Robust Operation and Applications of Emerging Quantum Technologies

Zhenyu Cai

Lincoln College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2020

This thesis is dedicated to
my beloved parents *Lijiao Chen* and *Kunsheng Cai*
for their unconditional love and support throughout my life,
and my beloved fiancée *Yifan Chen*
whose smile could brighten up even the darkest day.

谨以此文献给
我挚爱的父母 陈丽娇，蔡坤胜
感谢他们在我生命中对我无条件的爱与支持，
和我挚爱的未婚妻 陈怡帆
就算最黑暗的日子也可以被她的笑容点亮。

Acknowledgements

The past three years has been an incredible journey for me with an abundance of help and support from a great number of people. Thus I would like to take this opportunity to express my most sincere gratitude to the following people, without whom my journey to date would not be possible:

First and foremost, *Simon Benjamin*, for being the best PhD supervisor that one can hope for. You are incredibly patient, generous with your time and you always look out for the best interests for all of us. I could never fathom how you can juggle so many duties, yet still find the time to help us in every single way you can. You always manage to see a problem on a more fundamental level, giving me a fresh angle to the problem after every discussion. Without you, none of my achievements to date would be possible. I couldn't possibly thank you enough, for helping me grow both as a researcher and as a person.

Sam McArdle, for racing me to the office every day and for having all the answers to anything quantum chemistry related. You would approach problems in such a steady and down-to-earth manner, which I learnt a lot from. We probably spent the most time in the office together and I benefited greatly from all the inspiring and fruitful discussions we had.

Tyson Jones, The Mathematica Wizard, The QuEST Genie and The Prophet For All Code Questions, for your relentless energy in research and in life, and for all of your thought-provoking questions that can always spark interesting discussions in the office, not to forget all of your legendary stories (often food-related, cough cough) that are surely some of the most indelible memories of the group.

Suguru Endo, for always being humble and curious, for bringing all the joy and laughter to the office, and for being such a good companion when we went to conferences together. I still recall vividly when we drank teas together by a foggy West Lake, it was great fun! 本当にありがとう, 遠藤傑!

Xiaosi Xu, for being meticulous about all the projects we collaborated on and for answering all my questions when I began my PhD. I am sure you will flourish in your new position. 谢谢晓思!

Xiao Yuan, for having the answers to all the hard problems I had and for the fun time in numerous hotpot gatherings. Also thank you for taking us around Beijing when we visited and introducing us to a bunch of amazing people. We should really collaborate at some point! 谢谢袁骁!

Ying Li, for being the ultimate portal of hard physics and maths problems beyond *Xiao*. Thank you for hosting me as a visitor in Beijing. You are always very humble, down-to-earth and not afraid to tackle the toughest problems head-on, setting yourself as an example for me to follow as a researcher. 谢谢李颖!

Bálint Koczor, for being the gourmet chef in the group (who has read "the chefs bible") that cooked one of the nicest meals for us. You are always very humble and gentle and you can seemingly tackle any hard problems thrown at you.

The rest of the QTech Theory group, *Takahiro Tsunoda*, *Carlos Outeiral*, *Richard Meister*, *Sam Jaques*, *Armands Strikis* and *Qi Zhao*, for the fun time at the Christmas dinners, in the pub trips and in the virtual pub trips! You are one of the most fun and supportive crowds to work with, that I am proud to be part of.

John Morton, for being such a wonderful collaborator that meticulously went through all the details in the project, which could not come to fruition without you. Thank you for going the extra miles and helping me with all of my job applications, sharing your experience with me. None of my offers would be possible without you.

The rest of the UCL/Quantum Motion Team: *Michael Fogarty*, *Sofia Patomäki* and *Simon Schaal* for being the dream collaborators that always found time to answer all my questions related to actual hardware and being a fun crowd to chat with every time I went down to London. *James Palles-Dimmock*, for helping me out with various patent applications and for always being humble and curious about my research.

Jonathan Baugh, for being an excellent collaborator and for helping me with my various applications. Even though our project did not find its way into this thesis due to length constraint, it is the project that taught me so many things when I first started my PhD.

David Khmelnitskii, for helping me in so many ways during my undergrad. You set up extra classes to teach us the physics that we were interested in, arranged sections to discuss with us possible paths after graduations and helped me in many of my PhD applications. I could never be where I am today without your help, and I could only hope that I can follow your footsteps and grasp a non-negligible fraction of your endless physics knowledge someday.

Finally, I would like to thank all of my family and friends, whether here in the UK, at home, or elsewhere in the world, for all of your help and support in my research and in my life.

Abstract

Quantum computers promise huge speed-up over conventional computers in crucial problems like chemistry and materials simulations, decryption, and even machine learning. However, any imperfect manipulations of the qubits from which the machines are formed and any interactions with the environment can lead to errors and loss of their quantum properties. This thesis will develop practical schemes for fighting such errors in the quantum hardware and discuss possible applications of such noisy machines.

We can protect the quantum information against noise by employing additional qubits and performing quantum error correction, whose experimental implementation has been brought much closer to reality via the recent rapid advance of quantum hardware, but challenges still remain. One example is coherent errors, which can grow much faster than regular errors. In this thesis, we tackle it by improving upon the conventional scheme that transforms it into regular errors using twirling, and also by developing a new scheme called Pauli conjugation that makes use of its coherent properties to our advantage. Implementations of quantum error correction codes also bring about hardware challenges like wiring packing and leakage errors. For the silicon spin qubit platform, these can be overcome by a surface code architecture we develop.

It is challenging to use quantum error correction for the applications of near-term quantum hardware due to the constrained qubit count. Instead, we rely on quantum error mitigation to fight noise, which makes use of extra measurements instead of extra qubits. The Fermi-Hubbard variational quantum eigensolver is one of the most promising near-term quantum algorithms, thus we have performed a resource estimation for the task to gauge its feasibility. We then improve and combine the existing error mitigation schemes to boost their performance and thus to enhance the feasibility of near-term quantum algorithms, bringing practical applications of quantum hardware closer to reality.

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 3: Z. Cai and S. C. Benjamin, "Constructing Smaller Pauli Twirling Sets for Arbitrary Error Channels", Scientific Reports 9, 1–11 (2019).
- Chapter 4: Z. Cai, X. Xu, and S. C. Benjamin, "Mitigating coherent noise using Pauli conjugation", npj Quantum Information 6, 1–9 (2020).
- Chapter 5: Z. Cai, M. A. Fogarty, S. Schaal, S. Patomäki, S. C. Benjamin, and J. J. L. Morton, "A Silicon Surface Code Architecture Resilient Against Leakage Errors", Quantum 3, 212 (2019).
- Chapter 7: Z. Cai, "Resource Estimation for Quantum Variational Simulations of the Hubbard Model", Physical Review Applied 14, 014059 (2020).
- Chapter 8: Z. Cai, "Multi-exponential Error Extrapolation and Combining Error Mitigation Techniques for NISQ Applications", arXiv:2007.01265 [quant-ph] (2020).

Additional publications involving the author:

- B. Buonacorsi, Z. Cai, E. B. Ramirez, K. S. Willick, S. M. Walker, J. Li, B. D. Shaw, X. Xu, S. C. Benjamin, and J. Baugh, "Network architecture for a topological quantum computer in silicon", Quantum Science and Technology 4, 025003 (2019).

Contents

Nature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical, and by golly it's a wonderful problem, because it doesn't look so easy.

— Richard Feynman

1 Introduction

Classical computers have revolutionised our society in every possible aspect. However, there are still problems that are fundamentally out of reach for classical computers. One of them is the simulation of large quantum systems. In 1982, Feynman [1] suggested that we can turn the problem around and use quantum systems to make computers that are more powerful than the classical ones, which was later rigorously formulated by Deutsch [2]. In 1994, Shor discovered a polynomial-time quantum algorithm for factorisation [3], which can be used to break modern-day cryptography schemes like RSA. Shortly after, Grover discovered a quantum algorithm for database searching with a quadratic speed-up [4]. These early discoveries spurred an explosion of interests in the possible applications and the implementations of quantum computers and the field has flourished ever since.

While classical computers consist of *bits*, which can be in the state of 0 *or* 1, quantum computers consist of *qubits*, which can be in the *superposition* of both the states 0 and 1 at the same time. The *amplitude* of a given basis state in a quantum state is a complex number, whose modulus square is the probability that we obtain the given basis as the measurement outcome. Unlike classical probabilities, amplitudes can be both positive and negative, and thus amplitudes can *interfere* with each other constructively or destructively to boost or reduce the probability of a given measurement outcome. The basic idea of a quantum algorithm is just

to carefully design a series of quantum operations such that the amplitude of the basis state that corresponds to the right answer will interfere constructively and become the dominant measurement outcome. For a comprehensive introduction to the basic concepts of quantum computation and quantum algorithms, we refer the readers to the famed "Mike and Ike" [5].

For a quantum system, the loss of its quantum properties and thus its ability to interfere is called *decoherence*, which is one of the largest obstacles to the practical realisation of quantum computers. Any interactions between the quantum system and the environment will lead to decoherence, thus the quantum computer needs to be isolated from the classical environment. On the other hand, we need precise control and readout in the quantum computer to carry out quantum algorithms, which requires strong coupling between the quantum system and the classical controls. Such dichotomy means it is extremely difficult to make a quantum computer while suppressing both errors due to interaction with the environment and errors due to imperfect controls. The goal of this thesis is to develop schemes that enable the robust operation and applications of quantum devices in the presence of noise.

The solution to this was pioneered by Shor [6] and Steane [7] in the name of *quantum error correction*, in which we try to distribute the quantum information stored in a small number of qubits into a large number of qubits through quantum *entanglement*. In such a way, any local interaction with the environment will not be able to extract information from the collective state and destroy its interference. Similarly, any local disturbance due to imperfect control will not corrupt the information of the collective state. Hence, quantum error correction will enable us to preserve and operate on our quantum information in the presence of local noise, which we will discuss in more details in Chapter 2. However, quantum error correction will only work if we can prevent error accumulation by constantly detecting and correcting the errors. A certain type of errors called coherent errors is particularly damaging due to its rapid rate of accumulation compared to regular errors, for which we will develop efficient schemes to combat them in Chapter 3 and Chapter 4. When we try to implement quantum error correction in real hardware,

many other challenges emerge. In Chapter 5, we will introduce a way to implement one of the leading quantum error correction schemes – the surface code – in silicon spin qubits, which overcomes practical hardware challenges like dense wiring of the classical control lines and leakage errors.

Due to the large qubit overhead required by quantum error correction, implementing error-corrected fault-tolerant quantum algorithms remains a long term goal. The recent advance in quantum hardware has enabled quantum computers to venture into tasks that are classically intractable [8] without quantum error correction. To use these noisy near-term machines for any practically useful results, one needs to employ *quantum error mitigation*, which uses additional measurements instead of additional qubits to mitigate the damages done by noise. We will give a brief overview of quantum error mitigation in Chapter 6 and perform a resource estimation for one of the most promising tasks to be implemented on near-term machines in Chapter 7. What we will find is that improvements and combination of the existing error mitigation techniques are necessary for any practically meaningful near-term quantum algorithms, which is what we will develop and discuss in Chapter 8. At the end of this thesis, we will summarise and conclude our findings.

2 Quantum Error Correction

Contents

2.1 Pauli Operators

2.1.1 Pauli Group

The set of n-qubit *Pauli operators* is defined as:

$$\mathbb{G} = \{I, X, Y, Z\}^{\otimes n} \tag{2.1}$$

where X, Y, Z are the single-qubit Pauli operators given by:

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

The Pauli operators $\mathbb{G}$ will be a group under a composition rule that ignore all the phase factors, thus we will also call it the *Pauli group*. In this thesis, whenever we are talking about the Pauli group, its subgroups and any group-related concepts like generators, we will always assume this phase-less composition rule. This is different from the the conventional definition of Pauli group with the additional phase factors $\{\pm 1, \pm i\}$. However, all of our results will still be consistent with the conventional result because in the case that we need to apply any group-related concepts, like in the stabiliser formalism (Section 2.3), twirling (Chapter 3), conjugation (Chapter 4) or group channels (Chapter 8), the only key structure of the group that we care about is the commutation relationship between the elements, which remains unchanged under the multiplication of any phase factors on any elements in the group.

We will use $\widetilde{\ }$ to denote the subset of independent elements of a given set, which will just be the generators in the case that the given set forms a group. The standard generators of the Pauli group are:

$$\widetilde{\mathbb{G}} = \{X_1, X_2, \cdots, X_n, Z_1, Z_2, \cdots, Z_n\} \tag{2.2}$$

where X_k/Z_k denotes the X/Z operator acting on the k^{th}qubit.

2.1.2 Commutators

For two given operators A and B, their *commutator* $\eta(A, B)$ is defined to be:

$$AB = \eta(A, B)BA. \tag{2.3}$$

Note that the commutator remains unchanged under the multiplication of any non-zero constant factors on any of its operators:

$$\eta(A, B) = \eta(\alpha A, \beta B) \quad \forall \alpha, \beta \in \mathbb{C},\ \alpha, \beta \neq 0. \tag{2.4}$$

All Pauli operators will either commute or anti-commute with each other:

$$\eta(F, G) \in \{-1, +1\} \quad \forall F,\ G \in \mathbb{G}.$$

Therefore, the composition of the commutators of Pauli operators follows the same structure as the composition of the Pauli operators themselves:

$$\eta(FF', G) = \eta(F, G)\eta(F', G) \quad \forall F,\ F',\ G \in \mathbb{G}. \tag{2.5}$$

Hence, for a subgroup of the Pauli group $\mathbb{F} \subseteq \mathbb{G}$ whose generators is $\widetilde{\mathbb{F}}$ and a given Pauli operator $G \in \mathbb{G}$, we have:

$$\begin{aligned} \frac{1}{|\mathbb{F}|} \sum_{F \in \mathbb{F}} \eta(F, G) &= \prod_{\widetilde{F} \in \widetilde{\mathbb{F}}} \frac{1 + \eta(\widetilde{F}, G)}{2} \\ &= \begin{cases} 1 & \text{if } \eta(\widetilde{F}, G) = 1 \quad \forall \widetilde{F} \in \widetilde{\mathbb{G}} \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{2.6}$$

2.1.3 Pauli Transfer Matrix Formalism

In the Pauli transfer matrix formalism [9], the density operators are cast into vector form by decomposing into the Pauli basis $G \in \mathbb{G}$:

$$\rho = \frac{1}{2^n} \sum_{G \in \mathbb{G}} \mathrm{Tr}(G\rho) G \quad \Rightarrow \quad |\rho\rangle\!\rangle = \sum_{G \in \mathbb{G}} |G\rangle\!\rangle \langle\!\langle G|\rho\rangle\!\rangle$$

where we have defined the inner product as:

$$\langle\!\langle G|\rho\rangle\!\rangle = \frac{1}{\sqrt{2^n}} \mathrm{Tr}(G\rho).$$

We have added a scaling factor $\frac{1}{\sqrt{2^n}}$ when we use the Pauli operators as a basis, where n is the number of qubits. This is to ensure the normalisation of the basis set $\{|G\rangle\!\rangle\}$.

In such a way, a general quantum channel $\mathcal{E}$ can be written in matrix form:

$$\mathcal{E} = \sum_{G, G' \in \mathbb{G}} |G'\rangle\!\rangle \langle\!\langle G'|\mathcal{E}|G\rangle\!\rangle \langle\!\langle G|$$

with the matrix elements given by

$$\mathcal{E}_{G'G} = \langle\!\langle G'|\mathcal{E}|G\rangle\!\rangle = \langle\!\langle G'|\mathcal{E}(G)\rangle\!\rangle = \frac{1}{2^n} \mathrm{Tr}(G'\mathcal{E}(G)).$$

2.2 Error Channels

In this thesis, we will use $\overline{\ }$ to denote a super-operator:

$$\sum_E \overline{E}\,\rho = \sum_E E\rho E^\dagger. \tag{2.7}$$

A general error channel $\mathcal{E}$ can be written in its Kraus form:

$$\mathcal{E}(\rho) = \sum_E \overline{E}\,\rho \qquad \text{with } \sum_E E^\dagger E = I. \tag{2.8}$$

Pauli Errors

In the case of $\overline{E} = p_G\,\overline{G}$ for $G \in \mathbb{G}$ (note that p_G can be 0), we have:

$$\mathcal{E}(\rho) = \sum_{G\in\mathbb{G}} p_G\,\overline{G}\,\rho, \tag{2.9}$$

which is just a *Pauli error channel*, with p_G being the error probability of the Pauli component G.

Pauli noise plays an important role in quantum error correction research since its effects on quantum error correction codes can be simulated efficiently using the Gottesman–Knill theorem [10] and also because some of the common noise models in practice like depolarising noise and dephasing noise are Pauli noise.

Coherent Errors

The degree of coherence in an error channel can be quantified by how well it preserves the purity of an average incoming state [11]. Hence, the "most coherent" errors are just unitary errors, while incoherent errors generally refer to Pauli channels. *Coherent errors* characterise a broad spectrum of noise that sits closer to the unitary errors than the Pauli errors.

The rate of error accumulation is bounded by the worst-case error rate –*the diamond distance*. However, experimentally we can only measure the average error rate –*the fidelity* – efficiently. For Pauli errors (incoherent errors), the worst-case error rate is similar to the average case error rate, while for unitary errors (coherent errors), their worst-case error rate can scale as the square root of the average error

rate. Hence, at the same average error rate, coherent errors can be potentially more damaging than incoherent errors due to a larger worst-case error rate and thus a faster rate of error accumulation [12–18]. However, there are cases in which coherent errors are amenable to coherent control solutions like dynamic decoupling [19, 20] and thus can be much better mitigated than stochastic errors.

2.3 Stabiliser Code

2.3.1 Stabiliser Formalism

A *stabiliser code* is defined by a subgroup $\mathbb{S}$ of the Pauli group $\mathbb{G}$ called the *stabilisers*. The *code space*, i.e. the subspace that our quantum information lives in, is defined as space spanned by the states that are simultaneous $+1$ eigenstates for all the stabilisers:

$$\mathrm{V}_{\mathbb{S}} = \{|\psi\rangle \mid S|\psi\rangle = |\psi\rangle \ \ \forall S \in \mathbb{S}\}\,. \tag{2.10}$$

To have a non-trivial code space, we must have all the elements in $\mathbb{S}$ commuting with each other. In fact, this is the only restriction we need to define a valid stabiliser group [1].

The quantum information stored in the code space can be represented using *logical qubits*. By definition, the stabilisers act trivially on any states of the logical qubits, i.e. they map to the logical identity. All the logical operations on the logical qubits need to commute with the logical identity and thus need to commute with all the stabilisers. Hence, we can define the set of logical operators as the set of Pauli operators that commute with $\mathbb{S}$:

$$\overline{\mathbb{G}} := \{G \in \mathbb{G} \mid \forall S \in \mathbb{S},\ \eta(S, G) = 1\}$$

which is just the centraliser of $\mathbb{S}$ [2].

[1] Note that in the conventional case in which there are phase factors in the Pauli group, then we need to add one more condition: all phase factors besides $+1$ will not be in $\mathbb{S}$.

[2] The centraliser and the normaliser are the same for Pauli operators since all Pauli operators either commute or anti-commute.

The weight of a Pauli operator G is the number of qubits that the operator acts non-trivially on, denoted as $|G|$. The distance of a quantum code is the minimal weight of all non-trivial logical operators:

$$d = \min_{\overline{G} \in \overline{\mathbb{G}} - \mathbb{S}} |\overline{G}|.$$

In the presence of local noise, a code with distance d can correct arbitrary errors with weights up to $\lfloor d/2 \rfloor$. For a code with k logical qubits encoded into n physical qubits with a distance d, we usually denote it as an [[n, k, d]] code.

All the remaining Pauli operators $\mathbb{E} = \mathbb{G} - \overline{\mathbb{G}}$ will take a logical state out of the code space and thus will result in errors detectable via stabiliser measurements.

From Eq. (2.2) we know that there are $2n$ generators in the Pauli group

$$\left|\widetilde{\mathbb{G}}\right| = 2n,$$

which can be partitioned into three subsets for a given stabiliser code:

- Stabiliser Generators $\widetilde{\mathbb{S}}$:

  $$\mathbb{S} = \langle \widetilde{\mathbb{S}} \rangle.$$

  $\widetilde{\mathbb{S}}$ represent the constraints we place on the code space. Thus $\left|\widetilde{\mathbb{S}}\right| = n - k$ will result in k logical qubits. Following the definition of $\mathbb{S}$, all elements in $\widetilde{\mathbb{S}}$ must commute with each other.

- Logical Generators $\widetilde{\overline{\mathbb{G}}}$:

  $$\overline{\mathbb{G}} = \langle \widetilde{\overline{\mathbb{G}}} + \widetilde{\mathbb{S}} \rangle = \langle \widetilde{\overline{\mathbb{G}}} \rangle \times \mathbb{S}.$$

  For k logical qubits, we need $2k$ logical generators: $|\widetilde{\overline{\mathbb{G}}}| = 2k$. They need to be chosen such that for any given element in $\widetilde{\overline{\mathbb{G}}}$ there is one and only one other element in $\widetilde{\overline{\mathbb{G}}}$ that anti-commutes with it, so that we can map them to the logical operators $\overline{X}$ and $\overline{Z}$ on different logical qubits. Following the definition of $\overline{\mathbb{G}}$, elements in $\widetilde{\overline{\mathbb{G}}}$ commute with all elements in $\widetilde{\mathbb{S}}$.

- Error Generators $\widetilde{\mathbb{E}}$:

$$\widetilde{\mathbb{E}} = \widetilde{\mathbb{G}} - \widetilde{\overline{\mathbb{G}}} - \widetilde{\mathbb{S}}.$$

We will choose $\widetilde{\mathbb{E}}$ such that for any given element $\widetilde{S}_i$ in $\widetilde{\mathbb{S}}$, there is one and only one element $\widetilde{E}_i$ in $\widetilde{\mathbb{E}}$ that anti-commutes with it. In such away, by decomposing any Pauli operators into the generators, we can quickly identify its syndrome, which is just its commutation relation with the set of stabiliser generators $\widetilde{\mathbb{S}}$. Note that we have used the label 'error generators' here since each such element creates a code violation, but physical error process can give rise to elements in any of these three sets of generators, and in particular those in $\widetilde{\overline{\mathbb{G}}}$ which create undetectable logical errors.

2.3.2 Error Correction in Stabiliser Code

Since the code space is defined as the +1 eigenspace of all the stabiliser generators $\widetilde{\mathbb{S}}$ (and thus of all the stabilisers), we can detect whether a state has gone out of the code space by performing measurements of all the stabiliser generators $\widetilde{S}_i \in \widetilde{\mathbb{S}}$, which are also called stabiliser checks. For a Pauli error E, the measurement result of $\widetilde{S}_i$ will be $s_i = \eta(E, \widetilde{S}_i) \in \{-1, 1\}$. Using $s_i = (-1)^{m_i}$, we can define the measurement *syndrome* $\vec{m}$ with its entries being $m_i \in \{0, 1\}$. The stabiliser checks will collapse the noisy state into the corresponding $\vec{m}$-syndrome subspace using the projection operator:

$$\Pi_{\vec{m}} = \prod_{i=1}^{|\widetilde{\mathbb{S}}|} \frac{\left(1 + (-1)^{m_i}\, \widetilde{S}_i\right)}{2}.$$

Note that the $\vec{0}$-syndrome subspace is just the code space.

The process of translating the measurement syndrome $\vec{m}$ into the correction operation is called decoding. Let $E_{\vec{m}}$ be one of the Pauli errors that can lead to the syndrome $\vec{m}$ (e.g. $E_{\vec{m}} = \prod_{i=1}^{|\widetilde{\mathbb{E}}|} \widetilde{E}_i^{m_i}$), then the set of all the Pauli errors that can lead the same syndrome will be $E_{\vec{m}}\overline{\mathbb{G}}$. It can be partitioned into different cosets of $\mathbb{S}$, each represented by $E_{\vec{m}}\overline{G}\mathbb{S}$ for different $\overline{G} \in \overline{\mathbb{G}}$. Within each coset, all the operators differ by composition with a stabiliser, i.e. they are logically

equivalent. The total error probability of each coset of logically equivalent operators is $P(E_{\vec{m}}\overline{G}\mathbb{S}) = \sum_{S\in\mathbb{S}} P(E_{\vec{m}}\overline{G}S)$. In the optimal decoding process, which is called *maximum-likelihood decoding*, for a given code defined by $\mathbb{S}$ with the error syndrome $\vec{m}$, we will identify the coset $E_{\vec{m}}\overline{G}\mathbb{S}$ with maximum $P(E_{\vec{m}}\overline{G}\mathbb{S})$ out of different $\overline{G} \in \overline{\mathbb{G}}$, and reapply any element in $E_{\vec{m}}\overline{G}\mathbb{S}$ (usually the one with the minimal weight) as the recovery operation to minimise the probability of making a logical error after correction.

Let us suppose we are performing maximum-likelihood decoding on a stabiliser code that encodes k logical qubits into n physical qubits. There will be 4^k different coset probabilities $P(E_{\vec{m}}\overline{G}\mathbb{S})$ we need to calculate. And to calculate each of them we need to sum up the probability all 2^{n-k} elements within the coset. Thus, in total, we need to calculate $4^k 2^{n-k} = 2^{n+k}$ probabilities to be able to perform maximum likelihood decoding, i.e. the cost is exponential in n. Hence in practice we will often turn to other decoders that strike a good balance between runtime efficiency and decoding optimality.

One alternative is the minimal-weight decoder, in which we just look for the operator $E_{\vec{m}}\overline{G}$ that has the minimal weight for all $\overline{G} \in \overline{\mathbb{G}}$ and pick it to be the recovery operator. For local noise, as the local error probability becomes smaller, the sum of error probabilities of each error coset $E_{\vec{m}}\overline{G}\mathbb{S}$ will be increasingly dominated by the error with the minimum weight, thus minimal-weight decoding will move closer to the optimal maximum-likelihood decoding. The minimal-weight decoder is what we will be primarily using for surface codes in Section 2.6.

2.3.3 Examples of Stabiliser Codes

[[4,2,2]] error-detection code

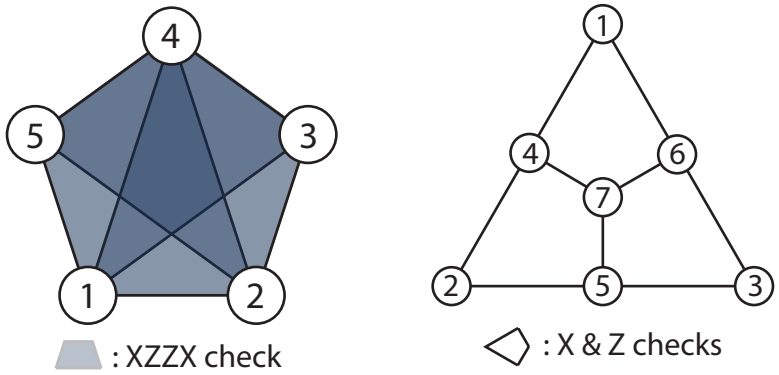
This is the smallest non-trivial quantum code capable of detecting a general error. It consists of 4 qubits with the 2 stabiliser checks being the X and Z parities of all 4 qubits. Its logical operator $\overline{X}$ is performing X on any two qubits, similarly for $\overline{Z}$, thus it has distance 2.

[[5,1,3]] error-correction code

This is the smallest quantum code that can detect and correct any single-qubit errors (since its distance is 3). It has four stabilisers that are $XXZZ$ operators acting on different subset of qubits as shown in Fig. 2.1 (a). Its logical operator $\overline{X}$ is performing X on all five qubits, similarly for $\overline{Z}$. These logical operators can compose with the stabilisers to give weight-3 logical operators, thus it has distance 3.

Steane Code

The Steane Code is a [[7,1,3]] code. It is the smallest of a family of codes called the 2D colour codes [21]. Its stabilisers are 4-qubit X and Z operators acting on the three plaquettes shown in Fig. 2.1 (b). Its logical operator $\overline{X}$ is performing X on all seven qubits, similarly for $\overline{Z}$. These logical operators can compose with the stabilisers to give weight-3 logical operators, thus it has distance 3.



(a) The five-qubit error-correction code.

(b) The Steane code.

Figure 2.1: The layout of two example stabiliser codes

2.4 Code Concatenation and Threshold Theorem

The idea of code concatenation is simply to further encode the physical qubits into codes, i.e. a concatenated code is a code within a code. The concatenation of a $[[n_1, 1, d_1]]$ code with a $[[n_2, 1, d_2]]$ code, will yield a code with $n_1 n_2$ physical qubits and a distance $\geq d_1 d_2$. The decoding of a concatenated code can simply be carried

out level by level independently, which is called 'hard decoding' [22]. Alternatively, we can pass the syndrome information to the next concatenation level for a more optimal but less efficient decoding scheme named 'soft decoding' [23].

A code with distance d can fail if $t = \lfloor \frac{d+1}{2} \rfloor$ or more qubits are corrupted. Now suppose each qubit is corrupted with an independent small probability p_0, then the probability that the code fails is roughly $p_1 = Cp_0^t$, where C is the number of ways these t errors can be distributed.

To obtain a higher level of protection, we can further encode the qubits again using the same encoding scheme. As we go to the second encoding level, the probability of failure of the code is now $p_2 = Cp_1^t = C(Cp_0^t)^t$. By recursively applying our encoding scheme, at the k^{th} encoding level, we have:

$$\begin{aligned} p_k &= \frac{1}{C}\left(Cp_0\right)^{t^k} \\ &= p_{th}\left(\frac{p_0}{p_{th}}\right)^{t^k} \end{aligned}$$

Hence, as long as we have p_0 below the threshold $p_{th} = \frac{1}{C}$, which means that the gain of encoding outweighs the cost of encoding, we can suppress the error rate of the logical qubit to as low as we want by increasing the number of encoding levels.

The number of qubits needed to encode one logical qubit grows exponentially with the increase in the number of concatenation levels k, while the logical error rate decreases at a double exponential rate. For the logical components to achieve a certain error rate ϵ, the number of physical qubits we need is in the order of $O(\text{poly}(-\log(\epsilon)))$.

The error threshold is a much more widely applicable concept beyond concatenated code as we will see later when we look into the surface code in Section 2.6.

2.5 Fault Tolerance

In practice the stabiliser checks are performed using quantum circuits like the one shown in Fig. 2.2, which can consist of faulty components. Fault-tolerant quantum

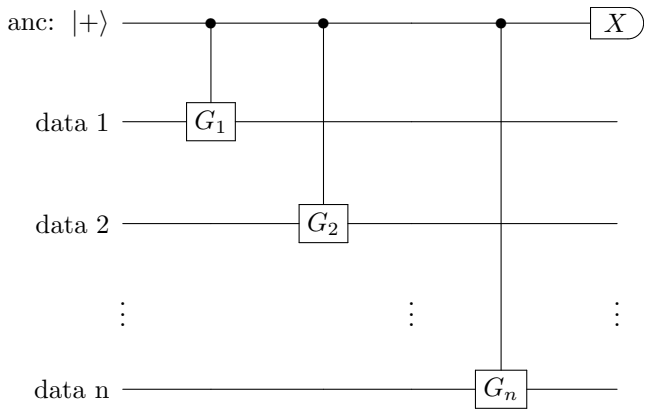


Figure 2.2: The circuit for performing the stabiliser check of $\widetilde{S} = G_1 \otimes G_2 \otimes \cdots \otimes G_n$ for $G_i \in \{I,\ X,\ Y,\ Z\}$.

error correction simply means that even with these faulty components in the quantum error correction circuit, we can still remove at least the leading order errors.

To combat errors in state preparation and gates, we need to design circuits and error checking procedures such that an individual local error cannot propagate and subsequently corrupt the logical qubits. The simplest way to achieve this is by encoding the ancilla qubit in Fig. 2.2 into the GHZ state: $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \rightarrow \frac{1}{\sqrt{2}}(|00\cdots0\rangle + |11\cdots1\rangle)$, and modifying the circuit such that each ancilla qubit will interact with one and only one data qubit [24]. Note that the GHZ state also needs to be prepared fault-tolerantly with the methods outlined in Ref. [24]. Alternatively, we can encode the ancilla qubit using the same code as the data qubits [25] or using additional flag qubits to catch the error propagation events in the ancilla [26].

The measured syndrome can be faulty due the the errors above and measurement errors. The simplest way to fight syndrome errors is by repeating the stabiliser measurements and performing majority vote, which is analogous to the classical repetition code.

2.6 Surface Codes

2.6.1 Background

As discussed in Section 2.4, once the physical error rate is below a certain threshold, we can suppress the error rate of the logical qubit indefinitely by scaling up the

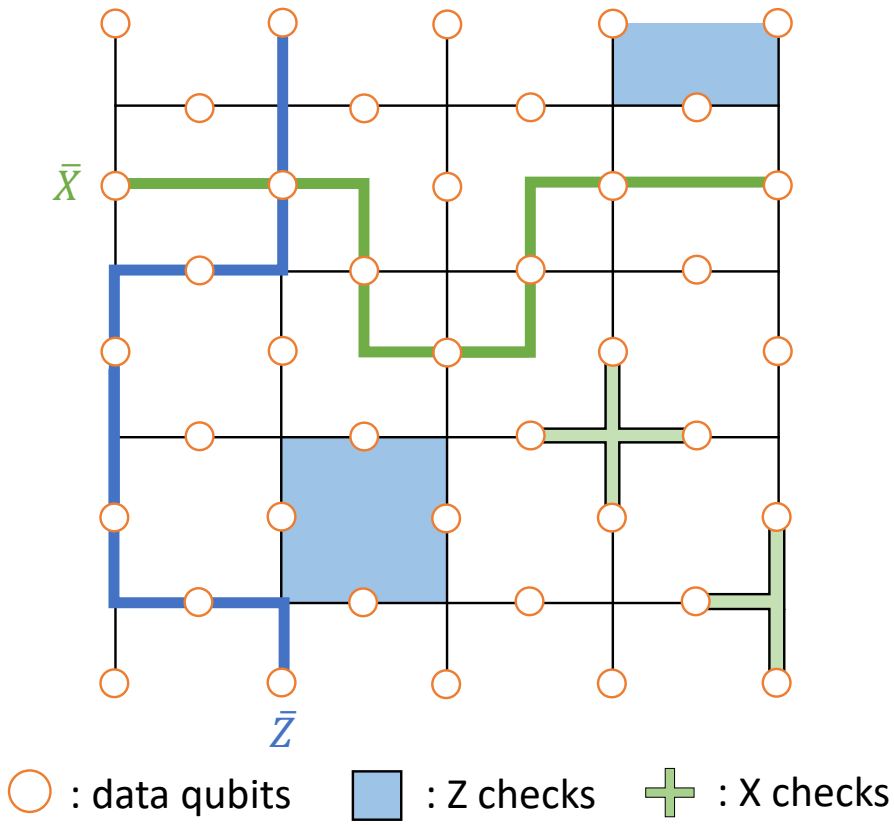


Figure 2.3: A distance-5 square surface code.

code via concatenation. However, at high concatenation level, the stabiliser checks that we need to perform will consist of logical operations that span over a large number of physical qubits, which is challenging to implement in practice. Hence, here we turn to *topological codes*, which are constructed with the geometric layout of the qubits in mind, and thus can maintain local constant-weight stabiliser checks as we scale up the code. In particular, the geometry that is most compatible for scaling up on many qubit platforms will be the 2D planar layout. Hence, much of the research focus has been devoted to 2D topological codes like the 2D colour code and the surface code. In this thesis, we will mainly be focusing the surface code, whose parity checks have lower weights compared to colour codes and thus are easier for practical implementation [3].

Surface codes are extensions of toric codes [28] by replacing the closed boundaries

[3]However, do note that on the other hand, 2D colour code allows transversal implementation of the full logical Clifford group [27] while the surface code does not.

on the torus with open boundaries through fixing the virtual boundary qubits to be always error-free [29]. The qubit layout and the stabilisers of a surface code are shown in Fig. 2.3. Looking at the underlying lattice, the edges map to the physical qubits, the vertices map to the X checks on the four neighbouring qubits (three at the left and right *smooth* boundaries) and the plaquettes map to the Z checks on the four neighbouring qubits (three at the top and bottom *rough* boundaries). The stabiliser checks are indeed local and low-weight as expected from a topological code. Any closed loops of X or Z operations on the lattice will just be another stabiliser operator that can be obtained by composing the stabiliser generators. In Fig. 2.3, we have depicted a 5×5 surface code. In general for a $L \times L$ surface code, we will have $n = L^2 + (L-1)^2$ qubits and $2L(L-1)$ independent stabiliser checks, thus it stores one logical qubit. The logical operator generators are also shown in Fig. 2.3, in which $\overline{X}$ is just *any* string of X operators connecting the smooth boundaries while $\overline{Z}$ is just *any* string of Z operators connecting the rough boundaries. The minimal length of such a string spans L qubits, thus the distance of a $L \times L$ surface code is just L, which scales as $\sqrt{n}$.

2.6.2 Error Correction in Surface Codes

The surface code is a CSS code in which we have independent X and Z stabiliser checks that will deal with Z and X noise, respectively. Hence, for simplicity, we can consider decoding the syndromes from the X and Z checks separately. As shown in Fig. 2.4, an error string in the bulk will result in two failed checks at both of its end points, and an error string at the boundary will result in one failed check. We will call these failed checks the *defects* in the lattice. During the error correction process, we are given a list of defects to infer the error strings present. In the limit of low error probability and thus low defect density, it is natural to just connect all the nearby defects in the bulk and connect the remaining defects to the nearest boundaries to infer the error string and apply the relevant correction. More generally, we will look at all defects in the code and pair them either with each other or with the boundaries to achieve the minimal total length of

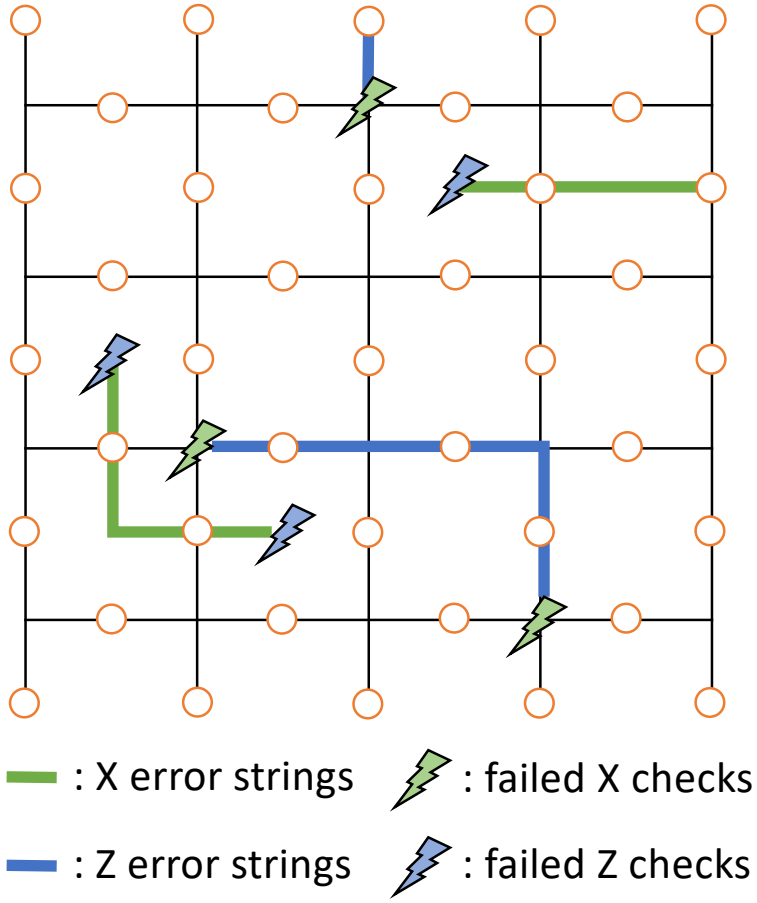


Figure 2.4: A distance-5 square surface code with failed stabiliser checks.

all the inferred error strings. We see that this is just the minimum-weight decoding described in Section 2.3.2 applied to surface codes, which can be computed using Edmond's minimum-weight matching algorithm [30]. It has a readily available `C++` implementation [31]. There are many different error strings that can lead to the same pair of defects. As long as we have paired up the defects correctly, any correction we apply will form a closed loop with the actual underlying error, which is just a stabiliser that will act trivially on our logical qubit.

In the above decoding scheme, we have been treating the syndromes from the X and Z checks separately. This will be fine in the case of independent X and Z noise. However, in the case of for example depolarising errors in which there are Y errors present, then we might want to consider both X and Z syndromes together. As mentioned in Section 2.3.2, minimum-weight decoders are efficient

but suboptimal. To implement the optimal maximum likelihood decoding in the context of surface code, when try to pair up two defects, we need to consider the total error probability of all possible error strings that connect them instead of just the shortest string. Maximal likelihood decoding in surface code can actually be mapped to the phase transition in 2D random-bond Ising model. Solving for the transition temperature yields the error threshold for surface codes, which is $p_{th} \sim 18.9\%$ for depolarising noise [32].

2.6.3 Fault-tolerant Threshold

Up to now we have only considered perfect stabiliser checks. However, as discussed in Section 2.5 we need to consider the errors in the stabiliser check circuit to achieve fault tolerance. In surface code, the error propagation during stabiliser measurements can actually be kept under control by merely ordering the gate sequence in a careful manner without needing additional ancillae [33]. The syndrome errors are overcome by repeating more rounds of stabiliser checks. We can view it as adding a time dimension to our surface code stabiliser measurements, with each time slice recording the defects detected in the surface code in one full round of stabiliser checks [34, 35]. In the fault-tolerant setting, the defects instead of being the failed stabiliser checks, they become the stabiliser checks that produce a *different* result compared to the previous time slice (except for the first time slice). After repeating enough measurements to accumulate enough time slices, we can perform a similar defect pairing scheme as in the perfect syndrome measurement case, but now the defects live in a 3D lattice instead of 2D. The number of times we need to repeat the syndrome measurements is related to the syndrome error probability.

As mentioned in Section 2.2, Pauli circuit noise can be efficiently simulated classically to obtain the threshold of the code. The fault-tolerant threshold of the surface codes has been obtained via classical simulations with the implementation details outlined in Refs. [33, 35, 36]. The threshold p_{th} sits within the range of $0.5 \sim 1.1\%$ under different noise models and different stabiliser-check circuits [37].

As mentioned before a code with distance d can be corrupted if more than $\lfloor \frac{d}{2} \rfloor$ errors occur, thus we will expect the logical error rate p_L scale as $p^{\lfloor \frac{d}{2} \rfloor}$ where p is the physical error rate. As calculated by Fowler *et al.* [33], the empirical formula for the logical error rate of surface codes is:

$$p_L \simeq 0.03 \left(\frac{p}{p_{th}} \right)^{\lfloor \frac{d}{2} \rfloor}.$$

Assuming $p_{th} = 1\%$ and $p = 0.1\%$, to achieve $p_L = 10^{-14}$, which is required for performing Shor's algorithm with a reasonable success probability [33], we need $d \sim 25$. Hence, with a gate error rate of $p = 0.1\%$, we need around $n = d^2 + (d-1)^2 \sim 1000$ qubits per logical qubit to achieve fault-tolerant quantum memory using surface codes.

3

Mitigating Coherent Noise: Constructing Smaller Pauli Twirling Sets

This chapter is adapted from the research published in Scientific Reports [38], of which the present author is the main author.

Contents

3.1 Introduction

As discussed in Section 2.2, coherent errors can be much more damaging than Pauli errors due to faster error accumulation. At the physical qubit level, coherent noise

can be mitigated using dynamical decoupling [19, 20], however there are limitations due to imperfect control pulses and finite pulse durations and intervals. In the context of quantum error correction, local physical coherent noise will be decohered at the logical level as the code scales up [39]. Their damage to the encoded state can be mitigated by using better decoders [22]. Gate-level coherent errors in a quantum error correction circuit can be mitigated by splitting the stabiliser check into two oppositely rotating halves [40] with some requirements on the gates available to the given architecture. A more general solution, however, would involve using Pauli twirling to turn coherent noise into a Pauli channel [41–44].

Twirling is a technique that has been long established in the quantum information literature. It was first used for mapping a diverse range of states into a canonical form in entanglement purification [41, 42]. Then it appeared again as an integral part in randomised benchmarking [43, 45] and was also used to reduce the number of experimental runs needed in quantum process tomography [44, 46], both are critical in benchmarking the performance of quantum systems, especially "Noisy Intermediate-Scale Quantum" (NISQ) systems [47]. More recently, twirling was used as means to boost the performance of NISQ through error mitigations [48–51] in which it enables simpler characterisations and mitigations of local error channels.

In this chapter, twirling is discussed as a technique to mitigate the damage caused by coherent errors. Twirling means each time we execute a circuit, we conjugate the given noise using a twirling gate selected from the twirling set, and we iterate over all possible twirling gates in the twirling set to obtain the average result. By choosing the twirling set to be the full set of Pauli operators, we can convert any noise channel into a Pauli channel whose noise elements correspond to the Pauli basis of the original noise [52]. In practice, the set of twirling gates is usually exponentially large such that we can only sample from it. If we can reduce the size of the twirling set, we might be able to iterate over the full twirling set to get the exact results without the shot noise due to sampling over the twirling set. Moreover, a smaller twirling set allows us to choose twirling gates that have

higher fidelities and/or act on fewer qubits. This can reduce the number of errors we introduce into the system due to twirling.

In this chapter, we will introduce a way to exploit the structure of the noise channel to reduce the size of the Pauli twirling set needed for the channel. The chapter is organised as follows. In Section 3.2, we introduce the theory of Pauli twirling, in which we obtain the requirement on the twirling set. In Section 3.3, we show a way to construct a twirling set that satisfied the conditions that we laid out. This is followed by an example. In Section 3.4, we discuss how to use stabiliser measurements to further reduce the size of our twirling set. Lastly, Section 3.5 provides a summary of our results and some possible future directions.

3.2 Pauli Twirling

3.2.1 Exact Twirling and Random Twirling

One can think of twirling as a super-super-operator that transforms one super-operator into another. It can transform coherent noise channels like $\frac{1}{2}\left(\overline{I+Z}\right)$ into incoherent channels like $\frac{1}{2}\left(\overline{I}+\overline{Z}\right)$, where $\overline{\ }$ denotes super-operators as defined in Eq. (2.7). Applying exact twirling $\mathcal{T}_{\mathbb{W}}$ using the twirling set $\mathbb{W}$ on the noise operator E is defined as:

$$\mathcal{T}_{\mathbb{W}}(\overline{E}) = \frac{1}{|\mathbb{W}|}\sum_{W\in\mathbb{W}} \overline{WEW^{\dagger}}\,. \tag{3.1}$$

In other words, each time we run the circuit, we conjugate the noise operator E with a different twirling gate W from the twirling set $\mathbb{W}$. After we iterate over the whole twirling set $\mathbb{W}$ and take the average of the results, we effectively have the process above.

The goal of twirling is to turn the noise operator E into a Pauli channel:

$$\mathcal{T}_{\mathbb{W}}(\overline{E}) = \sum_{G\in\mathbb{G}} p_G\,\overline{G}\,.$$

where p_G is the probability of the Pauli error G happening, and $\mathbb{G}$ is the Pauli group as defined in Section 2.1.1.

On the other hand, in random twirling, instead of systematically iterating over the whole twirling set $\mathbb{W}$, each run we choose a random element W_n from the twirling set $\mathbb{W}$:

$$\mathcal{T}^{rand}_{\mathbb{W},N}(\overline{E}) = \frac{1}{N}\sum_{n=1}^{N}\overline{W_n E W_n^\dagger}.$$

At finite N, there will be shot noise associated with the output of random twirling due to imperfect sampling over the twirling set. The shot noise can be reduced by increasing the number of runs N, allowing the effect of random twirling to approach the effect of exact twirling:

$$\lim_{N\to\infty}\mathcal{T}^{rand}_{\mathbb{W},N} = \mathcal{T}_{\mathbb{W}}.$$

In this chapter, we will focus on exact twirling, but most of the results are also applicable to random twirling. The details of how to apply twirling on erroneous quantum components are outlined in Appendix A.1.

3.2.2 Requirements of the Pauli Twirling Set

Now we will focus on Pauli twirling, which means our twirling set consists of only Pauli operators: $\mathbb{W} \subseteq \mathbb{G}$.

We can break any n-qubit noise operator E into its Pauli basis:

$$E = \frac{1}{2^n}\sum_{G\in\mathbb{G}}\mathrm{Tr}(GE)G = \frac{1}{2^n}\sum_{V\in\mathbb{V}}\mathrm{Tr}(VE)V$$

where set $\mathbb{V}$ is the Pauli basis of E:

$$\mathbb{V} = \{G \in \mathbb{G} \mid \mathrm{Tr}(GE) \neq 0\}. \tag{3.2}$$

Substituting this into Eq. (3.1) and applying it onto a state ρ, we have:

$$\mathcal{T}_{\mathbb{W}}(\overline{E})\rho = \frac{1}{|\mathbb{W}|}\frac{1}{2^{2n}}\sum_{V,V'\in\mathbb{V}}\mathrm{Tr}(VE)\,\mathrm{Tr}\left(V'E^\dagger\right)\sum_{W\in\mathbb{W}}WVW\rho WV'W, \tag{3.3}$$

where we have also use the fact that Pauli operators are Hermitian.

Now let us look at the sum over $\mathbb{W}$, we have

$$\sum_{W\in\mathbb{W}}WVW\rho WV'W = V\rho V'\sum_{W\in\mathbb{W}}\eta(W,V)\eta(W,V') = V\rho V'\sum_{W\in\mathbb{W}}\eta(W,VV'). \tag{3.4}$$

Here $\eta(A, B)$ is just the commutator of the operators A and B as defined in Eq. (2.3).

Substituting this into Eq. (3.3) we get:

$$\mathcal{T}_{\mathbb{W}}(\overline{E})\rho = \frac{1}{2^{2n}|\mathbb{W}|} \sum_{V,V' \in \mathbb{V}} \mathrm{Tr}(VE)\,\mathrm{Tr}\Big(V'E^{\dagger}\Big)\Big(\sum_{W\in\mathbb{W}} \eta(W, VV')\Big)V\rho V'$$

$$= \frac{1}{2^{2n}} \sum_{V\in\mathbb{V}} |\mathrm{Tr}(VE)|^2 V\rho V + \frac{1}{2^{2n}|\mathbb{W}|} \sum_{\substack{V,V' \in \mathbb{V} \\ V \neq V'}} \mathrm{Tr}(VE)\,\mathrm{Tr}\Big(V'E^{\dagger}\Big)\Big(\sum_{W\in\mathbb{W}} \eta(W, VV')\Big)V\rho V' \tag{3.5}$$

where we have made use the fact that $\eta(W, VV) = \eta(W, I) = 1$.

Here we have essentially rewritten the Pauli-twirled channel into the process matrix form. For this Pauli-twirled channel to be a Pauli channel, we require all the $V \neq V'$ terms to vanish. By the definition of $\mathbb{V}$ in Eq. (3.2), we know that $\mathrm{Tr}(VE) \neq 0$ and $\mathrm{Tr}\Big(V'E^{\dagger}\Big) \neq 0$, thus we need to choose a $\mathbb{W}$ such that

$$\sum_{W\in\mathbb{W}} \eta(W, VV') = 0 \quad \forall V,\ V' \in \mathbb{V} \text{ and } V \neq V'. \tag{3.6}$$

In such a case, the result of twirling the noise operator E is just

$$\mathcal{T}_{\mathbb{W}}(\overline{E}) = \frac{1}{2^{2n}} \sum_{V\in\mathbb{V}} |\mathrm{Tr}(VE)|^2\, \overline{V} \tag{3.7}$$

which is just the diagonal elements of the process matrix of the untwirled noise operator E. Our arguments can be easily extended to the full noise channel by treating E as a Kraus operator and adding $\sum_E$ in front of all the equations. In such case, $\mathbb{V}$ will be re-defined as the Pauli basis needed to construct all the noise elements in the noise channel. All the other results follow.

3.3 Construction of the Pauli Twirling Set

3.3.1 Overall Ideas

As we can see from the last section, the key to twirling is to find a twirling set $\mathbb{W}$ that can satisfy Eq. (3.6) for the Pauli basis $\mathbb{V}$ of the given noise.

The common choice is $\mathbb{W} = \mathbb{G}$, the full set of Pauli operators. In such a way, for any $V \neq V'$ (i.e. $VV' \neq I$), the number of elements in $\mathbb{G}$ that commute with VV' will always equal to the number of elements that anti-commute with VV',

thus Eq. (3.6) is always satisfied. Hence, if we choose $\mathbb{W} = \mathbb{G}$, we can transform any error channel into a Pauli channel.

However, as mentioned before, twirling with the full Pauli set is not always ideal. A systematic way to construct $\mathbb{W}$ of a smaller size is laid out in this section. Before proceeding to the steps of construction, we need to introduce the idea of a commutator table first.

For $\mathbb{A} \subseteq \mathbb{G}$, $\mathbb{B} \subseteq \mathbb{G}$, their *commutator table* $\eta(\mathbb{A}, \mathbb{B})$ is a matrix with the entries being $\eta(A, B)$ for $A \in \mathbb{A}$, $B \in \mathbb{B}$:

	B_1	B_2	$\cdots$
A_1	$\eta(A_1, B_1)$	$\eta(A_1, B_2)$	$\cdots$
A_2	$\eta(A_2, B_1)$	$\eta(A_2, B_2)$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\ddots$

Similarly we can define the *commutator vectors* $\eta(\mathbb{A}, B)$ and $\eta(A, \mathbb{B})$ which correspond to the columns and rows, respectively, in the commutator table $\eta(\mathbb{A}, \mathbb{B})$. Using Eq. (2.5), we can compose these commutator vectors using element-wise multiplication to obtain new commutator vectors:

$$\begin{aligned} \eta(\mathbb{A}, BB') &= \eta(\mathbb{A}, B) \odot \eta(\mathbb{A}, B') \\ \eta(AA', \mathbb{B}) &= \eta(A, \mathbb{B}) \odot \eta(A', \mathbb{B}). \end{aligned} \tag{3.8}$$

Generator tables are defined as commutator tables whose elements are of the form:

$$\eta(\widetilde{Q}_i, \widetilde{H}_j) = 1 - 2\delta_{ij}.$$

Example generator tables $\eta(\widetilde{\mathbb{Q}}, \widetilde{\mathbb{H}})$ for different sizes of $\widetilde{\mathbb{H}}$ are just:

	$\widetilde{H}_1$
$\widetilde{Q}_1$	-1

$\left|\widetilde{\mathbb{H}}\right| = 1$

	$\widetilde{H}_1$	$\widetilde{H}_2$
$\widetilde{Q}_1$	-1	1
$\widetilde{Q}_2$	1	-1

$\left|\widetilde{\mathbb{H}}\right| = 2$

	$\widetilde{H}_1$	$\widetilde{H}_2$	$\widetilde{H}_3$
$\widetilde{Q}_1$	-1	1	1
$\widetilde{Q}_2$	1	-1	1
$\widetilde{Q}_3$	1	1	-1

$\left|\widetilde{\mathbb{H}}\right| = 3$

$\cdots\cdots$

Note that by definition, we have

$$\left|\widetilde{\mathbb{H}}\right| = \left|\widetilde{\mathbb{Q}}\right|. \tag{3.9}$$

Using Eq. (3.8), we can compose the columns in the generator table to obtain new columns, which gives the new commutator table $\eta(\widetilde{\mathbb{Q}}, \mathbb{H})$ with $\mathbb{H} = \langle\widetilde{\mathbb{H}}\rangle$. For example for $\left|\widetilde{\mathbb{H}}\right| = 3$, we have:

	I	$\widetilde{H}_1$	$\widetilde{H}_2$	$\widetilde{H}_3$	$\widetilde{H}_1\widetilde{H}_2$	$\widetilde{H}_1\widetilde{H}_3$	$\widetilde{H}_2\widetilde{H}_3$	$\widetilde{H}_1\widetilde{H}_2\widetilde{H}_3$
$\widetilde{Q}_1$	1	-1	1	1	-1	-1	1	-1
$\widetilde{Q}_2$	1	1	-1	1	-1	1	-1	-1
$\widetilde{Q}_3$	1	1	1	-1	1	-1	-1	-1

Using Eq. (2.6), we have:

$$\frac{1}{|\mathbb{Q}|}\sum_{Q\in\mathbb{Q}}\eta(Q,H) = \begin{cases} 1 & \text{if } \eta(\widetilde{\mathbb{Q}}, H) = \vec{1} \\ 0 & \text{otherwise.} \end{cases}$$

where $\mathbb{Q} = \langle\widetilde{Q}\rangle$ and $\vec{1}$ denotes a vector with all the entries being 1.

Looking at the commutator table $\eta(\widetilde{\mathbb{Q}}, \mathbb{H})$, we realise that $\eta(\widetilde{\mathbb{Q}}, H) = \vec{1}$ only when $H = I$. Hence, we have:

$$\sum_{Q\in\mathbb{Q}}\eta(Q,H) = 0 \quad \forall\ H \in \mathbb{H} \text{ and } H \neq I.$$

Composition of any two elements $H,\ H' \in \mathbb{H}$ will just give another element $H'' \in \mathbb{H}$, and we know that $H'' = HH' \neq I$ if $H \neq H'$. Hence, we have:

$$\sum_{Q\in\mathbb{Q}}\eta(Q,HH') = 0 \quad \forall\ H,\ H' \in \mathbb{H} \text{ and } H \neq H'. \tag{3.10}$$

Compared this to Eq. (3.6), we see that to fully twirl a noise whose Pauli basis is $\mathbb{V}$ using the twirling set $\mathbb{W}$, we can construct the twirling generators $\widetilde{\mathbb{W}}$ such that

$$\eta(\widetilde{\mathbb{W}}, \widetilde{\mathbb{V}}) = \eta(\widetilde{\mathbb{Q}}, \mathbb{H}_{\widetilde{\mathbb{V}}}) \tag{3.11}$$

where $\widetilde{\mathbb{V}}$ is the subset of independent elements in $\mathbb{V}$ and $\mathbb{H}_{\widetilde{\mathbb{V}}}$ is some subset of $\mathbb{H}$ (more details in Appendix A.2). The steps to construct the smallest $\widetilde{\mathbb{W}}$ that can satisfy Eq. (3.11) is outlined in Section 3.3.3.

3.3.2 Expected Size of the Twirling Generators

Assuming the twirling set $\mathbb{W}$ is a group, using Eq. (2.6), the requirement on the twirling set (Eq. (3.6)) becomes:

$$\eta(\widetilde{\mathbb{W}}, VV') \neq \vec{1} \quad \forall V,\ V' \in \mathbb{V} \text{ and } V \neq V'. \tag{3.12}$$

Now using the commutator vector composition rule in Eq. (3.8), Eq. (3.12) becomes:

$$\eta(\widetilde{\mathbb{W}}, V) \neq \eta(\widetilde{\mathbb{W}}, V') \quad \forall V,\ V' \in \mathbb{V} \text{ and } V \neq V'.$$

i.e. we want to find a set of twirling generators $\widetilde{\mathbb{W}}$ such that different Pauli basis elements V of the noise will have different commutation vectors with $\widetilde{\mathbb{W}}$. These different commutation vectors $\eta(\widetilde{\mathbb{W}}, V)$ of different V will just be different vectors of length $\left|\widetilde{\mathbb{W}}\right|$ with entries ± 1, which are just the columns that we obtained by composing the columns of the generator tables as shown in Section 3.3.1. Therefore by mapping $\widetilde{\mathbb{W}}$ to the smallest generator table possible, we are essentially finding the *smallest* $\widetilde{\mathbb{W}}$ possible. This also corresponds to the smallest $\mathbb{W}$ possible if we assume $\mathbb{W}$ is a group.

To achieve the commutator table matching in Eq. (3.11), we need to ensure the sizes of the sets match each other:

$$\begin{aligned} \left|\widetilde{\mathbb{W}}\right| &= \left|\widetilde{\mathbb{Q}}\right| = \left|\widetilde{\mathbb{H}}\right| \\ |\mathbb{V}| &= |\mathbb{H}_{\mathbb{V}}| \leq |\mathbb{H}| = 2^{|\widetilde{\mathbb{H}}|} \end{aligned} \tag{3.13}$$

where we have used Eq. (3.9). Combining these equations we have:

$$|\mathbb{V}| \leq 2^{|\widetilde{\mathbb{W}}|}. \tag{3.14}$$

Since we are looking for the smallest $\widetilde{\mathbb{W}}$ that satisfy Eq. (3.14) and $|\mathbb{V}| \leq 2^{|\widetilde{\mathbb{V}}|}$, we have $\left|\widetilde{\mathbb{W}}\right| \leq \left|\widetilde{\mathbb{V}}\right|$, which along with Eq. (3.14) gives us:

$$\log_2(|\mathbb{V}|) \leq \left|\widetilde{\mathbb{W}}\right| \leq \left|\widetilde{\mathbb{V}}\right|.$$

Hence, unlike the full Pauli operator set whose number of generators $\left|\widetilde{\mathbb{G}}\right| = 2n$ is dependent on the number of qubits n that we are considering, the size of our

twirling generators $\left|\widetilde{\mathbb{W}}\right|$ is only dependent on the sizes of the Pauli basis of the noise channel ($\mathbb{V}$) and the generating set of the Pauli basis ($\widetilde{\mathbb{V}}$). Noise arising from real physical process usually has symmetries present. Such symmetry constraints will reduce the size of the Pauli basis that builds our noise, which will enable us to find a much smaller twirling set than the full Pauli set. One such example will be discussed in Section 3.3.4, in which the lower bound is reached: $\left|\widetilde{\mathbb{W}}\right| = \log_2(|\mathbb{V}|)$.

3.3.3 Steps to Construct the Twirling Generators

1. Decompose the noise operator E to obtain its Pauli basis $\mathbb{V}$

$$\mathbb{V} = \{G \in \mathbb{G} \mid \mathrm{Tr}(GE) \neq 0\}.$$

   For a general noise channel, $\mathbb{V}$ will be the union of the Pauli basis of all the noise elements in the noise channel.

2. Find the following set:

   $\widetilde{\mathbb{V}}$: the generating set of $\mathbb{V}$.

   $\widetilde{\mathbb{V}}_S$: the subset of elements in $\widetilde{\mathbb{V}}$ that are used to generate elements in $\mathbb{V} - \widetilde{\mathbb{V}}$.

3. Find the smallest integer N that satisfies both

$$N \geq \log_2(|\mathbb{V}|)$$
$$N \geq \left|\widetilde{\mathbb{V}}_S\right|$$

   We now define a generating set $\widetilde{\mathbb{H}}$ of size N and denote the complete set that it generates as $\mathbb{H} = \left\langle \widetilde{\mathbb{H}} \right\rangle$.

4. Map elements in $\mathbb{V}$ to elements in $\mathbb{H}$ using the following steps:

   (a) Map $\widetilde{\mathbb{V}}_S$ to a subset of elements in $\widetilde{\mathbb{H}}$

   (b) Map the elements in $\mathbb{V} - \widetilde{\mathbb{V}}$ to elements in $\mathbb{H} - \widetilde{\mathbb{H}}$ by following the composition relations of the elements in $\widetilde{\mathbb{V}}_S$.

   (c) Map elements in $\widetilde{\mathbb{V}} - \widetilde{\mathbb{V}}_S$ to any subset of the remaining elements in $\mathbb{H}$ (which includes the identity).

Using the steps above, we can obtain the subset of $\mathbb{H}$ that $\widetilde{\mathbb{V}}$ maps to, which we will denoted as $\mathbb{H}_{\widetilde{\mathbb{V}}}$: $\widetilde{\mathbb{V}} \mapsto \mathbb{H}_{\widetilde{\mathbb{V}}}$.

5. Starting with the generator table $\eta(\widetilde{\mathbb{Q}}, \widetilde{\mathbb{H}})$, we compose its columns to get the commutator table $\eta(\widetilde{\mathbb{Q}}, \mathbb{H}_{\widetilde{\mathbb{V}}})$.

6. With the commutator table $\eta(\widetilde{\mathbb{Q}}, \mathbb{H}_{\widetilde{\mathbb{V}}})$ and the mapping $\widetilde{\mathbb{V}} \mapsto \mathbb{H}_{\widetilde{\mathbb{V}}}$ determined, we can then construct $\widetilde{\mathbb{W}} \in \mathbb{G}$ that maps to $\widetilde{\mathbb{Q}}$ such that $\eta(\widetilde{\mathbb{W}}, \widetilde{\mathbb{V}}) = \eta(\widetilde{\mathbb{Q}}, \mathbb{H}_{\widetilde{\mathbb{V}}})$ (Eq. (3.11)). Note that $\widetilde{\mathbb{W}}$ is not unique because the generating sets are not unique.

3.3.4 Application to a Physical Noise Model

Suppose we want to implement the Steane code as shown in Fig. 3.1 using spin qubits, but there is a small global field causing a global rotation of a small angle θ in the Z direction, leading to the following coherent noise:

$$E = \exp\Big\{-i\theta \sum_{i=1}^{7} Z_i\Big\} = I - i\theta \sum_{i=1}^{7} Z_i + O(\theta^2).$$

We will ignore the higher order terms $O(\theta^2)$ in the noise channel. The exact steps of obtaining the reduced twirling set are:

1. The Pauli basis of E is

$$\mathbb{V} = \{I, Z_1, Z_2, Z_3, Z_4, Z_5, Z_6, Z_7\}$$

2. Within $\mathbb{V}$, there are no composition relations other than those involving the identity. Hence, we have:

$$\widetilde{\mathbb{V}} = \{Z_1, Z_2, Z_3, Z_4, Z_5, Z_6, Z_7\}$$
$$\widetilde{\mathbb{V}}_S = \{Z_1\}$$

3. The smallest integer N that satisfies both

$$N \geq \log_2(|\mathbb{V}|) = 3$$
$$N \geq \left|\widetilde{\mathbb{V}}_S\right| = 1$$

is $N = 3$. Hence, we will define a generating set $\widetilde{\mathbb{H}}$ of size 3.

4. Using the fact that $\widetilde{\mathbb{V}}_S = \{Z_1\}$, the following mapping $\widetilde{\mathbb{V}} \mapsto \mathbb{H}_{\widetilde{\mathbb{V}}} \subseteq \mathbb{H} = \left\langle \widetilde{\mathbb{H}} \right\rangle$ can be found:

   $$\{Z_1, Z_2, Z_3, Z_4, Z_5, Z_6, Z_7\} \mapsto \{\widetilde{H}_1,\ \widetilde{H}_2,\ \widetilde{H}_3,\ \widetilde{H}_1\widetilde{H}_2,\ \widetilde{H}_1\widetilde{H}_3,\ \widetilde{H}_2\widetilde{H}_3,\ \widetilde{H}_1\widetilde{H}_2\widetilde{H}_3\}$$

5. Now starting with the generator table of $\left|\widetilde{\mathbb{H}}\right| = 3$, we can construct the commutator table $\eta(\widetilde{\mathbb{Q}}, \mathbb{H}_{\widetilde{\mathbb{V}}})$:

	$I(I)$	$\widetilde{H}_1(Z_1)$	$\widetilde{H}_2(Z_2)$	$\widetilde{H}_3(Z_3)$	$\widetilde{H}_1\widetilde{H}_2(Z_4)$	$\widetilde{H}_1\widetilde{H}_3(Z_5)$	$\widetilde{H}_2\widetilde{H}_3(Z_6)$	$\widetilde{H}_1\widetilde{H}_2\widetilde{H}_3(Z_7)$
$\widetilde{Q}_1$	1	-1	1	1	-1	-1	1	-1
$\widetilde{Q}_2$	1	1	-1	1	-1	1	-1	-1
$\widetilde{Q}_3$	1	1	1	-1	1	-1	-1	-1

   In the brackets are the elements in $\widetilde{\mathbb{V}}$ that the elements in $\mathbb{H}_{\widetilde{\mathbb{V}}}$ map to.

6. Our goal is just to find $\widetilde{\mathbb{W}}$ such that Eq. (3.11): $\eta(\widetilde{\mathbb{W}}, \widetilde{\mathbb{V}}) = \eta(\widetilde{\mathbb{Q}}, \mathbb{H}_{\widetilde{\mathbb{V}}})$ is satisfied.

   A possible choice is to have $\widetilde{\mathbb{W}} = \{X_1X_4X_5X_7,\ X_2X_4X_6X_7,\ X_3X_5X_6X_7\}$, which will produce the following commutator table $\eta(\widetilde{\mathbb{W}}, \widetilde{\mathbb{V}})$:

	I	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_7
$X_1X_4X_5X_7$	1	-1	1	1	-1	-1	1	-1
$X_2X_4X_6X_7$	1	1	-1	1	-1	1	-1	-1
$X_3X_5X_6X_7$	1	1	1	-1	1	-1	-1	-1

   which is the same as the commutator table $\eta(\widetilde{\mathbb{Q}}, \mathbb{H}_{\widetilde{\mathbb{V}}})$ in the last step.

The result of applying this reduced twirling generating set ($X_1X_4X_5X_7$, $X_2X_4X_6X_7$, $X_3X_5X_6X_7$) as opposed to the full twirling set to the Steane code under global Z rotation is shown in Fig. 3.2. As we can see from the plot, the fidelity curve of the reduced twirling set has some deviation from the full twirling curve due to the approximation we made initially in which we discarded the $O(\theta^2)$ term in the noise operator. When looking at $\theta = 0.1$, such deviation is $\sim 3.5 \times 10^{-4}$, which is an order of magnitude smaller than the $\sim 2.5 \times 10^{-3}$ deviation of the untwirled curve from the twirled curve. In practice, the fully-twirled fidelity is obtained by sampling over the fidelities obtained from different twirling gates and taking their

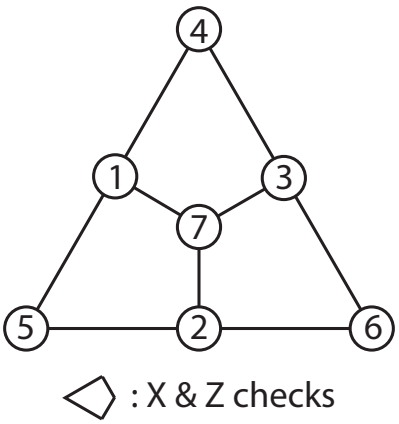


Figure 3.1: The Structure of Steane code.

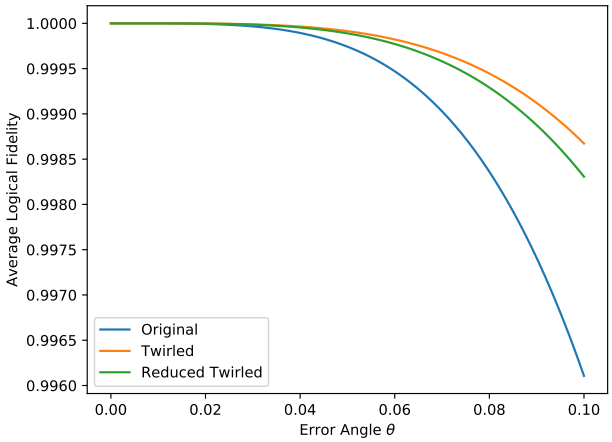


Figure 3.2: The logical fidelity of Steane code under coherent global Z noise.

average. Here we will assume the shot noise in *each* sample is represented by the deviation of the original fidelity from the fully-twirled fidelity (since the original fidelity corresponds to the identity twirling gate), which as mentioned above is an order of magnitude larger than the deviation of the reduced-twirled fidelity from the fully-twirled fidelity. When taking N samples, the shot noise of the average sampled fidelity will reduce by a factor of $\sqrt{N}$. When we have $\sqrt{N} \sim 10$, the shot noise will be reduced by an order of magnitude and match the performance of the reduced twirling by achieving similar fidelity deviation from the exact fully-twirled fidelity. This will require $N = 10^2 = 100$ random samples using the full twirling set, corresponding to 100 circuit runs, which is significantly more than the $2^3 = 8$ circuit runs needed when we iterate over the reduced twirling set.

More generally, for such noise due to the fluctuation of a global field, to the first order approximation we have $|\mathbb{V}| = n + 1$ where n is the number of qubits. Using our method of twirling set construction we can achieve $\left|\widetilde{\mathbb{W}}\right| = \log_2(n+1)$. Comparing to the twirling using the full set of Pauli operators: $\left|\widetilde{\mathbb{W}}\right| = 2n$, there is an exponential reduction of the size of the twirling set.

3.4 Pauli Twirling and Stabiliser Checks

3.4.1 One-gate Twirling

Let us consider the special case in which the twirling gate set is $\{I, \widetilde{W}\}$, i.e. there is only one extra gate other than the identity. We will call this a *one-gate twirling* operation and denote it as $\mathcal{T}_{\{I,\widetilde{W}\}}$.

Twirling with the Pauli subgroup $\mathbb{W}$ is equivalent to doing nested one-gate twirling using its generators $\widetilde{\mathbb{W}}$:

$$\mathcal{T}_{\mathbb{W}} = \prod_{\widetilde{W}\in\widetilde{\mathbb{W}}} \mathcal{T}_{\{I,\widetilde{W}\}}.$$

3.4.2 Equivalence of One-gate Twirling and Stabiliser Checks

For a given noise operator E and an one-gate twirling set $\mathbb{W} = \{I, W\}$, we can write

$$E = E_+ + E_-$$

where E_+ contains all the Pauli basis elements in E that commute with W, while E_- contains all the Pauli basis elements in E that anti-commute with W.

Then using Eq. (3.1), we have:

$$\begin{aligned}\mathcal{T}_{\{I,W\}}(\overline{E}) &= \frac{1}{2}\left[\overline{E} + \overline{WEW}\right] \\ &= \overline{E}_+ + \overline{E}_-\end{aligned} \tag{3.15}$$

i.e. an one-gate twirl $\mathbb{W} = \{I, W\}$ will decohere between the components in E that commute with W and the components that anti-commute with W.

We will use $\Pi_\pm$ to denote the projection operators for the ± 1 eigenspace of a W measurement. If we perform a W measurement and discard the measurement result, the effective channel we have is just:

$$\mathcal{S}_W = \overline{\Pi}_+ + \overline{\Pi}_-.$$

Now suppose we are working with a stabiliser code for which W is one of the stabilisers. Then if the noise E happens on a logical state $\overline{\rho}$ and we perform

$\mathcal{S}_W$ on it, we will have:

$$\begin{aligned}\mathcal{S}_W\,\overline{E}\,\overline{\rho} &= \left(\overline{\Pi}_+ + \overline{\Pi}_-\right)\overline{E}\,\overline{\rho} \\ &= \left(\overline{E_+\Pi_+ + E_-\Pi_-} + \overline{E_+\Pi_- + E_-\Pi_+}\right)\overline{\rho} \\ &= \left(\overline{E}_+ + \overline{E}_-\right)\overline{\rho}.\end{aligned} \tag{3.16}$$

where we have used $\overline{\Pi}_+\overline{\rho} = \overline{\rho}$ and $\overline{\Pi}_-\overline{\rho} = 0$.

We can follow similar analysis even if there is a Pauli error G on the logical state $\overline{\rho} \to \overline{G}\,\overline{\rho}$. The extra Pauli error may swap the ± 1 stabiliser check outcome, but will not change our error channel in Eq. (3.16), i.e. Eq. (3.16) will apply as long as $\overline{\rho}$ is an eigenstate of W, not necessarily needing to be the $+1$ eigenstate.

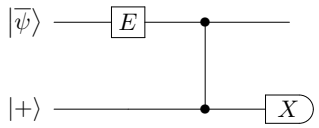
Comparing Eq. (3.16) to Eq. (3.15), we have:

$$\mathcal{T}_{\{I,W\}} \equiv \mathcal{S}_W$$

Hence, when we have an error E occurring on top of $\overline{G}\,\overline{\rho}$, twirling with $\mathbb{W} = \{I, W\}$ is equivalent to performing a W stabiliser check and throwing away the result.

3.4.3 Combining Stabiliser Checks with Pauli Twirling

We can use $\widetilde{W}$ stabiliser checks with discarded results as a substitute for the element $\widetilde{W}$ in the twirling generators to further reduce the size of the twirling generators. This is best shown through a simple example. Suppose we have the following circuit:



Here $\left|\overline{\psi}\right\rangle$ is stabilised by Z, i.e. $\left|\overline{\psi}\right\rangle = |0\rangle$ (note that here we have a trivial code space that encodes 0 logical qubits). In this circuit, we are effectively doing a Z stabiliser measurement on $\left|\overline{\psi}\right\rangle$, with a noise

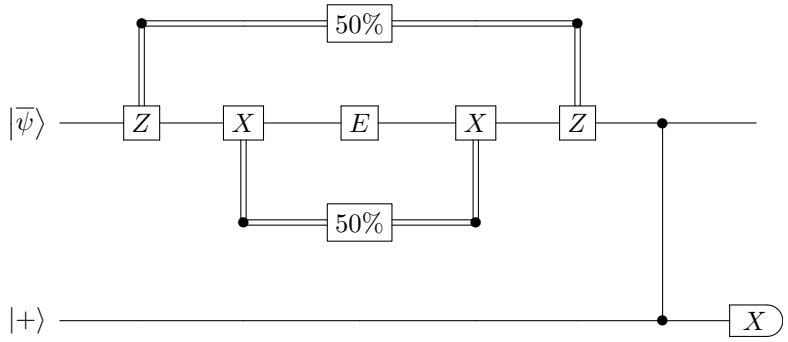
$$E \propto I + X + Y + Z$$

occurring in the circuit.

Now if we go through Section 3.3.3, we can obtain the twirling generating set $\widetilde{\mathbb{W}}$ needed to twirl E:

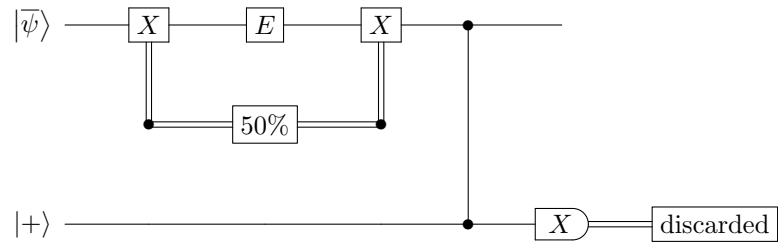
$$\widetilde{\mathbb{W}} = \{X, Z\}.$$

This can be implemented using the following nested one-gate twirling circuit (see Section 3.4.1):



where the pair of X will be applied with 50% probability, similarly and independently for the pair of Z.

However, if we discard the information specifying the result of the Z stabiliser check, then as argued before, the Z stabiliser check will have the same effect as the Z-twirling. Hence, we can turn E into Pauli errors with just the X-twirling:



This example shows how stabiliser measurements with thrown-away results can lead to a smaller set of twirling gates.

One possible application scenario in practice is when we try to perform fault-tolerant stabiliser measurements as discussed in Section 2.5. If we use a GHZ-state ancilla to extract the stabiliser measurement outcome, then what we care about is the parity of the measurement results of all the ancilla qubits. After we obtain the overall parity, we can discard the measurement results of the individual ancilla qubits, which as argued in this section will effectively twirl the noise on the corresponding data qubits that interact with the given ancilla qubit.

3.5 Discussion

In this chapter, we found the necessary and sufficient conditions for the set of twirling gates required to turn a given noise operator into a Pauli channel. We then demonstrated a way to construct the twirling set that satisfies the conditions. The size of the twirling set we obtained is lower-bounded by the size of the Pauli basis of the noise operator, and upper bounded by $2^{|\widetilde{\mathbb{V}}|}$ where $\widetilde{\mathbb{V}}$ is the set of independent Pauli basis elements of the noise operator. We showed that this is the smallest twirling set achievable assuming the twirling gate set forms a group. There can be an exponential reduction in the number of twirling gates in some cases. In the example of Steane code under global Z rotation noise, we show that in the limit of small rotation angles, using our method can approximately twirl the noise and improve its logical fidelity using only 8 twirling gates and thus 8 circuit runs. To achieve the same performance by sampling the full Pauli gate set for twirling we need around 100 circuit runs, around 10 times more than iterating over our reduced twirling set. In addition, we showed that in the case of stabiliser codes, we can replace elements in the generating set of the twirling set with existing stabiliser measurements with discarded results to further reduce the size of the twirling set.

For a general noise channel, the simple generalisation mentioned in Section 3.3.3 can indeed produce a twirling set smaller than the full set of Pauli operators. However, it is not the smallest possible set since we have not made use of the fact that different noise elements are inherently decohered. To obtain the optimal twirling set, we need to study the following property of twirling: if we know a twirling set that can twirl the noise operator E, and we know another twirling set that can twirl the noise operator F, then what is the twirling set that can twirl the noise channel $\overline{E} + \overline{F}$? Similarly, we can also ask what is the twirling set that can twirl the noise operator EF, which is essential in finding a single twirling operation that can twirl several consecutive erroneous components. We hope that this chapter will provide a framework for further explorations of properties of twirling like the two mentioned above.

4

Mitigating Coherent Noise: Pauli Conjugation

This chapter is adapted from the research published in npj Quantum Information [53], of which the present author is the main author. One of the co-authors Xiaosi Xu performs the numerical simulations in Section 4.4.3, while the other co-author Simon Benjamin contributes to the idea of logical twirling in Section 4.5.

Contents

4.1 Introduction

In the last chapter, we looked into using Pauli twirling to transform coherent errors into a Pauli channel, which as mentioned before can be much less damaging than coherent errors due to a slower rate of error accumulation. Twirling generally involves using a set of Pauli gates to sandwich the noise channel and averaging over the results. In this chapter, instead of using twirling to combat coherent errors, we propose to deterministically sandwich the noise channel using a chosen pair of Pauli gates, which we call Pauli conjugation.

We start by introducing some background concepts in Section 4.2. Then in Section 4.3, we find ways to reduce the search space for the optimal Pauli conjugation scheme. This is then used in Section 4.4 to compare the logical fidelity and concatenated threshold of Pauli conjugation to those of twirling for several quantum error correction codes under global Z rotation noise. In Section 4.5, we discuss the extension of our technique to multiple rounds of error correction and conjugation. This is followed by discussions in Section 4.6.

4.2 Pauli Conjugation

4.2.1 Logical Noise Channel

The process of stabiliser quantum error correction is described in Section 2.3.2, in which by performing the set of stabiliser checks $\widetilde{\mathbb{S}}$, we will obtain the error syndrome $\vec{m}$ and project the noisy state into the corresponding $\vec{m}$-syndrome subspace defined by the syndrome projection operators $\Pi_{\vec{m}}$. For each measured syndrome $\vec{m}$, we will apply the corresponding recovery operator $R_{\vec{m}}$, which is usually chosen to be the most likely Pauli error that leads to the given syndrome. The overall quantum error correction process can then be written as:

$$\mathcal{C} = \sum_{\vec{m}} \overline{R_{\vec{m}} \Pi_{\vec{m}}} = \sum_{\vec{m}} \overline{\Pi_0 R_{\vec{m}}}$$

where we have used $\Pi_{\vec{m}} = R_{\vec{m}} \Pi_0 R_{\vec{m}}$.

If we start within the logical subspace, the error correction process $\mathcal{C}$ will always project the state back to the logical subspace even after going through a noisy channel $\mathcal{N}$. Thus, the effective channel $\mathcal{N}_0 = \mathcal{C}\mathcal{N}$ will be a error channel that takes one logical state to another, i.e. it is a logical noise channel. The effective logical noise channel $\overline{\mathcal{N}}_0$ is defined to be the average of $\mathcal{N}_0$ over all logically equivalent starting and final states. Using Pauli transfer matrix (Section 2.1.3), $\overline{\mathcal{N}}_0$ can be written as:

$$\begin{aligned}
\overline{\mathcal{N}}_0 &= \sum_{G,G' \in \mathbb{G}} |\overline{G}'\Pi_0\rangle\rangle\langle\langle \overline{G}'\Pi_0|\mathcal{C}\mathcal{N}|\overline{G}\Pi_0\rangle\rangle\langle\langle \overline{G}\Pi_0| \\
&= \sum_{\vec{m}} \sum_{G,G' \in \mathbb{G}} |\overline{G}'\Pi_0\rangle\rangle\langle\langle \overline{G}'\Pi_0|\,\overline{\Pi_0 R_{\vec{m}}}\,\mathcal{N}|\overline{G}\Pi_0\rangle\rangle\langle\langle \overline{G}\Pi_0| \\
&= \sum_{\vec{m}} \sum_{G,G' \in \mathbb{G}} |\overline{G}'\Pi_0\rangle\rangle\langle\langle \overline{G}'\Pi_0|\overline{R}_{\vec{m}}\mathcal{N}|\overline{G}\Pi_0\rangle\rangle\langle\langle \overline{G}\Pi_0| \\
&= \mathcal{R}\mathcal{N}
\end{aligned} \tag{4.1}$$

where $\mathcal{R} = \sum_{\vec{m}} \overline{R}_{\vec{m}}$ [1].

4.2.2 Twirling and Conjugation

As discussed in Chapter 3, twirling is a technique for converting an arbitrary error channel into a Pauli channel, which is carried out by taking the average of the error channel conjugated with different gates chosen from a set of Pauli gates $\mathbb{W} \subseteq \mathbb{G}$ that we call the twirling set. Conventionally, twirling is carried out using the full set of Pauli gates as the twirling set: $\mathbb{W} = \mathbb{G}$. However, as shown in Chapter 3, it is possible to find a smaller $\mathbb{W}$ that is equivalent to the full Pauli set.

Twirling a noise channel $\mathcal{N}$ is just

$$\mathcal{T}(\mathcal{N}) = \frac{1}{|\mathbb{W}|} \sum_{W \in \mathbb{W}} \overline{W}\,\mathcal{N}\,\overline{W}\,. \tag{4.2}$$

Twirling can decohere the Pauli components in the noise channel and turn it into a Pauli channel. This will correspond to removing the off-diagonal elements of the Pauli transfer matrix of the channel.

[1] Note that here we have abused the notation of $\mathcal{R}\mathcal{N}$ assuming it will only act on the logical Pauli basis $\{|\overline{G}\Pi_0\rangle\rangle\}$ instead on all of the physical Pauli basis.

Using Eq. (4.1) and Eq. (4.2), the effective logical channel after twirling is:

$$\overline{\mathcal{N}}_T = \mathcal{R}\mathcal{T}(\mathcal{N}) = \frac{1}{|\mathbb{W}|}\sum_{W\in\mathbb{W}} \mathcal{R}\,\overline{W}\,\mathcal{N}\,\overline{W}\,.$$

Instead of averaging over all twirling gates, if we deterministically conjugate the noise process with a given twirling gate W, the effective logical channel can be written as

$$\overline{\mathcal{N}}(W) = \mathcal{R}\,\overline{W}\,\mathcal{N}\,\overline{W}$$

which we will call Pauli conjugation.

Then we have:

$$\begin{aligned}\overline{\mathcal{N}}_0 &= \overline{\mathcal{N}}(I)\\ \overline{\mathcal{N}}_T &= \frac{1}{|\mathbb{W}|}\sum_{W\in\mathbb{W}} \overline{\mathcal{N}}(W).\end{aligned}$$

The logical fidelity of $\overline{\mathcal{N}}(W)$ is

$$F(W) = \int \langle\langle\overline{\rho}|\overline{\mathcal{N}}(W)|\overline{\rho}\rangle\rangle\, \mathrm{d}\overline{\rho}$$

where $\overline{\rho}$ is a logical state and the integral is over the *pure state* surface using the Haar measure.

Since the fidelity F is a linear function of the noise process $\overline{\mathcal{N}}$, we can similarly obtain the original logical fidelity F_0 and the twirled logical fidelity F_T:

$$\begin{aligned}F_0 &= F(I)\\ F_T &= \frac{1}{|\mathbb{W}|}\sum_{W\in\mathbb{W}} F(W).\end{aligned}$$

There exists a $W_{max} \in \mathbb{W}$ such that $F(W_{max})$ is the maximum $F(W)$ that we can achieve. By definition we have

$$\begin{aligned}F(W_{max}) &\geq F(I)\\ F(W_{max}) &\geq F_T.\end{aligned}$$

Thus if we can find such W_{max} and deterministically apply it to the noise instead of doing nothing or randomly applying all $W \in \mathbb{W}$, we can obtain a higher fidelity $F(W_{max})$ than the original fidelity $F(I)$ and the twirled fidelity F_T.

4.2.3 Mechanism of Conjugation

Let us first consider the case when we perform quantum error correction on a unitary (completely coherent) noise channel and obtain the 0-syndrome ($m_i = 0 \quad \forall i$). The resultant effective noise channel will contain an error-free components represented by the coherent superposition of the stabiliser operators $\sum_i \alpha_i S_i$. When acting on a logical state, the effective amplitude corresponding to the logical identity will then be $\sum_i \alpha_i$.

Now if we apply Pauli conjugation using the operator W to the error channel, the error-free components will become $\sum_i \alpha_i W S_i W$, which corresponds to an amplitude of $\sum_i \eta(W, S_i)\alpha_i$ for the logical identity. Recall that $\eta(A, B)$ is the commutator between operators A and B.

Thus Pauli conjugation will change the sign of the Pauli components of the error channel, changing the way the Pauli components interfere. For the 0-syndrome case, if we can choose a conjugation operator W such that $\sum_i \eta(W, S_i)\alpha_i \geq \sum_i \alpha_i$, i.e. the error-free components (the stabilisers) interfere more constructively with conjugation than without, it will lead to an increase in the logical fidelity of the channel using conjugation. The normalisation of the channel also means that the logical error components of the channel will interfere more destructively when using conjugation. Similar arguments can be made for the non-zero-syndrome cases.

Hence for a given noise channel, as long as there is some coherent superposition of its Pauli components corresponding to the same logical operators for a given syndrome, Pauli conjugation should be able to improve its logical fidelity by changing the relative signs between the components and alter the way they interfere. One case for which Pauli conjugation will not be able to help is when the identity is the optimal conjugation gate $W_{max} = I$, i.e. the noise Pauli components are interfering in the optimal ways for the given code, which should be unlikely unless we have hand-picked our code to exactly fit the noise process.

4.3 Finding the Optimal Conjugation Gate

The usual Pauli twirling will have $\mathbb{W} = \mathbb{G}$. For n qubits, this means that there are 4^n elements in $\mathbb{W}$ that we need to search over to find W_{max}, which is exponentially difficult for large n. Hence, we first need to reduce the size of $\mathbb{W}$ in order to find W_{max} effectively.

Rather than dealing with the twirling set $\mathbb{W}$, we will first be working with its generator $\widetilde{\mathbb{W}}$. The reason we can work with the generators for our later purposes is outlined in Appendix B.3.

The generators of the conventional twirling set is just $\widetilde{\mathbb{W}} = \widetilde{\mathbb{G}}$. For a given quantum error correction code, the Pauli generators $\widetilde{\mathbb{G}}$ can be divided into the three partitions as discussed in Section 2.3: stabiliser generators $\widetilde{\mathbb{S}}$, logical generators $\widetilde{\overline{\mathbb{G}}}$ and error generators $\widetilde{\mathbb{E}}$. Hence, we have

$$\widetilde{\mathbb{W}} = \widetilde{\mathbb{G}} = \widetilde{\mathbb{S}} + \widetilde{\mathbb{E}} + \widetilde{\overline{\mathbb{G}}}.$$

We can of course start with the reduced twirling generators instead using the methods in Chapter 3 and still divide them into the three partitions like above. All the other arguments in the following sections will be similar. However, for simplicity, in this chapter we will assume we start with the full Pauli group as our twirling set.

4.3.1 The Structure of the Code

$\mathcal{R}$ and $\overline{S}$ commute because they are both Pauli channels which are diagonal in the form of Pauli transfer matrix. Hence, for any channel $\mathcal{N}$, and logical states $|\overline{\rho}\rangle\rangle$ and $|\overline{\rho}'\rangle\rangle$, we have:

$$\langle\langle\overline{\rho}'|\mathcal{R}\,\overline{S}\,\mathcal{N}\,\overline{S}|\overline{\rho}\rangle\rangle = \langle\langle\overline{\rho}'|\,\overline{S}\,\mathcal{R}\mathcal{N}\,\overline{S}|\overline{\rho}\rangle\rangle = \langle\langle\overline{\rho}'|\mathcal{R}\mathcal{N}|\overline{\rho}\rangle\rangle$$

which means that conjugation using stabilisers on any noise channel has a trivial effect on the effective logical channels [2]. Hence, we can remove all stabilisers from

[2]Note that this is different from our results in Section 3.4, in which we have proven that stabiliser measurements with discarded result are equivalent to stabiliser twirling. Here we do not need to discard the measurement results and instead of showing that stabiliser measurements and stabiliser twirling are equivalent, we only showed that their effects on the *logical* channel are equivalent (they both have trivial effects).

the twirling generator set and reduce it to:

$$\widetilde{\mathbb{W}} = \widetilde{\mathbb{E}} + \widetilde{\overline{\mathbb{G}}}.$$

Now if we are calculating the logical fidelity, we are integrating over all the logical pure states using the unitary Haar measure, which is by definition invariant under any unitary transformation. Thus we have:

$$F_0 = \int \langle\langle \overline{\rho} | \mathcal{R}\mathcal{N} | \overline{\rho} \rangle\rangle \, \mathrm{d}\overline{\rho} = \int \langle\langle \overline{\rho} | \, \overline{\overline{G}} \, \mathcal{R}\mathcal{N} \, \overline{\overline{G}} | \overline{\rho} \rangle\rangle \, \mathrm{d}\overline{\rho}$$

$\mathcal{R}$ and $\overline{\overline{G}}$ again commute since they are both Pauli channel. Hence we have:

$$F_0 = \int \langle\langle \overline{\rho} | \mathcal{R} \, \overline{\overline{G}} \, \mathcal{N} \, \overline{\overline{G}} | \overline{\rho} \rangle\rangle \, \mathrm{d}\overline{\rho} = F(\overline{G}) \quad \forall G \in \mathbb{G}$$

Hence, when calculating the logical fidelity, conjugation with logical Pauli operators also acts trivially and can be removed from the twirling generating group. The remaining non-trivial twirling generators are:

$$\widetilde{\mathbb{W}} = \widetilde{\mathbb{E}}$$

The way to construct $\widetilde{\mathbb{E}}$ such that it consists of only single-qubit X and Z gates is outlined in Appendix B.1 [3].

4.3.2 The Structure of the Noise

Two super-operators $\overline{A}$ and $\overline{B}$ will commute if their commutator $\eta(A, B) = e^{i\phi}$, i.e. their commutator is some phase factor.

We will write our noise channel $\mathcal{N}$ in terms of its noise elements $\overline{N}$:

$$\mathcal{N} = \sum_N \overline{N} \, .$$

Now if a twirling generator W satisfies $\eta(W, N) = e^{i\phi} \; \forall N$, then

$$\overline{W} \, \overline{N} \, \overline{W} = \overline{N} \quad \forall N$$

$$\overline{W} \, \mathcal{N} \, \overline{W} = \mathcal{N}$$

[3] Note that if we start with the reduced twirling set instead of the full Pauli group, then not all single-qubit operators will be in the twirling set and we might not be able to choose a $\widetilde{\mathbb{E}}$ that consists of only single-qubit gates.

i.e. it act trivially on noise $\mathcal{N}$ and hence can be removed.

After such reduction, the twirling generating set now becomes:

$$\widetilde{\mathbb{W}} = \{W \in \widetilde{\mathbb{E}} \mid \exists N \ \eta(W, N) \neq e^{i\phi}, \phi \in \mathbb{R}\}$$

4.3.3 Symmetry in Codes and Noise

The twirling set $\mathbb{W}$ can be generated from $\widetilde{\mathbb{W}}$ following Appendix B.2. Based on the symmetry existing in both the code and the noise, we can prove the equivalence between different elements in $\mathbb{W}$.

Suppose we manage to find a Clifford operation U such that the code state basis $\Pi_{\vec{0}}\overline{G}$ and the physical noise channel $\mathcal{N}$ are invariant under its transformation:

$$\begin{aligned} \left[U, \Pi_{\vec{0}}\overline{G}\right] &= 0 \quad \forall G \in \mathbb{G} \\ \left[\overline{U}, \mathcal{N}\right] &= 0 \end{aligned} \tag{4.3}$$

we can prove that (see Appendix B.4)

$$\overline{\mathcal{N}}(W) = \overline{\mathcal{N}}(U^\dagger W U) \tag{4.4}$$

i.e. the effective logical channel conjugated with W is the same as that with $U^\dagger W U$. All of such U will form a group $\mathbb{U}$.

Hence, we can define an equivalence relation:

$$W' \sim W \iff \exists U \in \mathbb{U} \quad W' = U^\dagger W U$$

In such a way, conjugacy with elements in $\mathbb{U}$ will split $\mathbb{W}$ into several equivalence classes. The elements in the same equivalence class will produce the same logical fidelity when used to conjugate the noise.

The simplest type of Clifford transformation to consider is qubit permutation, for which U consists of swap gates. Permutation symmetry of quantum error correction codes has been studied in Ref. [18] and [22]. Note that qubit permutation will preserve the weights of the operators, thus it is crucial to construct $\mathbb{W}$ to have the elements with the lowest weight possible (see Appendix B.2), so that more of them can be proven to be in the same equivalence class.

If a code has one logical qubit and its logical Pauli gates consist of applying physical Pauli gates to all the qubits, then such transversal logical Pauli gates are invariant [4] under any qubit permutation U, i.e. $\left[U, \overline{G}\right] = 0 \quad \forall G \in \mathbb{G}$. For such codes, we only need to further make sure that the set of stabilisers are invariant under the given qubit permutation U:

$$[U, \Pi_0] = 0 \tag{4.5}$$

to ensure the code symmetry requirement in Eq. (4.3) is satisfied. Furthermore, if some of the stabilisers commute with the noise, then these stabilisers will have trivial effect in the error correction process and thus can be safely ignored. In such a case, we will only need to consider the symmetry of the stabilisers that do not commute with the noise. For example, for a pure Z noise, we can safely ignore the Z stabilisers when we are considering code symmetry.

4.4 Mitigating Coherent Z Noise using Pauli Conjugation

In this section we will try to find the optimal Pauli conjugation gate for different quantum error correction codes under the global Z rotation noise:

$$N(\theta) = \prod_{j=1}^{J} e^{-i\theta Z_j} \tag{4.6}$$

where J is the number of qubit, which is also the noise model we looked at in Section 3.3.4. This noise is a coherent superposition of all possible Z operators (tensor products of I and Z). The weight-n Z operators in the superposition will have the amplitude $(-i\sin\theta)^n (\cos\theta)^{J-n}$.

Of course if we are allowed to flip all the qubits right in the middle of the channel, we can flip the direction of the rotation and cancel the coherent error, which is just a simple example of dynamical decoupling. However, if we look at for example high frequency global Z noise, in which the direction of the global Z rotation may flip

[4] More precisely, we can find at least one physical representation of logical $\overline{G}$ out of all of its logically equivalent counter-parts that satisfy this symmetry condition.

after a very short time interval in a random walk fashion, dynamical decoupling cannot be applied. In such a case, we have discussed how Pauli conjugation can be used to mitigate such noise in Appendix B.11. It builds from our discussion in this section, in which we will be looking at the coherent global Z rotation noise described in Eq. (4.6) without allowing gates to be performed in the middle of the channel.

Since the noise only consists of Z components, all pure Z twirling generators will act trivially on the noise, thus can be removed. This noise is symmetric under any qubit permutation. Hence, any permutation symmetry of the quantum error correction code will also exist for the noise.

For all the codes that we will discuss in this section, their logical Pauli gates consist of applying physical Pauli gates to all the qubits. Thus the code symmetry condition in Section 4.3.3 can be reduced into Eq. (4.5). Along with the fact that we have pure Z noise, we only need to focus on the symmetry of the X stabilisers in this section when we talk about the symmetry of a code, except for the five-qubit code. Transversal global logical Pauli gates also mean that $N(\frac{\pi}{2})$ will be the Z logical operator. Thus the logical fidelity curve against different θ will have a rotational symmetry about $\theta = \frac{\pi}{4}$ (see Appendix B.5), which means that we only need to look at $0 \leq \theta \leq \frac{\pi}{4}$ to see the effect of the noise on logical fidelity.

4.4.1 Steane Code

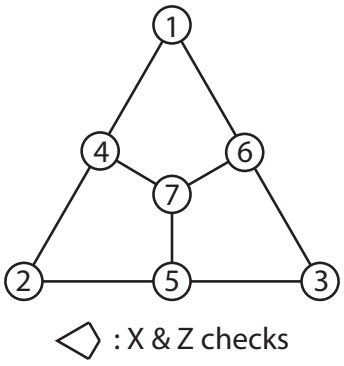


Figure 4.1: The Steane code.

In the Steane Code, we have

- Stabiliser generators $\widetilde{\mathbb{S}}$: X or Z checks on plaquettes $(1,4,6,7)$, $(2,4,5,7)$ and $(3,5,6,7)$.
- Logical generators $\widetilde{\overline{\mathbb{G}}}$: X or Z on all qubits.

Following Section 4.3.1, we can construct our twirling generators to be

$$\widetilde{\mathbb{W}} = \widetilde{\mathbb{E}} = \{X_1, X_2, X_3, Z_1, Z_2, Z_3\}.$$

Since the noise only consists of Z components, all pure Z twirling generators will act trivially on the noise, thus can be removed, we then have:

$$\widetilde{\mathbb{W}} = \{X_1, X_2, X_3\}$$

which generates the twirling set:

$$\mathbb{W} = \{I, X_1, X_2, X_3, X_4, X_5, X_6, X_7\}.$$

Note that here we have transformed the error operators to their lowest weight equivalence that produce the same error syndromes.

The Steane code has the same symmetry as the Fano plane [18], whose permutation symmetry group will be denoted as $\mathbb{U}$. Since our noise model is symmetric under any qubit permutation, all $U \in \mathbb{U}$ satisfied Eq. (4.3).

Now for every pair of single-qubit X operators $X_i, X_j \in \mathbb{W}$, we can find at least one $U \in \mathbb{U}$ such that

$$U^\dagger X_i U = X_j.$$

Hence, using Eq. (4.4) we know that all the remaining single-qubit X twirling operators are equivalent.

There are two equivalence class of twirling gates here, one is equivalent to I, while the other is equivalent to X_1 (or any single-qubit X gate).

The effect of different strategies on the logical fidelity of Steane code is shown in Fig. 4.2. We can see that twirling is consistently better than doing nothing, while X_1 conjugation will yield even higher fidelity than twirling.

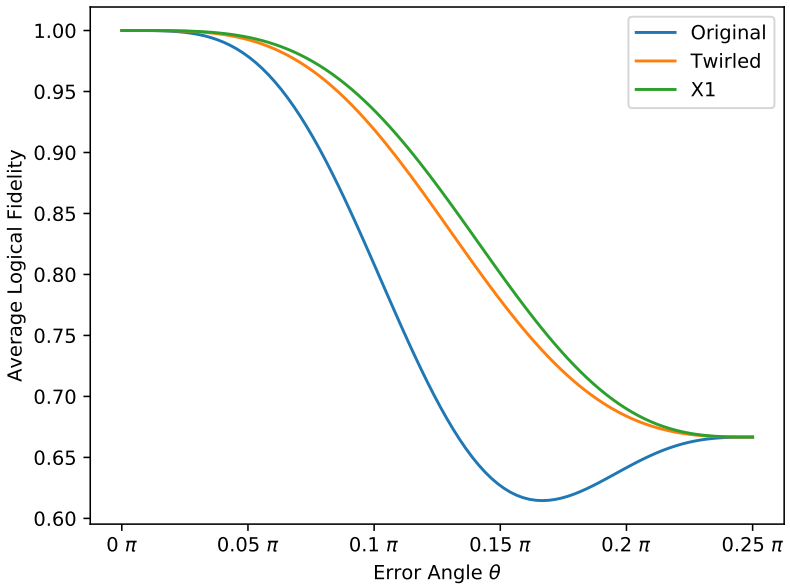


Figure 4.2: Logical fidelity of the Steane code under different noise strength and noise tailoring schemes.

4.4.2 Other Codes

In this section, we will explore the effect of Pauli conjugation using other codes under the same noise model. The details of finding the equivalent class of conjugating gates for different codes are outlined in Appendix B.6. Here we will just look at the effect of using conjugating gates in different equivalence classes and compare their effects to doing nothing and twirling.

Five-qubit code

The structure of five-qubit code is shown in Fig. 4.3. There is just one non-trivial conjugating strategy in five-qubit code, which is conjugation with any single-qubit X gate, the same we found in the Steane code. However, in our noise model, we found that this strategy makes no difference to the logical fidelity compared to doing nothing. Consequently, the twirled logical fidelity is also the same. Hence, rather interestingly under our noise model, none of the strategies works for the five-qubit code.

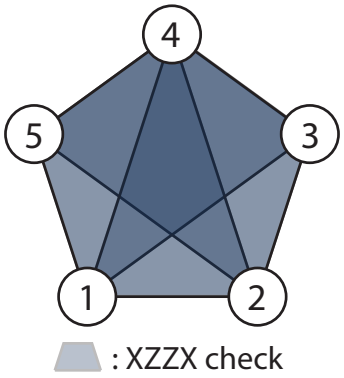


Figure 4.3: The five-qubit code.

This is mainly due to the fact that the five-qubit code is not a Calderbank-Shor-Steane (CSS) code. In a CSS code, only the X stabiliser generators can be used to detect and decohere Z noise. While in the case of the five-qubit code, all of its stabiliser generators can be used to detect and decohere Z noise. For the five-qubit code under the global Z rotation noise, there are 2^5 different Z noise components, which are decohered by the measurement of the 4 stabiliser generators and thus split into 2^4 groups for different syndrome outcomes. Hence, for a given syndrome outcome, the resultant noise will be the superposition of two Z noise components ($2^5/2^4 = 2$). One corresponds to the logical I while the other corresponds to the logical Z. Since there is just one component here corresponding to the logical I, there is no way to boost the logical I amplitude and increase the logical fidelity via Pauli conjugation.

Nine-qubit Shor code

The structure of the nine-qubit Shor code is shown in Fig. 4.4. There are three types of non-trivial Pauli conjugations in the nine-qubit Shor code for our noise model:

- Single qubit flip: X_1
- Two-qubit flip (in different rows): X_1X_4
- Three-qubit flip (in different rows): $X_1X_4X_7$

Figure 4.4: The nine-qubit Shor code.

The effects of these strategies on the logical fidelity are shown in Fig. 4.5. We see that doing nothing will result in a dip at $\theta = \frac{\pi}{6}$, where our noise turns into a logical operator. Twirling can definitely mitigate such a problem, leading to a great jump in fidelity. Superior improvements can be achieved by conjugating the noise with $X_1X_4X_7$.

The result for the other nine-qubit Shor code with the X and Z checks exchanged is shown in Appendix B.8.

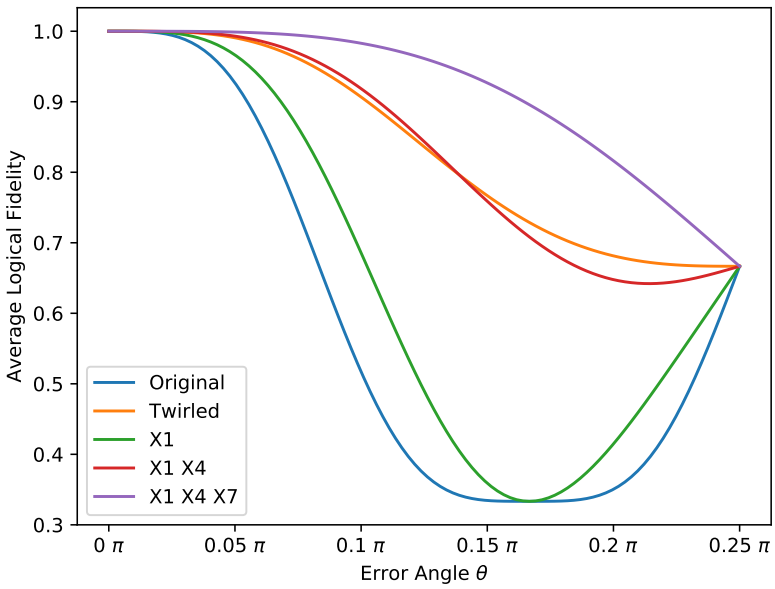


Figure 4.5: Logical fidelity of the nine-qubit shor code under different noise strength and noise tailoring schemes.

Distance-3 surface code

The structure of the distance-3 surface code is shown in Fig. 4.6. The non-trivial

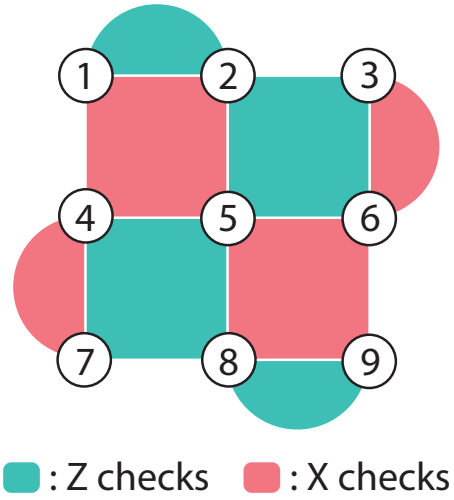


Figure 4.6: The surface code of distance 3.

conjugating strategies and their effects on the logical fidelity are shown in Fig. 4.7. Again we see improvement of the twirled fidelity over doing nothing, and a marked improvement of conjugating the noise with X_1X_8 over twirling.

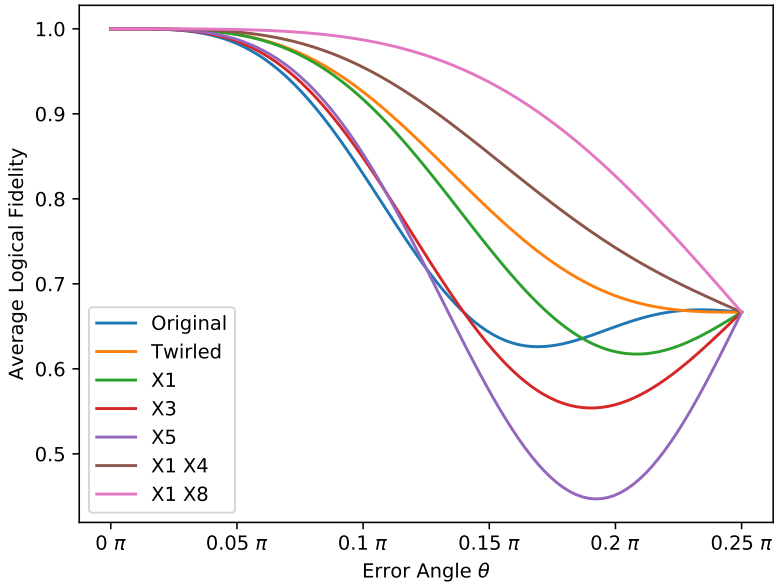


Figure 4.7: Logical fidelity of the distance-3 surface code under different noise strength and noise tailoring schemes.

4.4.3 Gate Error

In reality, applying extra Pauli gates does not come free due to the errors associated with the gates. We should expect the effect of such errors due to Pauli conjugation to be small since the quantum error correction circuits generally involve more gates than Pauli conjugation and also contain two-qubit gates which usually have much lower fidelity than single-qubit Pauli gates. Here we have simulated the performance of different schemes using different codes with depolarising gate error rates of 0.5% and 1% for the encoding circuit, the quantum error correction circuit and the Pauli conjugation gates (with the details of the circuits shown in Appendix B.9). From the result in Fig. 4.8 we can see that as we increase the gate error rate, the fidelity curves shift downward without much change to their shapes. Hence, the optimal Pauli conjugation schemes maintain their advantages over doing nothing when we take into account gate errors. The fidelity curves using twirling are not shown. However in our examples, we should expect the advantage of Pauli conjugation over twirling increases with increasing gate error rates since the average weights of the twirling gates are higher than that of the conjugation gates.

When trying to implement Pauli conjugation in practice, such gate errors can be mitigated by absorbing the conjugation gates into the existing gates in the circuit. Such a strategy has been proven to be effective in the case of twirling [54].

4.4.4 Concatenated Threshold

As discussed by Rahn *et al.* [55], after finding the map between the physical noise channel and the logical noise channel with one level of encoding, composing this map will give us the physical-logical noise map for the concatenated code. Here we have assumed that we are using a hard decoder which only takes into account of syndrome information of the current concatenation level. Finding such maps will allow us to compute the performance of a code with different levels of concatenation and hence find its concatenated threshold. Such analysis was carried out by Huang *et al.* [18] for a variety of codes. Here we will use the local Z noise map obtained in Ref. [18] to calculate the concatenated threshold for different codes when we

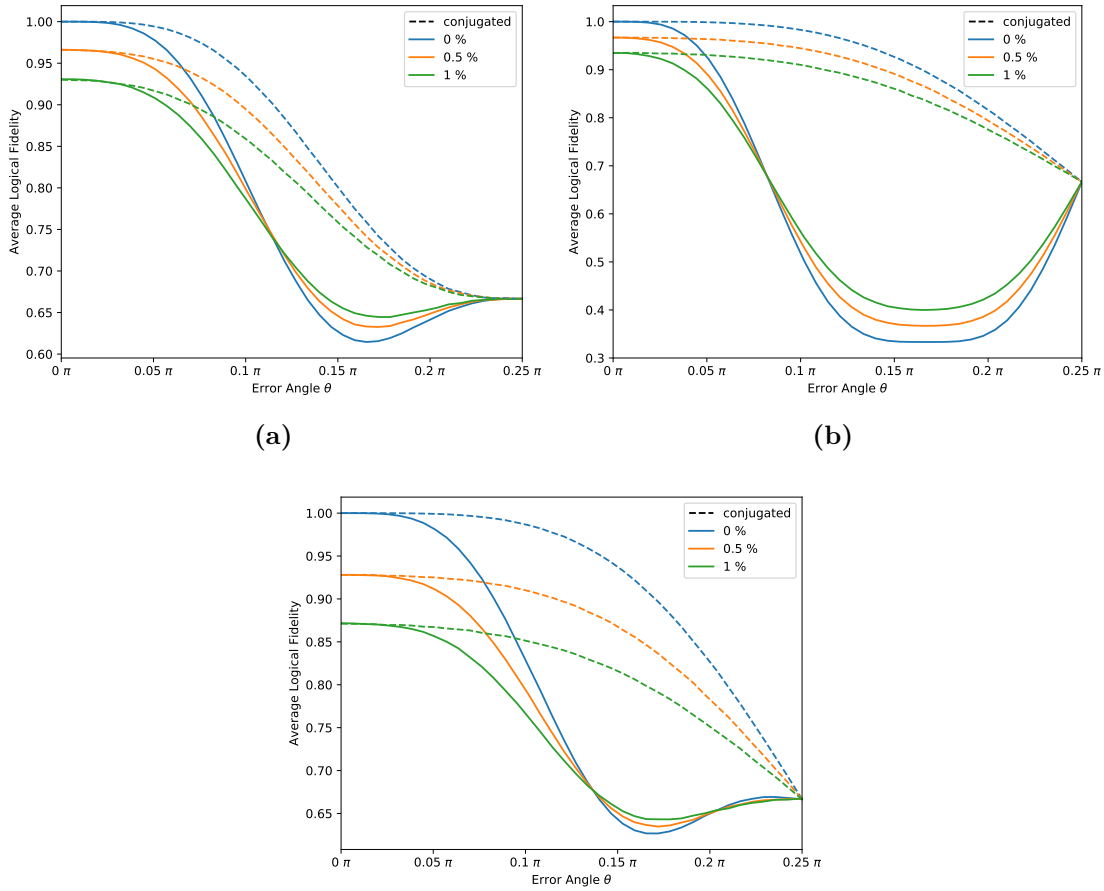


Figure 4.8: Logical fidelity with global Z rotation environmental noise of magnitude θ and depolarising gate noise of probability 0%, 0.5% and 1% with or without Pauli conjugation for (a) Steane code, (b) 9-qubit Shor code and (c) distance-3 surface code. The Pauli conjugations we used here are the optimal schemes that we found with zero gate error.

apply different kinds of noise tailoring schemes at the physical level (not at any subsequent levels of concatenation). From the results in Fig. 4.9, we can see the logical fidelity of the threshold crossing points of different noise tailoring schemes are essentially the same. Hence when we try to achieve the threshold logical fidelity with one level of encoding, if one scheme has a higher tolerance of the physical error than another scheme, we should expect a similar improvement in the concatenated threshold. The improvement of the conjugated threshold over the original threshold is 40%, 160% and 110% for the Steane code, 9-qubit Shor code and distance-3 surface code respectively. All of them also show improvements of the conjugated

thresholds over the twirled thresholds.

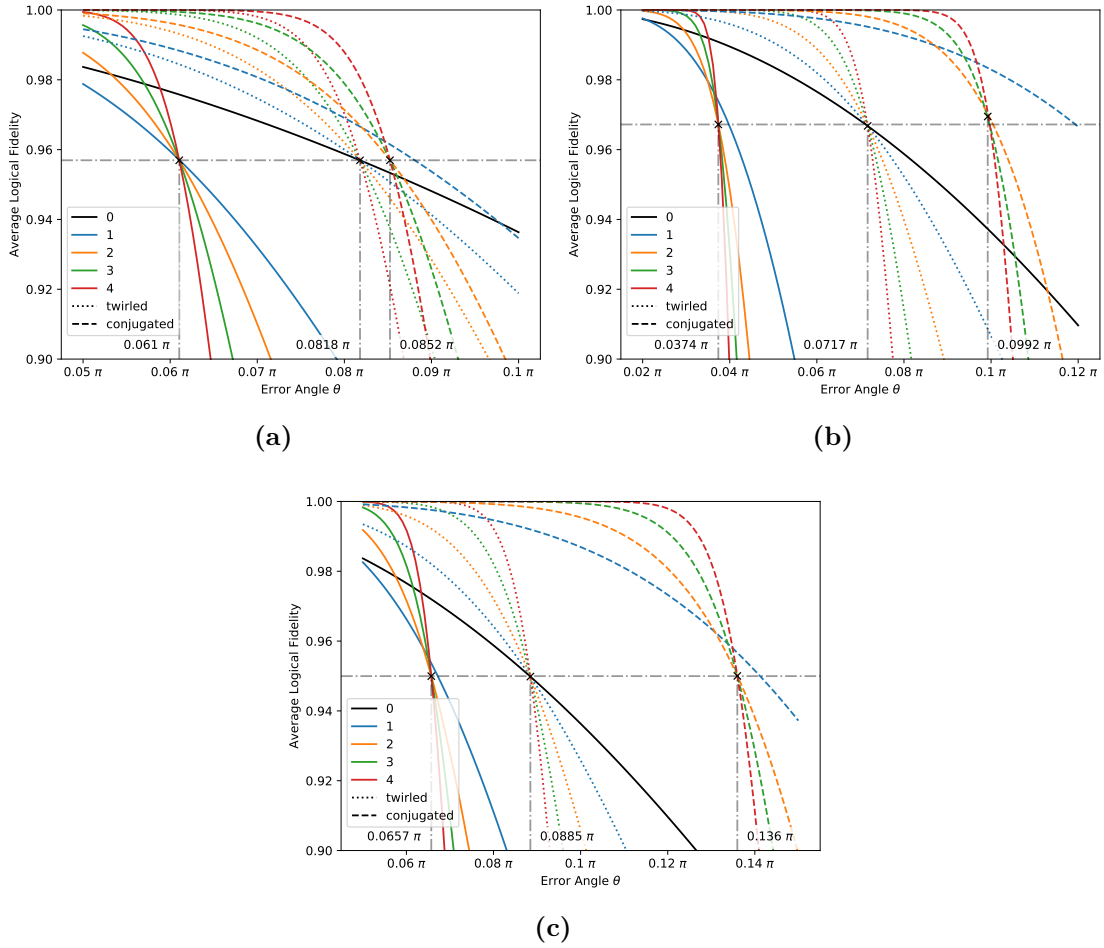


Figure 4.9: The concatenated threshold plot under global Z rotation noise for (a) Steane code, (b) 9-qubit Shor code and (c) distance-3 surface code. Different colours shows different levels of concatenation while different line styles shows applying different strategies like twirling or Pauli conjugation to the noise. The Pauli conjugations we used here are the optimal schemes that we found with zero gate error.

4.5 Composing the Error Channels

4.5.1 Multiple Rounds of Quantum Error Correction

As mentioned in Section 2.2, coherent errors can be more damaging than incoherent errors because they can accumulate at a faster rate. Hence, we use Pauli twirling to remove the coherent nature of the error channel for each round of error correction so that errors in multiple rounds of error correction will accumulate at a more favourable

scaling. As demonstrated in the previous sections, Pauli conjugation can improve the logical fidelity for coherent errors in one round of error correction. However, the error remains coherent after conjugation. Though there is evidence showing a logical channel will become less coherent as the distance of the code increases [18, 39], the remaining coherence in the logical channel still could mean that the advantages of conjugation can be lost when we go to multiple rounds of error correction.

Fortunately this can be overcome by injecting 'just enough' randomness – the solution might be called 'logical twirling' of the error channel (instead of twirling at the physical level). Logical twirling simply means twirling over the logical Pauli operators and decohere the Pauli components that corresponding to different logical operators. The resultant effective channel will be logically incoherent and thus the errors will accumulate at a more favourable rate in terms of logical fidelity. For one round of quantum error correction, applying logical twirling will not change the logical fidelity just like twirling a noise channel will not change its fidelity [5]. Hence, applying logical fidelity on top of conjugation can maintain the fidelity improvement brought by conjugation in each round of error correction while preventing the logical errors from rapid accumulation as we go to multiple rounds.

The Pauli components of a given noise channel can be partitioned into sets that correspond to different logical operators with different measured syndromes after quantum error correction. As discussed in Section 4.2.3, the coherence between the components that correspond to the same logical operator and the same syndrome can be used by conjugation to improve the logical fidelity of the channel in one round of quantum error correction (via destructive interference between the logical error components), while in this section we see that the coherence between different logical operators can be removed by logical twirling to fight the accumulation of logical errors in multiple rounds of quantum error correction.

In Appendix B.10, using global Z rotation as an example, we demonstrate that using conjugation alone, the advantages of conjugation over physical twirling will

[5] Applying twirling to a noise channel at the physical level will not change its *physical* fidelity, but may change the *logical* fidelity

diminish as we go to more rounds of quantum error correction and we also show how this is overcome by using logical twirling.

4.5.2 Multiple Rounds of Noise Tailoring

Instead of applying both noise tailoring and error correction at each time step, we can apply just noise tailoring in each time step and only do one round of error correction at the very end.

The matrix elements for the effective noise channel with K rounds of twirling are:

$$(\overline{\mathcal{N}}_{TK})_{G,G'} = \frac{1}{|\vec{\mathbb{W}}|} \sum_{\vec{W}\in\vec{\mathbb{W}}} \langle\langle \Pi_{\bar{0}} \overline{G} | \mathcal{R} \left(\prod_{k=1}^{K} \overline{W}_k \right) \left(\prod_{k=K}^{1} \mathcal{N} \overline{W}_k \right) | \Pi_{\bar{0}} \overline{G}' \rangle\rangle \tag{4.7}$$

Here we divide the noise process into K steps and apply a random Pauli gate W_k at the beginning of each step. At the end, we undo all these random Pauli gates by applying their inverse $\prod_{k=1}^{K} W_k$ and then perform quantum error correction. We denote the set of K Pauli gates chosen using a vector $\vec{W}$. Similar to our arguments in Section 4.2.2, multiple rounds of twirling correspond to the average of all the Pauli conjugation schemes, thus one of the Pauli conjugations will be optimal and outperforms twirling.

As detailed in Appendix B.12, if we want to find the equivalent conjugations to reduce the search space for multi-round conjugation, we can use similar arguments about the structure of the noise (Section 4.3.2) and the symmetries in both the noise and the code (Section 4.3.3), while the arguments about interaction with the code space to remove stabilisers and logical operators (Section 4.3.1) can only be applied to the outer-most round of conjugation.

The search space of possible conjugations grows exponentially with the number of rounds while the number of symmetries that we can utilise is less than the one-round case (since we cannot remove all the stabilisers and logical operators from the twirling generating set). Hence, iterating over the whole search space might not be practical for a large number of rounds. However, we can still sample different conjugation schemes in our reduced search space to find a better scheme than doing nothing or even twirling, though such a scheme might not be optimal.

4.6 Discussion

In this chapter, we have shown that when doing one round of quantum error correction on a coherent noise channel, part of its coherence can actually be used to improve its logical fidelity using Pauli conjugation, which outperforms twirling. To search for the optimal Pauli conjugation under a given noise model using a given quantum error correction code, we use the properties and the symmetries of the noise and the code to identify the equivalent conjugations to reduce our search space. We applied our techniques to the Steane code, the Shor code and the surface code under a global Z rotation noise, reducing the 4^n possibilities of Pauli conjugation to 2, 4 and 6 equivalent classes respectively for those three codes. Iterating over these different classes of conjugations, we managed to find the optimal conjugations for each code, which resulted in higher logical fidelities than the twirled and original noise channel. We have shown via simulation that the advantages of the optimal Pauli conjugation schemes remain with gate errors present. Conjugation can also lead to higher concatenated thresholds than the twirled threshold. The conjugated threshold showed improvements over the original thresholds by 40%, 160% and 110% for the three codes we considered under the coherent Z noise. We showed that by using logical twirling to remove the 'harmful' coherence within the error channel, we can extend the advantages of Pauli conjugation to multiple rounds of error correction. We also briefly discussed how to extend our arguments into multiple rounds of Pauli conjugation.

Compared to twirling, Pauli conjugations do not require the implementation of a random circuit, and the weights of the gates that we need to implement can be on average much smaller than twirling as shown by our examples. Being a deterministic scheme, it can be implemented in hardware systems in which modifying the circuit at each run is hard. It can also be used in quantum communication to combat coherent noise in the communication channel without needing to transmit the extra random bit needed by twirling. Single-qubit Pauli gates are usually the gates with the highest fidelity, combining with the fact that the Pauli conjugation gates we need to implement can be low-weight, it should be resilient to gate errors, as

shown by our simulation. Hence, Pauli conjugation can be a practical way forward to mitigate errors in real experiments.

The way we reduce the Pauli conjugation search space is highly dependent on the code we use and the noise model we have. Though our techniques work for the simple examples that we have considered, searching over all possible Pauli conjugations may not be feasible when the size of our system increase, when there are very few symmetries in the noise or when we are considering multiple rounds of Pauli conjugation. Hence, we might want to find a way to construct the optimal conjugation based on the mechanism of conjugation in Section 4.2.3, or at least find a better searching strategy than random sampling. Furthermore, we may not know the full noise model in practice. As discussed in Section 4.2.3, conjugation will only act on the coherent components in the channel, thus to find the optimal (or close-to optimal) conjugation gate we only need the information about the dominant coherent component in the channel without needing any information about the incoherent parts or the other small coherent components. In the worst case scenario, we can still sample over the different Pauli conjugations based on any limited information we have to find a scheme with better performance than the original noise channel instead of finding the optimal one.

The above ideas can be tested by applying Pauli conjugation to more general error channels beyond the global Z rotation. An example will be the general local Z noise channel considered in Ref. [18] or some non-biased noise models like those considered in Ref. [56]. To see if the conjugation technique is valuable in fault-tolerant computation, it will also be interesting to see how Pauli conjugation will perform against gate-level coherent noise and whether it can improve the surface code threshold (instead of the concatenated threshold) given a realistic noise model.

There are several degrees of freedom we can add to further optimise our noise tailoring schemes. Firstly, throughout this chapter we have been focusing on conjugation using Pauli gates, it will be interesting to extend our technique to Clifford gates or even general unitaries. We can also look into the case where we allow Clifford correction [22]. We definitely did not exhaust all the ways to reduce

the Pauli conjugation search space. For example, we have only been focusing on the permutation symmetry of code and noise, which at best can only prove the operators with the same weight are equivalent. A next step could be including other Clifford symmetries like CZ gates, etc.

Our conjugation scheme, especially the multi-round variant, in a way can be viewed as bang-bang dynamical decoupling tailored to a given quantum error correction code. Attempts have been made before to study the effect of dynamical decoupling within the context of quantum error correction [57], though without explicitly considering the code structure as we did in this chapter. It will be a fruitful area to adapt more schemes in the established literature of dynamical decoupling [20] into the context of quantum error correction taking into account the code structure. We may get a fuller understanding about how to search for better multi-round conjugation scheme from the way we optimise dynamical decoupling using average Hamiltonian arguments [58] and group theoretic arguments [59]. Ideas like non-equidistant pulses [60], robust decoupling sequences and higher-order decoupling [20] can also be extended into multi-round conjugation.

Besides applications in quantum error correction for memory, the conjugation technique can also be extended into other fields like quantum metrology and quantum simulation. For quantum metrology with error correction [61–63], we hope to find conjugation schemes that can tailor the noise into a form that is less damaging to the code and/or tailor the signal into a form towards which the code is more sensitive. When applied to symmetry verification in quantum simulation [64–66], conjugation may enable more noise to be detected via transformation of the previously undetected noise components. In the above applications, it is likely that we need to develop more complex conjugation schemes beyond one-round Pauli conjugation.

5

A Silicon Surface Code Architecture Resilient Against Leakage Errors

This chapter is adapted from the research published in Quantum [67], of which the present author is the main author. Section 5.2 in this chapter has drawn heavily on the hardware expertise of fellow co-authors Michael Fogarty, Simon Schaal, Sofia Patomäki and John Morton, especially for Section 5.2.5, which is predominantly the work of Michael Fogarty but is kept here for completeness. The other co-author Simon Benjamin contributed to the ideas related to the stabiliser cycles used in Section 5.4.

Contents

5.1 Introduction

In the previous two chapters we have described ways to mitigate the damage caused by coherent errors in the context of quantum error correction. In this chapter we will look into practical hardware implementation of quantum error correction schemes. As discussed in Section 2.6, one of the leading candidate codes for practical quantum error correction is the surface code due to its 2D structure, local checking operations and high error threshold close to 1% [68]. Surface code architectures have been proposed for leading quantum information processing platforms including superconducting qubits [33], trapped ions [69] and semiconductor spin qubits [70, 71]. However, the qubit overheads of surface codes can be significant: it is estimated that $> 2 \times 10^8$ physical qubits with gate error rate 10^{-3} might be needed to perform a non-trivial Shor's factoring algorithm [72]. These considerations motivate the development of qubit implementations which offer the prospect for high-density 2D arrays. The high-qubit density offered by silicon-based spin (SS) qubits (as high as 10^9 cm^{-2}) combined with the possibility of leveraging the conventional semiconductor integrated circuit industry [73] make this platform attractive for fault-tolerant universal quantum computing.

Like all qubit hardware approaches, scaling up SS qubits brings a number of practical requirements associated with qubit addressing for calibration, tuning, operation and readout. Indeed, the high qubit densities offered by SS qubits leads to challenges in routing classical control lines, while minimising cross-talk and managing heat dissipation [73]. A number of architectures for scaling up SS qubit arrays have been proposed to address such challenges: for example, Veldhorst *et al.* [74] proposed a compact quantum dot array controlled via a crossbar geometry, enabling N qubits to be controlled with $\sqrt{N}$ classical control lines, albeit using control transistors below the dimensions of current technology [73]. Li *et al.* [75] went further with a half-filled crossbar architecture that provides more space for classical control

lines, though the use of shared control lines brings tight requirements for qubit homogeneity and limitations on the parallelisability of operations. Buonacorsi *et al.* [76] have suggested connecting many small quantum dot modules using electron shuttling in order to provide the space for individual control lines. Smaller quantum dot modules are also easier to calibrate and the operations within the modules may be expected to have higher fidelities. However, such shuttling architectures require distribution of entanglement between modules and this is likely to impact the fidelity and speed of inter-module operations.

While such influential architectures have been designed to accommodate error correcting codes that compensate for computational errors, they do not address so-called 'leakage errors' in which the quantum system escapes out of the computational subspace. For SS qubits, one form of leakage errors arises from the migration of charge: Controlling SS qubits involves tuning tunnelling barriers, changing on-site energies and/or shuttling electrons, and each of these operations may lead to electrons escaping out of the quantum dots. Since leakage errors cannot be corrected (and may even be exacerbated) by the usual quantum error correction protocols, they will accumulate and eventually corrupt the surface code even if the probability of these leakage errors is very small [77]. Furthermore, unlike most of the other types of leakage errors [78–84] which occur as independent events, a leaked charge from one dot might propagate through the quantum dot surface code array and corrupt other dots. Charge leakage errors thus could be very damaging to the surface code due to the correlations in errors.

In this chapter, we will introduce a surface code architecture based on SS qubits that is designed to be robust against leakage errors. We first introduce the components of our hardware in Section 5.2, and then discuss leakage errors in our architecture in Section 5.3. Then, in Section 5.4, we describe how surface code stabiliser checks are performed, and obtain a threshold for the gate errors and leakage errors. Finally, we summarise the key features of this approach and discuss possible improvements and extensions.

5.2 Physical Implementation

The physical layout of the silicon quantum dot surface code architecture we consider is shown in Fig. 5.1. We have included elongated mediator dots [85] to provide the basic two-qubit gate operation while increasing the fundamental inter-qubit spacing to more readily accommodate measuring devices for ancilla readout, and electron reservoirs for initialisation and reset of quantum dots. Quantum information resides in the *data dots*, whose error information is extracted by the *ancilla double-dots* via interactions through the *mediator dots*.

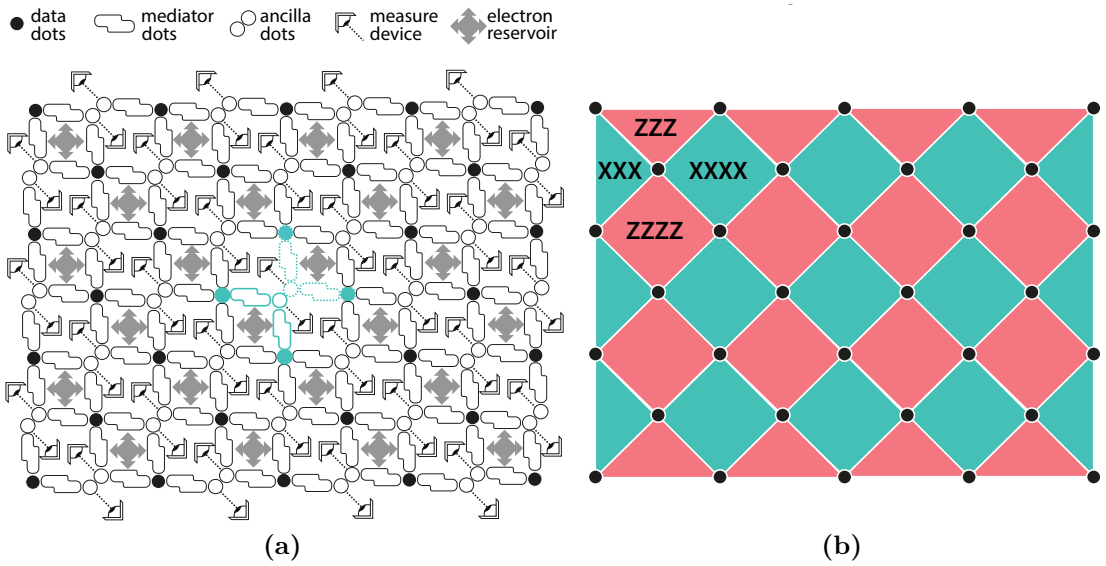


Figure 5.1: Overall architecture layout, including (a) the arrangement of the key physical components of the system and (b) its correspondence to the components of the surface code. An X-stabiliser plaquette is highlighted in (a), which can be divided into two parts each interacting with one half of the double quantum dot ancilla in the centre.

5.2.1 Data Qubits and Single-qubit Gates

Each data qubit is represented by the spin state of an electron within an electrostatically defined quantum dot [86]. The lifting of the spin degeneracy via an applied magnetic field gives access to electron-spin resonance (ESR) [87–89] or electrically-driven spin resonance (EDSR) [90, 91] techniques, which have been used to produce control fidelities of electron spin qubits in silicon of up to 99.6–99.9% [87, 91]. Qubit relaxation times (T_1) can reach $> 1\,\mathrm{s}$ [92–94] and using isotopically

enriched ^{28}Si substrates [95], qubit coherence times can be extended up to the limits of the fluctuation timescales within the magnetic environment (e.g. $T_2^* \sim$120 μs [87]). Decoupling schemes can then be used to yield longer qubit operation times ($T_2 \sim$ 28 ms [96]), and these can be integrated into algorithms [97, 98] or single qubit gates designed via gradient ascent pulsed engineering [99] to be inherently robust against environmental noise [89].

Qubits formed electrostatically in highly strained silicon also have the advantages of splitting off the excited states when quantum dots are strongly confined. Such systems often show valley excited states of $\sim$ 0.1 THz [87, 92, 100], and orbital energies of $\sim$ 1 THz [100], and such excited state energies can be electrostatically tuned via the Stark shift [92, 96]. In a confined quantum dot with diameter, say 30 nm, a large Coulomb repulsion $U \sim 2$ THz [100] is produced — this effectively prevents additional charges entering such dots during the execution of the code, leaving us to address the possibility of charge leakage out of the dot.

5.2.2 Ancilla Qubits and Read-out

Our proposed ancilla qubit is represented by the spin state of a *pair* of electrons distributed across two quantum dots (each similar in size to the data dots). By initialising in a singlet state, a failed stabiliser check of its neighbouring data qubits transforms the ancilla spins into a triplet state [97], such that we can use Pauli spin blockade (PSB) and its effect on interdot tunnelling [101, 102], to determine the outcome of the stabiliser cycle. PSB can be detected in single-shot through charge sensing [103], or via gate-based dispersive readout [104–106] as suggested by the measurement devices in Fig. 5.1[1] [103, 107]. The ancilla qubits are initialised via the (0,2) electron occupation state of the double quantum dot (or an equivalent $(n, n+2)$ state), where the ground state is a singlet and can be rapidly prepared through 'hot-spot' relaxation near the (1,1):(0,2) charge transition [108].

Previous schemes [74, 97] have employed a second quantum dot as part of the ancilla structure as a reference state which does not participate in the stabiliser

[1]Note that in Fig. 5.1 we have attached two measurement devices to each pair of double dots for enhanced measurement sensitivity, but one measurement device could be sufficient in practice.

check — the primary function being to enable measurement by PSB. In contrast, in our proposal we treat both ancilla dots on an equal footing, allowing both dots in the ancilla pair to interact with data qubits. In addition to reducing complexity in connectivity, this approach enables interactions between the data qubits and ancillae to be performed in parallel using both of the ancilla dots, halving the time needed to perform a stabiliser cycle.

Operations that are symmetric under the exchange of the two spins cannot bring the quantum state out of the singlet (exchange-antisymmetric) or the triplet (exchange-symmetric) subspace. Hence, global ESR (single qubit gates) can be applied to all the data qubits without affecting the double-dot ancilla, which is useful when switching between X and Z stabiliser check cycles of the surface code.

The type of error (e.g. X or Z) detected by single-dot ancillae depends on the basis in which they are prepared and measured, while two-dot ancillae prepared in the singlet state can be used to detect both X and Z errors [97]. In standard parity check circuits, the X and Z errors in the data qubits will be transformed into Z errors in the ancilla qubits using CZ and CNOT respectively, which can be detected by preparing and measuring the ancilla in the X-basis. In the case of double-dot ancilla, this can be achieved by mapping the singlet state and the zero-spin triplet state of the two-dot ancillae to the X-basis eigenstates:

$$\frac{1}{\sqrt{\sqrt{2}}}\left(|01\rangle \mp |10\rangle\right) \mapsto |\pm\rangle_{anc}\,.$$

On the L.H.S. we have the state of the two physical spins within the ancilla double-dot and on the R.H.S. we have the corresponding state of the ancilla qubit.

From the mapping we can see that while Z gates on individual physical spin correspond to Z gates on the ancilla qubit, X and Y gates on the individual physical spin will take the spin pairs out of the zero-spin subspace, resulting in leakage errors on the ancilla qubit. The effect of such leakage errors will be detailed in Section 5.3.

5.2.3 Mediators and Two-qubit Gates

When two-qubit gates are performed using direct exchange interactions between nearest-neighbour quantum dots, the resulting qubit pitch is typically on the scale of tens of nanometres. On the other hand, control and read-out electronics associated with each qubit are more comfortably accommodated with larger spacings at the level of at least several hundreds of nanometres. To extend the range of the exchange interaction, an elongated quantum dot can be used as a 'mediator' [109, 110]. Our architecture employs effective two-electron (i.e. even-occupation) quantum dots as mediators for the exchange interaction between a data dot and one half of the ancilla double-dot as shown in Fig. 5.2. For simplicity we assume two-electron occupation in the mediator, but in practice four-electron or other values may be preferable, for example to mitigate a small valley-orbit splitting [111].

The mediators do not themselves carry any quantum information and our computational subspace only consists of the spin states of the electrons in the two side dots. Ruderman-Kittel-Kasuya-Yosida (RKKY) exchange interactions communicated by the mediators occur between the spins in the two side dots, with a strength [109] given by:

$$J = -2\left(\frac{t_{R2}^* t_{R1} t_{L1}^* t_{L2}}{\Delta_R \Delta_M \Delta_L} + \text{c.c.}\right) \tag{5.1}$$

where t_{ab} is the tunnelling energy from orbital a to b and $\Delta_{R,M,L}$ are the energies associated with various electron hopping processes starting from the ground state as indicated in Fig. 5.2.

We consider mediator dots of dimensions 30 nm × 300 nm, which leads to $\Delta_M \sim 10\,\text{GHz}$, and assume a tunnelling energy between the mediators and data/ancilla dots of $t \sim 1\,\text{GHz}$ (corresponding to an interdot spacing of $\sim 10\,\text{nm}$). By tuning the on-site energy of the mediator dot, we can change the value of $\Delta_{R/L}$ and hence control the strength of the exchange interaction. $\Delta_{R/L}$ is bounded to be at least the tunnelling energy and at most the Coulomb repulsion energy in the data dots. Hence, we use $\Delta_{R/L} = \Delta_{\text{on}} = 10\,\text{GHz}$ to turn on the exchange interaction, and $\Delta_{R/L} = \Delta_{\text{off}} = 1\,\text{THz}$ to turn off the exchange interaction. Using Eq. (5.1), the

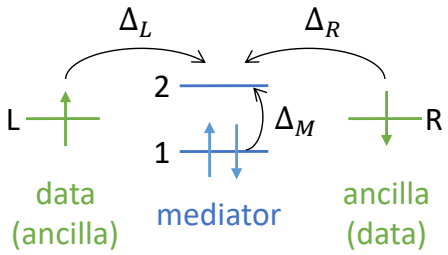
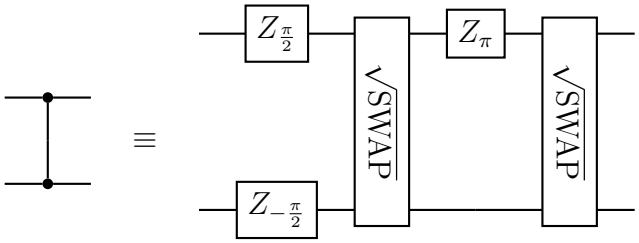


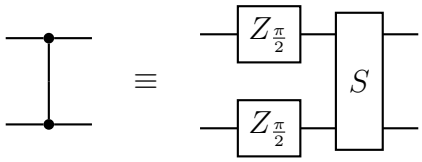
Figure 5.2: Core two-qubit gate between data and ancilla qubits, achieved via a mediator. Three quantum dots with orbital L/R in the left/right dot, each of which is either a data dot or one half of an ancilla structure, and orbitals 1 and 2 in the middle mediator dot. We consider a total of four electrons in this three-dot system and assume the charging energy of the side dots is sufficiently large (due to their small size) to forbid further occupancy. Electrons may be excited to the mediator state 2 from any of L,R or 1 orbitals, with some energy cost indicated.

strength of the exchange interaction is $J_{\text{on}} = \frac{t^4}{\Delta_{\text{on}}^2 \Delta_M} = 1\,\text{MHz}$ when on, and has a residual value $J_{\text{off}} = \frac{t^4}{\Delta_{\text{off}}^2 \Delta_M} = 100\,\text{Hz}$ when nominally off. This level of residual exchange interaction leads to an expected error probability ($\frac{J_{\text{off}}}{J_{\text{on}}} \approx 10^{-4}$) well below the threshold of the surface codes and hence ignored in our discussion. We assume the mediated exchange is controlled through the detuning of the mediator dot under the fixed tunnel coupling naturally formed between adjacent dots [96] — it is also possible to use additional electrodes for controlling the tunnel coupling between adjacent dots [112], albeit at the cost of greater gate complexity.

A difference in g-factors or in z-magnetic field in the left and right (L/R) dots produces a difference in the Zeeman splitting between them, which we denote Ω. When the device is tuned to satisfy $\Omega \ll J$, the exchange interaction enables us to implement $\sqrt{\text{SWAP}}$ gates (see Appendix C.1 for details). Along with single-qubit Z rotations, they can be used to create CZ gates as proposed by Loss and DiVincenzo [113]:



On the other hand, in the limit where $\Omega \gg J$, we can achieve a dipole-dipole like interaction between the two dots mediated by the exchange interaction, which can be used to implement $S = \frac{1}{\sqrt{2}}(I_1 I_2 + i Z_1 Z_2)$ (see Appendix C.1 for details). Along with single-qubit Z rotations, they can be used to create CZ gates as proposed by Meunier *et al.* [114]:



Two-qubit gates in silicon QDs based on direct exchange have been demonstrated [96, 115], whose fidelity has been improved to 98% [116], fast approaching the fault-tolerant threshold. Mediated exchange using empty [117] or multi-electron [85] mediator dots has also been demonstrated in GaAs quantum dots. In our architecture, we use an effective two-electron mediator dot to provide exchange interactions that can be more readily to switched on and off than with empty mediator dots due to a lower virtual energy cost, noting also that keeping the occupancy low leads to higher expected fidelity than the multi-electron mediators due to the simpler electron environment in the mediators [109].

5.2.4 Realisations of the $\Omega \ll J$ and $\Omega \gg J$ Regimes

As explained above, the RKKY exchange operation produced by the mediator dot can be utilised to construct the CZ operation either directly via the S gate when $\Omega \gg J$, or indirectly via $\sqrt{\text{SWAP}}$ operations when in the $\Omega \ll J$ regime. Embedding our device within a uniformly applied external magnetic field enables

single qubit operations via ESR [87], while also accessing the $\Omega \gg J$ regime through the natural variation in electron g-factor inherent to the qubit platform [96, 118].

However, single-qubit gates achieved by ESR have relatively slow speed ($\sim$ 1 MHz), limited by the magnitude of the oscillating magnetic field. An alternative method to implement single-qubit gates is EDSR [119], which can be achieved in a uniform magnetic field [120] by exploiting the spin-orbit coupling in silicon [121, 122]. More commonly, EDSR is achieved using a magnetic field gradient created at the quantum dot, usually by placing a micromagnet in proximity. In this way, when the electron is perturbed via an oscillating electric field, it experiences an effective oscillating magnetic field which drives the spin rotation. EDSR can be more than an order of magnitude faster ($>$ 10 MHz [91]) than ESR, however, qubits capable of EDSR driving can be more susceptible to decoherence from charge noise of the control gates. A balance can be struck between the speed of the single-qubit gates and qubit decoherence to achieve single-qubit EDSR gates with fidelity of 99.9% [91].

As shown in Section 5.2.2 and further discussed in Section 5.4.1, we do not need to apply single-qubit gates to our ancillae, and so there is no advantage in furnishing them with micromagnets. This fact, together with the additional spacing between qubits afforded by the mediator dots, facilitates the ability to deposit a micromagnet array such that each data qubit is located in the vicinity of a magnetic field gradient. This local field gradient observed by the data qubits facilitates EDSR, and also produces the offset field between data and ancilla qubits attaining the $\Omega \gg J$ regime.

A second regime in which the qubit array can be operated is the $\Omega \ll J$ regime. In order to achieve this regime, no field gradients due to micromagnets are utilised and the external magnetic field must be low, such that Ω attributed to the variation in the electron g-factor is minimal. Given current disorder levels within Si QDs, the use of an applied field of $\sim$30 mT (enabling ESR at $\sim$1 GHz) would yield $\Omega \sim J$. To push into the $\Omega \ll J$ regime, the platform could be further engineered for larger J values, or for the reduction in disorder levels giving rise to variation of electron g-factors such that they can be mitigated effectively using the Stark shift.

5.2.5 Charge Reservoirs and Initialisation

Charge reservoirs remain an integral component of modern test-bench quantum devices as they are used to supply electrons to quantum dots, facilitate traditional spin-to-charge readout [123] or more recent improved methods [124, 125], as well as providing a relaxation path for rapid spin initialisation [108]. However, modern concepts of scaled qubit platforms that exploit CMOS technology typically envisage larger devices with densely-packed quantum dots, leading to reservoirs being pushed to the borders of large 1D [97] or 2D [74] arrays. Other architectures have the capacity for reservoirs to be located in specialised modules where spins could then be shuttled into arrays through the use of long-distance highways [75]. With the relative absence of reservoirs in many modern architectures, spin initialisation and readout relies predominantly on Pauli spin blockade methods, with some schemes also utilising thermal relaxation as an initialisation method [73].

In the architecture presented here, we strive to maintain the advantages of having integrated spin reservoirs, without compromising the advantages of CMOS as a platform capable of realising arrays of densely-packed qubits. This is achieved through the spatial separation afforded by the larger scale mediator dot between each data/ancilla dot as seen in Fig. 5.1. With a gate pitch of 30–40 nm [87, 92] in recent 2D planar SiMOS QD designs, and with the possibility of reducing this through the use of smaller length scales (e.g. more recent CMOS technology nodes), the indicated 300 nm separation due to the mediator generates enough space for the integration of the reservoirs as well as the planar fan-out of metallic gate structures required to define/confine the 2D quantum dot structures. Specifically, this facilitates the ability to maintain gated connections between the reservoir and the mediator dot, meaning the tunnel rate can be tuned or made switchable for either rapid interaction as required during initial population of a qubit array, or appropriately tuned for slow reset of mediator dots during periods of inactivity.

The smallest energy scale for the mediator system is $\Delta_M \sim 10$ GHz, which remains $\sim 5\times$ larger than conservative electron temperatures of ~ 100 mK. Couplings required for Elzerman readout [123] are ~ 100 μs in typical CMOS

systems [87], which is long compared to the CZ execution time, however dispersive sensing has seen device operation with tunnel couplings on the order of tank circuit frequencies of 1–10 MHz, which would place the mediator reset, or initialisation protocols within an appreciable time budget with respect to the error correction scheme. Since we are attaching the mediators to the reservoirs for the purpose of charge reset instead of coherent operations such as readout, the tunnel rate between the mediators and the reservoirs can be made larger than timescales required for high fidelity single-shot detection via classical electronics.

5.3 Leakage Errors

5.3.1 Background

A *leakage error*, in which the state of the quantum system escapes out of the computational subspace, is not corrected by typical quantum error correction protocols. If left uncorrected, even low-probability leakage errors may accumulate and eventually corrupt the logical qubits. Wood and Gambetta have presented a recent overview on leakage error models and how they can be quantified [126]. To correct leakage errors, we need to first reduce them to errors that fall within the computational space, which can then be handled by the quantum error correction scheme. This can be achieved by detecting the leakage errors [127, 128] and replacing the leaked qubits with fresh qubits, or employing leakage reduction protocols [129–131] to all qubits without the need of leakage detection. In practice, the sources of, effects of, and solutions to leakage errors are strongly hardware-dependent.

In our architecture, we use the term *qubit dots* to refer both to data dots and ancilla dots in which quantum information resides. Within our computational subspace, all qubit dots will be in the ground charge configuration. The electrons in the data single-dots are allowed to have any spin configurations while the electron pairs in the ancilla double-dots are restricted to the spin-zero subspace. The wrong spin or charge configuration of the system will leads to spin leakage or charge leakage errors respectively.

5.3.2 Robustness against Spin Leakage Errors

Spin leakage error means the spin configuration of the system go out of the spin subspace that defines the computational subspace, given the right charge configuration. There is no spin leakage for the data qubits in our case since all of their spin configurations are within the computational subspace. Hence, the only spin leakage in our architecture will be the ancilla qubits escaping out of the spin-zero subspace. Ancilla spin leakage cannot spread to the data qubits via interactions (since there is no data spin leakage), which means that spin leakage cannot propagate in our architecture. Furthermore, the spin leakage errors of the ancilla qubits will be removed in every new round of stabiliser checks when we reinitialise the ancilla. These properties ensure spin leakage will not lead to spatially or temporally correlated errors in our architecture, permitting robustness against spin leakage.

Now if we take a look at the stabiliser check circuit in Fig. 5.3, we can see that before the readout, the stabiliser check process can be viewed as two non-interacting halves. Within each half, there will be two data qubits interacting with one spin within the ancilla spin pair in the same way as interacting with a single-spin ancilla qubit. Hence, before the readout, we can study all the errors on an ancilla qubit simply by treating each spin within the ancilla spin pair as an individual qubit. The spin leakage errors of the ancilla qubits can be taken into account in this way because they can be represented by unitaries applied on the ancilla spin-pair, e.g. X and Y gates on individual spins are two possible forms of ancilla spin leakage errors as mentioned in Section 5.2.2. Hence, we can see that the effect of spin leakage errors on our double-dot ancillae should be similar to the effect of computational errors in some alternative schemes using two single-dot ancillae.

In the readout stage, we need to consider the errors on both spins together. The double-dot ancilla singlet-triplet readout may fail when there are non-symmetric errors occurring on the two spins, compared to the single-dot ancilla X-basis readout which will fail under any non-X errors.

5.3.3 Robustness against Charge Leakage Errors

Charge leakage error means the charge configuration of the qubit dots moves away from the ground charge configuration that our computational subspace resides in. In our architecture, the electron-electron repulsion energies in the qubit dots are much higher than any other energy in our system, thus we do not consider the charge leakage errors due to extra electrons entering the qubit dots. Instead, we will focus on the charge leakage errors due to electrons escaping out of the qubit dots. Possible sources of such charge leakage errors include decoherence of charge eigenstates during exchange interactions [132] (see also Appendix C.6) or electrons escaping out of the 2D electron gas confinement.

Charge leakage is much more damaging than spin leakage in two ways. First of all, charge leakage can be transferred from one qubit to another via gate operations, which will lead to propagation of leakage errors in the qubit array. Secondly, it cannot be simply removed by reinitialisation of the spin configuration. The missing charge must be replenished using charge reservoirs, which can be hard to integrate into a densely-packed quantum dot arrays.

When an electron escapes from a qubit dot in our architecture, it can be restored via relaxation of electrons in the neighbouring mediator dots into the empty qubit dot. The time scale of such relaxation is indicated by the T_1 time of charge qubits in semiconductor quantum dots. Wang *et al.* [133] measured the charge relaxation time in Si/SiGe double quantum dots, showing strong dependence on the tunnelling energy between the orbitals and weak dependence on the detuning between the orbitals. For the tunnelling energy regime that we are interested in ($t \sim$1 GHz), the relaxation time was around 10 ns, which is much shorter than the other time scales in our systems (all the gates in our system operate at μs time scale). Hence, we can assume that once a charge leakage error occurs, a relaxation process quickly takes place, in which an electron in one of the adjacent mediator dots hops down to fill the empty qubit dot, restoring the charge configuration of the qubit dots. Therefore, even without any active leakage error detection and correction or applications of

any leakage reduction protocols, our architecture has a useful inherent behaviour whereby charge deficit transfers from qubit dots to mediator dots.

The relaxation process that restores the charges in the qubit dots can, however, result in missing/extra charges in the mediator dots, which, uncorrected, would produce faulty exchange gates. This can be corrected by connecting all the mediators to the charge reservoirs that are used for the initial population of the quantum dot array. Since the mediators do not carry any quantum information, such connection to reservoirs should not introduce qubit errors.

Errors due to unwanted coupling between the charge reservoirs and the qubit array are minimised by decreasing the tunnelling energy between the reservoir and the mediators, though this produces a longer reset time for the mediators. As we will see in Section 5.4, our surface code is partitioned into regions which are active/inactive at different times during a full cycle. This provides an opportunity for a given mediator to reset with its nearby reservoir during an idle period, without adding delay to the error correction processes.

Without the use of mediators, leakage errors occurring in the qubit dots require leakage correction schemes to be applied. As discussed in Appendix C.4, such schemes would introduce large qubit/runtime overheads [129, 131], limits on the choice of data/ancilla qubits [130] and/or require extra components for charge detection or reset introduced within a potentially dense qubit array. In contrast, in the architecture we propose here the leakage errors are addressed by the inherent charge deficit transfer from the qubits to the mediators and resetting the mediators using charge reservoirs. No additional components are needed since the reservoirs are also used for qubit initialisation and no additional runtime is introduced since the mediator resets can be carried out in parallel with other error checking cycles in the surface code.

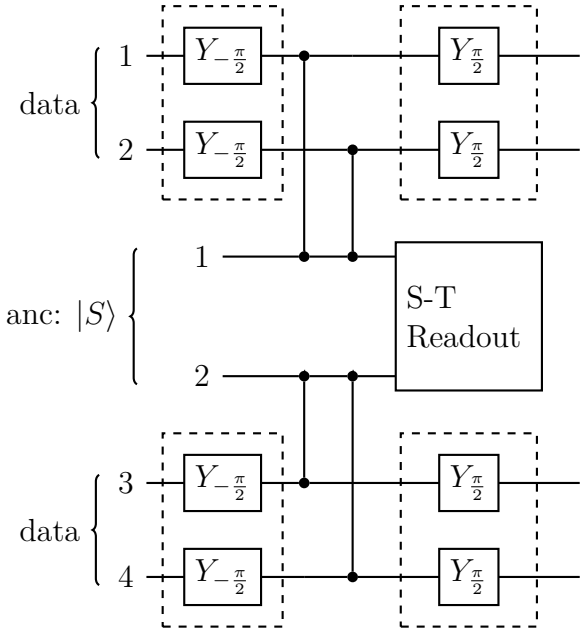


Figure 5.3: The stabiliser check circuit. The Y rotations in the dashed boxes are only included during X stabiliser checks. The two ancillae are initialised in a singlet state $|S\rangle$, and measured using Pauli spin blockade (singlet/triplet dependent tunnelling within the ancilla double-dot).

5.4 Surface Code Simulation

5.4.1 Stabiliser Check Circuit

The details about surface code error correction and threshold simulation is outlined in Section 2.6. Our surface code is implemented by checking the X/Z parities of the data qubits spanned by each plaquette in Fig. 5.1 using the stabiliser-check circuit shown in Fig. 5.3. Here the CZ gates must be further decomposed into $\sqrt{\text{SWAP}}$ or S as outlined in Section 5.2.3. Besides $\sqrt{\text{SWAP}}$ or S, we also need single-qubit Z rotations to construct CZ. Z rotations can be implemented as a combination of X and Y rotations (which can be slow as noted in Section 5.2.4), or using the Stark shift whose speed is limited by the detuning range and whose accuracy relies on careful calibration. Fortunately, in our stabiliser-check circuit, most of the Z rotations on the data qubits can be implemented in a virtual way by shifting the phases of all the future single-qubit rotations pulses [134], and the Z rotations on the ancillae can be omitted since we are performing symmetric operations on the singlet subspace (see Appendix C.1.4 for details). The only Z rotation that we

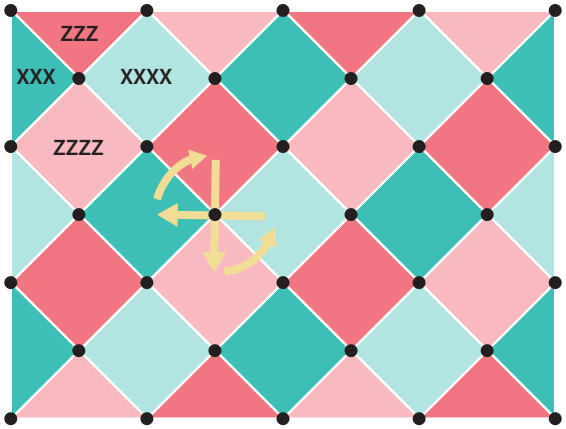


Figure 5.4: Ordering of stabiliser check cycles. Each plaquette is given one of four colours, such that plaquettes of the same colour share no data qubits between them. Stabiliser checks of all plaquettes of a given colour are carried out simultaneously, in the sequence indicated by the arrows.

need to explicitly implement is the Z_π sandwiched by the two $\sqrt{\text{SWAP}}$s, applied to the data qubits. This optimisation to remove single qubit gates substantially reduces the runtime and depth of the stabiliser-check circuit.

As shown in Fig. 5.3, our circuit applies $\sqrt{Y}$ to all data qubits to switch between X and Z stabiliser checks. Because the ancillae are initialised in singlet states, the $\sqrt{Y}$ operations can be achieved using global ESR operations applied to all spins (Section 5.2.2), or through local operations applied only to the data qubits, using EDSR.

5.4.2 Stabiliser Cycle and Error Model

We divide all stabiliser checks into four disjoint partitions, performed in sequence, as shown in Fig. 5.4. When one of the partitions becomes active, any two different stabiliser checks within it are separated by at least one inactive plaquette, across which leakage error cannot propagate. Hence, within each partition, errors (including leakage errors) of one stabiliser check are independent of that of another stabiliser check, such that there are no spatial error correlations beyond a given plaquette. During the stabiliser check of one partition, the mediator reset operation can be activated in the other partitions (see Section 5.3.3). In this way, leakage errors

arising during the active cycle of a given partition do not survive to its subsequent cycle, removing the potential for temporal error correlations. Using this partitioning and sequence of stabiliser updates, the errors in each stabiliser check should be Markovian, removing the temporal and spatial correlations in noise that can be highly damaging to the surface code, and greatly simplifying our error simulation.

Within each stabiliser check, we assume the following error model:

- **Two-qubit gates:** Charge noise leads to fluctuations in the exchange strength J, which can lead to the following errors for the two-qubit gates that we are considering (shown in Appendix C.2):
  - S gate has Z_1Z_2 error with probability p_2.
  - $\sqrt{\text{SWAP}}$ gate has SWAP error with probability $p_2/2$.

  To allow a simple comparison of the thresholds of the two kinds of two-qubit gates, we formulate our simulations in terms of a two-qubit gate error rate p_2 equal to that of the S gate. Assuming that S gates take twice the time required by $\sqrt{\text{SWAP}}$, the variance of the exchange phase Jt accumulated due to fluctuations in S is twice that of $\sqrt{\text{SWAP}}$, and thus the error probability of S is twice of that of $\sqrt{\text{SWAP}}$ (see Appendix C.1.5).

- **Readout:** The current state-of-the-art μs-scale readout scheme can achieve 98% fidelity [135], which is the same as the best two-qubit gate fidelity achieved [116]. Hence, here we will assume the readout error rate can be improved at the same pace as two-qubit gate error rate so that we have $p_{readout} = p_2$.

- **One-qubit gates and initialisation** are assumed to have a common depolarising error probability p_1. The fidelity of one-qubit gates is typically more than one order of magnitude better than two-qubit gates [87, 96, 115], thus we assume $\frac{p_1}{p_2} = 0.1$.

- **Spin Leakage:** As mentioned in Section 5.3.2, spin leakage in the ancilla qubits can be taken into account by considering all the possible errors on the individual spin within the ancilla spin pairs. Its effect on the measured parity can also be considered by flipping the parity result whenever there is asymmetric noise acting on the ancilla spin pairs. Note that is a more damaging noise model than the rigorous model[2], thus should give us a lower bound on the threshold.

- **Charge Leakage:** When considering charge leakage errors, we first note that each stabiliser check can be divided into two non-interacting halves, each with one ancilla dot interacting with two data dots via two mediator dots as shown in Fig. 5.1. Within each half of the stabiliser, when a leakage error occurs and gets restored by the mediators, we will assume the worst-case left-over computational errors in which we have depolarising errors on the whole half (on both of the data qubit and the ancilla spin), so that the leakage error thresholds we derive below can be taken as a lower bound.

  Charge leakage errors are most likely to occur during the tuning of potentials, thus we will assume here the charge leakages will only occur during the CZ gates in the stabiliser checks. If p_{leak} is the probability that a charge leakage error occurs during a CZ gate, then needing to perform two CZ gates in each half of the stabiliser checks means that there is a $2p_{\text{leak}}$ probability that the whole half of the stabiliser check will get depolarised in each round of error check.

5.4.3 Threshold Results

First we consider cases without charge leakage errors ($p_{\text{leak}} = 0$). As shown in Fig. 5.5 (a) and (b), the threshold for p_2 is 0.86% using S gates and 0.76% using $\sqrt{\text{SWAP}}$. Both are comparable to the threshold 0.75% obtained using simple

[2]E.g. if the correct state before the readout is the spin-zero triplet state, then even if leakage errors take our state into other triplet state, our readout result should still be correct even though a leakage due to asymmetric noise has happened

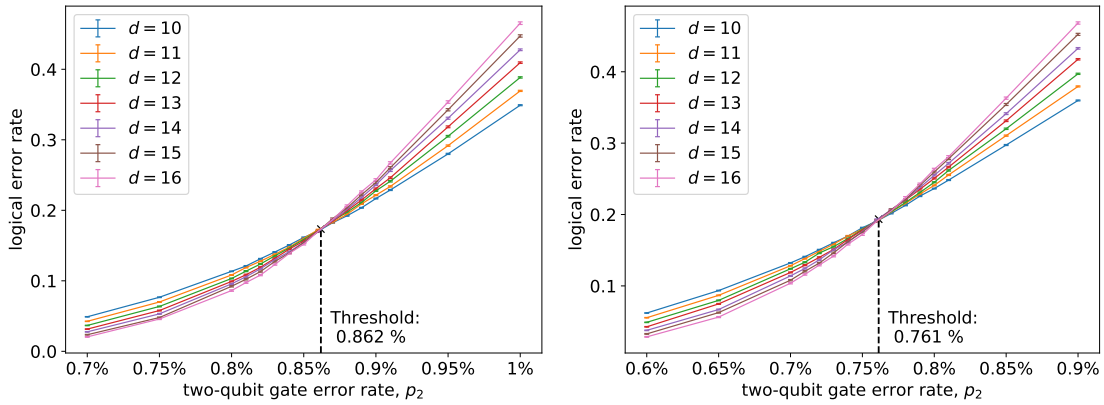


Figure 5.5: Surface code two-qubit gate error threshold calculations in the case of no leakage error ($p_{\text{leak}} = 0$) assuming (a) S gates with error rate p_2 or (b) $\sqrt{\text{SWAP}}$ gates with error rate $p_2/2$. In all calculations, the error rate of single-qubit gates (p_1) and two qubit gates (p_2) is assumed to be fixed $\left(\frac{p_1}{p_2} = 0.1\right)$. d is the code distance of the surface code.

depolarising noise model [136]. The lower threshold for the architecture using $\sqrt{\text{SWAP}}$ is primarily due to the additional Z_π needed to construct the CZ gate. Note that the gate errors here also include the spin leakage errors of the ancillae.

To achieve fault-tolerant quantum computation, our gate error rates need to be below the gate error thresholds. Suppose we manage to achieve a gate error rate below these thresholds at $p_2 = 0.5\%$, then the level of charge leakage error we can tolerate with such a gate error rate are indicated by the p_{leak} threshold in Fig. 5.6 (a) and (b), which are 0.27% with S gates and 0.23% with $\sqrt{\text{SWAP}}$. The charge leakage thresholds we obtained here are on the same order as the gate error rate we assumed here. The energy barrier of the charge leakage errors is usually higher than the errors in the spin space. Hence, we will expect the charge leakage error rate to be much lower than the usual gate error rate and thus below the charge leakage thresholds we obtained here.

If we can further push down the gate error rate (reducing p_2), the charge leakage error threshold will grow, and in the end be bounded by the limit in the case of no gate errors ($p_2 = 0$) where the threshold for p_{leak} is 0.66% (see Fig. 5.6 (c)). The similarity of this pure charge leakage error threshold to that from depolarising noise

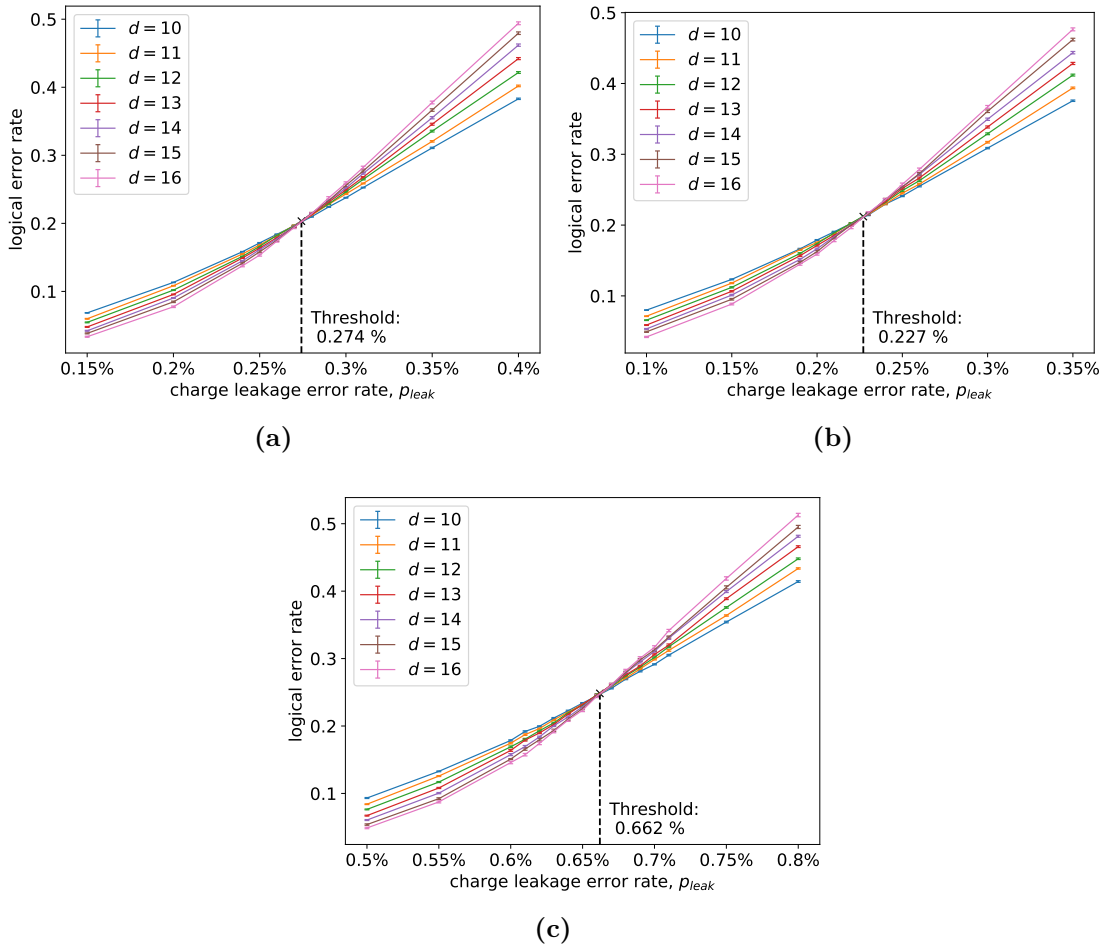


Figure 5.6: Surface code leakage error p_{leak} threshold calculations assuming the use of (a) S-gates with error probability $p_2 = 0.5\%$, (b) $\sqrt{\text{SWAP}}$ gates with error rate $p_2/2 = 0.25\%$, or (c) perfect gates ($p_2 = 0$). In all calculations, the error rate of single-qubit gates (p_1) and two-qubit gates (p_2) is assumed to be fixed $\left(\frac{p_1}{p_2} = 0.1\right)$. d is the code distance of the surface code.

threshold indicates that in our architecture charge leakage errors can be effectively reduced to computational errors (i.e. errors within the computational subspace) via charge relaxation. In other words, even though the resultant computational errors have strong correlations within a given stabiliser check, charge leakage errors can be limited to be no more damaging than other conventional gate errors. The trade-off between the charge leakage threshold and the gate error rates is further illustrated by additional threshold simulation results in Table 5.1. Overall, this architecture shows good tolerance towards the computational errors resulting from charge leakage errors, even with a reasonable amount of gate errors present.

p_2	S-gate	$\sqrt{\text{SWAP}}$
0%	0.66	
0.4%	0.35%	0.31%
0.5%	0.27%	0.23%
0.6%	0.20%	0.14%
0.7%	0.12%	0.05%
0.76%	–	0%
0.86%	0%	nil

Table 5.1: The charge leakage (p_{leak}) threshold under different gate error rate (p_2), assuming the use of S-gates with error rate p_2 or $\sqrt{\text{SWAP}}$ gates with error rate $p_2/2$.

5.4.4 Decoherence Errors

We denote the characteristic time scale of the exchange interaction $T_J = \frac{\pi}{J}$, that of a Hadamard gate ($\sqrt{Y}$) as T_H and that of a Z gate as T_Z. Using the stabiliser cycle outlined in Section 5.4.2 and the stabiliser circuit shown in Fig. 5.3, the time needed for one stabiliser cycle is $T_S^{\text{cycle}} = 8T_J + 2T_H$ assuming the use of S-gates and $T_{\sqrt{\text{SW}}}^{\text{cycle}} = 8T_J + 8T_Z + 2T_H$ with $\sqrt{\text{SWAP}}$. We have not accounted for the time required for initialisation and readout of the ancillae and the mediator resets because they can take place in parallel with other operations of the stabiliser circle. Such operations only become significant to the rate of the stabiliser check once they become an order of magnitude slower than the quantum gates, which is not the case in the range of parameters that we are considering (see Section 5.2.2 and Section 5.2.5).

Using parameters outlined in Section 5.2.3, we expect $T_J \sim 1$ μs. Based on demonstrated electrical tuning of the g-factor, we estimate the duration of Z gates implemented using Stark shifts to be $T_Z \sim 0.25$ μs [137], while the time needed for a Hadamard gate is likely to differ depending on the use of ESR ($T_H \sim 1\mu$s) or EDSR ($T_H < 0.1\mu$s). We can therefore consider two illustrative cases for the stabiliser cycle time. In one case, micromagnets are used to enable the use of S-gates and EDSR, giving $T_{\text{fast}}^{\text{cycle}} \sim 8$ μs (limited by T_J). In the other limit, the slower ESR gates and less efficient $\sqrt{\text{SWAP}}$ are used, giving $T_{\text{slow}}^{\text{cycle}} \sim 12$ μs (limited by T_H, T_Z and T_J).

For electron spins in quantum dots in isotopically-enriched silicon, decoherence times have been reported ranging from $T_2^* = 20$ μs and $T_{2,\text{CPMG}} = 3$ ms in systems with a micromagnet [91] to $T_2^* = 120$ μs and $T_{2,\text{CPMG}} = 28$ ms in systems without

micromagnets [96]. The probability of phase flip error *per stabiliser cycle* using Carr-Purcell-Meiboom-Gill (CPMG) decoupling is hence $\frac{T^{cycle}}{2T_2} \approx 2\times10^{-4}$ to 10^{-3} for the parameters we considered, well within the *per gate* error threshold we obtained in Section 5.4.3. We conclude that the finite decoherence time of spins in silicon measured in devices to date can be tolerated by our surface code architecture.

5.5 Discussion

We have introduced a surface code architecture implemented using spin qubits in silicon quantum dots that is robust against spin leakage errors through its use of single-dot data qubit and robust against charge leakage errors through its use of multi-electron mediator dots. Our approach efficiently unifies the task of maintaining a proper charge distribution (essential for any SS quantum device) together with the task of performing the stabiliser cycles required by the surface code. Charge leakage from the qubit dots is transferred to the mediator dots via fast charge relaxation, and removed using charge reservoirs attached to the mediators, reducing the charge leakage errors to the level of standard computational errors that can be corrected by the surface code. We find that our stabiliser check cycle removes time and space correlations in the remaining computational errors, which can be highly damaging to surface codes. The depth of the stabiliser-check circuit was reduced by the symmetry of the double-dot ancillae and virtual Z gates. Through simulations, we find that the surface code threshold for the computational errors arising from charge leakage errors is 0.66% in the absence of gate errors, showing that its effect can be limited to that of standard depolarising gate errors. Under a reasonable gate error rate 0.5% (which includes ancilla spin leakage errors), we obtain a charge leakage error threshold of $0.23 \sim 0.27\%$, showing good tolerance of our architecture towards charge leakage errors even under gate noise. The fidelity of two-qubit gates is expected to the principal bottleneck for reaching the fault-tolerant level, and experimentally demonstrating a high-fidelity mediated exchange interaction using isotopically enriched silicon will be a key step in validating this architecture.

Besides adding tolerance towards leakage errors, the elongated mediator dots in our structure also relax the density of the qubit dots, offering more space in-plane for the essential measuring devices, charge reservoirs and classical control lines, and facilitating fabrication using (e.g.) CMOS technology [73]. The extra space provided by the mediators and the unique properties of the double-dot ancilla enable more convenient integration of micromagnets which can increase the speed of both the single qubit rotations and stabliser check cycle.

We find that gate mechanisms and energy scales that have already been experimentally reported will suffice to realise a stabiliser cycle time approaching the MHz domain, and that this speed is sufficient to suppress environmental decoherence. This is not a fundamental limit to the operation speed of such a device, however, it makes use of a mediated exchange interaction, which is inherently slower than direct exchange. To push the speed further, data or ancilla spins could be shuttled onto the mediators for direct exchange with a neighbouring ancilla/data spins, and such exchange gates between a single-electron dot and a multi-electron dot have been demonstrated [85, 138]. Shuttling in combination with micromagnet-induced field gradients may introduce significant dephasing noise which may be challenging to correct (e.g. using calibration and single-qubit rotations). As in many approaches, there is a trade-off between speed and error rate to be carefully considered.

A second factor in the speed of the processor operation is charge relaxation, which we have assumed to be fast compared to the gates. If this were not the case charge leakage errors would not be rapidly transferred from the qubit dots to the mediators, and empty qubit dots may remain after a stabiliser cycle, leading to a non-trivial errors of a non-Markovian nature. Nevertheless, we would expect the charge leakage process and the relaxation restoring force to reach some equilibrium, leaving the proportion of the empty qubit dots in the surface code fixed. Further work could study the non-Markovian effects of the empty qubit dots, and the equilibrium value of the leaked qubit fractions under different charge leakage and relaxation models. In cases where charge relaxation time was non-negligible, existing leakage correction protocols like active leakage detection and correction, or leakage reduction units

could be adopted. Indeed, combining active methods with inherent robustness to leakage errors may be advantageous, especially if the native leakage rate is high.

Features from other silicon quantum computing architectures like shared control lines [73, 75] and modularity [76] could also be adopted into our structure, if challenges around the inhomogeneity of quantum dots and shuttling noise can be minimised. Conversely, the introduction of additional quantum dots and electrons into a system to create accessible metastable charge states could be adopted in other approaches to offer robustness against charge leakage errors.

6

Introducing Quantum Error Mitigation

Contents

6.1 Introduction

While many fault-tolerant algorithms are proven to offer up to exponential speed-up over their classical counterparts, their implementations remain a long term goal due to the large qubit overhead required for quantum error correction. With the recent rapid progress of quantum computer hardware in terms of both qubit quantity and quality, culminating with the "quantum supremacy" experiment [8],

one must wonder is it possible for us to perform classically intractable computations on such Noisy Intermediate-Scale Quantum (NISQ) hardware without quantum error correction [47]. However, even with the state-of-the-art gate error rates, the circuits that can perform classically intractable and practically useful tasks are still very noisy due to their large size. To obtain meaningful results from such a noisy circuit, it is essential to employ quantum error mitigation (QEM), which relies on making extra measurements, as opposed to employing extra qubits in the case of quantum error correction, to estimate the noise-free expectation values from the noisy measurement results.

In this chapter, we will use M to denote the number of possible error locations in the circuit, which is usually the number of gates in the circuit. These error locations might experience different noise with different error rates. The expected number of errors occurring in each circuit run will be called the *mean circuit error count* and denoted as μ. We will define the NISQ limit as the limit with large circuit sizes and mean circuit error counts around unity:

$$\begin{aligned} M &\gg 1 \\ \mu &\sim 1. \end{aligned} \tag{6.1}$$

This is also called the Poisson limit since using Le Cam's theorem, the number of errors occurring in each circuit run will follow the Poisson distribution – i.e. the probability that l errors occur will be:

$$P_l = e^{-\mu}\frac{\mu^l}{l!}. \tag{6.2}$$

To reduce the damage caused by these errors, here we will begin by introducing three of the most well-studied quantum error mitigation techniques: symmetry verification, quasi-probability and error extrapolation.

6.2 Symmetry Verification

6.2.1 Background

Suppose we want to perform a state preparation and we know that the correct state must follow a certain symmetry S, i.e. we expect our end state to be the eigenstate

of S with the correct eigenvalue s (or within a set of eigenvalues $\{s\}$). In such a case, we can perform S measurement on our output state and discard the circuit runs that produce states that violate our symmetry. This was first proposed and studied by McArdle *et al.* [65] and Bonet-Monroig *et al.* [64]. Discarding erroneous circuit runs results in an effective density matrix that is the original density matrix ρ projected into the $S = s$ subspace via the projection operator Π_s:

$$\rho_s = \frac{\Pi_s \rho \Pi_s}{\mathrm{Tr}(\Pi_s \rho \Pi_s)} = \frac{\Pi_s \rho \Pi_s}{\mathrm{Tr}(\Pi_s \rho)}.$$

Here we have used $\Pi_s \Pi_s = \Pi_s$.

Now let us suppose we want to measure an observable O, which commutes with our symmetry S. Thus they can both be measured in the same run and we will discard the measurement results in the runs that failed the symmetry verification. The symmetry-verified expectation value of the observable O is then:

$$\langle O_S \rangle = \mathrm{Tr}(O\rho_s) = \frac{\mathrm{Tr}(O\Pi_s \rho)}{\mathrm{Tr}(\Pi_s \rho)} \equiv \frac{\langle O\Pi_s \rangle}{\langle \Pi_s \rangle} \tag{6.3}$$

in which we have used $[S, O] = 0 \;\Rightarrow\; [\Pi_s, O] = 0$. Note that Π_s takes the value 1 if the symmetry verification is passed and 0 otherwise, hence $\langle \Pi_s \rangle = \mathrm{Tr}(\Pi_s \rho)$ is just the fraction of circuit runs that fulfil the symmetry condition, denoted as P_S. Recall that ρ_s is the effective density matrix of the non-discarded runs, which as mentioned is a P_S fraction of the total number of runs. Therefore to obtain statistics from ρ_s, we need a factor of

$$C_S = \frac{1}{\mathrm{Tr}(\Pi_s \rho)} = \frac{1}{P_S} \tag{6.4}$$

more circuit runs than obtaining directly from ρ.

In the method discussed above, $O\Pi_s$ is usually obtained through measuring O and S in the same run. However, sometimes this cannot be done due to, for example, inability to perform non-demolishing measurements. In such a case we need to break $O\Pi_s$ into its Pauli basis [64] and reconstruct it via post-processing, this is discussed in Appendix E.1. In this thesis, we will mainly be focusing on direct symmetry verification instead of post-processing verification, but most of the arguments are valid for both methods besides discussions about costs.

6.2.2 Detectable Errors

We want to produce the state $|\psi_f\rangle$ which is known to fall within the $S = s$ symmetry subspace:

$$S|\psi_f\rangle = s|\psi_f\rangle .$$

To produce the state, we usually start with a state $|\psi_0\rangle$ that follows the same symmetry and uses a circuit U that consists of components that conserve the symmetry:

$$|\psi_f\rangle = U|\psi_0\rangle , \quad S|\psi_0\rangle = s|\psi_0\rangle , \quad [U, S] = 0.$$

Suppose that some error E occurs during the circuit in between the symmetry-preserving components and it satisfies

$$\Pi_s E = E\Pi_{s'} \tag{6.5}$$

in which s, s' are some eigenvalues of the symmetry operator S. We then have:

$$\Pi_s E \Pi_s = E\Pi_{s'}\Pi_s = \begin{cases} E\Pi_s & s = s' \\ 0 & s \neq s' \end{cases} \tag{6.6}$$

$\Pi_s E \Pi_s = E\Pi_s$ means that E is a transformation within the $S = s$ subspace, hence E is undetectable by the symmetry verification using S. $\Pi_s E \Pi_s = 0$ means that E contains no components that map states in the $S = s$ subspace back into the same subspace, hence E is (completely) detectable by the symmetry verification using S. A general error will be a combination of detectable and undetectable error components.

In this thesis, we will be focusing on Pauli errors and Pauli symmetries, for which Eq. (6.6) is reduced to:

$$\begin{aligned} [S, E] = 0 \Rightarrow s = s' \qquad & E \text{ is undetectable} \\ \{S, E\} = 0 \Rightarrow s = -s' \qquad & E \text{ is detectable.} \end{aligned} \tag{6.7}$$

6.2.3 NISQ Limit

If we assume that every local error channel in the circuit can be approximated as the composition of an undetectable error channel and a detectable error channel,

then symmetry verification will have no effects on the undetectable error channels and we can focus only on the detectable error channels. Alternatively, as we will see later in Section 6.3.3, we can use quasi-probability to remove all the local undetectable errors in the circuit, leaving us with only detectable error channels. The expected number of *detectable* errors occurring in each circuit run is denoted as μ_d. Taking the NISQ limit and using Eq. (6.1), the probability that l detectable errors occur in the circuit is:

$$P_l = e^{-\mu_d} \frac{\mu_d^l}{l!}.$$

Using Eq. (6.7), an odd number of detectable errors will anti-commute with the symmetry and get detected while an even number of detectable errors will commute with the symmetry and pass the verification. Therefore, the fraction of circuit runs that pass the verification of *one* Pauli symmetry is thus:

$$P_S = \sum_{\text{even } l} P_l = e^{-\mu_d} \cosh(\mu_d) = \frac{1 + e^{-2\mu_d}}{2}. \tag{6.8}$$

Note that this is lower-bounded by $\frac{1}{2}$, i.e. at least half of the circuit runs will pass the symmetry verification of *one* Pauli symmetry in the presence of local Pauli noise.

Hence, using Eq. (6.4), the cost of implementing symmetry verification for *one* Pauli symmetry is:

$$C_{S,\mu_d} = \frac{1}{P_S} = \frac{1}{e^{-\mu_d} \cosh(\mu_d)} = \frac{2}{1 + e^{-2\mu_d}} \tag{6.9}$$

which is upper-bounded by 2.

After symmetry verification, the fraction of circuit runs that still have errors inside is:

$$P_{circ} = P_S - P_0 = \frac{1}{2} \left(1 - e^{-\mu_d}\right)^2 \tag{6.10}$$

and the mean circuit error count of the symmetry-verified circuit runs is:

$$\mu_S = \sum_{\text{even } l} \frac{P_l}{P_S} l = \frac{\mu_d}{P_S} \left(\sum_{\text{odd } l} P_l \right) = \frac{\mu_d (1 - P_S)}{P_S} = \mu_d \tanh(\mu_d). \tag{6.11}$$

6.3 Quasi-probability

6.3.1 Background

To describe the quasi-probability method, we will make use of the Pauli transfer matrix (PTM) formalism described in 2.1.3. The quasi-probability method was first introduced by Temme *et.al.* [49], and the implementation details were later studied by Endo *et.al.* [50]. Let us suppose we are trying to perform the operation $\mathcal{U}$, but in practice we can only implement its noisy version

$$\mathcal{U}_\epsilon = \mathcal{E}\mathcal{U}.$$

In addition to $\mathcal{U}_\epsilon$, we can also implement a set of basis operation $\{\mathcal{B}_n\}$. We can decompose the ideal operation $\mathcal{U}$ that we *want to* implement into a set of gates $\{\mathcal{B}_n\mathcal{U}_\epsilon\}$ that we *can* implement:

$$\mathcal{U} = \sum_n q_n \mathcal{B}_n \mathcal{U}_\epsilon \quad \Rightarrow \quad \mathcal{E}^{-1} = \sum_n q_n \mathcal{B}_n.$$

In this way, we are essentially trying to simulate the behaviour of the inverse noise channel $\mathcal{E}^{-1}$ using the set of basis operations $\{\mathcal{B}_n\}$, which can undo the noise $\mathcal{E}$.

If we have a state $|\rho\rangle\!\rangle$ passing through the circuit $\mathcal{U}$ and we perform measurement O, then the observable we obtain during the experiment will be:

$$\begin{aligned}\langle O\rangle = \langle\!\langle O|\mathcal{U}|\rho\rangle\!\rangle &= \sum_n q_n \langle\!\langle O|\mathcal{B}_n\mathcal{U}_\epsilon|\rho\rangle\!\rangle \\ &= Q\sum_n s_n \frac{|q_n|}{Q}\langle\!\langle O|\mathcal{B}_n\mathcal{U}_\epsilon|\rho\rangle\!\rangle.\end{aligned}$$

in which $Q = \sum_n |q_n|$ and $s_n = \mathrm{sgn}(q_n)$. This is implemented by sampling from the set of basis operations $\{\mathcal{B}_n\}$ with the probability distribution $\{\frac{|q_n|}{Q}\}$. We will weight each measurement outcome by the sign factor s_n and rescale the final expectation value by the factor Q.

Now if we break down our computation into many components $\mathcal{U} = \prod_{m=1}^{M}\mathcal{U}_m$, with noise associated with each component, then the observable that we want to measure is:

$$\langle O\rangle = \langle\!\langle O|\prod_{m=1}^{M}\mathcal{U}_m|\rho\rangle\!\rangle,$$

but in reality we can only implement the noisy version:

$$\langle O_\epsilon \rangle = \langle\langle O| \prod_{m=1}^{M} \mathcal{E}_m \mathcal{U}_m |\rho\rangle\rangle.$$

Each noise element can be removed by simulating the inverse channels using the set of basis operations $\{\mathcal{B}_n\}$:

$$\mathcal{E}_m^{-1} = \sum_n q_{mn} \mathcal{B}_n = R_m \sum_n s_{mn} \frac{|q_{mn}|}{R_m} \mathcal{B}_n$$

with $R_m = \sum_n |q_{mn}|$.

Hence, we can get back the noiseless observable using

$$\begin{aligned} \langle O \rangle &= \langle\langle O| \prod_{m=1}^{M} \left(\sum_{n_m} q_{mn_m} \mathcal{B}_{n_m} \right) \mathcal{E}_m \mathcal{U}_m |\rho\rangle\rangle \\ &= Q \sum_{\vec{n}} s_{\vec{n}} \frac{|q_{\vec{n}}|}{Q} \langle\langle O| \prod_{m=1}^{M} \mathcal{B}_{n_m} \mathcal{E}_m \mathcal{U}_m |\rho\rangle\rangle \end{aligned}$$

in which we have used $\vec{n}$ to denote the set of number $\{n_1, n_2, \cdots, n_M\}$ and we have defined $q_{\vec{n}} = \prod_{m=1}^{M} q_{mn_m}$, $s_{\vec{n}} = \prod_{m=1}^{M} s_{mn_m} = \mathrm{sgn}(q_{\vec{n}})$ and

$$Q = \prod_{m=1}^{M} R_m = \prod_{m=1}^{M} \sum_{n_m} |q_{mn_m}|. \tag{6.12}$$

To implement this, we simply sample the set of basis operations $\{\mathcal{B}_{n_1}, \mathcal{B}_{n_2}, \cdots, \mathcal{B}_{n_M}\}$ that we want to implement with the probability $|q_{\vec{n}}|/Q$ and weight each measurement outcome by a sign factor $s_{\vec{n}} = \mathrm{sgn}(q_{\vec{n}})$, so that the outcome we get is an effective Pauli observable O_Q. And the error-free observable expectation value can be obtained via:

$$\langle O \rangle = Q \langle O_Q \rangle . \tag{6.13}$$

Hence, to estimate $\langle O \rangle$ by sampling O_Q, we need C_Q times more samples than sampling O directly, where the sampling cost factor C_Q is:

$$C_Q = Q^2 = \left(\prod_{m=1}^{M} \sum_{n_m} |q_{mn_m}| \right)^2 . \tag{6.14}$$

6.3.2 Application to Pauli Errors

Pauli errors can be inverted using quasi-probability by employing Pauli gates as the basis operations. Any Pauli channel can be written in the form:

$$\mathcal{G}_{p_\epsilon} = (1 - p_\epsilon)\mathcal{I} + p_\epsilon \sum_{G \in \mathbb{G} - I} \alpha_G \, \overline{G} \tag{6.15}$$

with $\sum_G \alpha_G = 1$. We use p_ϵ to denote the total probability of all the non-identity components. This channel can be *approximately* inverted using the quasi-probability channel $\mathcal{G}_{-p_\epsilon}$ since:

$$\mathcal{G}_{-p_\epsilon}\mathcal{G}_{p_\epsilon} \approx \mathcal{I} + \mathcal{O}(p_\epsilon^2).$$

Hence, to the first order approximation, the cost of inverting $\mathcal{G}_{p_\epsilon}$ will be the cost of implementing $\mathcal{G}_{-p_\epsilon}$, which using Eq. (6.14) is

$$C_{Q1,0} \approx (1 + 2p_\epsilon)^2 \approx 1 + 4p_\epsilon. \tag{6.16}$$

Here we have only discussed approximately inverting a Pauli channel because the exact inverse channel can be hard to express in a compact analytical form. However, it can be obtained numerically by first obtaining the Pauli transfer matrix of the noise channel and then performing matrix inversion.

6.3.3 Error Channel Transformation

Instead of removing the error channel completely, quasi-probability can also be used to transform the nature of an error channel. In the case of Pauli channels, suppose we want to transform a channel of the form in Eq. (6.15) to

$$\mathcal{F}_{q_\epsilon} = (1 - q_\epsilon)\mathcal{I} + q_\epsilon \sum_{G \in \mathbb{G} - I} \beta_G \, \overline{G},$$

we can *approximately* achieve this transformation up to first order in q_ϵ and p_ϵ by applying the quasi-probability channel:

$$\mathcal{R} = (1 + (p_\epsilon - q_\epsilon))\mathcal{I} - \sum_{G \in \mathbb{G} - I} (p_\epsilon \alpha_G - q_\epsilon \beta_G) \, \overline{G} \, .$$

This will incur the implementation cost:

$$C_{Q1,q} = \left(|1 + (p_\epsilon - q_\epsilon)| + \sum_{G \in \mathbb{G}-I} |p_\epsilon \alpha_G - q_\epsilon \beta_G| \right)^2 .$$

In the limit of small p_ϵ and q_ϵ, we have:

$$\begin{aligned} C_{Q1,q} &\approx 1 + 2\,(p_\epsilon - q_\epsilon) + 2 \sum_{G \in \mathbb{G}-I} |p_\epsilon \alpha_G - q_\epsilon \beta_G| \\ &= 1 + 4 \sum_{\substack{G \in \mathbb{G}-I, \\ p_\epsilon \alpha_G > q_\epsilon \beta_G}} (p_\epsilon \alpha_G - q_\epsilon \beta_G)\,. \end{aligned}$$

In the last step we have used $\sum_G (p_\epsilon \alpha_G - q_\epsilon \beta_G) = p_\epsilon - q_\epsilon$ from $\sum_G \alpha_G = \sum_G \beta_G = 1$.

If we are suppressing all error components evenly, or if we are simply removing certain error components, we will have $p_\epsilon \alpha_G \geq q_\epsilon \beta_G \quad \forall G \in \mathbb{G} - I$. In this case, the cost of implementing the transformation using quasi-probability will simply be:

$$C_{Q1,q} \approx 1 + 4\,(p_\epsilon - q_\epsilon)\,. \tag{6.17}$$

6.3.4 NISQ Limit

In Eq. (6.17) we have only been focusing on applying quasi-probability to one error channel. Assuming there are M such channels in the circuit, then using Eq. (6.17) and taking the NISQ limit (Eq. (6.1)), the sampling cost factor of transforming all M error channels with error probability p_ϵ into new error channels with error probability $q_\epsilon \leq p_\epsilon$ using quasi-probability is:

$$C_{Q,Mq} = C_{Q1,q}^M \approx \lim_{M \to \infty} (1 + 4\,(p_\epsilon - q_\epsilon))^M = e^{4M(p_\epsilon - q_\epsilon)}. \tag{6.18}$$

At $q_\epsilon = 0$, we will obtain the sampling cost of removing all the noise using quasi-probability:

$$C_{Q,0} \approx e^{4Mp_\epsilon}. \tag{6.19}$$

Remember that we are focusing on Pauli errors and p_ϵ is defined to be the total probability of all non-identity components. This is not equivalent to the error probability p because sometimes there are some identity components in our definition of error probability such as the group errors that we will discuss in

Section 8.2. Similar to the definition of p_ϵ, we can denote the expected number of *non-identity* errors in each circuit run as μ_ϵ. In the above cases, we have $\mu_\epsilon = Mp_\epsilon$ and similarly we can define $\nu_\epsilon = Mq_\epsilon$. In the cases where different noise locations experience different noise, using Le Cam's theorem with negative probabilities and focusing on the zero-occurrence case, we can generalise Eq. (6.18) to:

$$C_{Q,\nu} = e^{4(\mu_\epsilon - \nu_\epsilon)}. \tag{6.20}$$

6.4 Error Extrapolation

6.4.1 Background

The idea of error extrapolation was first introduced by Li *et al.* [48] and Temme *et al.* [49], and was later successfully realised experimentally using superconducting qubits [51].

For a circuit with the expected number of errors μ, the noise will transform our observable O into a noisy observable O_μ. To the first order approximation, the noisy expectation value $\langle O_\mu \rangle$ will be a linear function in μ for small μ. Suppose that we run the circuit repeatedly at the minimal error rate μ to obtain $\langle O_\mu \rangle$, and then we *boost* the error rate by a factor of λ and perform a further set of runs to obtain $\langle O_{\lambda\mu} \rangle$. Then an estimate of the error-free observable expectation value, denoted as $\langle O_0 \rangle$, can be obtained via linear extrapolation [48, 49]:

$$\langle O_0 \rangle = \frac{\lambda \langle O_\mu \rangle - \langle O_{\lambda\mu} \rangle}{\lambda - 1}. \tag{6.21}$$

We will use $\overline{O}_\mu$, $\overline{O}_{\lambda\mu}$ to denote the sample averages we obtain in the experiment. The sample estimate of $\langle O_0 \rangle$, denoted as $\overline{O}_0$, can be obtained using Eq. (6.21) with the sample averages in place of the expectation values. Assuming $\mathrm{Var}[O_\mu] \approx \mathrm{Var}[O_{\lambda\mu}] \approx \mathrm{Var}[O]$, following arguments in Appendix E.6.1, the sampling cost factor of two-point extrapolation is given by

$$C_E = 2\left(\left(\frac{\partial \overline{O}_0}{\partial \overline{O}_\mu}\right)^2 + \left(\frac{\partial \overline{O}_0}{\partial \overline{O}_{\lambda\mu}}\right)^2\right). \tag{6.22}$$

The factor of 2 is due to the fact that we need to sample both O_μ and $O_{\lambda\mu}$. Using Eq. (6.21) and Eq. (6.22), the cost factor for two-point linear extrapolation is:

$$C_E = 2\frac{\lambda^2+1}{(\lambda-1)^2}.$$

At $\lambda = 2$, we have $C_E = 10$.

Do note that for the above arguments to hold, we must not change the form of the error channel while we try to increase the noise strength. One way to do this is to increase the pulse length of the quantum gates [48, 49]. We can also boost the noise strength by inserting additional gates that will cancel each other out or injecting classical noise into the control parameter of parametrised gates [139].

To improve upon our estimate, we can probe at more error rates and/or extrapolate with a higher-degree polynomial using Richardson extrapolation [49] or other curve fitting algorithms.

6.4.2 NISQ Limit

Using $\mathbb{L}$ to denote the set of locations that the errors have occurred, when l errors have occurred in the circuit, our observable O will be transformed to a noisy observable $O_{|\mathbb{L}|=l}$. Recall that in the NISQ limit, the probability that l errors happen (denoted as P_l) will follow the Poisson distribution (Eq. (6.2)). Therefore, the expectation value of the observable O at a given mean circuit error count μ is then:

$$\langle O_\mu \rangle = \sum_{l=0}^{\infty} P_l \left\langle O_{|\mathbb{L}|=l} \right\rangle = e^{-\mu} \sum_{l=0}^{\infty} \frac{\mu^l}{l!} \left\langle O_{|\mathbb{L}|=l} \right\rangle . \tag{6.23}$$

Hence, how $\langle O_\mu \rangle$ changes with μ is entirely determined by how $\left\langle O_{|\mathbb{L}|=l} \right\rangle$ changes with l. When we try to fit a nth degree polynomial of μ to $\langle O_\mu \rangle$, for example performing a linear extrapolation [48, 49], we are essentially assuming that $\left\langle O_{|\mathbb{L}|=l} \right\rangle$ is a nth degree polynomial of l using the expressions of the moments of the Poisson distribution.

At $l = 0$, we have the error-free result $\langle O \rangle$:

$$\left\langle O_{|\mathbb{L}|=0} \right\rangle = \langle O \rangle .$$

At large error number l, in the case of stochastic errors, the circuit will move closer to a random circuit. Hence, for a Pauli observable O we will expect:

$$\lim_{l\to\infty} \left\langle O_{|\mathbb{L}|=l} \right\rangle = 0.$$

A generic polynomial of l will not satisfy the above boundary conditions. Hence, to align with the above boundary conditions, we can instead assume an exponential decay of $\left\langle O_{|\mathbb{L}|=l} \right\rangle$ as l increase:

$$\left\langle O_{|\mathbb{L}|=l} \right\rangle = \langle O \rangle (1-\gamma)^l$$

in which γ is the observable decay rate that satisfies $0 \leq \gamma \leq 1$. More rigorous arguments about why such an assumption should hold will be discussed in Chapter 8. This will lead to an exponential function in μ:

$$\begin{aligned} \langle O_\mu \rangle &= \langle O \rangle e^{-\mu} \sum_{l=0}^{\infty} \frac{(\mu(1-\gamma))^l}{l!} \\ &= \langle O \rangle e^{-\gamma\mu}, \end{aligned} \tag{6.24}$$

which is just the exponential extrapolation curve that was first introduced in Ref. [50]. Exponential extrapolation has been shown to outperform linear extrapolation in numerical simulations Refs. [50, 139].

A two-point exponential extrapolation can be performed using the equation:

$$\langle O_0 \rangle = \left(\frac{\langle O_\mu \rangle^\lambda}{\langle O_{\lambda\mu} \rangle} \right)^{\frac{1}{\lambda-1}}. \tag{6.25}$$

Using Eq. (6.22), the cost of two-point exponential extrapolation can be shown to be (details in Appendix E.6.2):

$$C_E \approx 2 \frac{\lambda^2 e^{2\gamma\mu} + e^{2\lambda\gamma\mu}}{(\lambda-1)^2}. \tag{6.26}$$

This concludes our introduction to quantum error mitigation. The notations we introduced and the results we derived will be used extensively in the following chapters.

7

Resource Estimation for Quantum Variational Simulations of the Fermi-Hubbard Model

This chapter is adapted from the research published in Physical Review Applied [140], of which the present author is the sole author.

Contents

7.1 Introduction

As discussed in the last chapter, with the rapid advance of quantum hardware, we are moving towards practical NISQ applications with the help of error mitigation. One of the most promising candidates for NISQ applications is the variational quantum eigensolver (VQE) due to its shallow depth. In VQE, we aim to use a parametrised quantum circuit to prepare the eigenstate (usually the ground state) of a given Hamiltonian. The task is carried out through measuring observables using the quantum circuit and optimising the parameters using a classical computer. As we will see later, one of the lowest hanging fruits in this area is the preparation of the ground state of the strongly interacting Fermi-Hubbard model. Therefore in this chapter, I will perform resource estimation for the Fermi-Hubbard VQE taking into account of error mitigation.

A large number of circuit runs are needed to run VQE [141]. Luckily, the task is highly parallelisable and thus can be distributed to many quantum processors. Both silicon spin qubits and superconducting qubits are good candidates for such a task due to their compatibility with the commercial fabrication technology [73, 142], which should reduce the cost and enhance the reliability of reproducing multiple quantum processors for the same task. Furthermore, we presently explain that the ansatz circuit of the Hubbard model that we are interested in can be naturally broken into the native gates of both silicon spin qubits and superconducting qubits. In this chapter, we will first use silicon spin qubits as our example platform to carry out our gate count and runtime analysis and then we will apply the same analysis to the superconducting qubits.

The chapter is structured as follows, we begin by introducing the concept of VQE and why we choose to simulate the Hubbard model in Section 7.2. Then we will outline the ansatz circuit we use and its gate counts for the silicon platform in Section 7.3. To deal with the noise in the circuit, we will apply error mitigation to our example circuit in Section 7.4. Following that, in Section 7.5 we will introduce more details about our implementation and the estimated algorithm runtime on the silicon

platform. In Section 7.6, we will apply the same analysis to the superconducting platform. This is followed by our conclusion in Section 7.7.

7.2 Variational Quantum Eigensolver

7.2.1 Background

For a given Hamiltonian H, we want to find a circuit that can prepare its ground state $|\psi_0\rangle$ with the associated ground state energy E_0. We try to achieve this with a circuit with tunable gates that have M parameters $\vec{\theta} = \{\theta_1, \theta_2, \cdots, \theta_M\}$ that we can control, which will produce an output state $\left|\psi(\vec{\theta})\right\rangle$. In VQE, our goal is to obtain the set of optimal parameters $\vec{\theta}_{op}$ such that the state prepared by the circuit can well approximate the ground state we want $\left|\psi(\vec{\theta}_{op})\right\rangle \approx |\psi_0\rangle$. Then we can measure various properties of the ground state using $\left|\psi(\vec{\theta}_{op})\right\rangle$. Refs [143] and [144] provide a comprehensive overview on variational algorithms and more generally the field of quantum computational chemistry.

7.2.2 Choosing the Simulation Problem and the Ansatz

The parametrised circuit in VQE is called an ansatz circuit, or simply ansatz for short. It determines the quantum subspace that our output state can reach, hence it is the key to the success of our algorithm. An example of a simple ansatz is called the hardware-efficient ansatz (HEA) [145], which just uses gates available to the physical qubit systems to create many entangling blocks along the circuit. However, it is difficult to obtain an analytical or even a heuristic estimate of the number of gates required in an HEA for the success of our algorithm. A better approach will be using an ansatz inspired by the problem itself. For example, the unitary-coupled-cluster ansatz (UCCA) for quantum chemistry [146], the low depth circuit ansatz (LDCA) [147] and the Hamiltonian ansatz (HA) [141] for more general simulations of closed quantum systems.

The number of gates needed by HA scales as $\mathcal{O}(N_{blk}R)$ where R is the number of subterms in the Hamiltonian of our simulation and N_{blk} is the number of repeating blocks within the ansatz. For a general electronic structure Hamiltonian (which

applies to most chemistry Hamiltonians), we have $R \sim \mathcal{O}(N^4)$ due to the terms accounting for electron-electron interactions [141]. For periodic systems, the added structure allows us to transform our basis orbitals to achieve $R \sim \mathcal{O}(N^2)$ [148]. One of the systems that has the most favourable scaling while maintaining great physical interest is the 2D Hubbard model, in which by restricting to only on-site interactions and nearest-neighbour hopping, we can achieve $R \sim \mathcal{O}(N)$. Using HA to simulate the Hubbard model, the gates we apply are dependent on the qubit encoding we use. Using the Jordan-Wigner encoding, some of the gates will be non-local, which can be overcome by using $\mathcal{O}(N^{\frac{1}{2}})$ additional gates [149]. Other encodings like Verstraete-Cirac encoding [150] or superfast encoding [151] will ensure locality, but require at least doubling the number of qubits. Since we are focusing on near-term devices in which qubit resources can be limited, we will only consider the Jordan-Wigner encoding in this chapter. In such case, the number of gates needed to simulate the 2D Hubbard Model using HA scales as $N_{gates} \sim \mathcal{O}(N_{blk}N^{\frac{3}{2}})$. This is a factor of $N^{\frac{1}{2}}$ better than LDCA assuming the same number of blocks N_{blk} in both ansatze. To achieve results of sufficient precision, we likely need to go up to double excitation for UCCA, which means a gate scaling of $N_{gates} \sim \mathcal{O}(N^5)$ (assuming Jordan-Wigner encoding, see Ref. [144]). To achieve a similar precision, N_{blk} for HA is likely to scale better than $\mathcal{O}(N^{\frac{7}{2}})$ based on the previous Hubbard model numerical simulation results [141, 152, 153], thus HA should have a scaling advantage over UCCA in simulating the Hubbard model.

Hence, in the rest of this chapter, we will be focusing on the ground state preparation of the Hubbard model using Hamiltonian Ansatz (HA) under the Jordan-Wigner encoding.

Note that for the NISQ regime that we are interested in, the constants we ignored in the above scaling analysis can make a very significant difference. Thus there might be cases that a different ansatz implementation that are more practical than the one we are going to consider when we dig into the exact implementation details.

7.3 Ansatz Circuit

7.3.1 2D Fermi-Hubbard Model

The Hamiltonian of the 2D Fermi-Hubbard model is:

$$H = -t \sum_{\sigma,\langle v,w\rangle} \left(a^\dagger_{v,\sigma} a_{w,\sigma} + a^\dagger_{w,\sigma} a_{v,\sigma}\right) + U \sum_v n_{v,\uparrow} n_{v,\downarrow}$$

where $a^\dagger_{v,\sigma}/a_{v,\sigma}$ are the creation/annihilation operators of site v with spin σ. The first term represents the nearest neighbour hopping interactions, with t being the tunnelling energy and $\langle v, w\rangle$ representing summing up sites v and w that are adjacent to each other in a 2D geometry. The second term is the on-site repulsion energy. There are two spin orbitals to each site, hence the total number of orbitals (N) needed is twice the number of sites (V).

Extracting the ground state properties of the Hubbard model in the parameter regime of intermediate-coupling (e.g. $\frac{U}{t} \approx 4 \to 8$) at close to half-filling is believed to be relevant to the understanding of high-T_c cuprate superconductors [154, 155]. If we look at the 5×5 Hubbard model, which means $V = 25$ and $N = 50$, the dimension of the Hilbert space will reach 2^{50}, and thus can not be solved via classical exact diagonalisation. There are various numerical approximation methods available, however at the intermediate-coupling near-half-filled regime for the Hubbard model, there is no theoretical guarantee that these methods will converge at the correct results and numerically the results from these methods were shown to have large uncertainties in this parameter regime [154]. Hence, in this chapter, we will turn to quantum algorithm instead to solve for the ground state of the 5×5 Hubbard model in this classically challenging parameter regime. For simplicity we will be considering the half-filling case, i.e. for V sites we have V electrons, but most of our arguments can be generalised to any number of electrons, and the resultant resource estimate will be similar for all near-half-filling cases.

The Fermion-to-qubit mapping that we will employ in this chapter is the Jordan-Wigner encoding, in which the number of qubits required is equal to the

number of orbitals N. In the Jordan-Wigner encoding, the Fermionic creation and annihilation operators become:

$$a_i^\dagger \mapsto Z_{:i} A_i^\dagger, \quad a_i \mapsto Z_{:i} A_i$$

where i denotes the index of the canonical ordering of the orbitals (including both spins and sites). $A = |0\rangle\langle 1| = \frac{X+iY}{2}$ is the qubit lowering operator and $A^\dagger$ is the qubit raising operator. $Z_{:i} = \prod_{j=1}^{i-1} Z_j$ is the Z operator string that maintains the Fermionic commutation relationship.

The on-site repulsion term and the hopping term of the Hubbard Hamiltonian thus becomes:

$$\begin{aligned} n_i n_j = a_i^\dagger a_i a_j^\dagger a_j &\mapsto \frac{1}{4}(I - Z_i)(I - Z_j) \\ a_i^\dagger a_j + a_j^\dagger a_i &\mapsto \frac{1}{2}\left(X_i X_j + Y_i Y_j\right) Z_{i:j} \end{aligned} \tag{7.1}$$

where we have without loss of generality assumed $j > i$ and defined $Z_{i:j} = \prod_{k=i+1}^{j-1} Z_k$.

7.3.2 Hamiltonian Ansatz

The Hamiltonian ansatz was proposed by Wecker *et al.* [141]. Its circuit is essentially a Trotterised variational annealing path, which is inspired by both adiabatic state preparation and the quantum approximate optimisation algorithm [156, 157].

For a Hamiltonian of the form:

$$H = \sum_i \lambda_i h_i$$

where h_i are different interaction terms, a block of the Hamiltonian ansatz is just:

$$\prod_i e^{-i\theta_i h_i}$$

i.e. we implement a parametrised unitary gate corresponding to every interaction term. We will implement N_{blk} of such blocks in sequence with different sets of parameters. The Hamiltonian ansatz, though inspired by Trotterised annealing, is not equivalent to Trotterisation. Thus a Hamiltonian ansatz using higher-order Trotterisation does not necessarily have lower algorithmic errors than a Hamiltonian

ansatz using lower-order Trotterisation. In NISQ devices, we might want to avoid using higher-order Trotterisation in the Hamiltonian ansatz due to higher gate counts and deeper circuits. Hence, here we have used the first-order Trotter formula for the Hamiltonian ansatz.

From Eq. (7.1), we see that the gates corresponding to the repulsion terms are all local, while the gates that correspond to hopping will only be local if the two orbitals involved are close to each other in the canonical ordering due to the trailing Z string.

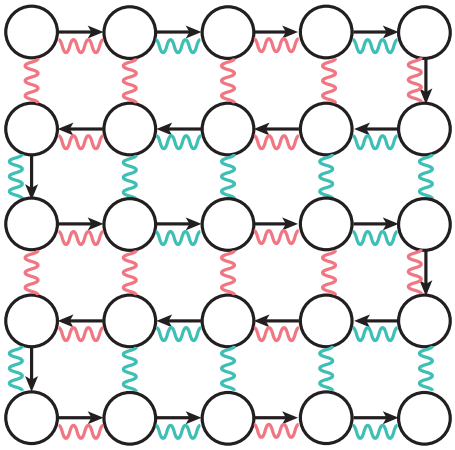


Figure 7.1: The black arrows denote the canonical order of the orbitals of different sites for a 5×5 Hubbard model. Here the ordering is in a snake-fold pattern running along the horizontal direction. The red/blue curly lines denote the even/odd hopping interaction.

Now if we choose a canonical ordering such that the two spin orbitals of the same site are always adjacent to each other, while the orbitals of different sites are ordered in a folding pattern running along the horizontal direction as shown in Fig. 7.1, then all the horizontal hopping terms will be local while some of the vertical hopping terms will not be. The non-local vertical hopping can be made local by using additional Fermionic swap (fSWAP) gates to swap the canonical order of the orbitals [158]:

$$f_{\text{swap}}^{i,i+1} \mapsto \frac{1}{2}\left[(X_i X_{i+1} + Y_i Y_{i+1}) + Z_i + Z_{i+1}\right].$$

Kivlichan *et al.* [149] outline a way to implement a block of the Hamiltonian ansatz for the open boundary 2D Hubbard model using only local gates with $\mathcal{O}(N^{\frac{1}{2}})$ depth using a Fermionic swap network. The exact structure of this ansatz is recapped in Appendix D.1.1. In Appendix D.1, we have decomposed the gates

corresponding to the repulsion terms, the hopping terms and the Fermionic swap into the native gates of silicon spin qubits, and found that we only need single-qubit Z rotation and partial swaps. Then we applied the gate decompositions to the Hamiltonian ansatz, optimised the circuit using the qubit exchange symmetry of the gates, and obtained the number of single-qubit gates and two-qubit gates needed for one block of Hamiltonian ansatz for V sites to be:

$$N_{1q,ha} = 4V^{\frac{3}{2}} + 7V - 4\sqrt{V}$$
$$N_{2q,ha} = 8V^{\frac{3}{2}} + V - 4\sqrt{V}.$$

For $V = 25$, we have

$$N_{1q,ha} \approx 650, \quad N_{2q,ha} \approx 1000.$$

7.3.3 Full Ansatz Circuit

To produce a good approximation to the Hubbard ground state, we need a good starting state for the Hamiltonian ansatz. In the context of adiabatic evolution, a good starting state should be in the same phase as the output state as discussed by Wecker *et al.* [155]. Since the Hamiltonian ansatz evolves out of adiabatic evolution, we should expect a similar result to hold. The starting state we use should be a single Slater determinant, which can be solved classically and can be efficiently prepared using a quantum circuit [149, 159], e.g. the ground state of the non-interacting Hubbard Hamiltonian (i.e. $U = 0$). In Appendix D.2 we have recapped the details about the Slater determinant preparation circuit using Givens rotation. We decomposed the Givens rotation into the native gates of silicon spin qubits and again found that we only need Z rotation and partial swaps. Then we applied the gate decompositions to the Slater determinant preparation circuit, optimised the circuit and obtained the number of single-qubit gates and two-qubit gates needed for V sites to be:

$$N_{1q,prep} = 2V^2, \quad N_{2q,prep} = 2V^2.$$

When the two spin subspaces of the starting state are decoupled (e.g. the non-interacting Hubbard ground state), we can prepare the Slater determinant separately in the two spin subspaces, which is also discussed in Appendix D.2. In this case, we need to start in a orbital ordering in which the spin-up and spin-down are separated, and end in a orbital ordering in which the spin-up and spin-down are interleaved for inputting into the Hamiltonian ansatz. We found that the saving in gate counts is limited due to the need to rearrange the orbital ordering while the runtime needed is longer. Hence, here we will stick with the simple Slater determinant preparation scheme in which we do not consider the two spin subspaces separately.

The total number of gates needed for the whole ansatz circuit on the silicon-spin-qubit platform is:

$$\begin{aligned} N_{1q} &= \underbrace{2V^2}_{N_{1q,prep}} + \underbrace{\left(4V^{\frac{3}{2}} + 7V - 4\sqrt{V}\right)}_{N_{1q,ha}} N_{blk} \\ N_{2q} &= \underbrace{2V^2}_{N_{2q,prep}} + \underbrace{\left(8V^{\frac{3}{2}} + V - 4\sqrt{V}\right)}_{N_{2q,ha}} N_{blk}. \end{aligned}$$

Hence, if N_{blk} scales better than $\mathcal{O}(\sqrt{V})$, then the Slater determinant preparation will dominate the gate count at large V, otherwise the Hamiltonian ansatz will dominate the gate count at large V. In the previous simulation of the Hubbard model in a ladder grid structure with periodic boundaries [141], N_{blk} scales super-linear to V, while Ref [152, 153] also shows similar results for the open-boundary Hubbard model. Hence, we will expect the gate cost due to the Slater determinant preparation part of the circuit to be negligible at large V.

If we take the optimistic assumption of $N_{blk} \sim V$, then the number of gates needed in the ansatz circuit for $V = 25$ on the silicon-spin-qubit platform will be:

$$N_{1q} \approx 17000, \quad N_{2q} \approx 26000. \tag{7.2}$$

Across all major qubit platforms, the error rate of the single-qubit gates is generally much lower than that of the two-qubit gates. If we assume an optimistic two-qubit

gate error rate of around 10^{-4} and a negligible single-qubit gate error rate, we can achieve the mean circuit error count:

$$\mu = 26000 \times 10^{-4} \sim 2.5, \tag{7.3}$$

which is still of the order of unity. Hence, to obtain a meaningful result out of our noisy circuit, we must apply error mitigation.

7.4 Error Mitigation

VQE is inherently robust against local systematic errors since they can be offset by shifts in the variational parameters [160], which was observed in experiments [161]. For the other error components, we can further mitigate them via symmetry verification.

7.4.1 Symmetry Verification in Hubbard Model Simulation

The basic ideas of symmetry verification have been introduced in Section 6.2. In the case of the preparing the Hubbard model ground state, we can verify the electron number parity symmetry of the output state:

$$S_\sigma = \prod_i (1 - 2n_i)\,.$$

Here n_i is the number operator of the ith orbital. In the Jordan-Wigner encoding, it simply maps to the product of all Z operators:

$$S_\sigma = \prod_i Z_i.$$

As shown in Appendix D.6, in the silicon platform, we can measure both the energy terms and the symmetry S_σ in the same circuit runs, enabling us to perform direct verification.

In fact, since our ansatz circuit also conserves the electron number within each spin subspace, by using a starting state with the right number of spin-up and spin-down electrons, we can actually verify the electron number parity symmetry within each spin subspace, which we will denote as $S_\uparrow$ and $S_\downarrow$.

The ansatz circuit can be decomposed into two-qubit components that represent different interaction terms, fSWAPs and Given rotations as listed in Appendix D.1.2. We will assume there are M of these components in the circuit, all are affected by depolarising channels with error probability p. Thus the expected number of errors occurring in each circuit run is:

$$\mu = Mp. \tag{7.4}$$

In between the two-qubit components where the errors occur, our state will have well-defined $S_\uparrow$ and $S_\downarrow$ (though the set of qubits that correspond to each spin subspace may be different from the final state due to the use of fSWAP). Hence, if a Pauli error occurs here and anti-commutes with $S_\uparrow$, it will flip the $S_\uparrow$ eigenvalue, leading to a failed $S_\uparrow$ test at the end. Since $S_\uparrow$ is the product of Z operators in the spin-up subspace, detectable errors will be the Pauli errors whose total weights in X and Y are odd. Similar arguments also apply to $S_\downarrow$ in the spin-down subspace.

There are two types of two-qubit components in the circuit, for which we will make two different approximations to their error channels:

- The components acting within *one* spin subspace: Within the 16 two-qubit Pauli error components (including the identity), we can detect the errors with *one* X or Y in it, which is half of them. Hence, for these two-qubit components, their depolarising errors can be approximated by the composition of a detectable error channel and an undetectable error channel, both with the error probability $\frac{p}{2}$.

- The components acting across *both* spin subspaces: Within the 16 two-qubit Pauli error components (including the identity), 4 are undetectable, 4 are detectable only by $S_\uparrow$, 4 are detectable only by $S_\downarrow$ and 4 are detectable by both. For simplicity, we will absorb the last case into the other two detectable cases and approximate the depolarising channel by the composition of three error channels:

    1. An undetectable error channel with the error probability $\frac{p}{4}$.

2. An error channel detectable by the $S_\uparrow$ with the error probability $\frac{3p}{8}$.

3. An error channel detectable by the $S_\downarrow$ with the error probability $\frac{3p}{8}$.

Our circuit consists of alternating layers of these two types of two-qubit components, thus there are approximately equal numbers of components for each type (i.e. $\frac{M}{2}$). Hence, when we focus only on the spin-up subspace, there are around $\frac{M}{4}$ components acting within the subspace with detectable error probability $\frac{p}{2}$ and around $\frac{M}{2}$ components acting across both spin subspaces with detectable error probability $\frac{3p}{8}$, which gives a total detectable mean circuit error count of

$$\mu_d = \frac{M}{4}\frac{p}{2} + \frac{M}{2}\frac{3p}{8} = \frac{5Mp}{16} = \frac{5\mu}{16} \tag{7.5}$$

The exact same arguments can be applied to the spin-down subspace which gives the same equation.

Using Eq. (6.8), in the large circuit limit the fraction of circuit runs that pass the verification of $S_\uparrow/S_\downarrow$ is:

$$P_{S,one} = \frac{1 + e^{-2\mu_d}}{2}.$$

Since in our error channel approximations, events of $S_\uparrow$ violation and $S_\downarrow$ violation are independent of each other, we can obtain the fraction of circuit runs that pass the verification of both $S_\uparrow$ and $S_\downarrow$ as:

$$P_S = P_{S,one}^2 = \frac{1}{4}\left(1 + e^{-2\mu_d}\right)^2.$$

Hence, using Eq. (6.4) and Eq. (7.5), the sampling cost factor of applying symmetry verification using $S_\uparrow$ and $S_\downarrow$ is

$$C_S \sim \frac{1}{P_S} = 4\left(1 + e^{-2\mu_d}\right)^{-2} = 4\left(1 + e^{-\frac{5\mu}{8}}\right)^{-2}. \tag{7.6}$$

When focusing on one of the symmetries, using Eq. (6.11) the expected number of circuit errors detectable by $S_\uparrow/S_\downarrow$ after symmetry verification is:

$$\mu_{S,one} = \mu_d \tanh(\mu_d).$$

Since the error events of $S_\uparrow$ violation and $S_\downarrow$ violation are independent of each other, we can obtain the resultant mean circuit error rate after applying symmetry verification using both $S_\uparrow$ and $S_\downarrow$:

$$\mu_S = \mu - 2\mu_d + 2\mu_{S,one} = \frac{3\mu}{8} + \frac{5\mu}{8}\tanh\left(\frac{5\mu}{16}\right) \tag{7.7}$$

where we have made use of Eq. (7.5). Hence, the mean circuit error count is reduced by a fraction of:

$$\frac{\mu - \mu_S}{\mu} = \frac{5}{8}\left(1 - \tanh\left(\frac{5\mu}{16}\right)\right) \tag{7.8}$$

after symmetry verification.

Note that the key quantity in our error mitigation analysis is the mean circuit error count $\mu = Mp$, not the gate error rate p. Hence, even though we are discussing in terms of two-qubit components here instead of elementary gates, as long as we are using a realistic μ, all of our estimates will be very similar to what we can obtain by considering elementary gates.

As shown in Eq. (7.3) and later in Eq. (7.18), we are interested in the cases in which the original mean circuit error count is around 2:

$$\mu \sim 2.$$

Substituting into Eq. (7.6), we can obtain the symmetry verification cost factor:

$$C_S \approx 2.4. \tag{7.9}$$

After applying symmetry verification using both $S_\uparrow$ and $S_\downarrow$, we should expect the mean circuit error count reduce by around 30% using Eq. (7.8).

7.4.2 Combining with Error Extrapolation

We can try to remove the remaining errors after symmetry verification using error extrapolation. To combine symmetry verification with error extrapolation, we simply apply symmetry verification to the two observables in Eq. (6.21):

$$\langle O_{SE} \rangle = \frac{\lambda \langle O_{S,\mu} \rangle - \langle O_{S,\lambda\mu} \rangle}{\lambda - 1} \tag{7.10}$$

here O_{SE} is the error-mitigated observable after both symmetry verification and error extrapolation while $O_{S,\mu}$ is the symmetry-verified observable at mean circuit error count μ.

Using $\overline{\ }$ to denote the sample averages, then using Eq. (7.10) and assuming $\mathrm{Var}\left[\overline{O}_{\mu}\right] \approx \mathrm{Var}\left[\overline{O}_{\lambda\mu}\right] \approx \mathrm{Var}\left[\overline{O}\right]$, we have:

$$\begin{aligned}\mathrm{Var}\left[\overline{O}_{SE}\right] &= \frac{\lambda^2 \mathrm{Var}\left[\overline{O}_{S,\mu}\right] + \mathrm{Var}\left[\overline{O}_{S,\lambda\mu}\right]}{(\lambda-1)^2} \\ &= \frac{\lambda^2 C_{S,1}\mathrm{Var}\left[\overline{O}_{\mu}\right] + C_{S,\lambda}\mathrm{Var}\left[\overline{O}_{\lambda\mu}\right]}{(\lambda-1)^2} \\ &= \frac{\lambda^2 C_{S,1} + C_{S,\lambda}}{(\lambda-1)^2}\mathrm{Var}\left[\overline{O}\right]\end{aligned}$$

where $C_{S,\lambda}$ is the cost of symmetry verification with the noise boosted by the factor λ and can be obtained using Eq. (7.6):

$$C_{S,\lambda} = 4\left(1 + e^{-\frac{5\lambda\mu}{8}}\right)^{-2}.$$

To keep the $\mathrm{Var}\left[\overline{O}_{SE}\right]$ at a target precision level, we thus need C_{SE} times more circuit runs than keeping $\mathrm{Var}\left[\overline{O}\right]$ at the same precision level, in which

$$\begin{aligned}C_{SE} &= 2\frac{\lambda^2 C_{S,1} + C_{S,\lambda}}{(\lambda-1)^2} \\ &= 8\frac{\lambda^2\left(1 + e^{-\frac{5\mu}{8}}\right)^{-2} + \left(1 + e^{-\frac{5\lambda\mu}{8}}\right)^{-2}}{(\lambda-1)^2}.\end{aligned}$$

The factor of 2 is to account for the samples of both O_{μ} and $O_{\lambda\mu}$.

For $\mu = 2$ and $\lambda = 2$, we have:

$$C_{SE} \sim 25. \tag{7.11}$$

i.e. if we want to apply symmetry verification and two-point linear extrapolation with $\lambda = 2$ to our example, we need 25 times more circuit runs to obtain an estimated observable than sampling from the circuit directly. The combination of symmetry verification and error extrapolation was proven to be very effective in numerical simulations [65]. In our example, we managed to suppress the original mean circuit error count by almost 30% using symmetry verification. Together

with the robustness of the variational circuits against local systematic errors, they should enable efficient applications of error extrapolation, and thus allowing us to extract meaningful results out of our noisy quantum circuit.

7.5 VQE Implementation

7.5.1 Parametrising the Hamiltonian Ansatz

We have outlined the structure of the ansatz circuit in Section 7.3, now we will turn to the way we parametrise the gates in the circuit. In the simplest scheme, we can assign a different parameter to each parametrised gate to allow an unconstrained optimisation of the ansatz circuit. However, a large number of parameters will mean a large number of dimensions in the parameter space. This can lead to difficulties in optimisation and may lead to long runtime since we need to probe more directions to obtain the gradient vector. Hence, here we will try to reduce the number of parameters by using the symmetry of the site layout.

For the open-boundary Hubbard model, the site layout has mirror symmetries along the horizontal and vertical direction. Thus the 2D Hubbard grid can be sliced into four equivalent partitions: $N_{eq} = 4$. On top of that if we have the same number of rows and columns, then the site layout also has an additional diagonal mirror symmetry, which gives $N_{eq} = 8$. The ground state of the Hubbard model is expected to follow the same symmetry.

The input Slater determinant should follows the same layout symmetry, hence so does the ansatz parametrisation. That is, the parametrised gates that represent the corresponding interaction terms in different equivalent partitions, which can be mapped to each other via layout-symmetry transformations, will share the same parameters.

Hence, the number of parameters in our ansatz is:

$$N_{para} \approx \frac{N_{para}^{site} V}{N_{eq}} N_{blk}. \tag{7.12}$$

where N_{para}^{site} is the number of parameters per site.

Ignoring the boundary case, there will be 5 interaction terms associated with each site: the repulsion term and the horizontal and vertical hopping terms of the two spins. The input Slater determinant and the output ground state must have the same spin symmetry since our ansatz preserves spins. Hence, the ansatz parametrisation will also have the same spin symmetry. Without spin-flip symmetry, the parameters for the spin-up and spin-down hopping terms will be different, thus 5 interaction terms means 5 parameters: $N_{para}^{site} = 5$. With spin-flip symmetry, the gates for the spin-up and spin-down hopping terms can share the same set of parameters, thus $N_{para}^{site} = 3$. If we are considering the half-filling ground state with the smallest total spin, then:

- Odd number of sites (electrons): different numbers of spin-up and spin-down electrons, which means no spin-flip symmetry and $N_{para}^{site} = 5$.

- Even number of sites (electrons): same number of spin-up and spin-down electrons, which means spin-flip symmetry and $N_{para}^{site} = 3$.

For the Hubbard model with periodic boundaries along the horizontal direction, we have translational symmetry of the sites along the horizontal direction, which means that all columns are equivalent on top of the vertical mirror symmetry we have, giving $N_{eq} = 2N_{col}$. This can lead to fewer parameters. However, as we will discuss in Section 7.5.2, the vertical interaction terms cannot be measured locally and efficiently in this case, thus we will not consider such a boundary condition.

For the Hubbard model with periodic boundaries in both directions, we have complete translational symmetry and thus every site is equivalent. However, we will also not consider this case since there is not yet an efficient Hamiltonian ansatz circuit for the periodic Hubbard model with a gate count and scaling as favourable as the one we have adopted for the open-boundary case.

7.5.2 Measurement

The measurement of energy is carried out by measuring the individual Pauli components of the Hamiltonian. The Pauli components within an individual repulsion term or a hopping term commute with each other. Thus with the availability of non-demolishing measurements, ideally we should be able to measure all the commuting interaction terms at one go. For the Hubbard model, there are five commuting subsets as shown in Fig. 7.1: repulsion terms, even horizontal hopping terms, odd horizontal hopping terms, even vertical hopping terms and odd vertical hopping terms. Thus we should be able to obtain one sample of each interaction term in five circuit runs. However, direct measurements of the vertical hopping terms can be costly due to their non-locality. These non-local terms can be broken down into non-local Pauli observables, which in turn can be obtained by performing local Pauli measurements and multiplying the results. However, such local measurements can break the commutativity of the vertical hopping terms such that we cannot measure them in parallel.

For the case of the open-boundary Hubbard model, we can tackle this by switching the canonical ordering of orbitals from running horizontally in the 2D grid to running vertically when we try to measure the vertical hopping terms. In such a way, just like the horizontal hopping terms are local in the horizontal-running canonical order, the vertical hopping terms will also be local in the vertical-running canonical order. Note that switching the orbital canonical order will require us to switch the ansatz accordingly, but the same parameters will be used for the parametrised gates that correspond to the same interaction terms. In such a way, we can still obtain one sample of all interactions terms in five circuit runs. The same measurement scheme can be used to measure the energy gradients since it involves the same Pauli measurements with some small modifications to the ansatz circuit (see Appendix D.4.2).

Now on top of obtaining the energy or energy gradient by measuring the Pauli components G_j of the Hubbard Hamiltonian, we also want to apply direct symmetry verification outlined in Section 6.2 by measuring the symmetry operator S in the

same run. However, it is often the case that S is not local, thus to obtain S we need to rely on local measurements and post-processing.

In our example, there is no need to measure the electron number parity symmetry, it can be obtained by composing the measurement results of the interaction terms. When measuring the repulsion terms, we will be performing single-qubit Z measurement for every qubit. The repulsion terms and the electron number parity can both be obtained via post-processing. In the case of measuring the hopping terms, we will be measuring XX and YY for the hopping pairs adjacent to each other in the canonical order. We can obtain the results of ZZ measurements by composing XX and YY (with an additional $-$ sign), composing with the Z measurements of the qubits not included in the hopping pairs, we can then obtain the electron number parity via post-processing.

It is worth noting that the Hubbard model simulation is quite friendly for efficient local measurements of the Hamiltonian terms. For more general problems, one might need to turn to more sophisticated measurement schemes [162–164].

In the above scheme, we have assumed we can carry out non-demolishing measurements, which can be carried out in silicon using ancilla qubits. In Appendix D.6, we outline a possible quantum dot layout that enables an efficient implementation of our measurement scheme.

7.5.3 Optimisation Method

We need to employ classical optimisation algorithms to obtain the optimal set of parameters for our parametrised quantum circuit. There are two general approaches, direct search and gradient-based. Direct search involves evaluating the cost function at different points and choosing the next set of points to evaluate based on the known points of the cost function, while gradient-based methods make use of the gradient of the cost function. In our case, since we are searching for the ground state, the cost function that we want to minimise is the energy of the state produced by our quantum circuit.

The energy can be straightforwardly evaluated by measuring the Pauli components of the Hamiltonian as mentioned in Section 7.5.2. As discussed in Ref. [165], to compete with the best classical algorithm for Hubbard model simulation, we need to estimate the energy per site to the precision $\epsilon_{E,site} = 10^{-3}t$. In Appendix D.5, we have translated this precision requirement into the precision requirement on the estimates of each Hamiltonian Pauli term. For the 5×5 Hubbard model, the number of circuit runs needed to estimate the energy to the required precision is:

$$M_E \approx 4 \times 10^5. \tag{7.13}$$

The gradient vector can be evaluated using finite difference, which involves evaluating the energy at two neighbouring points. In the simplest gradient descent scheme, by evaluating the energy points used in finite difference to the precision $\epsilon_{E,site}$ and also choosing the terminating threshold of the change in energy to be $\epsilon_{E,site}$, we will be able to find the local energy minimum with the required precision $\epsilon_{E,site}$ given the right gradient-descent step size. In such a case, we will need twice the number of measurements compared to energy estimation to evaluate the gradient in one direction (since we need to evaluate two energy points). We have N_{para} directions to probe, thus the number of circuit runs needed to evaluate the full energy gradient vector using finite difference is:

$$M_{grad}^{fd} = 2N_{para}M_E. \tag{7.14}$$

Using Eq. (7.12) and assuming $N_{blk} \sim V$, for 5×5 Hubbard model, the number of circuit runs needed to estimate the full gradient vector using finite difference is:

$$M_{grad}^{fd} = 2N_{para}M_E \approx 3.1 \times 10^8.$$

The precision of the gradient vector obtained here is dependent on the finite difference step size we choose. In Appendix D.5, we have outlined how to obtain the optimal step size and the gradient precision ϵ_{grad} that we can achieve using this optimal step size.

The gradient vector can also be evaluated using direct measurements of a modified ansatz circuit. The two approaches to obtain the gradient were compared

in Refs. [146, 166], in which they explain that more measurements are needed in finite difference to overcome the finite step size approximation it makes. We further compare them for our implementation in Appendix D.5, taking into account the fact that we have the many parametrised gates share the same parameters due to the symmetries in the site layout. We found that direct measurements require fewer samples compared to finite difference as the number of gates with shared parameters increases. In Appendix D.5, we explain that to achieve the same gradient precision ϵ_{grad} achieved above using finite difference (with the optimal step size), the number of circuit runs required using direct measurements is:

$$M_{grad} \approx 2.5 \times 10^7 \tag{7.15}$$

which is an order of magnitude better than finite difference in this case.

Direct search methods are generally more effective in noisy and non-smooth problems while for gradient-based methods, the number of function calls needed usually scales better in higher dimensional problems [167]. We have also proven above that evaluating the full gradient vector is usually much more costly than evaluating an energy point in our implementation. We can see that neither of the approaches are clearly preferred. Various direct search optimisation methods like Nelder-Mead simplex have been successfully implemented experimentally [168–174] for small-size problems due to the robustness of direct search against noise, while gradient-based method like SPSA, which requires a smaller number of samples than simple gradient descent due to its stochastic nature, have also found success in the simulations of small molecules [51, 145, 175]. With improvements in the quantum hardware noise rate, we will expect gradient-based methods to play a more and more important role in the experimental realisation of VQE, especially considering the success of advanced gradient-based methods like Adam and Adagrad in high dimension noisy optimisation problems in classical machine learning [176, 177]. There are also investigations into using machine learning for optimisation [178, 179], which might have faster convergence rate and higher robustness to noise. In the end, the optimisation scheme is likely to involve a combination of various

methods, with the aid of techniques like block-by-block optimisation [141] and sequential optimisation [180].

Cade *et al.* [153] have performed numerical simulation of the Hubbard VQE using SPSA optimisation. They followed a three-stage protocol with coarser gradient precision at the beginning and moving to finer and finer gradient precision as the optimisation progress. In the end they evaluated the gradient using finite-difference with the number of circuit runs for each energy estimation being $M_E \sim 5 \times 10^4$ (this is different from our result above due to different precision requirements). Their 3×3 Hubbard simulations converge when the number of circuit runs is around $M_{tot} \sim 2 \times 10^8$. There are 30 parameters in their circuit, using Eq. (7.14), we can translate the result of their simulation into the language of simple gradient descent, the number of 'effective' gradient descent iteration in their simulation is just:

$$n_{iter} = \frac{M_{tot}}{M_{grad}^{fd}} = \frac{M_{tot}}{2N_{para}M_E} = 67.$$

As mentioned above, the gradient-based method has been shown to scale extremely well with increased problem dimensions in practice. Thus, we can expect the number of 'effective' gradient descent iteration needed for 5×5 Hubbard simulation will also be in the region of

$$n_{iter} \sim 100. \tag{7.16}$$

7.5.4 Algorithm Runtime for Silicon Spin Qubits

Here we will estimate the algorithm runtime needed for running the VQE for the 5×5 open-boundary Fermi-Hubbard model.

From Appendix D.1 and D.2, we know the runtime T_{circ} needed for the ansatz circuit with Slater determinant preparation is:

$$T_{circ} \approx (49 + 45N_{blk})\,\tau_{1q} + (196 + 80N_{blk})\,\tau_{2q} + \tau_{in} + \tau_m$$

where τ_{1q} and τ_{2q} are the typical time needed to perform a $\frac{\pi}{2}$ rotation for Z rotation and partial swap respectively, and τ_{in}, τ_m are the time required for qubit initialisation and measurements.

In silicon quantum dot spin qubits, the Z rotations can be implemented using the Stark shift at the speed $\tau_{1q} \sim 0.1\mu s$ [137]. Partial swaps can be implemented using exchange interaction at sub-ns scale [102]. Here instead we will assume a two-qubit-gate time of tens of ns to prevent the gate fidelity being limited by the finite voltage rise time [181]. For readout, a scheme that can achieve more than 98% fidelity in under $6\mu s$ has been demonstrated [135], and a sub-μs scheme with 99.7% fidelity has been proposed [182]. For initialisation, the simplest way is via spin relaxation, which will be on the ms timescale. Faster initialisation can be achieved via spin-selective tunnelling from charge reservoirs [123] or electron shuttling and 'hotspot reset' [125, 183–185]. Initialisations at the μs scale have been achieved in silicon donor qubits [186] and other semiconductor quantum dot qubits [187]. Thus here we will assume the time needed for initialisation plus readout can be reduced to below $100\mu s$. Hence, for the 5×5 Hubbard model with $N_{blk} = V = 25$, the runtime needed for each circuit run is around

$$T_{circ} \sim 250\mu s.$$

In Section 7.5.3, we have obtained the number of circuit runs needed for estimating the energy and the energy gradient vector. However, as mentioned in Section 7.4, due to the high mean circuit error count, we need to apply error mitigation. The sampling cost of applying both direct symmetry verification and linear error extrapolation is $C_{SE} \sim 25$ as shown in Eq. (7.11). Thus the number of circuit runs needed to estimate the energy and the energy gradient vector with error mitigation is:

$$\begin{aligned} M_E^* &= C_{SE} M_E \approx 1 \times 10^7 \\ M_{grad}^* &= C_{SE} M_{grad} \approx 6 \times 10^8. \end{aligned} \tag{7.17}$$

Thus the time needed to evaluate the error-mitigated energy and energy gradient is:

$$\begin{aligned} T_E &= T_{circ} M_E^* = 2500 \text{ s} \\ T_{grad} &= T_{circ} M_{grad}^* = 1.5 \times 10^5 \text{ s} \approx 1.7 \text{ days}. \end{aligned}$$

Using the simplest optimisation scheme, gradient descent, each iteration step then involves the evaluation of one gradient vector, which requires 1.7 days. Such a long duration per iteration is hardly practical. However, the time cost is mostly due to the large number of samples needed, thus can be easily solved by running many circuits in parallel in multiple quantum processors. With 200 processors, the time required for each gradient-descent iteration is reduced to around 10 minutes, thus making gradient descent feasible runtime-wise even if we require thousands of iterations for convergence. In Eq. (7.16), we have estimated the number of iterations required to be on the order of a hundred, which will corresponds to a total runtime of around 1 day. 200 processors will mean 10^4 qubits, which can fit onto a single silicon chip along with the classical controls and measurement devices required, giving us one single integrated multi-core processor for the task. We stress that these cores would be independent of one another. Of course, we do not assert that simple gradient descent would necessarily be able to find the solution for the problem size we are considering. However, its runtime feasibility should be indicative of the runtime of the other more advanced gradient-based methods.

7.6 Superconducting Qubits Resource Estimates

When we switch from the silicon spin qubits to superconducting qubits, most of our arguments apply except we need to decompose our circuit using a different gate set and a different set of hardware operation times. For the superconducting qubits, a natural two-qubit gate to use for the Hubbard model simulation will be the partial iSWAP gate [188], which is implemented using XY-interaction and is just the hopping interaction gate we need to implement. The differences between partial iSWAP and partial SWAP have been discussed by Schuch *et al.* [189]. By using partial iSWAP as our only elementary two-qubit gate, it also enable us to implement all Z rotations virtually [134]. As derived in Appendix D.3, for the 5×5 Hubbard model with the number of ansatz blocks equal to the number of sites, the gate counts are:

$$N_{1q} \approx 2500, \quad N_{2q} \approx 14000.$$

Comparing to Eq. (7.2), we can see a massive decrease in the single-qubit gate count due to the use of virtual Z gates, which in turn reduces our fidelity requirement for the single-qubit gates. However, applying virtual Z gate would require us to implement partial iSWAP with a range of different frequency tunings [134], which would increase the difficulties in calibrating the partial iSWAP gates. We also see a reduction in the number of two-qubit gates required, but it is still of the same order and thus will lead to a similar gate fidelity requirement $\sim 10^{-4}$. The resultant mean circuit error count is reduced to

$$\mu \sim 1.5, \tag{7.18}$$

which will lead to improved performance of the error mitigation techniques. Individual superconducting quantum processor of size ~ 50 has been experimentally demonstrated to be able to perform certain tasks that cannot be performed efficiently on any classical computers [8]. There have also been demonstrations of successful implementations of various error mitigation techniques on the superconducting platform [51, 190].

The runtime required for one circuit run as derived in Appendix D.3 is

$$T_{circ} \approx 125\tau_{1q} + 650\tau_{2q} + \tau_{in} + \tau_m$$

where $\tau_{1q}\,\tau_{2q}, \tau_{in}$ and τ_m are the time required for single-qubit gates, two-qubit gates (iSWAP), initialisation and readout, respectively. In superconducting qubits, we have $\tau_{1q} \sim 20ns$ [191], $\tau_{2q} \sim 200ns$ [188] and $\tau_{in} \sim \tau_m \sim 100ns$ [192, 193]. Thus the total runtime is around:

$$T_{circ} \approx 150\mu s$$

which is similar to the runtime estimate for silicon qubits. The difference is that the runtime bottleneck here is the two-qubit gate speed while in silicon the two-qubit gates contribute the least to the overall runtime compared to the other operations.

Using Eq. (7.17), the time needed to evaluate the error-mitigated energy and energy gradient are:

$$T_E = T_{circ} M_E^* = 1500 \text{ s}$$
$$T_{grad} = T_{circ} M_{grad}^* = 1 \times 10^5 \text{ s} \approx 1.2 \text{ days}$$

which are also similar to the silicon platform.

Parallelisation in multiple quantum processors is still essential to bring the total runtime down to a practical level. We need around 150 superconducting-qubit quantum processors, each with 50 qubits (thus a total of around 7500 qubits), to bring time required for each gradient-descent iteration to around 10 minutes. The number of quantum processors required is similar to that of the silicon spin qubits. However, unlike silicon spin qubits which can fit all of the required quantum processors in the same chip, the superconducting processors likely need to be distributed into multiple chips due to a lower qubit density.

7.7 Discussion

In this chapter, we have investigated the resource requirements on obtaining the ground state of a Hamiltonian in a quantum computer using VQE, in which the Hamiltonian cannot be solved exactly classically. The Hamiltonian we have chosen is the 5×5 open-boundary Fermi-Hubbard model due to its favourable scaling in both circuit size and depth. We began our analysis by considering silicon spin qubits as our example hardware platform for the resource estimation. Our ansatz circuit makes use of one of the latest schemes for the input Slater determinant preparation [159] and the Hubbard Hamiltonian ansatz implementation [149], which translates into 17000 single-qubit gates (all are Z rotations) and 26000 two-qubit gates (all are partial swaps) assuming the number of Hamiltonian blocks in the ansatz is equal to the number of sites. Hence, with perfect single-qubit Z rotations and a two-qubit gate error rate on the order of 10^{-4}, we can achieve a circuit error of ~ 2.5. To obtain meaningful results with this mean circuit error count, we must incorporate error mitigation techniques like error extrapolation and symmetry

verification for which we have discussed their combined application to our example. We have devised a measurement scheme that allows us to estimate various terms in the Hamiltonian in parallel and apply symmetry verification at the same time. Bringing all these together, we have estimated the runtime needed for one circuit execution on the silicon-spin-qubit platform to be around $250\mu s$ and thus one iteration in a simple gradient descent optimisation is around 1.7 days due to the large number of samples needed. Hence, we have to run VQE in parallel across multiple quantum processors to reduce the runtime to a feasible level. Around 200 quantum processors with 50 data qubits each, which can fit onto a single silicon chip, can reduce the time required for one gradient-descent iteration to around 10 minutes, and the estimated total algorithm runtime in this case will be around 1 day.

We then applied the same analysis to superconducting qubits. We found a large reduction in the number of single-qubit gates due to the ability to implement all Z rotation virtually by using partial iSWAP as our elementary two-qubit gate. By assuming a two-qubit gate error rate of 10^{-4} and an equal or lower single-qubit gate error rate, we can achieve a mean circuit error count of ~ 1.5. To implement the same algorithm, the circuit runtime is around $150\mu s$ and thus one gradient-descent iteration takes around 1.2 days. This can be reduced to around 10 minutes if we parallelise our task across 150 superconducting quantum processors. Resource estimates for other qubit platforms can be readily obtained by following similar arguments.

To implement Hubbard VQE on NISQ machines, the first key difficulty is to maintain the mean circuit error count at the order of unity or less so that error mitigation can be effective. Superconducting qubits have the advantage here due to the lower gate counts from the good fit of its elementary gate set to the problem, and the recent demonstration of high performance superconducting quantum processors of the relevant size [8]. The second key difficulty is the extremely long runtime due to the huge number of circuit runs required, which motivates the necessity of parallelising our tasks over hundreds or thousands of quantum processors. Silicon spin qubits have better potential here due to its higher qubit density, enabling us

to fit all of these quantum processing units into one single multi-core processor, leaving a smaller and more manageable physical footprint.

We can see that implementing a 50-qubit Hubbard model VQE on a NISQ machine sits right at the boundary of being practical in terms of gate counts and mean circuit error count. Hence, even a constant factor improvement in the mean circuit error count can have big impacts on bringing such an application of NISQ machines into reality, which can be brought about via further optimisation of our simulation schemes, improvements in the gate fidelity, improvements in the optimisation scheme and advances in the error mitigation techniques. We also need to rely on parallelisation to tackle the runtime issue. It is worth noting that the number of qubits required in noisy VQE can become comparable to the fault-tolerant implementation. In the case of VQE, what we need is a lot of independent small units for parallelisation instead of a single integrated device, which should massively reduce the difficulties in manufacturing and calibration even if the total number of qubits is of the same order. However, this also places great emphasis on the ability of the hardware platform to reproduce many copies of the quantum processor once we manage to manufacture a good one. Both of the platforms we discussed have advantages in this respect since their compatibility with commercial CMOS fabrication technology can provide a high-precision, relatively low-cost and highly reproducible manufacturing process.

Due to the large number of hyper-parameters in VQE, our gate and runtime estimates are only indicative of the canonical case. There can be variability in the estimates when we choose a different set of hyper-parameters. One important assumption we made is the number of ansatz blocks in the circuit is equal to the number of sites: $N_{blk} = V$, which is an optimistic assumption given what we have observed in the numerical simulations for small size system [141, 152, 153]. An increase in N_{blk} will lead to a linear increase in the gate counts and a quadratic increase in the runtime (due to the increase in both the gate counts and the parameter counts).

Any increase in the gate counts will lead to an increase in the mean circuit error count, which needs to be suppressed using stronger error mitigation. For example, instead of two-point error extrapolation, we can sample at more error rates, which will increase the number of circuit runs. We can also try to verify for additional symmetries. However, when we have multiple symmetries, it might not be possible to measure the energy terms along with all the symmetries using local measurements in a single circuit run any more, which may lead to additional runtime overhead. It is also worth exploring the possible combinations with other error mitigation techniques like quasi-probability [49, 50], to try to gain improvements in estimation accuracy and/or sampling costs. Steps in this direction, for general algorithms, are taken in the next chapter. We can also try to develop new error mitigation techniques. One possible avenue could be tailoring the noise in real machines using simple gates to increase the sensitivity of our symmetry verification against the noise. The effectiveness of a similar idea in the context of a quantum error correction code has been shown in Chapter 4.

Any increase in the algorithm runtime can be mitigated via further parallelisation by adding more quantum processors. Without any sudden large changes in the parameters during the optimisation, we should expect the energy and the energy gradient change in a relatively smooth manner as the optimisation progresses. Hence, we can use the energy and the energy gradients we obtained in the previous steps as the prior for the estimation of the new energy and the energy gradients in a Bayesian manner [194]. This should enable us to achieve the same precision using much fewer samples and thus much shorter algorithm runtime. Other important factors that can influence the algorithm runtime include the way we parametrised our circuit and the exact optimisation algorithm we choose, both worth further explorations.

Since our ansatz circuit has a relatively short depth ($\mathcal{O}(N^{\frac{1}{2}})$), the main limiting factor preventing us from simulating the Hubbard model much beyond the size 5×5 is due to the increased gate counts and the resultant increased mean circuit error count. Stronger error mitigation or even new error mitigation can only alleviate this problem. To fully tackle this, one possibility is to switch to a different kind of qubit

encoding: the Majorana loop stabiliser code [195]. This encoding allows local vertical hopping terms using mediator qubits, thus no Fermionic swap network is needed and we can achieve a Hamiltonian ansatz block and a Slater determinant preparation circuit with depth $\mathcal{O}(1)$. Furthermore, it can detect and correct single-qubit errors. Hence, it can potentially suppress circuit errors and take us much beyond our current problem size. However, more qubits are needed for the encoding and we need to implement higher-weight operators for the interaction terms. In addition, when considering the stabiliser checks, we need to consider the errors introduced in implementing the check circuits and the connectivity requirement on the hardware. On the other hand, we may implement these stabiliser checks in a post-processing way similar to symmetry verification [66], but many more circuit runs will be needed.

8

Multi-exponential Error Extrapolation and Combining Error Mitigation Techniques

This chapter is adapted from the arXiv manuscript [196], of which the present author is the sole author.

Contents

8.1 Introduction

Previously all the error mitigation techniques have been studied separately. As discussed in Chapter 6, they make use of different information about the hardware and the computation problems to perform different sets of extra circuit runs for error mitigation. Symmetry verification makes use of the symmetry in the simulated system and performs circuit runs with additional measurements. Quasi-probability makes use of the error models of the circuit components and performs circuit runs with different additional gates in the circuit. Error extrapolation makes use of the knowledge about tuning the noise strength via physical control of the hardware, and performs additional circuit runs at different noise levels. Consequently, the three error mitigation techniques are equipped to combat different types of noise with different additional sampling costs (number of additional circuit runs required). Hence, it is natural to wonder how these techniques might complement each other. For NISQ applications, as discussed in Chapter 7, it is important to understand and develop ways for these error mitigation techniques to work in unison, to achieve better performance than the individual techniques in terms of lower errors in the noise-free expectation values estimates and/or lower sampling costs. Thus one key focus of this chapter is on the methods for combining these error mitigation techniques, and trying to gauge their performance under different scenarios through analytical arguments and numerical simulations.

To achieve combinations of the mitigation techniques, it is essential to understand the mechanism behind individual techniques, especially in the NISQ limit in which the number of errors in the circuit will follow a Poisson distribution. Heuristic arguments and numerical validations have been made by Endo *et al.* [50] on error extrapolation using exponential decay curves in this NISQ limit, which was also discussed in Section 6.4.2. This chapter will take this further and provide a more rigorous argument about error extrapolation in this limit, showing that extrapolating

with a multi-exponential decay curve will be the more general solution with lower estimation errors. Our study of error extrapolation will also form the theoretical basis for the combination of error extrapolation with the other error mitigation techniques.

In this chapter, by introducing the concepts of group channels in Section 8.2, we will dive deeper into the mechanism behind error extrapolation and introduce multi-exponential extrapolation in Section 8.3. Using the concepts in the previous sections, we then introduce ways to combine different error mitigation techniques in Section 8.4. In Section 8.5, we apply the error mitigation techniques we developed to the quantum simulation of the Fermi-Hubbard model along with numerical simulations to validate our arguments and demonstrate the performance of the methods. In Section 8.6, we discuss and conclude our findings while suggesting possible future directions.

8.2 Group Errors

Here we will introduce a special kind of error channel: *group error channels*, which will enable us to make more analytical predictions about various error mitigation techniques.

The group error $\mathcal{J}_{p,\mathbb{E}}$ of the group $\mathbb{E}$ is defined to be:

$$\mathcal{J}_{p,\mathbb{E}} = (1-p)\,\overline{I} + \frac{p}{|\mathbb{E}|}\sum_{E\in\mathbb{E}}\overline{E} \tag{8.1}$$

By groups we mean the subgroups of the Pauli group with a composition rule that ignores all the irrelevant phase factors as defined in Section 2.1.1. For the case of $p = 1$, we will call $\mathcal{J}_{1,\mathbb{E}}$ the *pure* group errors.

Many physically interesting noise models like depolarising channels, dephasing channels, Pauli-twirled swap errors and dipole-dipole errors are all group errors.

8.2.1 Application to Symmetry Verification

Here we will consider the effect of applying a set of Pauli symmetry checks $\mathbb{S}$ to the group error in Eq. (8.1). Using Eq. (6.7), $\mathbb{S}$ can remove and detect components in $\mathcal{J}_{p,\mathbb{E}}$ that anti-commute with any elements in $S \in \mathbb{S}$. We look at the action of

$\mathbb{S}$ on the subset of qubits affected by $\mathcal{J}_{p,\mathbb{E}}$, and denote the set of these operators on the subset of qubits as $\mathbb{S}_{sub}$. The commutation relationship between $\mathbb{S}$ and $\mathbb{E}$ is equivalent to that of $\mathbb{S}_{sub}$ and $\mathbb{E}$. We denote their generators as $\widetilde{\mathbb{S}}_{sub}$ and $\widetilde{\mathbb{E}}$. Note that here $\mathbb{S}_{sub}$ is not a group, by $\widetilde{\mathbb{S}}_{sub}$ we just mean the set of independent elements in $\mathbb{S}_{sub}$. For Pauli generators, we can choose $\widetilde{\mathbb{E}}$ in such a way that for every $\widetilde{S}_{sub} \in \widetilde{\mathbb{S}}_{sub}$, there will at most be only one element in $\widetilde{\mathbb{E}}$ that anti-commutes with it. We will denote the elements in $\widetilde{\mathbb{E}}$ that commute with all elements in $\widetilde{\mathbb{S}}_{sub}$ as $\widetilde{\mathbb{Q}}$:

$$\widetilde{\mathbb{Q}} = \{\widetilde{E} \in \widetilde{\mathbb{E}} \mid \left[\widetilde{E}, \widetilde{S}_{sub}\right] = 0 \quad \forall \widetilde{S}_{sub} \in \widetilde{\mathbb{S}}_{sub}\}$$

and it will generate the remaining error components in $\mathbb{E}$ that are not detectable, which we denote as $\mathbb{Q}$. Hence the detectable error components are just $\mathbb{E} - \mathbb{Q}$

Going back to our error channel in Eq. (8.1), the probability that the error gets detected is just the total probability of the detectable error components:

$$p_d = \frac{|\mathbb{E}| - |\mathbb{Q}|}{|\mathbb{E}|} p. \tag{8.2}$$

Removing the detected errors in $\mathcal{J}_{p,\mathbb{E}}$ and renormalising the error channel by the factor $1 - p_d$ gives the effective channel after verification, which is just another group error channel

$$\mathcal{J}_{r,\mathbb{Q}} = (1 - r)\,\overline{I} + \frac{r}{|\mathbb{Q}|} \sum_{Q \in \mathbb{Q}} \overline{Q}$$

with

$$r = \frac{|\mathbb{Q}|p}{|\mathbb{E}|\,(1 - p_d)} = \frac{|\mathbb{Q}|p}{|\mathbb{E}|\,(1-p) + |\mathbb{Q}|p} \approx \frac{|\mathbb{Q}|}{|\mathbb{E}|} p + \mathcal{O}(p^2).$$

8.2.2 Application to Quasi-probability

For a given general Pauli channel, we have only discussed its *approximate* inverse channel in Section 6.3.2. This is because its *exact* inverse channel can be hard to express in a compact analytical form. However, for any group channel, we can easily write down the explicit form of its inverse channel.

As shown in Appendix E.2, it is easy to verify that the inverse of a group channel $\mathcal{J}_{p,\mathbb{E}}$ is just:

$$(\mathcal{J}_{p,\mathbb{E}})^{-1} = \mathcal{J}_{-\alpha,\mathbb{E}} = (1+\alpha)\,\mathcal{I} - \alpha\mathcal{J}_{1,\mathbb{E}} \tag{8.3}$$

with

$$\alpha = \frac{p}{1-p}.$$

Using Eq. (8.3) and Eq. (6.14), the cost of using quasi-probability to invert $\mathcal{J}_{p,\mathbb{E}}$ is thus:

$$\begin{aligned} C_{Q1,0} &= \left(1 + 2\frac{(|\mathbb{E}|-1)\,p}{|\mathbb{E}|(1-p)}\right)^2 \\ &\approx 1 + 4\frac{|\mathbb{E}|-1}{|\mathbb{E}|}p + \mathcal{O}(p^2), \end{aligned} \tag{8.4}$$

which is the same as Eq. (6.16) with

$$p_\epsilon = \frac{|\mathbb{E}|-1}{|\mathbb{E}|}p. \tag{8.5}$$

8.2.3 Transformation to Detectable Error Channels

As shown in Section 8.2.1, for a given group error $\mathcal{J}_{p,\mathbb{E}}$, the resultant error channel after symmetry verification is another group channel $\mathcal{J}_{r,\mathbb{Q}}$ where $\mathbb{Q}$ is a subgroup of $\mathbb{E}$ and $r = \frac{|\mathbb{Q}|}{|\mathbb{E}|}p$. The remaining errors can then be completely removed by implementing $\mathcal{J}_{r,\mathbb{Q}}^{-1}$ using quasi-probability.

Similarly if we implement the quasi-probability inverse channel $\mathcal{J}_{r,\mathbb{Q}}^{-1}$ first and then perform symmetry verification, we can still completely remove the group error $\mathcal{J}_{p,\mathbb{E}}$. The gates we need to implement in the inverse channel $\mathcal{J}_{r,\mathbb{Q}}^{-1}$ will not be detected and thus will not be affected by the symmetry verification. As shown in Appendix E.3, the resultant error channel after applying $\mathcal{J}_{r,\mathbb{Q}}^{-1}$ is:

$$\mathcal{J}_{r,\mathbb{Q}}^{-1}\mathcal{J}_{p,\mathbb{E}} = (1-p_d)\,\overline{I} + \frac{p_d}{|\mathbb{E}|-|\mathbb{Q}|}\sum_{V\in\mathbb{E},V\notin\mathbb{Q}}\overline{V}\,. \tag{8.6}$$

This is a channel that only contains the error components that are detectable by the symmetry verification with the error rate p_d.

As discussed in Section 6.3.2 and explicitly shown in Appendix E.3, we can implement additional quasi-probability operations to further reduce the error rate of the resultant channel to $q \leq p_d$. The resultant detectable error channel will be:

$$\begin{aligned} \mathcal{V}_q &= (1-q)\,\overline{I} + \frac{q}{|\mathbb{E}| - |\mathbb{Q}|} \sum_{V \in \mathbb{E}, V \notin \mathbb{Q}} \overline{V} \\ &= (1-q)\,\overline{I} + q\mathcal{V}_1 \qquad q \leq p_d. \end{aligned} \tag{8.7}$$

8.3 Multi-exponential Error Extrapolation

8.3.1 Multi-exponential Decay of Expectation Values

In this section we will try to gain a deeper insight about the reason behind the exponential decay of $\left\langle O_{|\mathbb{L}|=l} \right\rangle$. If a Pauli error G occurs at the end of a circuit and we are trying to measure an Pauli observable O, then the expectation value is just:

$$\mathrm{Tr}(G\rho GO) = \eta(G,O)\,\mathrm{Tr}(\rho O)$$

where $\eta(G,O)$ is the commutator between G and O:

$$GO = \eta(G,O)OG.$$

If a pure group error $\mathcal{J}_{1,\mathbb{E}}$ occurs at the end of the circuit, then the resultant expectation value is:

$$\begin{aligned} \mathrm{Tr}(\mathcal{J}_{1,\mathbb{E}}(\rho)O) &= \frac{1}{|\mathbb{E}|} \sum_{E \in \mathbb{E}} \mathrm{Tr}(E\rho EO) \\ &= \left(\frac{1}{|\mathbb{E}|} \sum_{E \in \mathbb{E}} \eta(E,O) \right) \mathrm{Tr}(\rho O) \end{aligned}$$

Using Eq. (2.6), we can rewrite the above formula in terms of the generator of $\mathbb{E}$:

$$\mathrm{Tr}(\mathcal{J}_{1,\mathbb{E}}(\rho)O) = \left(\prod_{\widetilde{E} \in \widetilde{\mathbb{E}}} \frac{1+\eta(\widetilde{E},O)}{2} \right) \mathrm{Tr}(\rho O) \tag{8.8}$$

in which

$$\prod_{\widetilde{E} \in \widetilde{\mathbb{E}}} \frac{1+\eta(\widetilde{E},O)}{2} = \begin{cases} 1 & \text{if } \eta(E,O) = 1 \quad \forall E \in \mathbb{E} \\ 0 & \text{otherwise.} \end{cases}$$

Hence, if a pure group error $\mathcal{J}_{1,\mathbb{E}}$ occurs right before measuring a Pauli observable O, then the resultant expectation value $\mathrm{Tr}(\mathcal{J}_{1,\mathbb{E}}(\rho)O)$ will only remain unchanged if O commutes with all elements in $\mathbb{E}$, otherwise the expectation value will be 0.

If we decompose the gates in a unitary circuit $U = \prod_{m=M}^{1} V_m$ into their Pauli components: $V_m = \sum_{k_m} \alpha_{mk_m} G_{mk_m}$, then we have:

$$U = \prod_{m=M}^{1} \sum_{k_m} \alpha_{mk_m} G_{mk_m} = \sum_{\vec{k}} \alpha_{\vec{k}} G_{\vec{k}}$$

where

$$\sum_{\vec{k}} = \prod_{m=M}^{1} \sum_{k_m}, \; G_{\vec{k}} = \prod_{m=M}^{1} G_{mk_m}, \; \alpha_{\vec{k}} = \prod_{m=M}^{1} \alpha_{mk_m}.$$

i.e. the circuit U can be viewed as the superposition of many Pauli circuits $G_{\vec{k}}$.

The expectation value of observing O after applying the circuit U on ρ is

$$\langle O \rangle = \mathrm{Tr}\Big(U\rho U^\dagger O\Big) = 2 \sum_{\vec{j}>\vec{k}} \mathrm{Re}\Big\{\alpha_{\vec{j}}^* \alpha_{\vec{k}} \, \mathrm{Tr}\Big(\rho G_{\vec{j}}^\dagger O G_{\vec{k}}\Big)\Big\}. \tag{8.9}$$

Now let us assume that a given pure group error can occur in between any two gates. There are some error locations within the Pauli circuit $G_{\vec{j}}$ and $G_{\vec{k}}$ at which if the group error occurs, the measurement result $\mathrm{Tr}\Big(\rho G_{\vec{j}}^\dagger O G_{\vec{k}}\Big)$ will not be affected. As proven in Appendix E.4, if we use $r_{\vec{j},\vec{k}}$ to denote the average fraction of such benign error locations within all possible error locations for given $\vec{j}$ and $\vec{k}$, then in the limit of large M and non-vanishing $r_{\vec{j},\vec{k}}$, the expectation value given l errors occurred can be approximated to be:

$$\Big\langle O_{|\mathbb{L}|=l} \Big\rangle \approx 2 \sum_{\vec{j}>\vec{k}} \mathrm{Re}\Big\{\alpha_{\vec{j}}^* \alpha_{\vec{k}} \, \mathrm{Tr}\Big(\rho G_{\vec{j}}^\dagger O G_{\vec{k}}\Big)\Big\} r_{\vec{j},\vec{k}}^l, \tag{8.10}$$

which is just a multi-exponential decay curve.

If we consider the case in which many error locations in the circuit are affected by the same type of noise, and adding onto the fact that in practice there are usually many repetitions of the circuit structures along the circuit and across the qubits, we can expect many $r_{\vec{j},\vec{k}}$ of different $\vec{j}$ and $\vec{k}$ to be very similar. Hence, by grouping the terms with similar $r_{\vec{j},\vec{k}}$ together, Eq. (8.10) becomes

$$\Big\langle O_{|\mathbb{L}|=l} \Big\rangle = \sum_{k=1}^{K} A_k r_k^l \tag{8.11}$$

where A_k is the sum of $2\,\mathrm{Re}\left\{\alpha^*_{\vec{j}}\alpha_{\vec{k}}\,\mathrm{Tr}\left(\rho G^\dagger_{\vec{j}} O G_{\vec{k}}\right)\right\}$ for some subset of $\vec{j}$ and $\vec{k}$. Note that A_k are independent of l and we have $\sum_{k=1}^{K} A_k = \langle O\rangle$.

So far we have only been considering group error channels. However, as shown in Appendix E.5, by writing a general Pauli channel as the linear combination of pure group channels, we can prove that decay of the expectation value under general Pauli noise can also be approximated by a sum of exponentials like in Eq. (8.11), which by defining $\gamma_k = 1 - r_k$ can be rewritten as:

$$\left\langle O_{|\mathbb{L}|=l}\right\rangle = \sum_{k=1}^{K} A_k (1-\gamma_k)^l. \tag{8.12}$$

8.3.2 Error Extrapolation

Eq. (8.12) can be translated into a multi-exponential decay of $\langle O_\mu\rangle$ over μ using Eq. (6.23):

$$\langle O_\mu\rangle = e^{-\mu}\sum_{l=0}^{\infty}\frac{\mu^l}{l!}\sum_{k=1}^{K} A_k(1-\gamma_k)^l \tag{8.13}$$

$$= \sum_{k=1}^{K} A_k e^{-\gamma_k \mu}. \tag{8.14}$$

Here we can see that different γ_k are just the decay constants for the different exponential components. This can be rewritten as

$$\langle O_\mu\rangle = \sum_{k=1}^{K} A_k \left(e^{-\gamma_k}\right)^\mu \approx \sum_{k=1}^{K} A_k \left(1-\gamma_k\right)^\mu.$$

Comparing with Eq. (8.12), we see that the shape of $\langle O_\mu\rangle$ and $\left\langle O_{|\mathbb{L}|=l}\right\rangle$ are the same up to the leading order of γ_k with the mean circuit error count μ in place of the circuit error count l.

The simplest shape that we can fit over $\langle O_\mu\rangle$ is a single exponential decay curve ($K = 1$), which is the one we used in exponential extrapolation. We see here that a natural extension of this will be probing $\langle O_\mu\rangle$ at more than two different error rates and trying to fit them using a sum of exponentials ($K > 1$). The estimate of the error-free observable $\langle O\rangle$ can then be obtained by substituting $\mu = 0$ into our fitted curve.

From Eq. (8.14), we see that the k^{th}exponential component can only survive up to the mean circuit error count $\mu \sim \frac{1}{\gamma_k}$, thus we can only obtain information about this component by probing at the mean circuit error count $\mu \lesssim \frac{1}{\gamma_k}$. Since $0 \leq \gamma_k \leq 1$, we have $\frac{1}{\gamma_k} \geq 1$ for all k, i.e. we should be able to retrieve all exponential components if we can probe enough points within $\mu \lesssim 1$. In practice, there is a minimum mean circuit error count that we can achieve, which we denote as μ^*. To have an accurate multi-exponential fitting, it is essential that for all components with non-negligible A_k, we have $\mu^* \lesssim \frac{1}{\gamma_k}$, i.e. none of the critical exponential components has died away at the minimal error rate that we can probe.

8.4 Combination of Error Mitigation Techniques

8.4.1 Quasi-probability with Exponential Extrapolation (QE)

As discussed in Section 8.3, the decay of Pauli expectation values under Pauli noise can be approximated using multi-exponential decay curves, and the number of exponential decays in the sum can be reduced with increased degree of symmetry in the circuit and/or if the error channels are group channels. Thus we can use quasi-probability to transform the error channels into similar group error channels for easier curve fitting in the extrapolation process. In addition, we can use quasi-probability to reduce the mean circuit error count from μ to ν with the multiplicative sampling cost $C_{Q,\nu} = e^{4(\mu_\epsilon - \nu_\epsilon)}$ as shown in Eq. (6.17), and then mitigate the rest of the errors using multi-exponential extrapolation. In the case of two-point extrapolation using a single exponential curve, the two natural noise strengths to probe are just μ and ν. The cost of probing at ν is the combined cost of quasi-probability and extrapolation, while probing at μ only requires the cost of extrapolation. Following Appendix E.6.3, the total sampling cost of combining quasi-probability and two-point exponential extrapolation is

$$C_{QE} \sim \frac{2\left(\lambda^2 e^{\frac{2}{\lambda}[\gamma\mu + 2(\lambda-1)\mu_\epsilon]} + e^{2\gamma\mu}\right)}{(\lambda-1)^2} \tag{8.15}$$

with $\mu = \lambda\nu$.

8.4.2 Quasi-probability with Symmetry Verification (QS)

The natural way to combine symmetry verification and quasi-probability is just using quasi-probability to remove the error components that cannot be detected by the given symmetry, which we have discussed in the case of group channels in Section 8.2.3. Note that these additional quasi-probability operations may contain gates that take us from one symmetry space to another, for which we need to adjust our symmetry verification criterion accordingly.

Using quasi-probability, we can remove all the local undetectable errors in the circuit, which reduces the mean circuit error count from μ to μ_d. The effect of applying symmetry verification at this point has been discussed in Section 6.2.3. We can further suppress the remaining errors to achieve a mean circuit error count of $\nu \leq \mu_d$ and follow the same arguments in Section 6.2.3 with ν in place of μ_d. Using Eq. (6.20), the sampling cost factor of the required quasi-probability transformation is:

$$C_{Q,\nu} \approx e^{4(\mu_\epsilon - \nu)} \quad \nu \leq \mu_d \tag{8.16}$$

where we have made use of $\nu_\epsilon = \nu$ for the detectable error channels like $\mathcal{V}_q$ in Eq. (8.7).

In the resultant circuit with only local detectable errors present, the sampling cost of applying symmetry verification is given by Eq. (6.9) with ν in place of μ_d:

$$C_{S,\nu} = \frac{1}{e^{-\nu}\cosh(\nu)}.$$

Hence, the total sampling cost including quasi-probability is:

$$C_{QS,\nu} = C_{Q,\nu} C_{S,\nu} = \frac{e^{4\mu_\epsilon}}{e^{3\nu}\cosh(\nu)} \tag{8.17}$$

which is smaller than the cost of using quasi-probability to remove all of our errors without the help of symmetry verification in Eq. (6.19): $C_{Q,0} \approx e^{4\mu_\epsilon}$.

However, we need to note that with the pure quasi-probability method, we can remove all of the noise, while when we combine with symmetry verification,

the fraction of erroneous circuit runs after applying our combined error mitigation is given by Eq. (6.10) with ν in place of μ_d:

$$P_{circ} = \frac{1}{2}\left(1 - e^{-\nu}\right)^2.$$

Hence, we must choose a ν such that P_{circ} is the circuit error rate that we can tolerate.

The saving of sampling cost when combining symmetry verification with quasi-probability over pure quasi-probability can be written as:

$$\frac{C_{Q,0}}{C_{QS,\nu}} = e^{3\nu}\cosh(\nu) = \left(1 - \sqrt{2P_{circ}}\right)^{-4}\left(1 - \sqrt{2P_{circ}} + P_{circ}\right).$$

To achieve $P_{circ} \ll 1$, we have:

$$\frac{C_{Q,0}}{C_{QS,\nu}} \approx 1 + 3\sqrt{2P_{circ}} + \mathcal{O}(P_{circ}).$$

Therefore by combining with symmetry verification, the factor of saving in the sampling cost over pure quasi-probability is ${}^{C_{Q,0}}/_{C_{QS,\nu}} = 1.5$ for the circuit error rate $P_{circ} = 10^{-2}$ and ${}^{C_{Q,0}}/_{C_{QS,\nu}} = 1.15$ for the circuit error rate $P_{circ} = 10^{-3}$. To push the circuit error rate any lower, the saving in the sampling cost will be negligible. And in the limit of $P_{circ} = 0 \Rightarrow \nu = 0$, we just have pure quasi-probability which removes all the errors.

8.4.3 Combining All Three Techniques: Quasi-probability with Hyperbolic Extrapolation (QH)

In Section 8.4.2 we use quasi-probability to remove all local error components that are undetectable by the given symmetry. Thus after symmetry verification, only instances with an even number of occurrence of local detectable errors will contribute to our circuit errors. We can try to suppress them by applying additional quasi-probability operations to lower the local error rate as shown in Section 8.4.2. Here instead, we will try to remove these errors by applying error extrapolation.

As mentioned, we can transform all the error channels in the circuit with the total mean error count μ into *detectable* error channels with total mean error count ν using quasi-probability. The effect of one symmetry verification essentially split

our circuit runs into two separate sets: one has even number of errors occurring and thus will pass the symmetry test, the other has odd number of errors occurring and thus will fail the symmetry test. Splitting the expectation value (Eq. (8.13)) into the weighted sum of these two parts we have

$$\langle O_\nu \rangle = e^{-\nu} \left(\cosh(\nu) \langle O_{c,\nu} \rangle + \sinh(\nu) \langle O_{s,\nu} \rangle \right) \tag{8.18}$$

in which $e^{-\nu}\cosh(\nu)$ and $e^{-\nu}\sinh(\nu)$ are the probability to have even and odd number of errors occurring in the circuit, respectively. And $\langle O_{c,\nu} \rangle$, $\langle O_{s,\nu} \rangle$ are the corresponding expectation values in these cases with

$$\begin{aligned} \langle O_{c,\nu} \rangle &= \frac{1}{\cosh(\nu)} \sum_{k=1}^{K} A_k \cosh((1-\gamma_k)\,\nu), \\ \langle O_{s,\nu} \rangle &= \frac{1}{\sinh(\nu)} \sum_{k=1}^{K} A_k \sinh((1-\gamma_k)\,\nu). \end{aligned} \tag{8.19}$$

Remember that γ_k is the decay constant for the k^{th} exponential component as defined in Section 8.3.2.

We will consider the case that the decay of our expectation value $\langle O_\nu \rangle$ over increased mean circuit error count ν follows a single exponential curve ($K = 1$) for simplicity, we then have:

$$\begin{aligned} \langle O_{c,\nu} \rangle &= \langle O \rangle \frac{\cosh((1-\gamma)\,\nu)}{\cosh(\nu)} \\ \langle O_{s,\nu} \rangle &= \langle O \rangle \frac{\sinh((1-\gamma)\,\nu)}{\sinh(\nu)} \end{aligned} \tag{8.20}$$

which gives:

$$\langle O \rangle = \operatorname{sgn}\left(\langle O_{c,\nu} \rangle \right) \sqrt{\langle O_{c,\nu} \rangle^2 \cosh^2(\nu) - \langle O_{s,\nu} \rangle^2 \sinh^2(\nu)}. \tag{8.21}$$

Note that we have used the fact that $\langle O_{c,\nu} \rangle$ and $\langle O \rangle$ have the same sign since $1 - \gamma > 0$ and $\nu > 0$. Here with the help of symmetry verification and quasi-probability, we can now obtain an estimate of the error-free expectation value $\langle O \rangle$ by combining the expectation value of the passed runs and failed runs at one error rate ν instead of combining the expectation value of runs at different error rates in the conventional error extrapolation. Note that here we have assumed

that we know the value of the mean circuit error count ν, which needs be known before we can apply the quasi-probability step anyway. The method we employed here will be called *hyperbolic extrapolation*.

As derived in Appendix E.6.4, the sampling cost factor of hyperbolic extrapolation is

$$C_{H,\nu} = \cosh(2\,(1-\gamma)\,\nu)\cosh(\nu)e^{\nu}. \tag{8.22}$$

To combine all three error mitigation techniques, we have used quasi-probability to remove the error components that are undetectable by symmetry verifications. Then symmetry verification will split all circuit runs into two sets: runs with even number of errors and runs with odd number of errors, obtaining two separate expectation values. Using our understanding about the decay of the expectation value from our study of error extrapolation, we can simply combine these two erroneous expectation values and obtain the error-free expectation value. Using Eq. (8.16) and Eq. (8.22), the total sampling cost factor for such a process is:

$$C_{QH,\nu}(\gamma) = C_{Q,\nu}C_{H,\nu} = e^{4\mu_\epsilon}\frac{\cosh(\nu)\cosh(2\,(1-\gamma)\,\nu)}{e^{3\nu}}. \tag{8.23}$$

We note that this is always smaller than the cost of pure quasi-probability $C_{Q,0} = e^{4\mu_\epsilon}$. Its cost saving over pure quasi-probability will increase with the increase of ν and γ. We need to note that the cost expression here is only valid for $0 \leq \nu \leq \mu_d$ following from Eq. (8.16), thus the minimum cost that we can achieve will be at $\nu = \mu_d$ for which we have:

$$C_{QH}(\gamma) = e^{4\mu_\epsilon}\frac{\cosh(\mu_d)\cosh(2\,(1-\gamma)\,\mu_d)}{e^{\mu_d}}. \tag{8.24}$$

The only reason to try to push ν below μ_d is when such an action will result in easier (smaller K) and/or better fitting to our extrapolation curves.

8.5 Numerical Simulation

8.5.1 Simulation Scheme

In this section, we will try to apply our results to the Fermi-Hubbard model simulation circuit as outlined in Chapter 7, which consists of local two-qubit

components that correspond to different interaction terms and the fermionic swap operation. It can be used for both eigenstate preparation and time evolution simulation. We will assume there are M of these two-qubit components, and they all suffer from two-qubit depolarising noise of error probability p, which is just a group channel of the two-qubit Pauli group (without the phase factors). Using Eq. (8.5), we have

$$p_\epsilon = \frac{|\mathbb{E}| - 1}{|\mathbb{E}|} p = \frac{15}{16} p \qquad \Rightarrow \qquad \mu_\epsilon = \frac{15}{16} \mu. \tag{8.25}$$

Since we usually know the number of fermions in the system, we can try to verify the fermion number parity symmetry of the output state, which is simply:

$$S_\sigma = \prod_i Z_i$$

in the Jordan-Wigner qubit encoding. All the local two-qubit components in the circuit conserve the symmetry S_σ. Hence when we start in a state with the right fermion number, the output state should also have the correct fermion number, enabling us to perform symmetry verification.

By checking $S_\sigma = \prod_i Z_i$, we can detect all error components with *one* X or Y in the local two-qubit depolarising channels since they anti-commute with S_σ. We will remove the other error components in the local two-qubit depolarising channels using quasi-probability. The removed components are the components that can be generated from the set $\widetilde{\mathbb{Q}} = \{Z_1, Z_2, X_1 X_2\}$ following Section 8.2.1. Thus we have $|\mathbb{Q}| = 2^3 = 8$ and using Eq. (8.2) we also have

$$p_d = \frac{|\mathbb{E}| - |\mathbb{Q}|}{|\mathbb{E}|} p = \frac{p}{2} \qquad \Rightarrow \qquad \mu_d = \frac{\mu}{2}. \tag{8.26}$$

The resultant noise channel after the application of quasi-probability is given by Eq. (8.7), which is just a uniform distribution of the two-qubit Pauli errors that are detectable by S_σ. We will call it the *detectable noise*.

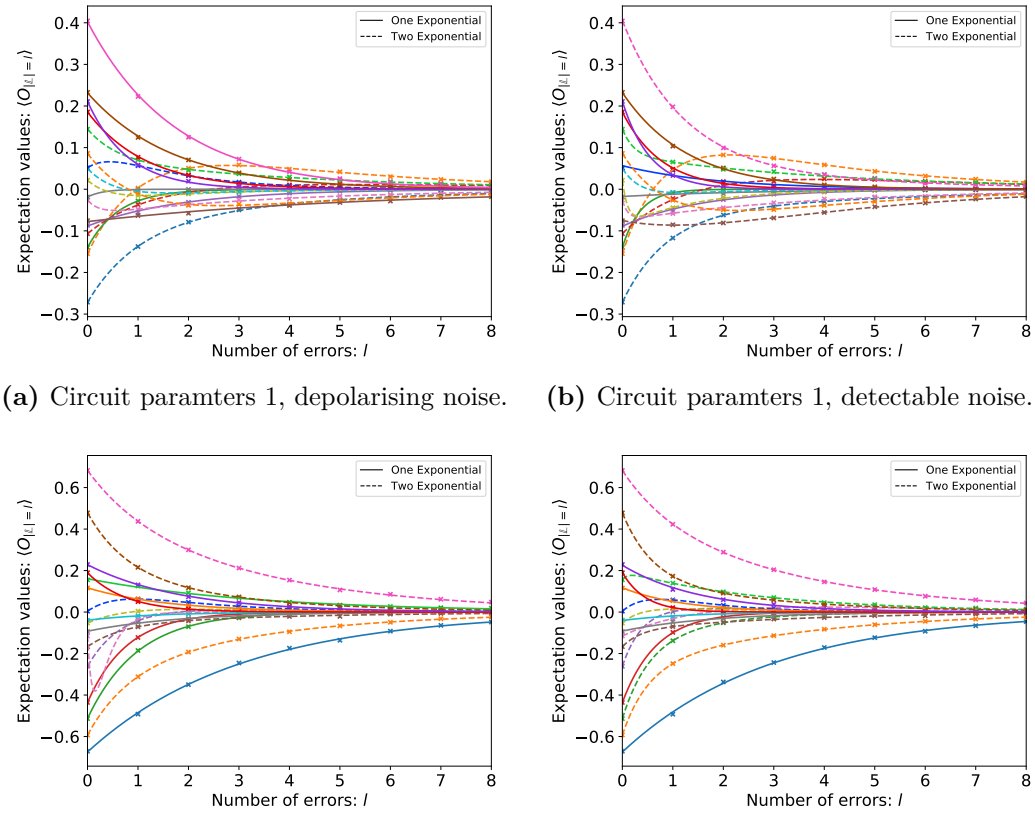
In the following sections, we will perform numerical simulations using the circuit for the 2×2 half-filled Fermi-Hubbard model, which consists of 8 qubits and 144 two-qubit gates. The two-qubit gates in the circuit that correspond to

interaction terms are parameterizable gates with the parameters indicating the strength of the interaction. In our simulation, we will obtain the results for two sets of randomly chosen gate parameters. We will also look at two different error scenarios: depolarising errors and detectable errors. One of them is a group channel while the other is a more general Pauli channel. The measurements that we perform will be the Pauli components of the energy operator, from which we can reconstruct the energy of the output state. The simulations are performed using the Mathematica interface [197] of the high-performance quantum computation simulation package QuEST [198].

8.5.2 Performance of Multi-exponential Extrapolation

Recall that in Section 6.4, we use $\left\langle O_{|\mathbb{L}|=l} \right\rangle$ to denote the expectation value of the Pauli observable O when there are l errors in the circuit. In Fig. 8.1, we have plotted the simulation results showing how $\left\langle O_{|\mathbb{L}|=l} \right\rangle$ changes with l for two sets of circuit parameters under two-qubit depolarising noise and two-qubit detectable noise. What we will first notice is the similarity between the data of the two different noise models for the same circuit. This should not come as a surprise since the detectable error channel can be viewed as the linear combination of the depolarising channel and the undetectable group channel. Just as predicted by Eq. (8.12), we can see that our $\left\langle O_{|\mathbb{L}|=l} \right\rangle$ data points can be closely fitted using a sum of exponentials. In fact, all of the observables in Fig. 8.1 can be fitted using a sum of just one or two exponentials, even though our circuit is generated from a set of random parameters and thus should be lacking symmetries.

As discussed in Section 8.3.2, a multi-exponential decay of $\left\langle O_{|\mathbb{L}|=l} \right\rangle$ will translate into a multi-exponential decay of $\langle O_\mu \rangle$ of similar shape. In Fig. 8.2, we have plotted $\langle O_\mu \rangle$ of each observable at the mean circuit error counts $\mu = 0.5,\ 1,\ 1.5,\ 2$ and performed both single- and dual-exponential extrapolation on them. The true expectation values are plotted at $\mu = 0$, using which we can compare to our fitted curves and calculate the absolute errors in the single- and dual-exponential



(a) Circuit paramters 1, depolarising noise.

(b) Circuit paramters 1, detectable noise.

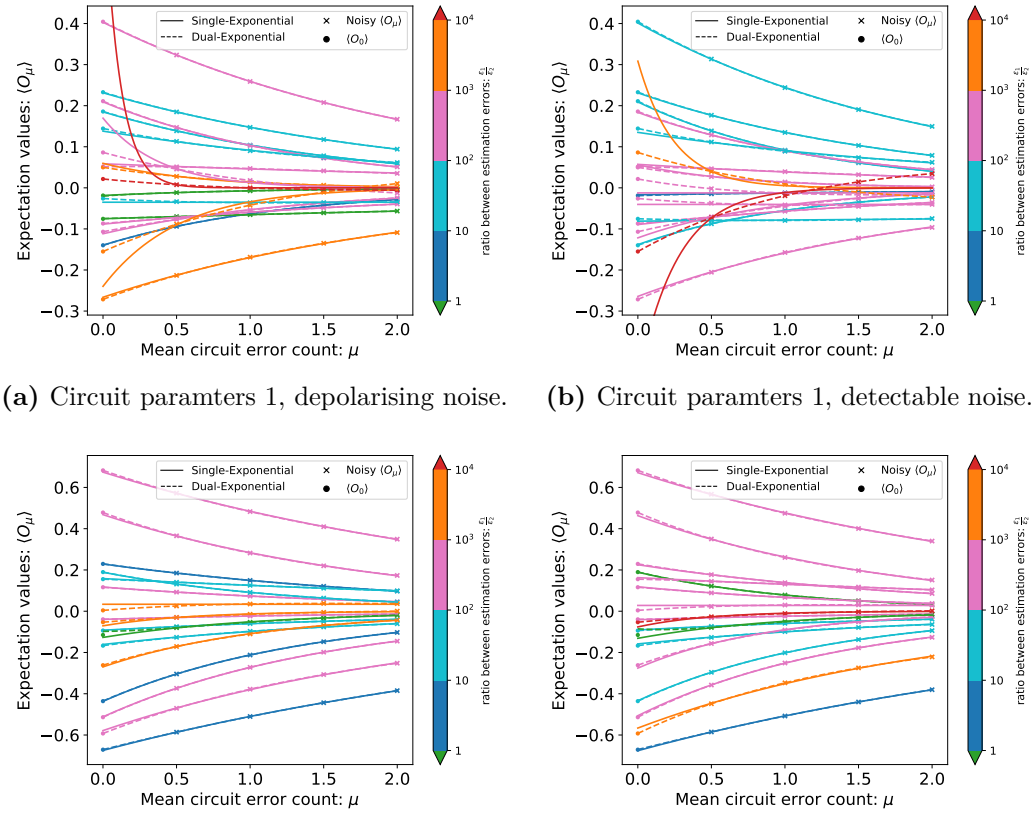
(c) Circuit paramters 2, depolarising noise.

(d) Circuit paramters 2, detectable noise.

Figure 8.1: Plots showing the decay of the expectation values of various Pauli observables as the number of errors in the circuit increases. Here we are looking at two different sets of parameters in the circuit and two different noise models in a 8-qubit simulation. Different colours denote different measured observables for the given circuit and there is no colour correspondence between different circuit parameters. The lines are the fitted curves to the data points, for which the solid lines denote single-exponential curves, while the dashed lines denote dual-exponential curves. All data generated in our simulation can be fitted using a sum of at most two exponentials.

extrapolation estimates, which we denote as ϵ_1 and ϵ_2 respectively. Different colours in the plots correspond to different estimation error ratios ϵ_1/ϵ_2.

For 54 out of the 58 observables we plotted, dual-exponential extrapolation can achieve smaller estimation errors than single-exponential extrapolation ($\epsilon_1/\epsilon_2 > 1$). Within the four cases that dual-exponential extrapolation is outperformed (the green curves in Fig. 8.2), two of them are the cases in which single-exponential extrapolation performs so well that ϵ_1 are small enough to be of the same order as ϵ_2. The other two are cases with large ϵ_2, mainly because we are only using



(a) Circuit paramters 1, depolarising noise. **(b)** Circuit paramters 1, detectable noise.

(c) Circuit paramters 2, depolarising noise. **(d)** Circuit paramters 2, detectable noise.

Figure 8.2: Plots showing the noisy expectation values of different Pauli observables obtained at the four mean circuit error counts $\mu = 0.5,\ 1,\ 1.5,\ 2$ (cross markers) under two different sets of circuit parameters and two different noise models in a 8-qubit simulation. The single- and dual-exponential extrapolation curves fitted to the data points are represented by the solid and dashed lines, respectively. The circular markers lie at $\mu = 0$ and denote the true noiseless expectation values. Different colours represent different ratios between the estimation errors of using single- and dual-exponential extrapolation (e.g. orange means that for the given observable, the estimation errors of single-exponential extrapolation is between 10^3 to 10^4 times larger than that of dual-exponential extrapolation).

the bare minimum of 4 data points to fit a dual-exponential curve with 4 free parameters, thus any small irregularities in our data points will lead to large errors in the fitting. This can be exacerbated if the true expectation value is very small, leading to large uncertainties in the fitting parameters. It can simply be solved by probing at more error rates to obtain more data points. On the other hand, there are also a few cases in which the ϵ_1 are exceptionally large (e.g. certain orange and red curves in Fig. 8.2 (a) and (b)). These are usually observables whose decay

curves have extrema and/or crossing over the x-axis, thus it is impossible to get a good fit with a single-exponential curve. For these observables, dual-exponential extrapolation can still perform extremely well and achieve $\epsilon_2 \sim 10^{-5}$, which is up to tens of thousands times lower than ϵ_1: $\epsilon_1/\epsilon_2 \sim 10^4$.

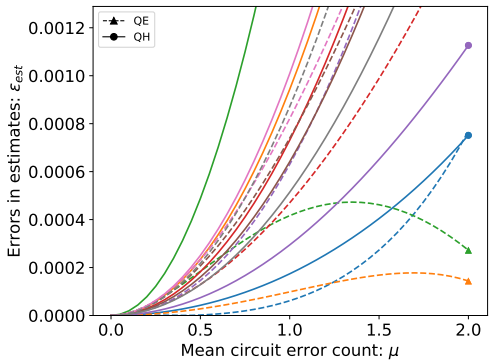
Now we will exclude the few observables above with exceptionally large ϵ_1 or ϵ_2 and take the average of the remaining ϵ_1 and ϵ_2 to obtain a more representative performance of dual-exponential extrapolation against single-exponential extrapolation. This is shown in Table 8.1, from which we see that by using dual-exponential extrapolation instead of single-exponential extrapolation, we can achieve tens or even a hundred times reduction in estimation errors across both circuit parameters and both noise models. Note that in Fig. 8.2 it appears to the eye that the true (noiseless) expectation values, marked by filled circles, never deviate from the dual-exponential (dashed) lines. In fact there are minute discrepancies as specified in Table 8.1, but the extrapolation is remarkably successful.

$\bar{\epsilon}_1,\bar{\epsilon}_2$ $/10^{-4}$	Depolarising		Detectable	
	Single-exp	Dual-exp	Single-exp	Dual-exp
Param 1	150	1.0	74	1.0
Param 2	67	1.1	96	1.3

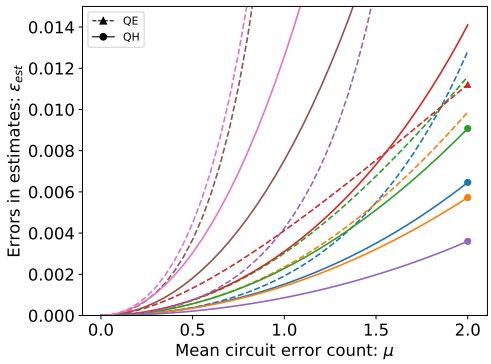
Table 8.1: The errors in the single- and dual-extrapolation estimates averaged over all observables within each plots in Fig. 8.2. The entries are in the unit of 10^{-4}.

8.5.3 Comparison between Quasi-probability with Exponential and with Hyperbolic Extrapolation

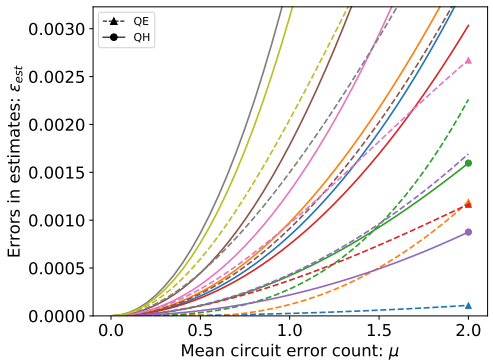
In the last section, we probed at four different error rates to perform single- and dual-exponential extrapolation. However, for many error sources in practice, there might be challenges to physically adjust the hardware error rate for extrapolation. One way to overcome this is to use the combined method 'quasi-probability with exponential extrapolation' (QE) outlined in Section 8.4.1, in which we use quasi-probability to suppress the mean circuit error count from μ to ν and use the data



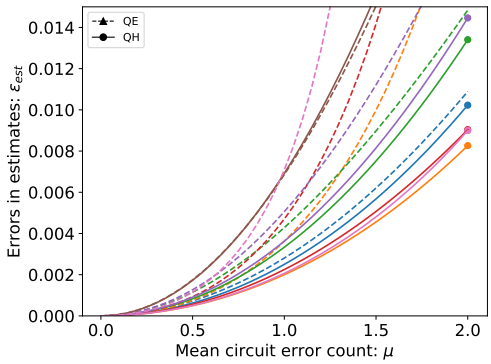
(a) Circuit parameters 1, single-exponential observables.



(b) Circuit parameters 1, dual-exponential observables.



(c) Circuit parameters 2, single-exponential observables.



(d) Circuit parameters 2, dual-exponential observables.

Figure 8.3: Plots showing the errors in our estimated expectation values using the two different mitigation techniques against the mean circuit error count. Here we are looking at two different sets of circuit parameters and for each case, we split all the observables into two sets, one for observables following single-exponential decay and the other for observables following dual-exponential decay. Within each plot, different colours represent different observables. The solid lines denote quasi-probability with hyperbolic extrapolation (QH), while the dashed lines denote quasi-probability with exponential extrapolation (QE). At the mean circuit error counts $\mu = 2$, for each observable, we use markers to denote the method that has a lower estimation error out of the two. For a given observable, circular markers denote lower estimation errors when using QH, while triangle markers denote lower estimation errors when using QE. Note that in (a) there are two QE curves that are not monotonously increasing. Though these two curves are labelled as single-exponential observables, they are actually dual-exponential observables that can be fitted using a single-exponential decay curve with small losses due to their small magnitudes. In such case, when fitting using single-exponential QE at a given error rate, we are essentially fitting to the exponential component that are dominating at that particular error rate, thus the estimation errors is dominated by the other exponential component. This leads to a turning point in the estimation error curve which signals the transition between different exponential components dominating the estimation errors.

at these two error rates to perform single-exponential extrapolation. If we have a symmetry present in the system, another way will be using quasi-probability to remove the local error components undetectable by the symmetry, giving a resultant mean detectable error count ν, and then perform hyperbolic extrapolation. This is just the method 'quasi-probability with hyperbolic extrapolation' (QH) discussed in Section 8.4.3. In this section, we will compare the performance of QE and QH in the case of Fermi-Hubbard model simulation with local two-qubit depolarising noise with a mean circuit error count μ and we will use fermionic number parity symmetry for QH. For the quasi-probability we apply in QH, we will just remove the undetectable error components without suppressing any of the detectable components, which means $\nu = \mu_d = \frac{\mu}{2}$ following the discussions in Section 8.4.2 and Eq. (8.26). For the quasi-probability in QE in this section, we will keep it at the same strength as that in QH: $\nu = \frac{\mu}{2}$. Note that even though resultant channels after the partial quasi-probability in both QE and QH give the same mean circuit error count $\nu = \frac{\mu}{2}$, in one case the resultant noise is still depolarising while in the other case the resultant noise is locally detectable. In this section, we will assume the quasi-probability process is performed perfectly.

As shown in Fig. 8.1, for our example circuits, the observables will follow either single- or dual-exponential decay, which we will call single-exponential observables and dual-exponential observables, respectively. In Fig. 8.3, we have plotted the absolute estimation errors ϵ_{est} using the two different extrapolation techniques against the mean circuit error count μ under two different circuit configurations and for the single-exponential and dual-exponential observables separately. First, we can see that the estimation errors for the dual-exponential observables are almost one order of magnitude higher than the errors for the single-exponential observables. This should not come as a surprise since both single-exponential extrapolation and hyperbolic extrapolation are derived under the assumptions of single-exponential observables. At the mean circuit error counts $\mu = 2$, for each observable, we have used markers to label the method that can achieve lower estimation error out of the two. We see that the number of single-exponential

observables that can achieve a lower estimation error using QE is greater than that of QH. On the other hand, almost all dual-exponential observables can achieve a lower estimation error using QH.

$\bar{\epsilon}_{est}/10^{-4}$	1-Exp. Obs.		2-Exp. Obs.		All Obs.	
	QE	QH	QE	QH	QE	QH
Param 1	5.1	7.8	100	56	53	32
Param 2	7.4	14	49	32	26	22

(a) $\mu = 1$

$\bar{\epsilon}_{est}/10^{-3}$	1-Exp. Obs.		2-Exp Obs.		All Obs.	
	QE	QH	QE	QH	QE	QH
Param 1	1.8	3.2	82	20	39	11
Param 2	2.5	6.1	110	13	49	9.2

(b) $\mu = 2$

Table 8.2: The errors in the error-mitigated estimates using QE and QH averaged over single-exponential, dual-exponential and all observables for two sets of circuit parameters, at the mean circuit error counts (a) $\mu = 1$ and (b) $\mu = 2$. The entries in (a) and (b) are in the units of 10^{-4} and 10^{-3}, respectively.

In Table 8.2, we further calculate the average estimation errors of single-exponential, dual-exponential and all observables separately at $\mu = 1, 2$, which re-confirm all of our observations above. We see that the estimation errors of QE are almost half of that of QH for single-exponential observables, and on the other hand QH can achieve much lower estimation errors for dual-exponential observables. In other words, the performance of QH is more robust against whether the observable is single-exponential or not. When looking at the estimation errors averaged over all observables, we see the estimation errors of QH is always lower than QE and can be 4 to 5 times smaller than QE at $\mu = 2$. The all-observable averages can be more indicative about the practical performance of the mitigation techniques since in experiments we do not know whether a given observable should be fitted with single-exponential or not beforehand.

There is another added layer of robustness when we try to apply QH instead of QE to multi-exponential observables when we look back at the hyperbolic extrapolation equation: Eq. (8.21). We can see that if the shape of the observable is far off from a single-exponential decay, then this might lead to a negative number in the square root of Eq. (8.21), allowing us to realise that we need to probe at more error rates to perform multi-exponential extrapolation instead, avoiding performing a bad extrapolation with very large errors. In the simulation, we indeed identify a few observables that we cannot perform QH on. For these observables, we can still perform QE, but it will lead to huge errors in the estimates. These observables have been excluded in our comparison between QE and QH.

8.5.4 Cost of Quasi-probability with Exponential and with Hyperbolic Extrapolation

Using Eq. (8.24), Eq. (8.25) and Eq. (8.26), the sampling cost factor of performing QH in our example circuit is

$$C_{QH}(\gamma) = e^{\frac{9}{4}\mu} \cosh\left(\frac{\mu}{2}\right) \cosh((1-\gamma)\,\mu),$$

where γ is the decay rate of the observable expectation values under noise.

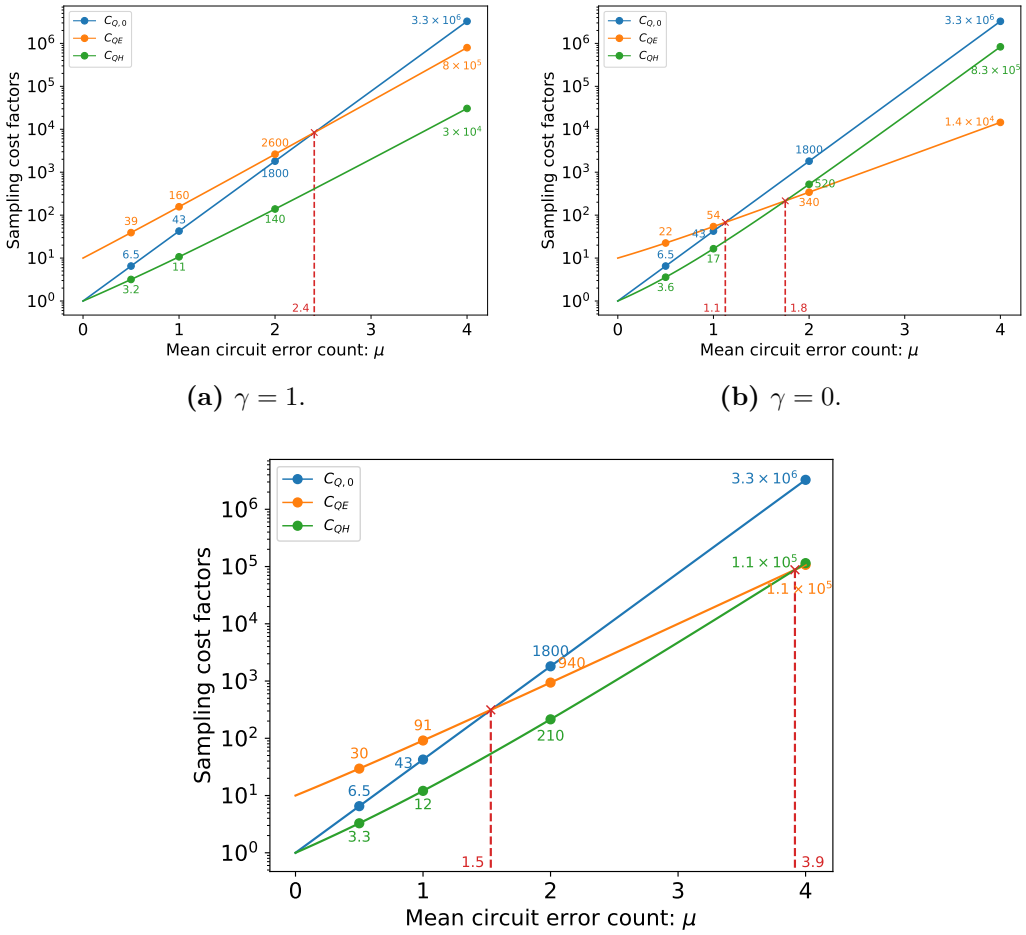
Using Eq. (8.15), Eq. (8.25) and $\nu = \mu_d = \frac{\mu}{2} \Rightarrow \lambda = 2$, the sampling cost factor of performing QE is:

$$C_{QE} = 2\left(4e^{(\gamma+\frac{15}{8})\mu} + e^{2\gamma\mu}\right).$$

For comparison purpose, we also write down the sampling cost factor for removing all the errors using quasi-probability given by Eq. (6.19):

$$C_{Q,0} \approx e^{4\mu_\epsilon} = e^{\frac{15}{4}\mu}.$$

The comparison between $C_{Q,0}$, $C_{QE}(\gamma)$ and $C_{QH}(\gamma)$ at different γ is plotted in Fig. 8.4. We can see that $C_{QH}(\gamma)$ is always lower than $C_{Q,0}$ across all μ and γ, i.e. we can always get a sampling cost saving by applying QH instead of pure quasi-probability, which is also proven in Section 8.4.3. On the other hand, at



(a) $\gamma = 1$.

(b) $\gamma = 0$.

(c) $\gamma = 0.5$.

Figure 8.4: Plots showing the sampling cost factors of three different error mitigation techniques: pure quasi-probability (Q), quasi-probability with exponential extrapolation (QE) and quasi-probability with hyperbolic extrapolation (QH) against the mean circuit error count at $\gamma = 0,\ 0.5,\ 1$. Here γ is the decay rate of the observables under noise. We have labelled the values of the lines at the mean circuit error counts $\mu = 0.5,\ 1,\ 2,\ 4$. We have also labelled the intersects using red markers.

$\gamma = 1$, C_{QE} is larger than C_{QH} for all μ and larger than $C_{Q,0}$ for $\mu < 2.4$. As γ decreases, $C_{QH}(\gamma)$ will increase while $C_{QE}(\gamma)$ will decrease. Thus they naturally complement each other as QH will be more suitable for large-γ error mitigation while QE will be more suitable for small-γ error mitigation. At $\gamma = 0$, we see that C_{QE} becomes lower than both C_{QH} and $C_{Q,0}$ at $\mu > 1.8$.

The average fitted γ of all the single-exponential observables within each plot in Fig. 8.1 all lie within the range $0.5 - 0.6$. Hence, we will now focus on the $\gamma = 0.5$ plot in Fig. 8.4 to get an indication of the practical sampling costs of

implementing different mitigation techniques.

At $\mu = 1$, the sampling cost factor of quasi-probability is 43. QE requires a higher sampling cost, thus there is no point performing QE since pure quasi-probability can remove all the noise perfectly in theory with a lower cost. Compared to quasi-probability, QH can reduce the cost by more than 70% while still achieving the small estimation errors $\bar{\epsilon}_{QH} \sim 3 \times 10^{-3}$ shown in Table 8.2. In order for quasi-probability to have any advantages over QH, we must sample enough times such that the shot noise of pure quasi-probability is smaller than the estimation errors of QH (more rigorous arguments in Appendix E.7), which will require $N^* \sim {}^{C_{Q,0}}/_{\bar{\epsilon}^2_{QH}} \approx 4.3 \times 10^6$ samples for each observable. Therefore in practice, QH could be the preferred method over pure quasi-probability as it is challenging to sample more than N^* for each observable within reasonable runtime constraints (as seen in Chapter 7).

At $\mu = 2$, now the sampling cost factor of quasi-probability is 1800, which is hardly practical. QE can reduce this sampling cost by half while achieving an estimation error around 4×10^{-2} (Table 8.2), and QH can reduce this sampling cost by almost 90% while achieving an estimation error around 1×10^{-2} (Table 8.2), thus they both would be preferred over pure quasi-probability in practice following similar arguments in the $\mu = 1$ case. We also see that QH outperforms QE in terms of both sampling cost and estimation errors at $\mu = 2$, and thus would be preferred over QE. The cost of QE will only become lower than QH at $\mu = 3.9$, however at this point, neither of their sampling costs are likely to be practical.

8.6 Discussion

In this chapter, by introducing the concepts of group errors, we managed to prove that the change of the expectation value of a Pauli observable with increased Pauli noise strength can be approximated using multi-exponential decay, enabling us to extend exponential error extrapolation to multi-exponential extrapolation. We then performed 8-qubit numerical simulations using two different circuits for Fermi-Hubbard simulation under two different Pauli noise models, finding that the decay of their Pauli expectation values can all be fitted using single- or dual-exponential

curves, confirming our earlier proof of multi-exponential decay. Using the same circuits, we performed dual-exponential extrapolation by probing at four different error rates, which is minimal number of data points required, and managed to obtain low estimation errors of $\lesssim 10^{-4}$ for almost all 58 observables except for 2 fringe cases. In our simulations, the estimation errors of dual-exponential extrapolation are on average $50 \sim 100$ times lower than that of single-exponential extrapolation, with the maximum factor of error reduction reaching $\sim 10^4$.

We then proceeded to combine different error mitigation techniques in the context of well-characterised local Pauli noise. Instead of using quasi-probability to completely remove all the noise, we can use it to suppress the noise strength and perform error extrapolation, which is named 'quasi-probability with exponential extrapolation' (QE). Alternatively, we can use quasi-probability to remove the local undetectable noise and then perform symmetry verification. On top of that, instead of discarding all the circuit runs that fail the symmetry test, we have developed a new way to recombine the expectation values of the 'failed' and 'passed' runs to obtain an estimate of the noiseless observable, and we called the full combined method 'quasi-probability with hyperbolic extrapolation' (QH). Note that both QE and QH are free of the requirement to adjust the hardware error rate despite the name 'extrapolation'. By performing 8-qubit Fermi-Hubbard model simulations under local depolarising noise and using the fermionic number parity symmetry, we found that QH outperforms QE in terms of both estimation errors and sampling costs for almost all cases. When compared to pure quasi-probability, QH can achieve factor-of-4 and factor-of-9 sampling cost savings at the mean circuit error count $\mu = 1$ and $\mu = 2$, respectively, while still maintaining low estimation errors of $10^{-3} \sim 10^{-2}$. Hence, QH would outperform pure quasi-probability in our examples unless we obtain an impractical number of samples (more than millions) per observable.

QH is derived under the assumption that the observables decay along single-exponential curves with increased noise. Our simulation shows that QH can be robust against violation of this assumption when applied to dual-exponential observables. However, such robustness may not persist with a further increase in the number of

exponential components. A multi-exponential version of QH can be done through probing at more error rates and fitting Eq. (8.19) to the data. Alternatively, instead of probing at more error rates, we can also try to verify more symmetries. In such a way, we can obtain a set of expectation values corresponding to different verification syndromes for the multi-exponential version of the hyperbolic fitting. An example can be using the separate fermion number parity symmetries for each spin subspace, which will lead to expectation values corresponding to the four possible verification syndromes. However, how to recombine these expectation values in the case of multiple symmetries and how to use quasi-probability to transform the local error channels into the suitable forms for such a recombination is not a simple extension of the single-symmetry case we considered.

In our derivation, the number of exponential components in the expectation value decay curve in Eq. (8.11) is expected to scale exponentially with the number of gates. However, in our simulations, we fitted at most two exponential components for each of the observable decay curves. More analysis is needed to bridge the gap between the expected and the actual number of exponential components required, possibly based on the symmetry of the circuit. This will help us understand how the number of exponential components scales with the system size, enabling us to gauge the performance and the costs of scaling up the multi-exponential extrapolation method. It might be useful to draw ideas from non-Clifford randomized benchmarking [199–201], in which multi-exponential decay is also employed for the fitting of the fidelity curves. When applying multi-exponential extrapolation in practice, we might want to develop Bayesian methods to determine whether we need to probe at more error rates, which error rates to probe, and whether to change the number of exponential components of our fitted curve based on the existing data. This has been done in the context of randomized benchmarking [202] and it would be interesting to see its performance in the context of multi-exponential extrapolation.

One combination of error mitigation techniques that we have not explored here is pairing symmetry verification with error extrapolation without using quasi-probability. The naive version of such a combination was discussed in Chapter 7.

To make use of the results in this chapter, one possible way is to approximate all the local error channels as the compositions of detectable and undetectable error channels, so that we can deal with them separately using hyperbolic extrapolation and exponential extrapolation, respectively. It would be very interesting to see the implementation details of such a method and how it compares to pure error extrapolation.

We have only considered Pauli noise in this chapter, thus it will also be interesting to see whether our arguments can be extended to other error channels like amplitude damping or coherent errors. In practice, we can transform any error channels into Pauli channels using Pauli twirling (as shown in Chapter 3) and then apply our methods. Note that we can even perform further twirling like Clifford twirling to transform the error channels into group channels, which can be better mitigated as we have observed. Ways to transform a given error channel into a group channel can be an interesting area of investigation.

9 Conclusion

In the course of this thesis, we have developed and studied a range of practical schemes for the robust operations, implementations and applications of the emerging noisy quantum hardware.

In Chapter 3, we tried to mitigate the damage caused by coherent errors using twirling, which is a technique widely used for converting arbitrary error channels into Pauli channels by conjugating the error channel with the gates randomly chosen from the twirling set. Reducing the size of the twirling set might lead to simpler twirling gates and might enable the iteration over the full twirling set, removing the shot noise in twirling. We developed a scheme to construct the twirling set using the structure of the noise, whose size can be up to exponentially smaller than the conventional twirling set. We applied our techniques to the Steane code under global Z rotation noise and found that our twirling scheme can reduce the number of circuit runs by around 10 times compared to the conventional scheme. We also showed that twirling is equivalent to stabiliser measurements with discarded measurement results, which enables us to further reduce the size of the twirling set.

In Chapter 4, we showed that some of the coherence of a given coherent error channel can actually be used to improve its logical fidelity by simply sandwiching the noise with a chosen pair of Pauli gates, which we call Pauli conjugation. Using the optimal Pauli conjugation, we can achieve a higher logical fidelity than using

twirling and doing nothing. We devised a way to search for the optimal Pauli conjugation scheme and apply it to Steane code, 9-qubit Shor code and distance-3 surface code under global coherent Z noise. The optimal conjugation schemes show improvement in logical fidelity over twirling while the weights of the conjugation gates we need to apply are lower than the average weight of the twirling gates. In our example noise and codes, the concatenated threshold obtained using conjugation is consistently higher than the twirling threshold and can be up to 1.5 times higher than the original threshold where no mitigation is applied. Our simulations show that Pauli conjugation can be robust against gate errors. With the help of logical twirling, the undesirable coherence in the noise channel can be removed and the advantages of conjugation over twirling can persist as we go to multiple rounds of quantum error correction.

We then turned the hardware implementation of quantum error correction schemes in Chapter 5 and in the process we also looked at more hardware-specific errors like leakage errors. Leakage errors cannot be corrected by quantum error correction codes, making them potentially very damaging even when their probability is small. We proposed a surface code architecture for silicon quantum dot spin qubits that is robust against leakage errors by incorporating multi-electron mediator dots. Charge leakage in the qubit dots is transferred to the mediator dots via charge relaxation processes and then removed using charge reservoirs attached to the mediators. A stabiliser-check cycle, optimised for our hardware, then removes the correlations between the residual physical errors. Through simulations we obtained the surface code threshold for the charge leakage errors and show that in our architecture the damage due to charge leakage errors is reduced to a similar level to that of the usual depolarising gate noise. Our use of elongated mediator dots creates spaces throughout the quantum dot array for charge reservoirs, measuring devices and control gates, providing the scalability in the design.

Implementing quantum-error-corrected fault-tolerant algorithms will require large qubit overheads. As the advances in quantum hardware bring us into the

noisy intermediate-scale quantum (NISQ) era, we also begin to look for algorithms that can be implemented on the NISQ hardware without quantum error correction.

The Fermi-Hubbard variational quantum eigensolver (VQE) is one of the most promising NISQ algorithms due to its favourable circuit size scaling and its practical implications in areas like superconductivity. In Chapter 7, we outlined its implementation details about the gate sequence, the measurement scheme and the relevant error mitigation techniques. We performed resource estimation for both silicon spin qubits and superconducting qubits for a 50-qubit simulation, which cannot be solved exactly via classical means, and find similar results. The number of two-qubit gates required is on the order of 20000. Hence, to suppress the circuit error rate to a level such that we can obtain meaningful results with the aid of error mitigation, we need to achieve a two-qubit gate error rate of $\sim 10^{-4}$. When searching for the ground state, we need a few days for one gradient-descent iteration, which is impractical. This can be reduced to around 10 minutes if we distribute our task among hundreds of quantum processing units. Hence, the scalability of the hardware platform is essential to overcome the runtime issue via parallelisation. We found that implementing a 50-qubit Hubbard model VQE on a NISQ machine can be on the brink of being feasible in near term, but further improvements of the error mitigation schemes are crucial for its success.

In Chapter 8, we found one such improvement by extending exponential error extrapolation to multi-exponential error extrapolation and we provided a more rigorous proof for its effectiveness under Pauli noise. This was further validated via our numerical simulations, showing orders of magnitude improvements in the estimation accuracy over single-exponential extrapolation. Moreover, we developed methods to combine error extrapolation with two other error mitigation techniques: quasi-probability and symmetry verification. As shown in our simulation, our combined method can achieve low estimation errors with a sampling cost multiple times smaller than quasi-probability while without needing to be able to adjust the hardware error rate as required in canonical error extrapolation.

With the recent advance in the number of qubits and the level of control achieved by quantum hardware, two exciting goals lie ahead of us. One is the practical realisation of quantum error correction, which underpins the idea of fault-tolerant quantum computation and thus is fundamental to the implementation of all the well-known quantum algorithms that have profound and provable speed-ups. To this end, we need to develop schemes for combating dangerous errors arising in practice and for overcoming various hardware constraints. The other goal is the practical application of the NISQ computers that we will soon have, for which we need to work out the implementation details of the suitable tasks and develop quantum error mitigation schemes to fight the noise without incurring an unrealistic resource cost. This thesis has taken sizeable steps towards both of these goals and we hope that it can form the basis of the further advances to come. We started this thesis with a quote from Richard Feynman, and here we will end it with a poem from John Preskill, who is the current *Richard P. Feynman Professor* in Caltech:

Quantum's inviting,
just as Feynman knew.
The future's exciting,
if we see it through.
— *From "One Entangled Evening"*

Appendices

A

Appendices for Mitigating Coherent Noise: Constructing Smaller Pauli Twirling Sets

A.1 Twirling of Gate Noise

In real circuits, a noise operator is not a physical gate, hence it is impossible to bracket the noise operator with twirling gates. We can instead bracket the source of the noise with twirling gates..

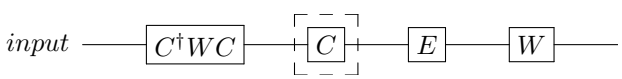
Suppose we want to apply a gate C, but an error E occurs after gate C with a probability p:

$$\mathcal{C}_e(\rho) = (1-p)\,\overline{C}\,\rho + p\,\overline{EC}\,\rho. \tag{A.1}$$

Now for the noisy part of the process $\overline{EC}\,\rho$, we want to twirl the noise E.

$$\begin{aligned}\mathcal{C}_e(\rho) &\xrightarrow{twirling} (1-p)\,\overline{C}\,\rho + p\mathcal{T}(\overline{E})\,\overline{C}\,\rho \\ &= (1-p)\,\overline{C}\,\rho + p\frac{1}{|\mathbb{W}|}\sum_{W\in\mathbb{W}}\overline{(WEW)\,C}\,\rho \\ &= (1-p)\,\overline{C}\,\rho + p\frac{1}{|\mathbb{W}|}\sum_{W\in\mathbb{W}}\overline{WEC\left(C^\dagger WC\right)}\,\rho\end{aligned} \tag{A.2}$$

which is just the following circuit:



Hence, by bracketing the erroneous gate C with the twirling gate W and its complementary gate $C^\dagger WC$, we are effectively twirling the noise E coming out of C.

Substituting Eq. (3.7) into Eq. (A.2) we have

$$\mathcal{C}_e(\rho) \xrightarrow{twirling} (1-p)\,\overline{C}\,\rho + \frac{p}{2^{2n}} \sum_{V\in\mathbb{V}} |\mathrm{Tr}(VE)|^2\,\overline{VC}\,\rho.$$

Hence, after twirling, $\mathcal{C}_e(\rho)$ becomes a error channel with Pauli error $V \in \mathbb{V}$ happening with the probability $\frac{p|\mathrm{Tr}(VE)|^2}{2^{2n}}$.

A.2 Requirements on the Twirling Set

To fully twirl the noise whose Pauli basis is $\mathbb{V}$, we need the twirling set $\mathbb{W}$ to satisfy Eq. (3.6), which is just Eq. (3.10) with the following **bijective** mappings:

$$\begin{aligned} \mathbb{W} &\mapsto \mathbb{Q} \\ \mathbb{V} &\mapsto \mathbb{H}_{\mathbb{V}} \subseteq \mathbb{H} \end{aligned}$$

which can be simplified to

$$\begin{aligned} \widetilde{\mathbb{W}} &\mapsto \mathbb{Q}_{\widetilde{\mathbb{W}}} \subseteq \mathbb{Q} \\ \widetilde{\mathbb{V}} &\mapsto \mathbb{H}_{\widetilde{\mathbb{V}}} \subseteq \mathbb{H}. \end{aligned}$$

Remember that $\mathbb{Q} = \left\langle \widetilde{\mathbb{Q}} \right\rangle$. If we want to find the **smallest** $\mathbb{W}$ that maps to $\mathbb{Q}$, the only way is to have $\mathbb{Q}_{\widetilde{\mathbb{W}}} = \widetilde{\mathbb{Q}}$ and $\mathbb{W} = \left\langle \widetilde{\mathbb{W}} \right\rangle$. Hence, our requirement on the twirling set becomes finding the following mappings

$$\begin{aligned} \widetilde{\mathbb{W}} &\mapsto \widetilde{\mathbb{Q}} \\ \widetilde{\mathbb{V}} &\mapsto \mathbb{H}_{\widetilde{\mathbb{V}}} \subseteq \mathbb{H}. \end{aligned} \tag{A.3}$$

The way to find such mapping is outlined in Section 3.3.3.

B

Appendices for Mitigating Coherent Noise: Pauli Conjugation

B.1 Construction of Error Generators

Recall that the error generators $\widetilde{\mathbb{E}}$ are just all the remaining generators needed to complement the stabiliser generators and the logical generators for generating the full Pauli gate set. The requirements of $\widetilde{\mathbb{E}}$ are

1. All of its elements are independent.

2. The size of $\widetilde{\mathbb{E}}$ is $|\widetilde{\mathbb{E}}| = |\widetilde{\mathbb{S}}|$.

3. The full set of elements that can be generated by $\widetilde{\mathbb{E}}$ does not contain any elements in $\mathbb{S}$ or $\overline{\mathbb{G}}$, otherwise we can replace the generators in $\widetilde{\mathbb{E}}$ with the elements in $\mathbb{S}$ or $\overline{\mathbb{G}}$.

The full Pauli set $\mathbb{G}$ can be generated using all the single physical qubit X and Z gate, after removing the elements that are dependent on each other through composition with stabiliser generators and/or logical generators, we will be left with the generating set $\widetilde{\mathbb{E}}$. Hence, we can always find a $\widetilde{\mathbb{E}}$ that consist of only single qubit X or Z operators. Here we will show how do we construct it.

Any practical stabiliser error correction code will be able to detect and correct all single qubit X and Z errors, hence all of these single-qubit errors will violate different subsets of stabiliser checks. The way we construct $\widetilde{\mathbb{E}}$ is:

1. Find all single-qubit X and Z errors that violate only one stabiliser check and add them to $\widetilde{\mathbb{E}}$. We will denote the set of stabiliser checks that they violate as $\mathbb{S}_E$.

2. Starting with $n = 2$, search in the checked physical qubits of the stabiliser checks in $\mathbb{S}_E$, there will be X or Z errors on these qubits that fail n stabiliser checks, with one and only one of the failed stabiliser checks *not* in $\mathbb{S}_E$. For each of such error we found, we will add it into $\widetilde{\mathbb{E}}$ and add the one additional violated stabiliser check into $\mathbb{S}_E$. Note that for each new element added into $\mathbb{S}_E$, we will have more physical qubits to check.

3. Repeat step 2 with n increasing by 1 in each iteration until $\mathbb{S}_E = \widetilde{\mathbb{S}}$, i.e. until $\mathbb{S}_E$ contains all the stabiliser checks (or equivalently until $|\widetilde{\mathbb{E}}| = |\widetilde{\mathbb{S}}|$).

In the case of topological code with boundaries, the above scheme is just starting by adding the stabiliser checks at the boundary into $\mathbb{S}_E$ and slowly progressing inwards, adding the inner stabiliser checks into $\mathbb{S}_E$ until all stabiliser checks are within $\mathbb{S}_E$.

In this way of construction, there is no way to find any composition of elements in $\widetilde{\mathbb{E}}$ such that there are no stabiliser checks fail, hence there is no way to compose stabilisers or logical operators out of these elements.

B.2 Construction of Twirling Set

With the twirling generators $\widetilde{\mathbb{W}}$ obtained in Section 4.3.2, we can now generate the full set of twirling gate $\mathbb{W}$. The elements in the twirling set $\mathbb{W}$ will correspond to the error operators that are detectable by our quantum error correction code and will all have different syndromes. For the purpose of twirling, we would want to replace these operators with the lowest weight error operators that produce the same syndrome (i.e. equivalent up to composition with stabilisers and logical

operators), since operators with lower weight will be easier to implement with fewer errors induced. These are usually just the recovery operators of the given syndromes, in such case we can just get them from the decoder. For a distance-d code, by definition all errors with weight $d-1$ or lower produce non-trivial error syndromes, and all errors with weight $\lfloor \frac{d-1}{2} \rfloor$ or below will have different syndromes (thus correctable). Hence, for all $W \in \mathbb{W}$ with weight $\lfloor \frac{d-1}{2} \rfloor$ or below, they are already the lowest weight operators that can produce the given syndrome, while for the others in $\mathbb{W}$ it may be possible to find a lower weight equivalence (not guaranteed to find since some correctable errors can be of higher weight than $\lfloor \frac{d-1}{2} \rfloor$, e.g. a surface code with very long Z boundaries and very short X boundaries).

B.3 Twirling Generators Reduction

Using $\coloneqq$ to denote 'super-super-operators':

$$\overline{\overline{A}}(\overline{C}) = \overline{A}\,\overline{C}\,\overline{A^\dagger}$$

we can rewrite that twirling process as:

$$\mathcal{T}(\mathcal{N}) = \frac{1}{|\mathbb{W}|}\sum_{W\in\mathbb{W}} \overline{\overline{W}}(\mathcal{N}) = \prod_{W\in\widetilde{\mathbb{W}}} \frac{I+\overline{\overline{W}}}{2}\mathcal{N}$$

$$\overline{\mathcal{N}}_T = \mathcal{R}\mathcal{T}(\mathcal{N}) = \mathcal{R}\prod_{W\in\widetilde{\mathbb{W}}} \frac{I+\overline{\overline{W}}}{2}\mathcal{N} \tag{B.1}$$

Here we have implicitly assumed that $\overline{\overline{W}}$ only acts on $\mathcal{N}$: $\overline{\overline{W}}\mathcal{N} = \overline{\overline{W}}(\mathcal{N})$.

Hence, the matrix elements for the twirled logical channel are

$$\overline{\mathcal{N}}_{T,GG'} = \frac{1}{2^{|\widetilde{\mathbb{G}}|}}\langle\langle \overline{G}\Pi_0|\mathcal{R}\prod_{W\in\widetilde{\mathbb{G}}}\left(I+\overline{\overline{W}}\right)\mathcal{N}|\overline{G}'\Pi_0\rangle\rangle$$

All Pauli super-operators commute, thus all Pauli super-super-operators also commute. Hence, we can arrange the order of the twirling generators in $\prod_{W\in\widetilde{\mathbb{G}}}\left(I+\overline{\overline{W}}\right)$ in any way we want. We will arrange it in the following way

$$\frac{1}{2^{|\widetilde{\mathbb{G}}|}}\prod_{W\in\widetilde{\mathbb{G}}}\left(I+\overline{\overline{W}}\right) = \frac{1}{2^{|\widetilde{\mathbb{G}}|}}\prod_{S\in\widetilde{\mathbb{S}}}\left(I+\overline{\overline{S}}\right)\prod_{\overline{G}\in\widetilde{\overline{\mathbb{G}}}}\left(I+\overline{\overline{G}}\right)\prod_{E_n\in\widetilde{\mathbb{E}}_n}\left(I+\overline{\overline{E}}_n\right)\prod_{E_c\in\widetilde{\mathbb{E}}_c}\left(I+\overline{\overline{E}}_c\right)$$

where $\widetilde{\mathbb{E}}_c$ is a subset of $\widetilde{\mathbb{E}}$ that acts trivially on noise $\mathcal{N}$ when used for twirling as discussed in Section 4.3.2.

Thus $\frac{1}{2^{|\widetilde{\mathbb{E}}_c|}} \prod_{E_c \in \widetilde{\mathbb{E}}_c} \left(I + \overline{\overline{E}}_c\right)$ will act trivially on $\mathcal{N}$ and can be absorbed by $\mathcal{N}$ when we put them closest to $\mathcal{N}$.

On the other hand, the twirling of $\widetilde{\mathbb{S}}$ and $\widetilde{\overline{\mathbb{G}}}$ will act trivially on the error correction code and the logical states (see Section 4.3.1), hence we can put them nearest to the logical states to remove them.

In Section 4.3.3, the equivalence of the twirling gates is obtained via interaction with both the noise elements and the logical states. To allow a generator to interact with both the noise elements and the logical states, we need to permute the symmetry operator of the noise elements or the logical states through the other generators, which will modify them. Hence instead of proving equivalence of generators, we expand out the product of generators to obtain a linear combination of the elements in the twirling set, and prove their equivalence instead.

B.4 Equivalence of Conjugation Gates due to Shared Symmetries between Codes and Noise

If all the code state basis $\Pi_{\vec{0}}\overline{G}$ and the physical noise channel $\mathcal{N}$ are invariant under the Clifford transformation U:

$$\left[U, \Pi_{\vec{0}}\overline{G}\right] = 0 \quad \forall G \in \mathbb{G}$$
$$\left[\overline{U}, \mathcal{N}\right] = 0$$

then the recovery channel $\mathcal{R}$ will also be invariant under the same transformation since it is completely based on the code and the error channel: $\left[\overline{U}, \mathcal{R}\right] = 0$.

Hence, we have:

$$\left[U, \Pi_{\vec{0}}\overline{G}\right] = 0 \quad \forall G \in \mathbb{G}$$
$$\left[\overline{U}, \mathcal{N}\right] = 0, \ \left[\overline{U}, \mathcal{R}\right] = 0 \Rightarrow \left[\overline{U}, \mathcal{R}\mathcal{N}\right] = 0$$

Thus

$$U\Pi_{\vec{0}}\overline{G}U^\dagger = \Pi_{\vec{0}}\overline{G} \quad \Rightarrow \quad \overline{U}|\Pi_{\vec{0}}\overline{G}\rangle\rangle = |\Pi_{\vec{0}}\overline{G}\rangle\rangle \quad \forall G \in \mathbb{G}$$
$$\overline{U}\,\mathcal{R}\mathcal{N}\,\overline{U^\dagger} = \mathcal{R}\mathcal{N}$$

Since $\overline{W}$ and $\mathcal{R}$ are both Pauli channel, they commutes, hence

$$\begin{aligned}\overline{\mathcal{N}}(W)_{G,G'} &= \langle\langle\Pi_{\vec{0}}\overline{G}|\mathcal{R}\,\overline{W}\,\mathcal{N}\,\overline{W}|\Pi_{\vec{0}}\overline{G}'\rangle\rangle \\ &= \langle\langle\Pi_{\vec{0}}\overline{G}|\,\overline{W}\,\mathcal{R}\mathcal{N}\,\overline{W}|\Pi_{\vec{0}}\overline{G}'\rangle\rangle \\ &= \langle\langle\Pi_{\vec{0}}\overline{G}|\,\overline{U^\dagger}\,\overline{W}\,\overline{U}\,\mathcal{R}\mathcal{N}\,\overline{U^\dagger}\,\overline{W}\,\overline{U}|\Pi_{\vec{0}}\overline{G}'\rangle\rangle\end{aligned}$$

Since U is Clifford, $U^\dagger WU$ is Pauli, hence $\overline{U^\dagger WU}$ commute with $\mathcal{R}$:

$$\begin{aligned}\overline{\mathcal{N}}(W)_{G,G'} &= \langle\langle\Pi_{\vec{0}}\overline{G}|\mathcal{R}\,\overline{U^\dagger WU}\,\mathcal{N}\,\overline{U^\dagger WU}|\Pi_{\vec{0}}\overline{G}'\rangle\rangle \\ &= \overline{\mathcal{N}}(U^\dagger WU)_{G,G'}\end{aligned}$$

B.5 Shape of Fidelity Curve under Global Z Rotation

B.5.1 Rotational symmetry of the fidelity curve

For the noise model in Eq. (4.6) and for all the codes we considered which have global logical Z gates, we have:

$$N(\frac{\pi}{2}) = \prod_{j=1}^{J}(-iZ_j) \equiv \overline{Z}$$

For the worst case fidelity Q for such pure Z noise, we will start and measured in the logical $|+_L\rangle$ eigenstate:

$$\begin{aligned}Q(\theta) &= |\langle+_L|\,N(\theta)\,|+_L\rangle|^2 \\ &= |(\langle+_L|\,N(\theta)\,|+_L\rangle)^*|^2 \\ &= |\langle+_L|\,N(-\theta)\,|+_L\rangle|^2\end{aligned}$$

$$\begin{aligned}
Q(\frac{\pi}{2}-\theta) &= \left|\langle +_L| N(\frac{\pi}{2}-\theta) |+_L\rangle\right|^2 \\
&= \left|\langle +_L| N(\frac{\pi}{2})N(-\theta) |+_L\rangle\right|^2 \\
&= |\langle -_L| N(-\theta) |+_L\rangle|^2 \\
&= 1 - |\langle +_L| N(-\theta) |+_L\rangle|^2 \\
&= 1 - Q(\theta)
\end{aligned}$$

For fidelity of one qubit, we have:

$$F(\theta) = \frac{2}{3}Q(\theta) + \frac{1}{3}$$

Hence, we have:

$$\begin{aligned}
F(\frac{\pi}{2}-\theta) &= \frac{2}{3}(1 - Q(\theta)) + \frac{1}{3} \\
&= \frac{4}{3} - \frac{2}{3}Q(\theta) - \frac{1}{3} \\
&= \frac{4}{3} - F(\theta)
\end{aligned}$$

Hence, the logical fidelity curve $F(\theta)$ is rotationally symmetry about a point at $\theta = \frac{\pi}{4}$

B.5.2 Fidelity at $\theta = \frac{\pi}{4}$

For the noise model in Eq. (4.6), at $\theta = \frac{\pi}{4}$ we have:

$$N(\frac{\pi}{4}) = \frac{1}{\sqrt{2^J}} \prod_{j=1}^{J} (I - iZ_j) \tag{B.2}$$

For an operator U consist of tensor product of single-qubit Z, we will write the set of qubit index that we apply Z gate on as $\vec{U}$:

$$U = \prod_{i \in \vec{U}} Z_i$$

Hence, the terms in the expansion of Eq. (B.2) will be $(-i)^{|\vec{U}|} \prod_{i\in\vec{U}} Z_i = (-i)^{|\vec{U}|}U$. In the case of measuring the zero syndrome, $N(\frac{\pi}{4})$ will collapse into a superposition of stabilisers and Z logical operators. For each stabiliser term $(-i)^{|\vec{S}|}S$ in the

expansion, there will be a corresponding logical Z operator term differed by apply Z to all qubits: $(-i)^{J-|\vec{S}|}S\left(\prod_j Z_j\right) = (-i)^{J-|\vec{S}|}S\overline{Z}$, hence the terms in the expansion correspond to zero syndrome is:

$$\sum_S (-i)^{|\vec{S}|}S + \overline{Z}\left(\sum_S (-i)^{J-|\vec{S}|}S\right) = \sum_S (-i)^{|\vec{S}|}S\left[I + (-i)^{J-2|\vec{S}|}\overline{Z}\right]$$
$$\equiv \sum_S \left[\overline{I} + (-i)^{J-2|\vec{S}|}\overline{Z}\right]$$

If all stabilisers have even weights, i.e. $|\vec{S}| = 2n$, then $2|\vec{S}| = 4n$, hence $(-i)^{J-2|\vec{S}|} = (-i)^J(-i)^{-2|\vec{S}|} = (-i)^J$.

Thus if all the stabilisers have even weights, and the logical Z operator consist of applying Z to all the qubits, then the terms in the expansion of Eq. (B.2) is:

$$|\mathbb{S}|\left[\overline{I} + (-i)^J\overline{Z}\right]$$

and similarly for other syndromes. Note that terms of other syndromes will also result in the same amplitude $|\mathbb{S}|$. Hence, we have the same probability of collapse into any syndrome. For odd number of qubits J, we then have a logical Z rotation of the angle $\frac{\pi}{2}$ (or $-\frac{\pi}{2}$), which means a worst case fidelity of $\frac{1}{2}$ and an average fidelity of $\frac{2}{3}\times\frac{1}{2}+\frac{1}{3} = \frac{2}{3}$.

B.6 Equivalent Conjugation Classes for Codes under Global Z Rotation

Recalled the arguments in Section 4.4. We can make the following simplification based on the codes and the noise model we consider.

- Our noise model is global, thus have all possible qubit permutation symmetry. Along with the fact that our codes have one logical qubit and global Pauli gates mean that we only need to consider the permutation symmetry of the stabilisers when we try to reduce the twirling set.

- The noise model is pure Z noise, thus all Z twirling generators can be removed. All Z stabiliser checks will also have trivial effects (besides the five-qubit code which does not have pure Z checks), thus we only need to consider the permutation symmetry of the X stabilisers.

B.6.1 Five-qubit code

In five-qubit code, the generators are

- Stabiliser generators $\widetilde{\mathbb{S}}$: $XZZXI$ and three of its cyclic permutations $IXZZX$, $XIXZZ$, $ZXIXZ$
- Logical generators $\widetilde{\widetilde{\mathbb{G}}}$: X or Z on all qubits.

Following Section 4.3.1, we can construct the twirling generators:

$$\widetilde{\mathbb{W}} = \widetilde{\mathbb{E}} = \{X_1, X_2, Z_3, Z_5\}$$

As mentioned above, the Z twirling generators can be safely removed since we have pure Z noise:

$$\widetilde{\mathbb{W}} = \{X_1, X_2\}$$

which generates the twirling set:

$$\mathbb{W} = \{I, X_1, X_2, Z_4\}$$

Here we have transform the error operators X_1X_2 to its lowest weight equivalence with the same error syndromes Z_4. Conjugating the noise with Z_4 has trivial effect since we have pure Z noise.

The five-qubit code has cyclic permutation symmetry (there are also additional symmetries that we do not need to use here [18]). As discussed in Section 4.3.2, using these symmetry transformation, we can easily prove that conjugating the noise with X_1 is equivalent to X_2 since $X_2 = U^\dagger X_1 U$ where U is one of the qubit cyclic permutation operator.

Hence, there are two equivalent class of twirling gate, one is equivalent to I, while the other is equivalent to X_1 (or any single-qubit X gate by cyclic permutation).

B.6.2 Nine-qubit Shor code

With Local Z checks

As shown in Fig. 4.4, in nine-qubit Shor Code, the generators are

- Stabiliser generators $\widetilde{\mathbb{S}}$: $\{Z_iZ_{i+1} \mid i \in \{1,2,4,5,7,8\}\}$ and $\{\prod_{j=0}^{5} X_{i+j} \mid i \in \{1,4\}\}$
- Logical generators $\widetilde{\mathbb{G}}$: X or Z on all qubits.

Following Section 4.3.1, we can construct our twirling generators:

$$\widetilde{\mathbb{W}} = \widetilde{\mathbb{E}} = \{X_1, X_3, X_4, X_6, X_7, X_9, Z_1, Z_7\}$$

As mentioned above, the Z twirling generators can be safely removed since we have pure Z noise:

$$\widetilde{\mathbb{W}} = \{X_1, X_3, X_4, X_6, X_7, X_9\}$$

When looking at the Z stabiliser checks, which will produce the syndromes for these X error operators, we realise they are divided into 3 non-overlapping set (no shared checked qubits), which are individual rows in Fig. 4.4. All the error syndromes within each row can be produced by single-qubit X errors within that row. The Z-check syndrome of different rows are independent of each other since they do not share any qubits. Hence, to produce all possible syndromes using error operators with the lowest weight, we will have zero or one single-qubit X errors in each row. This set of error operators will be the full twirling set W that is generated.

Permutation symmetries that exist in the 9-qubit Shor code will be any permutation of the elements within each row and any permutation between the rows shown in Fig. 4.4. In such a case, all the operators with the same weight in our twirling set can be shown to be equivalent, leaving us with the following four equivalent classes of twirling operators.

- Identity: I
- Single qubit flip: X_1

- Two-qubit flip (in different row): X_1X_4
- Three-qubit flip (in different row): $X_1X_4X_7$

With Local X checks

It is still a nine-qubit Shor code, with a swap between the X and Z stabilisers.

Following Section 4.3.1, we can construct our twirling generators:

$$\widetilde{\mathbb{W}} = \widetilde{\mathbb{E}} = \{Z_1, Z_3, Z_4, Z_6, Z_7, Z_9, X_1, X_7\}$$

As mentioned above, the Z twirling generators can be safely removed since we have pure Z noise:

$$\widetilde{\mathbb{W}} = \{X_1, X_7\}$$

which generates the twirling set:

$$\mathbb{W} = \{I, X_1, X_4, X_7\}$$

Here we have transform the error operators X_1X_7 to its lowest weight equivalence with the same error syndromes X_4.

We have the same code symmetry as the other Shor code, which allows us to prove the equivalence of conjugation with X_1, X_4 and X_7.

Hence, there are two equivalent class of twirling gate, one is equivalent to I, while the other is equivalent to X_1 (or any single-qubit X gate).

B.6.3 Distance-3 surface code

The stabiliser generators of the distance-3 surface code is shown in Fig. 4.6.

Following Section 4.3.1, we can construct our twirling generators:

$$\widetilde{\mathbb{W}} = \widetilde{\mathbb{E}} = \{X_1, X_3, X_7, X_9, Z_1, Z_3, Z_7, Z_9\}$$

As mentioned above, the Z twirling generators can be safely removed since we have pure Z noise:

$$\widetilde{\mathbb{W}} = \{X_1, X_3, X_7, X_9\}$$

which generates the twirling set:

- Weight-1: X_1, X_2, X_3, X_5, X_7, X_8, X_9
- Weight-2: X_2X_7, X_1X_7, X_1X_8, X_1X_9, X_2X_9, X_3X_9, X_3X_8, X_2X_8

The logical Pauli gates of the code are just applying the corresponding Pauli gates to all the physical qubits (because it is an odd distance surface code), hence we only need to look at the symmetry of its stabilisers as discussed in Section 4.3.3. We can group the operators that are equivalent due to the rotational symmetry of the code: (X_1, X_9), (X_2, X_8), (X_3, X_7), (X_5), (X_2X_7, X_3X_8), (X_1X_7, X_3X_9), (X_1X_8, X_2X_9), (X_1X_9), (X_2X_8).

Since we have only pure Z noise, we only need to look at the symmetry exists in the X stabilisers, leading to additional symmetry in the exchange between qubits (1, 2) and between qubits (8, 9). Applying on top of the rotational symmetry, we have the following classes of equivalent conjugations:

- I
- X_1, X_2, X_8, X_9
- X_3, X_7
- X_5
- X_1X_7, X_3X_9, X_2X_7, X_3X_8
- X_1X_9, X_2X_8, X_1X_8, X_2X_9

B.7 Effective Z Logical Channel Conditioned on Syndrome

When we expand the physical noise $e^{-i\theta\sum_j Z_j}$ into the sum of tensor products of Z, all the odd-weight term will form the imaginary part, while all even-weight terms will form the real part. Hence, when we flip the sign of θ, which is equivalent to taking the complex conjugate of our noise channel, all the odd-weight terms

(the imaginary part) will flip their signs while all the even-weight terms (the real part) will remain the same.

By saying the operator is real (imaginary) here, we mean that the operator has real (imaginary) amplitude. Since we are only considering Z noise, composing two Z operators together will not lead to any extra phase factor i. Thus when we compose an imaginary Z operator with a real Z operator, we will get an imaginary Z operator, while composing two imaginary or two real operators will give a real Z operator.

For all the codes we consider here, they have stabiliser generators that are all even-weight, which means that they are all real in the noise expansion. For our codes, we can find one of the logical operators $\overline{Z}$ is odd-weight, which corresponds to imaginary amplitude in the expansion, thus all $\overline{Z}$ are imaginary since they can be obtained by composing the imaginary $\overline{Z}$ with the real stabilisers. Hence, when we measured the 0 syndrome, the noise is collapsed into a coherent superposition of $\overline{I}$ with real amplitude and $\overline{Z}$ with imaginary amplitude. When we flip the sign of the physical error angle θ, it is the same as taking the complex conjugate, which will flip the sign of $\overline{Z}$ since its amplitude is pure imaginary. For other syndromes with error E, we still have one of $E\overline{Z}$ and $E\overline{I}$ being one real and the other being imaginary and hence similar argument follows.

For codes with odd-weight stabiliser generators, we will have a complex amplitude (mix of real and imaginary) for $\overline{I}$. On the other hand, for codes with even-weight logical operators (even distance code), it will give real $\overline{Z}$ for real $\overline{I}$ and imaginary $\overline{Z}$ for imaginary $\overline{I}$. In both case, since the phase of $\overline{I}$ and $\overline{Z}$ are no longer guaranteed to be differed by i, the logical channel for a given syndrome can no longer be written as a logical $\overline{Z}$ rotation, but instead a combination of logical $\overline{Z}$ rotation and logical dephasing channel. This was observed by Huang *et. al.* [18] for even-distance repetition code and distance-4 surface code.

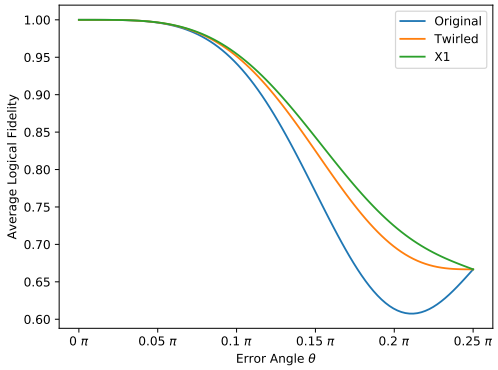


Figure B.1: Logical fidelity of the nine-qubit shor code with local X checks under difference noise strength and noise tailoring schemes.

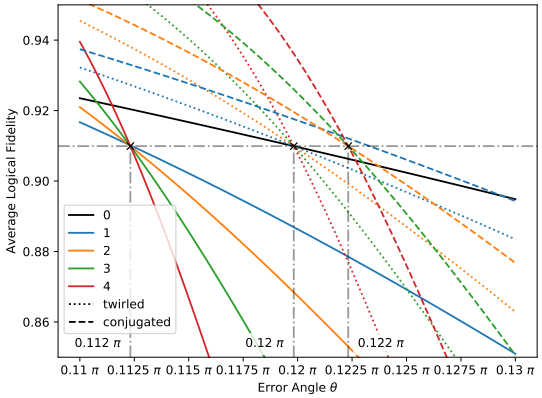


Figure B.2: The concatenated threshold plot under global Z rotation noise for the 9-qubit Shor code with local X checks. Different colours show different levels of concatenation while different line styles show applying different strategies like twirling or Pauli conjugation to the noise. The Pauli conjugations we used here are the optimal schemes that we found with zero gate error.

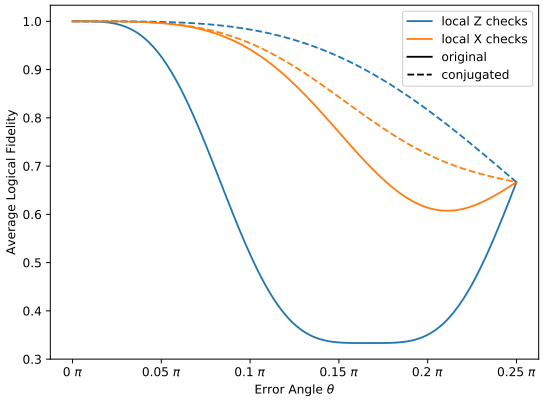


Figure B.3: Comparison between Shor code with local X checks and local Z checks with Pauli conjugation and without. The Pauli conjugations we used here are the optimal schemes that we found with zero gate error.

B.8 Pauli Conjugation for the Other 9-qubit Shor Code

In the main text, we have only shown results for the 9-qubit Shor code local Z checks. For the 9-qubit Shor code with local X check, its average fidelity of different conjugation schemes and its concatenated threshold plots are shown in Fig. B.1 and Fig. B.2.

For the Z noise we considered, the 9-qubit Shor code that has local X checks will actually have better performance since it has more X checks which are sensitive to Z noise. This is shown by the large gap between the two original fidelity curve in Fig. B.3. However, after using conjugation to tailor the noise to fit the code, the Shor code with local Z checks receives a huge boost in fidelity such that it even exceeds the fidelity of the other Shor code with conjugation. This further exemplifies the power of Pauli conjugation when there is a misfit between the code and the noise.

B.9 Detailed Circuit for the Codes

Here in Fig. B.4, Fig. B.5 and Fig. B.6, we outline the encoding circuits and the parity check circuits we used for our codes.

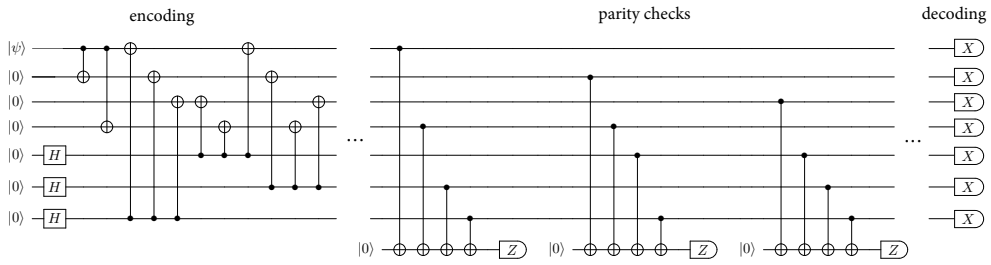


Figure B.4: The encoding and the parity check circuit for the Steane code.

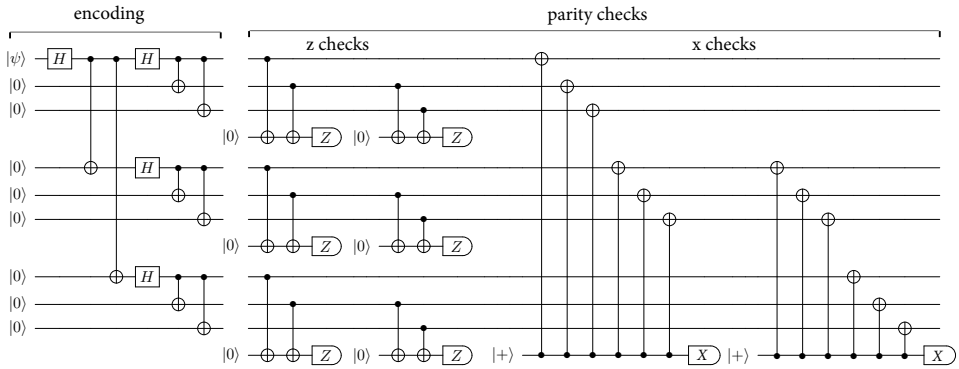


Figure B.5: The encoding and the parity check circuit for the Shor code.

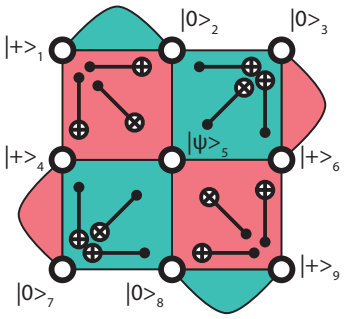


Figure B.6: The encoding circuit for the Surface code. The parities are checked using similar circuits as our Steane code circuit and Shor code circuit, with one ancilla per parity check and using CNOT for the interaction between data and ancilla.

B.10 Multiple Rounds of Error Correction under Global Z Rotation

B.10.1 Without logical twirling

All the codes that we have considered in this section have one logical qubit and transversal Z gates. When they undergo coherent Z noise, the effective logical error for a given measured syndrome $\vec{m}$ after correction will be a logical Z rotation of angle $\theta_{\vec{m}}$. Hence, the effective logical error channel averaged over all the syndrome measurements is:

$$\overline{\mathcal{N}}_0 = \sum_{\vec{m}} p_{\vec{m}} \, \overline{\overline{Z}(\theta_{\vec{m}})} \tag{B.3}$$

The Pauli transfer matrix of $\overline{Z}(\theta)$ is

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

with the eigenvalues $1, 1, e^{-i\theta}, e^{i\theta}$, and the same eigenvectors independent of θ. Hence, $\overline{\mathcal{N}}_0$ will have the same eigenvectors with the eigenvalues $1, 1, \sum_{\vec{m}} p_{\vec{m}} e^{-i\theta_{\vec{m}}}, \sum_{\vec{m}} p_{\vec{m}} e^{i\theta_{\vec{m}}}$.

Hence, after k round of error correction, the logical fidelity is:

$$F(\overline{\mathcal{N}}_0^k) = \frac{\frac{1}{2}\operatorname{Tr}\left\{\overline{\mathcal{N}}_0^k\right\} + 1}{3} = \frac{\operatorname{Re}\left\{\left(\sum_{\vec{m}} p_{\vec{m}} e^{-i\theta_{\vec{m}}}\right)^k\right\} + 2}{3} \tag{B.4}$$

If we twirl the Z noise channel, then we will have a logical dephasing channel instead. Thus $\overline{\mathcal{N}}_T$ is a diagonal matrix with eigenvalues: $1, 1-2p_d, 1-2p_d, 1$ where p_d is the dephasing probability.

Hence, for k round of error correction (each round undergo the same noise as before), the logical fidelity is:

$$F(\overline{\mathcal{N}}_T^k) = \frac{\frac{1}{2}\operatorname{Tr}\left\{\overline{\mathcal{N}}_T^k\right\} + 1}{3} = \frac{(1-2p_d)^k + 2}{3} \tag{B.5}$$

Using these formulae, in Fig. B.7 we have plotted the logical fidelity of different schemes for different codes after 100 cycles; in each cycle, error correction follows a period of exposure to the environment which induces global Z rotation with angle θ. In all codes, we can see the improvements of the Pauli conjugation schemes and the twirling scheme over doing nothing in small θ. In Shor code and surface code, we can see the advantage of our optimal conjugation scheme over twirling still remains for *small error angle θ in each round* even after 100 rounds. Nevertheless, we can see that in many cases twirling becomes superior to even our best Pauli conjugation after sufficient cycles have occurred; fortunately this can be entirely remedied by an adaption we term 'logical twirling'.

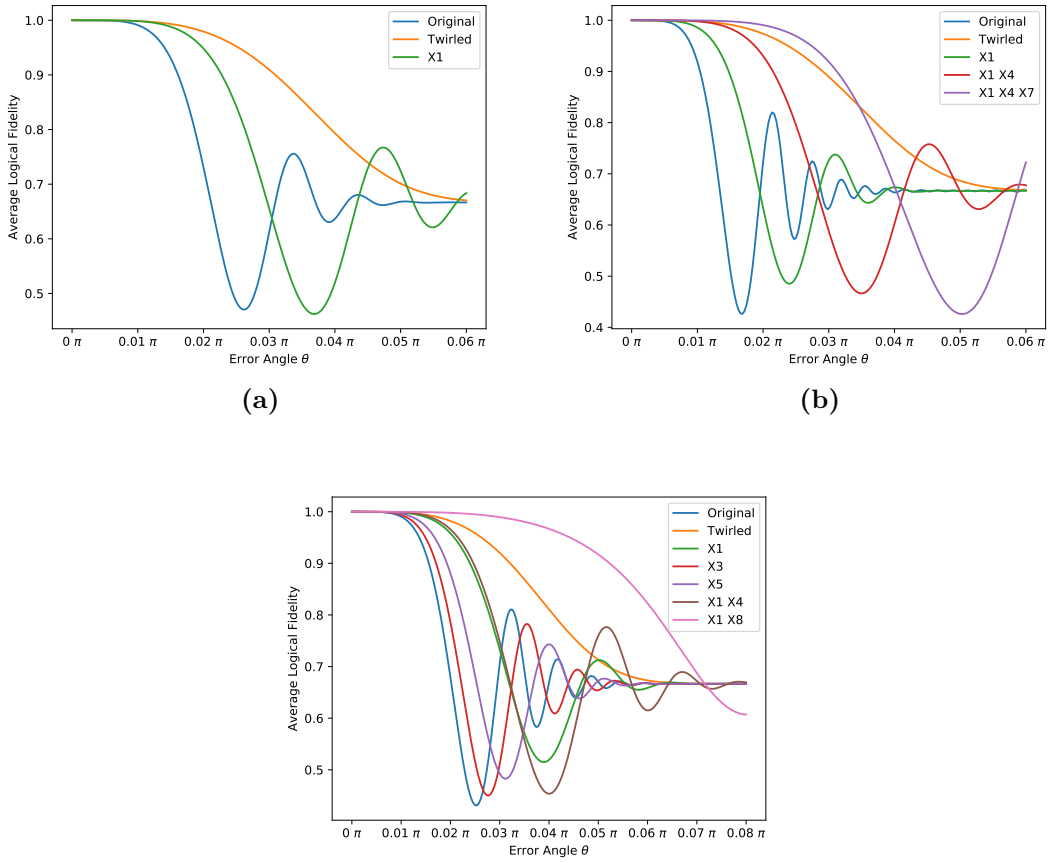


Figure B.7: Logical fidelity after 100 cycles of noise with quantum error correction using different strategies for (a) Steane code, (b)nine-qubit Shor code and (c) distance-3 surface code. Within *each* round the noise is the same coherent rotation of the strength θ. These figure illustrate the issue that is tackled through 'logical twirling' as we explain Section 4.5.

B.10.2 With logical twirling

The logically twirled version of the noise channel $\overline{\mathcal{N}}_0$ described in Eq. (B.3) is a logical dephasing channel of the form

$$\overline{\mathcal{N}}_{0,LT} = \sum_{\vec{m}} p_{\vec{m}} \left(\cos^2\left(\frac{\phi_{\vec{m}}}{2}\right) \overline{\overline{I}} + \sin^2\left(\frac{\phi_{\vec{m}}}{2}\right) \overline{\overline{Z}} \right) \tag{B.6}$$

The corresponding conjugated noise channel $\overline{\mathcal{N}}_c$ will be in a form similar to Eq. (B.3) with different $\phi_{\vec{m}}$. Applying logical twirling on top of conjugation will lead to the channel $\overline{\mathcal{N}}_{c,LT}$. Recall that the corresponding physically twirled noise channel is denoted as $\overline{\mathcal{N}}_T$.

Our previous simulations for one round of error correction show that:

$$F(\overline{\mathcal{N}}_c) \geq F(\overline{\mathcal{N}}_T) \geq F(\overline{\mathcal{N}}_0).$$

Since logical twirling will not change the logical fidelity (since the eigenvalues of the Pauli transfer matrices are not affected), we have:

$$F(\overline{\mathcal{N}}_{c,LT}) \geq F(\overline{\mathcal{N}}_T) \geq F(\overline{\mathcal{N}}_{0,LT}). \tag{B.7}$$

in which $\overline{\mathcal{N}}_{c,LT}$, $\overline{\mathcal{N}}_T$ and $\overline{\mathcal{N}}_{0,LT}$ are all logical dephasing channel with different dephasing probability p_d.

For a single-qubit dephasing channel $\overline{\mathcal{N}}_d$ with dephasing probability p_d, the eigenvalues of its Pauli transfer matrix will be $\{\lambda_i\} = \{1, 1-2p_d, 1-2p_d, 1\}$. Hence, the logical fidelity of k rounds of $\overline{\mathcal{N}}_d$ is [203, 204]:

$$\begin{aligned} F(\overline{\mathcal{N}}_d^k) &= \frac{\mathrm{Tr}\left\{\overline{\mathcal{N}}_d^k\right\} + 2}{6} \\ &= \frac{\sum_i \lambda_i^k + 2}{6} \\ &= \frac{(1-2p_d)^k + 2}{3} \end{aligned}$$

Thus we have:

$$F(\overline{\mathcal{N}}_d) \geq F(\overline{\mathcal{N}}'_d) \Rightarrow F(\overline{\mathcal{N}}_d^k) \geq F(\overline{\mathcal{N}}_d'^k) \quad \forall k \in \mathbb{Z}_+$$

Combining with Eq. (B.7), we then have:

$$F(\overline{\mathcal{N}}_{c,LT}^k) \geq F(\overline{\mathcal{N}}_T^k) \geq F(\overline{\mathcal{N}}_{0,LT}^k).$$

for any positive integer k. Hence, with the help of logical twirling, the improvement of logical fidelity using Pauli conjugation over twirling (or doing nothing) with single-round of error correction in global Z rotation will indeed persist when we go to multiple rounds of error corrections.

B.10.3 Random walk noise model

Up to now, we have only considered the case in which the global Z rotations in each round of error correction are rotations of the same angle in the same direction. In practice, for such a noise model, all we need to do is flip all the qubits right in the middle of the whole process which flips the direction of the rotation and cancels the coherent error. This is just a simple case of dynamical decoupling.

It may be interesting to look at the other extreme in which the error channel is a random walk. Within each round of error correction, there is a equal probability of positive or negative rotation of angle θ: $N(\pm\theta) = e^{\pm i\theta \sum_j Z_j}$.

A a global Z rotation $e^{-i\theta \sum_j Z_j}$ will lead to an effective logical error channel as described in Eq. (B.3):

$$\overline{\mathcal{N}}_0 = \sum_{\vec{m}} p_{\vec{m}} \overline{\overline{Z}(\phi_{\vec{m}})}$$

When the sign of rotation of the physical error θ is flipped, the sign of the logical rotation $\phi_{\vec{m}}$ will also be flipped for all the codes that we are considering (see Appendix B.7). For each time step, since we have equal probabilities of positive and negative physical rotations, we also have equal probabilities of positive and negative logical rotations, leading to the effective logical channel:

$$\overline{\mathcal{N}}_{0,\pm} = \sum_{\vec{m}} p_{\vec{m}} \left(\cos^2\left(\frac{\phi_{\vec{m}}}{2}\right) \overline{\overline{I}} + \sin^2\left(\frac{\phi_{\vec{m}}}{2}\right) \overline{\overline{Z}} \right)$$

which is just the logically twirled channel $\overline{\mathcal{N}}_{0,LT}$. Hence, for such a random walk noise model, the logical channel is already logically twirled and we just need to apply conjugation to it to reduce the effect of the noise.

B.11 Conjugating High Frequency Noise

For a coherent noise:

$$U(\theta) = e^{-iHt},$$

the Hamiltonian H can be broken down into its Pauli basis $\mathbb{G}_H$:

$$H = \sum_{g_i \in \mathbb{G}_H} \beta_i g_i$$

Note that β_i are real since H is Hermitian.

Now we define the magnitude of H to be E, and the normalised version of H to be h where:

$$E = \sqrt{\sum_i \beta_i^2} \tag{B.8}$$

$$h = \frac{H}{E} = \sum_{g_i \in G_H} \frac{\beta_i}{E} g_i = \sum_{g_i \in G_H} \alpha_i g_i \tag{B.9}$$

for $\alpha_i = \frac{\beta_i}{E}$ and $\sum_i \alpha_i^2 = 1$.

Now the evolution operator is just:

$$U(t) = e^{-iHt} = e^{-ihEt}$$

$$U(\theta) = e^{-i\theta h}$$

with $\theta = Et$.

Suppose our noise channel is some high frequency noise that is only coherent for a very short amount of time δt, resulting in a rotation angle of $\epsilon = E\delta t$:

$$\begin{aligned} U(\pm\epsilon) &= e^{\pm i\epsilon h} \\ &\approx I \pm i\epsilon h - \frac{\epsilon^2}{2} h^2 \\ &= I - \frac{\epsilon^2}{2} h^2 \pm i\epsilon h \end{aligned}$$

Within each time period δt, the coherent noise will have a 50-50 chance for rotations in the positive and negative directions, just like a random walk. Thus the effective channel over a time period δt is just:

$$\begin{aligned} \mathcal{U}_\epsilon(\rho) &= \frac{1}{2} U(\epsilon)\rho U^\dagger(\epsilon) + \frac{1}{2} U(-\epsilon)\rho U^\dagger(-\epsilon) \\ &= \left(I - \frac{\epsilon^2}{2} h^2\right) \rho \left(I - \frac{\epsilon^2}{2} h^2\right) + \epsilon^2 h\rho h \end{aligned} \tag{B.10}$$

which in the Pauli transfer matrix formalism is just:

$$\mathcal{U}_\epsilon|\rho\rangle\rangle = \left(\overline{I - \frac{\epsilon^2}{2}h^2} + \epsilon^2\,\overline{h}\right)|\rho\rangle\rangle$$

Composing N of such channels together we have:

$$\begin{aligned}
\mathcal{U}_\epsilon^N|\rho\rangle\rangle &= \left(\overline{I - \frac{\epsilon^2}{2}h^2} + \epsilon^2\,\overline{h}\right)^N |\rho\rangle\rangle \\
&\approx \left(\left(\overline{I - \frac{\epsilon^2}{2}h^2}\right)^N + N\epsilon^2\,\overline{h}\left(\overline{I - \frac{\epsilon^2}{2}h^2}\right)^{N-1}\right)|\rho\rangle\rangle \\
&= \left(\overline{\left(I - \frac{\epsilon^2}{2}h^2\right)^N} + N\epsilon^2\,\overline{h}\,\overline{\left(I - \frac{\epsilon^2}{2}h^2\right)^{N-1}}\right)|\rho\rangle\rangle \\
&\approx \left(\overline{I - \frac{N\epsilon^2}{2}h^2} + N\epsilon^2\,\overline{h}\right)|\rho\rangle\rangle \\
&= \mathcal{U}_{\sqrt{N}\epsilon}|\rho\rangle\rangle
\end{aligned}$$

Now if H (and thus h) contains coherent superposition of multiple Pauli components, then as discussed in Section 4.2.3, conjugation can be used to improve the logical fidelity of the channel by changing the way these components interfere. In particular, if a conjugation scheme works for the channel $\mathcal{U}_\epsilon$, then the same scheme should also work for the composite channel $\mathcal{U}_\epsilon^N \approx \mathcal{U}_{\sqrt{N}\epsilon}$ since their Pauli components interfere in similar ways (as can be seen from their similar structural dependence on h). In the case of the global Z rotation that we considered in Section 4.4, we have discussed why conjugation would work for a single step of the random walk channel $\mathcal{U}_\epsilon$ in Appendix B.10.3 (in which the channel is denoted as $\overline{\mathcal{N}}_{0,\pm}$). Hence, by the arguments above, the same conjugation scheme will also work for the composite channel which corresponds to high frequency global Z noise.

B.12 Multi-round Twirling Set Reduction

The effective error channel with K rounds of twirling is:

$$(\overline{\mathcal{N}}_{TK})_{G,G'} = \langle\langle \Pi_{\vec{0}}\overline{G}|\mathcal{R}\left[\prod_{k=1}^{K}\mathcal{T}(\mathcal{N})\right]|\Pi_{\vec{0}}\overline{G}'\rangle\rangle$$

The argument about structure of noise (Section 4.3.2) can still be applied to the twirling within each individual rounds here, giving us a smaller set of twirling generators $\widetilde{\mathbb{W}}$, from which we can obtained a reduced twirling set $\mathbb{W}$:

$$(\overline{\mathcal{N}}_{TK})_{G,G'} = \frac{1}{|\mathbb{W}|^K} \sum_{\vec{W}\in\mathbb{W}^K} \langle\langle \Pi_{\vec{0}}\overline{G}|\mathcal{R} \prod_{k=1}^{K} \overline{W}_k \mathcal{N} \overline{W}_k |\Pi_{\vec{0}}\overline{G}'\rangle\rangle$$

Since we are summing all possible $\vec{W}$ and the twirling set is a group on super-operator composition, we can do the following change of variables: $\overline{W}_k\overline{W}_{k+1} \Rightarrow \overline{W}_{k+1}$, which gives:

$$(\overline{\mathcal{N}}_{TK})_{G,G'} = \frac{1}{|\mathbb{W}|^K} \sum_{\vec{W}\in\mathbb{W}^K} \langle\langle \Pi_{\vec{0}}\overline{G}|\mathcal{R} \left(\prod_{k=1}^{K} \overline{W}_k\right) \left(\prod_{k=K}^{1} \mathcal{N}\overline{W}_k\right) |\Pi_{\vec{0}}\overline{G}'\rangle\rangle$$

In this form, our arguments about interaction of twirling with the code space in Section 4.3.1 can be applied to the outermost twirling set, obtaining a reduced twirling set $\mathbb{W}_1$. Hence, we have

$$(\overline{\mathcal{N}}_{TK})_{G,G'} = \frac{1}{|\mathbb{W}_1||\mathbb{W}|^{K-1}} \sum_{\vec{W}\in\mathbb{W}_1\times\mathbb{W}^{K-1}} \langle\langle \Pi_{\vec{0}}\overline{G}|\mathcal{R} \left(\prod_{k=1}^{K} \overline{W}_k\right) \left(\prod_{k=K}^{1} \mathcal{N}\overline{W}_k\right) |\Pi_{\vec{0}}\overline{G}'\rangle\rangle$$

Similar arguments to Section 4.3.3 can be made about the symmetries in both noise and code. However, rather than proving the equivalence of using two different Pauli operators in conjugation, we now will prove the equivalence of using two different *sets* of Pauli operators in conjugation: i.e. after find the symmetry U, we can say a Pauli conjugation set $\vec{W}'$ is equivalent to $\vec{W}$ when $U\vec{W}U^\dagger = \vec{W}'$. Here $U\vec{W}U^\dagger$ is defined as:

$$U\vec{W}U^\dagger = (UW_1U^\dagger, UW_2U^\dagger, \cdots, UW_KU^\dagger)$$

Note that this is not a simple tensor product of the single round case. For example, if W_1 equivalent to W_1' due to symmetry U and W_2 equivalent to W_2' due to another symmetry U', this does not means that $\vec{W} = (W_1, W_2)$ is equivalent to $\vec{W}' = (W_1', W_2')$ since the two elements are related by different symmetry: $U\vec{W}U^\dagger \neq \vec{W} \neq U'\vec{W}U'^\dagger$.

C

Appendices for a Silicon Surface Code Architecture Resilient Against Leakage Errors

C.1 Two Ways to Achieve CZ between Data and Ancilla Qubits

C.1.1 Hamiltonian

The two-spin Hamiltonian is:

$$H = \underbrace{\frac{1}{2}\left(E_1 Z_1 + E_2 Z_2\right)}_{H_0\text{: Zeeman splitting}} + \underbrace{\frac{J}{2}\text{SWAP}}_{H_{ex}\text{: exchange interactions}} \tag{C.1}$$

The Zeeman splitting H_0 can be further split into:

$$\underbrace{\frac{1}{2}\left(E_1 Z_1 + E_2 Z_2\right)}_{H_0\text{: Zeeman splitting}} = \underbrace{\frac{E_z}{2}\left(Z_1 + Z_2\right)}_{H_Z\text{: average Zeeman splitting}} + \underbrace{\frac{\Omega}{2}\left(Z_1 - Z_2\right)}_{H_\Delta\text{: Zeeman splitting gradient}}$$

where $E_z = \frac{E_1+E_2}{2}$, $\Omega = \frac{E_1-E_2}{2}$.

C.1.2 $\Omega \ll J$: simple exchange interaction

Since $\Omega \ll J$, and $[H_{ex}, H_Z] = [\text{SWAP}, Z_1 + Z_2] = 0$,

$$H_{ex,I} = e^{iH_0 t} H_{ex} e^{-iH_0 t} = H_{ex}$$

i.e. to perform the exchange interaction in the rotating frame is just the same as performing the exchange interaction in the lab frame.

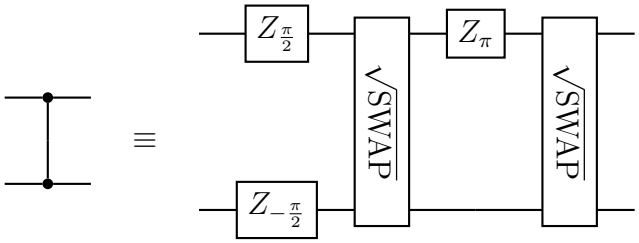
The evolution operator due to H_{ex} is given by:

$$U_{ex}(t) = e^{-iH_{ex}t} = e^{-i\text{SWAP}\frac{Jt}{2}}$$

A SWAP gate corresponds to $\frac{Jt}{2} = \frac{\pi}{2}$, and a $\sqrt{\text{SWAP}}$ gate corresponds to $\frac{Jt}{2} = \frac{\pi}{4}$.

The error in applying the exchange interaction arising from imprecise pulse timing or charge fluctuations is analysed in Appendix C.2.

A CZ can be implemented using $\sqrt{\text{SWAP}}$ in the following way:



C.1.3 $\Omega \gg J$: dipole-dipole interaction

Following arguments from [114, 115], without exchange interaction we have

$$H_0 = \frac{1}{2}\begin{pmatrix} E_z & 0 & 0 & 0 \\ 0 & \Omega & 0 & 0 \\ 0 & 0 & -\Omega & 0 \\ 0 & 0 & 0 & -E_z \end{pmatrix}$$

We can see that E_z determine the eigenenergies in the parallel spin subspace, while Ω determine the eigenenergies in the anti-parallel spin subspace.

If we add in the exchange Hamiltonian

$$H_{ex} = \frac{J}{2}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{C.2}$$

in the parallel spin subspace, the energy of both states will be shifted up by $\frac{J}{2}$. In the anti-parallel spin subspace, if $\Omega \gg J$, then H_{ex} can be treated as perturbation. Using first order perturbation theory, the shift in eigenenergies for the anti-parallel spin states is 0.

Hence, to first order approximation, in which the eigenstate do not change and only eigenenergies change, the exchange Hamiltonian (which is to first order the shift in eigenenergies) becomes

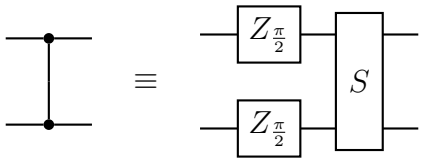
$$H_{ex} = \frac{J}{2}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{C.3}$$

This is just a *dipole-dipole* interaction, which, because it commutes with H_0, has a rotating frame form identical to its lab form.

Allowing this Hamiltonian to evolve for a time period $\frac{\pi}{J}$, produces the following gate:

$$S \propto \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -i & 0 & 0 \\ 0 & 0 & -i & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

A CZ gate can be built from S using:



C.1.4 Virtual Z gate and symmetric operations on ancilla

Whether using S or $\sqrt{\text{SWAP}}$ to construct a CZ, the only type of single-qubit gate needed is the Z rotation, which can be implemented in a virtual way by shifting the rotating reference frame by a given phase [134]. Such Z rotations are essentially error-free and require zero time. This corresponds to adding a phase offset to all subsequent X, Y gate pulses, and switching all subsequent two-qubit gates into the new rotating frame after the virtual Z rotation. Two-qubit gates whose Pauli components consist of only tensor products of I and Z are invariant under changing

rotating reference frame, hence we do not need to modify these two-qubit gates after the virtual Z rotation. The other two-qubit gates usually have different forms in the shifted rotating frame and might not be achievable through our Hamiltonian.

Following such arguments, we find that for the CZ gate constructed using the exchange-interaction, the Z_π bracketed by the two $\sqrt{\text{SWAP}}$s cannot be applied in a virtual way, while the two Z rotations outside the $\sqrt{\text{SWAP}}$s can. For the dipole-dipole CZ gate, all the Z rotations can be applied in a virtual way.

However, there is another caveat. For the virtual Z rotation to work, we need to do the measurements in Z basis at the end, so that all the remnant Z rotation for compensating for the virtual Z gates will have no effect on the measurements (though we can use the shifted one qubit gate to change the measurement basis). Our ancilla measurement does not use a standard basis: our measurement only tells us whether the ancilla is in the singlet or triplet state, where the singlet state and the triplet states does not corresponds to a qubit representation. Thus, we cannot use virtual Z gates here for our ancilla qubits, but can instead permute all the Z rotations (besides the one bracketed by $\sqrt{\text{SWAP}}$) to the position right after the initialisation of the singlet state. We then use the fact that the initial singlet state is invariant under symmetric gates operating on both ancilla dots, to see that there is no need to apply the Z rotations at the ancilla (besides the one bracketed by $\sqrt{\text{SWAP}}$).

Hence, under either approach to implement a CZ gate, the only single-qubit gate that we need to implement is the Z_π bracketed by $\sqrt{\text{SWAP}}$s. All the other Z rotations can be either implemented in a virtual way or can be omitted due to the property of our ancilla qubits.

C.1.5 Comparison of the two implementations of CZ

Operation time

We denote the characteristic time scale of exchange interaction as $T_J = \frac{\pi}{J}$, and that of Z gate as T_Z. The time we needed to achieve a CZ using dipole-dipole like interaction is just T_J, no single-qubit gates needed. On the other hand, the time

we need to achieve a CZ using exchange interaction is $T_J + T_Z$. The extra term here is due to the Z_π gate that we need to explicitly implement.

Errors

Errors due to fluctuation of Jt:

The ideal exchange phase for $\sqrt{\text{SWAP}}$ is $\theta_{sw} = Jt_{sw} = \frac{\pi}{2}$. We will denote the variance in θ_{sw} due to fluctuations in exchange strength J or operation time t as ϵ_{sw}^2.

The ideal exchange phase for S is $\theta_s = Jt_s = \pi$. If we divide the accumulation of phase θ_s into two independent stages, with each stage accumulating phase $\frac{\pi}{2} = \theta_{sw}$, then we have $\theta_s = \theta_{sw,1} + \theta_{sw,2}$. Hence, the variance of θ_s is just $\epsilon_s^2 = 2\epsilon_{sw}^2$.

As shown in Appendix C.2, such fluctuations will lead to:

- $\sqrt{\text{SWAP}}$: $p_{sw} = \epsilon_{sw}^2$ probability of having a swap error.
- S: $p_s = \epsilon_s^2 = 2p_{sw}$ probability of having a Z_1Z_2 error.

Errors due to approximations made:

The main approximation made in deriving the exchange interaction is ignoring the higher order exchange terms which will not change the form of interaction (shift of energy in the singlet subspace w.r.t. the triplet subspace), but only shift the strength of exchange interaction. This is possible to overcome via careful calibrations. Of course there are also perturbations to the eigenstates that we have not considered, which might lead to leakage errors as shown in Appendix C.6.

Since both $\sqrt{\text{SWAP}}$ and S make use of exchange interactions, they are equally affected by the approximations made in the treatment of the exchange interaction. In addition, there are higher order corrections to the S gate due to the assumption $J \ll \Omega$ of magnitude $\frac{J}{\Omega}$. Similarly, there are higher order corrections to the $\sqrt{\text{SWAP}}$ gate due to the assumption $\Omega \ll J$ of magnitude $\frac{\Omega}{J}$.

C.2 Errors due to Fluctuation in Interaction Strength and Time

C.2.1 General theory

Suppose the Pauli basis of Hamiltonian H is the set G_H:

$$H = \sum_{g_i \in G_H} \beta_i g_i$$

note that β_i are real since H is Hermitian.

Then we can define the magnitude of H to be E, and the normalised version of H to be h where:

$$E = \sqrt{\sum_i \beta_i^2} \tag{C.4}$$

$$h = \frac{H}{E} = \sum_{g_i \in G_H} \frac{\beta_i}{E} g_i = \sum_{g_i \in G_H} \alpha_i g_i \tag{C.5}$$

for $\alpha_i = \frac{\beta_i}{E}$ and we have $\sum_i \alpha_i^2 = 1$.

Now the evolution operator is just:

$$U(t) = e^{-iHt} = e^{-ihEt}$$

$$U(\theta) = e^{-i\theta h}$$

with $\theta = Et$.

However, over- and under-rotations of θ occur in the experiment due to imprecise pulse timing t or fluctuation of interaction strength E. If there is a 50% percent chance of over and under rotation by $\epsilon \ll 1$, we have:

$$\begin{aligned} U(\theta \pm \epsilon) &= e^{-i(\theta \pm \epsilon)h} \\ &\approx e^{-i\theta h}\left(I \mp i\epsilon h - \frac{\epsilon^2}{2}h^2\right) \\ &= U(\theta)\left(I - \frac{\epsilon^2}{2}h^2 \mp i\epsilon h\right) \end{aligned}$$

Then the effective operation is just

$$\begin{aligned} \mathcal{U}_{\theta,\epsilon}(\rho) &= \frac{1}{2}U(\theta+\epsilon)\rho U^\dagger(\theta+\epsilon) + \frac{1}{2}U(\theta-\epsilon)\rho U^\dagger(\theta-\epsilon) \\ &= \left(I - \frac{\epsilon^2}{2}h^2\right)U(\theta)\rho U^\dagger(\theta)\left(I - \frac{\epsilon^2}{2}h^2\right) + \epsilon^2 h U(\theta)\rho U^\dagger(\theta)h \end{aligned} \tag{C.6}$$

Similar channels are obtained for other symmetric over/under-rotation distributions that are centred on the correct rotation angles.

h is unitary

If h is unitary (and remember it is also Hermitian since it is the normalised Hamiltonian), e.g. h is SWAP or Pauli, then Eq. (C.6) turns into

$$\mathcal{U}_{\theta,\epsilon}(\rho) = \left(1-\epsilon^2\right) U(\theta)\rho U^\dagger(\theta) + \epsilon^2 h U(\theta)\rho U^\dagger(\theta) h \tag{C.7}$$

i.e. we have either perfect $U(\theta)$ or ϵ^2 probability of having a h error on top of $U_{ex}(\theta)$.

Twirling

Twirling is a technique use for transforming the given error channel into a Pauli channel to obtain a simpler description of the error channel.

The Pauli decomposition of $I - \frac{\epsilon^2}{2}h^2$ is

$$\begin{aligned} I - \frac{\epsilon^2}{2}h^2 &= I - \frac{\epsilon^2}{2}\left[\sum_{i,j}\alpha_i\alpha_j g_i g_j\right] \\ &= (1-\frac{\epsilon^2}{2})I - \frac{\epsilon^2}{2}\left[\sum_{i\neq j}\alpha_i\alpha_j g_i g_j\right] \end{aligned}$$

After twirling, the noise due to non-identity Pauli components scales as $O(\epsilon^4)$ in the Pauli channel, and hence is negligible.

The Pauli decomposition of h is just Eq. (C.5). Hence, after twirling, the effective error channel we have is just:

$$\begin{aligned} \mathcal{U}_{\theta,\epsilon}(\rho) &= (1-\frac{\epsilon^2}{2})U(\theta)\rho U(\theta)^\dagger(1-\frac{\epsilon^2}{2}) + \epsilon^2\left[\sum_{g_i\in G_H}\alpha_i^2 g_i U(\theta)\rho U(\theta)^\dagger g_i\right] \\ &= (1-\epsilon^2)U(\theta)\rho U(\theta)^\dagger + \epsilon^2\left[\sum_{g_i\in G_H}\alpha_i^2 g_i U(\theta)\rho U(\theta)^\dagger g_i\right] \end{aligned} \tag{C.8}$$

i.e. it is an error channel with $\epsilon^2\alpha_i^2$ probability of the Pauli error g_i happening on top of the perfect operation $U(\theta)$.

C.2.2 Applications

Exchange Interaction

For an exchange interaction, we have:

$$H = \frac{J}{2}\text{SWAP}$$

We have fluctuation $\epsilon_{sw} \ll 1$ in $\theta = \frac{Jt}{2}$ and $h = \text{SWAP}$ is unitary. Hence, using Eq. (C.7), we have:

$$\mathcal{U}_{ex,\theta,\epsilon_{sw}}(\rho) = \left(1-\epsilon_{sw}^2\right) U_{ex}(\theta)\rho U_{ex}^\dagger(\theta) + \epsilon_{sw}^2 \text{SWAP}\ U_{ex}(\theta)\rho U_{ex}^\dagger(\theta)\ \text{SWAP}$$

i.e. we have either perfect $U_{ex}(\theta)$ or ϵ_{sw}^2 probability of having a SWAP error on top of $U_{ex}(\theta)$.

Dipole-dipole Interaction

For a dipole-dipole interaction, we have:

$$H = \frac{J}{2}\left(Z_1 Z_2\right)$$

We have fluctuation $\epsilon_s \ll 1$ in $\theta = \frac{Jt}{2}$ and $h = Z_1Z_2$ is unitary. Hence, using Eq. (C.7), we have:

$$\mathcal{U}_{dd,\theta,\epsilon_s}(\rho) = \left(1-\epsilon_s^2\right) U_{dd}(\theta)\rho U_{dd}^\dagger(\theta) + \epsilon_s^2 Z_1Z_2\ U_{dd}(\theta)\rho U_{dd}^\dagger(\theta)\ Z_1Z_2$$

i.e. we have ϵ_s^2 probability of having a Z_1Z_2 error.

C.3 Background Exchange Interaction

In our system, t_{ab} and Δ_M are generally fixed in a given device, however, their values can be engineered in the device design. The mediated exchange coupling (and hence the CZ gate) can be turned on and off by shifting the detuning of the mediator dot with respect to the side dots to switch $\Delta_{L/R}$ between Δ_{on} and Δ_{off}. Since Δ_{off} is finite, there is a residual exchange interaction even in the off

stage. Using Eq. (5.1), we obtain the strength of such residual exchange interaction compared to our intended exchange interaction:

$$\frac{J_{\text{off}}}{J_{\text{on}}} = \left(\frac{\Delta_{\text{on}}}{\Delta_{\text{off}}}\right)^2$$

If we look at the direct exchange interaction instead, we have $J \propto \frac{|t|^2}{\Delta}$ and hence $\frac{J_{\text{off}}}{J_{\text{on}}} = \frac{\Delta_{\text{on}}}{\Delta_{\text{off}}}$. Hence, we see that the residual exchange interaction of mediated exchange is more suppressed than direct exchange when only tuning the on-site energy of quantum dots.

An imperfect 'off' state also leads to next-nearest-neighbour interactions. For direct exchange interaction, the next nearest neighbour interaction is approximated as $\left(\frac{t}{\Delta_{\text{off}}}\right)^2$ of the nearest neighbour interaction. In the mediated exchange interaction however, the next nearest neighbour interaction is approximately $\left(\frac{t}{\Delta_{\text{off}}}\right)^4$ of the nearest neighbour interaction, which is again much more heavily suppressed than the direct exchange case.

Hence, by using mediated exchange interactions we can more confidently ignore the effect of residual exchange interactions and next nearest neighbour interactions in our analysis.

C.4 Comparison of Leakage Resilience to Architectures without Mediators

As mentioned before there are two general schemes to deal with leakage errors in qubits: using leakage reduction protocols or detecting leaked qubits and replacing them.

Using leakage reduction units [129, 131] requires a large number of additional ancilla qubits, which can be hard to integrate due to space constraints in addition to the qubit overhead they bring. We can reuse some of the ancilla qubits to alleviate such challenges, but this in turn significantly increases the surface code runtime and circuit depth. Another way to achieve leakage reduction is by swapping the data and ancilla qubits at the end of every full stabiliser cycle [130], which does not require any additional ancilla qubits. However, such a scheme is not compatible

with architectures that have single-dot data qubit and double-dot ancilla. Moreover, it assumes that the initialisation process of ancilla dots will fix the leakages. This will only be true if we use charge reservoirs for the initialisation of ancilla during the error correction cycle. To prevent the initialisation process of ancilla qubits from affecting other qubits, the charge reservoirs would need to be integrated into the structure and attached to every dot instead of placed at the boundary and relying on shuttling, which is challenging to achieve in a dense quantum dot array without mediators due to space constraints.

As with leakage reduction circuits, leakage *detection* circuits [127, 128] also require a signifincant increase in ancilla number or bring a significant cost in surface code runtime and circuit depth. A more practical approach would instead be to use physical charge detectors for leakage detection. In architectures without mediators, the leading leakage errors are one missing or one extra charge in the quantum dot. Charge detectors would therefore need to be interpersed within a densely packed quantum dot array and capable of accurately distinguish between the three different charge states. Furthermore, after the detection of a charge leakage, we cannot correct them by simply shuttling the leaked charge back because charge leakage can propagate across the array of quantum dots. Overall, this leads to significant practical challenges and spatial constraints. Furthermore, the general leakage reduction/detection circuits described above assume the two-qubit gates in the leakage reduction/detection stage do not induce further leakage or transfer leakage and this is not the case for general two-qubit gates implemented in coupled quantum dot spins.

The practical challenges associated with integrating additional components or ancillae for leakage correction may be solved by using a modular structure [76]. However, such a scheme creates a new source of leakage errors since it involves shuttling electrons across dozens of quantum dots. To keep the leakage error rate of *across-array* shuttling low, we need to have an extremely low rate of *between-dot* shuttling leakage, which means that we need to tune the gate voltages very slowly to maintain excellent adiabaticity. This leads to a trade-off between leakage

suppression and the processing speed of the architecture. In addition, additional schemes to cope of leakage errors from the shuttling itself would be needed.

In architectures without mediators, if the parameters of direct exchange are chosen such that they have similar speed as mediated exchange, then the probability of leakage under direct exchange will be smaller than that using mediated exchange due to the higher energy of the excited charge state. However, if we did not take any active measures against the leakage errors, regardless of how small the leakage error probability is (as long as it is non-negligible), the leakages will keep accumulating until they break our code. As seen from above, active leakage correction schemes lead to a large runtime/qubit overhead. For a dense array of quantum dots, the reservoirs needed for leakage reset or the charge detectors needed for leakage detection are challenging to integrated due to space constraints, while in the modular scheme, the required electron shuttling creates a new source of leakage. In contrast, to handle leakage in our architecture, there are no additional components nor complex schemes required. We merely reset the mediators when they are idle, making our architecture more robust against charge leakage errors compared to the other quantum dot architectures.

C.5 Resultant Computational Error from Leakage and Restoration

For our system, there is no reason to assume that either the leakage event or the restoring charge relaxation are spin-conserving. Hence when a spin in a qubit dot is leaked and restored, we can assume that all the spin information is lost, which is equivalent to a depolarising error. When we look at the exchange interaction between qubit A and B via a mediator. If qubit A has leaked and been restored, it will be depolarised. Before the leakage, qubit B interacts with the original qubit A, and after the leakage qubit B interacts with the depolarised qubit A (via a mediator that might be faulty). The leakage and restoration can happen at any point during the exchange interaction, such uncertainty leads to a random depolarising error on the qubit B as well.

Besides the depolarisation of the data qubit and the ancilla qubit involved in the exchange interaction, a leakage error may also lead to faulty mediator dots and hence affect the subsequent gates. Each stabiliser check cycle can be divided into two halves (interacting only inasmuch as they each include one dot of an ancilla double-dot pair): a five-dot system with one ancilla dot (A) connecting to two data dots (D1 and D2) via two mediators (M1 and M2). A interacts with D1 first via M1 in stage 1, then with D2 via M2 in stage 2. An error in stage 1 only affects stage 2 if A has leaked and been restored using an electron from M2. In such a case, the left-over electron in M2 will be in a random state, thus when the electrons in A and D2 interact with the left over electrons in M2 in stage 2, they will also be depolarised regardless of whether further leakages and restorations happens in stage 2 or not.

Hence, we have the following leakage error table for the five-dot system with a exchange gate leakage probability p:

Stage 1	Stage 2	Errors
$1-p$: No leakages	$1-p$: No leakages	Perfect
	p : Any leakages	A and D2 depolarised
$\frac{p}{4}$: Leaked A and restored from M2	Any	All depolarised
$\frac{3p}{4}$: Other leakages	$1-p$: No leakages	A and D1 depolarised
	p : Any leakages	All depolarised

Hence, we can see here we have $(1-p)^2 = 1-2p+p^2$ probability of having no leakage, and otherwise we will have partial or full depolarisation errors to the three qubits. In the calculations described in the main text, we have assumed an error model where we have $1-2p$ probability of having no leakage and otherwise have full depolarisation errors on all three qubits, which is a more severe error model than the more detailed one we describe here.

C.6 One Possible Leakage Mechanism

C.6.1 Three-dot system

With reference to Fig. 5.2 in the main text, the charge configuration of the ground state is $(1, 2, 1)$. In the exchange Hamiltonian, the charge ground state is connected to the excited state charge states $(0, 3, 1)$ and $(1, 3, 0)$ via the tunnelling energies t_{L2} and t_{R2}. Hence, the eigenstates of the exchange Hamiltonian are a superposition of the ground and excited state charge configurations. As noise causes such a superposition to decohere, there is a possibility that the exchange eigenstates will collapse into the excited state charge configuration, bringing the three dot setup out of the computational subspace, and leading to leakage errors.

The probability of such a leakage error is related to the amplitude of the excited state in the coupled system eigenstate. Using perturbation theory, such an amplitude has a magnitude of $\frac{t}{\Delta}$, where t is the tunnelling energy between the high energy state and the ground state while Δ is the energy difference between them. Hence, the possibility of our ground charge configuration $(1, 2, 1)$ escaping into the high energy charge configuration $(0, 3, 1)$, $(1, 3, 0)$ will be on the order of $\left(\frac{t_{L2}}{\Delta_L}\right)^2$ and $\left(\frac{t_{R2}}{\Delta_R}\right)^2$, which means that such leakage process will only be significant during the exchange interaction (when $|\Delta_{L,R}|$ is small). Below we present a detailed analysis for the two-dot case, which can be easily generalised to our case.

C.6.2 Two-dot system

Hamiltonian

For our two-dot system, we denote $|T\rangle$ as the triplet state with zero z-component, $|S\rangle$ as the singlet state, $|\text{ion}_+\rangle$ as the state that has two electrons in one dot that can be reached by $|S\rangle$ via hopping, and $|\text{ion}_-\rangle$ as the other state with two electrons in one dot but is orthogonal to $|\text{ion}_+\rangle$.

We divide the Hamiltonian H into two parts, a dominating diagonal part $H^{(0)}$:

$$H^{(0)} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & U & 0 \\ 0 & 0 & 0 & U \end{pmatrix} \begin{matrix} |T\rangle = \left|0^{(0)}\right\rangle \\ |S\rangle = \left|1^{(0)}\right\rangle \\ |\mathrm{ion}_+\rangle = \left|2^{(0)}\right\rangle \\ |\mathrm{ion}_-\rangle = \left|3^{(0)}\right\rangle \end{matrix}$$

and a small off diagonal (tunnelling) part $rH^{(1)}$.

$$rH^{(1)} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & t+t^* & 0 \\ 0 & t+t^* & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} |T\rangle = \left|0^{(0)}\right\rangle \\ |S\rangle = \left|1^{(0)}\right\rangle \\ |\mathrm{ion}_+\rangle = \left|2^{(0)}\right\rangle \\ |\mathrm{ion}_-\rangle = \left|3^{(0)}\right\rangle \end{matrix}$$

r here is the ratio between the off-diagonal tunnelling energy t and the diagonal detuning energy Δ:

$$r = \frac{t}{\Delta} \ll 1.$$

Here we see that $rH^{(1)}$ only mixes $|S\rangle$ and $|\mathrm{ion}_+\rangle$ and leaves $|T\rangle$ and $|\mathrm{ion}_-\rangle$ unchanged.

Starting from the eigenstates and the eigenenergies of $H^{(0)}$, we can obtain the eigenstates and the eigenenergies of H using perturbation theory:

$$\begin{aligned} H &= H^{(0)} + rH^{(1)} \\ |n\rangle &= \left|n^{(0)}\right\rangle + r\left|n^{(1)}\right\rangle + r^2\left|n^{(2)}\right\rangle + \cdots \\ E_n &= E_n^{(0)} + rE_n^{(1)} + r^2E_n^{(2)} + \cdots \end{aligned}$$

the superscript (m) denotes the m^{th}-order correction.

Perturbation theory

- Change in states ⇒ **leakage error**:

$$\begin{aligned} r\left|1^{(1)}\right\rangle &= \sum_{E_n^{(0)}\neq E_1^{(0)}} \left|n^{(0)}\right\rangle \frac{\left\langle n^{(0)}\right| rH^{(1)} \left|1^{(0)}\right\rangle}{E_1^{(0)} - E_n^{(0)}} \\ &= \left|2^{(0)}\right\rangle \frac{\left\langle 2^{(0)}\right| rH^{(1)} \left|1^{(0)}\right\rangle}{E_1^{(0)} - E_2^{(0)}} \\ &= -\frac{t+t^*}{U}\left|2^{(0)}\right\rangle \end{aligned} \tag{C.9}$$

$$\begin{aligned} r\left|2^{(1)}\right\rangle &= \sum_{E_n^{(0)}\neq E_2^{(0)}} \left|n^{(0)}\right\rangle \frac{\left\langle n^{(0)}\right| rH^{(1)} \left|2^{(0)}\right\rangle}{E_2^{(0)} - E_n^{(0)}} \\ &= \left|1^{(0)}\right\rangle \frac{\left\langle 1^{(0)}\right| rH^{(1)} \left|2^{(0)}\right\rangle}{E_2^{(0)} - E_1^{(0)}} \\ &= \frac{t+t^*}{U}\left|1^{(0)}\right\rangle \end{aligned} \tag{C.10}$$

Hence

$$\begin{aligned} |1\rangle &= \left|1^{(0)}\right\rangle - \frac{t+t^*}{U}\left|2^{(0)}\right\rangle \\ |2\rangle &= \left|2^{(0)}\right\rangle + \frac{t+t^*}{U}\left|1^{(0)}\right\rangle \end{aligned}$$

- Change in the ground state energy ⇒ **exchange interaction**:

The leading non-vanishing order of energy shift is

$$\begin{aligned} r^2 E_1^{(2)} &= -2\frac{(t+t^*)^2}{U} \\ r^2 E_2^{(2)} &= 2\frac{(t+t^*)^2}{U} \end{aligned}$$

Leakage oscillation

Now if we start in the state of $|S\rangle = \left|1^{(0)}\right\rangle$ the probability of leaking into $|\text{ion}_+\rangle = \left|2^{(0)}\right\rangle$ is:

$$\begin{aligned}
\left\langle 2^{(0)}\right| e^{-i\hat{H}t} \left|1^{(0)}\right\rangle &= \sum_n e^{-iE_n t} \left\langle 2^{(0)}\middle|n\right\rangle \left\langle n\middle|1^{(0)}\right\rangle \\
&= e^{-iE_1 t} \underbrace{\left\langle 2^{(0)}\middle|1\right\rangle}_{-\frac{t+t^*}{U}} \underbrace{\left\langle 1\middle|1^{(0)}\right\rangle}_{1} + e^{-iE_2 t} \underbrace{\left\langle 2^{(0)}\middle|2\right\rangle}_{1} \underbrace{\left\langle 2\middle|1^{(0)}\right\rangle}_{\frac{t+t^*}{U}} \\
&= \frac{t+t^*}{U}\left(e^{-iE_2 t} - e^{-iE_1 t}\right) \\
&= \frac{t+t^*}{U} e^{-i\frac{E_2+E_1}{2}t}\left(e^{-i\frac{E_2-E_1}{2}t} - e^{i\frac{E_2-E_1}{2}t}\right) \\
&= \frac{t+t^*}{U} e^{-i\frac{E_2+E_1}{2}t}(-2i)\sin\left(\frac{E_2-E_1}{2}t\right)
\end{aligned}$$

Hence,

$$\left|\left\langle 2^{(0)}\right| e^{-i\hat{H}t} \left|1^{(0)}\right\rangle\right|^2 = 4\left(\frac{t+t^*}{U}\right)^2 \sin^2\left(\frac{E_2-E_1}{2}t\right)$$

To the leading order $E_2 - E_1 = U$. Hence, the probability of leaking has the magnitude of r^2 and oscillates with the frequency $\frac{U}{2}$

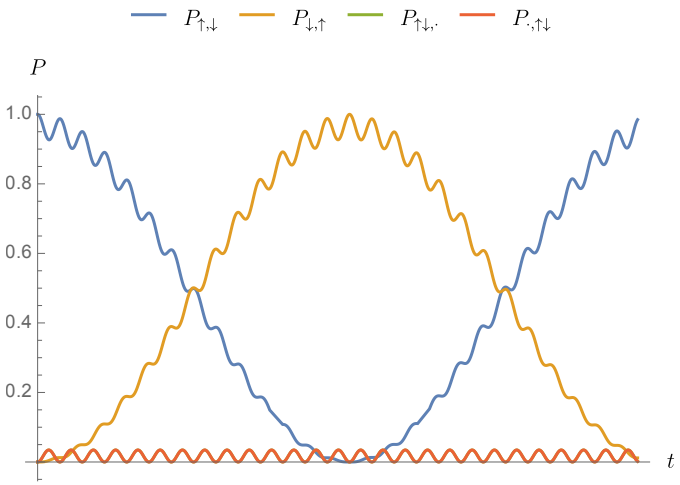


Figure C.1: The probability of being in a different spin/charge states during one period of exchange interaction, following an initial $|\uparrow,\downarrow\rangle$ state. Note that the green and red lines completely overlap, and both represent a leakage probability. Here we have used $r = \frac{t}{\Delta} = 0.1$.

C.7 Threshold Simulation Details

Based on the circuit and the error model outlined in Section 5.4, we can obtain two error tables that outlines the probabilities of all possible error patterns (including both the errors on data qubits and the parity errors of the measurement results) when performing the X and Z stabiliser checks respectively. This will enable us to perform a Monte Carlo simulation of the stabiliser check process with errors arising according to the probability obtained from the error tables. Each round of stabiliser checks will give rise to a 2D grid of parity check results. For a distance-d surface code, we will repeat our stabiliser measurement for d times to fight with measurement errors, which can be viewed as stacking up d layers of 2D parity result grid, giving rise to a 3D grid with one of the dimension being time [37]. We can then try to match the failed parity checks to the boundary or to any change in the parity results in the time direction using minimum-weight perfect matching(MWPM), which is carried out using the Blossom V package [31]. The surface code threshold simulation module we used is published on Github [205]. For simplicity, in our simulation we have the same weight for the edges in the spatial direction and the edges in the time direction. However, threshold improvements can be gained by optimising the weight ratios between them due to the different probabilities of failure. Further improvements of the threshold can be achieved by using more advance decoders, e.g. the maximum likelihood decoder [206].

D

Appendices for Resource Estimation for Quantum Variational Simulations of the Fermi-Hubbard Model

D.1 Hubbard Model Hamiltonian Ansatz

D.1.1 Simulation scheme

Here we recap the scheme to implement the 2D open-boundary Hubbard model Hamiltonian ansatz as described in [149]. We will be considering a 2D Hubbard models of V sites, with the starting canonical ordering of the orbitals as shown in Fig. D.1.

Gates will be local if they are applying to the orbitals adjacent to each other in the canonical ordering. In the canonical ordering shown in Fig. D.1, the orbitals of the same site and different spins are adjacent to each other, enabling the application of local parametrised on-site repulsion gates. To apply local hopping gates, we need to apply fermionic swaps to the orbitals to move them around in the canonical ordering. There are two types of fermionic swaps that we can apply: swaps between or within spins, which corresponding to swaps between or within rows for the orbital layout in Fig. D.1. The swap scheme in [149] involving alternating swaps within and between spins. Now suppose we focus only on the spin-down orbitals. The spin-down

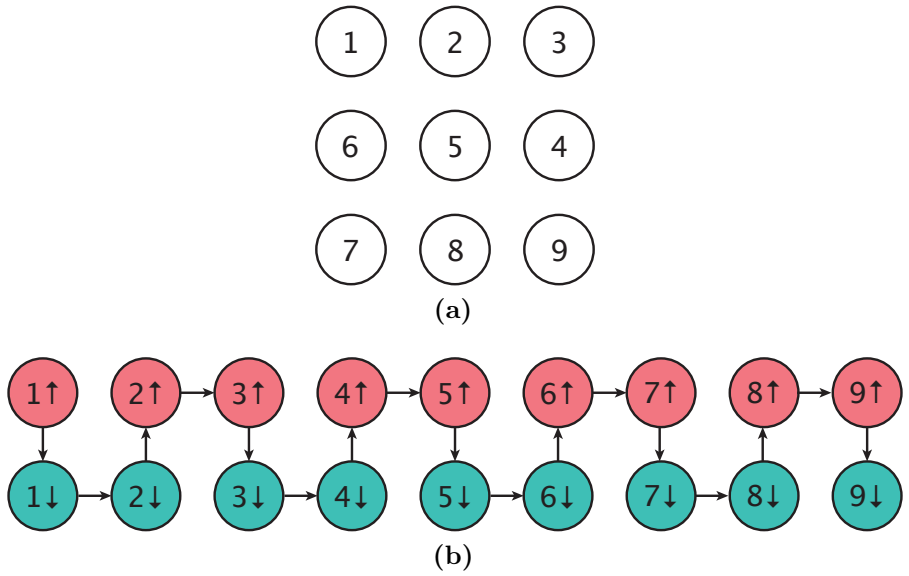


Figure D.1: (a) The layout and labelling of the sites in a 3-by-3 Hubbard model. (b) The corresponding canonical ordering of the orbitals at the beginning of the circuit. The two rows of different colours denote different spins.

orbitals start in row 2 in which we can only perform local swaps and local hopping interactions between the even pairs of orbitals. Then we swap between spins, moving spin down to row 1, and now we can do local swaps and local hopping interactions between the odd pairs of orbitals. Repeating these steps will enable us to alternate between odd- and even-pair swaps and hopping interactions within the spin-down orbitals, which are interleaved with swaps between the two spins. Such a scheme can apply all relevant hopping interactions in a local manner as shown in Fig. D.2.

We will define an *edge pair* as a pair of sites that are adjacent to each other in the canonical ordering and are vertical neighbours in the site layout. For example, in Fig. D.1, the edge pairs are $(3, 4)$ and $(6, 7)$. The way we perform one block of Hamiltonian ansatz is by repeating $2 * N_{col}$ rounds of the following two steps:

1. Swap between spins: swapping orbitals of the same sites but different spins.

2. Swap within the same spins, except for edge pairs on which we perform hopping interaction instead.

This will return all the orbitals to their original positions. Within the process, we need to

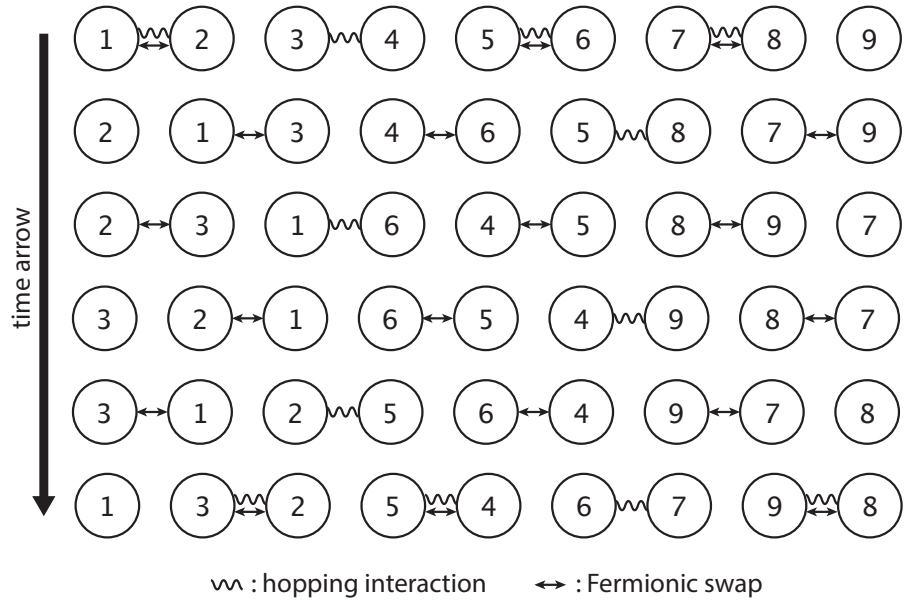


Figure D.2: The scheme for swapping spin-down orbitals and applying the relevant hopping interactions for a 3×3 Hubbard model. In between steps, there are also swaps between the two spins that we did not show. In the end, the spin orbitals return to the original ordering, enabling the application of the next block of the Hamiltonian ansatz. A similar scheme is applied to the spin-up orbitals except that it starts with odd-pair swaps first.

- Substitute step 1 in the first round with repulsion interactions and swaps between spins.

- Substitute step 2 in the first and last round with hopping interactions and swaps interaction within same spins, except for edge pairs on which we only perform hopping interactions, no swaps.

We can also implement the periodic boundary condition in the horizontal direction by adding a pair of hopping interaction into certain rounds of step 2.

D.1.2 Gate count analysis for ansatz

The gates in the Hubbard model simulation scheme are parametrised gates for on-site repulsion, adjacent-site hopping, fermionic swaps and combinations of them. Here we will decompose them into single-qubit rotations and partial swaps, which form one of the basic gate sets in silicon qubits.

We will find the hopping gates, the fermionic swaps, and the fSWAP+hopping all starts with $Z_{\frac{\pi}{2}}$ and end with $Z_{-\frac{\pi}{2}}$ on one of the qubits. For on-site repulsion and fSWAP+repulsion, we have $Z_{-\frac{\pi}{2}}$ at the end and we can easily add an $Z_{\frac{\pi}{2}}$ in front by adding a Z rotation pair $Z_{\frac{\pi}{2}}Z_{-\frac{\pi}{2}}$. All of these gates are symmetric under the exchange of qubits, thus we can choose which qubit to place the Z rotations.

We will choose to place these Z rotation on the odd qubits in the spin-up space and on the even qubits in the spin-down space. In this way, these Z rotation will cancel [1]. For the hopping term, the fermionic swap and the fSWAP+hopping, this means removing two single-qubit rotations. For on-site repulsion and fSWAP+repulsion in which we have added a Z rotation pair $Z_{\frac{\pi}{2}}Z_{-\frac{\pi}{2}}$ in front, the gate counts do not change.

In the following section, we will use the below notations for the time units of the gates:

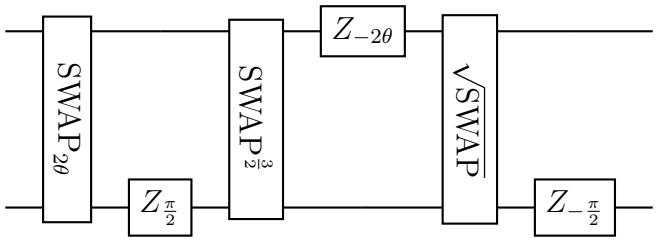
- τ_{1q}: the time unit for a single-qubit gate, which is the time needed to perform a $\frac{\pi}{2}$ rotation. We will assume the time needed to carry out a single-qubit gate with a variable parameter, i.e. gates like Z_θ, is on average τ_{1q}.

- τ_{2q}: the time unit for a two-qubit gate, which is the time needed to perform a $\sqrt{\text{SWAP}}$. We will assume the time needed to carry out a partial swap with a variable parameter, i.e. gates like SWAP_θ, is on average τ_{2q}.

The decompositions of the gates into partial swaps and single-qubit rotations and their resource estimates are shown below:

- On-site repulsion: $U_U(\theta) = e^{-i\frac{\theta}{2}(I-Z_1)(I-Z_2)}$

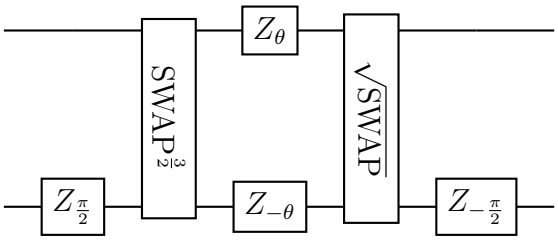
[1] Note that in each iteration, we will have $\sqrt{V}-1$ interactions within the same spin, while we have $\sqrt{V}$ interaction in between different spins, hence, the Z gate on one of the dot will not be cancelled, this will be the last dot in either the spin up space or the spin-down space. Also, the Z rotations at the beginning and the end of the circuit are not cancelled. However, when estimating the number of gates needed, for a large number of sites (hence large number of rounds of iterations), we will assume such boundary effects are negligible.



Gate counts: $G_{1q,U} = 3$, $G_{2q,U} = 3$.

Time needed: $\tau_U = 4\tau_{1q} + 6\tau_{2q}$

- Hopping interaction: $U_t(\theta) = e^{-i\frac{\theta}{2}(XX+YY)}$

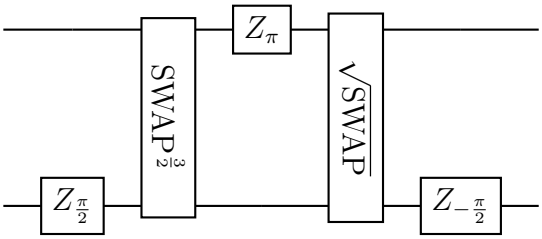


Gate counts: $G_{1q,t} = 2$, $G_{2q,t} = 2$

Time needed: $\tau_t = \tau_{1q} + 4\tau_{2q}$

- Fermionic swap:

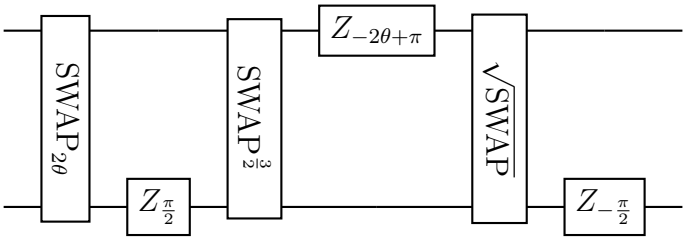
  $F_{sw} = \frac{1}{2}(XX + YY + ZI + IZ) = \text{SWAP} \cdot \text{CZ}$



Gate counts: $G_{1q,F} = 1$, $G_{2q,F} = 2$

Time needed: $\tau_f = 2\tau_{1q} + 4\tau_{2q}$

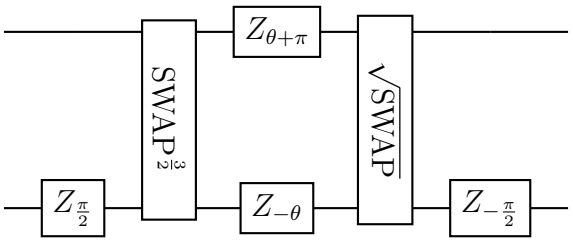
- Fermionic swap + on-site repulsion: $F_{sw}U_U$



Gate counts: $G_{1q,FU} = 3$, $G_{2q,FU} = 3$.

Time needed: $\tau_{FU} = 6\tau_{1q} + 6\tau_{2q}$

- Fermionic swap + hopping interaction: $F_{sw}U_t$



Gate counts: $G_{1q,Ft} = 2$, $G_{2q,Ft} = 2$

Time needed: $\tau_{Ft} = 3\tau_{1q} + 4\tau_{2q}$

Following the above gate decomposition and the Hamiltonian ansatz circuit outlined in Appendix D.1, we can obtain the following estimates for the total number of one-qubit gates needed N_{1q} (all are Z rotations), total number of two-qubit gates needed N_{2q} (all are partial swaps) and the total length of time needed T to perform one block of Hamiltonian ansatz for 2D Hubbard model of V sites using basic single-qubit rotation and partial swaps:

$$
\begin{aligned}
N_{1q} &= 4V^{\frac{3}{2}} + 7V - 4\sqrt{V} \\
N_{2q} &= 8V^{\frac{3}{2}} + V - 4\sqrt{V} \\
T &= \left(8\sqrt{V} + 5\right)\tau_{1q} + \left(16\sqrt{V} + 2\right)\tau_{2q}
\end{aligned}
$$

For $V = 25$, we have:

$$N_{1q} \approx 650$$
$$N_{2q} \approx 1000$$
$$T \approx 45\tau_{1q} + 80\tau_{2q}$$

D.2 Slater Determinant Preparation

Here we recap the Slater determinant preparation scheme outlined in [149, 159]. Note that the input Slater determinant we choose to prepare in this Article will follow the same spin and site-layout symmetry as the output ground state since the ansatz we choose preserves these two symmetries.

D.2.1 Background

We will use N_{orb} to denote the total number of orbitals that we are considering while N_e will be the number of electrons (i.e. the number of occupied orbitals).

We start with the qubits representing the eigenorbitals of the initial Hamiltonian (e.g. non-interacting Hubbard Hamiltonian). The initial state is the ground state of the initial Hamiltonian with the first N_e orbitals being occupied (i.e. the first N_e qubits are initialised to 1 while the rest are initialised to 0). Now the role of the state preparation circuit is to transform our qubits from representing the eigenstates of the initial Hamiltonian (with orbital creation operators $\{a_j^\dagger\}$) to the orbitals that can have a compact description of our target Hamiltonian, i.e. to the site orbital basis (with orbital creation operators $\{b_j^\dagger\}$). This basis transformation can be described by the transformation matrix $Q^\dagger$ (also called the Slater determinant) of the shape $N_{orb} \times N_e$:

$$\vec{b}^\dagger = Q^\dagger \vec{a}^\dagger = U^\dagger \Lambda W \vec{a}^\dagger = U^\dagger \Lambda \vec{a}'^\dagger$$

Here we have carried out singular value decomposition of $Q^\dagger$. W is a rotation within the filled-orbital subspace to find a new set of basis $\{a_i'^\dagger\}$ other than the eigenstate of the initial Hamiltonian $\{a_i^\dagger\}$. The qubit ground state in basis $\{a_i'^\dagger\}$ is

the same as basis $\{a_i^\dagger\}$ with all the orbitals filled. Hence, we do not need to carry out the transformation W explicitly, we only need to keep in mind that now we are working in this new basis for the input state instead of the initial basis [159]. Λ is a rectangular matrix of the $N_{orb} \times N_e$ with ones at the diagonal and zeros elsewhere. This is just an isometry to expand our space from the filled orbital subspace to the full orbital space by attaching $N_{orb} - N_e$ empty orbitals. Then the transformation $U^\dagger$ on the full orbital space will complete our transformation of basis. The transformation $U^\dagger$ will be implemented as compositions of Givens rotations in the quantum circuit.

D.2.2 Givens rotation

A Givens rotation is just a general rotation operation within a 2D complex subspace. There are two parts to a Givens rotation, one is the rotation to change the amplitude, and the other is phase operator to change the relative phase between the two basis in the subspace [159]:

$$G(\theta, \phi) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix}$$

We will be mostly dealing with real Slater determinant, so let us ignore the phase part here. The rotation part can be carried out using the operator:

$$R_{ij}(\theta) = e^{\frac{\theta}{4}(a_i^\dagger a_j - a_j^\dagger a_i)}$$

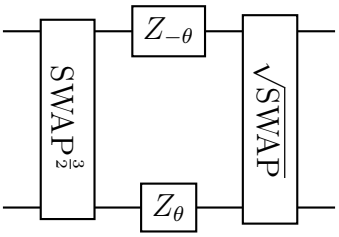
JW transform the neighbouring orbital case, we have

$$\begin{aligned} \left(a_i^\dagger a_{i+1} - \left(a_i^\dagger a_{i+1}\right)^\dagger\right) &= 2i \operatorname{Im}\left\{a_i^\dagger a_{i+1}\right\} \\ &\Rightarrow 2i \operatorname{Im}\{(X - iY)(X + iY)\} \\ &= 2i(XY - YX) \end{aligned}$$

Hence, the given rotation for adjacent orbitals are:

$$R(\theta) = e^{i\frac{\theta}{2}(XY-YX)} = e^{-i\frac{\theta}{2}(YX-XY)}$$

which translate into the following circuit using partial swap and Z rotations.



Using the notation in Appendix D.1.2, we have:

Gate counts: $G_{1q,G} = 2$, $G_{2q,G} = 2$

Time needed: $\tau_G = \tau_{1q} + 4\tau_{2q}$

D.2.3 Gate count for Givens rotation

Simple scheme

The transformation from the initial eigenbasis $\{a_i^\dagger\}$ to the target eigenbasis $\{b_i^\dagger\}$ can be viewed as a process of trying to diagonalise the transformation matrix (Slater determinant) Q. The transformation W can zero out a triangle of entries along the N_e dimension. We need one Givens rotation to zero out each remaining non-zero off-diagonal elements. The number of Givens rotation needed is [159]:

$$N_e \left(N_{orb} - N_e\right)$$

For the half-filling Hubbard model with V sites, we have $N_{orb} = 2V$ and $N_e = V$. Hence, we have V^2 off-diagonal elements to be zeroed out. Each Givens rotation is decomposed into two partial-swaps and two Z rotations, thus the number of gates we needed are (following the notation in Appendix D.1.2)

$$N_{1q} = 2V^2$$

$$N_{2q} = 2V^2$$

Using the parallel scheme suggested in [149], we have a circuit of depth

$$(N_{orb} - 1 - (N_e - 1)) + (N_e - 1) = N_{orb} - 1 = 2V - 1$$

which translate into the circuit runtime of

$$T = (2V - 1)\left(\tau_{1q} + 4\tau_{2q}\right)$$

For $V = 25$, we have:

$$N_{1q} = 1250$$
$$N_{2q} = 1250$$
$$T = 49\tau_{1q} + 196\tau_{2q}$$

Using spin conservation

For the Hubbard Hamiltonian that we are considering, the two spin spaces are decoupled. Hence, we can consider the Slater determinants of the two spin subspace separately, each of the shape $\frac{N_e}{2} \times \frac{N_{orb}}{2}$. Hence, the number of Givens rotation needed in each spin subspace is

$$\frac{N_e}{2}\left(\frac{N_{orb}}{2} - \frac{N_e}{2}\right) = \frac{V^2}{4}.$$

To keep the applications of the Givens rotations within the spin subspace on only adjacent orbitals, we need to start in an orbital ordering with the first $N_{orb}/2$ being the spin-up orbitals and the next $N_{orb}/2$ being the spin-down orbitals. Hence, we need to carry out orbital rearrangement after the Givens rotations to restore the spin-orbital ordering for the Hamiltonian ansatz (Appendix D.1.1).

We can arrange the Givens rotations such that all the Givens rotation on the last spin-up orbital (the $\frac{N_{orb}}{2}$th orbital) and the first spin-down orbital (the $\frac{N_{orb}}{2} + 1$th orbital) finished first. In the next time step, while we are carrying out other Givens rotations, we can start performing fSWAP on the two orbitals that have finished Givens rotation. In the next step, two more orbitals will finish their Givens rotation (orbitals $\frac{N_{orb}}{2} - 1$ and $\frac{N_{orb}}{2} + 2$), thus now we can perform fSWAP on orbitals $(\frac{N_{orb}}{2} - 1, \frac{N_{orb}}{2})$ and $(\frac{N_{orb}}{2} + 1, \frac{N_{orb}}{2} + 2)$. Carry on we will have more and more orbitals finishing their Givens rotations, and in each time step we will perform fSWAP on all orbitals that finished Givens rotations, alternating between the odd pair of orbitals and even pair of orbitals. After $\frac{N_{orb}}{2} - 2$ layer

of swap, we will have alternating up and down orbitals. One more layer of swaps between all orbitals $4n-1$ and $4n$, will give us the $\uparrow\downarrow\downarrow\uparrow\uparrow\cdots$ order we used in the Hamiltonian ansatz (Appendix D.1.1). Note here we only talk about how to arrange the orbitals to have the right spin ordering, for the ordering of the orbitals within the same spin, it is determined by order of the rows and columns of the Slater determinants that we wrote down. In total, we need

$$\frac{(\frac{N_{orb}}{2}-2+1)\left(\frac{N_{orb}}{2}-2\right)}{2}+\lfloor\frac{N_{orb}}{4}\rfloor\approx\frac{N_{orb}^2}{8}-\frac{N_{orb}}{2}=\frac{V^2}{2}-V$$

fSWAPs to achieve the desired orbital order.

For the fSWAP gates, similar to Appendix D.1.2, we can arrange the gates such that the $\sqrt{Z}$ only acts on the odd orbitals, which will enable the cancellation of all the $\sqrt{Z}$ other than those at the boundary. Hence, we can ignore the $\sqrt{Z}$ gates required by the fSWAPs. Our scheme need $\frac{V^2}{4}\times 2$ Givens rotations and $\frac{V^2}{2}-V$ fSWAPs. When decomposed into partial swap and Z rotation, the number of one-qubit gates and two-qubit gates needed are

$$N_{1q}=2\left(\frac{V^2}{2}\right)+\frac{V^2}{2}-V=\frac{3}{2}V^2-V$$
$$N_{2q}=2\left(\frac{V^2}{2}\right)+2\left(\frac{V^2}{2}-V\right)=2V^2-2V$$

The depth of the circuit before finishing the first Givens rotation is $N_{orb}-N_e=V$. This section of the circuit consist of only Givens rotation, hence require runtime:

$$T_a=V\left(\tau_{1q}+4\tau_{2q}\right)$$

After this, we have the fSWAP network with Givens rotation happening concurrently, the depth of the circuit here is $V-1$, the runtime is limited by the fSWAP instead of Givens rotation since fSWAP contains π-rotations of Z. Hence, the runtime needed for the fSWAP networks is:

$$T_b=(V-1)\left(2\tau_{1q}+4\tau_{2q}\right)$$

Hence, the total runtime needed for the state preparation circuit is:

$$T = T_a + T_b = (3V - 2)\,\tau_{1q} + (8V - 4)\,\tau_{2q}$$

For $V = 25$, we have:

$$N_{1q} = 910$$
$$N_{2q} = 1250$$
$$T = 73\tau_{1q} + 196\tau_{2q}$$

Hence, the spin subspace scheme leads to some reduction in the number of one-qubit gate. However, it also leads to longer circuit runtime.

Comparison to Ansatz circuit

When compared to Appendix D.1.2, the number of one-qubit gates of the Slater determinant preparation circuit is twice of that of one layer of the Hamiltonian ansatz, the number of two-qubit gates and the runtime are comparable.

Do note that the number of gates of the Slater determinant preparation circuit scale as $\mathcal{O}(V^2)$, which is worse than the $V^{3/2}$ ansatz scaling. The depth of the Slater determinant preparation circuit scale as $\mathcal{O}(V)$, which is also worse than the ansatz scaling $\mathcal{O}(\sqrt{V})$ as well. However, we need to note that we did not take into account of the number of blocks of Hamiltonian ansatz might be needed in the ansatz scaling, and the number of blocks needed is very likely to scale worse than $\sqrt{V}$, which means that the state preparation should have a better scaling than the ansatz if we take that into account.

D.3 Gate Count Analysis for Superconducting Qubits

Now we will follow the same analysis in Appendix D.1 and Appendix D.2, but switch to superconducting qubits. In superconducting qubits, we will use partial iSWAP [188] instead of partial SWAP as our elementary two-qubit gate. The differences between the two are discussed in Ref. [189]

Note that partial iSWAP is *not* $e^{-i\frac{\theta}{2}\text{i SWAP}}$, instead it is just a gate based on XY interaction:

$$\text{iSWAP}_\theta := e^{-i\frac{\theta}{2}(XX+YY)}$$

$$\text{iSWAP} := \text{iSWAP}_{-\pi/2}.$$

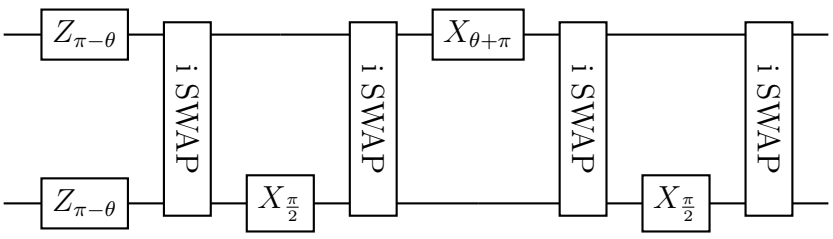
In our circuit, partial iSWAP will be the only type of two-qubit gate we use. This would enable us to implement *all* Z rotations in a virtual way since the iSWAP can be easily transformed to implement virtual Z gate [134]. Thus in the following gate count section we will omit all Z rotations in our gate counts and time counts.

In the following section, we will use the below notations for the time units of the gates:

- τ_{1q}: the time unit for a single-qubit gate, which is the time needed to perform a $\frac{\pi}{2}$ rotation. We will assume the time needed to carry out a single-qubit gate with a variable parameter, i.e. gates like X_θ, is on average τ_{1q}.
- τ_{2q}: the time unit for a two-qubit gate, which is the time needed to perform a iSWAP. We will assume the time needed to carry out a partial swap with a variable parameter, i.e. gates like iSWAP_θ, is on average τ_{2q}.

The decompositions of the gates in the ansatz into partial iSWAPs and single-qubit rotations and their resource estimates are shown below:

- On-site repulsion: $U_U(\theta) = e^{-i\frac{\theta}{2}(I-Z_1)(I-Z_2)}$



Gate counts: $G_{1q,U} = 3$, $G_{2q,U} = 4$.

Time needed: $\tau_U = 5\tau_{1q} + 4\tau_{2q}$

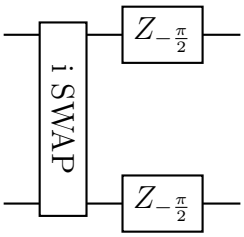
- Hopping interaction: $U_t(\theta) = e^{-i\frac{\theta}{2}(XX+YY)} = \text{iSWAP}_\theta$

  Gate counts: $G_{1q,t} = 0$, $G_{2q,t} = 1$

  Time needed: $\tau_t = \tau_{2q}$

- Fermionic swap:

  $F_{sw} = \frac{1}{2}(XX + YY + ZI + IZ)$

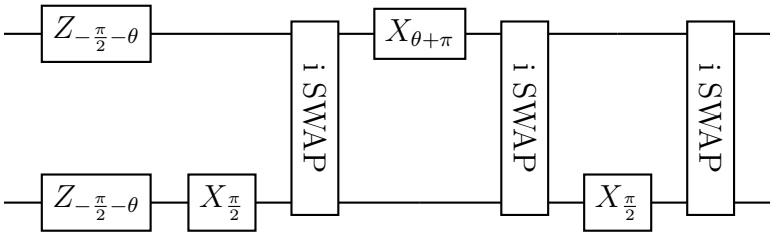


  Gate counts: $G_{1q,F} = 0$, $G_{2q,F} = 1$

  Time needed: $\tau_f = \tau_{2q}$

- Fermionic swap + on-site repulsion: $F_{sw}U_U$

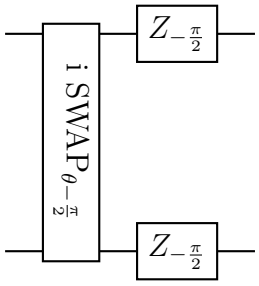
  Using the fact that iSWAP commute with $Z_\theta \otimes Z_\theta$ and iSWAP·iSWAP $= Z \otimes Z$, we have:



  Gate counts: $G_{1q,FU} = 3$, $G_{2q,FU} = 3$.

  Time needed: $\tau_{FU} = 5\tau_{1q} + 3\tau_{2q}$

- Fermionic swap + hopping interaction: $F_{sw}U_t$



Gate counts: $G_{1q,Ft} = 0$, $G_{2q,Ft} = 1$

Time needed: $\tau_{Ft} = 2\tau_{2q}$

Following the above gate decomposition and the Hamiltonian ansatz circuit outlined in Appendix D.1, we can obtain the following estimates for the total number of one-qubit gates needed $N_{1q,antz}$ (all are X rotations), total number of two-qubit gates needed $N_{2q,antz}$ (all are partial swaps) and the total length of time needed T_{antz} to perform one block of Hamiltonian ansatz for 2D Hubbard model of V sites using basic single-qubit rotation and partial iSWAPs:

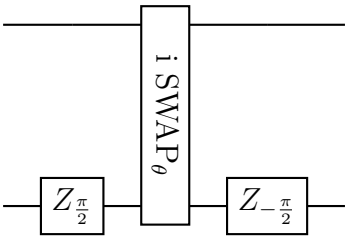
$$\begin{aligned} N_{1q,antz} &= 3V \\ N_{2q,antz} &= 4V^{\frac{3}{2}} + 2V - 2\sqrt{V} \\ T_{antz} &= 5\tau_{1q} + 4\left(\sqrt{V} + 1\right)\tau_{2q} \end{aligned}$$

For $V = 25$, we have:

$$\begin{aligned} N_{1q,antz} &\approx 75 \\ N_{2q,antz} &\approx 540 \\ T_{antz} &\approx 5\tau^{(1)} + 24\tau^{(2)} \end{aligned}$$

Now we will turn to initial Slater determinant preparation outlined in Appendix D.2. The gate needed is Given rotation, which can be decomposed as

- Givens rotation: $R(\theta) = e^{-i\frac{\theta}{2}(YX-XY)}$



Gate counts: $G_{1q,G} = 0$, $G_{2q,G} = 1$

Time needed: $\tau_G = \tau_{2q}$.

Following the same arguments in Appendix D.2.3, the number of two-qubit gates and one-qubit gates needed for Slater determinant preparation is:

$$N_{1q,prep} = V^2$$
$$N_{2q,prep} = V^2.$$

The depth of the Slater determinant preparation circuit is

$$D = 2V - 1$$

which translate into the circuit runtime of

$$T_{prep} = (2V - 1)\,\tau_{2q}.$$

Together we can obtain the resource estimate for the full ansatz circuit with N_{blk} ansatz blocks:

$$\begin{aligned} N_{1q} &= V^2 + 3VN_{blk} \\ N_{2q} &= V^2 + \left(4V^{\frac{3}{2}} + 2V - 2\sqrt{V}\right) N_{blk} \\ T &= (2V - 1)\,\tau_{2q} + \left(5\tau_{1q} + 4\left(\sqrt{V} + 1\right)\tau_{2q}\right) N_{blk} \end{aligned}$$

For $V = N_{blk} = 25$, we have:

$$N_{1q} \approx 2500$$
$$N_{2q} \approx 14000$$
$$T \approx 125\tau_{1q} + 650\tau_{2q}.$$

D.4 Obtaining Energy Gradient in Quantum Computers

D.4.1 Background

As mentioned in Section 7.2. For a given Hamiltonian H, we want to find the set of optimal parameters $\vec{\theta}$ for an ansatz circuit such that the state $\left|\psi(\vec{\theta})\right\rangle$ it produce is as close to the ground state of the Hamiltonian as possible, i.e. we want to find $\vec{\theta}$ such that $E_{tot}(\vec{\theta}) = \left\langle\psi(\vec{\theta})\right| H \left|\psi(\vec{\theta})\right\rangle$ is minimised.

However, since the energy $E_{tot}(\vec{\theta})$ cannot be directly measured, we need to rewrite our Hamiltonian in terms of its Pauli components:

$$H = \sum_j \lambda_j G_j$$
$$\left\langle\psi(\vec{\theta})\right| H \left|\psi(\vec{\theta})\right\rangle = \sum_j \lambda_j \left\langle\psi(\vec{\theta})\right| G_j \left|\psi(\vec{\theta})\right\rangle.$$

Hence,

$$E_{tot}(\vec{\theta}) = \sum_j \lambda_j E_i(\vec{\theta})$$

where $E_j(\vec{\theta}) = \left\langle\psi(\vec{\theta})\right| G_j \left|\psi(\vec{\theta})\right\rangle$, which is just the expectation value of a Pauli observable, hence can be directly obtained from the circuit.

To find the ground state, we can use gradient-based optimisation methods like gradient decent and L-BFGS-B, etc. This require us to obtain $\frac{\partial E_{tot}(\vec{\theta})}{\partial \theta_m}$, which in turns means that we need to measure $\frac{\partial E_j(\vec{\theta})}{\partial \theta_m}$ for every Hamiltonian component j and every parameter m.

We can obtain $\frac{\partial E_j(\vec{\theta})}{\partial \theta_m}$ simply using finite difference, in which we will measure $E_j(\theta_1, \theta_2, \cdots \theta_m, \cdots)$ and $E_j(\theta_1, \theta_2, \cdots \theta_m + \delta\theta_m, \cdots)$, and take their difference divided by $\delta\theta_m$ to approximate the gradient.

D.4.2 The circuit approach to obtain gradients

The exact gradient $\frac{\partial E_j(\vec{\theta})}{\partial \theta_m}$ can be obtained using circuit measurement instead of using finite difference approximation. Suppose our ansatz circuit consist of parametrised

rotation gates R_n of the form

$$R_n(\theta_n) = e^{-i\theta_n F_n}$$

where F_n are both Hermitian and unitary, e.g. Pauli or swaps.

Then our parametrised circuit can be written as:

$$\left|\psi(\vec{\theta})\right\rangle = \prod_{n=N}^{1} R_n(\theta_n) \left|0\right\rangle = R(\vec{\theta}) \left|0\right\rangle$$

where we have denote the whole circuit using $R(\vec{\theta})$.

A string of parametrised rotation can be denoted as:

$$\prod_{n=a}^{b} R_n(\theta_n) = R_{a:b}$$

which means that $R(\vec{\theta}) = R_{N:1}$

Hence,

$$\begin{aligned}\frac{\partial R(\vec{\theta})}{\partial \theta_m} &= R_{N:m+1} \frac{\partial R_m(\theta_m)}{\partial \theta_m} R_{m-1:1} \\ &= -iR_{N:m+1} F_m R_{m:1}\end{aligned}$$

Hence we have

$$\begin{aligned}\frac{\partial E_j(\vec{\theta})}{\partial \theta_m} &= 2\,\mathrm{Re}\Big\{\Big\langle\psi(\vec{\theta})\Big| G_j \left(\partial_m \Big|\psi(\vec{\theta})\Big\rangle\right)\Big\} \\ &= 2\,\mathrm{Re}\Big\{\langle\overline{0}|\, R^\dagger(\vec{\theta}) G_j \left(\partial_m R(\vec{\theta})\right) |\overline{0}\rangle\Big\} \\ &= 2\,\mathrm{Im}\Big\{\langle\overline{0}|\, (R_{N:1})^\dagger\, G_j R_{N:m+1} F_m R_{m:1} \,|\overline{0}\rangle\Big\}\end{aligned}$$

This can be measured via two kind of circuits:

- Indirect measurement [48]:

  The circuit is shown in Fig. D.3 (a) will measure $\frac{1}{2}\frac{\partial E_j(\vec{\theta})}{\partial \theta_m}$. Here we use an ancilla which probes the main ansatz with a control unitary. The advantage of such a scheme is that we can obtain the exact gradient of the circuit via measurement of only one qubit. However, it requires an extra ancilla qubit. The need for performing the control unitary between the ancilla and any other qubits also lead to connectivity challenges. Control unitaries like control swap are also not straightforward to implement in lots of architectures.

- Direct measurement [207]:

  The circuit is shown in Fig. D.3 (b). Denoting $|\phi\rangle = R_{m:1}|\overline{0}\rangle$, $W = R_{N:m+1}$, the circuit will measure:

$$\begin{aligned} A_{jm,\pm} &= \langle\phi| e^{\pm i\frac{\pi}{4}F_m} W^\dagger G_j W e^{\mp i\frac{\pi}{4}F_m} |\phi\rangle \\ &= \frac{1}{2}\langle\phi| (1 \pm iF_m) W^\dagger G_j W (1 \mp iF_m) |\phi\rangle \\ &= \frac{1}{2}\langle\phi| W^\dagger G_j W |\phi\rangle + \frac{1}{2}\langle\phi| F_m W^\dagger G_j W F_m |\phi\rangle \\ &\pm i\frac{1}{2}\Big(\langle\phi| F_m W^\dagger G_j W |\phi\rangle - \langle\phi| W^\dagger G_j W F_m |\phi\rangle\Big) \end{aligned}$$

  Hence, the gradient $\frac{\partial E_j(\vec{\theta})}{\partial\theta_m}$ can be obtained via

$$\begin{aligned} A_{jm,+} - A_{jm,-} &= i\Big(\langle\phi| F_m W^\dagger G_j W |\phi\rangle - \langle\phi| W^\dagger G_j W F_m |\phi\rangle\Big) \\ &= 2\,\mathrm{Im}\Big\{\langle\phi| W^\dagger G_j W F_m |\phi\rangle\Big\} \\ &= \frac{\partial E_j(\vec{\theta})}{\partial\theta_m} \end{aligned}$$

  For this method, we do not require any extra ancilla or control unitaries. However, we need to measure two expectation values $A_{jm,+}$ and $A_{jm,-}$ for the estimation of the gradient $\frac{\partial E_j(\vec{\theta})}{\partial\theta_m}$ instead of one in the case of indirect measurement.

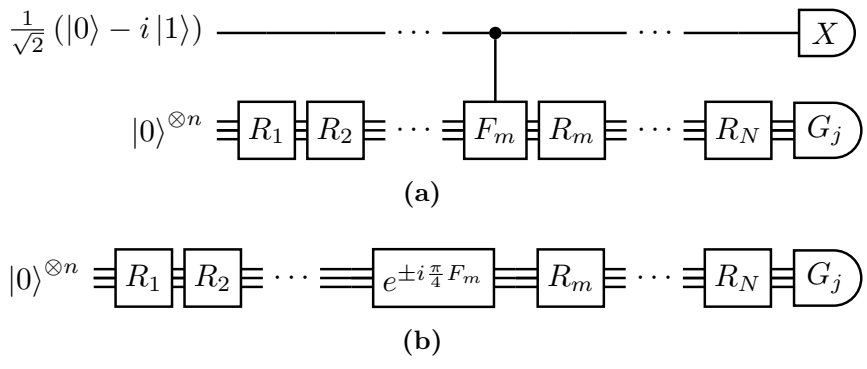


Figure D.3: These are the circuits used to measure the gradient $\frac{\partial E_j(\vec{\theta})}{\partial\theta_m}$ using (a) indirect measurements, (b) direct measurements. Here $E_j(\vec{\theta}) = \left\langle\psi(\vec{\theta})\right| G_j \left|\psi(\vec{\theta})\right\rangle$ is the expectation value of the jth component of the Hamiltonian. θ_m is the parameter of the mth parametrised gate $R_m(\theta_m) = e^{-i\theta_m F_m}$ of the ansatz circuit, where F_m is both Hermitian and unitary. Note that here we have assumed that all parametrised gates have independent parameters.

D.4.3 Shared parameters

For the case where there are parametrised gates with shared parameters, we can still obtain the gradients using finite difference in a similar way.

When using gradient circuits, the story is more complicated. Firstly, we add a scaling factor β_m to each parameters: $\theta_m \to \beta_m \theta_m$:

$$\begin{aligned} \frac{\partial R(\{\theta_n\})}{\partial \theta_m} &= -iR_{N:m+1} F_m R_{m:1} \\ \Rightarrow \frac{\partial R(\{\beta_n \theta_n\})}{\partial \beta_m \theta_m} &= -iR_{N:m+1} F_m R_{m:1} \\ \frac{\partial R(\{\beta_n \theta_n\})}{\partial \theta_m} &= -i\beta_m R_{N:m+1} F_m R_{m:1} \end{aligned}$$

Now we change the labelling of the parametrised gates: $m \to m, v$, for which gates with the same m will share the same parameters, and v labels the different parametrised gates that share the same parameter:

$$\theta_{m,v} = \theta_m \quad \forall v$$

In such case, we have

$$\frac{\partial R(\{\beta_{n,w}\theta_{n,w}\})}{\partial \theta_m} = \sum_v \left[\frac{\partial R(\{\beta_{n,w}\theta_{n,w}\})}{\partial \theta_{m,v}} \right]_{\theta_{m,v}=\theta_m \forall v}$$

i.e. the gradient w.r.t. to a given parameter θ_m is the sum of all the gradient w.r.t. the parameter of each parametrised gate that shared the parameter value. Hence, when trying to obtain the gradient using circuit measurements, we still need to treat the parameters in each gate as independent, and then sum those gradient up based on which gates have shared parameters.

D.5 Number of Samples Needed for Energy and Energy Gradient

D.5.1 Number of samples in finite difference

Gradient precision

The equation for the estimation of the gradient of the j^{th} Pauli term in the Hamiltonian using finite difference is:

$$\frac{\partial E_j(\vec{\theta})}{\partial \theta_m} = \frac{\overline{E}_j(\vec{\theta} + \frac{\vec{\delta}_m}{2}) - \overline{E}_j(\vec{\theta} - \frac{\vec{\delta}_m}{2})}{\delta} \tag{D.1}$$

where $\vec{\delta}_m$ is a vector with the m^{th} parameter set to δ and all other parameters set to 0. $\overline{E}_j$ denote the sampling average of E_j.

Hence, we have

$$\mathrm{Var}\left[\partial_m E_j\right] = \frac{2}{\delta^2}\mathrm{Var}\left[\overline{E}_j\right] \tag{D.2}$$

i.e. for larger δ, we can achieve smaller variance in gradient for a fixed variance in energy. However, we cannot increase δ indefinitely because there is an error associated with the finite step size when using finite difference, which has the magnitude of:

$$\frac{\delta^2}{24}\sum_{u,v,w=1}^{N_{sh}} \frac{\partial^3 E_j(\vec{\theta})}{\partial\theta_{m,v}\partial\theta_{m,u}\partial\theta_{m,w}} \approx N_{sh}^3 \frac{\delta^2}{24}\partial_m^3 E_j$$

where number of parametrised gates share parameter θ_m is N_{sh}. The sum over v, u, w are the sum over all the parametrised gates that share the same parameter θ_m as discussed in Appendix D.4.3. We have made the assumption that all third-order derivatives in the sum have similar magnitudes and the same sign. Note that the finite step size error increases with δ.

A balance between the two errors can be achieved when we choose the step size δ to satisfy

$$\sqrt{\frac{2}{\delta^2}\mathrm{Var}\left[\overline{E}_j\right]} = N_{sh}^3\frac{\delta^2}{24}\partial_m^3 E_j$$

$$\delta = \left(\frac{24\sqrt{2\mathrm{Var}\left[\overline{E}_j\right]}}{N_{sh}^3\partial_m^3 E_j}\right)^{\frac{1}{3}}$$

Note that

$$\frac{\partial^3 E_j(\vec{\theta})}{\partial \theta_m^3} = 2\,\mathrm{Re}\Big\{\Big\langle\psi(\vec{\theta})\Big|\, G_j\left(\partial_m^3\left|\psi(\vec{\theta})\right\rangle\right)\Big\} + 6\,\mathrm{Re}\Big\{\Big(\partial_m\left\langle\psi(\vec{\theta})\right|\Big)\, G_j\left(\partial_m^2\left|\psi(\vec{\theta})\right\rangle\right)\Big\}$$

where $\mathrm{Re}\Big\{\Big(\partial_m\left\langle\psi(\vec{\theta})\right|\Big)\, G_j\left(\partial_m^2\left|\psi(\vec{\theta})\right\rangle\right)\Big\}$ and $\mathrm{Re}\Big\{\Big\langle\psi(\vec{\theta})\Big|\, G_j\left(\partial_m^3\left|\psi(\vec{\theta})\right\rangle\right)\Big\}$ can be measured using circuit similar to the first order derivative in Appendix D.4.2. Hence, the magnitude of $\frac{\partial^3 E_j(\vec{\theta})}{\partial \theta_m^3}$ is around $\sqrt{2^2+6^2} \approx 6$.

Thus we have:

$$\begin{aligned}\delta &= \left(\frac{24\sqrt{2\mathrm{Var}\left[\overline{E}_j\right]}}{6N_{sh}^3}\right)^{\frac{1}{3}} \\ &= \frac{1.78}{N_{sh}}\mathrm{Var}\left[\overline{E}_j\right]^{\frac{1}{6}}.\end{aligned}$$

Substituting into Eq. (D.2), we have:

$$\begin{aligned}\mathrm{Var}\left[\partial_m E_j\right] &= 0.63 N_{sh}^2 \mathrm{Var}\left[\overline{E}_j\right]^{\frac{2}{3}} \\ \mathrm{Var}\left[\overline{E}_j\right] &= \frac{2}{N_{sh}^3}\mathrm{Var}\left[\partial_m E_j\right]^{\frac{3}{2}}.\end{aligned} \tag{D.3}$$

This is the smallest variance in the gradient that we can achieve for a given variance in the energy estimation by choosing the optimal step size δ.

Number of samples needed

Assuming $\mathrm{Var}\left[E_j\right] \sim \mathcal{O}(1)$, then the number of samples needed to achieve the sample average variance $\mathrm{Var}\left[\overline{E}_j\right]$ is

$$\frac{\mathrm{Var}\left[E_j\right]}{\mathrm{Var}\left[\overline{E}_j\right]} \sim \frac{1}{\mathrm{Var}\left[\overline{E}_j\right]}$$

Since we need to evaluate sample average $\overline{E}_j$ at two points in finite difference, the total number of samples needed is:

$$M_{fd} \sim \frac{2}{\mathrm{Var}\left[\overline{E}_j\right]} = \frac{N_{sh}^3}{\mathrm{Var}\left[\partial_m E_j\right]^{\frac{3}{2}}} \tag{D.4}$$

D.5.2 Number of samples in direct measurement

Gradient precision

The equation of estimation of the gradient of a the j^{th} Pauli term in the Hamiltonian using direct measurement is:

$$\begin{aligned}\frac{\partial E_j(\vec{\theta})}{\partial \theta_m} &= \sum_{v=1}^{N_{sh}} \left[\frac{\partial E_j(\vec{\theta})}{\partial \theta_{m,v}}\right]_{\theta_{m,v}=\theta_m \forall v} \\ &= \sum_{v=1}^{N_{sh}} \left(\overline{A}_{jmv,+} - \overline{A}_{jmv,-}\right)\end{aligned}$$

Here the sum over v is the sum over all the parametrised gates that share the same parameter θ_m as discussed in Appendix D.4.3.

Hence, we have:

$$\mathrm{Var}\left[\partial_m E_j\right] = 2N_{sh}\mathrm{Var}\left[\overline{A}\right]$$

Number of samples needed

Assuming $\mathrm{Var}\left[A\right] \sim \mathcal{O}(1)$, then the number of samples needed to achieve sample average variance $\mathrm{Var}\left[\overline{A}\right]$ is

$$\frac{\mathrm{Var}\left[A\right]}{\mathrm{Var}\left[\overline{A}\right]} \sim \frac{1}{\mathrm{Var}\left[\overline{A}\right]}$$

Since we need to evaluate sample average $\overline{A}$ at $2N_{sh}$ points in direct measurement, the total number of samples needed is:

$$M_{dm} \sim \frac{2N_{sh}}{\mathrm{Var}\left[\overline{A}\right]} = \frac{(2N_{sh})^2}{\mathrm{Var}\left[\partial_m E_j\right]} \tag{D.5}$$

When compared to Eq. (D.4) of finite difference, we can see that direct measurement has better scaling in terms of both the number of shared of parameters N_{sh} and the target gradient precision $\mathrm{Var}\left[\partial_m E_j\right]$, thus direct measurement is preferred.

D.5.3 Comparison between finite difference and direct measurement

To compare the two methods, we use Eq. (D.4) and Eq. (D.5) to study

$$\frac{M_{dm}}{M_{fd}} = \frac{4\sqrt{\mathrm{Var}\left[\partial_m E_j\right]}}{N_{sh}}$$

If we define the breaking point of the gradient precision as:

$$\epsilon^*_{grad} = \frac{N_{sh}}{4} \tag{D.6}$$

then we have:

- $\sqrt{\text{Var}\left[\partial_m E_j\right]} \geq \epsilon^*_{grad} \Rightarrow \frac{M_{dm}}{M_{fd}} \geq 1$:

  Finite difference need less samples to achieve the given precision.

- $\sqrt{\text{Var}\left[\partial_m E_j\right]} < \epsilon^*_{grad} \Rightarrow \frac{M_{dm}}{M_{fd}} < 1$:

  Direct measurement need less samples to achieve the given precision.

$N_{sh} = 4N_{eq}$ for hopping term (if we assume spin symmetry) and $N_{sh} = 3N_{eq}$ for repulsion term where N_{eq} is the number of equivalent partition in the site layout due to symmetry. Here we will take the approximation that all $N_{sh} = 4N_{eq}$ since there are more hopping term than repulsion terms. Thus we have:

$$\text{Var}\left[\partial_m E_j\right] = 8N_{eq}\text{Var}\left[\overline{A}\right] \tag{D.7}$$

Using the parametrisation discuss in Section 7.5.1, for open boundary Hubbard model the breaking point of the gradient precision ϵ^*_{grad} is:

- Square site layout:

$$N_{eq} = 8 \Rightarrow \epsilon^*_{grad,sq} = 2.5 \times 10^{-4}$$

- Rectangular site layout:

$$N_{eq} = 4 \Rightarrow \epsilon^*_{grad,rt} = 1 \times 10^{-3}$$

D.5.4 Number of samples needed for energy

As discussed in [165], to compete with the best classical algorithm we need to estimate the energy per site E_{site} to $10^{-3}t$ precision.

We can decompose the total energy into its subterms E_j:

$$E_{tot} \approx \sum_{j=1}^{J} h_j E_j$$

here h_j is the coefficient of the Pauli decomposition of the Hamiltonian $H = \sum_{j=1}^{J} h_j G_j$.

The repulsion terms and hopping terms in the Hamiltonian can be decomposed into their Pauli components:

$$E_{rep} = \frac{1}{4} + \sum_{j=1}^{3} \frac{1}{4} E_j$$
$$E_{hop} = \sum_{j=1}^{2} \frac{1}{2} E_j$$

Hence, the variance in their sampling average are:

$$\mathrm{Var}\left[\overline{E}_{rep}\right] = \frac{3}{4^2}\mathrm{Var}\left[\overline{E}_j\right]$$
$$\mathrm{Var}\left[\overline{E}_{hop}\right] = \frac{1}{2}\mathrm{Var}\left[\overline{E}_j\right]$$

There are V repulsion terms and $4V$ hopping terms, hence the total energy is:

$$\sum_{k=1}^{V} E_{rep,k} + \sum_{k=1}^{4V} E_{hop,k} = V E_{site}$$

which translate into the following equation for variance:

$$V\mathrm{Var}\left[\overline{E}_{rep}\right] + 4V\mathrm{Var}\left[\overline{E}_{hop}\right] = V^2\mathrm{Var}\left[\overline{E}_{site}\right]$$
$$\mathrm{Var}\left[\overline{E}_j\right] = \frac{16}{35} V \mathrm{Var}\left[\overline{E}_{site}\right]$$

As mentioned above, we want to achieve $\mathrm{Var}\left[\overline{E}_{site}\right] = (10^{-3}t)^2$, assuming $t \sim \mathcal{O}(1)$, we have:

$$\mathrm{Var}\left[\overline{E}_j\right] = \frac{16}{35} V \times 10^{-6} \tag{D.8}$$

Assuming $\mathrm{Var}\left[E_j\right] \sim \mathcal{O}(1)$, then the number of samples needed to achieve sample average variance $\mathrm{Var}\left[\overline{E}_j\right]$ is

$$M_{Ej} \sim \frac{\mathrm{Var}\left[E_j\right]}{\mathrm{Var}\left[\overline{E}_j\right]} \sim \frac{1}{\mathrm{Var}\left[\overline{E}_j\right]} = \frac{35}{16V} \times 10^6.$$

As mentioned in Section 7.5.2, we need 5 circuit runs to evaluate all energy subterms. Thus, the total number of circuit runs needed to evaluate the energy to the required precision is:

$$M_E = 5M_{Ej} = \frac{1.1 \times 10^7}{V}$$

Thus for $V = 25$, we have:

$$M_E \approx 4 \times 10^5$$

D.5.5 Number of samples needed for energy gradients

First we need to decide what precision of the energy gradient is needed. In Appendix D.5.4, we have obtained the precision of the energy subterms that we want to achieve. If we are using finite difference method, in order to achieve such a precision in the final result energy subterms, we can evaluate the energy points in the gradient estimation to the same precision, and our terminating threshold of change in the estimated energy subterms can be set to the same precision. In such case, the gradient precision that we can achieve can be obtained using Eq. (D.3) and Eq. (D.8):

$$\begin{aligned} \mathrm{Var}\left[\partial_m E_j\right] &= 0.63 N_{sh}^2 \mathrm{Var}\left[\overline{E}_j\right]^{\frac{2}{3}} \\ &= 3.7 N_{sh}^2 V^{\frac{2}{3}} \times 10^{-5} \end{aligned} \tag{D.9}$$

To achieve the same precision using direct measurement, the number of sample needed can be obtained by substituting this into Eq. (D.5):

$$M_{dm} \approx \frac{(2N_{sh})^2}{\mathrm{Var}\left[\partial_m E_j\right]} = 1.1 V^{-\frac{2}{3}} \times 10^5. \tag{D.10}$$

Note that this is just the number of samples needed to obtain the gradient of an energy subterm w.r.t. *one* parameters. To obtain the full gradient vector, we need to iterate over all parameters. Using Eq. (7.12), the number of circuit runs needed to evaluate $\frac{\partial E_j(\vec{\theta})}{\partial \theta_m}$ for all parameters m using direct measurement is

$$\begin{aligned} M_{grad,j} &= M_{dm} N_{para} \\ &\approx \frac{1.1 N_{para}^{site} N_{blk} V^{\frac{1}{3}}}{N_{eq}} \times 10^5 \end{aligned}$$

As mentioned in Section 7.5.2, we need 5 circuit runs to measure all energy subterms. Thus, the total number of circuit runs needed to evaluate the energy gradient vector to the required precision is:

$$M_{grad} = 5 M_{grad,j} = \frac{5.5 N_{para}^{site} N_{blk} V^{\frac{1}{3}}}{N_{eq}} \times 10^5$$

For 5×5 Hubbard model, we have $V = 25$, $N_{eq} = 8$ and $N_{para}^{site} = 5$. Assuming $N_{blk} = V$, the number of circuit runs needed is:

$$M_{grad} \approx 2.5 \times 10^7$$

D.6 Quantum Dot Layout

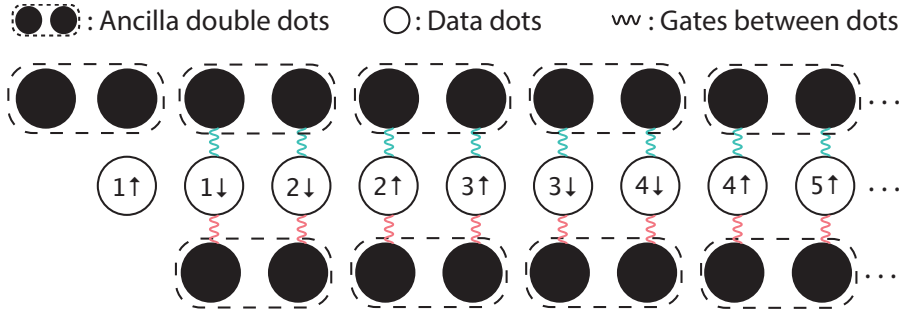


Figure D.4: The measurements of the hopping terms. The blue interaction here are control-X gate from the ancilla to the data for the measurement of XX in the hopping term. The red interaction here are control-Y gate from the ancilla to the data for the measurement of YY in the hopping term.

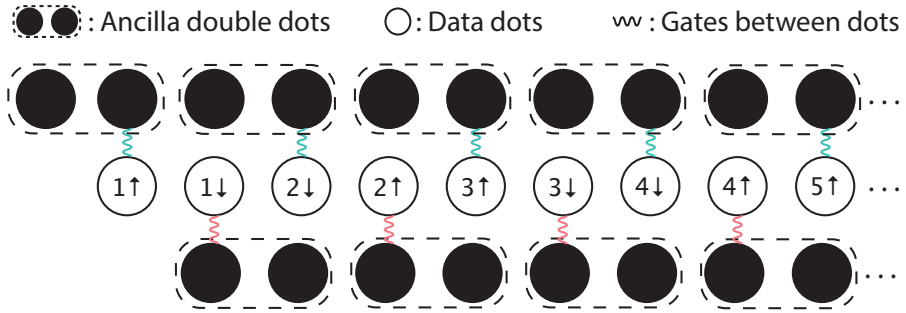


Figure D.5: The measurements of the repulsion terms. All the interaction here are control-Z gates. One row of ancilla is in charge of Z measurements of the odd data dots while the other row of ancilla is in charge of Z measurements of the even data dots.

Fig. D.4, Fig. D.5 shows how we can perform the non-demolishing measurements required by our problem using three lines of quantum dot, one line for data and two lines for ancilla. The ancilla here come in the form of double dot, which will be initialised in singlet and will be readout using Pauli spin blockade which enable us to distinguish singlet and triplet state. Fig. D.4 shows how we can measure XX and YY of the hopping terms. After that, we can compose XX and YY

to obtain ZZ (with an additional $-$ sign) which in terms can give us electron number parity for symmetry verification. Alternatively we can also measure XX and ZZ, and obtain YY via post-processing since control-Z gate can be easier to implement in silicon qubits. Fig. D.4 shows how we perform Z-measurement for every data dots to obtain the repulsion terms. The electron number parity in symmetry verification can again be obtained via composing our measurement results. Our measurement can also be carried out using only one row of ancilla. In such a case, we essentially using one row of ancilla to carry out two rounds of measurements with ancilla reinitialisation in between. Other architectures that incorporating spin-to-charge readout using reservoirs are also possible.

E

Appendices for Multi-exponential Error Extrapolation and Combining Error Mitigation Techniques

E.1 Post-processing Verification

As discussed in Section 6.2.1, when we cannot obtain the observable O and the symmetry S in the same run, we need to instead measure the Pauli components of $O\Pi_s$ [64]:

$$O\Pi_s = \sum_{k=0}^{K-1} \alpha_k G_k = A \sum_{k=0}^{K-1} \mathrm{sgn}(\alpha_k) \frac{|\alpha_k|}{A} G_k$$

in which $A = \sum_{k=0}^{K-1} |\alpha_k|$. Note that α_k is real since Π_s is Hermitian. When we run the circuit, we will perform G_k measurement with $\frac{|\alpha_k|}{A}$ probability and the measurement result will be multiplied with the sign factor $\mathrm{sgn}(\alpha_k)$. The expectation value of such a sampling scheme scaled by A will be $\langle \Pi_s O \rangle$.

In the case that we cannot measure S directly, then we also need to measured Π_s using a similar scheme by breaking Π_s into Pauli basis.

For the case of a Pauli observable O, to perform symmetry verification in this way will require a sampling cost factor of [162]

$$C_{S2} \sim \frac{1}{\mathrm{Tr}(\Pi_s \rho)^2}.$$

which is C_S^2, i.e. we need to square the sampling cost to overcome our limitation of unable to measure our observables and symmetries in the same run.

E.2 Composition of Group Channels

The group error $\mathcal{J}_{p,\mathbb{E}}$ of the group $\mathbb{E}$ with an error probability p is defined in Section 8.2. Using $\widetilde{\mathbb{E}}$ to denote the set of generators of the group $\mathbb{E}$, we can rewrite the pure group error $\mathcal{J}_{1,\mathbb{E}}$ as:

$$\mathcal{J}_{1,\mathbb{E}} = \frac{1}{|\mathbb{E}|}\sum_{E\in\mathbb{E}}\overline{E} = \prod_{\widetilde{E}\in\widetilde{\mathbb{E}}}\frac{\overline{I}+\overline{\widetilde{E}}}{2}$$

Using

$$\left(\frac{\overline{I}+\overline{\widetilde{E}}}{2}\right)\left(\frac{\overline{I}+\overline{\widetilde{E}}}{2}\right) = \frac{\overline{I}+\overline{\widetilde{E}}}{2}$$

we have:

$$\mathcal{J}_{1,\mathbb{E}}\mathcal{J}_{1,\mathbb{B}} = \left(\prod_{\widetilde{E}\in\widetilde{\mathbb{E}}}\frac{\overline{I}+\overline{\widetilde{E}}}{2}\right)\left(\prod_{\widetilde{B}\in\widetilde{\mathbb{B}}}\frac{\overline{I}+\overline{\widetilde{B}}}{2}\right) = \prod_{\widetilde{D}\in\widetilde{\mathbb{E}}\cup\widetilde{\mathbb{B}}}\frac{\overline{I}+\overline{\widetilde{D}}}{2} = \mathcal{J}_{1,\mathbb{D}} \tag{E.1}$$

where $\mathbb{D}$ is the group generated by the union of of the generators of $\mathbb{E}$ and $\mathbb{B}$:

$$\widetilde{\mathbb{D}} = \widetilde{\mathbb{E}}\cup\widetilde{\mathbb{B}}.$$

i.e. $\mathbb{D}$ is the group of elements that one can obtained by composing the elements in $\mathbb{E}$ and $\mathbb{B}$.

In the case of $\widetilde{\mathbb{E}}$ and $\mathbb{B}$ have no overlaps, i.e. $\widetilde{\mathbb{E}}\cap\mathbb{B} = 0$ (note not $\widetilde{\mathbb{B}}$ here since there is a degree of freedom in choosing $\widetilde{\mathbb{B}}$), we have $\widetilde{\mathbb{D}} = \widetilde{\mathbb{E}} + \widetilde{\mathbb{B}}$. We will have the same result if we have $\mathbb{E}\cap\widetilde{\mathbb{B}} = 0$ instead. In such as case, for the subgroup $\mathbb{E}$ of $\mathbb{D}$, $\mathbb{B}$ is the corresponding quotient group, and vice versa.

In the case of $\mathbb{E}$ is a subgroup of $\mathbb{B}$, we have

$$\mathcal{J}_{1,\mathbb{E}}\mathcal{J}_{1,\mathbb{B}} = \mathcal{J}_{1,\mathbb{B}} \tag{E.2}$$

which leads to

$$\mathcal{J}_{p,\mathbb{E}}\mathcal{J}_{1,\mathbb{B}} = \mathcal{J}_{1,\mathbb{B}} \quad \forall p \text{ and } \mathbb{E}\subseteq\mathbb{B}. \tag{E.3}$$

Using Eq. (E.3) we then have:

$$\begin{aligned}\mathcal{J}_{p,\mathbb{E}}\mathcal{J}_{q,\mathbb{E}} &= (1-q)\mathcal{J}_{p,\mathbb{E}} + q\mathcal{J}_{1,\mathbb{E}} \\ &= (1-q)(1-p)\mathcal{I} + (q+p-pq)\,\mathcal{J}_{1,\mathbb{E}} \\ &= \mathcal{J}_{q+p-pq,\mathbb{E}}\end{aligned}$$

which is still a group error of the same group $\mathbb{E}$ with a modified error probability.

For $\mathcal{J}_{q,\mathbb{E}}$ to be the inverse of $\mathcal{J}_{p,\mathbb{E}}$, we require:

$$\begin{aligned}q+p-pq &= 0 \\ q &= -\frac{p}{1-p}.\end{aligned}$$

Hence,

$$\mathcal{J}_{p,\mathbb{E}}^{-1} = \mathcal{J}_{-\frac{p}{1-p},\mathbb{E}} \tag{E.4}$$

E.3 Removing Subgroup Components in a Group Channel

E.3.1 Form of the quasi-probability channel

For a group error channel of the group $\mathbb{E}$, we would want to remove the error components $\mathbb{Q}$ in the channel where $\mathbb{Q}$ is a subgroup of $\mathbb{E}$. i.e. Thus we want to transform the channel:

$$\mathcal{J}_{p,\mathbb{E}} = (1-p)\mathcal{I} + p\mathcal{J}_{1,\mathbb{E}}$$

into the channel:

$$\mathcal{V}_q = (1-q)\mathcal{I} + q\mathcal{V}_1$$

where

$$\begin{aligned}\mathcal{V}_1 &= \frac{1}{|\mathbb{E}|-|\mathbb{Q}|}\sum_{V\in\mathbb{E},V\notin\mathbb{Q}}\overline{V} \\ &= \frac{1}{|\mathbb{E}|-|\mathbb{Q}|}\left(|\mathbb{E}|\mathcal{J}_{1,\mathbb{E}} - |\mathbb{Q}|\mathcal{J}_{1,\mathbb{Q}}\right).\end{aligned} \tag{E.5}$$

Using Eq. (E.2), we have

$$\mathcal{V}_q \mathcal{J}_{1,\mathbb{E}} = \mathcal{J}_{1,\mathbb{E}},$$

along with Eq. (E.4), the channel we need to perform to transform $\mathcal{J}_{p,\mathbb{E}}$ to $\mathcal{V}$ will be:

$$\begin{aligned} \mathcal{V}_q \mathcal{J}_{p,\mathbb{E}}^{-1} = \mathcal{V}_q \mathcal{J}_{\frac{p}{p-1},\mathbb{E}} &= \frac{1}{1-p}\left(\mathcal{V}_q - p\mathcal{J}_{1,\mathbb{E}}\right) \\ &= \frac{1}{1-p}\Big[(1-q)\,\mathcal{I} + q\mathcal{V}_1 - p\mathcal{J}_{1,\mathbb{E}}\Big]. \end{aligned} \tag{E.6}$$

Using Eq. (E.5), we have:

$$\mathcal{J}_{1,\mathbb{E}} = \frac{|\mathbb{E}| - |\mathbb{Q}|}{|\mathbb{E}|}\mathcal{V}_1 + \frac{|\mathbb{Q}|}{|\mathbb{E}|}\mathcal{J}_{1,\mathbb{Q}}.$$

It turns Eq. (E.6) into:

$$\begin{aligned} \mathcal{V}_q \mathcal{J}_{p,\mathbb{E}}^{-1} &= \frac{1}{1-p}\left[(1-q)\,\mathcal{I} + (q-p_d)\,\mathcal{V}_1 - \frac{|\mathbb{Q}|}{|\mathbb{E}|}p\mathcal{J}_{1,\mathbb{Q}}\right] \\ &= \frac{1}{1-p}\left[(1-q)\,\overline{I} + \frac{q-p_d}{|\mathbb{E}| - |\mathbb{Q}|}\sum_{\substack{V\in\mathbb{E}\\ V\notin\mathbb{Q}}}\overline{V} - \frac{p}{|\mathbb{E}|}\sum_{Q\in\mathbb{Q}}\overline{Q}\right]. \end{aligned} \tag{E.7}$$

in which $p_d = \frac{|\mathbb{E}|-|\mathbb{Q}|}{|\mathbb{E}|}p$ as defined in Eq. (8.2), which is the probability of the errors in $V \in \mathbb{E} - \mathbb{Q}$ occurring.

At $q = p_d$, the channel we need to apply is just:

$$\mathcal{V}_{p_d} \mathcal{J}_{p,\mathbb{E}}^{-1} = \frac{1}{1-p}\left[(1-p_d)\,\mathcal{I} - \frac{|\mathbb{Q}|}{|\mathbb{E}|}p\mathcal{J}_{1,\mathbb{Q}}\right]. \tag{E.8}$$

i.e. we only need to apply components in $\mathbb{Q}$ in our quasi-probability channel.

And using Eq. (E.5), our resultant channel is simply:

$$\begin{aligned} \mathcal{V}_{p_d} &= (1-p_d)\mathcal{I} + p_d\mathcal{V}_1 \\ &= (1-p_d)\,\overline{I} + \frac{p_d}{|\mathbb{E}| - |\mathbb{Q}|}\sum_{V\in\mathbb{E}, V\notin\mathbb{Q}}\overline{V}\,. \end{aligned}$$

same as what we derived in Section 8.4.2.

E.3.2 Cost of the quasi-probability channel

Decomposing the quasi-probability channel that we need to implement into its components:

$$\mathcal{V}_q \mathcal{J}_{p,\mathbb{E}}^{-1} = \sum_{E\in\mathbb{E}} \beta_E \overline{E}$$

and compared to Eq. (E.7), we have:

$$\beta_Q = -\frac{p}{(1-p)\,|\mathbb{E}|} \leq 0 \quad \forall Q \in \mathbb{Q}, Q \neq I$$
$$\beta_V = \frac{1}{1-p}\left(\frac{q-p_d}{|\mathbb{E}|-|\mathbb{Q}|}\right) \quad \forall V \in \mathbb{E}, V \notin \mathbb{Q}.$$

The normalisation factor of implementing the quasi-probability channel $\mathcal{V}\mathcal{J}_{p,\mathbb{E}}^{-1}$ is:

$$B = \sum_{E\in\mathbb{E}} |\beta_E| = 1 + 2\left(\sum_{\beta_E<0} |\beta_E|\right) \tag{E.9}$$

where we have used $\sum_{E\in\mathbb{E}} \beta_E = 1$.

It means that the cost of implementing the quasi-probability channel is split into two case by the threshold error rate $q = p_d$ of the final channel $\mathcal{V}_q$.

Targeted error rate above or at the threshold

This means:

$$q \geq p_d \quad \Rightarrow \quad \beta_V \geq 0.$$

Now using Eq. (E.9), we have

$$\begin{aligned} B_1 &= 1 + 2\,(|\mathbb{Q}|-1)\,|\beta_Q| \\ &= 1 + 2\frac{(|\mathbb{Q}|-1)\,p}{|\mathbb{E}|\,(1-p)}. \end{aligned}$$

The cost to implementing the transformation channel $\mathcal{V}_q\mathcal{J}_{p,\mathbb{E}}^{-1}$ is then

$$C_{Q1,q} = B_1^2 \approx 1 + 4\frac{(|\mathbb{Q}|-1)\,p}{|\mathbb{E}|} + \mathcal{O}(p^2).$$

i.e. the cost to implementing the quasi probability channel in this regime is independent of q. However, the exact quasi-probability channel we need to implement will still change with q.

Uses Eq. (8.5) and Eq. (8.2), we have:

$$C_{Q1,q} = 1 + 4\,(p_\epsilon - p_d)\,.$$

Target error rate below the threshold

This means:

$$q < p_d \quad \Rightarrow \quad \beta_V < 0$$

Now using Eq. (E.9), we have

$$\begin{aligned} B_2 &= 1 + 2\left(|\mathbb{Q}| - 1\right)|\beta_Q| + 2\left(|\mathbb{E}| - |\mathbb{Q}|\right)|\beta_V| \\ &= B_1 + \frac{2}{1-p}\left(\frac{|\mathbb{E}| - |\mathbb{Q}|}{|\mathbb{E}|}p - q\right) \\ &= 1 + 2\frac{\left(|\mathbb{E}| - 1\right)p}{|\mathbb{E}|\left(1-p\right)} - \frac{2q}{1-p} \end{aligned}$$

The cost to implementing the transformation channel $\mathcal{V}_q\mathcal{J}_{p,\mathbb{E}}^{-1}$ is just

$$\begin{aligned} C_{Q1,q} = B_2^2 &= 1 + 4\left(\frac{|\mathbb{E}| - 1}{|\mathbb{E}|}p - q\right) + \mathcal{O}(p^2) + \mathcal{O}(pq) \\ &\approx 1 + 4\left(p_\epsilon - q\right) \end{aligned}$$

where we have use the expression of p_ϵ in Eq. (8.5). Here $1 + 4p_\epsilon$ is the cost to invert the entire channel $\mathcal{J}_{p,\mathbb{E}}$. For each bit of remaining error probability q in our final channel $\mathcal{V}_q$, we can reduce the cost factor by $4q$ until we hit the threshold $q = p_d$. This is the same as what we obtained in Eq. (6.17).

E.4 Decay of Pauli Expectation Value under Group Channels

E.4.1 Overall derivation

In Section 8.3.1, we are looking at a circuit of the form $U = \prod_{m=M}^{1} V_m$ with each gate decomposed into their Pauli components: $V_m = \sum_{k_m} \alpha_{mk_m} G_{mk_m}$.

Now if a pure group error $\mathcal{J}_{1,\mathbb{E}_f}$ happens between V_f and V_{f+1}, and we denote

$$\{a : b\} = \{a, a+1, \cdots b\}$$

$$U_{\mathbb{M}} = \prod_{m\in\mathbb{M}} V_m, \quad G_{\vec{k}_{\mathbb{M}}} = \prod_{m\in\mathbb{M}} G_{mk_m}, \quad \alpha_{\vec{k}_{\mathbb{M}}} = \prod_{m\in\mathbb{M}} \alpha_{mk_m},$$

then the expectation value of a Pauli observable O is:

$$\begin{aligned}
&\mathrm{Tr}\Big(U_{f+1:M}\mathcal{J}_{1,\mathbb{E}_f}(U_{1:f}\rho U_{1:f}^\dagger)U_{f+1:M}^\dagger O\Big)\\
=&\sum_{\vec{j},\vec{k}}\alpha_{\vec{j}}^*\alpha_{\vec{k}}\,\mathrm{Tr}\Big(G_{\vec{k}_{f+1:M}}\mathcal{J}_{1,\mathbb{E}_f}(G_{\vec{k}_{1:f}}\rho G_{\vec{j}_{1:f}}^\dagger)G_{\vec{j}_{f+1:M}}^\dagger O\Big)\\
=&\sum_{\vec{j},\vec{k}}X_f(\vec{j},\vec{k})\alpha_{\vec{j}}^*\alpha_{\vec{k}}\,\mathrm{Tr}\Big(\rho G_{\vec{j}}^\dagger O G_{\vec{k}}\Big)
\end{aligned}$$

where in the last step we have use Eq. (8.8) and we have defined:

$$X_f(\vec{j},\vec{k})=\frac{1}{2^{|\widetilde{\mathbb{E}}_f|}}\prod_{\widetilde{E}\in\widetilde{\mathbb{E}}_f}\Big(1+\eta(\widetilde{E},G_{\vec{j}_{f+1:M}}^\dagger O G_{\vec{k}_{f+1:M}})\Big).$$

Here we see that the effect of the group error $\mathcal{J}_{1,\mathbb{E}_f}$ is simply removing the terms in Eq. (8.9) whose effective observable $G_{\vec{j}_{f+1:M}}^\dagger O G_{\vec{k}_{f+1:M}}$ right after the noise does not commute with all elements in $\mathbb{E}_f$. Note the the effect of the noise does not relate to the gates implemented before it at all.

Denoting $\langle O_{\mathbb{L}}\rangle$ as the expectation value we obtain when the set of error location is $\mathbb{L}$, we have:

$$\langle O_{\mathbb{L}}\rangle=\sum_{\vec{j},\vec{k}}\left(\prod_{f\in\mathbb{L}}X_f(\vec{j},\vec{k})\right)\alpha_{\vec{j}}^*\alpha_{\vec{k}}\,\mathrm{Tr}\Big(\rho G_{\vec{j}}^\dagger O G_{\vec{k}}\Big).$$

Recall that $\left\langle O_{|\mathbb{L}|=l}\right\rangle$ is the expectation value we obtain when there are l errors in the circuit, regardless of the error location. By definition we have:

$$\begin{aligned}
\left\langle O_{|\mathbb{L}|=l}\right\rangle&=\frac{1}{{}^{M}C_l}\sum_{|\mathbb{L}|=l}\langle O_{\mathbb{L}}\rangle\\
&=\sum_{\vec{j},\vec{k}}\left(\frac{1}{{}^{M}C_l}\sum_{|\mathbb{L}|=l}\prod_{f\in\mathbb{L}}X_f(\vec{j},\vec{k})\right)\alpha_{\vec{j}}^*\alpha_{\vec{k}}\,\mathrm{Tr}\Big(\rho G_{\vec{j}}^\dagger O G_{\vec{k}}\Big).
\end{aligned}\tag{E.10}$$

We will define

$$r_{\vec{j},\vec{k}}=\frac{1}{M}\sum_{f=1}^{M}X_f(\vec{j},\vec{k})$$

which is the average fraction of error location that will not affect the measurement result for given $\vec{j}$ and $\vec{k}$.

As proven in Appendix E.4.2, in the limit of large M and non-vanishing $r_{\vec{j},\vec{k}}$, we have:

$$\frac{1}{{}^{M}C_l}\sum_{|\mathbb{L}|=l}\prod_{f\in\mathbb{L}}X_f(\vec{j},\vec{k})\approx r^l_{\vec{j},\vec{k}}.$$

Thus Eq. (E.10) can be approximated as:

$$\begin{aligned}\left\langle O_{|\mathbb{L}|=l}\right\rangle &\approx \sum_{\vec{j},\vec{k}} r^l_{\vec{j},\vec{k}}\alpha^*_{\vec{j}}\alpha_{\vec{k}}\operatorname{Tr}\left(\rho G^\dagger_{\vec{j}}OG_{\vec{k}}\right)\\ &= 2\sum_{\vec{j}>\vec{k}} r^l_{\vec{j},\vec{k}}\operatorname{Re}\left\{\alpha^*_{\vec{j}}\alpha_{\vec{k}}\operatorname{Tr}\left(\rho G^\dagger_{\vec{j}}OG_{\vec{k}}\right)\right\}\end{aligned} \tag{E.11}$$

where we have use $r_{\vec{j},\vec{k}} = r_{\vec{k},\vec{j}}$ from the definition of $r_{\vec{j},\vec{k}}$.

E.4.2 Expansion of the sum of Bernoulli samples

We denote X_f as the fth sample from a Bernoulli distribution. From the total M samples taken, we have estimate a success probability of r:

$$\frac{\sum_f X_f}{M} = r.$$

Now we define

$$Y_l = \sum_{|\mathbb{L}|=l}\prod_{f\in\mathbb{L}}X_f.$$

Then Y_1 is just the sum of these samples:

$$Y_1 = \sum_f X_f = Mr.$$

Using multinomial expansion, we have

$$Y_1^l = \left(\sum_{f=1}^M X_f\right)^l = \sum_{\sum_f n_f=l}\binom{l}{n_1,n_2,\cdots,n_M}\prod_{f=1}^M X_f^{n_f}$$

where the multinomial coefficient is

$$\binom{l}{n_1,n_2,\cdots,n_M} = \frac{l!}{\prod_{f=1}^M n_f!}.$$

It is the coefficient of the term $\prod_{f=1}^M X_f^{n_f}$, which is the number of ways the distribute l distinct balls (the total power of l) into M distinct bins (M terms of X_f for $f\in\{1,2,3,\cdots,M\}$) such that the number of balls in bin f is n_f (the power of X_f is n_f).

Now using $X_f^n = X_f$ for any n, we have

$$\left(\sum_{f=1}^{M} X_f\right)^l = \sum_{b=1}^{l} \left\{ {l \atop b} \right\} b! \sum_{|\mathbb{L}|=b} \prod_{f\in\mathbb{L}} X_f$$
$$Y_1^l = \sum_{b=1}^{l} \left\{ {l \atop b} \right\} b! Y_b$$

where $\mathbb{L}$ is the subset of non-empty bins and $\left\{ {l \atop b} \right\}$ is the Stirling number of the second kind, which is the number of ways to distribute l *distinct* balls into these b *identical* bins such that none are empty. And $\left\{ {l \atop b} \right\} b!$ is just the number of ways of distribute l *distinct* balls into these b *distinct* bins such that none are empty.

Hence,

$$\left\{ {l \atop l} \right\} l! Y_l = Y_1^l - \sum_{b=1}^{l-1} \left\{ {l \atop b} \right\} b! Y_b$$
$$l! Y_l = Y_1^l - \sum_{b=1}^{l-1} \left\{ {l \atop b} \right\} b! Y_b \tag{E.12}$$

where we have used $\left\{ {l \atop l} \right\} = 1$.

Hence, in the limit of large M and assuming the success sample fraction r is non-vanishing, we have:

$$\frac{Y_l}{{}^M C_l} \approx \frac{l! Y_l}{M^l} \approx \frac{Y_1^l + \mathcal{O}(Y_1^{l-1})}{M^l} = r^l + \mathcal{O}(M^{-1}).$$

From the definition of Y_l we have:

$$\frac{1}{{}^M C_l} \sum_{|\mathbb{L}|=l} \prod_{f\in\mathbb{L}} X_f \approx r^l + \mathcal{O}(M^{-1}).$$

E.5 Decay of Pauli Expectation Value under Pauli Channels

Any Pauli channel can be decomposed into a set of group channels basis:

$$\mathcal{P}_p = (1-p)\,\mathcal{I} + p \sum_j \beta_j \mathcal{J}_{1,\mathbb{E}_j} \tag{E.13}$$

where $\sum_j \beta_j = 1$. In the most naive way, we can have $\mathbb{E}_j = \{I, G_j\}$ for the Pauli operators G_j and $\frac{p\beta_j}{2}$ being the error rate of the Pauli error G_j.

For a circuit with M Pauli error locations, different error locations might experience different Pauli noise of different strengths. We will denote the union of the group channel basis needed to describe all of these Pauli channels as $\{\mathcal{J}_{1,\mathbb{E}_j} \mid 1 \leq j \leq J\}$. Now we can split each Pauli error location into J group error locations, with the jth location can only have the group error $\mathcal{J}_{1,\mathbb{E}_j}$ occurring. Hence, we have in total MJ group error locations. Using the same arguments in Appendix E.4, we can obtain the same equation as Eq. (E.11), which will lead to Eq. (8.11):

$$\left\langle O_{|\mathbb{L}|=l} \right\rangle = \sum_{k=1}^{K} A_k r_k^l, \tag{E.14}$$

but now $|\mathbb{L}|$, l and r_k are defined in terms of *group* error locations instead of simply error locations.

We can also follow the same arguments in Section 6.1, but focusing on group errors instead of simply errors. When a Pauli channel is written in the form of Eq. (E.13), the error rate p defined in this way is the probability that one group error occurs (which could be any one of the basis group errors.). i.e. the number of group error at each Pauli error location is a Bernoulli variable with the success probability p. The mean circuit *group* error count is just the sum of the group error probability of all the error locations, which we will denote as μ. Again taking the NISQ limit and using the Le Cam's theorem, the number of *group* errors occurring in the circuit will follow a Poisson distribution with the mean μ, which is just Eq. (6.2):

$$P_l = e^{-\mu} \frac{\mu^l}{l!}. \tag{E.15}$$

Combining with Eq. (E.14), we can again obtain the exponential decay of the expectation value with increase of mean circuit (group) error count μ just like in Section 8.3.2.

E.6 Cost of Error Extrapolation

E.6.1 Cost for two-point extrapolation

Suppose for an observable O, we can estimate its expectation value by combining the expectation value of two other observables A and B. Then by denoting the

estimate as $\langle O_0 \rangle$, we have:

$$\langle O \rangle \approx \langle O_0 \rangle := f(\langle A \rangle, \langle B \rangle) \tag{E.16}$$

for some estimation function f.

Now suppose we take N samples in total and α is the fraction of A samples within, then using $\overline{A}$, $\overline{B}$ to denote the sample averages, we can obtain the sample estimate of $\langle O_0 \rangle$ (and thus $\langle O \rangle$):

$$\overline{O}_0 := f(\overline{A}, \overline{B}). \tag{E.17}$$

Note that we have abuse the notation here since $\langle O_0 \rangle$ and $\overline{O}_0$ are *not* the expectation value and the sample average of some observable O_0. There is no such an observable. Rather, $\overline{O}_0$ is *exactly defined* as the estimate of $\langle O \rangle$ after N total samples of A and B using the equation above, and similarly $\langle O_0 \rangle$ is defined as the case $N \to \infty$. Note that $O_0 \not\equiv O$ as well since O is an actual observable of the noiseless computation that lead us to $\langle O \rangle$.

We will try to compare the variance of the sample estimate $\overline{O}_0$ against the variance of the noiseless sample average $\overline{O}$. The variances of various sampling averages follow these equations:

$$\begin{aligned}
\operatorname{Var}\left[\overline{A}\right] &= \frac{\operatorname{Var}[A]}{\alpha N} \\
\operatorname{Var}\left[\overline{B}\right] &= \frac{\operatorname{Var}[B]}{(1-\alpha)N} \\
\operatorname{Var}\left[\overline{O}\right] &= \frac{\operatorname{Var}[O]}{N}.
\end{aligned}$$

Hence, assuming $\operatorname{Var}[A] \approx \operatorname{Var}[B] \approx \operatorname{Var}[O]$, the variance of the sample estimate is:

$$\begin{aligned}
\operatorname{Var}\left[\overline{O}_0\right] &= \left(\frac{\partial \overline{O}_0}{\partial \overline{A}}\right)^2 \operatorname{Var}\left[\overline{A}\right] + \left(\frac{\partial \overline{O}_0}{\partial \overline{B}}\right)^2 \operatorname{Var}\left[\overline{B}\right] \\
&= \left(\left(\frac{\partial \overline{O}_0}{\partial \overline{A}}\right)^2 \frac{1}{\alpha} + \left(\frac{\partial \overline{O}_0}{\partial \overline{B}}\right)^2 \frac{1}{(1-\alpha)}\right) \operatorname{Var}\left[\overline{O}\right].
\end{aligned}$$

Hence, when sampling for $\overline{O}_0$ instead of $\overline{O}$, the additional cost factor is:

$$C(\alpha) = \frac{a^2}{\alpha} + \frac{b^2}{1-\alpha} \tag{E.18}$$

with

$$a = \left|\frac{\partial \overline{O}_0}{\partial \overline{A}}\right|, \quad b = \left|\frac{\partial \overline{O}_0}{\partial \overline{B}}\right|$$

We then have the extremal point being

$$\alpha_\pm = \frac{a}{a \pm b}.$$

Since $\alpha_- = \frac{a}{a-b} \geq 1$, we will only keep α_+. We can also obtain $C''(\alpha_+) \geq 0$, which means that it is a local minimum, whose value is

$$C(\alpha_+) = (a+b)^2. \tag{E.19}$$

i.e. the minimal cost factor is achieved when the fraction of A samples is α_+.

For the naive version of evenly distributing all the samples: $\alpha = 0.5$, we have

$$C(0.5) = 2(a^2 + b^2). \tag{E.20}$$

When compared to the optimal distribution, we have:

$$C(\alpha_+) \leq C(0.5) = 2(a^2+b^2) \leq 2(a+b)^2 = 2C(\alpha_+).$$

i.e. the saving in the number of samples by using optimal sample distribution will be less than half. The saving will be exactly half in the case of $a \gg b$ or $b \gg a$. In practice, a and b is often unknown and thus it is hard to achieve the optimal sample distribution. Hence, in most of the cases in this thesis, we just use the naive sample distribution, which should be of the same order of magnitude as the optimal case.

E.6.2 Cost of exponential extrapolation

For an observable O, we can estimate its expectation value by probing at the error rate μ and $\lambda\mu$ and fitting with an exponential curve to get the zero-noise value:

$$\langle O \rangle \approx \langle O_0 \rangle := \left(\frac{\langle O_\mu \rangle^\lambda}{\langle O_{\lambda\mu} \rangle}\right)^{\frac{1}{\lambda - 1}} \tag{E.21}$$

Note that $\langle O \rangle$ will only be exactly the same as $\langle O_0 \rangle$ if the observable follows strictly a single exponential decay with increased noise.

Now following Appendix E.6.1 with $A = O_\mu$, $B = O_{\lambda\mu}$ and using $\overline{O}_\mu \approx \overline{O}_0 e^{-\gamma\mu}$, we have:

$$\begin{aligned} a &= \left|\frac{\partial \overline{O}_0}{\partial \overline{O}_\mu}\right| = \frac{\lambda}{\lambda-1}\left(\frac{\overline{O}_\mu}{\overline{O}_{\lambda\mu}}\right)^{\frac{1}{\lambda-1}} = \frac{\lambda}{\lambda-1}e^{\gamma\mu} \\ b &= \left|\frac{\partial \overline{O}_0}{\partial \overline{O}_{\lambda\mu}}\right| = \frac{1}{\lambda-1}\left(\frac{\overline{O}_\mu}{\overline{O}_{\lambda\mu}}\right)^{\frac{\lambda}{\lambda-1}} = \frac{1}{\lambda-1}e^{\lambda\gamma\mu}. \end{aligned} \tag{E.22}$$

Hence, the sampling cost factor of exponential extrapolation can be obtained using Eq. (E.20):

$$C_E = 2\left(a^2+b^2\right) = 2\frac{\lambda^2 e^{2\gamma\mu} + e^{2\lambda\gamma\mu}}{(\lambda-1)^2}. \tag{E.23}$$

E.6.3 Cost of quasi-probability with exponential extrapolation

As outlined in Section 8.4.1, we use quasi-probability to suppress the error rate from μ to $\nu = \frac{\mu}{\lambda}$ and then use the point at $\mu = \lambda\nu$ and ν to perform exponential extrapolation. Thus we have the same equation as Eq. (E.21) with $\mu \to \nu$:

$$\langle O_0 \rangle = \left(\frac{\langle O_\nu \rangle^\lambda}{\langle O_{\lambda\nu} \rangle}\right)^{\frac{1}{\lambda-1}}. \tag{E.24}$$

Here $\lambda\nu = \mu$ is the original error rate and thus $\langle O_{\lambda\nu} \rangle$ will be obtained via direct sampling, while ν is the quasi-probability suppressed error rate, thus $\langle O_\nu \rangle$ will be obtain via quasi-probability. Thus using Eq. (6.13), we have

$$\langle O_\nu \rangle = Q \langle O_Q \rangle .$$

Now following Appendix E.6.1 with $A = O_Q$, $B = O_{\lambda\nu}$ and using Eq. (E.22) with ν in place of μ, we have:

$$\begin{aligned} a &= \left|\frac{\partial \overline{O}_0}{\partial \overline{O}_Q}\right| = \left|\frac{\partial \overline{O}_0}{\partial \overline{O}_\nu}\frac{\mathrm{d}\overline{O}_\nu}{\mathrm{d}\overline{O}_Q}\right| = \frac{\lambda}{\lambda-1}e^{\gamma\nu}Q \\ b &= \left|\frac{\partial \overline{O}_0}{\partial \overline{O}_{\lambda\nu}}\right| = \frac{1}{\lambda-1}e^{\lambda\gamma\nu}. \end{aligned}$$

Using Eq. (6.14), we have $Q = \sqrt{C_{Q,\nu}}$. From Eq. (6.20), we have $C_{Q,\nu} = e^{4(\mu_\epsilon - \nu_\epsilon)}$. Since in extrapolation, we need to suppress all error components evenly, we have $\mu_\epsilon = \lambda \nu_\epsilon$ just like the relation between μ and ν. Hence, we have:

$$\begin{aligned} C_{Q,\nu} &= e^{4(\lambda-1)\nu_\epsilon} \\ Q = \sqrt{C_{Q,\nu}} &= e^{2(\lambda-1)\nu_\epsilon}, \end{aligned}$$

which gives

$$\begin{aligned} a &= \left| \frac{\partial \overline{O}_0}{\partial \overline{O}_Q} \right| = \left| \frac{\partial \overline{O}_0}{\partial \overline{O}_\nu} \frac{\mathrm{d}\overline{O}_\nu}{\mathrm{d}\overline{O}_Q} \right| = \frac{\lambda}{\lambda - 1} e^{\gamma\nu + 2(\lambda-1)\nu_\epsilon} \\ b &= \left| \frac{\partial \overline{O}_0}{\partial \overline{O}_{\lambda\nu}} \right| = \frac{1}{\lambda - 1} e^{\lambda\gamma\nu}. \end{aligned}$$

Hence, the sampling cost factor of quasi-probability with exponential extrapolation can be obtained using Eq. (E.20):

$$\begin{aligned} C_{QE} = 2\left(a^2 + b^2\right) &= 2\frac{\lambda^2 e^{2\gamma\nu + 4(\lambda-1)\nu_\epsilon} + e^{2\lambda\gamma\nu}}{(\lambda-1)^2} \\ &= 2\frac{\lambda^2 e^{\frac{2}{\lambda}[\gamma\mu + 2(\lambda-1)\mu_\epsilon]} + e^{2\gamma\mu}}{(\lambda-1)^2}. \end{aligned} \tag{E.25}$$

E.6.4 Cost of hyperbolic extrapolation

Performing error extrapolation using Eq. (8.21), we have:

$$\langle O_0 \rangle = \mathrm{sgn}\left(\langle O_{c,\nu} \rangle\right) \sqrt{\langle O_{c,\nu} \rangle^2 \cosh^2(\nu) - \langle O_{s,\nu} \rangle^2 \sinh^2(\nu)}.$$

Following the arguments in Appendix E.6.1 with $A = O_{c,\nu}$, $B = O_{s,\nu}$, and using Eq. (8.20), we have

$$\begin{aligned} a &= \left| \frac{\partial \overline{O}_0}{\partial \overline{O}_{c,\nu}} \right| = \cosh^2(\nu) \frac{\overline{O}_{c,\nu}}{\overline{O}_0} = \cosh(\nu)\cosh((1-\gamma)\,\nu) \\ b &= \left| \frac{\partial \overline{O}_0}{\partial \overline{O}_{s,\nu}} \right| = \sinh^2(\nu) \frac{\overline{O}_{s,\nu}}{\overline{O}_0} = \sinh(\nu)\sinh((1-\gamma)\,\nu). \end{aligned}$$

The fraction of $O_{c,\nu}$ samples among all the samples can be obtained from Eq. (8.18):

$$\alpha = e^{-\nu}\cosh(\nu).$$

Hence, the cost factor can be obtain using Eq. (E.18):

$$\begin{aligned} C_{H,\nu} &= \frac{a^2}{\alpha} + \frac{b^2}{1-\alpha} \\ &= \Big(\cosh(\nu) \cosh^2((1-\gamma)\,\nu) + \sinh(\nu) \sinh^2((1-\gamma)\,\nu) \Big) e^{\nu}. \end{aligned}$$

In this thesis, we will use its *upper bound* as the sampling cost instead for a simpler expression:

$$\begin{aligned} C_{H,\nu} &\leq \Big(\cosh^2((1-\gamma)\,\nu) + \sinh^2((1-\gamma)\,\nu) \Big) \cosh(\nu) e^{\nu} \\ &= \cosh(2\,(1-\gamma)\,\nu) \cosh(\nu) e^{\nu}. \end{aligned}$$

E.7 Comparison Between Error Mitigation Techniques

For a given error mitigation technique whose average estimation errors for a Pauli observable is ϵ and whose sampling cost is C, we can define Δ to be the effective observable that represents the error in each circuit run for obtaining the error-mitigated Pauli observable, which has:

$$\begin{aligned} \langle \Delta \rangle &\sim \epsilon \\ \mathrm{Var}\left[\Delta\right] &\sim C. \end{aligned}$$

In another word, ϵ is the systematic error of the error mitigation techniques while C is the strength of the random errors (shot noise) of the technique.

We will label the average error after N sample runs as $\overline{\Delta}$, for which we have:

$$\begin{aligned} \left\langle \overline{\Delta} \right\rangle &\sim \epsilon \\ \mathrm{Var}\left[\overline{\Delta}\right] &\sim \frac{C}{N}. \end{aligned}$$

Hence, the expected square errors of applying the error mitigation technique with N samples is thus:

$$\begin{aligned} \left\langle \overline{\Delta}^2 \right\rangle &= \mathrm{Var}\left[\overline{\Delta}\right] + \left\langle \overline{\Delta} \right\rangle^2 \\ &\sim \frac{C}{N} + \epsilon^2 \end{aligned}$$

Now suppose we have two different error mitigation techniques, and w.l.o.g. we will assume technique 1 will have lower systematic errors $\epsilon_1 \leq \epsilon_2$. If $C_1 < C_2$, then technique 1 is obviously the better technique. If $C_2 \leq C_1$ however, there is a break-even point which both methods have similar mean square errors:

$$\begin{aligned} \left\langle \overline{\Delta}_1^2 \right\rangle &\sim \left\langle \overline{\Delta}_2^2 \right\rangle \\ N^* \sim \frac{C_1 - C_2}{\epsilon_2^2 - \epsilon_1^2} &\approx \frac{C_1}{\epsilon_2^2} + \mathcal{O}\left(\frac{C_2}{C_1}\right) + \mathcal{O}\left(\frac{\epsilon_1^2}{\epsilon_2^2}\right). \end{aligned}$$

i.e. N^* is roughly the number of samples needed using technique 1 to reach a shot noise level that is equal to the systematic error of technique 2 (ϵ_2). When the number of samples $N \lesssim N^*$, the shot noise will dominate and thus we will choose technique 2 over technique 1 due to the lower sampling cost. When the number of samples increase and reach $N \gtrsim N^*$, then the systematic errors will dominate over shot noise and thus we will choose technique 1 over technique 2 due to the lower systematic errors.

References

[1] R. P. Feynman, "Simulating physics with computers", International Journal of Theoretical Physics **21**, 467–488 (1982).

[2] D. Deutsch and R. Penrose, "Quantum theory, the Church–Turing principle and the universal quantum computer", Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences **400**, 97–117 (1985).

[3] P. Shor, "Algorithms for quantum computation: discrete logarithms and factoring", in Proceedings 35th Annual Symposium on Foundations of Computer Science (Nov. 1994), pp. 124–134.

[4] L. K. Grover, "A Fast Quantum Mechanical Algorithm for Database Search", in Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing, STOC '96 (1996), pp. 212–219.

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

[6] P. W. Shor, "Scheme for reducing decoherence in quantum computer memory", Physical Review A **52**, R2493–R2496 (1995).

[7] A. M. Steane, "Error Correcting Codes in Quantum Theory", Physical Review Letters **77**, 793–797 (1996).

[8] F. Arute et al., "Quantum supremacy using a programmable superconducting processor", Nature **574**, 505–510 (2019).

[9] D. Greenbaum, "Introduction to Quantum Gate Set Tomography", arXiv:1509.02921 [quant-ph] (2015).

[10] D. Gottesman, "The Heisenberg Representation of Quantum Computers", arXiv:quant-ph/9807006 (1998).

[11] J. Wallman, C. Granade, R. Harper, and S. T. Flammia, "Estimating the coherence of noise", New Journal of Physics **17**, 113020 (2015).

[12] Y. R. Sanders, J. J. Wallman, and B. C. Sanders, "Bounding quantum gate error rate based on reported average fidelity", New Journal of Physics **18**, 012002 (2015).

[13] M. Gutiérrez and K. R. Brown, "Comparison of a quantum error-correction threshold for exact and approximate errors", Physical Review A **91**, 022335 (2015).

[14] R. Kueng, D. M. Long, A. C. Doherty, and S. T. Flammia, "Comparing Experiments to the Fault-Tolerance Threshold", Physical Review Letters **117**, 170502 (2016).

[15] S. Bravyi, M. Englbrecht, R. König, and N. Peard, "Correcting coherent errors with surface codes", npj Quantum Information **4**, 55 (2018).

[16] D. Greenbaum and Z. Dutton, "Modeling coherent errors in quantum error correction", Quantum Science and Technology **3**, 015007 (2018).

[17] P. Iyer and D. Poulin, "A small quantum computer is needed to optimize fault-tolerant protocols", Quantum Science and Technology **3**, 030504 (2018).

[18] E. Huang, A. C. Doherty, and S. Flammia, "Performance of quantum error correction with coherent errors", Physical Review A **99**, 022313 (2019).

[19] D. A. Lidar, "Review of Decoherence-Free Subspaces, Noiseless Subsystems, and Dynamical Decoupling", Quantum Information and Computation for Chemistry, 295–354 (2014).

[20] D. Suter and G. A. Álvarez, "Colloquium: Protecting quantum information against environmental noise", Reviews of Modern Physics **88**, 041001 (2016).

[21] H. Bombin and M. A. Martin-Delgado, "Topological Quantum Distillation", Physical Review Letters **97**, 180501 (2006).

[22] C. Chamberland, J. Wallman, S. Beale, and R. Laflamme, "Hard decoding algorithm for optimizing thresholds under general Markovian noise", Physical Review A **95**, 042332 (2017).

[23] D. Poulin, "Optimal and efficient decoding of concatenated quantum block codes", Physical Review A **74**, 10.1103/PhysRevA.74.052333 (2006).

[24] P. Shor, "Fault-tolerant quantum computation", in Proceedings of 37th Conference on Foundations of Computer Science (Oct. 1996), pp. 56–65.

[25] A. M. Steane, "Active Stabilization, Quantum Computation, and Quantum State Synthesis", Physical Review Letters **78**, 2252–2255 (1997).

[26] R. Chao and B. W. Reichardt, "Quantum Error Correction with Only Two Extra Qubits", Physical Review Letters **121**, 050502 (2018).

[27] H. Bombín, "Gauge color codes: optimal transversal gates and gauge fixing in topological stabilizer codes", New Journal of Physics **17**, 083002 (2015).

[28] A. Y. Kitaev, "Fault-tolerant quantum computation by anyons", Annals of Physics **303**, 2–30 (2003).

[29] S. B. Bravyi and A. Y. Kitaev, "Quantum codes on a lattice with boundary", arXiv:quant-ph/9811052 (1998).

[30] J. Edmonds, "Paths, Trees, and Flowers", Canadian Journal of Mathematics **17**, 449–467 (1965/ed).

[31] V. Kolmogorov, "Blossom V: a new implementation of a minimum cost perfect matching algorithm", Mathematical Programming Computation **1**, 43–67 (2009).

[32] H. Bombin, R. S. Andrist, M. Ohzeki, H. G. Katzgraber, and M. A. Martin-Delgado, "Strong Resilience of Topological Codes to Depolarization", Physical Review X **2**, 021004 (2012).

[33] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, "Surface codes: Towards practical large-scale quantum computation", Physical Review A **86**, 032324 (2012).

[34] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, "Topological quantum memory", Journal of Mathematical Physics **43**, 4452–4505 (2002).

[35] D. S. Wang, A. G. Fowler, A. M. Stephens, and L. C. L. Hollenberg, "Threshold error rates for the toric and planar codes", Quantum Information & Computation **10**, 456–469 (2010).

[36] A. G. Fowler, A. M. Stephens, and P. Groszkowski, "High-threshold universal quantum computation on the surface code", Physical Review A **80**, 10.1103/PhysRevA.80.052312 (2009).

[37] A. M. Stephens, "Fault-tolerant thresholds for quantum error correction with the surface code", Physical Review A **89**, 10.1103/PhysRevA.89.022321 (2014).

[38] Z. Cai and S. C. Benjamin, "Constructing Smaller Pauli Twirling Sets for Arbitrary Error Channels", Scientific Reports **9**, 1–11 (2019).

[39] S. J. Beale, J. J. Wallman, M. Gutiérrez, K. R. Brown, and R. Laflamme, "Quantum Error Correction Decoheres Noise", Physical Review Letters **121**, 190501 (2018).

[40] D. M. Debroy, M. Li, M. Newman, and K. R. Brown, "Stabilizer Slicing: Coherent Error Cancellations in Low-Density Parity-Check Stabilizer Codes", Physical Review Letters **121**, 250502 (2018).

[41] C. H. Bennett, G. Brassard, S. Popescu, B. Schumacher, J. A. Smolin, and W. K. Wootters, "Purification of Noisy Entanglement and Faithful Teleportation via Noisy Channels", Physical Review Letters **76**, 722–725 (1996).

[42] C. H. Bennett, D. P. DiVincenzo, J. A. Smolin, and W. K. Wootters, "Mixed-state entanglement and quantum error correction", Physical Review A **54**, 3824–3851 (1996).

[43] E. Knill, D. Leibfried, R. Reichle, J. Britton, R. B. Blakestad, J. D. Jost, C. Langer, R. Ozeri, S. Seidelin, and D. J. Wineland, "Randomized benchmarking of quantum gates", Physical Review A **77**, 012307 (2008).

[44] J. Emerson, M. Silva, O. Moussa, C. Ryan, M. Laforest, J. Baugh, D. G. Cory, and R. Laflamme, "Symmetrized Characterization of Noisy Quantum Processes", Science **317**, 1893–1896 (2007).

[45] E. Magesan, J. M. Gambetta, and J. Emerson, "Scalable and Robust Randomized Benchmarking of Quantum Processes", Physical Review Letters **106**, 10.1103/PhysRevLett.106.180504 (2011).

[46] D. Lu et al., "Experimental Estimation of Average Fidelity of a Clifford Gate on a 7-Qubit Quantum Processor", Physical Review Letters **114**, 140505 (2015).

[47] J. Preskill, "Quantum Computing in the NISQ era and beyond", Quantum **2**, 79 (2018).

[48] Y. Li and S. C. Benjamin, "Efficient Variational Quantum Simulator Incorporating Active Error Minimization", Physical Review X **7**, 021050 (2017).

[49] K. Temme, S. Bravyi, and J. M. Gambetta, "Error Mitigation for Short-Depth Quantum Circuits", Physical Review Letters **119**, 180509 (2017).

[50] S. Endo, S. C. Benjamin, and Y. Li, "Practical Quantum Error Mitigation for Near-Future Applications", Physical Review X **8**, 031027 (2018).

[51] A. Kandala, K. Temme, A. D. Córcoles, A. Mezzacapo, J. M. Chow, and J. M. Gambetta, "Error mitigation extends the computational reach of a noisy quantum processor", Nature **567**, 491–495 (2019).

[52] W. Dür, M. Hein, J. I. Cirac, and H.-J. Briegel, "Standard forms of noisy quantum operations via depolarization", Physical Review A **72**, 052326 (2005).

[53] Z. Cai, X. Xu, and S. C. Benjamin, "Mitigating coherent noise using Pauli conjugation", npj Quantum Information **6**, 1–9 (2020).

[54] J. J. Wallman and J. Emerson, "Noise tailoring for scalable quantum computation via randomized compiling", Physical Review A **94**, 052325 (2016).

[55] B. Rahn, A. C. Doherty, and H. Mabuchi, "Exact performance of concatenated quantum codes", Physical Review A **66**, 032304 (2002).

[56] M. Gutiérrez, C. Smith, L. Lulushi, S. Janardan, and K. R. Brown, "Errors and pseudothresholds for incoherent and coherent noise", Physical Review A **94**, 042338 (2016).

[57] H. K. Ng, D. A. Lidar, and J. Preskill, "Combining dynamical decoupling with fault-tolerant quantum computation", Physical Review A **84**, 012305 (2011).

[58] L. Viola, E. Knill, and S. Lloyd, "Dynamical Decoupling of Open Quantum Systems", Physical Review Letters **82**, 2417–2421 (1999).

[59] P. Zanardi, "Symmetrizing evolutions", Physics Letters A **258**, 77–82 (1999).

[60] G. S. Uhrig, "Keeping a quantum bit alive by optimized π-pulse sequences", Physical Review Letters **98**, 100504 (2007).

[61] E. M. Kessler, I. Lovchinsky, A. O. Sushkov, and M. D. Lukin, "Quantum Error Correction for Metrology", Physical Review Letters **112**, 150802 (2014).

[62] W. Dür, M. Skotiniotis, F. Fröwis, and B. Kraus, "Improved Quantum Metrology Using Quantum Error Correction", Physical Review Letters **112**, 080801 (2014).

[63] S. Zhou, M. Zhang, J. Preskill, and L. Jiang, "Achieving the Heisenberg limit in quantum metrology using quantum error correction", Nature Communications **9**, 78 (2018).

[64] X. Bonet-Monroig, R. Sagastizabal, M. Singh, and T. E. O'Brien, "Low-cost error mitigation by symmetry verification", Physical Review A **98**, 062339 (2018).

[65] S. McArdle, X. Yuan, and S. Benjamin, "Error-Mitigated Digital Quantum Simulation", Physical Review Letters **122**, 180501 (2019).

[66] J. R. McClean, Z. Jiang, N. C. Rubin, R. Babbush, and H. Neven, "Decoding quantum errors with subspace expansions", Nature Communications **11**, 636 (2020).

[67] Z. Cai, M. A. Fogarty, S. Schaal, S. Patomäki, S. C. Benjamin, and J. J. L. Morton, "A Silicon Surface Code Architecture Resilient Against Leakage Errors", Quantum **3**, 212 (2019).

[68] D. S. Wang, A. G. Fowler, and L. C. L. Hollenberg, "Surface code quantum computing with error rates over 1%", Physical Review A **83**, 10.1103/PhysRevA.83.020302 (2011).

[69] B. Lekitsch, S. Weidt, A. G. Fowler, K. Mølmer, S. J. Devitt, C. Wunderlich, and W. K. Hensinger, "Blueprint for a microwave trapped ion quantum computer", Science Advances **3**, e1601540 (2017).

[70] C. D. Hill, E. Peretz, S. J. Hile, M. G. House, M. Fuechsle, S. Rogge, M. Y. Simmons, and L. C. L. Hollenberg, "A surface code quantum computer in silicon", Science Advances **1**, e1500707–e1500707 (2015).

[71] J. O'Gorman, N. H. Nickerson, P. Ross, J. J. Morton, and S. C. Benjamin, "A silicon-based surface code quantum computer", npj Quantum Information **2**, npjqi201519 (2016).

[72] J. O'Gorman and E. T. Campbell, "Quantum computation with realistic magic-state factories", Physical Review A **95**, 10.1103/PhysRevA.95.032338 (2017).

[73] L. M. K. Vandersypen, H. Bluhm, J. S. Clarke, A. S. Dzurak, R. Ishihara, A. Morello, D. J. Reilly, L. R. Schreiber, and M. Veldhorst, "Interfacing spin qubits in quantum dots and donors—hot, dense, and coherent", npj Quantum Information **3**, 1–10 (2017).

[74] M. Veldhorst, H. G. J. Eenink, C. H. Yang, and A. S. Dzurak, "Silicon CMOS architecture for a spin-based quantum computer", Nature Communications **8**, 1766 (2017).

[75] R. Li et al., "A crossbar network for silicon quantum dot qubits", Science Advances **4**, eaar3960 (2018).

[76] B. Buonacorsi, Z. Cai, E. B. Ramirez, K. S. Willick, S. M. Walker, J. Li, B. D. Shaw, X. Xu, S. C. Benjamin, and J. Baugh, "Network architecture for a topological quantum computer in silicon", Quantum Science and Technology **4**, 025003 (2019).

[77] N. C. Brown, A. W. Cross, and K. R. Brown, "Critical faults of leakage errors on the surface code", arXiv:2003.05843 [quant-ph] (2020).

[78] F. Motzoi, J. M. Gambetta, P. Rebentrost, and F. K. Wilhelm, "Simple Pulses for Elimination of Leakage in Weakly Nonlinear Qubits", Physical Review Letters **103**, 110501 (2009).

[79] A. Ferrón and D. Domínguez, "Intrinsic leakage of the Josephson flux qubit and breakdown of the two-level approximation for strong driving", Physical Review B **81**, 104505 (2010).

[80] L.-M. Duan, J. I. Cirac, and P. Zoller, "Geometric Manipulation of Trapped Ions for Quantum Computation", Science **292**, 1695–1697 (2001).

[81] H. Haffner, C. Roos, and R. Blatt, "Quantum computing with trapped ions", Physics Reports **469**, 155–203 (2008).

[82] B. H. Fong and S. M. Wandzura, "Universal Quantum Computation and Leakage Reduction in the 3-qubit Decoherence Free Subsystem", Quantum Info. Comput. **11**, 1003–1018 (2011).

[83] S. Mehl, H. Bluhm, and D. P. DiVincenzo, "Fault-tolerant quantum computation for singlet-triplet qubits with leakage errors", Physical Review B **91**, 085419 (2015).

[84] N. C. Brown and K. R. Brown, "Leakage mitigation for quantum error correction using a mixed qubit scheme", Physical Review A **100**, 032325 (2019).

[85] F. K. Malinowski et al., "Fast spin exchange across a multielectron mediator", Nature Communications **10**, 1196 (2019).

[86] S. J. Angus, A. J. Ferguson, A. S. Dzurak, and R. G. Clark, "Gate-Defined Quantum Dots in Intrinsic Silicon", Nano Letters **7**, 2051–2055 (2007).

[87] M. Veldhorst et al., "An addressable quantum dot qubit with fault-tolerant control-fidelity", Nature Nanotechnology **9**, 981–985 (2014).

[88] K. W. Chan et al., "Assessment of a silicon quantum dot spin qubit environment via noise spectroscopy", Physical Review Applied **10**, 10.1103/PhysRevApplied.10.044017 (2018).

[89] C. H. Yang et al., "Silicon qubit fidelities approaching incoherent noise limits via pulse engineering", Nature Electronics **2**, 151 (2019).

[90] E. Kawakami et al., "Gate fidelity and coherence of an electron spin in an Si/SiGe quantum dot with micromagnet", Proceedings of the National Academy of Sciences **113**, 11738–11743 (2016).

[91] J. Yoneda et al., "A quantum-dot spin qubit with coherence limited by charge noise and fidelity higher than 99.9%", Nature Nanotechnology **13**, 102–106 (2018).

[92] C. H. Yang, A. Rossi, R. Ruskov, N. S. Lai, F. A. Mohiyaddin, S. Lee, C. Tahan, G. Klimeck, A. Morello, and A. S. Dzurak, "Spin-valley lifetimes in a silicon quantum dot with tunable valley splitting", Nature Communications **4**, 2069 (2013).

[93] E. Kawakami, P. Scarlino, D. R. Ward, F. R. Braakman, D. E. Savage, M. G. Lagally, M. Friesen, S. N. Coppersmith, M. A. Eriksson, and L. M. K. Vandersypen, "Electrical control of a long-lived spin qubit in a Si/SiGe quantum dot", Nature Nanotechnology **9**, 666–670 (2014).

[94] R. C. C. Leon et al., "Coherent spin control of s-, p-, d- and f-electrons in a silicon quantum dot", Nature Communications **11**, 797 (2020).

[95] K. M. Itoh and H. Watanabe, "Isotope engineering of silicon and diamond for quantum computing and sensing applications", MRS Communications **4**, 143–157 (2014).

[96] M. Veldhorst et al., "A two-qubit logic gate in silicon", Nature **526**, 410–414 (2015).

[97] C. Jones, M. A. Fogarty, A. Morello, M. F. Gyure, A. S. Dzurak, and T. D. Ladd, "Logical Qubit in a Linear Array of Semiconductor Quantum Dots", Physical Review X **8**, 021058 (2018).

[98] W. M. Witzel, I. Montaño, R. P. Muller, and M. S. Carroll, "Multiqubit gates protected by adiabaticity and dynamical decoupling applicable to donor qubits in silicon", Physical Review B **92**, 081407 (2015).

[99] N. Khaneja, T. Reiss, C. Kehlet, T. Schulte-Herbrüggen, and S. J. Glaser, "Optimal control of coupled spin dynamics: design of NMR pulse sequences by gradient ascent algorithms", Journal of Magnetic Resonance **172**, 296–305 (2005).

[100] C. H. Yang, W. H. Lim, N. S. Lai, A. Rossi, A. Morello, and A. S. Dzurak, "Orbital and valley state spectra of a few-electron silicon quantum dot", Physical Review B **86**, 115319 (2012).

[101] K. Ono, D. G. Austing, Y. Tokura, and S. Tarucha, "Current Rectification by Pauli Exclusion in a Weakly Coupled Double Quantum Dot System", Science **297**, 1313–1317 (2002).

[102] J. R. Petta, A. C. Johnson, J. M. Taylor, E. A. Laird, A. Yacoby, M. D. Lukin, C. M. Marcus, M. P. Hanson, and A. C. Gossard, "Coherent Manipulation of Coupled Electron Spins in Semiconductor Quantum Dots", Science **309**, 2180–2184 (2005).

[103] A. C. Johnson, J. R. Petta, C. M. Marcus, M. P. Hanson, and A. C. Gossard, "Singlet-triplet spin blockade and charge sensing in a few-electron double quantum dot", Physical Review B **72**, 165308 (2005).

[104] A. C. Betz, R. Wacquez, M. Vinet, X. Jehl, A. L. Saraiva, M. Sanquer, A. J. Ferguson, and M. F. Gonzalez-Zalba, "Dispersively Detected Pauli Spin-Blockade in a Silicon Nanowire Field-Effect Transistor", Nano Letters **15**, 4622–4627 (2015).

[105] A. West et al., "Gate-based single-shot readout of spins in silicon", Nature Nanotechnology, 1 (2019).

[106] P. Pakkiam, A. V. Timofeev, M. G. House, M. R. Hogg, T. Kobayashi, M. Koch, S. Rogge, and M. Y. Simmons, "Single-Shot Single-Gate rf Spin Readout in Silicon", Physical Review X **8**, 041032 (2018).

[107] C. H. Yang, W. H. Lim, F. A. Zwanenburg, and A. S. Dzurak, "Dynamically controlled charge sensing of a few-electron silicon quantum dot", AIP Advances **1**, 042111 (2011).

[108] A. C. Johnson, J. R. Petta, J. M. Taylor, A. Yacoby, M. D. Lukin, C. M. Marcus, M. P. Hanson, and A. C. Gossard, "Triplet–singlet spin relaxation via nuclei in a double quantum dot", Nature **435**, 925–928 (2005).

[109] V. Srinivasa, H. Xu, and J. M. Taylor, "Tunable Spin-Qubit Coupling Mediated by a Multielectron Quantum Dot", Physical Review Letters **114**, 10.1103/PhysRevLett.114.226803 (2015).

[110] S. Mehl, H. Bluhm, and D. P. DiVincenzo, "Two-qubit couplings of singlet-triplet qubits mediated by one quantum state", Physical Review B **90**, 10.1103/PhysRevB.90.045404 (2014).

[111] P. Harvey-Collard et al., "Coherent coupling between a quantum dot and a donor in silicon", Nature Communications **8**, 1029 (2017).

[112] M. D. Reed et al., "Reduced Sensitivity to Charge Noise in Semiconductor Spin Qubits via Symmetric Operation", Physical Review Letters **116**, 110402 (2016).

[113] D. Loss and D. P. DiVincenzo, "Quantum computation with quantum dots", Physical Review A **57**, 120 (1998).

[114] T. Meunier, V. E. Calado, and L. M. K. Vandersypen, "Efficient controlled-phase gate for single-spin qubits in quantum dots", Physical Review B **83**, 10.1103/PhysRevB.83.121403 (2011).

[115] T. F. Watson et al., "A programmable two-qubit quantum processor in silicon", Nature **555**, 633–637 (2018).

[116] W. Huang et al., "Fidelity benchmarks for two-qubit gates in silicon", Nature **569**, 532 (2019).

[117] T. A. Baart, T. Fujita, C. Reichl, W. Wegscheider, and L. M. K. Vandersypen, "Coherent spin-exchange via a quantum mediator", Nature Nanotechnology **12**, 26–30 (2017).

[118] R. Ferdous, K. W. Chan, M. Veldhorst, J. C. C. Hwang, C. H. Yang, H. Sahasrabudhe, G. Klimeck, A. Morello, A. S. Dzurak, and R. Rahman, "Interface-induced spin-orbit interaction in silicon quantum dots and prospects for scalability", Physical Review B **97**, 241401 (2018).

[119] Y. Tokura, W. G. van der Wiel, T. Obata, and S. Tarucha, "Coherent Single Electron Spin Control in a Slanting Zeeman Field", Physical Review Letters **96**, 10.1103/PhysRevLett.96.047202 (2006).

[120] A. Corna et al., "Electrically driven electron spin resonance mediated by spin–valley–orbit coupling in a silicon quantum dot", npj Quantum Information **4**, 6 (2018).

[121] R. M. Jock et al., "A silicon metal-oxide-semiconductor electron spin-orbit qubit", Nature Communications **9**, 1768 (2018).

[122] T. Tanttu et al., "Controlling Spin-Orbit Interactions in Silicon Quantum Dots Using Magnetic Field Direction", Physical Review X **9**, 021028 (2019).

[123] J. M. Elzerman, R. Hanson, L. H. Willems van Beveren, B. Witkamp, L. M. K. Vandersypen, and L. P. Kouwenhoven, "Single-shot read-out of an individual electron spin in a quantum dot", Nature **430**, 431–435 (2004).

[124] P. Harvey-Collard et al., "High-Fidelity Single-Shot Readout for a Spin Qubit via an Enhanced Latching Mechanism", Physical Review X **8**, 021046 (2018).

[125] M. A. Fogarty et al., "Integrated silicon qubit platform with single-spin addressability, exchange control and single-shot singlet-triplet readout", Nature Communications **9**, 4370 (2018).

[126] C. J. Wood and J. M. Gambetta, "Quantification and characterization of leakage errors", Physical Review A **97**, 10.1103/PhysRevA.97.032306 (2018).

[127] J. Preskill, "Fault-tolerant quantum computation", in *Introduction to Quantum Computation and Information* (WORLD SCIENTIFIC, Oct. 1998), pp. 213–269.

[128] D. Gottesman, "Stabilizer Codes and Quantum Error Correction", PhD thesis ().

[129] P. Aliferis and B. M. Terhal, "Fault-tolerant Quantum Computation for Local Leakage Faults", Quantum Info. Comput. **7**, 139–156 (2007).

[130] A. G. Fowler, "Coping with qubit leakage in topological codes", Physical Review A **88**, 10.1103/PhysRevA.88.042308 (2013).

[131] M. Suchara, A. W. Cross, and J. M. Gambetta, "Leakage suppression in the toric code", 2015 IEEE International Symposium on Information Theory (ISIT), 1119–1123 (2015).

[132] S. D. Barrett and C. H. W. Barnes, "Double-occupation errors induced by orbital dephasing in exchange-interaction quantum gates", Physical Review B **66**, 125318 (2002).

[133] K. Wang, C. Payette, Y. Dovzhenko, P. W. Deelman, and J. R. Petta, "Charge Relaxation in a Single-Electron Si / SiGe Double Quantum Dot", Physical Review Letters **111**, 10.1103/PhysRevLett.111.046801 (2013).

[134] D. C. McKay, C. J. Wood, S. Sheldon, J. M. Chow, and J. M. Gambetta, "Efficient Z gates for quantum computing", Physical Review A **96**, 022330 (2017).

[135] G. Zheng, N. Samkharadze, M. L. Noordam, N. Kalhor, D. Brousse, A. Sammak, G. Scappucci, and L. M. K. Vandersypen, "Rapid gate-based spin read-out in silicon using an on-chip resonator", Nature Nanotechnology **14**, 742–746 (2019).

[136] R. Raussendorf, J. Harrington, and K. Goyal, "Topological fault-tolerance in cluster state quantum computation", New Journal of Physics **9**, 199 (2007).

[137] J. C. C. Hwang, C. H. Yang, M. Veldhorst, N. Hendrickx, M. A. Fogarty, W. Huang, F. E. Hudson, A. Morello, and A. S. Dzurak, "Impact of g-factors and valleys on spin qubits in a silicon double quantum dot", Physical Review B **96**, 045302 (2017).

[138] F. K. Malinowski et al., "Spin of a Multielectron Quantum Dot and Its Interaction with a Neighboring Electron", Physical Review X **8**, 011045 (2018).

[139] T. Giurgica-Tiron, Y. Hindy, R. LaRose, A. Mari, and W. J. Zeng, "Digital zero noise extrapolation for quantum error mitigation", arXiv:2005.10921 [quant-ph] (2020).

[140] Z. Cai, "Resource Estimation for Quantum Variational Simulations of the Hubbard Model", Physical Review Applied **14**, 014059 (2020).

[141] D. Wecker, M. B. Hastings, and M. Troyer, "Progress towards practical quantum variational algorithms", Physical Review A **92**, 042303 (2015).

[142] M. Kjaergaard, M. E. Schwartz, J. Braumüller, P. Krantz, J. I.-J. Wang, S. Gustavsson, and W. D. Oliver, "Superconducting Qubits: Current State of Play", Annual Review of Condensed Matter Physics **11**, 369–395 (2020).

[143] S. McArdle, S. Endo, A. Aspuru-Guzik, S. C. Benjamin, and X. Yuan, "Quantum computational chemistry", Reviews of Modern Physics **92**, 015003 (2020).

[144] Y. Cao et al., "Quantum Chemistry in the Age of Quantum Computing", Chemical Reviews **119**, 10856–10915 (2019).

[145] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets", Nature **549**, 242–246 (2017).

[146] J. Romero, R. Babbush, J. R. McClean, C. Hempel, P. J. Love, and A. Aspuru-Guzik, "Strategies for quantum computing molecular energies using the unitary coupled cluster ansatz", Quantum Science and Technology **4**, 014008 (2018).

[147] P.-L. Dallaire-Demers, J. Romero, L. Veis, S. Sim, and A. Aspuru-Guzik, "Low-depth circuit ansatz for preparing correlated fermionic states on a quantum computer", Quantum Science and Technology **4**, 045005 (2019).

[148] R. Babbush, N. Wiebe, J. McClean, J. McClain, H. Neven, and G. K.-L. Chan, "Low-Depth Quantum Simulation of Materials", Physical Review X **8**, 011044 (2018).

[149] I. D. Kivlichan, J. McClean, N. Wiebe, C. Gidney, A. Aspuru-Guzik, G. K.-L. Chan, and R. Babbush, "Quantum Simulation of Electronic Structure with Linear Depth and Connectivity", Physical Review Letters **120**, 110501 (2018).

[150] F. Verstraete and J. I. Cirac, "Mapping local Hamiltonians of fermions to local Hamiltonians of spins", Journal of Statistical Mechanics: Theory and Experiment **2005**, P09012–P09012 (2005).

[151] S. Bravyi and A. Kitaev, "Fermionic quantum computation", Annals of Physics **298**, 210–226 (2002).

[152] J.-M. Reiner, F. Wilhelm-Mauch, G. Schön, and M. Marthaler, "Finding the ground state of the Hubbard model by variational methods on a quantum computer with gate errors", Quantum Science and Technology **4**, 035005 (2019).

[153] C. Cade, L. Mineh, A. Montanaro, and S. Stanisic, "Strategies for solving the Fermi-Hubbard model on near-term quantum computers", arXiv:1912.06007 [quant-ph] (2019).

[154] Simons Collaboration on the Many-Electron Problem et al., "Solutions of the Two-Dimensional Hubbard Model: Benchmarks and Results from a Wide Range of Numerical Algorithms", Physical Review X **5**, 041041 (2015).

[155] D. Wecker, M. B. Hastings, N. Wiebe, B. K. Clark, C. Nayak, and M. Troyer, "Solving strongly correlated electron models on a quantum computer", Physical Review A **92**, 062318 (2015).

[156] E. Farhi, J. Goldstone, and S. Gutmann, "A Quantum Approximate Optimization Algorithm", arXiv:1411.4028 [quant-ph] (2014).

[157] E. Farhi and A. W. Harrow, "Quantum Supremacy through the Quantum Approximate Optimization Algorithm", arXiv:1602.07674 [quant-ph] (2016).

[158] F. Verstraete, J. I. Cirac, and J. I. Latorre, "Quantum circuits for strongly correlated quantum systems", Physical Review A **79**, 032316 (2009).

[159] Z. Jiang, K. J. Sung, K. Kechedzhi, V. N. Smelyanskiy, and S. Boixo, "Quantum Algorithms to Simulate Many-Body Physics of Correlated Fermions", Physical Review Applied **9**, 044036 (2018).

[160] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, "The theory of variational hybrid quantum-classical algorithms", New Journal of Physics **18**, 023023 (2016).

[161] P. J. J. O'Malley et al., "Scalable Quantum Simulation of Molecular Energies", Physical Review X **6**, 031007 (2016).

[162] W. J. Huggins, J. McClean, N. Rubin, Z. Jiang, N. Wiebe, K. B. Whaley, and R. Babbush, "Efficient and Noise Resilient Measurements for Quantum Chemistry on Near-Term Quantum Computers", arXiv:1907.13117 [physics, physics:quant-ph] (2019).

[163] O. Crawford, B. van Straaten, D. Wang, T. Parks, E. Campbell, and S. Brierley, "Efficient quantum measurement of Pauli operators", arXiv:1908.06942 [quant-ph] (2019).

[164] P. Gokhale and F. T. Chong, "$O(N\hat{3})$ Measurement Cost for Variational Quantum Eigensolver on Molecular Hamiltonians", arXiv:1908.11857 [quant-ph] (2019).

[165] P. Corboz, T. M. Rice, and M. Troyer, "Competing States in the t-J Model: Uniform d-Wave State versus Stripe State", Physical Review Letters **113**, 046402 (2014).

[166] G. G. Guerreschi and M. Smelyanskiy, "Practical optimization for hybrid quantum-classical algorithms", arXiv:1701.01450 [quant-ph] (2017).

[167] T. Kolda, R. Lewis, and V. Torczon, "Optimization by Direct Search: New Perspectives on Some Classical and Modern Methods", SIAM Review **45**, 385–482 (2003).

[168] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, "A variational eigenvalue solver on a photonic quantum processor", Nature Communications **5**, 4213 (2014).

[169] Y. Shen, X. Zhang, S. Zhang, J.-N. Zhang, M.-H. Yung, and K. Kim, "Quantum implementation of the unitary coupled cluster for simulating molecular electronic structure", Physical Review A **95**, 020501 (2017).

[170] R. Santagati et al., "Witnessing eigenstates for quantum simulation of Hamiltonian spectra", Science Advances **4**, eaap9646 (2018).

[171] J. I. Colless, V. V. Ramasesh, D. Dahlen, M. S. Blok, M. E. Kimchi-Schwartz, J. R. McClean, J. Carter, W. A. de Jong, and I. Siddiqi, "Computation of Molecular Spectra on a Quantum Processor with an Error-Resilient Algorithm", Physical Review X **8**, 011021 (2018).

[172] C. Hempel et al., "Quantum Chemistry Calculations on a Trapped-Ion Quantum Simulator", Physical Review X **8**, 031022 (2018).

[173] R. Sagastizabal et al., "Experimental error mitigation via symmetry verification in a variational quantum eigensolver", Physical Review A **100**, 010302 (2019).

[174] C. Kokail et al., "Self-verifying variational quantum simulation of lattice models", Nature **569**, 355 (2019).

[175] M. Ganzhorn et al., "Gate-Efficient Simulation of Molecular Eigenstates on a Quantum Computer", Physical Review Applied **11**, 044092 (2019).

[176] I. Goodfellow, Y. Bengio, A. Courville, and F. Bach, *Deep Learning* (MIT Press, Cambridge, Massachusetts, Jan. 2017).

[177] S. Ruder, "An overview of gradient descent optimization algorithms", arXiv:1609.04747 [cs] (2017).

[178] G. Verdon, M. Broughton, J. R. McClean, K. J. Sung, R. Babbush, Z. Jiang, H. Neven, and M. Mohseni, "Learning to learn with quantum neural networks via classical neural networks", arXiv:1907.05415 [quant-ph] (2019).

[179] M. Wilson, S. Stromswold, F. Wudarski, S. Hadfield, N. M. Tubman, and E. Rieffel, "Optimizing quantum heuristics with meta-learning", arXiv:1908.03185 [quant-ph] (2019).

[180] K. M. Nakanishi, K. Fujii, and S. Todo, "Sequential minimal optimization for quantum-classical hybrid algorithms", arXiv:1903.12166 [physics, physics:quant-ph] (2019).

[181] K. C. Nowack, M. Shafiei, M. Laforest, G. E. D. K. Prawiroatmodjo, L. R. Schreiber, C. Reichl, W. Wegscheider, and L. M. K. Vandersypen, "Single-Shot Correlations and Two-Qubit Gate of Solid-State Spins", Science **333**, 1269–1272 (2011).

[182] S. Schaal et al., "Fast Gate-Based Readout of Silicon Quantum Dots Using Josephson Parametric Amplification", Physical Review Letters **124**, 067701 (2020).

[183] V. Srinivasa, K. C. Nowack, M. Shafiei, L. M. K. Vandersypen, and J. M. Taylor, "Simultaneous Spin-Charge Relaxation in Double Quantum Dots", Physical Review Letters **110**, 196803 (2013).

[184] B. Bertrand, H. Flentje, S. Takada, M. Yamamoto, S. Tarucha, A. Ludwig, A. D. Wieck, C. Bäuerle, and T. Meunier, "Quantum Manipulation of Two-Electron Spin States in Isolated Double Quantum Dots", Physical Review Letters **115**, 096801 (2015).

[185] C. H. Yang et al., "Operation of a silicon quantum processor unit cell above one kelvin", Nature **580**, 350–354 (2020).

[186] A. Morello et al., "Single-shot readout of an electron spin in silicon", Nature **467**, 687–691 (2010).

[187] M. D. Shulman, S. P. Harvey, J. M. Nichol, S. D. Bartlett, A. C. Doherty, V. Umansky, and A. Yacoby, "Suppressing qubit dephasing using real-time Hamiltonian estimation", Nature Communications **5**, 5156 (2014).

[188] D. C. McKay, S. Filipp, A. Mezzacapo, E. Magesan, J. M. Chow, and J. M. Gambetta, "Universal Gate for Fixed-Frequency Qubits via a Tunable Bus", Physical Review Applied **6**, 064007 (2016).

[189] N. Schuch and J. Siewert, "Natural two-qubit gate for quantum computation using the $\mathrm{XY$ interaction", Physical Review A **67**, 032301 (2003).

[190] A. Chiesa, F. Tacchino, M. Grossi, P. Santini, I. Tavernelli, D. Gerace, and S. Carretta, "Quantum hardware simulating four-dimensional inelastic neutron scattering", Nature Physics **15**, 455–459 (2019).

[191] G. Wendin, "Quantum information processing with superconducting circuits: a review", Reports on Progress in Physics **80**, 106001 (2017).

[192] M. D. Reed, B. R. Johnson, A. A. Houck, L. DiCarlo, J. M. Chow, D. I. Schuster, L. Frunzio, and R. J. Schoelkopf, "Fast reset and suppressing spontaneous emission of a superconducting qubit", Applied Physics Letters **96**, 203110 (2010).

[193] T. Walter et al., "Rapid High-Fidelity Single-Shot Dispersive Readout of Superconducting Qubits", Physical Review Applied **7**, 054020 (2017).

[194] J. R. McClean, M. E. Kimchi-Schwartz, J. Carter, and W. A. de Jong, "Hybrid quantum-classical hierarchy for mitigation of decoherence and determination of excited states", Physical Review A **95**, 042308 (2017).

[195] Z. Jiang, J. McClean, R. Babbush, and H. Neven, "Majorana Loop Stabilizer Codes for Error Mitigation in Fermionic Quantum Simulations", Physical Review Applied **12**, 064041 (2019).

[196] Z. Cai, "Multi-exponential Error Extrapolation and Combining Error Mitigation Techniques for NISQ Applications", arXiv:2007.01265 [quant-ph] (2020).

[197] T. Jones and S. C. Benjamin, "QuESTlink – Mathematica embiggened by a hardware-optimised quantum emulator", Quantum Science and Technology, 10.1088/2058-9565/ab8506 (2020).

[198] T. Jones, A. Brown, I. Bush, and S. C. Benjamin, "QuEST and High Performance Simulation of Quantum Computers", Scientific Reports **9**, 1–11 (2019).

[199] A. W. Cross, E. Magesan, L. S. Bishop, J. A. Smolin, and J. M. Gambetta, "Scalable randomised benchmarking of non-Clifford gates", npj Quantum Information **2**, 1–5 (2016).

[200] J. Helsen, X. Xue, L. M. K. Vandersypen, and S. Wehner, "A new class of efficient randomized benchmarking protocols", npj Quantum Information **5**, 1–9 (2019).

[201] S. T. Flammia and J. J. Wallman, "Efficient estimation of Pauli channels", arXiv:1907.12976 [quant-ph] (2019).

[202] C. Granade, C. Ferrie, I. Hincks, S. Casagrande, T. Alexander, J. Gross, M. Kononenko, and Y. Sanders, "QInfer: Statistical inference software for quantum applications", Quantum **1**, 5 (2017).

[203] S. Kimmel, M. P. da Silva, C. A. Ryan, B. R. Johnson, and T. Ohki, "Robust Extraction of Tomographic Information via Randomized Benchmarking", Physical Review X **4**, 011050 (2014).

[204] J. Helsen, F. Battistel, and B. M. Terhal, "Spectral quantum tomography", npj Quantum Information **5**, 1–11 (2019).

[205] *https://github.com/czydbb/SurfaceCodeModule*.

[206] S. Bravyi, M. Suchara, and A. Vargo, "Efficient algorithms for maximum likelihood decoding in the surface code", Physical Review A **90**, 10.1103/PhysRevA.90.032326 (2014).

[207] K. Mitarai and K. Fujii, "Methodology for replacing indirect measurements with direct measurements", Physical Review Research **1**, 013006 (2019).