by Bravyi and Haah and the corresponding estimate of the global error rate given by $C_l\epsilon^{2^l}$. It can be seen that the benefit (in error rate) of module checking scales with both k and the number of rounds of distillation.

protocol is characterised by a function η that summarises how well it tolerates leading order errors

Definition 3 *For every distance-2 G-matrix code that distills $n\to k$ qubits, we define a function $\eta:\mathbb{Z}_2^k\to\mathbb{Z}$ taking values*

$$\eta(y) := \#\{y : |x|=2, G_0x=0, y=G_1x\}, \tag{4.33}$$

where $|\ldots|$ is the weight $|y|=\sum_j y_j$, and $\#$ counts the number of elements in a set $\{\ldots\}$. In other words, the value $\eta(y)$ counts the number of inputs x such that:

1. *they are weight 2 (formally $(|x|=2)$) ; and*
2. *they are undetected by the protocol (formally $G_0x=0$); and*
3. *give y as output (formally $G_1x=y$).*

Since $\eta(y)$ counts the number of lowest-weight errors leading output y, the total error rate for a single round of distillation can be simply estimated as

$$\epsilon_g = \left(\sum_y \eta(y)\right)\epsilon^2 + O(\epsilon^4), \tag{4.34}$$

Counting errors over many rounds is a more subtle problem, but we find that η still provides sufficient information to perform this calculation. If each round can even use a different protocol, we label the corresponding function with a subscript. We now state our key result

Theorem 1 *Consider L rounds of distillation with module checking, with associated functions $\eta_1, \eta_2, \ldots \eta_L$. Such a protocol outputs a multi-qubit magic state where the l^{th} level modules succeed with probability*

$$P_{\text{suc},l} \simeq \frac{A_l + B_l}{(A_{l-1} + B_{l-1})^{n_l}} \tag{4.35}$$

and output states have global infidelity

$$\epsilon_g^{(l)} \simeq \frac{B_l}{A_l + B_l}. \tag{4.36}$$

where we have made use of the following

$$A_l = (1-\epsilon)^{(n_1 n_2 \ldots n_l)}, \tag{4.37}$$

$$B_l = C_l \epsilon^{2^l} (1-\epsilon)^{(n_1 n_2 \ldots n_l - 2^l)}, \tag{4.38}$$

$$C_l = \prod_{j=1}^{l} \left(\sum_v \eta_j(v)^{2^{l-j}} \right). \tag{4.39}$$

The most important quantity in the theorem is C_l. After two rounds C_2 is simply $(\sum_y \eta_1(y)^2) \cdot (\sum_y \eta_2(y))$. After l rounds, C_l is a product of l terms. One may approximate the theorem to leading order $\epsilon_g^{(l)} \sim C_l \epsilon^{2^l}$. However, our later numerical investigations found that such an approximation was too coarse, and we really need the slightly more complex form given in the theorem. We postpone proof of this result to Sec. 4.5.1.

For the Bravyi-Haah protocols it is easy to verify the following

$$\eta_{\text{BH}_k}(y) = \begin{cases} 3 & , |y| = 2; \\ 4 & , |y| = k; \\ 0 & , \text{otherwise.} \end{cases} \tag{4.40}$$

so that for $k > 2$ we have

$$\sum_y \eta_{\text{BH}_k}(y)^m = 4^m + 3^m \binom{k}{2} = 4^m + 3^m \frac{k(k-1)}{2}. \tag{4.41}$$

where the binomial factor arises in counting the number of y where $|y| = 2$. When $k = 2$, we have a special case as then the $|y| = 2$ terms and $|y| = k$ terms are all the same, and so

$$\sum_y \eta_{\text{BH}_2}(y)^m = 7^m. \tag{4.42}$$

For the Toffoli protocol we have

$$\eta_{\text{Tof}}(y) = 4, y \neq 0. \tag{4.43}$$

and so

$$\sum_y \eta_{\mathrm{Tof}}(y)^m = 7 \cdot 4^m. \tag{4.44}$$

Given expressions for $\eta(y)^m$ we can compose these protocols anyway we wish and obtain an estimate of the global error rate as given in Thm. 1.

4.5.1 Proof of error tracking

When iterating distillation protocols we have a collection of inputs, which may have been outputs of an earlier round. We illustrated the structure in Fig. 4.3, where we introduced the terminology of branches and modules. Our proof proceeds by considering how an individual module maps probability distributions over its inputs to a probability distribution over outputs. If a level-l module uses many $n_l \to k_l$ blocks, then it requires n_l branches of inputs from earlier rounds. If each branch carries N_l qubits, then we need N_l instances of the $n_l \to k_l$ protocol so that the whole module maps $N_l n_l \to N_l k_l$ qubits. The size of N_l equals the product of the k_l values for all earlier rounds, which can be verified by simply counting back through the tree.

We use $x^{(j)} \in \mathbb{Z}_2^N$ to denote the error distribution of the input contribution from the j^{th} branch, and use probability $\Pr(x^{(j)})$ for the probability of this event. For now we take this probability as given and note only that for a block that quadratically suppresses noise, we have that $\Pr(x^{(j)} \neq 0)$ is to first order proportional to ϵ^{2^l} after l rounds of distillation. If a single block of a $n_l \to k_l$ protocol is described by a matrix G, then N copies within a module are described by $\mathcal{G} = G \otimes \mathbb{1}_N$ where $\otimes$ is the tensor product and $\mathbb{1}_N$ is the identity matrix of dimension N. On the first round, $N = 1$ and so we simply have $\mathcal{G} = G$. Before proceeding we must clarify how this tensor product structure relates to the input strings x. We define $\delta^{(j)}$ as a length n_l binary vector with entries: 1 in the j^{th} location, and 0 everywhere else. It follows that $x = \sum_j (\delta^{(j)} \otimes x^{(j)})$, so that $\mathcal{G}x = \sum_j (G\delta^{(j)} \otimes x^{(j)})$. This ordering ensures that no two qubits from the same branch collide into the same block, which must be prevented because of correlations within branches. We call this the canonical ordering and it is assumed throughout. Other orderings exist that prevent such collisions, such as an arbitrary permutation of qubits within any branch. Potentially such permutations could perform better or worse than the canonical choice, leaving room for further optimisation. We continue with the canonical choice as it is particularly natural and amenable to analysis.

After a module passes module checking, the output branch has errors distributed as

$$\Pr_l(y) = \frac{1}{\mathrm{P}_{\mathrm{suc},l}} \sum_{\{x:\mathcal{G}_0 x=0,\mathcal{G}_1 x=y\}} \Pr(x), \tag{4.45}$$

where the denominator is a normalisation constant accounting for the success probability

$$\mathrm{P}_{\mathrm{suc},l} = \sum_y \sum_{\{x:\mathcal{G}_0 x=0,\mathcal{G}_1 x=y\}} \Pr(x). \tag{4.46}$$

For our proof, it is useful to work with unnormalised probabilities that we write as

$$P_l(y) = \sum_{\{x:\mathcal{G}_0 x=0,\mathcal{G}_1 x=y\}} P_{l-1}(x), \tag{4.47}$$

where we will renormalise later. Herein, we focus on the smallest weight errors that can lead to a particular y. The no error case ($y = 0$) occurs when there the initial states had no errors, so

$$P_l(0) \simeq (1-\epsilon)^{m_l}, \tag{4.48}$$

where m_l counts the total number of raw qubits needed for l rounds of distillation, namely $m_l = \prod_{j=1,\ldots l} n_j$. For protocols quadratically suppressing noise, 2^l is the smallest number of errors that can evade detection. Therefore, we take the approximation

$$P_l(y) \simeq C_l(y)\epsilon^{2^l}(1-\epsilon)^{m_l-2^l}, \tag{4.49}$$

where $C_l(y)$ counts the number of different weight 2^l errors that lead to output y. For one round of distillation this is simply

$$C_1(y) = \eta(y). \tag{4.50}$$

Recall from Definition. 3 that that $\eta(y)$ is the function that counts precisely the number of weight-2 errors that lead to a particular output.

Finding $C_l(y)$ for higher levels ($l > 1$) is more involved. The relation between input and output error strings is $y = \mathcal{G}_1 x$ (assuming $\mathcal{G}_0 x = 0$), so x vectors of weight 2 can result into a variety of weights y vectors, potentially increasing the weight, and so some care is needed.

Consider a module (on some $l > 1$ level) where some of the incoming branches contain errors. The probability of two branches a and b containing errors is

$$\begin{aligned} &P_{l-1}(x^{(a)})P_{l-1}(x^{(b)})P_{l-1}(0)^{n_l-2} \\ &\simeq C_{l-1}(x^{(a)})C_{l-1}(x^{(b)})\epsilon^{2^l}(1-\epsilon)^{m_l-2^l}. \end{aligned} \tag{4.51}$$

These errors may contribute to undetected errors leaving the module. Fewer or more branch failures do not provide leading order contributions. Consider when only 1 branch contains any errors. After this branch is split up and fed into different blocks, each $n_l \to k_l$ block can contain at most 1 error, and so it will be detected. If $t > 2$ branches contain an error, the probability will be weighted by $\epsilon^{t*2^{(l-1)}}$, which for small ϵ is a rare process compared to two failed branches.

Furthermore, with two erroneous branches a and b, we can further deduce that either $x^{(a)} = x^{(b)}$ or the error will be detected. To see this, we begin by noting that errors will only be undetected if $(G_0 \otimes \mathbb{1}_N)x = 0 \pmod 2$, and we have

$$(G_0 \otimes \mathbb{1}_N)x = (G_0\delta^{(a)}) \otimes x^{(a)} + (G_0\delta^{(b)}) \otimes x^{(b)}. \tag{4.52}$$

Both $G_0\delta^{(a)}$ and $G_0\delta^{(b)}$ are columns of the G_0 and so are nonzero. Since no terms are zero, it can only vanish $\mod 2$ if both $G_0\delta^{(a)} = G_0\delta^{(b)}$ and $x^{(a)} = x^{(b)}$. This is a central pivot of the proof, so we will give a second explanation of what is happening here. Note that if $x^{(a)} \neq x^{(b)}$, then without loss of generality, $x^{(a)}$ had an error in at least 1 location that is absent from $x^{(b)}$. If the error's location is t, then the t^{th} distillation block will receive an input with only 1 error and must detect it. We conclude that the dominant source of errors is from the identical failure of pairs of branches.

To simplify notation, let $u = \delta^{(a)} + \delta^{(b)}$ and $f = x^{(a)} = x^{(b)}$, so we have the more concise expression $x = u \otimes f$ for relevant errors, with u being weight 2. Above we noted that an undetected error must satisfy $G_0\delta^{(a)} = G_0\delta^{(b)}$, which is equivalent to $G_0 u = 0 \pmod 2$. The output from such an error is $y = (G_1 \otimes \mathbb{1}_N)(u \otimes f) = (G_1 u) \otimes f$. Therefore, we can now deduce that

$$P_l(v \otimes f) = \sum_{\{u: G_0 u = 0, G_1 u = v\}} P_{l-1}(f)^2 P_{l-1}(0)^{n_l - 2}.$$

by counting over all u that lead to the same v. Therefore,

$$C_l(v \otimes f) = \sum_{\{u: G_0 u = 0, G_1 u = v, |u| = 2\}} C_{l-1}(f)^2.$$

The summation is over weight 2 vectors u, but the terms are independent of u, and so we find that

$$C_l(y) = C_l(v \otimes f) = \eta_l(v) C_{l-1}(f)^2,$$

where we have again used the η notation introduced in Def. 3. For two rounds of distillation, $l = 2$, and so $C_{l-1} = C_1$ and we can end the recursion by using Eq. 4.50, so that

$$C_2(y) = C_2(v \otimes f) = \eta_2(v)\eta_1(f)^2, \tag{4.53}$$

However, if we have more than 2 rounds, notice that the relevant f will again have the form $f = v' \otimes f'$ so that

$$\begin{aligned} C_l(y) &= \eta_l(v) \cdot [C_{l-1}(v' \otimes f')]^2, \\ &= \eta_l(v) \cdot [\eta_{l-1}(v') C_{l-2}(f')^2]^2, \\ &= \eta_l(v) \cdot [\eta_{l-1}(v')]^2 \cdot [C_{l-2}(f')]^4, \end{aligned} \tag{4.54}$$

and so on, until we reach C_1 and can use Eq. 4.50 to end the recursion.

From $C_l(y)$ we have a good leading order approximation of the probability of an error $P_l(y)$. The total (unnormalised) probability of an error is

$$\begin{aligned} B_l &= \sum_{y \neq 0} P_l(y) \\ &= \epsilon^{2^l} (1-\epsilon)^{m_l - 2^l} \sum_{y \neq 0} C_l(y) \\ &= \epsilon^{2^l} (1-\epsilon)^{m_l - 2^l} C_l \end{aligned} \tag{4.55}$$

where we have used the shorthand

$$\begin{aligned} C_l &= \sum_{y \neq 0} C_l(y), \\ &= \prod_{j=1}^{l} \left(\sum_v \eta_j(v)^{2^{l-j}} \right). \end{aligned} \tag{4.56}$$

Equating also $A_l = P_l(0) = (1-\epsilon)^{m_l}$, we find we have reached the expressions of Eqs 4.37. To renormalise the error probability B_l, we simply divide through by the total probability $A_l + B_l$, yielding one equation of Thm. 1. The denominator $A_l + B_l$ represents that success probability not just that a single module succeeds, but that all events feeding into that module also succeed. We are actually interested in the success probability conditioned on previous modules being successful, which is $(A_l + B_l)$ divided by $(A_{l-1} + B_{l-1})^{n_l}$. This divider here comes from the unconditional success probability of an $(l-1)$-level module, $(A_{l-1} + B_{l-1})$ and that n_l of these feed into an l-level module. This completes the proof of Thm. 1.

4.6 Analysis of module checking

4.6.1 Numerical analysis

We perform numerical Monte Carlo simulations by sampling from the probability distribution of the raw magic states, where $\Pr(x) = \epsilon^{|x|}(1-\epsilon)^{N-|x|}$, and track its evolution through the magic state factory. To the author's knowledge, this is the first such numerical investigation of correlations inside a magic state factory.

The simulations track the progress of potentially damaging input error configurations of the initial raw magic states though the factory. Direct Monte Carlo simulation is possible for 1-2 rounds of distillation, but with 3 or more rounds the failure events become so rare that brute force simulation fails to provide statistically meaningful data. Rather we use a novel "rare events" technique that preselects cases with two or more corrupt branches. A full description of the method employed can be found in Sec. 4.7. We thus investigate the performance of a factory which makes use of module checking for a range of raw magic state error rates between 0.1% and 1%. We find that the leading order analytic estimates match well with the numeric results, with the difference between the two being of the order of a few percent in the investigated parameter regime. In fact wherever the block protocols are involved, if $k \leq 14$ we find the percentage difference in between the numerical simulation and analytic estimate is $< 10\%$. This discrepancy between the analytic estimate and numerical simulations is not visible on a log-log plots presented in Sec. 4.7.

The cost of a protocol is the average number of raw magic states consumed to produce one higher fidelity magic state. For an $n \to k$ protocol, which takes in states with error p, this is

$$\mathcal{C}(p) = \frac{n}{kP_s(p)} \tag{4.57}$$

For l rounds of distillation we have $\mathcal{C}_l(p) = \prod_{i=1}^{l-1} \mathcal{C}_i(p_i)$.

Fig. 4.4 compares the cost of a Bravyi-Haah magic state factory, with block checking and module checking, showing the minimum cost achievable for given output error. For output error rate and success probability we use known expressions for block checking, and for module checking the success probability and global error given by Thm. 1 with an estimate on the reduced error rate on a single qubit given by the global error rate estimate divided by the total number of qubits in this factory's output.

We find that module checking is superior to block checking for a large proportion of target error rates, and can use up to four times fewer raw magic states in some

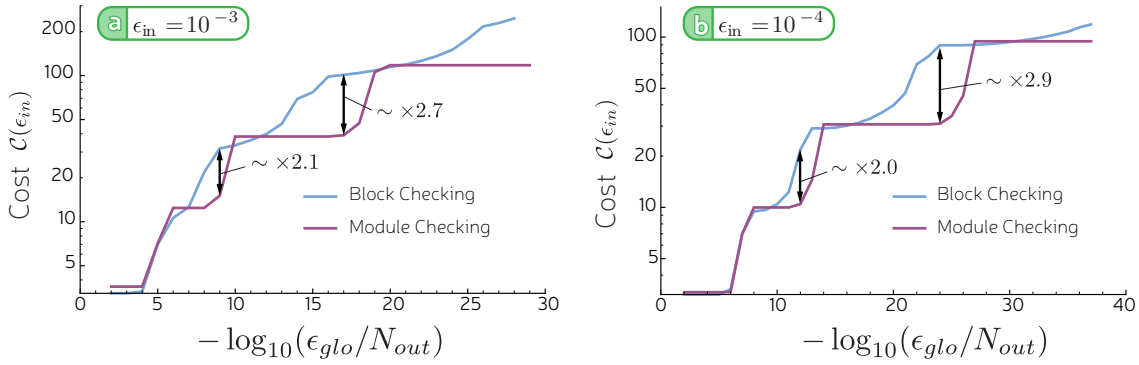


Figure 4.4: (a) & (b) The cost $\mathcal{C}$ of the Bravyi-Haah protocols utilising both block checking and module checking. It can be seen that only around the transitions to an additional level of distillation is block checking very slightly preferable.

regimes. However, near a transition from j to $j+1$ rounds of distillation, module checking loses it advantage and may even be slightly outperformed. The best error rates that can be achieved for a given number of rounds use low k block codes, for these the benefit in global error rate of module checking over block checking is smaller (see Table II) while the success probability is of course much inferior. Above the transition higher k values are being used, for which the success probability is much lower, and this washes out the benefit of the superior error suppression over block-checking, which has a higher success probability. In these regimes, as one might expect, module checking is not the optimal approach. In those regions between the transitions, module checking allows use of higher k protocols, which are more efficient, to achieve the error rates of lower k block checked protocols.

4.7 Numerical Simulations

4.7.1 Brute Force method

In simulating a magic state factory it quickly becomes apparent that a brute force method of simulation is inadequate. The "brute force" method simply involves randomly generating a boolean string of length $\prod_l n_l$ to describe the input to the magic state factory and performing calculations of the stabilisers and logical output of each module of the factory. The explicit procedure for a three round factory is given in Appendix D, Algorithm. D.1. For each module in round one, a random input is generated and tested until the stabiliser checks for the module are passed. The logical output of each of these successful modules is saved until enough have been generated

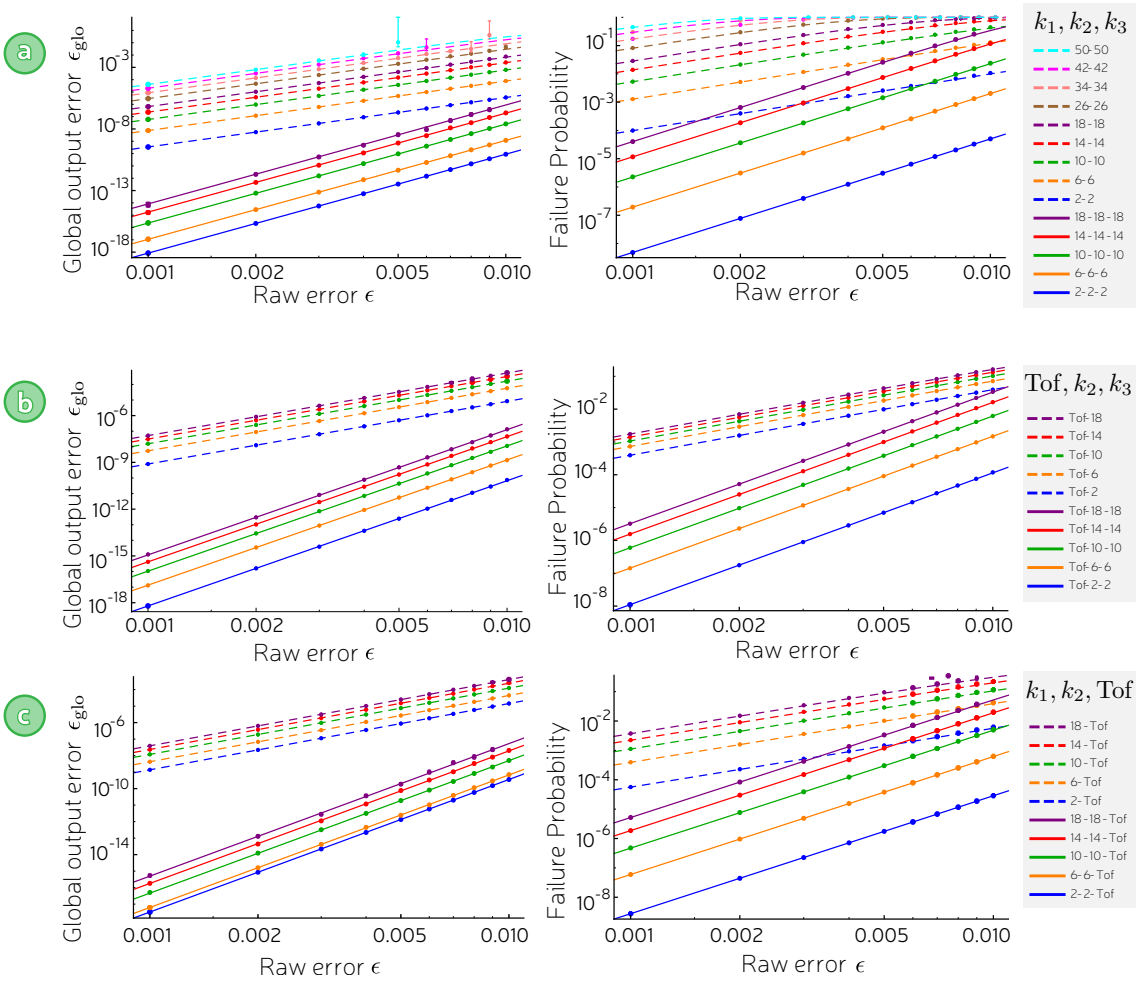


Figure 4.5: Numeric vs analytic. (a) Bravyi-Haah block protocols with two and three rounds of distillation. (b) Toffoli protocol followed by one or two rounds of Bravyi-Haah. (c) One or two rounds of Bravyi-Haah followed by the Toffoli protocol. The close correspondence of our analytic treatment of the factory and numerical simulations is demonstrated for two and three rounds of the magic state distillation with module checking. Points represent simulation data, while the lines are the corresponding analytic estimates as determined by Thm. 1. The protocols are labelled by the k value of each round, which is set to be equal for each round of distillation. The global error, the probability of a logical error anywhere in the output of the factory is shown. For a "2-2-2" factory this is the probability of an error anywhere in the 8 output magic states, for a "26-26-26" this corresponds to an error anywhere in the 17,576 magic states that this factory produces.

to feed into round two. These outputs are shuffled according to Equation 4.52 before entering round two. Here the module checking is performed again. If all the module checks are passed, then we again proceed by feeding the logical output of round 2 to round 3, after shuffling. Having reached round 3 we then determine the characteristics of this module by recording where the module check fails, and if it succeeds whether the output of the module contains a logical (undetected error).

Clearly a large number of iterations of this protocol are required to obtain reliable statistics for the performance of the factory. For example, our analytic treatment estimates that with a raw magic state error rate $\epsilon = 0.001$ that the rate of undetected logical error of a round 3 module is $\sim 10^{-21}$. As such, successfully simulating this by brute force would require $\gg 10^{21}$ attempts at the algorithm to be made, not just to build statistics but to ensure that enough instances of the third round module are generated in the first place. We find that the simulation of two round factories by brute force is achievable for $2 < k < 50$, but to simulate three rounds a different method is required.

4.7.2 Rare Events method

An undetected error in the factory's output after three rounds of distillation is a rare event. To simulate these and gain adequate statistics we must use a method in which we as far as possible eliminate the simulations of input error configurations that we know cannot lead to a logical error. Our chosen method of doing this proceeds as follows. We know that a module only has a chance of failing if at least two of the branches entering that module contain an error. For a module in the third round, we focus on cases where at least 2 of its input branches contain errors. We do this by a process that can be called preselection in contrast to postselection. In postselection, we sample from a distribution and reject instances that do not meet a certain criteria. This is not feasible when the criteria (here having at least 2 corrupt branches) is rare, and so we instead construct a new probability distribution conditioned on the criteria being meet. The first step of the algorithm therefore is to first decide how many error containing branches will enter this module, given that this number is ≥ 2, based on the statistics already gathered for the rate of errors in the output of round 2 modules. Later we analytically adjust for this process of preselection.

For each of the 'corrupt' branches entering round three, we know that these must originate from a round 2 module which itself had 2 or more corrupt branches entering it. These branches again would have originated in a round 1 module (equivalently a block) which had at least two errors fed to it. We can thus greatly reduce the size of the

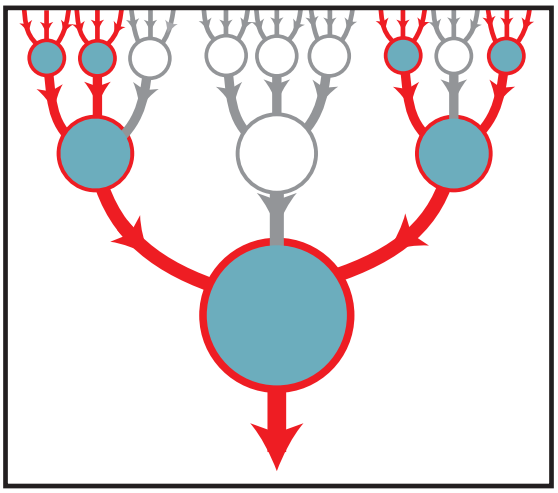


Figure 4.6: Rare events method. When simulating the fictional factory of Fig. 4.3 with our 'Rare Events' method, only the red highlighted parts of the factory are actively simulated, i.e. require computation of the stabiliser outcomes and the logical state of the outputs.

simulation and the number of iterations required by simulating only on a subsection of the factory where these errors have occurred, see Fig. 4.6. The probability of an undetected error after the first round was determined analytically for Bravyi-Haah by explicitly calculating the weight enumerator

$$\begin{aligned} W(k,\epsilon) = 2 \sum_{m \text{ is odd}}^{k} (1-2\epsilon)^{3m+4} + \sum_{m=0}^{k/2} (1-2\epsilon)^{3(2m)} \\ + 6\sum_{m=0}^{k} (1-2\epsilon)^{2k-m+4} + \sum_{m=0}^{k/2} (1-2\epsilon)^{3(2m)+8} \end{aligned} \tag{4.58}$$

which allows analytic calculation of the global fidelity of the output of a single round of distillation. The corresponding result for the Toffoli protocol is given in Ref [142].

Having chosen the number of failed branches that need to enter round three we attempt to generate these failed branches. We thus simulate the reduced factory, as described above, discarding and repeating each module not only if the stabiliser module check has failed, but also when the check is passed and the output does not contain a logical error. Using this method we eliminate the simulation of a vast number of the possible input error strings, while holding smaller boolean vectors in memory at all but the last step, the full simulation of the module in round 3. Some pseudo-code is presented in App. D, Algorithm D.2.

This method of simulation allows us to calculate the correlated error rate as follows.

A key number is the probability p_{num} that two or more branches leaving round 2 of the factory contain an error.

$$p_{\text{num}} = 1 - (1 - \epsilon_{\text{glo}}^{(2)})^{n_3} - n_3 \epsilon_{\text{glo}}^{(2)} (1 - \epsilon_{\text{glo}}^{(2)})^{n_3 - 1} \tag{4.59}$$

where $\epsilon_{\text{glo}}^{(2)}$ is the probability that a branch exiting round 2 contains an undetected error, as determined by numerical simulations of a two round factory.

Using this allows us to determine the success probability and the global error in the output of the round 3 module.

$$p_{\text{suc}}^{(3)} = (1 - \epsilon_{\text{glo}}^{(2)})^{n_3} + p_{\text{num}}(a_3 + b_3) \tag{4.60}$$

and

$$\epsilon_{\text{glo}}^{(3)} = \frac{b_3 \; p_{\text{num}}}{p_{\text{suc}}} \tag{4.61}$$

where

$$b_3 = \frac{\#\{\text{ERROR}\}}{\#\{\text{SUCCESS}\} + \#\{\text{FAIL}\} + \#\{\text{ERROR}\}}$$

is the estimate of the probability of a logical error in the output branch of round 3 from output of the simulation; and

$$a_3 = \frac{\#\{\text{SUCCESS}\}}{\#\{\text{SUCCESS}\} + \#\{\text{FAIL}\} + \#\{\text{ERROR}\}},$$

and in these equations $\#\{q\}$ is the total number of outputs q from the programmes described in Appendix D.

4.8 Conclusion

In this chapter magic state distillation and some protocols achieving it were described in more depth. G-matrices were used to express some of the key protocols in the same formalism which allowed us to make better estimates of the output error rate of the magic state factory that is composed when using multiple rounds of different protocols. We introduced one metric, the 'cost' of a magic state protocol, which gives some asymptotic properties of the distillation process – namely the average number of noisy magic states needed to produce a distilled magic state of some desired error rate. We found that for an important class of protocols, the Bravyi-Haah protocols, the probability that a factory's output contains no error has been underestimated. However, as a performance metric the cost falls some way short of describing what

is really useful for planning and designing a quantum computer. What we really want to know is how many *physical* qubits are required to implement a post-classical algorithm and how long it will take to do so. Such an algorithm has a finite number of steps, a finite number of gates and logical qubits, and as such asymptotic limits are not necessarily the most useful way to go about determining the feasibility of a proposal. It is well understood that error correction and magic state distillation are theoretically efficient and we can come up with techniques that reduce the exponent in the respective polylog(ϵ^{-1}) scalings, but how far are we today from a real device? And what other techniques can we apply to reduce the real qubit and time overhead of a quantum computer? We will now focus on these questions.

Chapter 5

The overhead of magic state distillation

This chapter sets out to analyse the total resources required of a surface code magic state factory, and thus the likely size and speed of a quantum computer, that would allow evaluation of Shor's algorithm for a post-classical factoring task. Sections 1-3 present explicit circuit level realisations of important distillation protocols, introducing the concept of 'gauge' magic state distillation to improve efficiency. Section 4 outlines intensive numerical simulations of the spacetime overhead of magic state factories for both $|A\rangle$ and $|\mathrm{Tof}\rangle$ states. All simulations were performed by the author using C and Mathematica. The original research in this chapter was published in Physical Review A [118]. Note that the text of this chapter is adapted from the text the author wrote for Ref. [118]

The specification of the distillation protocol is just one aspect of designing magic state factories. The size of magic state factories depends both on the distillation protocols and also the underlying error correction codes, and significant saving can be made by judiciously reducing the error correction code at earlier rounds of magic state distillation. The descriptions of distillation protocols we have met so far are high-level, leaving open many aspects of how to implement these protocols. To assess the full resource cost, we require a low-level description of distillation protocols in terms of elementary operations, such as one and two qubit gates, preparations and measurements. We call such a description a realisation of a protocol, and a given protocol can have different realisations with varying time and qubit overhead.

Therefore, while the 'cost' metric that we met in the previous chapter is a useful guide to the performance of a distillation protocol, it fails to capture several important features of real magic state factories. The very purpose of magic state distillation is

to supplement the shortcomings of a particular error correcting code, which is to say that a magic state factory is implemented at the level of logical qubits, with the quantum information already encoded. As such, the more relevant question is not how many noisy input magic states are required per output state, but rather the number of the physical qubits that will build up such a factory and the rate at which the distilled magic states are produced. Both of these numbers are of key importance to determining the size of a quantum computer and the run time of an algorithm. A single number, the 'spacetime overhead' of the magic state factory captures these both as a single figure of merit, one which was also studied in Ref [122].

In this Chapter, we present a blueprint for a factory capable of delivering enough magic states to solve large Shor's algorithm tasks, beyond the reach of classical computation, within a surface code quantum computer. We take the surface code as our substrate for quantum computation, providing our error tolerance and non-magic gates, and then investigate the spacetime overhead of our various implementations of magic state distillation. Our results are summarised in Table 5.1, where we look at both the spacetime overhead required to produce a Toffoli magic state and the physical footprint of the factory required to produce magic states at an average rate that can just keep up with the 'time optimal' surface code implementation of the algorithm [147], which we further discuss in Sec. 5.4.

5.1 Realising the Bravyi-Haah protocols

There are many routes to realising magic state distillation. Assuming perfect Clifford operations, different realisations suppress errors equally, but differ in terms of temporal depth and required ancillas. Many of these potential realisations have only been sketched, without a complete assessment of resources involved. Here we introduce a new method particularly suitable for architectures implementing logical gates via transversal operations or lattice surgery [52]. Conceptually, inspiration comes from notions of subsystem codes and gauge fixing techniques, and so the approach may be called approach gauge-MSD. We will apply this to a subset of the Bravyi-Haah protocols. We consider only even k with $k \in \{2, 6, \ldots, 4m+2, \ldots\}$ as then the Bravyi-Haah codes have transversal T gates. For $k \in \{4, 8, \ldots, 4m, \ldots\}$, the Bravyi-Haah codes have also transversal T gates, but only up to a non-local Clifford correction.

As we have seen the Bravyi-Haah protocols have three X stabilisers which have weight 4, $2k + 4$ and $2k + 4$, see G_0 in Sec. 4.2.3. The 2 X-type observables of weight $2k + 4$ imply a potentially expensive time cost, assuming that a qubit cannot

be involved in multiple entangling gates simultaneously. However, the concept of gauge subsystem codes provides a method whereby such complex measurements can be broken down into a larger number of simpler measurements [148–151]. We remark that constant time realisation was also found Fowler *et. al.* [122], but was tied to a monolithic braiding architecture.

Here we present a rigorous demonstration that the Bravyi-Haah code can be viewed as a subsystem code, and using a combination of gauge fixing and a cat state ancilla we create a circuit of depth 4 for both X and Z measurement sets. We call this general approach gauge-MSD, where MSD is short for magic state distillation.

5.1.1 Subsystem Bravyi-Haah codes

Given a G-matrix (see Sec. 4.2) we define a subsystem code with stabilisers given by

$$\tilde{\mathcal{S}} := \langle Z[g_1], \ldots, Z[g_b], X[g_1], \ldots, X[g_b] \rangle$$

where $\{g_1, \ldots, g_b\}$ are the rows of G_0. Notice that now there are equal numbers of X and Z stabilisers and they both correspond to the rows of G_0. Bravyi and Haah considered a class of matrices obeying triorthgonality conditions, which require that $G_0 \subset \mathcal{G}^\perp$. Therefore, we have that $\tilde{\mathcal{S}} \subset \mathcal{S}$, with the subsystem code having strictly fewer stabilisers than the original Bravyi-Haah code. We denote the subsystem projector as $\tilde{\Pi} \propto \sum_{s \in \tilde{\mathcal{S}}} s$. We further take the logical operator for the subsystem code to be identical to those of the original Bravyi-Haah code. This leaves some degrees of freedom as neither stabilisers nor logical operators. We define the gauge group $\mathcal{S}_g := \langle Z[f_1], \ldots, Z[f_a], X[f_1], \ldots, X[f_a] \rangle$ where $\{f_1, \ldots f_a\}$ are rows of $G^\perp$. Notice that $\mathcal{S}_g$ contains $\mathcal{S}$, by virtue of $G_0 \subset \mathcal{G}^\perp$.

Let us recap. Our subsystem code is defined by its stabiliser $\tilde{\mathcal{S}}$ and gauge group $\mathcal{S}_g$, whereas the original Bravyi-Haah code has stabiliser $\mathcal{S}$, and these groups satisfy $\tilde{\mathcal{S}} \subset \mathcal{S} \subset \mathcal{S}_g$. However, $\mathcal{S}_g$ is inflated in size compared to $\mathcal{S}$ and is no longer abelian. Furthermore, one can verify that $\mathcal{S}_g$ does not contain any logical operators as follows. First, note that Bravyi-Haah use triorthogonal (also called triply even) matrices where for any $f, g \in G$ we have that $(f, g) = 1$ if and only if $f = g$ and $f \in G_1$.

As logical operators we take $X[l]$ and $Z[l]$ for each l in G_1. From the triorthogonality of G we see that $X[l]$ and $Z[l]$ anticommute, but $X[l]$ and $Z[l']$ commute when $l \neq l'$. To properly describe a subsystem code, where measuring the gauge operators does not damage the logical qubits, we require that the logical operators are not elements of the gauge group $\mathcal{S}_g$. Recall the gauge group is defined by vectors that reside in the dual code $\mathcal{G}^\perp$. Therefore, every gauge operator must have vanishing

inner product with every row in $\mathcal{G}$. However, $l \in \mathcal{G}$, and $(l, l) = 1$, so l is not in the dual space and the logical operators are not gauge operators. This completes our proof that the logical operators indeed lie outside the gauge group.

5.1.2 Outline of gauge-MSD

Here we present an outline of gauge-MSD, with details of how to realise multi-qubit Pauli measurements postponed until the next section. First we specify some notation. Refer back to Sec. 4.2 for definitions of the G-matrix and $G^{\perp}$-matrix. Let K be a binary matrix such that $G^{\perp} \cdot R = \mathbb{1} \pmod 2$, which is ensured to exist by virtue that $G^{\perp}$ is full rank. Furthermore, let M be a binary matrix such that $M \cdot G^{\perp} = G_0 \pmod 2$, which must exist since G_0 is in the span of $G^{\perp}$. Explicit examples of $G^{\perp}$, K and M will be given in the next section. Measurement outcomes will be recorded in binary: 0 for "$+1$" eigenvalues and 1 for "-1" eigenvalues. Let $\mathcal{O}$, $\mathcal{X}$, $\mathcal{Z}$ be disjoint sets

$$\begin{aligned} \mathcal{O} &= \{6 + 3j | j = 1, 2, \dots k\}, \\ \mathcal{X} &= \{1, 2, 3\}, \\ \mathcal{Z} &= \{4, 5, 6, 7, 8, 7 + 3j, 8 + 3j | j = 1, 2, \dots, k\}. \end{aligned} \tag{5.1}$$

Associated with these sets are binary matrices that allow us to compute Pauli corrections, H_Z is a k-by-$|\mathcal{X}|$ matrix and H_X is a k-by-$|\mathcal{Z}|$ matrix as follows

$$H_Z = \begin{array}{c} \begin{matrix} 4 & 5 & 6 & 7 & 8 & 10 & 11 & 13 & 14 & 16 & 17 & \dots \end{matrix} \\ \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & \dots \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & \\ & & & & & \vdots & & & \vdots & & & \end{pmatrix} \end{array} \tag{5.2}$$

$$H_X = \begin{array}{c} \begin{matrix} 1 & 2 & 3 \end{matrix} \\ \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ & \vdots & \end{pmatrix} \end{array} \tag{5.3}$$

where the numbers above the columns correspond to the elements in sets $\mathcal{Z}$ and $\mathcal{X}$. Lastly, we will also make use of a k-by-dim$(G^{\perp})$ matrix denoted Q that satisfies $G_1 = W + QG^{\perp}$ where W has $W_{j,6j+3} = 1$, $W_{j,1} = W_{j,2} = 1$ and zero everywhere else. An example Q is given in the next section.

We now state the protocol.

1. Measure all $Z[f_k]$ where f_k is the k^{th} row of $G^{\perp}$, recording outcomes as $\mu = (\mu_1, \mu_2, \ldots, \mu_k)$;
2. Apply $A[w]$ where $w = K\mu \pmod 2$;
3. Measure all $X[f_k]$ where f_k is the k^{th} row of $G^{\perp}$, recording outcome as $\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_k)$;
4. Declare SUCCESS if $M\gamma = (0, \ldots, 0) \pmod 2$, and FAILURE otherwise;
5. Measure qubits in set $\mathcal{X}$ in the X basis, recording outcomes as m_X; Simultaneously, measure qubits in set $\mathcal{Z}$ in the Z basis, recording outcomes as m_Z.
6. Qubits in $\mathcal{X}$ and $\mathcal{Z}$ are discarded, while qubits in set $\mathcal{O}$ are retained as output as qubits $(1, 2, \ldots, k)$;
7. Apply Pauli corrections $X[H_Z m_Z]Z[H_X m_X + Q\gamma]$, or update Pauli frame accordingly.

After steps 1-2, the state is deterministically projected by $\Pi_Z \propto \sum_{s \in \mathcal{S}_Z} s$ where $\mathcal{S}_Z := \langle Z[f_1], \ldots, Z[f_a] \rangle$, exactly as in the original Bravyi-Haah protocol. After step 3, the system is an eigenstate of $(-1)^{\gamma_k} X[f_k]$ and we can associate some projector with $\Pi_{X,\gamma}$ with this process. The postselection in step 4 ensures the system is an eigenstate of $+X[g_k]$ where g_k is the k^{th} row of $M \cdot G^{\perp}$. Since we defined M such that $M \cdot G^{\perp} = G_0$, we have that g_k are the rows of G_0. It follows that $\Pi_{X,\gamma} = \Pi_X \Pi_{X,\gamma}$ where Π_X is the projector for the X stabiliser of the Bravyi-Haah stabiliser code. Combining, steps 1-4 we have the projections

$$\Pi_{X,\gamma}\Pi_Z = \Pi_{X,\gamma}\Pi_X\Pi_Z = \Pi_{X,\gamma}\Pi, \tag{5.4}$$

where $\Pi = \Pi_X \Pi_Z$ is the full projector onto the Bravyi-Haah stabiliser codespace.

We have obtained the desired Π projection required by the Bravyi-Haah protocol. But we have picked up an additional $\Pi_{X,\gamma}$. This additional projection results from measuring some the gauge operators of the subsystem variant of Bravyi-Haah. Thus the logically encoded qubits are unharmed but there has been a change to the gauge degrees of freedom.

In step 5, we perform single qubit measurements to isolate the k logical qubits from the n qubits in the subsystem code. Recall that in the original presentation of Bravyi Haah we have logical operators $Z[l_j]$ and $X[l_j]$ for the j^{th} logical qubit, where l_j is the j^{th} row of G_1. The measurements localise these logical operators onto single

qubits, up to a Pauli correction depending on the measurement outcomes. That is, the measurements cause the state to become stabilised by some new $\pm Z[w]$ operators, and every logical $Z[l_j]$ can be multiplied by these operators to obtain a single qubit $\pm Z$ operator acting on the $(6+3j)^{\text{th}}$ qubit. It is straightforward to verify that the $\pm$ sign is corrected to $+$ by the Pauli correction $X[H_Z m_Z]$. To show that X logical operators are also localised on a single output qubit, we first multiply $X[l_j]$ by X-type gauge operators until it is acts on qubits 2,3 and $(6+3j)$. Measuring qubits 1,2 and 3 completes the localisation of $X[l_j]$ onto the $(6+3j)^{\text{th}}$ qubit. The required Pauli operator is now $X[H_Z m_Z + Q\gamma]$, where now there is also some dependence on the eigenvalues of gauge operators obtained in step 3.

5.1.3 Implementing Pauli measurements in minimum depth

Implementing the protocol requires a set of Z measurements, followed by a set of X measurements. In the original standard implementation the X measurements involved many qubits and so were difficult to implement. However, the previous subsection shows that the difficult X measurements can be replaced with lower weight measurements mirroring the Z measurements performed. The complexity of these measurements depends on the row weights of $G^{\perp}$. The matrix $G^{\perp}$ must generate the space $\mathcal{G}^{\perp}$, but there is some freedom in how we choose the generating rows. In the work of Bravyi-Haah and the previous chapter of this thesis, $G^{\perp}$ was only implicitly defined (as the dual of G), leaving it unclear how much time it would take to implement the required measurements. It is desirable that $G^{\perp}$ is as sparse as possible to minimise the resource overheads. Therefore, we wish to find a very sparse $G^{\perp}$. We found a family of $G^{\perp}$ matrices where all rows, except one, are weight 4 and the single exception has weight $k+2$. The construction is as follows:

$$G^{\perp}(k) = \left(\begin{array}{cccc|c|c|c|c|c} W & W & E & E & E & E & E & \cdots & E \\ L & L^* & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & L & R & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & R^* & R^* & 0 & 0 & 0 & \cdots & 0 \\ \hline 0 & 0 & 0 & R^* & R^* & 0 & 0 & \cdots & 0 \\ \hline 0 & 0 & 0 & 0 & R^* & R^* & 0 & & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & R^* & R^* & \ddots & 0 \\ \hline \vdots & \vdots & \vdots & \vdots & & \ddots & \ddots & \ddots & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & R^* & R^* \end{array}\right) \tag{5.5}$$

where L and R are defined as before; $W = (0,0,1,0)$, $E = (1,0,0)$ and

$$L^* = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix} \quad R^* = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \tag{5.6}$$

We present the exact form of this $G^\perp$ for $k = 2$, along with associated K, M and Q matrices used in the previous section.

$$G^\perp = \left(\begin{array}{cccccccc|ccc|ccc} 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{array}\right), \tag{5.7}$$

$$K = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix},$$

$$Q = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix},$$

$$M = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Notice that all but the top row have weight 4. The measurements corresponding to weight four rows can be implemented with each measurement using a single ancilla in the $|+\rangle$ state and four entangling gates (control-Z or control-X depending on the measurements). Therefore, for these measurements it is possible that all these gates can be realised in four time steps, while respecting that a qubit can only be involved in a single entangling gate at a time. Unfortunately, the top row of $G^\perp$ has weight $k + 2$, and so using a single ancilla to perform this measurement would result a

growing time cost with k. Therefore, for this single measurement we make use of a cat state $|0\rangle^{\otimes k+2} + |1\rangle^{\otimes k+2}$ so that the entangling gates can be performed in parallel. The cat state itself is constructed by merge operators on $|+\rangle^{\otimes k+2}$ qubits. The merge operations, as described in Refs [52] and [152], perform a joint ZZ measurement on the two logical qubits and thus project onto $|00\rangle\langle 00| + |11\rangle\langle 11|$ or $|01\rangle\langle 01| + |10\rangle\langle 10|$ subspaces. They thus commute with the control gates and so the cat state can be built concurrently with the control gates. This opens the possibility of realising each round of measurements in 4 times steps, but depends on whether the entangling gates can be scheduled in an economical manner. The scheduling problem is equivalent to a graph colouring problem and we find that it can be solved in 4 time steps (e.g. using 4 colours) as in Fig. 5.1.

5.1.4 Circuit-level description of the Bravyi-Haah protocols

Drawing on all the tools we have introduced in the previous sections it is possible to construct an explicit low-resource realisation of the Bravyi-Haah protocols, which is presented in Fig. 5.1. To recap: The first step uses ancillas to measure operators composed of Pauli-Z operators, and the second step applies a correction dependent on the measurement outcomes. The third step uses ancillas to measure operators composed of Pauli-X operators, with only certain measurement outcomes kept. After a successful third step, the k output magic states are within an encoded state and delocalised across $3k + 8$ sites. The fourth step uses measurements to localise the output qubits to specific sites. All these steps are detailed in the figure, and show how the multi-qubit measurements are broken down into ancilla preparation, two-qubit gates, and single qubit measurements. Observe in Fig. 5.1 that each multi-qubit measurement involves only four entangling gates, with each such gate designated a distinct coloured link. When using a single ancilla to measure a stabiliser of weight m, we need m time steps to perform the required controlled-gate operations. The low, and constant, weight of our measurements gives the realisation a constant time cost.

The time complexity of our realisation is simply

$$t_{\text{block}} = 8t_{\text{cnot}} + t_{\text{A}} + 2t_{\text{prep}} + 3t_{\text{measure}} \sim 8dt_{\text{sc}}, \tag{5.8}$$

where t_{cnot} is the 2-qubit gate (e.g. CNOT) time, t_{A} is the gate time for single qubit $A = (X + Y)/\sqrt{2}$ rotation, t_{prep} is the single qubit preparation time, t_{measure} is the single qubit measurement time and d is the code distance. Here, all operations are applied fault-tolerantly to logical qubits within an error correcting code. Therefore,

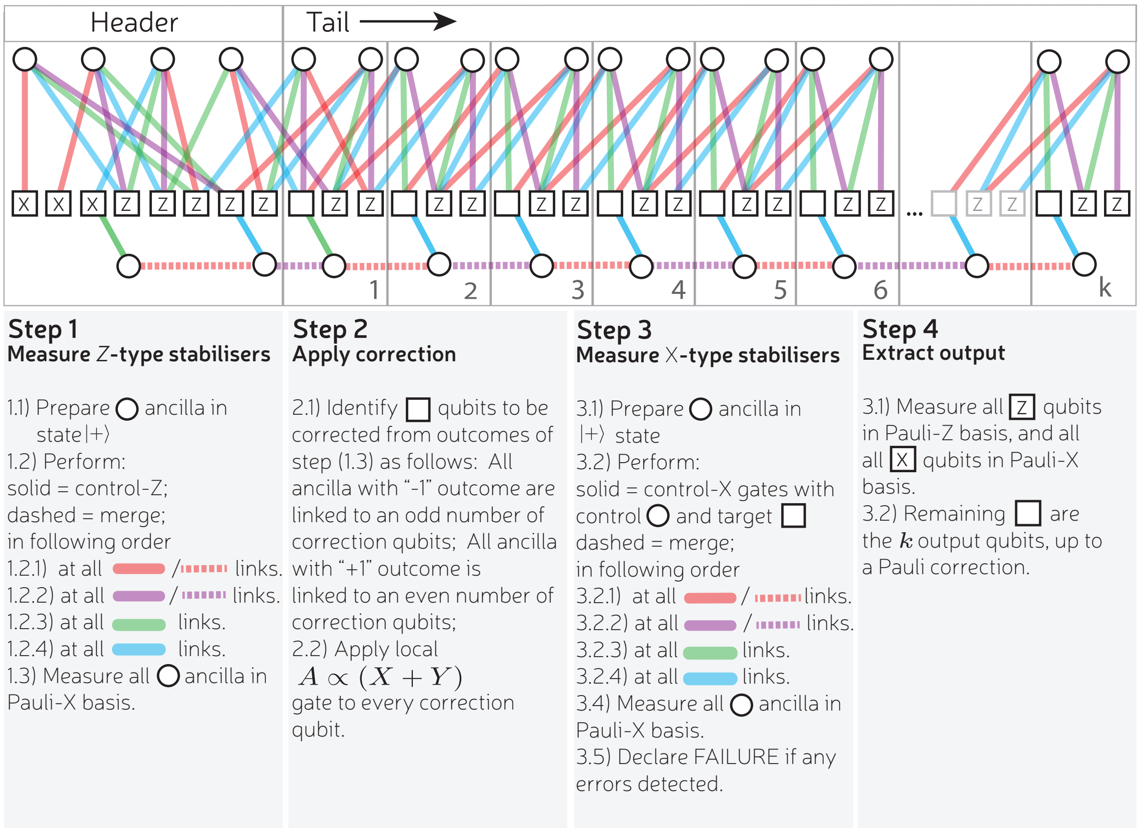


Figure 5.1: Explicit circuit for realising Bravyi-Haah $(3k + 8) \to k$ block protocols for $k = 2, 6, 10, 14, \ldots$. Squares indicate the $(3k + 8)$ noisy $|A\rangle$ magic states to be distilled, and circles represent ancilla qubits used to effect measurements on the magic states. Increasing k does not increase the number of time steps in the protocol, but increases the number of qubits involved in a block. As we increase k we add qubits to the tail end, with the protocol translationally invariant along the tail. Independent confirmation of the validity of the protocol for $k = 2$ by full wavefunction simulation was performed, which further confirmed that all single errors are detected and all two errors processes lead to outputs with correlated errors.

the time scales are for fault-tolerant gates. We will assume throughout that logical CNOT gates are applied transversally and thus take time $t_{\text{cnot}} = t_{\text{g}} + d \times t_{\text{sc}}$ where t_g is the time for the physical gate and t_{sc} is the time for a round of stabiliser measurements. Therefore, for large distances the time is dominated by $8dt_{\text{sc}}$. We will also take this as the time cost of a merge operation [52, 152] which we use to project two ancillary qubits into the even or odd parity subspace. We assume entangling gates can be performed in parallel, but a qubit can only participate in one gate at a time. We allow entangling gates to be long-range as is feasible within distributed architectures for quantum computing [153–160], though this may be relaxed at only a modest increase in resources. Note here that this is in fact a rather different to the architecture explored in Chapter 2, where one might imagine that any long range interactions may be very hard to implement. The problem of interacting a large number of logical qubits given only local interactions in the surface code is an interesting one, and will rely on braiding or lattice surgery in all likelihood, but is not explored here.

Our realisation uses a number of ancilla qubits, so that in addition to the $n = 8{+}3k$ qubits being distilled we also use $n_{\text{anc}} = 3k + 6$ ancilla qubits, giving $n_{\text{tot}} = 6k + 14$ logical qubits in total. These ancillas appear as circles in Fig. 5.1. With these logical qubits encoded in a distance d toric code, the total qubit cost is $N_{\text{tot}} = (6k + 14)d^2$. Therefore, the total space-time cost amounts to $N_{\text{tot}} t_{\text{block}} \sim (48k + 112)d^3$.

Comparison with braiding

It has been shown that Bravyi-Haah can be realised in constant time [122] assuming the architecture supports constant time implementations of multi-target CNOT gates (in time $t_{\text{MT}-\text{cnot}}$). This has time cost

$$t^{(2)}_{\text{block}} = 12t_{\text{MT}-\text{cnot}} + t_{\text{A}} + t_{\text{prep}} + t_{\text{inject}} + t_{\text{measure}}, \tag{5.9}$$

where t_{inject} is the time to inject a magic state into the circuit, and we can infer $t_{\text{inject}} = t_{\text{cnot}} + t_{\text{measure}}$.

In the standard circuit model, multi-target CNOT gates do not take constant time to implement. In the braiding picture, using ancillary defects, this is possible. The "time" cost of braiding 12 such gates is $12 \times 1.25 \times d \times t_{\text{sc}}$. The total space-time cost was reported as $(96k + 216)$ so-called plumbing pieces, which converts into $(\frac{5}{4})^3(96k + 216)$ qubit-rounds. It has been recently shown that lattice surgery also supports multi-target CNOT gates in constant time [161]. Though the Bravyi-Haah protocol was not considered in this setting, one can infer a lattice surgery time cost also scaling with $\sim 12d$, but with the qubit cost is not currently known.

The above discussion implies a modest space-time saving of using gauge-MSD in a distributed architecture rather than braiding in a nearest neighbour picture. We remark that gate times and qubit expense will vary on a much greater scale between different hardware platforms. In particular, the long-range gates of distributed schemes are often much slower, with photonics protocols impeded by photon loss and the potential need for entanglement purification [153–160].

When using the toric code A gates are also not naturally fault-tolerant and thus require their own process of state distillation and injection. The time t_A to implement such a gate is therefore the time taken by the logical CNOT which teleports the gate into the computation plus any Clifford correction that is required, although this can be rolled into a later Clifford operation. The resources required of the state distillation of the A are far less than that of the T-gate, and as the ancilla resource is reusable for many A gates [162, 163]. As such we neglect the overhead of these gates as a small additional overhead to the main process of $|A\rangle$ distillation.

5.2 Reed-Muller circuit

The usual form of the 15-qubit punctured Reed-Muller code can be described with the G matrix

$$G_{\mathrm{RM}} = \left(\begin{array}{cccccccccccccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{array}\right) \tag{5.10}$$

For our purposes an efficient choice of dual is

$$G_{\mathrm{RM}}^{\perp} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \tag{5.11}$$

which has a maximum row weight of 4 and a max column weight 5. There is a 5 colourable graph associated with this matrix shown in Fig. 5.2, similar to what we

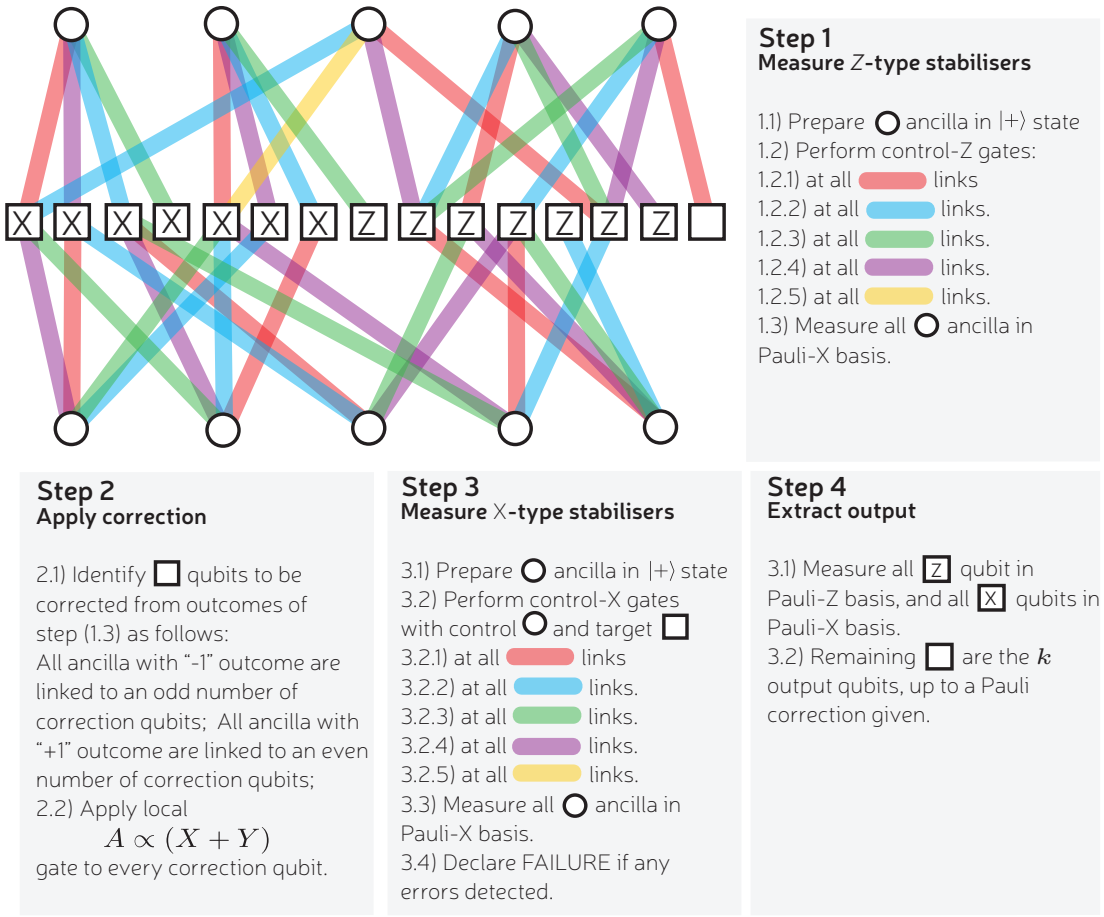


Figure 5.2: Squares represent magic states to be distilled, and circles are ancillas used to implement measurements in Reed-Muller. Edges show required hardware connections and associated required control-phase gates. Edges show time ordering of control phase gates, e.g. red, purple, green, blue and gold, demonstrating the required entanglement can be established in 5 time steps. Sometimes we call this the graph representing G_z^{RM}.

found for the Bravyi-Haah protocol. It is well known that the Reed-Muller code can also be viewed as a subsystem code [164], and so the Pauli-X measurements can clone the pattern of the Pauli-Z measurements. This approach yields the realisation shown in Fig. 5.2.

5.3 Toffoli protocol circuit

The Toff protocol converts 8 T-states into one Toffoli state

$$|\mathrm{Tof}\rangle = \frac{1}{\sqrt{8}} \sum_{a,b,c} (-1)^{(a+1)(b+1)(c+1)} |a, b, c\rangle. \tag{5.12}$$

It has previously been described in the circuit picture with its performance found by brute force counting of errors [142, 143]. Here we consider an equivalent protocol (with the same performance when using block checking) but with the G matrix formalism as the Bravyi-Haah protocols. The G matrix achieving this is simply

$$G_{\mathrm{Tof}} = \left(\begin{array}{cccccccc} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ \hline 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array} \right), \tag{5.13}$$

where proof that this distills was given in Ref. [145, 146]. Here we show in Fig. 5.3 how to convert this abstract protocol to a realisation with low spacetime overhead using the fact the G_{Tof} is self-dual, i.e. $G_{\mathrm{Tof}} = G_{\mathrm{Tof}}^{\perp}$.

5.4 Factory overhead analysis

In this section we present a comparison based on the spacetime overhead of implementing a magic state factory in a surface code quantum computer, utilising module checking, where appropriate, and our novel implementations of the Reed-Muller, Bravyi-Haah and Toffoli protocols.

We consider a number of issues in this estimate of the footprint of a magic state factory, including

1. *balanced investment*: the use of smaller surface codes during early rounds of magic state distillation;

2. *clock rate zoning*: cycling through distillation attempts faster during early rounds of magic state distillation;

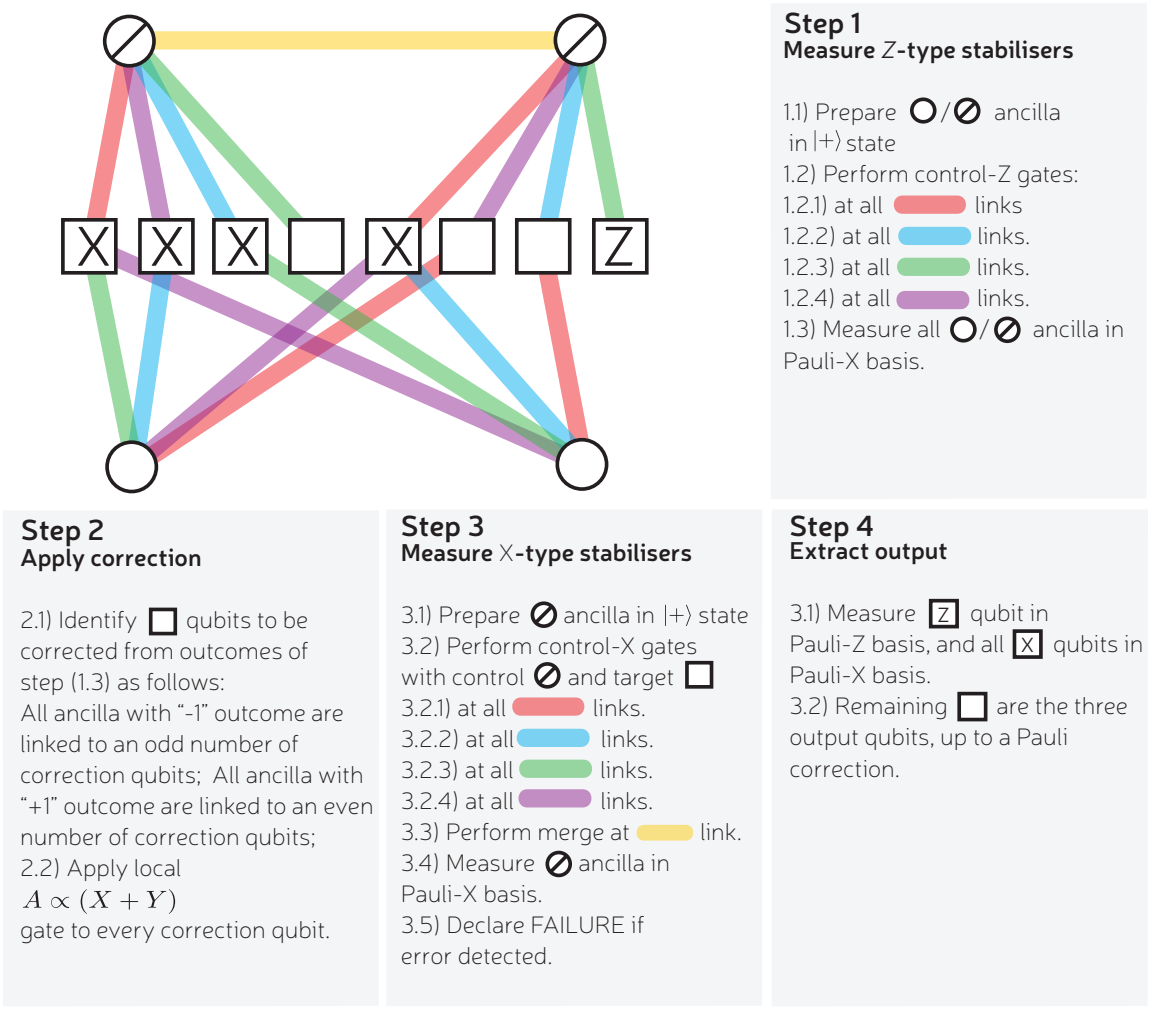


Figure 5.3: Squares represent magic states to be distilled, and circles are ancillas used to implement the stabiliser measurements which project the T-states into the Toffoli state. As before edges show required hardware connections and associated control-phase gates. Edges show time ordering of control phase gates, e.g. red, purple, green, blue and gold, demonstrating the required entanglement can be established in 4 time steps for the Z stabilisers and 5 time steps for the single X stabiliser, which utilises just two ancillas. We call this the graph representing G_z^{Toff}.

We will assume throughout that we use the method outlined by Li in Ref [165] to inject the initial raw magic state into a logical surface code. We will therefore assume throughout that the initial error rate on a magic state before distillation is $\epsilon_{\text{in}} = 0.4p_{\text{g}}$. We also use the 'rotated lattice' surface codes [52], such that a distance d surface requires d^2 physical data qubits. Of course a practical surface code requires the use of physical ancilla qubits to make the stabiliser measurements of the code, we leave this as an extra multiplicative factor to be applied to our overhead calculated here, as different physical realisations have different requirements. We estimate the surface code distance required to protect a logical qubit up to error rate P_l using the relation $P_l(d, p_{\text{g}}) = d(100p_{\text{g}})^{\frac{d+1}{2}}$ [122].

Given an algorithm, and some implementation of it, the number of magic states required is determined in advance. For a given distillation protocol we must then determine whether its magic state factory is capable of producing magic states of fidelity great enough that there is a high probability that none of the magic states required for the algorithm fails. Thus we set a target global error rate that our factory must achieve.

$$\epsilon_{\text{target}} = 1 - (P_{\text{suc,alg}})^{1/N} \tag{5.14}$$

where $P_{\text{suc,alg}}$ is the desired success probability of our algorithm (i.e. the probability that every non-Clifford gate works) and N is the number of successful iterations of the factory needed to produce the desired number of magic states. For example, a magic state factory which utilises 3 rounds of Bravyi-Haah distillation with a k value of 10 in each round will produce 10^{15} $|A\rangle$ states after 10^{12} successful iterations. A 90% success probability for the algorithm as a whole then implies $\epsilon_{\text{target}} = 1.05 \times 10^{-13}$. We must then check that the factory is capable of producing an output of this quality. If the factory is module checked then this '10-10-10' factory has a global error rate $\epsilon_{\text{glo}} = 2.3 \times 10^{-16}$, making this a valid protocol. However the estimate of the global error rate of a block checked factory gives $10^3 \times \epsilon_{BC} \sim 10^{-11}$ so this would not be a valid factory for this task.

5.4.1 Balanced investment

Once we have established that a factory is valid we can calculate the spacetime overhead per magic state that it produces. This is done by: calculating the distance of surface code required at each level of distillation d_i; the length of time in surface

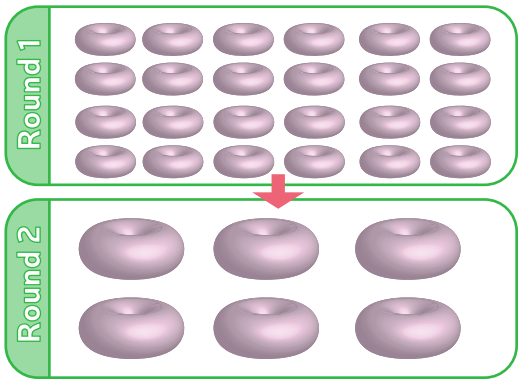


Figure 5.4: The concept of balanced investment. During the initial rounds of distillation smaller surface codes can be used to encode the logical information. The factory requires only logical surfaces of the distance corresponding to the target error rate of the round in which that qubit is involved. This target in each round will depend on the final error target, the protocol being used to achieve it and the error rate of the raw state.

code cycles that each round of distillation will take T_i and the number of logical qubits Q_i, including logical ancillas, required at each level of the factory.

Determining the distance required at each level requires slightly different methods depending on whether module or block checking is used. To obtain the benefits of module checking we cannot make full use of balanced investment at the lower levels, as to do so would inject noise at a rate comparable or greater than the rate of correlated error. We can estimate the total global error output in the output of a $n \to k$ protocol as $\epsilon_{\text{tot}} \sim \epsilon_{\text{glo}} + k\epsilon_{\text{enc}}$, where ϵ_{enc} is the random error rate resulting from size of the logical encoding chosen. As such we determine the distance of the code at the intermediate levels based on a desired error rate ϵ_{enc} to be $0.1 \cdot \epsilon_{\text{glo}}/k$ to ensure that the error due to each qubit's finite encoding is much less than the reduced error $\sim \epsilon_{\text{glo}}/k$ on that qubit. The final output of the factory can then encoded according to the ϵ_{target}.

For a block checked factory (or the original 15-to-1 Reed-Muller protocol) we do not have to worry about 'protecting' correlated errors. This means we can work backwards from our error target to determine an efficient balanced investment of qubits. For a local target error of $p_{\text{top}} = 10^{-14}$ the top level of distillation needs $vP_L(d, \epsilon_{\text{enc}}) < 0.1 \times 10^{-14}$ where v is the spacetime volume of a single block of distillation; which is the number of surface code qubits in the block multiplied by the number of times they undergo d rounds of surface code stabiliser measurements. Again we use a distance corresponding to a lower error rate by a factor of 10 to

suppress any error injected by the logical circuitry above that which is left over after distillation. The distance required of the next level down can then be determined by, in this example $p_{i-1} = \sqrt[3]{p_{\text{top}}/35}$ for the 15-to-1 protocol, which gives distance d_{i-1}. And so on until $p_i > p_{\text{in}}$.

The length of time T_i for each round of distillation can be simply determined by the protocol used and how many times we attempt it before abandoning the round. We assume that measurements can be completed in one time step, and the time scale of the CNOTs, preparation and A gates is dominated by the requirement for d rounds of stabilisation afterwards. Therefore we let the time for each of these be $d \times t_{sc}$. The time taken to implement the distillation protocol in round i is then,

$$\tau_i = \begin{cases} 11t_{\text{sc}} \times d, & \text{round } i \text{ uses Bravyi-Haah;} \\ 12t_{\text{sc}} \times d, & \text{round } i \text{ uses Toffoli.} \\ 13t_{\text{sc}} \times d, & \text{round } i \text{ uses Reed-Muller.} \end{cases} \tag{5.15}$$

5.4.2 Clock-rate zoning

Balanced investment also allows another advantage. In the context of surface code computing, the distance of the code is not only relevant for the spatial dimensions of the computer, but also the execution time. A surface code of distance d must undergo d rounds of parallel stabiliser measurements to protect from measurement errors. As such, the time taken for a logical operation is proportional to d (except those which can be handled in software) and therefore using balanced investment the initial rounds of distillation will take less time. In the hypothetical case that a round of distillation leads to a squaring of the input error rate, this would correspond to a doubling of the code distance required by the next round (by the exponential suppression of error with distance of a sub-threshold surface code). Therefore one can repeat the first round of distillation twice in the time taken for the second round of distillation to be completed. This increases the chance that all the necessary magic states for the second round will have been produced in time for the next round of distillation, without the need to decrease the rate of the factory. As such the time taken for a round to complete in the distillation factory is $T_i = \tau_i t_i$ where t_i is the number of attempts that you allow at each round. In all our simulations, any 'idle time' that the qubits experience is counted towards the spacetime cost.

5.4.3 Numerical simulations

We have therefore arrived at the expression for the full spacetime volume $\mathcal{V}$ in qubits-rounds occupied by a factory

$$\begin{aligned}\mathcal{V}(\epsilon_{\text{in}}, p_{\text{g}}, \mathcal{N}, \{k_i\}, \{t_i\}, P_{\text{suc,alg}}) &= \frac{\sum_{i=1}^{r} Q_i T_i d_i^2}{\prod_i^r k_i \mathcal{P}_{\text{suc},i}} \\ &= \frac{\sum_{i=1}^{r} Q_i \tau_i t_i d_i^3}{\prod_i^r k_i \mathcal{P}_{\text{suc},i}},\end{aligned} \tag{5.16}$$

where k_i labels the magic states protocol used at each round and $\mathcal{P}_{\text{suc},i}$ is the probability that round i of distillation will succeed in producing enough magic states to feed the next round. All rounds must succeed for the factory to successfully output $\prod_i^r k_i$ magic states. As discussed $\mathcal{V}$ is a function of several variables, the raw magic state error rate ϵ_{in}, the error of gate operations p_{g}, the total states required $\mathcal{N}$, the protocol chosen $\{k_i\}$, the repetitions allowed in each round $\{t_i\}$ and the probability with which one wishes the algorithm to succeed $P_{\text{suc,alg}}$.

In practice we calculate $\mathcal{V}$ for one, two and three rounds of distillation using combinations of the Bravyi-Haah, Reed Muller and Toffoli protocols with our proposed implementations, with both block and module checking, and a variety of combinations of t_i. We then search for the most spacetime efficient method of producing either T or Toffoli magic states, given a p_{g}, assuming $\epsilon_{\text{in}} = 0.4p_{\text{g}}$, the number of magic states desired and an arbitrary choice of 90% for $P_{\text{suc,alg}}$.

The results of these simulations suggest that module checking can provide an improvement of a factor of 3 in the spacetime overhead in certain parameter regimes. We also see that in some regions which, as for the cost analysis, correspond to areas in which there is a transition from i to $i+1$ rounds of Bravyi-Haah and it can be in fact detrimental by a small amount. Note that this conclusion is similar to that drawn in the analysis of the cost of a protocol in Section 4.6.1, where the underlying surface code cost was not considered. In fact the benefit of module checking appears to be less when one accounts for the underlying surface code. It should be clear from the discussion of the methods employed that making use of the correlations in the output of the block protocols means employing a less efficient form of balanced investment to ensure that the rate of random error is much less than that of the correlated errors in the intermediate levels of distillation. This serves to wash out some of the benefit of module checking when taking into account the full overhead analysis of a surface code magic state factory.

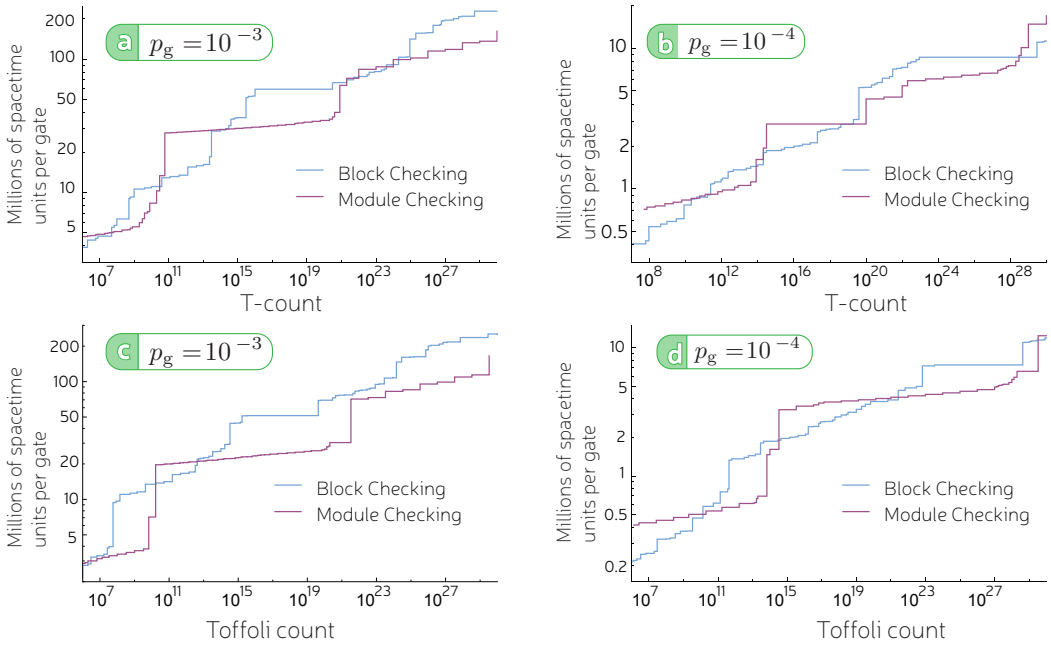


Figure 5.5: Spacetime overheads of producing both (a) & (b) $|A\rangle$ states and (c) & (d) $|\text{Toff}\rangle$ states. Note that module checking is more beneficial when one of the rounds of distillation is Toffoli.

We use the numbers we have generated to estimate the size of a magic state factory required to perform some post-classical factoring tasks using Shor's algorithm. Our results are summarised in Table 5.1, in which we approximate Shor's algorithm as modular exponentiation - as this is by far its most expensive part - and choose the minimum Toffoli gate-count implementation, which has Toffoli-count and -depth equal to $40N^3$ for an N bit number [166]. We then determine the minimum possible spacetime overhead per magic state for this task and also the smallest possible factory — in terms of physical qubits — that can produce all the magic states necessary while keeping up with a time optimal quantum computation [147].

The smallest possible factory is not necessarily the most spacetime efficient factory possible: these factories tend to use larger k blocks which can make the factory formidably vast (cf. Fig. 4.3), but able to produce more qubits in the same time as lower k protocols. However, these may well produce magic states much faster than required by the fastest possible implementation of the algorithm. Fig. 5.6 shows that it is possible to use fewer qubits than required by the time optimal implementation and perform the computation at a reduced speed. Equally it is of course possible to increase the size of the factory to produce magic states at a faster rate. In this case though the extra qubit overhead is effectively wasted, unless the computer is performing many calculations in parallel. We find that all the magic states required

Problem	Magic states required		Spacetime overhead per magic state in qubit-rounds		Physical qubits in factory (**and evaluation time**) required for time-optimal computation			
					$p_g = 10^{-3}$, $t_{meas/ff} = 0.1t_{sc}$		$p_g = 10^{-4}$, $t_{meas/ff} = 0.1t_{sc}$	
	Type	Count	$p_g = 10^{-3}$	$p_g = 10^{-4}$	$t_{sc} = 10^{-3}$ s	$t_{sc} = 10^{-5}$ s	$t_{sc} = 10^{-3}$ s	$t_{sc} = 10^{-5}$ s
1000 bit Shor	Toffoli	$10^{10.60}$	1.41×10^7	5.35×10^5	1.73×10^8 **(6.6 weeks)**	1.73×10^8 **(11 hours)**	6.30×10^6 **(6.6 weeks)**	6.30×10^6 **(11 hours)**
2000 bit Shor	Toffoli	$10^{11.51}$	1.66×10^7	5.71×10^5	2.18×10^8 **(53 weeks)**	2.18×10^8 **(3.7 days)**	6.97×10^6 **(53 weeks)**	6.97×10^6 **(3.7 days)**
4000 bit Shor	Toffoli	$10^{12.41}$	1.94×10^7	6.12×10^5	2.50×10^8 **(8 years)**	2.50×10^8 **(4.2 weeks)**	7.69×10^6 **(8 years)**	7.69×10^6 **(4.2 weeks)**

Table 5.1: The size and time requirements of some examples of magic state factories. We consider an implementation of Shor's algorithm requiring $40N^3$ Toffoli gates, which dominates the overhead. We realise each of these gates using single Toffoli magic state or seven T states in parallel [167], whichever proves optimal. In this algorithm, the Toffoli gates are all sequential, and so using time-optimal methods [147] the fastest possible runtime is $40N^3 t_{meas/ff}$ where $t_{meas/ff}$ is the time taken to make a physical measurement and feed-forward the result to selectively perform a single qubit gate elsewhere in the quantum computer. The number of 'physical qubits in factory' neglects qubit cost associated with measuring surface code stabilisers, and so for many architecture this number will be doubled. The variable t_{sc} is the time taken to perform a single round of the parallel stabiliser measurements of the surface code - a process involving four CNOT gates, two single qubit gates and a measurement. We assume throughout that $t_{meas/ff} = 0.1 \cdot t_{sc}$, which is reasonable for a distributed architecture such as ion traps[15].

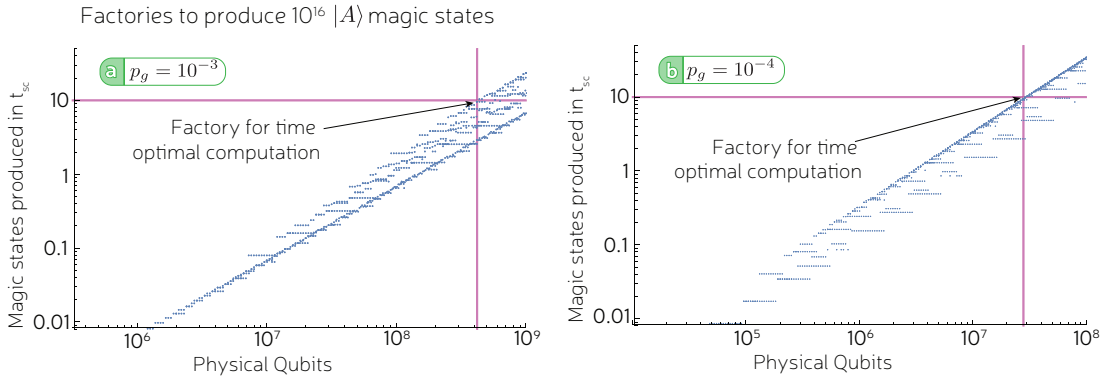


Figure 5.6: To produce $|A\rangle$ magic states at a 'time-optimal' rate the factory must, on average, produce states at a rate equal to the fastest rate at which they can be used in sequence by an algorithm. For these calculations, where we have assumed that $t_{\rm sc} = 10t_{\rm meas/ff}$, this means 10 magic states must be produced on average every $t_{\rm sc}$ to keep up with the time optimal implementation of the algorithm. As this figure makes clear, it is indeed possible to produce a given number of magic states faster (slower) than this using more (fewer) qubits. Each data point represents one possible magic state factory given the input error and target number of 10^{16} $|A\rangle$ magic states, only the factories near the boundary between possible and impossible factories are shown. Not shown in the lower-right of each graph are myriad other possible factories of lower rate and higher overhead. As seen clearly in panel (a) doubling the size of the factory can allow one to more than double the rate of magic state production when the larger factory allows the use of the higher k (and therefore higher rate) block codes. This point is less evident in panel (b) where only two rounds of distillation can be sufficient and the higher k block codes are not necessarily optimal.

for a time-optimal factorisation of a 1000 bit number can be produced with a surface code magic state factory of 6.3 million 'data qubits', if the infidelity of operations on physical qubits is 10^{-4}. This is based on the cost of d^2 physical qubits to store the information in the rotated lattice surface code [52]. Although, for many architectures this number must be doubled to provide ancillas responsible for syndrome extraction. In this case the physical qubit overhead would be ~ 13 million.

A summary of the time and space overheads for some example Shor tasks can be found in Table 5.1. We selected Shor's algorithm as a benchmark as the number of non-Clifford gates required has been well studied and these results have been used in previous analyses. We note that due to the large resource overhead that we have demonstrated, early quantum computers may focus on other problems, particularly in quantum chemistry. A recent analysis of some such problems [168] demonstrates that they are solvable with lower overheads than we find here, albeit that work makes more optimistic assumptions of gate times and fidelities.

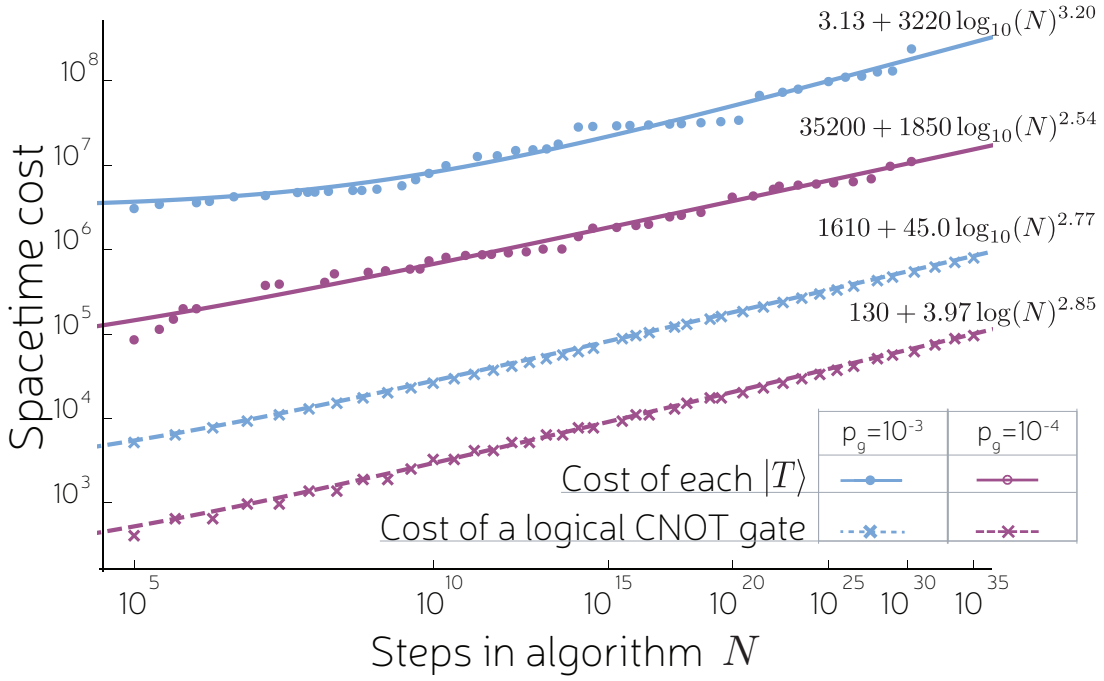


Figure 5.7: The scaling of resource cost in qubit-rounds per magic state is not worse than that of the surface code. This corroborates the predictions of Raussendorf [70] *et al.* and suggests that the asymptotic overhead scaling $\sim O(\log(N)^3)$ of the surface code is applicable to *universal* fault-tolerant computing with gate counts in the regime of practical interest. Lines are fitted functions of the form $\mathcal{V}(p_g, N) = a \log(N)^b + c$

Of significant interest is how the cost of magic state distillation compares to the surface code overhead. Raussendorf *et al.* (see Sec. 6.2 of Ref. [70]) were the first to note that using balanced investment in a magic state distillation leads to a constant factor overhead compared to the CNOT gate. Our numerical results in Fig. 5.7 show a ratio between T-gate cost and CNOT cost in the range $\sim 150 - 310$ when $10^{10} < N < 10^{30}$. This ratio is much smaller than estimated by Raussendorf *et al.* who did not make use of Bravyi-Haah distillation routines. A similar ratio can be extracted from the data tables provided in Ref. [122], though the numbers are not directly comparable. For instance, we also count the spacetime volume due to idle qubits, while they wait for distillation circuits to succeed.

The work of Bravyi and Haah predicted a much greater improvement because they quantifed cost by the expected conversion efficiency of raw to high-fidelity magic states. In a fully costed analysis, as we perform here, error correction costs overwhelm and dominate the cost of magic state factories. We saw that an efficiently designed factory using balanced investment is entirely limited by the surface code cost, and so refinement in distillation protocols can only offer constant factor improvements.

It is likely that this analysis represents an overestimate of the spacetime overhead of the implementations of the distillation circuits we describe. We have assumed the need for d rounds of surface code measurements after every two-qubit gate. However, it is not clear that this is necessary when performing transversal gates. In the implementation of Bravyi-Haah (see Fig.5.1) it may prove feasible to perform d rounds of error correction only after, say, the completion of each of the 4 steps described - reducing the time overhead from $11t_{\rm sc}d$ to $4t_{\rm sc}d$ [169]. The overhead analysis here could thus be further developed by considering the effect on the performance of the underlying surface code if multiple transversal operations are performed between rounds of error correction.

Recent ideas including 3D gauge colour codes [129, 131] do not require magic states. But they have their own costs. For 3D gauge colour codes, spatial overheads scale as $\sim O(\log(N)^3)$ and time overhead as $O(1)$. Using balanced investment and surface codes we see similar asymptotic scaling of resources. However, current evidence indicates an order of magnitude worse phenomenological threshold for colour codes [170, 171] compared to the phenomenological threshold for the surface code. Although a full circuit-based threshold has not yet been determined, it is unlikely to challenge that of the surface code due to the higher weight stabilisers required. This points towards 3D colour codes requiring physical error rates of below 0.1%. Resource costs are heavily influenced by proximity to threshold, and so 3D colour codes seem to require significantly lower physical error rates before they start can compete with surface codes augmented by magic states. Therefore, with current technology and fidelities, known schemes for avoiding magic states might be a false economy. An additional benefit of the magic state paradigm is that it can also eliminate the additional burden of gate-synthesis costs by preparing exotic magic states [145, 146, 172–175], which we will be exploring in the next chapter.

5.5 Conclusion

We have seen that the surface code supplemented by magic states could achieve a magic state factory for fault-tolerant quantum computation with on the order of 10-100 million physical qubits, with some reasonable assumptions made about the error rates achievable in these systems. If we revisit the orbital probe scheme described in Chapter 2 then a 7 million data qubit device required to achieve a 2000 bit Shor algorithm with physical errors at the $\sim 10^{-4}$ level and an interqubit spacing of 400 nm would have a size of roughly 1 mm^2. Of course this neglects the extra qubit cost

associated with the lack of long range interaction and only counts the physical qubits in the magic state factory – so this is very much a lower bound on both physical size and qubit cost, but it is instructive. To address the non-magic state factory qubits, we consider a total of $2N$ computational logical qubits (i.e. the qubits on which the computation is taking place, not those producing magic states) that are required for the implementation of Shor's algorithm that uses $40N^3$ non-Clifford Toffoli gates. In this case, as the distance of the surface code being used at the top level is $d = 16$, this part of the computer requires ~ 1 million more data qubits, so roughly an extra 0.4 mm^2 of device. Assuming one measurement or probe qubit for each data qubit, that is a total of ~ 16 million physical qubits. The majority of the quantum computer in this case, and many others, is the magic state factory.

The next chapter examines how magic state techniques can be used to provide gates from higher levels of the Clifford hierarchy.

Chapter 6

Magic state methods for small-angle rotations

In this chapter an improved method of generating magic states for higher levels of the Clifford hierarchy is summarised. The author contributed to the original compression of the distillation protocol and the analysis of noise propagation in the circuits used which are presented here, while the numerical simulations summarised and the invention of 'magic state dilution' are credited to Earl Campbell with whom the author collaborated for this research. For completeness, this thesis includes further numerical simulations performed by Earl Campbell as Appendix A as well as a full description of magic state dilution adapted directly from Ref. [176] upon which the chapter is based.

In the previous chapter we took an in-depth look at the resources required to implement operations from the third level of the Clifford hierarchy (cf. Section 4.1.1). We know that given the ability to do one of these operations in addition to the Clifford operations we can efficiently simulate any quantum algorithm. However we need not restrict ourselves necessarily to such a gate set. Rather than synthesising gates from higher levels of the Clifford hierarchy from our Clifford+T set, we can directly distill magic states that fault-tolerantly perform the gates [173]. The question of which approach is better is simply to a matter of determining which of these processes is the less resource intensive.

6.1 Distilling small angle magic states

Logical small angle rotations are important in many quantum algorithms and can be fault-tolerantly implemented given a supply of an unconventional species of magic state [173]. In this chapter we investigate the unitaries $R_\ell := \exp(i\theta_\ell Y)$ with $\theta_\ell =$

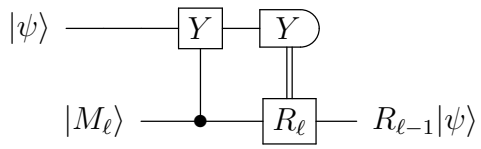


Figure 6.1: State-injection using a magic state $|M_\ell\rangle$ to perform a non-Clifford gate R_ℓ. If the Y-basis measurement outcome is "+1", then a correction $R_{\ell-1}$ is needed.

$\pi/2^\ell$. For $\ell \geq 3$, the gates are non-Clifford and belong to the ℓ^{th} level of the Clifford hierarchy. The R_3 gate is non-Clifford and equivalent to the T gate. Thus to be performed in a quantum code offering protection only to Clifford operations they require extra overhead to implement. Let us define some Hermitian operators in the Clifford hierarchy, $H_\ell := R_\ell X R_\ell^\dagger = R_\ell^2 X = R_{\ell-1} X$. Note that $H_\ell \in \mathcal{C}_{\ell-1}$ and H_3 is equivalent to the Hadamard. We define the magic states that are eigenstates of the H_ℓ operators, so that

$$|M_\ell\rangle = H_\ell |M_\ell\rangle = R_\ell |+\rangle, \tag{6.1}$$

$$\left|\bar{M}_\ell\right\rangle = (-H_\ell)\left|\bar{M}_\ell\right\rangle = R_\ell |-\rangle, \tag{6.2}$$

which we refer to as ℓ^{th} level magic states. Such resources can be used to inject R_ℓ rotations into circuits as shown in Fig. (6.1). Therefore, one can accomplish small angle rotations with a 'cost' (cf. Chapter 4) that depends on the cost of distilling high-fidelity $|M_\ell\rangle$ magic states.

In contrast to the earlier protocols we have investigated, the distillation protocols in this chapter take as input three different kinds of noisy input state: $|M_3\rangle$, $|M_\ell\rangle$ and the $|M_{\ell-1}\rangle$ and produce distilled $|M_\ell\rangle$ states. In contrast, gate synthesis methods prescribe distilling just $|M_3\rangle$ (equivalently $|A\rangle$) states, as in Chapter 4 & 5 and finding a gate sequence $R_\ell \approx C_1 T C_2 \ldots T C_n$ in terms of T gates and Cliffords C_j. The cost being then total cost of distilling the number of $|A\rangle$ (equivalent to $|M_3\rangle$) magic states equal to the T-count of that gate sequence. There are many gate synthesis algorithms for finding gate sequences and here we consider the Selinger and Ross [177] algorithm (SR) and probabilistic quantum circuits with fallback (PQF) [178]. SR is provably optimal or unitary synthesis. PQF uses ancilla and measurements to achieved the best known performance, about a factor 3 more efficient than SR.

Duclos-Cianci and Poulin [174] modified a protocol proposed by Meier, Eastin and Knill [119] to prepare magic states that provide small angle rotations. This protocol takes on average 2 $|M_\ell\rangle$ states, 16 $|M_3\rangle$ states, and one 'pivotol' $|M_{\ell-1}\rangle$ state and output 2 $|M_\ell\rangle$ of higher fidelity, quadratically suppressing the error rate on the input

$|M_\ell\rangle$ states and $|M_3\rangle$ magic states. This protocol (referred to as the DP_ℓ protocol) was shown to be superior to gate synthesis in many instances.

Here we report comparison of the cost of distilling $|M_\ell\rangle$ states directly using a protocol based on the compression of the DP_ℓ distillation procedure in Ref. [174], versus distilling $|M_3\rangle$ states and then synthesising the gate $|R_\ell\rangle$. It was observed that certain syndrome information gathered in the original Duclos-Cianci and Poulin protocol was not necessary to achieve quadratic error suppression. Removing the measurements that extracted this information from the circuit allows the circuit to be compressed in such a way that multiple R_3 and $R_3^\dagger$ gates, each of which requires an input resource magic state $|M_3\rangle$, can be brought together to either cancel out to $\mathbb{1}$ or create an R_2 gate, which is Clifford and therefore also a free resource. This circuit reduction halves the number of $|M_3\rangle$ states needed to distill one $|M_\ell\rangle$ state. Crucially, this compression maintains the error protection of the original circuit. We call the improved protocol the MEK_ℓ protocol, as it proves an apt generalisation of the protocol of Meier, Eastin and Knill [119] which distilled $|M_3\rangle$ from other $|M_3\rangle$ magic states and corresponds to MEK_3 in our protocol.

6.2 Quantifying resource cost

In this section, the work of Meier, Eastin, and Knill (MEK) [119] is summarised, upon which Duclos-Cianci and Poulin built their construction. The MEK protocol takes 10 input magic states and with some probability outputs 2 magic states. Therefore, we will call MEK a $10 \to 2$ protocol. We will also label the species of magic states, involved so MEK is a $10_3 \to 2_3$ protocol where the subscripts indicate that MEK both inputs and outputs 3rd level magic states. We use DP_ℓ to label the Duclos-Cianci and Poulin protocol for distilling ℓ^{th} level magic states. Each round of DP_ℓ consumes a cocktail of different input magic states, and we describe it as a

$$\left\{16_3, 2_\ell, 1_{\ell-1}, \left(\frac{1}{2}\right)_{\ell-2}, \left(\frac{1}{4}\right)_{\ell-3}, \ldots \left(\frac{1}{2^{\ell-3}}\right)_3\right\} \to 2_\ell \tag{6.3}$$

protocol. Again, subscripts show what level magic state is used. We show the expected number of inputs required per attempt, which is sometimes a fraction, taking into account the probabilistic nature of injection and the fact this operation itself requires a magic state if $\ell > 3$. The fractional sequence terminates at the third level, because lower levels correspond to Clifford resources and are considered free. The DP_ℓ protocol was originally presented as a direct generalisation of MEK, but in the case of $\ell = 3$ the DP_3 protocol is a $18_3 \to 2_3$ protocol and so actually needs almost

twice as many input states as MEK. Our compression of the circuit presented in 6.4 is a truer generalisation, so it is called MEK_ℓ and is a

$$\left\{8_3, 2_\ell, 1_{\ell-1}, \left(\frac{1}{2_{\ell-2}}\right), \left(\frac{1}{4}\right)_{\ell-3}, \ldots \left(\frac{1}{2^{\ell-3}}\right)_3\right\} \to 2_\ell \tag{6.4}$$

protocol.

Both MEK and DP_ℓ make use of a simple 4 qubit code. We use E, short for encoder, to denote the Clifford circuit that brings qubits into the codespace and acts on pairs of Pauli operators as

$$(Z_1, X_1) \to (Z_1Z_2Z_3Z_4, X_1X_3X_4), \tag{6.5}$$
$$(Z_2, X_2) \to (X_1X_2X_3X_4, Z_2), \tag{6.6}$$
$$(Z_3, X_3) \to (Z_1Z_4, X_1X_3), \tag{6.7}$$
$$(Z_4, X_4) \to (X_1X_4, Z_1Z_3). \tag{6.8}$$

Preparing the first two qubits in the state $|0\rangle$ and running the encoder will yield the code stabilised by $Z_1Z_2Z_3Z_4$ and $X_1X_2X_3X_4$. The logical state in the codespace is determined by the initial states of the last two qubits.

6.3 Uncompressed DP_ℓ protocol

The protocol DP_ℓ is illustrated in Fig. (6.2d). Throughout, we label the top wire as the control qubit c, and the subsequent qubits are labelled 1 to 4. The protocol can be summarised

1. Prepare qubits c, 1 and 2, in stabiliser states $|+\rangle$, $|0\rangle$ and $|0\rangle$, respectively;
2. Prepare qubits 3 and 4 in noisy $|M_\ell\rangle$ states with ϵ_ℓ error;
3. Perform the circuit gates shown in Fig. (6.2d): the R_3 gates are achieved by state injection using eight $|M_3\rangle$ states, and $R_{\ell-1}$ is achieved with state injection resulting in overall $\eta_{\ell-1}$ error;
4. Qubit c is measured in X basis, and we continue if the outcome is "$+1$" and otherwise declare FAILURE;
5. Qubits 1 and 2 are measured in Z basis, and we declare SUCCESS if the outcome is "$+1$" and otherwise declare FAILURE;
6. If successful, qubits 3 and 4 are output as $|M_\ell\rangle$ states of higher fidelity.

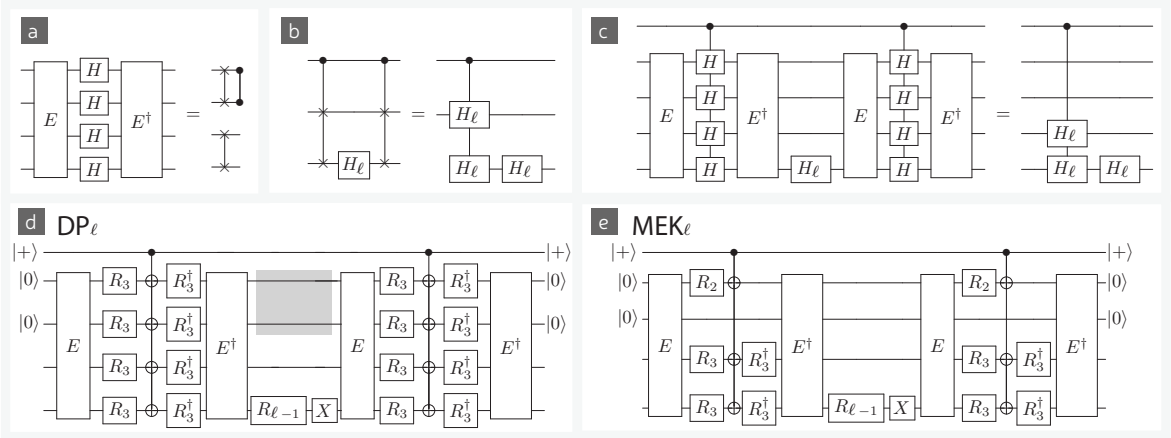


Figure 6.2: (a,b,c) Circuit identities used to construct distillation protocols. (d) a non-compressed distillation circuit. Adding $|0\rangle\langle 0|$ measurements and preparations within the grey box gives the DP$_\ell$ protocol. (e) the compressed distillation circuit describing MEK$_\ell$. In both (d) and (e) noisy $|M_\ell\rangle$ states are input on the bottom two circuits lines (labelled qubits 3 and 4 in the main text) and all R_ℓ gates with $\ell \geq 3$ are approximately implemented by injection of a noisy $|M_\ell\rangle$ states. [Figure credit: Earl Campbell].

Note that step 3 can include some additional postselected measurements highlighted in Fig. (6.2d). These measurements were included by Duclos-Cianci and Poulin and detect some additional errors, but even without these measurements the protocol quadratically suppresses noise. It was by removing these measurements that the compression of the circuit is achieved.

The codespace provides a SWAP gate between the two logical qubits that can be implemented by a transversal Hadamard gate $H \otimes H \otimes H \otimes H$. More generally, one can verify that $E(H \otimes H \otimes H \otimes H)E^\dagger$ acts as shown in Fig. (6.2a). We see that this circuit implements a swap and a swap combined with a controlled-phase gate. Furthermore, implementing controlled Hadamards within the encoding will realise controlled versions of the swap and phase-swap. Next, we note that for any Hermitian unitary, such as H_ℓ, we have that conjugation by controlled-swaps will produce a controlled $H_\ell \otimes H_\ell$ as shown in Fig. (6.2b). We combine these properties to get the identity of Fig. (6.2c). An ancilla on the control of the controlled $H_\ell \otimes H_\ell$ is used to measure the $H_\ell \otimes H_\ell$ observable. If the desired magic state is an eigenstate of H_ℓ, then measuring $H_\ell \otimes H_\ell$ allows us to detect a single error between two noisy $|M_\ell\rangle$ states input on the bottom two circuit lines.

Although the controlled-Hadamard rotations and H_ℓ rotation may be non-Clifford, noisy magic states can be used to implement these operations. To see this recall that

$H = R_3 X R_3^\dagger$, and so a control-Hadamard can be implemented by a control-X gate sandwiched between R_3 and $R_3^\dagger$. In turn, R_3 and $R_3^\dagger$ can each be implemented at the cost of a single $|M_3\rangle$ magic state. Given noisy $|M_3\rangle$ magic states, we implement noisy R_3 gates. The R_3 gates are performed on qubits within the code, so we will detect a single error in any $|M_3\rangle$ magic states. When using state injection, if a magic state carries an error, then it will result in $Y \cdot R_3$ instead of R_3, and so there is an additional Y acting on one of the four qubits of the code. Inside the encoding, the state is an eigenstate of $X \otimes X \otimes X \otimes X$, but a Y error will cause the state to become an eigenstate of $-X \otimes X \otimes X \otimes X$. At the end of the circuit, we decode and measure, which is equivalent to measuring the $X \otimes X \otimes X \otimes X$ observable. Since we postselect on all $+1$ outcomes, any error on a single $|M_3\rangle$ will be detected. In contrast, the central $R_{\ell-1}$ rotation occurs outside the protection of the codespace therefore, the protocol cannot detect a single qubit error on this gate. This demands that this rotation must be high fidelity and so we call it the *pivotal rotation*. Nevertheless, we can construct good distillation protocols provided magic states used in the pivotal rotation have already been distilled to a much higher fidelity than all other elements of the circuit. These high fidelity resources will be costly, but the protocol remains efficient because resources for performing $R_{\ell-1}$ are less valuable than the $|M_\ell\rangle$ states we are distilling.

6.4 Compressed MEK$_\ell$ protocol

The circuit used by Duclos-Ciani and Poulin can be compressed into Fig. (6.2e) when the internal measurements of qubit 1 and 2 are removed. This allows us to cancel several R_3 rotations to reduce the number of magic states consumed. The steps of the protocol roughly follows those listed in the previous section, except step 3 now uses the circuit of Fig. (6.2e), and we use only 8 magic states to inject the R_3 gates, not 16. Even when compressed the noisy R_3 gates still occur within the four-qubit error correction code, and so we still expect to detect the failure of any single R_3 gate.

To compress the circuit we start with the protocol in Fig. (6.2d). Taking Fig. (6.2d), we identify a subcircuit V shown in Fig. (6.3a) inside a shaded box. Next we establish two properties of V, which are illustrated in Figs. (6.3b) and (6.3c). The V circuit is simply $V = E \exp(i\theta_{\ell-1} Y_4) X_4 E^\dagger$, and using Eqs. (6.5) we see that $V = \exp(-i\theta_{\ell-1} Y_1 Z_3 X_4) Z_1 Z_3$. This acts trivially on the second qubit and control qubit, giving the circuit identity of Fig. (6.3b). We also notice that $VY_1 = -Y_1 V$ and so $\exp(i\theta_3 Y_1) V \exp(-i\theta_3 Y_1) = \exp(i2\theta_3 Y_1) V$. Using $2\theta_3 = \theta_2$, we conclude

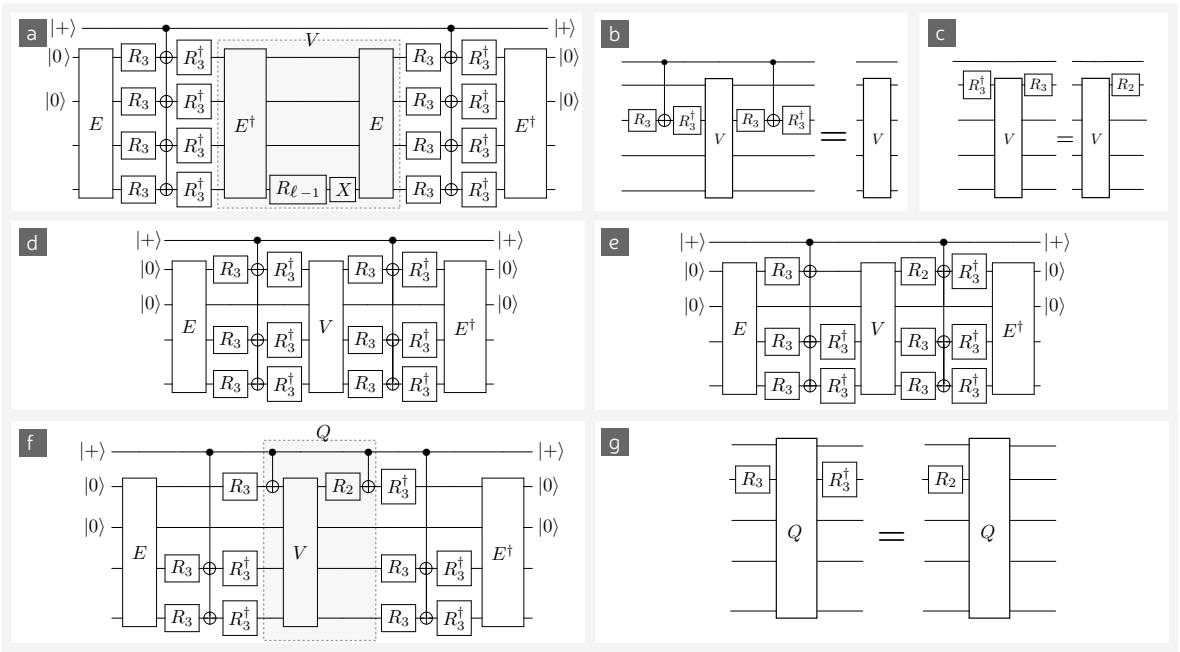


Figure 6.3: Circuit identities and reductions used to obtain MEK_ℓ. (a) circuit before compression, with subcircuit V labelled. (b) and (c) show properties of V. (d) shows partially compressed circuit after applying identity (b). (e) further compressed circuit after applying identity (c). In (f) we identify subcircuit Q, with (g) showing a property of Q. Applying (g) to (f) yields the final compressed circuit, shown in Fig. (6.2e). [Figure credit: Earl Campbell].

$\exp(i\theta_3 Y_1)V\exp(-i\theta_3 Y_1) = \exp(i\theta_2 Y_1)V$. Recall that $\exp(i\theta_2 Y_1)$ is the R_2 gate acting on qubit 1, and so we have the identity shown in Fig. (6.3c). Applying the identity Fig. (6.3b), shows that Fig. (6.3d) is equivalent to the original circuit. So far 4 non-Clifford gates have been removed by bring them together to make larger angle rotations in the Clifford group. Next, we apply the identity of Fig. (6.3c), to obtain Fig. (6.3e). Since R_2 is Clifford, we have removed 2 further non-Cliffords from the circuit.

Next, we group together a new collection of circuit elements Q shown in dashed box of Fig. (6.3f). Algebraically, Q is

$$\begin{aligned} Q &= C^X_{c,1}\exp(i\theta_2 Y_1)VC^X_{c,1} \\ &= C^X_{c,1}\exp(i\theta_2 Y_1)\exp(-i\theta_{\ell-1}Y_1Z_3X_4)Z_1Z_3C^X_{c,1} \\ &= \exp(i\theta_2 Z_cY_1)\exp(-i\theta_{\ell-1}Z_cY_1Z_3X_4)Z_cZ_1Z_3 \end{aligned} \tag{6.9}$$

where we have used C^X for CNOT gates. From this expression for Q we can again see that $QY_1 = -Y_1Q$, which entails that $\exp(-i\theta_3 Y_1)Q\exp(i\theta_3 Y_1) = Q\exp(i2\theta_3 Y_1) =$

$Q\exp(i\theta_2 Y_1)$. This demonstrates the identity of Fig. (6.3g). Applying this identity to Fig. (6.3f), yields the final representation of MEK_ℓ as shown in Fig. (6.2e).

6.5 Measuring performance

This section introduces several definitions and notations used to describe the performance of protocols, and quantify their cost.

6.5.1 Quantifying noise

If ρ is a noisy $|\psi\rangle$ state then we say it has error rate ϵ where $\epsilon := \frac{1}{2}||\rho - |\psi\rangle\langle\psi|||_{\text{tr}}$ and $||X||_{\text{tr}} = \text{tr}[\sqrt{XX^\dagger}]$ is the trace norm as used in Chapter 3. In this appendix we assume diagonal noise so that ρ is diagonal in the same basis as $|\psi\rangle$, and for such states one can show $\epsilon = 1 - \langle\psi|\rho|\psi\rangle$. This is valid if we allow ourselves to twirl, as in the distillation procedures for T magic states, but for ℓ level magic states we twirl through an operation from the $(\ell-1)^{\text{th}}$ level of the hierarchy. The protocols use a cocktail of input resources. We use ϵ_ℓ and ϵ_3 to denote the input error rates of the noisy $|M_\ell\rangle$ and $|M_3\rangle$ states used.

When a R_ℓ gate fails due to noise, it is followed by an additional Y rotation. There are 8 locations that errors can occur on R_3 gates and we define a binary vector $x = (x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8)$ that records whether a Y error occurs at a particular R_3 gate. We use the binary variable $y = 0, 1$ to track an error in the pivotal rotation $R_{\ell-1}$. When the $R_{\ell-1}$ gate fails, we have a rotation $H_\ell(iY)$ rather than H_ℓ. Notice the complex phase i, which makes no physical difference to the unitary. However, aspects of the proof rely on H_ℓ being Hermitian, and the additional phase keeps it Hermitian. The resulting random circuit is shown in Fig. (6.4a). [t] We pull some Y noise operators through the control phase gates, which can cause Z noise on the control as shown in Fig. (6.4b). The central portion of the circuit now contains no noise terms (except y) and so we can replace it with a logical control $(H_\ell \otimes H_\ell)$ or its equivalent when $y = 1$. This yields Fig. (6.4c).

Next, we pull the Y noise backwards through the encoder E. We already know how E acts on Pauli operators from which we conclude that the inverse action satisfies

$$E^\dagger : Y_3 \to Y_1 X_2 Z_3 Z_4, \tag{6.10}$$

$$E^\dagger : Y_4 \to Y_1 X_2 X_3 X_4. \tag{6.11}$$

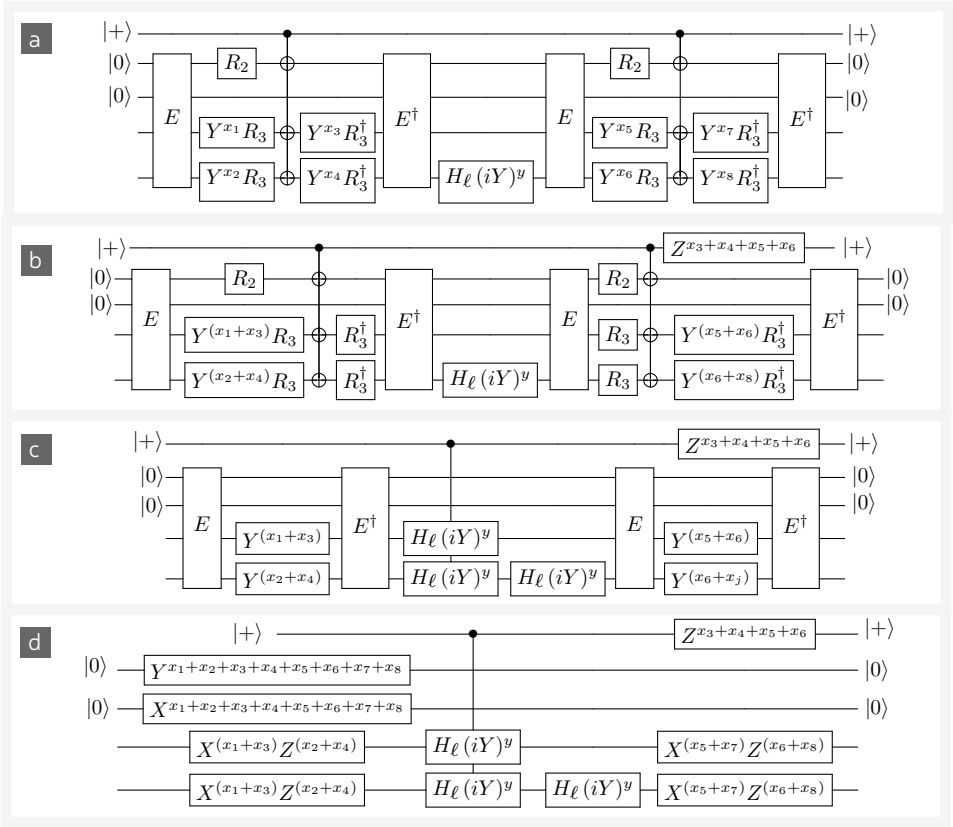


Figure 6.4: Propagation of noise terms around distillation circuit.

Similarly random unitaries $Y_3^\alpha Y_4^\beta$ map as

$$E^\dagger : Y_3^\alpha Y_4^\beta \to Y_1^{\alpha+\beta} X_2^{\alpha+\beta} (Z_3^\alpha X_3^\beta)(Z_4^\alpha X_4^\beta) \tag{6.12}$$

Applying this rule to our circuit yields Fig. (6.4d).

From Fig. (6.4d), we see that to obtain the correct measurement outcome on qubit 1 or 2 requires that

$$|x| := x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 = 0 \pmod 2, \tag{6.13}$$

which detects any single error on the R_3 gates. This shows that the error correction properties of our original circuit have survived the compression.

From this circuit we can deduce the leading terms presented in Eqs. 6.14 for the success probability of the distillation protocol and the output error on the distilled magic states. This calculation is detailed in Appendix A, with justifications for the expressions found in Eqs. 6.14 which are then confirmed by symbolic simulation of the circuit in Mathematica.

6.5.2 Distillation cost

The output error from MEK_ℓ is always measured as the error on a single output qubit (ignoring correlations) and we found (see Appendix A)

$$\delta_\ell(\epsilon_3, \epsilon_\ell, \eta_{\ell-1}) \sim 8\epsilon_3^2 + \epsilon_\ell^2 + \frac{1}{4}\eta_{\ell-1} + \ldots, \tag{6.14}$$

$$P_{\text{suc}}(\epsilon_3, \epsilon_\ell, \eta_{\ell-1}) \sim 1 - 8\epsilon_3 - 2\epsilon_\ell - \frac{1}{2}\eta_{\ell-1} + \ldots. \tag{6.15}$$

Within the context of a single distillation round the performance is independent of the level ℓ, and is solely a function of the noise of the input states ϵ_ℓ, ϵ_3 and η. When we ask how much the input states cost, we find this can increase with ℓ. Next, we consider many distillation rounds and combine all performance metrics into a single quantity, the expected resource cost $\mathfrak{C}(M_\ell, \delta)$ to distill a $|M_\ell\rangle$ state of δ error. Lower δ can require more rounds of distillation, which drives up costs. For our protocol we use that the cost is

$$\mathfrak{C}(M_\ell, \delta_\ell) = \frac{2\mathfrak{C}(M_\ell, \epsilon_\ell) + 8\mathfrak{C}(M_3, \epsilon_3) + \mathfrak{C}(R_{\ell-1}, \eta_{\ell-1})}{2P_{\text{suc}}(\epsilon_3, \epsilon_\ell, \eta_{\ell-1})}, \tag{6.16}$$

where δ_ℓ obeys Eq. (6.14). In our analysis, we optimise the cost over many thousands of possible combinations of input resources by brute force search. Notice that we capture the cost of the pivotal rotation as $\mathfrak{C}(R_\ell, \eta_\ell)$, which will in turn depend on the cost of magic states used to implement the rotation as

$$\mathfrak{C}(R_\ell, \eta_\ell) = \mathfrak{C}(M_\ell, \epsilon_\ell) + \frac{1}{2}\mathfrak{C}(R_{\ell-1}, \eta_{\ell-1}), \tag{6.17}$$

where

$$\eta_\ell = \frac{1}{2}\epsilon_\ell + \frac{1}{2}(\epsilon_\ell(1 - \eta_{\ell-1}) + (1 - \epsilon_\ell)\eta_{\ell-1}). \tag{6.18}$$

For the lowest non-Clifford levels, $\mathfrak{C}(M_3, \delta_3)$ are found by minimising over different combinations of protocols including Bravyi-Haah [120], MEK_3 [119] and using the 15 qubit Reed-Muller code [41]. Recall Clifford operations are considered free and perfect so that $\mathfrak{C}(M_2, 0) = 0$ and $\mathfrak{C}(R_2, 0) = 0$. Throughout we assume that raw initial non-Clifford states can be prepared with a fidelity that is independent of ℓ, and these have unit cost, so that $\mathfrak{C}(M_\ell, \epsilon_{\text{raw}}) = 1$. This last assumption is warranted in light of the results of Li [165].

6.5.3 Gate-synthesis cost

In general, gate synthesis techniques have an inherent cost that we denote as $\mathfrak{T}(U, \epsilon_{\rm GS})$, where $\epsilon_{\rm GS}$ is the precision of the synthesis. This T-count assumes perfect $|M_3\rangle$ magic states are available. Also accounting for the cost of distilled $|M_3\rangle$ states, the full cost is

$$\mathfrak{C}^{GS}(R_\ell, \eta_\ell) = \mathfrak{T}(R_\ell, \epsilon_{\rm GS}) \cdot \mathfrak{C}(M_3, \epsilon_3) \tag{6.19}$$

where

$$\eta_\ell \simeq \epsilon_{\rm GS} + \mathfrak{T}(R_\ell, \epsilon_{\rm GS}) \cdot \epsilon_3. \tag{6.20}$$

Notice that gate synthesis has an inherent error $\epsilon_{\rm GS}$ and will also fail if one of the $\mathfrak{T}$ noisy states fail. In our analysis, we present data points for combinations of actual instances of gate synthesis using the SR protocol. The SR data is obtained using the Gridsynth package [1]. While SR is optimal for unitary synthesis, lower overheads can be using PQF, which uses ancilla, measurements and feed forward. For PQF, we use the approximation

$$\begin{aligned}\mathfrak{T}^{PQF}(U, \epsilon_{\rm GS}) \simeq &\log_2(\sqrt{2}\epsilon_{\rm GS}^{-1}) \\ &+ 4\log_2(\log_2(\sqrt{2}\epsilon_{\rm GS}^{-1}) + 1.187,\end{aligned} \tag{6.21}$$

which is the lowest currently known overhead for gate synthesis.

6.6 Results of resource comparison

It was found that the novel protocol we introduce is $\sim$ 20 times less costly than leading gate synthesis techniques for modest $\ell = 2, \ldots, 7$, this is shown in Figure 6.5. Approximately 1/2 of this improvement is due to the circuit compression, while the original uncompressed protocol DP_ℓ is responsible for roughly a factor 10 improvement. It can also be seen that as ℓ increases, so MEK_ℓ becomes more costly, so for larger ℓ the benefits over gate synthesis is lost. This dependence of cost on ℓ is due to the increasing cost of producing pre-distilled 'pivotal' magic states in the protocol for increasing ℓ. However, for large ℓ the magic state $|M_\ell\rangle$ can be very close to a stabiliser state and this fact can be used to reduce resources substantially, as we will now see. We fix $\epsilon_\ell = \epsilon_3 = \epsilon$ and $\eta_{\ell-1} = \epsilon^2$. We remark that $\eta_{\ell-1}$ must be set significantly lower than other errors as the protocol can not detect noise in the pivotal rotation. We consider the cases of raw error rates of $\epsilon_{\rm raw} = 0.01$ and $\epsilon_{\rm raw} = 0.001$. The case of higher

[1] the Gridsynth package is available at http://www.mathstat.dal.ca/ selinger/newsynth/

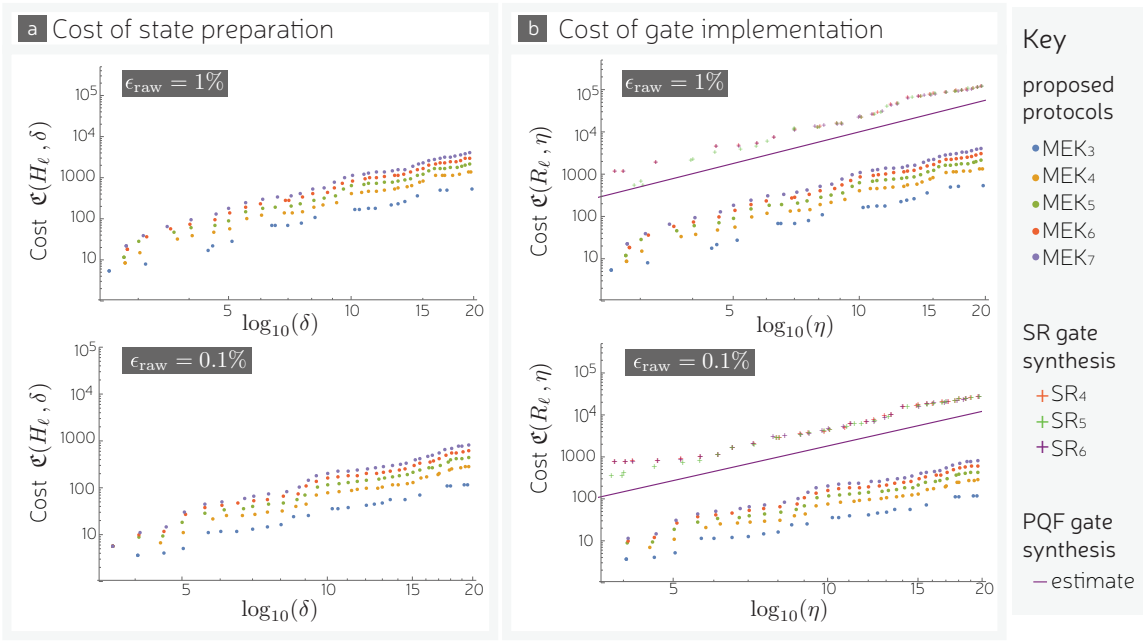


Figure 6.5: The resource cost of MEK$_\ell$. (a) Shows cost $\mathfrak{C}(|M_\ell\rangle, \delta)$ of preparing a magic state $|M_\ell\rangle$ at error rate δ. (b) Shows cost $\mathfrak{C}(R_\ell, \eta)$ of performing a non-Clifford R_ℓ at error rate η. For comparison, (b) also shows the cost of gate-synthesis using the protocol of Ref. [126]. Both (a) and (b) use initial error rates of $\epsilon = 0.01$ and $\epsilon = 0.001$ for axial rotations of angle $\theta_\ell = \pi/2^\ell$ with $\ell = 3, 4, \ldots 7$. [Figure credit: Earl Campbell]

raw noise is an important benchmark as it has been widely studied [119, 120, 174], and here we see improvements over gate synthesis. For instance, at $\delta = 10^{-15}$ we find PQF is ~ 22.5 times more costly than MEK$_6$. However, the lower raw noise regime $\epsilon_{\text{raw}} = 0.001$ is in many ways more interesting. For instance, at target error rate $\delta = 10^{-15}$ we find PQF is ~ 24 times more costly than MEK$_6$, which is a larger factor than in the high noise regime. This widening gap between gate synthesis and MEK$_\ell$ is seen for all ℓ and δ. This increase in gap is intuitive because the distillation cost drops with ϵ_{raw}, but the T-count of gate synthesis is independent of ϵ_{raw}.

While the high noise regime is widely studied, it can be argued that the lower noise regime is also more realistic, and throughout the latter half of this thesis it is noise rates $\leq 10^{-3}$ that have been focussed on, as to prevent very large overheads physical gates must be comfortably below the threshold (the effect of reducing the low level error on the overhead can be seen in Chapter 5), and so we assume all physical gates have infidelities well below 1%. Because preparing raw magic states will involve several physical gates, it had often been assumed that ϵ_{raw} will be higher than the physical gate error rate. However as previously discussed, Li [165] has shown that we

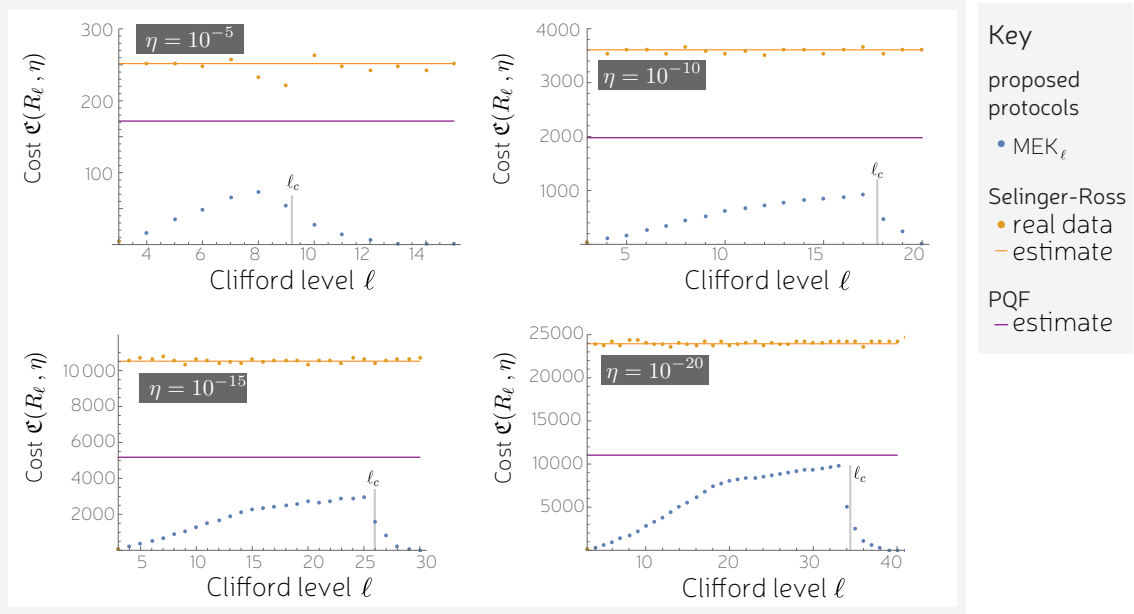


Figure 6.6: The resource cost R_ℓ rotations using MEK$_\ell$ distillation combined with magic state dilution with $\epsilon_{\text{raw}} = 0.1\%$. Each plot is for a different final error rate η, and is against the Clifford level ℓ beginning with the first nonClifford level $\ell = 3$. We compare perform against gate-synthesis methods, which perform mostly independently of ℓ. For the Selinger-Ross (SR) gate synthesis protocol [126], we show exact data extracted from the gridsynth application and also the analytically proven typical performance. We also show the typical performance for probabilistic quantum circuits with fallback (PQF) [178]. All results here assume $\epsilon_{\text{raw}} = 0.001$. [Figure credit: Earl Campbell]

can probabilistically prepare raw magic states at infidelities of about half the physical CNOT gate error, assuming CNOT failure is the dominant noise mechanism. This indicates that $\epsilon_{\text{raw}} = 0.001$ is a feasible regime, more plausible than $\epsilon_{\text{raw}} = 0.01$. As such, following subsections focus on the low-noise regime.

An additional technique called 'magic state dilution' can achieve far greater advantage over gate synthesis for large ℓ. It achieves this by probabilistically mixing $|M_\ell\rangle$ with a stabiliser state to produce a noisy $|M_{\ell+1}\rangle$ state. In some regimes when ℓ is sufficiently large, the resulting noisy $|M_{\ell+}1\rangle$ magic state can have less noise than the resource $|M_\ell\rangle$ state used. Figure 6.6 now compares the cost of using the optimal combination of distillation *and* dilution (found by brute force search) versus gate synthesis. For modest ℓ distillation is superior to gate synthesis, but this advantage decreases with increasing ℓ, in a middle regime of large but not very large ℓ dilution is used in combination with distillation and the rate at which the advantage decreases with ℓ slows. Above some critical value ℓ_c cost is decreased exponentially, it is here

that dilution is used exclusively as it achieves the aforementioned reduction in error rate when used. Here the advantages over gate synthesis are substantial.

6.7 Conclusion

As was performed in Chapter 5 a true assessment of the benefits of small angle magic states versus gate synthesis needs a full spacetime overhead calculation. While improving the 'cost' usually indicates an improvement in spacetime overhead, consideration of concepts such as balanced invested is important, as we have seen, and in the case of small angle rotations where the pivotal rotation must be pre-distilled to a high level of fidelity this is yet to be investigated.

The simple question 'which is better?' of MSD and gate synthesis can be a far more subtle one. Campbell and Howard have recently shown how both can be combined in a process of 'synthillation' [145, 146], where a class of circuits can be implemented with the same number of T gates as conventional synthesis but with quadratic error suppression which can lead to significant resource savings, combining both synthesis and magic state distillation (where $|A\rangle$ states are used to distill a multi-qubit higher level magic state) in one step.

Chapter 7

Conclusion

In the course of thesis I have attempted to cover aspects of quantum computation from the first demonstrations of successful quantum error correction all the way to the overhead of achieving universality.

In Chapter 2, a novel method of implementing the surface code in a silicon donor based architecture was introduced. It was demonstrated that the challenging exchange coupling required by many previous proposed architectures in this system could be replaced by the use of long range dipole-dipole interactions. Given local control of the 'probe' spins and a repeated mechanical motion we showed that the surface code could be implemented, which can form the substrate of universal quantum computation, and threshold simulations demonstrated levels of tolerance to qubit misplacement greater than that provided by exchange gates. There are a number of areas of further work that could and are being explored for further work by myself and others. In particular further development of the orbital probe quantum computer is continuing apace and a deeper investigation into the feasibility of the 'all electronic' version is required.

In Chapter 3 a method to benchmark the performance of small error correcting codes in near-future experiments based on 'integrity' was introduced. The relationship between 'integrity' and trace distance was described and a number of 'milestones' were introduced. The Alice-Igor-Bob model of an experiment was introduced as a means to provide a simple method for experimentalists to assess the performance of qubits with respect to these benchmarks. An obvious extension to the current work is the examine to Alice-Igor-Bob scenario and the integrity metric when trying to compute a series of Clifford gates on the logical qubit, including two-qubit gates performed transversally or, if the number of qubits is great enough, using a technique such lattice surgery for the Steane and surface codes. This, and the further work to develop models of systems that wish to demonstrate large-scale computation (as in Chapter 2), require

close collaboration between experimentalist and theoreticians to develop the most realistic possible models of errors within the individual system under investigation.

As we confirmed in Chapter 5 only constant factor improvements, rather than superior scaling, are likely to come about by further improving magic state distillation and using it with surface code computation. There remains a need to understand whether more radical codes like the 3D gauge colour codes are competitive with surface code + magic states. Of course, the qubit overhead is not the only question here. The ability of the 3D gauge colour code to perform single shot quantum error correction, as well as a full set of universal operations, might prove to be an important advantage given that a surface code computer's runtime will be limited by both T-depth and measurement time. In the same way that we have for Shor's algorithm, a similar analysis of the overhead of hybrid algorithms [179] which use many shallow circuit compilations might be interesting to evaluate. This could potentially have less intimidating overheads as these typically require 'shallow' low T-depth circuits, which are repeated by the quantum computer potentially as part of a feedback loop to a classical circuit. With a shallow circuit, logical error rates need not be so low and thus lower distance surface codes for example could be used, reducing the overall overhead. A circuit of T-depth 10^6 on a surface code quantum computer, for example, could be several times (but less than an order of magnitude) less resource intensive (cf.Fig. 5.7) than the examples considered in this thesis. Of course, a full surface code might not be the optimal ways to achieve such error rates, which could be possible with a single level of another low distance quantum code. If the idea of quantum error *mitigation* [180, 181] continues to emerge, increasing the error tolerance of shallow circuits, then it is reasonable to ask whether such ideas could combine with small codes to achieve speedups with smaller quantum computers.

Chapter 6 demonstrated that by compressing the circuit of the existing protocol of Duclos-Cianci and Poulin [174] a more efficient means of distilling magic states for small angles rotations could be demonstrated. Given the metric of the distillation 'cost', that is the number of noisy magic states required to output a distilled state, we found that this protocol was an improvement of a factor of 2 over the original protocol, but more interestingly was around 20 times less costly than leading gate synthesis techniques for levels $\ell = 4, 5, 6, 7$ of the Clifford hierarchy. Magic state dilution leads to even larger savings over gate synthesis for large ℓ (higher levels of the hierarchy). A full overhead analysis (such as the one in Chapter 5) for small angle magic states would be an important step in gaining a realistic comparison between

this approach and gate synthesis but was beyond the scope of the work presented here.

Throughout I have attempted to make contact with the current experimental state-of-the-art, hopefully demonstrating that prototype devices in certain systems are currently operating at a level that, if replicated in a large-scale device, would allow for universal fault-tolerant quantum computation with a 'reasonable' overhead. Of course, the question of whether a device whose components operate with one 1 error in 10,000 operations and comprises over 10 million components is feasible is an open one, and it certainly can seem an intimidating task when looking at the current and near future devices which have much fewer than 100 qubits. However, these numbers are not obviously beyond the realms of physical possibility: one does not estimate that a quantum computer will be the size of a planet or the solar system. We have seen that one proposal (if we are being ambitious) could fit onto a chip! We should remember that the world's current largest supercomputer Tianhe-2 has 1,375 TiB of memory, 12.4 PB of storage and runs at a speed of 33.86-petaflop. This machine has over 3 million cores, each of which contains hundreds of millions of transistors. We estimate that with our current best technology and techniques that 'just' 10 million quantum bits (or that order) could perform a calculation that *could never be done* on by this or any other classical computer. This doesn't even take into account the improvements made in error correction and fault-tolerance theory that are yet to be discovered, and there is still time for improvements to be made while experiments are scaled up.

Appendix A

Appendices for small angle rotations

In this Appendix a full description of some aspects of the improved methods of distilling small angle magic states, due entirely to Earl Campbell, are described. This is adapted directly from Ref. [176]. First the noise analysis of the compressed distillation circuit is described in greater detail. The direct comparison between the uncompressed and compressed protocols demonstrates that the compressed protocol is superior. Then a new method of mixing stabiliser states and very small angle magic states to obtain significant resource savings in very small angle rotations. Numerical simulations and the invention of 'magic state dilution' are credited to Earl Campbell.

A.1 Noise analysis

From the circuit 6.4 we can deduce the leading terms presented in Eqs. 6.14 for the success probability of the distillation protocol and the output error on the distilled magic states. First consider the failure probability, $P_{\text{fail}} = 1 - P_{\text{suc}}$. If $|x| := x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 = 1 \pmod 2$, then we detect an error by measuring the control qubit and have a failure. The leading order contribution is single error processes, and there are 8 such errors, so this contributes $8\epsilon_3$ to P_{fail}. Otherwise, if $x = (0, 0, \ldots 0)$ and the pivotal rotation also works, then we perform a perfect projection onto the $+H_\ell \otimes H_\ell$ subspace. This projection then detects an error if one of the two noisy $|M_\ell\rangle$ states is faulty, which occurs with probability $2\epsilon_\ell$. If everything works correctly except the pivotal rotation, which fails with probability $\eta_{\ell-1}$ it results in the operation shown

in Fig. (6.2c), but with the replacement $H_\ell \to iH_\ell Y$. Algebraically, this operation is

$$V = \frac{1}{2}\left[\mathbb{1} + (iH_\ell Y)\otimes(iH_\ell Y)\right]\left[(iH_\ell Y)\otimes\mathbb{1})\right] \tag{A.1}$$
$$= \frac{-i}{2}\left[(YH_\ell)\otimes\mathbb{1} + \mathbb{1}\otimes(YH_\ell)\right]$$

Applying this channel to the input magic states $|M_\ell\rangle|M_\ell\rangle$, and noting that $Y|M_\ell\rangle = \left|\bar{M}_\ell\right\rangle$, we have

$$|\psi\rangle = V|M_\ell\rangle|M_\ell\rangle = \frac{-i}{2}(\left|\bar{M}_\ell\right\rangle|M_\ell\rangle + |M_\ell\rangle\left|\bar{M}_\ell\right\rangle) \tag{A.2}$$

Therefore, the probability of an even parity measurement, combined with pivotal rotation failure, is $\eta_{\ell-1}\langle\psi|\psi\rangle$. Recalling, $\langle M_\ell|\bar{M}_\ell\rangle = 0$, we find $\langle\psi|\psi\rangle = 1/2$. Therefore, the failure probability gains a contribution of $\frac{1}{2}\eta_{\ell-1}$. Thus the total probability of the leading order processes that lead to a detected failure is $8\epsilon_3 - 2\epsilon_\ell - \frac{1}{2}\eta_{\ell-1}$.

Next we consider the leading errors that go undetected and output the wrong magic state $\left|\bar{M}_\ell\right\rangle$. The protocol outputs a two qubit state, and we trace out the second qubit and evaluate the error rate on the first qubit. Switching the first and second output qubit will yield the same result. First we consider pairs of errors in the R_3 gates. There are 28 such pairs, all satisfying $|x| = 0 \pmod 2$. However, due to the measurement of the ancilla if $x_3 + x_4 + x_5 + x_6 = 1 \pmod 2$ an error is detected. This cuts the number of undetected error pairs down to 14. Not all undetected error pairs lead to a logical error on the first qubit. For 2 of the undetected errors $x_1 = x_2 = 1$ and $x_7 = x_8 = 1$ we see that there is a logical $Y\otimes Y$ error, and so both these processes contribute to errors on the output. Now consider the remaining 12 undetected errors, the parity projection becomes deformed so that it projects onto a state that on average has 1/2 overlap with $|M_\ell\rangle$. This totals the error contribution from failed R_3 gates to $(2 + \frac{12}{2})\epsilon_3^2 = 8\epsilon_3^2$. If the R_3 gates do not fail, but instead we have a perfect parity projection, then a logical error occurs if the projection is applied to $\left|\bar{M}_\ell\right\rangle\left|\bar{M}_\ell\right\rangle$, which occurs with probability ϵ_ℓ^2. The full symbolic simulation of the circuit that determined this, and higher order terms, can be found in Appendix A. Finally if the pivotal rotation carries an error we see from Eqn. (A.2) that $|\langle\bar{M}_\ell, M_\ell|\psi\rangle| = 1/4$ and so the this leads to a logical error with probability $\frac{1}{4}\delta_{\ell-1}$. To reiterate: the leading order contribute here is linear, and not quadratically suppressed, and so the pivotal rotation must be high fidelity to enable distillation.

A.2 Results for small ℓ

The analysis summarised in the following sections is the result of numerical simulations performed by Earl Campbell, based on the circuits and noise analysis derived in

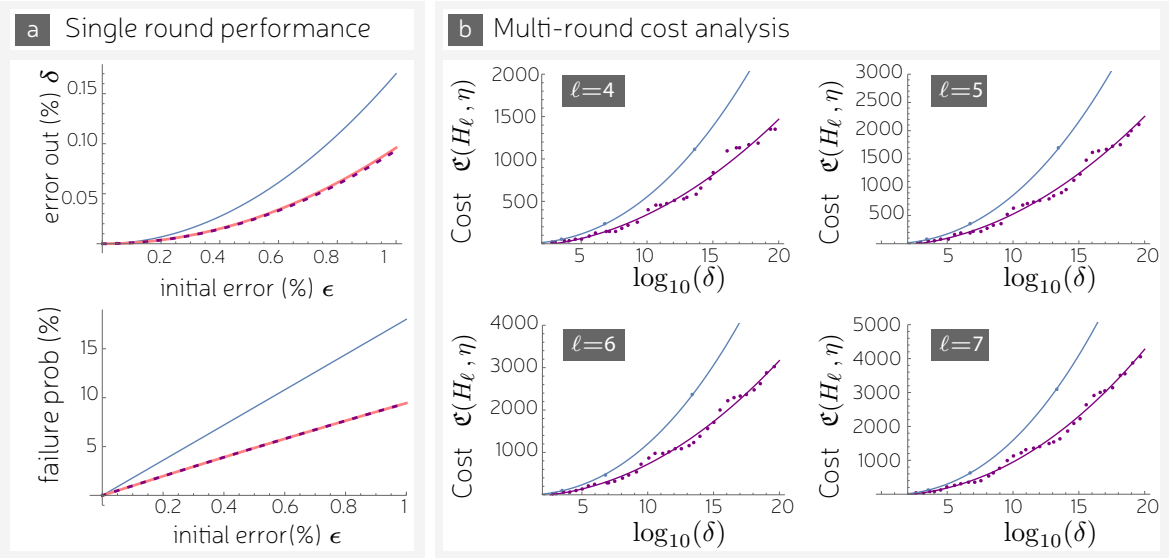


Figure A.1: Comparison of the resource cost of distillation using MEK_ℓ and DP_ℓ. (a) shows performance when $\epsilon_3 = \epsilon$, $\epsilon_\ell = \epsilon$ and $\delta = \epsilon^2$. These are benchmarked against the standard MEK_3 protocol (pink), where the curves for MEK_ℓ (purple and dotted) are barely distinguishable from MEK_3, whereas DP_ℓ (blue) performs much worse. The curves for DP_ℓ are based on leading order approximations [182]. (b) shows the full resource cost of MEK_ℓ protocol (purple) and DP_ℓ (blue) of distilling a $|M_\ell\rangle$ state of final error of δ using resources with an initial error of 1%. Data for DP_ℓ taken from Table 1 of Ref. [174]. Lines are fitted functions of the form $\mathfrak{C} = a\log(\delta)^b + c$. [Figure credit: Earl Campbell]

previous sections.

A.2.1 Comparison with DP protocol

There are three aspects to the comparison: the number of resources required per attempt, the success probability and how the protocols suppress errors. Recall each round of MEK_ℓ uses 8 fewer $|M_3\rangle$ states than a round of DP_ℓ. In Figs. (A.1a) and (A.1b) we fix $\epsilon_\ell = \epsilon_3 = \epsilon$ and $\eta_{\ell-1} = \epsilon^2$ and compare the performance of MEK_ℓ and DP_ℓ, both benchmarked against MEK_3. We remark that $\eta_{\ell-1}$ must be set significantly lower than other errors as the protocol can not detect noise in the pivotal rotation. In this context, MEK_ℓ is barely indistinguishable from MEK_3. This is expected as the protocols perform identically when $\eta_{\ell-1} = 0$, and since $\eta_{\ell-1} = \epsilon^2$ is very small we only observe a very slight difference between MEK_ℓ and MEK_3. In contrast, DP_ℓ performs worse and consumes more resources.

We also consider the full cost of performing many rounds of MEK_ℓ and DP_ℓ. Because the cost of the pivotal rotation increases with ℓ, we now see a variation in

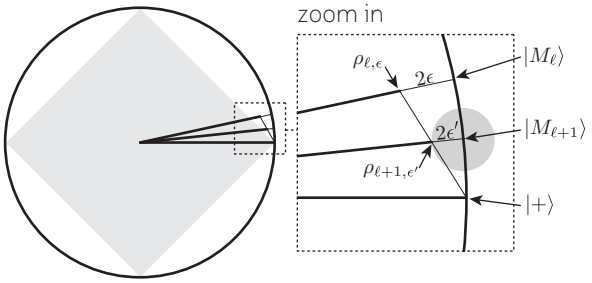


Figure A.2: A geometric representation of the dilution protocol in a cross-section of the Bloch sphere. Note that two points separated by a geometric distance of d in the Bloch sphere, correspond to operators with $d/2$ distance in trace norm. The grey diamond shows the set of stabiliser states. Given a $\rho_{\ell,\epsilon}$ state and a pure $|+\rangle$ state we can prepare mixtures on the line between these points. This line intersects the line for states of the form $\rho_{\ell,\epsilon'}$. We see that for sufficiently large ϵ, the resulting ϵ' can be reduced $\epsilon' \leq \epsilon$. Furthermore, we show a $2\epsilon'$ radius ball about the point $|M_{\ell+1}\rangle$ to highlight that both $\rho_{\ell,\epsilon}$ state and $|+\rangle$ are further than $2\epsilon'$ away, and so these states would provide a worse approximation of $|M_{\ell+1}\rangle$ than their mixture. [Figure credit: Earl Campbell]

performance with ℓ. The results are shown for $\ell = 4, 5, 6, 7$ in Fig. (A.1b) and show a factor 1/2 reduction in cost compared to those reported by Duclos-Cianci and Poulin.

Roughly, we observe , providing a clear cut case for using our compressed MEK$_\ell$ protocol rather than the original DP$_\ell$ proposal. Given that all component metrics (resources per round, failure probability and error out) are very favourable towards MEK$_\ell$, one may have expected a more dramatic reduction in cost over DP$_\ell$. One explanation is that both MEK$_\ell$ and DP$_\ell$ require 1 very high fidelity pivotal rotation, which is very costly, and this shared cost limits the extent to which MEK$_\ell$ can outperform DP$_\ell$.

A.3 Magic state dilution

The invention and investigation of magic state dilution described here is credited to Earl Campbell. This and the following section are a summary reproduced from Ref [175] and are included for the sake of completeness.

As we ascend the Clifford hierarchy, we expect the cost of our protocol to increase. However, for sufficiently small angles the associated magic states become very close to stabiliser states, and we should be able to exploit this to reduce resource costs. In the analysis of Duclos-Cianci and Poulin, they replaced noisy $|M_\ell\rangle$ states with a stabiliser state whenever ℓ exceeded 8. This enabled resource costs to eventually drop

with ℓ. The $|+\rangle$ state can be considered a noisy $|M_\ell\rangle$ state, though with coherent (non-diagonal) noise. We calculate the noise of this stabiliser state with respect to $|M_\ell\rangle$ and find

$$\begin{aligned}\frac{1}{2}||(|+\rangle\langle+| - |M_\ell\rangle\langle M_\ell|)||_{\text{tr}} &= |\sin(\theta_\ell)| \\ &\sim \theta_\ell = \frac{\pi}{2^\ell},\end{aligned} \tag{A.3}$$

where the last line gives the small angle approximation. Therefore, the resource becomes free whenever $\pi/(2^\ell)$ is smaller than the target error rate.

Magic state solution is an alternative solution that keeps us within the framework of diagonal noise and ensures rapid decrease of costs whenever $\theta_\ell^2/\sqrt{2}$ is smaller than the required error rate. If we choose to generate ρ with probability λ and ρ' with probability $1-\lambda$, then we have the random mixture $\lambda\rho + (1-\lambda)\rho'$. Furthermore, the expected cost is $\lambda\mathfrak{C} + (1-\lambda)\mathfrak{C}'$. We consider mixing a noisy $|M_\ell\rangle$ state with a $|+\rangle$ to obtain a good approximation of a noisy $|M_{\ell+1}\rangle$ state. We say the state $|M_\ell\rangle$ has been diluted, since this allows a source of $|M_\ell\rangle$ states to provide, on average, a greater number of noisy $|M_{\ell+1}\rangle$ states. We will see that while dilution may increase noise, there are practically relevant regimes where dilution reduces noise.

We use $\rho_{\ell,\epsilon}$ for a $|M_\ell\rangle$ state with ϵ diagonal noise, so that

$$\rho_{\ell,\epsilon} = (1-\epsilon)|M_\ell\rangle\langle M_\ell| + \epsilon|\bar{M}_\ell\rangle\langle\bar{M}_\ell|. \tag{A.4}$$

Preparing $\rho_{\ell,\epsilon}$ costs $\mathfrak{C}(M_\ell, \epsilon)$ resources. The principle result of magic state dilution is the relation

$$\rho_{\ell+1,\epsilon'} = \lambda\rho_{\ell,\epsilon} + (1-\lambda)|+\rangle\langle+|, \tag{A.5}$$

where

$$\lambda = \frac{1}{2(1-\epsilon)}, \tag{A.6}$$

$$\epsilon' = \frac{1}{2}\left(1 - (1-2\epsilon)\left[\frac{\cos(\theta_\ell)}{(1-\epsilon)}\right]\right). \tag{A.7}$$

The dilution provides the noisy $|M_{\ell+1}\rangle$ magic state $\rho_{\ell+1,\epsilon'}$ at a cost $\lambda\mathfrak{C}(M_\ell, \epsilon)$, which is half the cost of the $\rho_{\ell,\epsilon'}$ state when ϵ is small. Dilution can even decrease the error rate, provided ℓ is large enough that the fraction in square bracket exceeds 1, then we have $\epsilon' \leq \epsilon$. Iterating this process decreases costs exponentially with ℓ. The underlying geometric intuition behind Eq. (A.5) is presented in Fig. (A.2) and the proof can be found in the appendix of Ref. [175].

A.4 Results for higher ℓ

Here we discuss the performance of MEK_ℓ combined with dilution at performing small angle rotations, beyond $\ell = 7$. Our results are generated iteratively. After having compiled a list of achievable costs $\mathfrak{C}(M_\ell, \epsilon)$ and $\mathfrak{C}(R_\ell, \eta)$ for different values of ϵ, we next built lists for $\ell + 1$. The first step is to build a list of $\mathfrak{C}(M_{\ell+1}, \epsilon')$ derived from $\mathfrak{C}(M_\ell, \epsilon)$ using the dilution protocol in the previous section. Next, we considered different combinations of input states into the MEK_ℓ, allowing for using diluted states without any further distillation or inputting diluted states into MEK_ℓ. The results are presented in Fig. (6.6) as a function of ℓ, showing how resource costs scale with decreasing angle.

For the target error rates $10^{-10}, 10^{-15}, 10^{-20}$ we can see three clear regions, with the middle region absent for the 10^{-5} plot. First, the resource cost increases roughly linearly with ℓ. Next, there is a transition where the gradient becomes gentler. Lastly, at some cutoff the cost starts to fall exponentially with ℓ. In this last regime, we rely solely on diluting magic states. This exponential cliff was predicted by the analysis in the previous section, and is labeled in the plots by ℓ_c. The behaviour of the middle region is also due to dilution. Here we typically find that one round of MEK_ℓ is used, with the input noisy $|M_\ell\rangle$ states produced by dilution, as opposed to using two or more rounds of MEK_ℓ on a raw resource of error rate ϵ_{raw}. Since dilution is less effective at low ℓ, in this regime we are completely reliant on MEK_ℓ and here observe the most rapid increase in costs.

Our results are also presented against the cost of two gate-synthesis methods SR and PQF. We see that for small ℓ, and large $\ell > \ell_c$ there is a significant gain over both gate-synthesis methods by over an order of magnitude. In the intermediate regime, our protocol still outperforms gate-synthesis but approaches a similar order of magnitude.

A.5 Supplementary materials

A.5.1 Explicit calculation of δ_ℓ and P_{suc} for MEK_ℓ

This section is taken directly from the Supplementary Material of Ref. [175]. It is a Mathematica notebook that calculates the error probability $\delta_\ell(\epsilon_3, \epsilon_\ell, \eta_{\ell-1})$ and $P_{\text{suc}}(\epsilon_3, \epsilon_\ell, \eta_{\ell-1})$ whose leading order contributions were described in Eq. 6.14, the intuition behind which was detailed in Section A.1.

Finding the MEK_L Performance

Definitions

Definitions

```
ID = {{1, 0}, {0, 1}};
X = {{0, 1}, {1, 0}};
Z = {{1, 0}, {0, -1}};
Y = i * X.Z;
HL = Cos[θ] * X + Sin[θ] Z;
HLη[xη_] = HL.MatrixPower[i * Y, xη];
```

Defining some matrices

```
ρONE[ ϵ_] = (ID + (1 - 2 ϵ) * HL) / 2;
ρPerfectGlobal = KroneckerProduct[(ID + HL) / 2, (ID + HL) / 2];
ρPerfectLocal1 = KroneckerProduct[ID, (ID + HL) / 2];
ρPerfectLocal2 = KroneckerProduct[ID, (ID + HL) / 2];
ρ[ ϵ_] = KroneckerProduct[ρONE[ ϵ], ρONE[ ϵ]];
```

We think of MEK being a protocol that takes 2 magic states in the state $\rho[\epsilon]$ and pumps on them with a circuit using a further 8 copies.

Channels

Channels

```
U1[x1_, x2_, x3_, x4_, x5_, x6_, x7_, x8_] =
  MatrixPower[KroneckerProduct[X, X], x1 + x3].
   MatrixPower[KroneckerProduct[Z, Z], x2 + x4];
U2[x1_, x2_, x3_, x4_, x5_, x6_, x7_, x8_] =
  MatrixPower[KroneckerProduct[X, X], x5 + x7].
   MatrixPower[KroneckerProduct[Z, Z], x6 + x8];
Proj[x1_, x2_, x3_, x4_, x5_, x6_, x7_, x8_, xη_] =
  KroneckerProduct[ID, HLη[xη]].(KroneckerProduct[ID, ID] +
       (-1)^(x3+x4+x5+x6) KroneckerProduct[HLη[xη], HLη[xη]]) / 2;

Kraus[x1_, x2_, x3_, x4_, x5_, x6_, x7_, x8_, xη_] :=
  U2[x1, x2, x3, x4, x5, x6, x7, x8].
   Proj[x1, x2, x3, x4, x5, x6, x7, x8, xη].U1[x1, x2, x3, x4, x5, x6, x7, x8];
```

We will apply a noisy channel to $\rho[\epsilon]$ which is a sum over Kraus operators labelled by [x1_,x2_,x3_,x4_,x5_,x6_,x7_,x8_], where
xj=1 indicates an error on the j^th of te 8 pumping magic states.

Appendix B

Smoothing systematic error

To characterise the entire process of the stabilizer measurement we carry out a full analysis of the measurement procedure including all sources of noise noted in the section above, and generate a superoperator from the result to completely describe the action of the stabilizing measurement procedure.

$$\mathcal{S}(\rho) = \sum_{i=0} p_i K_i \rho K_i^\dagger \tag{B.1}$$

This probabilistic decomposition describes the operation as a series of Kraus operators, K_i, applied to the initial state with probabilities p_i, which depend on the chosen protocol, noise model and the error rates. The leading term $i = 0$ will have corresponding K_0 representing the reported parity projection, and large p_0. For the protocols considered here, the other Kraus operations can be decomposed and expressed as a parity projection with additional erroneous operations applied.

Consider a known deterministic set of phase errors over a 4-qubit stabilizer-by-probe. The probe and data qubits mutually acquire phase through their dipole-dipole interaction. This interaction between probe and single data qubit leads to the following gate

$$S(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \exp(i\theta) & 0 & 0 \\ 0 & 0 & \exp(i\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{B.2}$$

where $\theta = \pi/2 + \delta(x, y, z)$ is a function of the position of the data qubit.

This means that after the probe has passed over one of the four qubits the state of the system is

$$|system\rangle \propto \left(|0\rangle V^a S_1^a + |1\rangle (iV^a Z^a) S_1'^a \right) |data\rangle , \tag{B.3}$$

where $V = diag\{1, i\}$, $S_k = diag\{1, e^{i\delta_k}\}$, $S'_k = diag\{e^{i\delta_k}, 1\}$, $Z = diag\{1, -1\}$ and the superscripts $\{a, b, c, d\}$ label the data qubit on which the operator acts.

After the probe has passed four data qubits, each of which injects some erroneous phase δ_i onto the probe qubit, the state of the system is proportional to

$$\left(|0\rangle\, S_4^d S_3^c S_2^b S_1^a + |1\rangle\, Z^d Z^c Z^b Z^a S_4'^d S_3'^c S_2'^b S_1'^a\right) V^d V^c V^b V^a\, |data\rangle\,. \tag{B.4}$$

We want to measure the probe in the $|\pm\rangle$ basis, the result of which will determine our estimate of the parity of the data qubits. Rewriting Equation B.4,

$$|\pm\rangle\, \left(S_4^d S_3^c S_2^b S_1^a \pm Z^d Z^c Z^b Z^a S_4'^d S_3'^c S_2'^b S_1'^a\right) V^d V^c V^b V^a\, |data\rangle\,. \tag{B.5}$$

If measurement of the probe finds it in the $|+\rangle$ then we interpret this as an attempted even parity projection. We neglect for now the unconditional phases $V^d V^c V^b V^a$. The actual projection we have performed on the data qubits is

$$\begin{aligned}
\hat{P}'_{\text{even}} = \; & S_4^d S_3^c S_2^b S_1^a + Z^d Z^c Z^b Z^a S_4'^d S_3'^c S_2'^b S_1'^a \\
\propto \; & c_1 \left(|0000\rangle\langle 0000| + |1111\rangle\langle 1111|\right) + \\
& c_2 \left(|0011\rangle\langle 0011| + |1100\rangle\langle 1100|\right) + \\
& c_3 \left(|0101\rangle\langle 0101| + |1010\rangle\langle 1010|\right) + \\
& c_4 \left(|0110\rangle\langle 0110| + |1001\rangle\langle 1001|\right) + \\
& is_1 \left(|1110\rangle\langle 1110| - |0001\rangle\langle 0001|\right) + \\
& is_2 \left(|1101\rangle\langle 1101| - |0010\rangle\langle 0010|\right) + \\
& is_3 \left(|1011\rangle\langle 1011| - |0100\rangle\langle 0100|\right) + \\
& is_4 \left(|1000\rangle\langle 1000| - |0111\rangle\langle 0111|\right).
\end{aligned} \tag{B.6}$$

This is clearly not a true parity projection, as different even parity subspaces $P^{(i)}_{\text{even}}$ have different weightings c_i, e.g. $c_1 = \cos\left(\frac{\delta_1+\delta_2+\delta_3+\delta_4}{2}\right)$ for $P^{(1)}_{\text{even}} = |0000\rangle + |1111\rangle$ and there is some weight on projection onto odd parity subspaces $P^{(i)}_{\text{odd}}$, e.g. $s_1 = \sin\left(\frac{\delta_1+\delta_2+\delta_3-\delta_4}{2}\right)$ for $P^{(1)}_{\text{odd}} = |1110\rangle + |0001\rangle$.

Consider the following protocol to smooth out this systematic error in our parity measurement: we randomly select one of four patterns of X operators on the data qubits and apply it before and after the $\hat{P}'_{\text{even}}$ projector. We choose from the set $U_1 = \mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1}, U_2 = \mathbb{1}\mathbb{1}XX, U_3 = \mathbb{1}X\mathbb{1}X, U_4 = \mathbb{1}XX\mathbb{1}$ to smooth the weightings c_i and s_i of Equation B.6.

The action of this protocol on the state ρ of the data qubits is thus,

$$\begin{aligned} P_{\text{even}}^{\text{smooth}}(\rho) = \frac{1}{4}\big[&(U_1\hat{P}'_{\text{even}}U_1)\rho(U_1\hat{P}'^{\dagger}_{\text{even}}U_1)+ \\ &(U_2\hat{P}'_{\text{even}}U_2)\rho(U_2\hat{P}'^{\dagger}_{\text{even}}U_2)+ \\ &(U_3\hat{P}'_{\text{even}}U_3)\rho(U_3\hat{P}'^{\dagger}_{\text{even}}U_3)+ \\ &(U_4\hat{P}'_{\text{even}}U_4)\rho(U_4\hat{P}'^{\dagger}_{\text{even}}U_4)\big] \end{aligned} \tag{B.7}$$

The operations U_i have the effect of permuting the weightings of projecting into the different subspaces in $\hat{P}'_{\text{even}}$. For example $U_2\hat{P}'_{\text{even}}U_2$ has the same form as $\hat{P}'_{\text{even}}$ with the weightings redistributed according to the relabelling: $1 \leftrightarrow 2$, $3 \leftrightarrow 4$. Expanding out Equation B.7 we find 16 'even' terms $P^{(i)}_{\text{even}}\rho P^{(j)}_{\text{even}}$, 16 'odd' terms $P^{(i)}_{\text{odd}}\rho P^{(j)}_{\text{odd}}$ and 32 'cross' terms $P^{(i)}_{\text{even}}\rho P^{(j)}_{\text{odd}}$. We add another level to our protocol, applying $(\mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1})$ or $(ZZZZ)$ with probability 1/2 to kill off the cross terms.

We then find that it is possible to re-express $P_{\text{even}}^{\text{smooth}}(\rho)$ as the probabilistic sum of perfect odd and even parity projections, followed by Z errors on either one or two data qubits,

$$\begin{aligned} P_{\text{even}}^{\text{smooth}}(\rho) = \omega_{\text{even}}&P_{\text{even}}\rho P_{\text{even}}+ \\ &\Gamma_{\mathbb{1}Z\mathbb{1}Z}P_{\text{even}}^{\mathbb{1}Z\mathbb{1}Z}\rho P_{\text{even}}^{\mathbb{1}Z\mathbb{1}Z}+ \\ &\Gamma_{\mathbb{1}Z\mathbb{1}Z}P_{\text{even}}^{\mathbb{1}Z\mathbb{1}Z}\rho P_{\text{even}}^{\mathbb{1}Z\mathbb{1}Z}+ \\ &\Gamma_{\mathbb{1}\mathbb{1}ZZ}P_{\text{even}}^{\mathbb{1}\mathbb{1}ZZ}\rho P_{\text{even}}^{\mathbb{1}\mathbb{1}ZZ}+ \\ &\Delta_{Z\mathbb{1}\mathbb{1}\mathbb{1}}P_{\text{odd}}^{Z\mathbb{1}\mathbb{1}\mathbb{1}}\rho P_{\text{odd}}^{Z\mathbb{1}\mathbb{1}\mathbb{1}}+ \\ &\Delta_{\mathbb{1}Z\mathbb{1}\mathbb{1}}P_{\text{odd}}^{\mathbb{1}Z\mathbb{1}\mathbb{1}}\rho P_{\text{odd}}^{\mathbb{1}Z\mathbb{1}\mathbb{1}}+ \\ &\Delta_{\mathbb{1}\mathbb{1}Z\mathbb{1}}P_{\text{odd}}^{\mathbb{1}\mathbb{1}Z\mathbb{1}}\rho P_{\text{odd}}^{\mathbb{1}\mathbb{1}Z\mathbb{1}}+ \\ &\Delta_{\mathbb{1}\mathbb{1}\mathbb{1}Z}P_{\text{odd}}^{\mathbb{1}\mathbb{1}\mathbb{1}Z}\rho P_{\text{odd}}^{\mathbb{1}\mathbb{1}\mathbb{1}Z}, \end{aligned} \tag{B.8}$$

where the Kraus operators

$$P_{\text{even/odd}}^{U_aU_bU_cU_d} = (U_aU_bU_cU_d)P_{\text{even/odd}}, \tag{B.9}$$

are each applied with a certain probability. Writing Equation B.8 in terms of $P^{(i)}_{\text{even}}\rho P^{(j)}_{\text{even}}$ and $P^{(i)}_{\text{odd}}\rho P^{(j)}_{\text{odd}}$ and equating with Equation B.7, we see that the probabilities can be

expressed as in terms of the weightings c_i, s_i as follows,

$$\begin{pmatrix} \omega_{\text{even}} \\ \Gamma_{\mathbb{1}Z\mathbb{1}Z} \\ \Gamma_{\mathbb{1}Z\mathbb{1}Z} \\ \Gamma_{\mathbb{1}\mathbb{1}ZZ} \end{pmatrix} = \frac{1}{4} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \end{pmatrix} \begin{pmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \end{pmatrix}$$
$$\begin{pmatrix} \Delta_{Z\mathbb{1}\mathbb{1}\mathbb{1}} \\ \Delta_{\mathbb{1}Z\mathbb{1}\mathbb{1}} \\ \Delta_{\mathbb{1}\mathbb{1}Z\mathbb{1}} \\ \Delta_{\mathbb{1}\mathbb{1}\mathbb{1}Z} \end{pmatrix} = \frac{1}{4} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \begin{pmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \end{pmatrix} \quad , \tag{B.10}$$

where $C_1 = \frac{1}{4}(c_1^2 + c_2^2 + c_3^2 + c_4^2)$, $C_2 = \frac{1}{2}(c_1c_2 + c_3c_4)$, $C_3 = \frac{1}{2}(c_1c_3 + c_3c_4)$, $C_4 = \frac{1}{2}(c_1c_4 + c_2c_3)$, $S_1 = \frac{1}{4}(s_1^2 + s_2^2 + s_3^2 + s_4^2)$, $S_2 = \frac{1}{2}(s_1s_2 + s_3s_4)$, $S_3 = \frac{1}{2}(s_1s_3 + s_3s_4)$ and $S_4 = \frac{1}{2}(s_1s_4 + s_2s_3)$.

Defining $\mathcal{C}_i = \cos(\frac{\delta_i}{2})$ and $\mathcal{S}_i = \sin(\frac{\delta_i}{2})$, the explicit forms of the resulting probabilities expressed as functions of the phase errors δ_i are

$$\begin{aligned}
\omega_{\text{even}} &= [\mathcal{C}_1\mathcal{C}_2\mathcal{C}_3\mathcal{C}_4 + \mathcal{S}_1\mathcal{S}_2\mathcal{S}_3\mathcal{S}_4]^2 \\
\Gamma_{\mathbb{1}Z\mathbb{1}Z} &= [\mathcal{C}_1\mathcal{S}_2\mathcal{S}_3\mathcal{C}_4 + \mathcal{S}_1\mathcal{C}_2\mathcal{C}_3\mathcal{S}_4]^2 \\
\Gamma_{\mathbb{1}Z\mathbb{1}Z} &= [\mathcal{C}_1\mathcal{S}_2\mathcal{C}_3\mathcal{S}_4 + \mathcal{S}_1\mathcal{C}_2\mathcal{S}_3\mathcal{C}_4]^2 \\
\Gamma_{\mathbb{1}\mathbb{1}ZZ} &= [\mathcal{C}_1\mathcal{C}_2\mathcal{S}_3\mathcal{S}_4 + \mathcal{S}_1\mathcal{S}_2\mathcal{C}_3\mathcal{C}_4]^2 \\
\Delta_{Z\mathbb{1}\mathbb{1}\mathbb{1}} &= [\mathcal{S}_1\mathcal{C}_2\mathcal{C}_3\mathcal{C}_4 + \mathcal{C}_1\mathcal{S}_2\mathcal{S}_3\mathcal{S}_4]^2 \\
\Delta_{\mathbb{1}Z\mathbb{1}\mathbb{1}} &= [\mathcal{C}_1\mathcal{S}_2\mathcal{C}_3\mathcal{C}_4 + \mathcal{S}_1\mathcal{C}_2\mathcal{S}_3\mathcal{S}_4]^2 \\
\Delta_{\mathbb{1}\mathbb{1}Z\mathbb{1}} &= [\mathcal{C}_1\mathcal{C}_2\mathcal{S}_3\mathcal{C}_4 + \mathcal{S}_1\mathcal{S}_2\mathcal{C}_3\mathcal{S}_4]^2 \\
\Delta_{\mathbb{1}\mathbb{1}\mathbb{1}Z} &= [\mathcal{C}_1\mathcal{C}_2\mathcal{C}_3\mathcal{S}_4 + \mathcal{S}_1\mathcal{S}_2\mathcal{S}_3\mathcal{C}_4]^2
\end{aligned}$$

We have thus shown that random application of one of a set of four unitaries before and after an 'imperfect' parity projection $\hat{P}'_{\text{even}}$ can be expressed as a superoperator on the data qubits. This has the form of the probabilistic application of 'perfect' parity projectors followed by Pauli-Z errors on subsets of the data qubits. When the phase errors δ_i are small the most probable operation is the desired perfect even parity projection P_{even} with no errors. This information on stabilizer performance then enables classical simulation of a full planar code array, and its fault tolerance threshold can be assessed.

The above considers the superoperator for a noisy parity projection in our probabilistic protocol predicated on obtaining the 'even' result when measuring the probe.

A similar result can be derived in the case that the probe is measured and found in the $|-\rangle$ state and for the three-qubit stabilizers which define the boundaries of the planar code. The superoperators for these edge stabilizers are also expressed as perfect P_{even} and P_{odd} projections followed with some probability by one- and two-qubit Z errors.

Appendix C

Appendices for small codes chapter

C.1 Simulating the small codes

We make use of the following stabiliser generators and low weight logical operators for our simulations of the small codes in Chapter 3. This appendix also presents the circuits used for the encoding and decoding of each code, with fault-tolerant options presented where appropriate.

The majority of this appendix appeared in Ref. [98].

Five qubit code

The stabilisers and logical operators used for the simulation of the five qubit code.

$$S_1 = \mathbb{1}XZZX$$

$$S_2 = X\mathbb{1}XZZ$$

$$S_3 = ZX\mathbb{1}XZ$$

$$S_4 = ZZX\mathbb{1}X.$$

$$X_L = Z\mathbb{1}\mathbb{1}ZX$$

$$Z_L = ZY\mathbb{1}\mathbb{1}Y$$

Seven qubit Steane code

The Steane code can be represented on a three-colourable graph (note that it is the smallest possible instance of a 2D colour code). The logical operators are chain of X or Z operators that span the code e.g. from corner to corner.

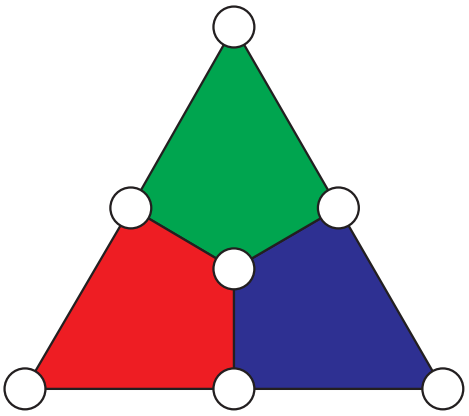


Figure C.1:
Seven qubit Steane code.
$S_1 = XXXX\mathbb{1}\mathbb{1}\mathbb{1}$,
$S_2 = \mathbb{1}X\mathbb{1}XXX\mathbb{1}$,
$S_3 = \mathbb{1}\mathbb{1}XX\mathbb{1}XX$,
$S_4 = ZZZZ\mathbb{1}\mathbb{1}\mathbb{1}$,
$S_5 = \mathbb{1}Z\mathbb{1}ZZZ\mathbb{1}$,
$S_6 = \mathbb{1}\mathbb{1}ZZ\mathbb{1}ZZ$,
$X_L = X\mathbb{1}X\mathbb{1}\mathbb{1}\mathbb{1}X$,
$Z_L = Z\mathbb{1}Z\mathbb{1}\mathbb{1}\mathbb{1}Z$.

A transversal measurement of every qubit in the X (Z) basis allows fault-tolerant measurement of the logical state in 'one shot'. From the 8 bits of information from measurements results the X (Z) logical state and the X (Z) stabiliser information can be inferred, providing protection against measurement error.

Nine qubit (rotated) surface code

$$S_1 = XX\mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1}$$

$$S_2 = \mathbb{1}XX\mathbb{1}XX\mathbb{1}\mathbb{1}\mathbb{1}$$

$$S_3 = \mathbb{1}\mathbb{1}\mathbb{1}XX\mathbb{1}XX\mathbb{1}$$

$$S_4 = \mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1}XX$$

$$S_5 = \mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1}ZZ\mathbb{1}ZZ$$

$$S_6 = \mathbb{1}\mathbb{1}\mathbb{1}Z\mathbb{1}\mathbb{1}Z\mathbb{1}\mathbb{1}$$

$$S_7 = ZZ\mathbb{1}ZZ\mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1}$$

$$S_8 = \mathbb{1}\mathbb{1}Z\mathbb{1}\mathbb{1}Z\mathbb{1}\mathbb{1}\mathbb{1}$$

$$X_L = \mathbb{1}X\mathbb{1}\mathbb{1}X\mathbb{1}\mathbb{1}X\mathbb{1}$$

$$Z_L = \mathbb{1}\mathbb{1}\mathbb{1}ZZZ\mathbb{1}\mathbb{1}\mathbb{1}$$

The planar surface code can be modified from the variant we met in Chapter 1, by rotating the lattice by 45 degrees. Then as shown in the example in Figure C.2 we have a much more efficient version. Now boundaries involve weight-2 operators and all the measured stabilisers are independent. We can encode one logical qubit with nine data qubits, as opposed to the 13 qubits required in the canonical variant. This 9 qubit code is the smallest example of a surface code. Like the seven qubit Steane code, we can measure a logical X or Z state transversally and have fault-tolerance to one measurement error.

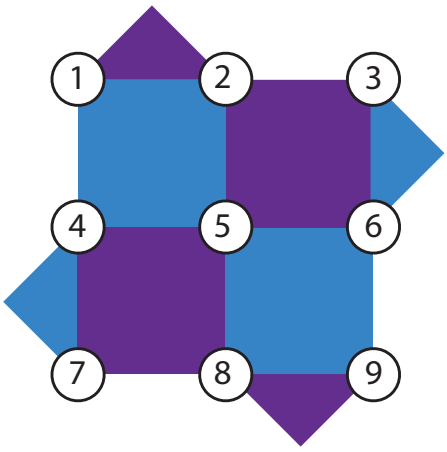


Figure C.2: The smallest surface code. On a 45 rotated lattice, 9 data qubits and 8 stabilisers (4 weight-4 and 4 weight-2) encode one logical qubit.

C.1.1 Circuits diagrams

Alice and Bob's circuits

In Figure C.3 we show the encoding circuits which we employ when Alice (taken to be ideal) encodes the physical qubit $|\psi\rangle$ which she has chosen to place into the memory. The encoding circuits come from Refs. [183], [184] and [165], respectively. Because Alice is perfect, there is no need for fault tolerance in these encoders. Bob employs the inverse of these encoders as a step in his analysis, see Table 3.1.

In order to experimentally investigate the integrity of a memory channel, we must use circuits for Alice and Bob that are as compact as possible and, as a strong preference, fault tolerant. Fortunately we need not consider general encode/decode circuits since (for a broad class of noise models) we know that the worst-case choice of state for Alice to transmit will be a Pauli basis state. Thus it is such states that we need to Alice to prepare and Bob to differentiate. A suitable compact, fault tolerant encoding circuit for the Steane code is shown in Fig. C.4(b) which is adopted from Ref. [185]. An equally compact, but non fault tolerant encoding circuit for the five-qubit code is shown in Fig. C.4(b). For both the seven-qubit and the five-qubit cases, our Bob now simply measures all the qubits; however importantly for the seven-qubit case he can perform classical error correction on the measurement results making his inference process fault tolerant.

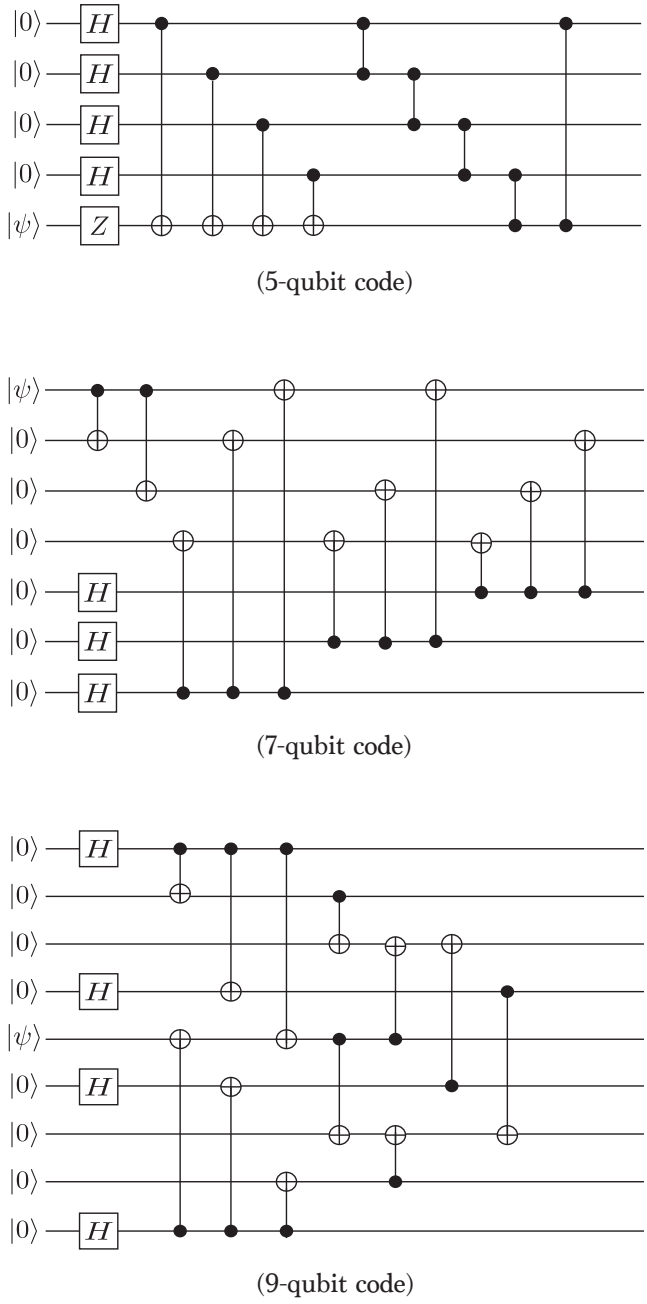


Figure C.3: **General encoding circuits suitable for the five-qubit, Steane, and nine-qubit codes.** In all cases the physical state of ψ is encoded into the logical state $|\psi\rangle_L$

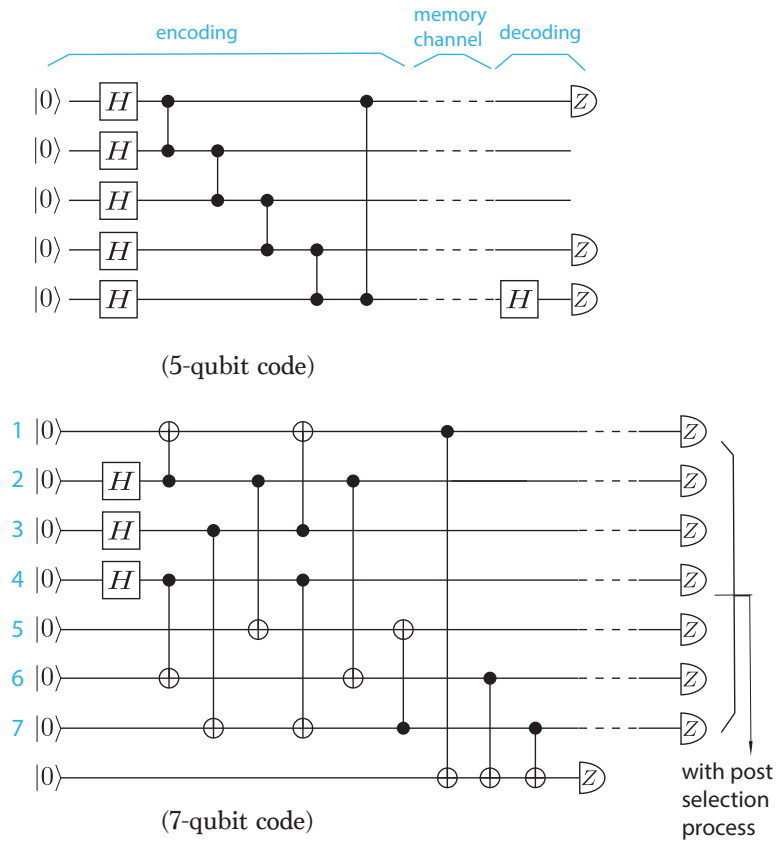


Figure C.4: **Encoding circuits for imperfect encoding procedures with the five- and seven-qubit codes.** For the five-qubit code shown in the upper panel, the encoded state $|-\rangle_L$ is prepared in a non-fault-tolerant fashion, and Bob subsequently identifies the received state by measuring three of the received qubits and computing their parity (again, a non-fault-tolerant process). For the seven-qubit code shown in the lower panel, physical qubits are encoded into $|0\rangle_L$ using additional qubit for detection of errors: if returns 1, Alice restarts the encoding until it returns 0. Such method reduces propagation of some errors in a noisy encoding process. Bob is also fault tolerant: he measures all 7 qubits, and may opt to flip one of the outcomes if it is necessary to do so in order to produce a legitimate outcome; the parity of subsets 4,5,6,7; 1,3,5,7; 2,3,6,7 should all be the same as to allow him to guess between $|0\rangle_L$ and $|1\rangle_L$.

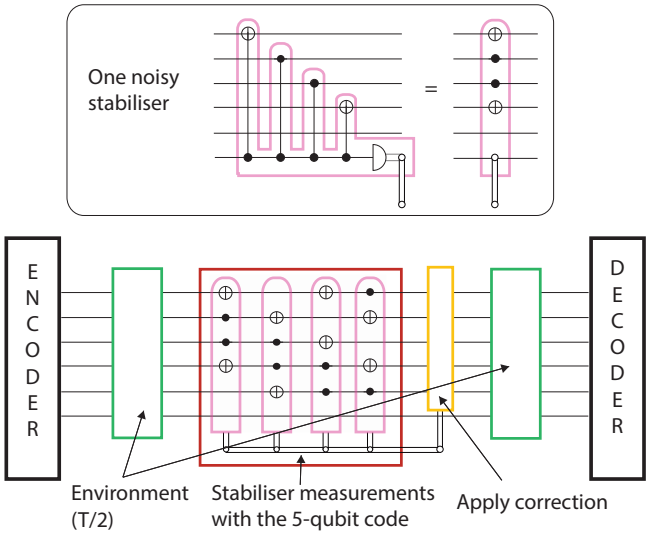


Figure C.5: **Diagram of one whole cycle of Alice-Igor-Bob scenario with the five-qubit code.** Firstly five physical qubits are encoded into the logical state, then the logical qubit is subjected to environmental noise for a time period of $T/2$, followed by a cycle of stabilizer measurements and error correction, and then the logical qubit is again subjected to environmental noise for $T/2$. Lastly the logical qubit is decoded and measured.

Igor's circuits

Figure C.5 shows the entire Alice-Igor-Bob process. In this figure, the memory channel employs the five-qubit code and Igor's error correction is not fault tolerant. Consequently the overall circuit is one of the more simple examples; but cases where we employ the Steane code or the nine-qubit code are analogous, as are cases where we opt to make Igor's process fault tolerant. The specific sub-circuits for these cases are shown in Figure C.6.

In our simulations (e.g. Fig. 3.6) we considered more than one type of fault tolerance. The most common method to avoid weight-2 errors is to encode four ancilla qubits into a cat state, verified with additional qubit, and apply transversal CNOT gates within each stabiliser check, which may be known as Shor's method. Circuits in Figure C.6 (a) and (b) demonstrate this approach. A slight difference between these two diagrams exists, regarding measurement of the ancilla: for the five-qubit code, the encoded ancilla qubit needs to be decoded by applying the gates used in encoding in reverse before measuring the decoded physical qubit, while for the Steane code, since each stabiliser check detects only one type of error, we can simply measure all the four physical qubits in the corresponding basis and check the

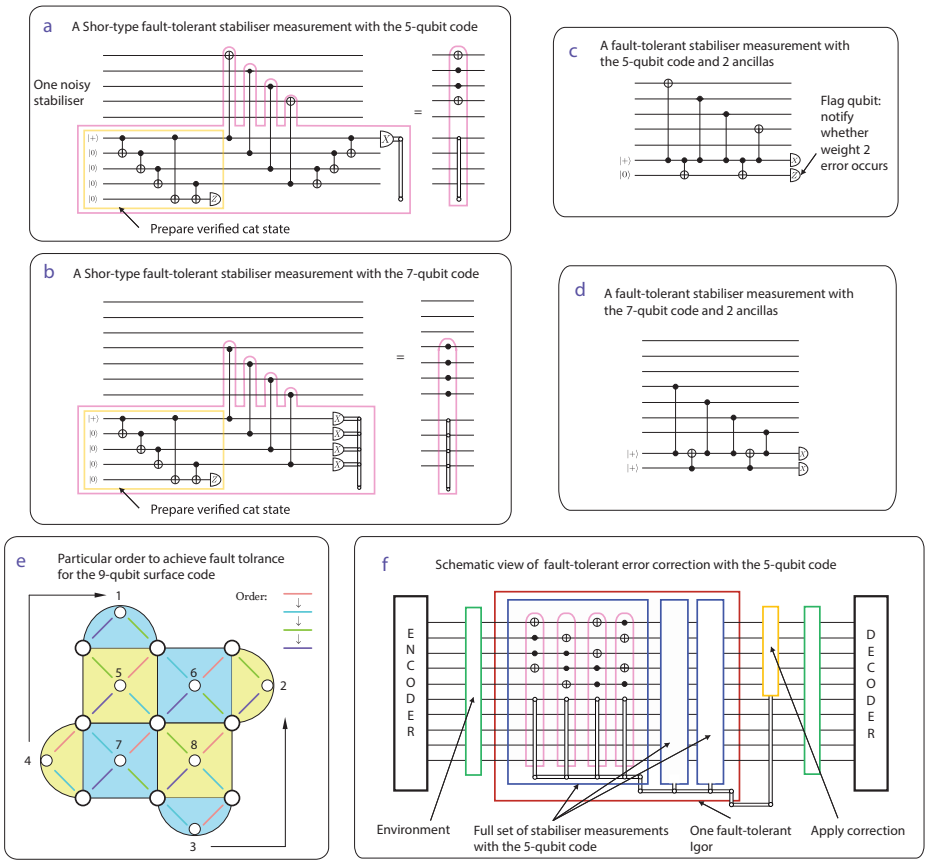


Figure C.6: **Circuits and diagrams for fault tolerant error correction.** (a) a round of fault tolerant stabilizer measurement with the five-qubit code. Here we use the Shor's method. After the stabilizer measurement, the ancillas are decoded, followed by measurement in x-basis. (b) a round of fault tolerant stabilizer checks with the seven-qubit Steane code, again using Shor's method. Since each stabilizer check detects either phase or bit flips, results can be obtained by checking the parity of measurement results of all four ancillas without decoding. (c) the circuit to achieve fault tolerant correction of the five-qubit code with only two ancillas. The first ancilla is used for stabilizer measurement, while the other one acts as the flag qubit: it returns -1 once any weight 2 errors occurs, and all such errors render a unique error syndrome thus can be corrected. (d) the same approach as in (c), but for for the seven-qubit Steane code. (e) stabilizer measurements of ancillas following a particular order to achieve fault tolerance with the rotated nine-qubit surface code. The large circles stand for the data qubits and the small circles are ancillas. The stabilizer measurement should follow the order denoted by the colour orange, blue, green and finally purple. Since the ancilla labelled 3 can physically act as the ancilla labelled 2 after finishing the measurement in the blue half-circle and that also works for ancilla 4, which can act as ancilla 1 after the measurement in the yellow half-circle, in total six ancillas are required to demonstrate fault tolerance. (f) schematic view of the whole cycle of Shor-type fault-tolerant Alice-Igor-Bob scenario with the five-qubit code ((a) in this figure). The same procedure also works for all the others described above. Three rounds of a full set of stabilizer measurements are required to obtain fault-tolerance to measurement error.

parity of the measurement results.

The alternative fault-tolerant circuits with only two ancilla qubits are shown in Figure C.6 (c) and (d) for the five-qubit and seven-qubit codes, respectively. Here we are employing the ideas recently introduced in Ref. [36]. The first ancilla qubit acts the same as that in the non-fault-tolerant circuit, and the second ancilla qubit acts as the flag qubit: once any weight-2 error occurs, the measurement of it will turn from 0 to 1. For both the five-qubit and seven-qubit codes, each weight-2 error corresponds to a unique error syndrome if applying a set of normal stabiliser checks, thus we can detect any weight-2 error by measurement of the flag qubit and correct by mapping the stabiliser measurement results with the unique error syndrome.

The nine-qubit code has the unusual and desirable property that the techniques described above, involving multiple ancillas, are not needed for fault tolerance. As shown in Fig. C.6(e), weight-2 errors can be avoided simply by taking care to measure the stabilisers in a certain order (as has been discussed in Ref. [117] and Ref. [33]). Since only one round of stabiliser checks is to be preformed, fewer gates compensate the cost of six ancilla qubits required.

The full diagram for evaluating the memory with fault-tolerant error correction is shown in Figure C.6 (f), where we take the Shor-type five-qubit code (Figure C.6 (a)) as an example – analogous circuits apply for the other cases. Compared with the non-fault-tolerant error correction as shown in Figure C.5, three rounds of stabiliser measurements are required in order to avoid additional errors introduced by error correction based on wrong error syndromes.

C.2 Significance of imposing a minimum

The observation reported here is in large part due to Niel de Beaudrap, and is included here for completeness.

In all simulations previously described in this report the environmental decoherence was purely depolarising. Consequently the environment has no preferred basis, and one finds that Bob's probability of successfully guessing the nature of the state selected by Alice does not vary according to her choice. Thus the minimum appearing in the definition of integrity, Eqn. 3.5, is redundant in the sense that the minimum and maximum are the same. In order to show that this will not generally be true, and that therefore it is indeed necessary to specify the minimum, we need only switch from a pure depolarising environment to a pure dephasing environment.

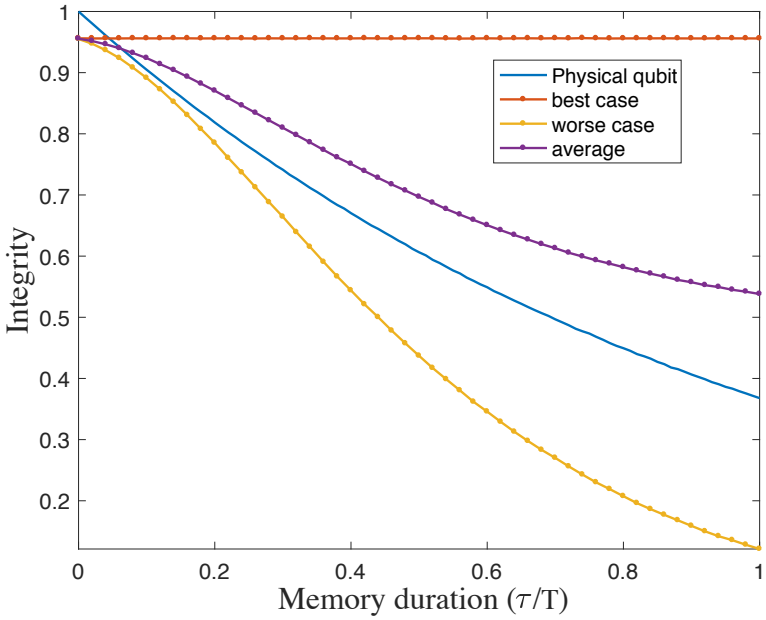


Figure C.7: **Integrity in a pure dephasing environment.** Plot shows the integrity for a memory channel using a the seven-qubit Steane code, and and single round of Igor's error correction procedure with a gate error rate of 0.5%. Whereas all other plots in this paper correspond to a pure depolarising environment, here we have a pure dephasing environment. Consequently Bob's ability to guess the nature of the received state depends strongly on Alice's choice of which qubit to send: If she sends a z-basis eigenstate then Bob's success is certain. [Figure credit: Xiao-si Xu].

The results of such simulations are shown in Fig. C.7 which shows a Steane-code protected memory but now with all environmental noise being pure dephasing. The interesting point is that now Bob's ability to guess the original encoded state varies dramatically with Alice's choice of initial state. If she chooses either $|0\rangle$ or $|1\rangle$ then the logical qubits are in fact immune to phase noise, so that Bob's performance impaired only by the noise introduced by Igor – the corresponding line (red) is therefore flat i.e. not a function of the memory duration. In contrast Bob's 'worst case' performance is obtained when Alice's choice for the encoded state is $|+\rangle$ or $|-\rangle$ as shown by the yellow line, and it is this that would define the integrity of memory channel.

In the following section we explain for many common environmental noise models the 'worst case' will be found among the Pauli eigenstates.

C.3 When does it suffice to prepare Pauli eigenstates?

The observation reported here is in large part due to Niel de Beaudrap, and is included here for completeness.

In the main text and in the preceding appendix we noted that Alice's choice of state to encode can influence Bob's performance when he guesses the nature of the received state. Therefore integrity is defined from the worst case performance. In the main text we noted that when indeed this occurs, we will often find that the worst case corresponds to Alice choosing a Paul eigenstate. Here we explain that this is typical for a broad range of error models. In the following, when we refer to 'weak' noise this is in the sense that the error probability is ≤ 0.5, which is in general the region of interest where the milestones M1-M4 can be met.

Recall that we are assessing single-qubit memories, represented by a channel of the form $\Phi : \mathrm{Herm}(\mathcal{H}_1) \to \mathrm{Herm}(\mathcal{H}_1)$, in the presence of realistic noise using experimentally viable methods.

We write $\mathbf{P} = \{\sigma^{(0)}, \sigma^{(1)}, \sigma^{(2)}, \sigma^{(3)}\}$ for the set of single-qubit Pauli operators (we sometimes write $\mathbb{1} = \sigma^{(0)}$, $X = \sigma^{(1)}$, $Y = \sigma^{(2)}$, and $Z = \sigma^{(3)}$). Let $\mathbf{P}^M$ denote the set of M-fold tensor products of Pauli operators $\sigma^{(a_1)} \otimes \cdots \otimes \sigma^{(a_M)}$. A *Pauli channel* is a channel on $M \geq 1$ qubits whose Kraus operators are each proportional to an element of $\mathbf{P}^M$, representing random Pauli operators acting on those qubits according to some distribution. We may denote such a channel by

$$\Phi(\rho) = \sum_{\tau \in \mathbf{P}^M} p_\tau \, \tau \rho \tau^\dagger. \tag{C.1}$$

(All such channels are unital, i.e. $\Phi(\frac{1}{d}\mathbb{1}) = \frac{1}{d}\mathbb{1}$.) A *weak Pauli channel* is such a channel in which $p_{\mathbb{1}\otimes\cdots\otimes\mathbb{1}} \geq \frac{1}{2}$; and a *weak i.i.d. Pauli channel* is such a channel which consists of a tensor product $\Phi_1 \otimes \Phi_1 \otimes \cdots \otimes \Phi_1$ of identical channels.

We can experimentally estimate the integrity of a weak Pauli channel $\Phi : \mathrm{Herm}(\mathcal{H}_1) \to \mathrm{Herm}(\mathcal{H}_1)$ on a single qubit, as follows. First note that we may simplify the formula of memory integrity from Eqn. (3.5) by noting that

$$\begin{aligned}\mathcal{R}(\Phi) &= \min_{\psi_0 \perp \psi_1} \tfrac{1}{2}\big\|\Phi(\psi_1) - \Phi(\psi_0)\big\|_{\mathrm{tr}} \\ &= \min_{\psi} \big\|\Phi(\psi) - \Phi(\tfrac{1}{2}\mathbb{1})\big\|_{\mathrm{tr}} \,. \end{aligned} \tag{C.2}$$

For such channels we have $\Phi(\sigma^{(j)}) = \alpha_j \sigma^{(j)}$ for some $\alpha_j \geq 0$: in particular, as these channels are unital, $\alpha_0 = 1$. Then for any single-qubit state $\rho = \frac{1}{2}\big[\mathbb{1} + r_1\sigma^{(1)} + r_2\sigma^{(2)} + r_3\sigma^{(3)}\big]$, we have

$$\Big\|\Phi(\rho) - \Phi(\tfrac{1}{2}\mathbb{1})\Big\|_{\mathrm{tr}} = \left\|\sum_{j=1}^{3} \tfrac{1}{2} r_j \alpha_j \sigma^{(j)}\right\|_{\mathrm{tr}} = \sum_{j=1}^{3} r_j^2 \alpha_j^2. \tag{C.3}$$

This is a convex combination of the scalars α_j^2, which is minimised by setting $r_j^2 = 1$ for the smallest coefficient α_j and $r_j = 0$ otherwise. Thus

$$\mathcal{R}(\Phi) = \min_j \alpha_j^2 = \min_j \mathcal{D}\Big(\Phi(\varphi_+^{(j)}), \Phi(\varphi_-^{(j)})\Big), \tag{C.4}$$

where $\varphi_\pm^{(j)}$ are the ± 1-eigenstates of the respective Pauli operator $\sigma^{(j)}$.

This motivates the following procedure to experimentally assess the quality of an isolated quantum memory on a weak Pauli channel: prepare a state $\varphi_\pm^{(j)} = |\phi\rangle\langle\phi|$, apply Φ to it, and test the probability with which we obtain the outcome $|\phi\rangle\langle\phi|$ when a $\sigma^{(j)}$ measurement is performed on it. Performing the above many times for each Pauli operator $\sigma^{(j)}$, we may determine with some level of confidence for which operator $\sigma^{(j)}$ this fails most often. This determines the pair of orthogonal states which Φ does the poorest job at keeping distinguishable; using Eqn. (3.6), we may then compute $\mathcal{R}(\Phi)$.

Appendix D

Analytics and Numerics of multi-round error tracking

In this appendix pseudocode describing the 'Brute Force' and 'Rare Events' simulation that produced Fig. 4.5 are described. Taken verbatim from Ref [118].

We limit our simulations to a limited set of k values, with $k_i = k$ for each simulation. This is an arbitrary choice and simplifies the comparison made in Fig. 4.5. For three rounds the simulations are limited to low k values, as these can be simulated in a reasonable timeframe and adequate statistics gathered to infer the output error rate.

1: select protocol: $\{k_1, k_2, k_{\text{Toff}}\}$
2: generate number of modules in each round: $\{M_1, M_2, M_{\text{Toff}}\}$ {Round One}
3: **for** $i < M_1$ **do**
4: randomly generate binary string v length n_1
5: measure stabilisers $G_0(k_1).v$
6: **if** stabilisers failed **then** return to line 3
7: **end if**
8: calculate logical output of module $v = G_1(k_1).v$
9: Append v to list of logical outputs V
10: **end for**
11: shuffle output of round 1 to firewall correlations $V \to V'$ {Round Two}
12: **for** $j < M_2$ **do**
13: **for** each block ii in a round 2 module (there are k_1) **do**
14: take $(ii \times j)^{\text{th}}$ string of length n_2 from V': v'
15: measure stabilisers $G_0(k_2).v'$
16: **if** stabilisers failed **then** return to line 1
17: **end if**
18: calculate logical output of module $w = G_1(k_2).v'$
19: Append w to list of logical outputs W
20: **end for**
21: **end for**
22: shuffle output of round 2 to firewall correlations $W \to W'$ {Round Three}
23: **for** $l < k_1 k_2$ **do**
24: measure stabilisers $G_0(k_3).w'$
25: **if** stabilisers failed **then return** FAIL
26: **end if** stabilisers passed
27: calculate logical output of module $G_1(k_3).w'$
28: Search for logical error in output
29: **if** logical error found **then return** ERROR
30: **else**logical error not found **return** SUCCESS
31: **end if**
32: **end for**

ALGORITHM D.1: Brute force simulation algorithm

1: select protocol: $\{k_1, k_2, k_{\text{Toff}}\}$
2: generate number of modules in each round: $\{M_1, M_2, M_{\text{Toff}}\}$
3: generate number of corrupt modules in round 2: N_2 {Round One}
4: **for** $i < N_2$ **do**
5: generate number of corrupt round 1 modules N_1
6: **for** $j < N_1$ **do**
7: randomly generate binary string v length n_1
8: measure stabilisers $G_0(k_1).v$
9: **if** stabilisers failed **then** return to line 6
10: **end if**
11: calculate logical output of module $G_1.v$
12: **if** there is a logical error **then** Append v to list of logical outputs V
13: **else** return to line 6
14: **end if**
15: **end for**
16: Pad V with 0s so it is length $k_1 n_2$
17: shuffle output of corrupt module to firewall correlations ... $V \to V'$ {Round Two}
18: **for** each block ii in a round 2 module (there are k_1) **do**
19: take $(ii)^{\text{th}}$ string of length n_2 from V': v'
20: measure stabilisers $G_0(k_2).v'$
21: **if** stabilisers failed **then** return to line 6
22: **end if**
23: calculate logical output of module $w = G_1(k_2).v'$
24: **if** there is a logical error **then** Append w to list W
25: **else** return to line 6
26: **end if**
27: **end for**
28: **end for**{ Round Three }
29: Pad W with 0s so it is length $k_1 k_2 n_3$
30: shuffle output of round 2 to firewall correlations $W \to W'$
31: **for** $l < k_1 k_2$ **do**
32: measure stabilisers $G_0(k_3).w'$
33: **if** stabilisers failed **then return** FAIL
34: **end if** stabilisers passed
35: calculate logical output of module $G_1(k_3).w'$
36: Search for logical error in output
37: **if** logical error found **then return** ERROR
38: **else**logical error not found **return** SUCCESS
39: **end if**
40: **end for**

ALGORITHM D.2: Rare events simulation algorithm

Bibliography

[1] R. P. Feynman, International Journal of Theoretical Physics **21**, 467 (1982), arXiv:9508027 [quant-ph] .

[2] D. Deutsch, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences **400**, 97 (1985), arXiv:0407008 [arXiv:quant-ph] .

[3] D. Deutsch and R. Jozsa, Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences **439**, 553 LP (1992).

[4] P. Shor, in *Proceedings 35th Annual Symposium on Foundations of Computer Science* (Santa Fe, NM, 1994) pp. 124–134, arXiv:9605043 [quant-ph] .

[5] S. Jordan, "Quantum Algorithm Zoo, http://math.nist.gov/quantum/zoo/," (2016).

[6] L. K. Grover, Proceedings of the twenty-eighth annual ACM symposium on Theory of computing - STOC '96 , 212 (1996), arXiv:9605043 [quant-ph] .

[7] T. P. Harty, D. T. C. Allcock, C. J. Ballance, L. Guidoni, H. A. Janacek, N. M. Linke, D. N. Stacey, and D. M. Lucas, Physical Review Letters **113** (2014), 10.1103/PhysRevLett.113.220501, arXiv:1403.1524 .

[8] Y. Wang, M. Um, J. Zhang, S. An, M. Lyu, J. N. Zhang, L. M. Duan, D. Yum, and K. Kim, Nature Photonics **11**, 646 (2017), arXiv:1701.04195 .

[9] R. Barends, J. Kelly, A. Megrant, D. Sank, E. Jeffrey, Y. Chen, Y. Yin, B. Chiaro, J. Mutus, C. Neill, P. O'Malley, P. Roushan, J. Wenner, T. C. White, A. N. Cleland, and J. M. Martinis, Physical Review Letters **111**, 080502 (2013), arXiv:1304.2322 .

[10] C. Wang, C. Axline, Y. Y. Gao, T. Brecht, Y. Chu, L. Frunzio, M. H. Devoret, and R. J. Schoelkopf, Applied Physics Letters **107** (2015), 10.1063/1.4934486, arXiv:1509.01854 .

[11] N. Bar-Gill, L. M. Pham, A. Jarmola, D. Budker, and R. L. Walsworth, Nature Communications **4**, 1743 (2013), arXiv:1211.7094 .

[12] A. M. Tyryshkin, S. Tojo, J. J. Morton, H. Riemann, N. V. Abrosimov, P. Becker, H. J. Pohl, T. Schenkel, M. L. Thewalt, K. M. Itoh, and S. A. Lyon, Nature Materials **11**, 143 (2012), arXiv:1105.3772 .

[13] D. Leibfried, B. DeMarco, V. Meyer, D. Lucas, M. Barrett, J. Britton, W. M. Itano, B. Jelenković, C. Langer, T. Rosenband, and D. J. Wineland, Nature **422**, 412 (2003).

[14] J. Benhelm, G. Kirchmair, C. F. Roos, and R. Blatt, Nature Physics **4**, 463 (2008), arXiv:0803.2798 .

[15] C. J. Ballance, T. P. Harty, N. M. Linke, M. A. Sepiol, and D. M. Lucas, Physical Review Letters **117**, 4 (2016), arXiv:1512.04600 .

[16] J. P. Gaebler, T. R. Tan, Y. Lin, Y. Wan, R. Bowler, A. C. Keith, S. Glancy, K. Coakley, E. Knill, D. Leibfried, and D. Wineland, Phys. Rev. Lett. **117**, 060505 (2016).

[17] C. Ospelkaus, U. Warring, Y. Colombe, K. R. Brown, J. M. Amini, D. Leibfried, and D. J. Wineland, Nature **476**, 181 (2011), arXiv:arXiv:1104.3573v1 .

[18] T. P. Harty, M. A. Sepiol, D. T. Allcock, C. J. Ballance, J. E. Tarlton, and D. M. Lucas, Physical Review Letters **117**, 1 (2016), arXiv:1606.08409 .

[19] C. J. Ballance, V. M. Schäfer, J. P. Home, D. J. Szwer, S. C. Webster, D. T. Allcock, N. M. Linke, T. P. Harty, D. P. Aude Craik, D. N. Stacey, A. M. Steane, and D. M. Lucas, Nature **528**, 384 (2015), arXiv:1505.04014 .

[20] T. R. Tan, J. P. Gaebler, Y. Lin, Y. Wan, R. Bowler, D. Leibfried, and D. J. Wineland, Nature **528**, 380 (2015), arXiv:1508.03392 .

[21] S. Weidt, J. Randall, S. C. Webster, K. Lake, A. E. Webb, I. Cohen, T. Navickas, B. Lekitsch, A. Retzker, and W. K. Hensinger, Physical Review Letters **117**, 220501 (2016), arXiv:1603.03384 .

[22] S. Debnath, N. M. Linke, C. Figgatt, K. A. Landsman, K. Wright, and C. Monroe, Nature **536**, 63 (2016), arXiv:1603.04512 .

[23] L. Dicarlo, J. M. Chow, J. M. Gambetta, L. S. Bishop, B. R. Johnson, D. I. Schuster, J. Majer, A. Blais, L. Frunzio, S. M. Girvin, and R. J. Schoelkopf, Nature **460**, 240 (2009), arXiv:0903.2030 .

[24] J. M. Chow, J. M. Gambetta, A. D. Corcoles, S. T. Merkel, J. A. Smolin, C. Rigetti, S. Poletto, G. A. Keefe, M. B. Rothwell, J. R. Rozen, M. B. Ketchen, and M. Steffen, Physical Review Letters **109**, 60501 (2012), arXiv:1202.5344 .

[25] R. Barends, J. Kelly, A. Megrant, A. Veitia, D. Sank, E. Jeffrey, T. C. White, J. Mutus, A. G. Fowler, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, C. Neill, P. O'Malley, P. Roushan, A. Vainsencher, J. Wenner, A. N. Korotkov, A. N. Cleland, and J. M. Martinis, Nature **508**, 500 (2014), arXiv:1402.4848 .

[26] A. Fruchtman and I. Choi, *https://nqit.ox.ac.uk/content/technical-roadmap-fault-tolerant-quantum-computing*, Tech. Rep. October (NQIT, UK National Quantum Technologies Programme, 2016).

[27] C. J. Ballance, T. P. Harty, N. M. Linke, M. A. Sepiol, and D. M. Lucas, Physical Review Letters **117** (2016), 10.1103/PhysRevLett.117.060504, arXiv:1512.04600 .

[28] P. W. Shor, Physical Review A **52**, 2493 (1995), arXiv:0506097 [arXiv:quant-ph] .

[29] A. R. Calderbank and P. W. Shor, Physical Review A - Atomic, Molecular, and Optical Physics **54**, 1098 (1996), arXiv:9512032 [quant-ph] .

[30] D. P. Di Vincenzo and P. W. Shor, Phys. Rev. Lett. **77**, 3260 (1996).

[31] D. Gottesman, Physical Review A **57**, 30 (1997), arXiv:9702029 [quant-ph] .

[32] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, Physical Review A - Atomic, Molecular, and Optical Physics **86**, 32324 (2012), arXiv:1208.0928 .

[33] J. R. Wootton, A. Peter, J. R. Winkler, and D. Loss, Physical Review A **96** (2017), 10.1103/PhysRevA.96.032338, arXiv:1608.05053 .

[34] P. W. Shor, in *Proceedings of 37th Conference on Foundations of Computer Science* (1996) pp. 56–65.

[35] A. M. Steane, Physical Review Letters **78**, 2252 (1997).

[36] R. Chao and B. W. Reichardt, arXiv preprint arXiv:1705.02329 **9**, 541 (2017), arXiv:1705.02329 .

[37] A. Y. Kitaev, A. Shen, and M. N. Vyalyi, *Classical and quantum computation*, Vol. 47 (American Mathematical Society Providence, 2002).

[38] B. Eastin and E. Knill, Phys. Rev. Lett. **102**, 110502 (2009).

[39] B. Zeng, A. Cross, and I. L. Chuang, IEEE Transactions on Information Theory **57**, 6272 (2011), arXiv:0706.1382 .

[40] A. Steane, Proc. R. Soc. London, Ser.A **452**, 2551 (1996), arXiv:9601029 [quant-ph] .

[41] S. Bravyi and A. Kitaev, Phys. Rev. A **71**, 022316 (2005).

[42] A. Y. Kitaev, **52**, 53 (1997).

[43] A. Y. Kitaev, in *Proceedings of the Third International Conference on Quantum Communication and Measurement*, Vol. 2013, edited by O. Hirota, A. S. Holevo, and C. M. Caves (1997) pp. 181–188.

[44] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, Journal of Mathematical Physics **43**, 4452 (2002), arXiv:0110143v1 [quant-ph] .

[45] A. G. Fowler, Physical Review Letters **109**, 180502 (2012).

[46] V. Kolmogorov, Mathematical Programming Computation **1**, 43 (2009).

[47] D. S. Wang, A. G. Fowler, A. M. Stephens, and L. C. L. Hollenberg, Quantum Information & Computation **10**, 14 (2009), arXiv:0905.0531 .

[48] A. Jamiolkowsky, Reports on Mathematical Physics **3**, 275 (1972).

[49] M. D. Choi, Linear Algebra and Its Applications **10**, 285 (1975).

[50] N. H. Nickerson, Y. Li, and S. C. Benjamin, Nature Communications **4**, 1756 (2013), arXiv:arXiv:1211.2217v3 .

[51] A. Y. Kitaev, Annals of Physics **303**, 2 (2003), arXiv:9707021 [quant-ph] .

[52] C. Horsman, A. G. Fowler, S. Devitt, and R. V. Meter, New Journal of Physics **14**, 123011 (2012), arXiv:1111.4022 .

[53] A. J. Landahl and C. Ryan-Anderson, arXiv:1407.5103 , 13 (2014), arXiv:1407.5103 .

[54] J. O'Gorman, N. H. Nickerson, P. Ross, J. J. L. Morton, and S. C. Benjamin, npj Quantum Information **2**, 15019 (2016).

[55] B. E. Kane, Nature **393**, 133 (1998).

[56] L. C. L. Hollenberg, A. D. Greentree, A. G. Fowler, and C. J. Wellard, Physical Review B - Condensed Matter and Materials Physics **74**, 45311 (2006), arXiv:0506198v2 [arXiv:quant-ph] .

[57] S. R. Schofield, N. J. Curson, M. Y. Simmons, F. J. Rueß, T. Hallam, L. Oberbeck, and R. G. Clark, Physical Review Letters **91**, 136104 (2003), arXiv:0307599 [cond-mat] .

[58] M. Fuechsle, J. A. Miwa, S. Mahapatra, H. Ryu, S. Lee, O. Warschkow, L. C. Hollenberg, G. Klimeck, and M. Y. Simmons, Nature Nanotechnology **7**, 242 (2012), arXiv:1505.03565 .

[59] G. Berman, G. Brown, M. Hawley, and V. Tsifrinovich, Physical review letters **87**, 097902 (2001).

[60] M. Schaffry, S. C. Benjamin, and Y. Matsuzaki, New Journal of Physics **14**, 1 (2012), arXiv:arXiv:1106.0613v1 .

[61] S. C. Benjamin and S. Bose, Phys. Rev. A **70**, 032314 (2004).

[62] J. J. Pla, K. Y. Tan, J. P. Dehollain, W. H. Lim, J. J. L. Morton, D. N. Jamieson, A. S. Dzurak, and A. Morello, Nature **489**, 541 (2012), arXiv:1305.4481 .

[63] J. T. Muhonen, J. P. Dehollain, A. Laucht, F. E. Hudson, R. Kalra, T. Sekiguchi, *et al.*, Nat Nano **9**, 986 (2014).

[64] J. T. Muhonen, A. Laucht, S. Simmons, J. P. Dehollain, R. Kalra, F. E. Hudson, S. Freer, K. M. Itoh, D. N. Jamieson, J. C. McCallum, A. S. Dzurak, and A. Morello, Journal of Physics Condensed Matter **27**, 154205 (2015), arXiv:1410.2338 .

[65] J. J. L. Morton, A. M. Tyryshkin, A. Ardavan, K. Porfyrakis, S. A. Lyon, and G. A. D. Briggs, Phys. Rev. Lett. **95**, 200501 (2005).

[66] V. Kolmogorov, Math. Prog. Comp. **1**, 1 (2008).

[67] J. Edmonds, Canadian Journal of Mathematics **17**, 449 (1965).

[68] A. G. Fowler and J. M. Martinis, Physical Review A **89**, 32316 (2014).

[69] C. Horsman, A. G. Fowler, S. Devitt, and R. V. Meter, New Journal of Physics **14**, 123011 (2012).

[70] R. Raussendorf and J. Harrington, Physical Review Letters **98**, 190504 (2007), arXiv:0610082 [quant-ph] .

[71] T. Harness and R. R. A. Syms, Journal of Micromechanics and Microengineering **10**, 7 (2000).

[72] D. Mukhopadhyay, J. Dong, E. Pengwang, and P. Ferreira, Sensors and Actuators A: Physical **147**, 340 (2008).

[73] J. Dong, D. Mukhopadhyay, and P. M. Ferreira, Journal of Micromechanics and Microengineering **17**, 1154 (2007).

[74] B. Koo, X. Zhang, J. Dong, S. Salapaka, and P. Ferreira, Journal of Microelectromechanical Systems **21**, 13 (2012).

[75] L. L. Chu and Y. B. Gianchandani, Journal of Micromechanics and Microengineering **13**, 279 (2003).

[76] H.-H. Liao, H.-H. Shen, B.-T. Liao, Y.-J. Yang, Y.-C. Chen, and W.-W. Pai, 5th IEEE International Conference on Nano/Micro Engineered and Molecular Systems (NEMS) , 549 (2010).

[77] S. J. Hile, M. G. House, E. Peretz, J. Verduijn, D. Widmann, T. Kobayashi, S. Rogge, and M. Y. Simmons, Applied Physics Letters **107**, 093504 (2015).

[78] G. Pica, G. Wolfowicz, M. Urdampilleta, M. L. W. Thewalt, H. Riemann, N. V. Abrosimov, *et al.*, Phys. Rev. B **90**, 195204 (2014).

[79] A. Laucht, J. T. Muhonen, F. A. Mohiyaddin, R. Kalra, J. P. Dehollain, S. Freer, F. E. Hudson, M. Veldhorst, R. Rahman, G. Klimeck, K. M. Itoh, D. N. Jamieson, J. C. McCallum, A. S. Dzurak, and A. Morello, Science Advances **1** (2015), 10.1126/sciadv.1500022.

[80] A. Morello, J. J. Pla, F. A. Zwanenburg, K. W. Chan, K. Y. Tan, H. Huebl, *et al.*, Nature **467**, 687 (2010).

[81] J. J. L. Morton, A. M. Tyryshkin, R. M. Brown, S. Shankar, B. W. Lovett, A. Ardavan, T. Schenkel, E. E. Haller, J. W. Ager, and S. A. Lyon, Nature **455**, 1085 (2008), arXiv:0803.2021 .

[82] C. Vieu, F. Carcenac, A. Pépin, Y. Chen, M. Mejias, A. Lebib, *et al.*, Applied Surface Science **164**, 111 (2000).

[83] D. M. Toyli, C. D. Weis, G. D. Fuchs, T. Schenkel, and D. D. Awschalom, Nano Letters **10**, 3168 (2010).

[84] C. D. Weis, A. Schuh, A. Batra, A. Persaud, I. W. Rangelow, J. Bokor, *et al.*, Journal of Vacuum Science & Technology B **26**, 2596 (2008).

[85] D. N. Jamieson, C. Yang, T. Hopf, S. M. Hearne, C. I. Pakes, S. Prawer, *et al.*, Applied Physics Letters **86** (2005).

[86] L. Robledo, L. Childress, H. Bernien, B. Hensen, P. F. Alkemade, and R. Hanson, Nature **477**, 574 (2011), arXiv:1301.0392v1 .

[87] W. Pfaff, B. J. Hensen, H. Bernien, S. B. Van Dam, M. S. Blok, T. H. Taminiau, M. J. Tiggelman, R. N. Schouten, M. Markham, D. J. Twitchen, and R. Hanson, Science **345**, 532 (2014), arXiv:1404.4369 .

[88] G. Waldherr, Y. Wang, S. Zaiser, M. Jamali, T. Schulte-Herbrüggen, H. Abe, T. Ohshima, J. Isoya, J. F. Du, P. Neumann, and J. Wrachtrup, Nature **506**, 204 (2014), arXiv:arXiv:1309.6424v1 .

[89] N. Bar-Gill, L. M. Pham, A. Jarmola, D. Budker, and R. L. Walsworth, Nat Commun **4**, 1743 (2013).

[90] F. Dolde, H. Fedder, M. W. Doherty, T. Nöbauer, F. Rempp, G. Balasubramanian, T. Wolf, F. Reinhard, L. Hollenberg, F. Jelezko, *et al.*, Nature Physics **7**, 459 (2011).

[91] O. A. Williams, Diamond and Related Materials **20**, 621 (2011).

[92] S. Pezzagna, D. Wildanger, P. Mazarov, A. D. Wieck, Y. Sarov, I. Rangelow, B. Naydenov, F. Jelezko, S. W. Hell, and J. Meijer, Small **6**, 2117 (2010).

[93] M. Lesik, P. Spinicelli, S. Pezzagna, P. Happel, V. Jacques, O. Salord, *et al.*, physica status solidi (a) **210**, 2055 (2013).

[94] Y.-C. Chen, P. S. Salter, S. Knauer, L. Weng, A. C. Frangeskou, C. J. Stephen, S. N. Ishmael, P. R. Dolan, S. Johnson, B. L. Green, G. W. Morley, M. E. Newton, J. G. Rarity, M. J. Booth, and J. M. Smith, Nature Photonics **11**, 77 (2016).

[95] B. Naydenov, V. Richter, J. Beck, M. Steiner, P. Neumann, G. Balasubramanian, J. Achard, F. Jelezko, J. Wrachtrup, and R. Kalish, Applied Physics Letters **96**, 10 (2010).

[96] M. S. Grinolds, S. Hong, P. Maletinsky, L. Luan, M. D. Lukin, R. L. Walsworth, and A. Yacoby, Nature Physics **9**, 215 (2013), arXiv:1209.0203 .

[97] M. Veldhorst, J. C. Hwang, C. H. Yang, A. W. Leenstra, B. De Ronde, J. P. Dehollain, J. T. Muhonen, F. E. Hudson, K. M. Itoh, A. Morello, and A. S. Dzurak, Nature Nanotechnology **9**, 981 (2014), arXiv:arXiv:1407.1950v1 .

[98] X. Xu, N. D. Beaudrap, J. O'Gorman, and S. C. Benjamin, New Journal of Physics **20**, 023009 (2018), arXiv:1707.09951 .

[99] P. Schindler, J. T. Barreiro, T. Monz, V. Nebendahl, D. Nigg, M. Chwalla, M. Hennrich, and R. Blatt, Science **332**, 1059 (2011).

[100] D. Nigg, M. Müller, E. A. Martinez, P. Schindler, M. Hennrich, T. Monz, M. A. Martin-Delgado, and R. Blatt, Science **345**, 302 (2014), arXiv:1403.5426 .

[101] N. M. Linke, M. Gutierrez, K. A. Landsman, C. Figgatt, S. Debnath, K. R. Brown, and C. Monroe, arXiv:1611.06946 , 1 (2016), arXiv:1611.06946 .

[102] M. D. Reed, L. Dicarlo, S. E. Nigg, L. Sun, L. Frunzio, S. M. Girvin, and R. J. Schoelkopf, Nature **482**, 382 (2012), arXiv:1109.4948 .

[103] J. Kelly, R. Barends, A. G. Fowler, A. Megrant, E. Jeffrey, T. C. White, D. Sank, J. Y. Mutus, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, I.-C. Hoi, C. Neill, P. J. J. OMalley, C. Quintana, P. Roushan, A. Vainsencher, J. Wenner, A. N. Cleland, and J. M. Martinis, Nature **519**, 66 (2015).

[104] A. D. Córcoles, E. Magesan, S. J. Srinivasan, A. W. Cross, M. Steffen, J. M. Gambetta, and J. M. Chow, Nat. Comms. **6**, 7979 (2015).

[105] D. Rist, S. Poletto, M.-Z. Huang, A. Bruno, V. Vesterinen, O.-P. Saira, and L. DiCarlo, Nature Comms. **6**, 6983 (2015).

[106] J. Cramer, N. Kalb, M. A. Rol, B. Hensen, M. S. Blok, M. Markham, D. J. Twitchen, R. Hanson, and T. H. Taminiau, Nature Communications **7**, 11526 (2016), arXiv:1508.01388 .

[107] D. Gottesman, arXiv preprint arXiv:1610.03507 (2016), arXiv:1610.03507 .

[108] T. Harty, D. Allcock, C. Ballance, L. Guidoni, H. Janacek, N. Linke, D. Stacey, and D. Lucas, Phys. Rev. Lett. **113**, 220501 (2014).

[109] N. de Beaudrap and S. C. Benjamin, In preparation (2018).

[110] D. Aharonov, A. Kitaev, and N. Nisan, in *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98 (ACM, New York, NY, USA, 1998) pp. 20–30.

[111] R. J. Blume-Kohout, APS March Meeting (2017).

[112] L. S. Bishop, S. Bravyi, A. Cross, J. M. Gambetta, and J. A. Smolin, *Executive Summary IBM Research*, Tech. Rep. (2017).

[113] A. Bermudez, X. Xu, R. Nigmatullin, J. O'Gorman, V. Negnevitsky, P. Schindler, T. Monz, U. G. Poschinger, C. Hempel, J. Home, F. Schmidt-Kaler, M. Biercuk, R. Blatt, S. Benjamin, and M. Müller, Physical Review X **7** (2017), 10.1103/PhysRevX.7.041061, arXiv:1705.02771 .

[114] A. W. Cross, D. P. DiVincenzo, and B. M. Terhal, Quantum Information and Computation **9**, 541 (2007), arXiv:0711.1556 .

[115] R. Laflamme, C. Miquel, J. P. Paz, and W. H. Zurek, Physical Review Letters **77**, 198 (1996), arXiv:9602019 [quant-ph] .

[116] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, Journal of Mathematical Physics **43**, 4452 (2002).

[117] Y. Tomita and K. M. Svore, Physical Review A - Atomic, Molecular, and Optical Physics **90**, 62320 (2014), arXiv:1404.3747 .

[118] J. O'Gorman and E. T. Campbell, Physical Review A **95**, 32338 (2017).

[119] A. M. Meier, B. Eastin, and E. Knill, Quant. Inf. and Comp. **13**, 195 (2013).

[120] S. Bravyi and J. Haah, Phys. Rev. A **86**, 052329 (2012).

[121] C. Jones, Phys. Rev. A **87**, 042305 (2013).

[122] A. G. Fowler, S. J. Devitt, and C. Jones, Scientific reports **3**, 1939 (2013).

[123] V. Kliuchnikov, D. Maslov, and M. Mosca, Physical review letters **110**, 190502 (2013).

[124] A. Paetznick and K. M. Svore, Quantum Information & Computation **14**, 1277 (2014).

[125] D. Gosset, V. Kliuchnikov, M. Mosca, and V. Russo, Quantum Information & Computation **14**, 1261 (2014).

[126] N. J. Ross and P. Selinger, arXiv preprint arXiv:1403.2975 (2014).

[127] M. Amy and M. Mosca, arXiv preprint arXiv:1601.07363 (2016).

[128] J. T. Anderson, G. Duclos-Cianci, and D. Poulin, Phys. Rev. Lett. **113**, 080501 (2014).

[129] H. Bombin, arXiv preprint arXiv:1311.0879 (2013).

[130] H. Bombin and M. A. Martin-Delgado, Physical review letters **97**, 180501 (2006).

[131] H. Bombin, R. W. Chhajlany, M. Horodecki, and M. A. Martin-Delgado, New Journal of Physics **15**, 055023 (2013).

[132] B. J. Brown, N. H. Nickerson, and D. E. Browne, arXiv preprint arXiv:1503.08217 (2015).

[133] C. Wang, J. Harrington, and J. Preskill, Annals of Physics **303**, 31 (2003).

[134] R. Raussendorf and J. Harrington, Phys. Rev. Lett. **98**, 190504 (2007).

[135] R. Raussendorf, J. Harrington, and K. Goyal, New Journal of Physics **9**, 199 (2007), quant-ph/0703143 .

[136] A. G. Fowler, A. C. Whiteside, A. L. McInnes, and A. Rabbani, Physical Review X **2**, 041003 (2012).

[137] D. Gottesman and I. Chuang, Nature **402**, 390 (1999).

[138] S. Bravyi and R. Koenig, Physical Review Letters **110** (2013), 10.1103/PhysRevLett.110.170503, arXiv:1206.1609 .

[139] E. T. Campbell and D. E. Browne, Physical Review Letters **104**, 030503 (2010), arXiv:0908.0836 .

[140] E. T. Campbell and D. E. Browne, in *On the Structure of Protocols for Magic State Distillation BT - Theory of Quantum Computation, Communication, and Cryptography: 4th Workshop, TQC*, edited by A. Childs and M. Mosca (Springer Berlin Heidelberg, Berlin, Heidelberg, 2009) pp. 20–32.

[141] A. Paetznick and B. W. Reichardt, Phys. Rev. Lett. **111**, 090505 (2013).

[142] B. Eastin, Phys. Rev. A **87**, 032321 (2013).

[143] C. Jones, Phys. Rev. A **87**, 022328 (2013).

[144] J. von Neumann, in *Probabilistic logics and the synthesis of reliable organisms from unreliable components, Automata Studies* (Princeton Univ. Prss, Princeton, NJ, 1956) pp. 43–98.

[145] E. T. Campbell and M. Howard, Physical Review A **95**, 22316 (2017), arXiv:1606.01904 .

[146] E. T. Campbell and M. Howard, Physical Review Letters **118**, 1 (2017), arXiv:1606.01906 .

[147] A. G. Fowler, arXiv preprint arXiv:1210.4626 (2012).

[148] D. Poulin, Phys. Rev. Lett. **95**, 230504 (2005).

[149] D. Bacon, Phys. Rev. A **73**, 012340 (2006).

[150] P. Aliferis and A. W. Cross, Phys. Rev. Lett. **98**, 220502 (2007).

[151] H. Bombin, Phys. Rev. A **81**, 032301 (2010).

[152] A. J. Landahl and C. Ryan-Anderson, arXiv preprint arXiv:1407.5103 (2014).

[153] S. D. Barrett and P. Kok, Phys. Rev. A **71**, 060310 (2005).

[154] L. Jiang, J. M. Taylor, A. S. Sørensen, and M. D. Lukin, Phys. Rev. A **76**, 062323 (2007).

[155] D. K. L. Oi, S. J. Devitt, and L. C. L. Hollenberg, Phys. Rev. A **74**, 052313 (2006).

[156] D. L. Moehring, P. Maunz, S. Olmschenk, K. C. Younge, D. N. Matsukevich, L. M. Duan, and C. Monroe, Nature **449**, 68 (2007).

[157] E. T. Campbell and S. C. Benjamin, Phys. Rev. Lett. **101**, 130502 (2008).

[158] E. Campbell and J. Fitzsimons, Int. J. Quantum Inf. **8**, 219 (2010).

[159] H. Bernien, B. Hensen, W. Pfaff, G. Koolstra, M. S. Blok, L. Robledo, T. H. Taminiau, M. Markham, D. J. Twitchen, L. Childress, and R. Hanson, Nature **497**, 86 (2013).

[160] N. H. Nickerson, J. F. Fitzsimons, and S. C. Benjamin, Phys. Rev. X **4**, 041041 (2014).

[161] D. Herr, F. Nori, and S. J. Devitt, arXiv preprint arXiv:1608.05208 (2016).

[162] N. C. Jones, R. Van Meter, A. G. Fowler, P. L. McMahon, J. Kim, T. D. Ladd, and Y. Yamamoto, Phys. Rev. X **2**, 031007 (2012).

[163] J. T. Anderson, "On the power of reusable magic states," ArXiv:1205.0289.

[164] J. T. Anderson, G. Duclos-Cianci, and D. Poulin, Physical Review Letters **113**, 80501 (2014), arXiv:1403.2734 .

[165] Y. Li, New Journal of Physics **17**, 5 (2015), arXiv:1410.7808 .

[166] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, Phys. Rev. A **86**, 032324 (2012).

[167] P. Selinger, Phys. Rev. A **87**, 042302 (2013).

[168] L. Rao, X. Xu, and C. Adamo, ACS Catalysis **6**, 1567 (2016), arXiv:1605.03590 .

[169] A. Fowler, Private Communication.

[170] B. J. Brown, N. H. Nickerson, and D. E. Browne, Nature Comm. **7**, 12302 (2016).

[171] S. Bravyi and A. Cross, arXiv preprint arXiv:1509.03239 (2015).

[172] G. Duclos-Cianci and K. M. Svore, Phys. Rev. A **88**, 042325 (2013).

[173] A. J. Landahl and C. Cesare, arXiv preprint arXiv:1302.3240 (2013).

[174] G. Duclos-Cianci and D. Poulin, Phys. Rev. A **91**, 042315 (2015).

[175] E. T. Campbell and J. O'Gorman, Quant. Sci. Tech. **1**, 015007 (2016).

[176] E. T. Campbell and J. O'Gorman, arXiv:1603.04230v2 (2016), 10.1088/2058-9565/1/1/015007, arXiv:1603.04230 .

[177] N.˜Ross and P.˜Selinger, arXiv:1403.2975 (2014), arXiv:1403.2975 .

[178] A. Bocharov, M. Roetteler, and K. M. Svore, Physical Review A **91**, 052317 (2015).

[179] B. Bauer, D. Wecker, A. J. Millis, M. B. Hastings, and M. Troyer, Physical Review X **6**, 31045 (2016).

[180] K. Temme, S. Bravyi, and J. M. Gambetta, Physical Review Letters **119**, 180509 (2017).

[181] Y. Li and S. C. Benjamin, Physical Review X **7**, 21050 (2017).

[182] Duclos-Cianci, private communication.

[183] A. De and L. P. Pryadko, Physical Review A **93**, 42333 (2016), arXiv:1509.01239 .

[184] D. P. DiVincenzo and P. Aliferis, Physical review letters **98**, 020501 (2007).

[185] H. Goto, Scientific Reports **6**, 19578 (2016).