Practical quantum computing: error reduction techniques from software to hardware

Adam Siegel

Exeter College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Michaelmas 2025

Acknowledgements

During the three years of my PhD, I had the chance to meet and interact with many people who helped me through my journey. I would like to spend a few words here to properly thank them all.

First of all, the person who obviously deserves to be top of this list and that I cannot thank enough is my supervisor *Simon Benjamin*. I'm honestly not sure he has ever received anything but praises in such PhD acknowledgements... Simon is both an incredibly clever scientist, but also an extremely socially-aware and empathetic person. I always admired how, even when only remotely involved in some of my projects, he would always find the loophole, or the little trick that would magically improve the results. On top of this, I cannot thank him enough for enabling me to situate myself with a collaborating group in Japan and Australia when I very much needed to take some time away from Oxford.

I would then like to thank all my supervisors and collaborators around the world, who agreed to host me in their research groups for a few months: *Prof. Keisuke Fujii* and *Prof. Kosuke Mitarai* at Osaka University; *Michael Fogarty* and *Simon Devitt* in Sydney; and *Nobuyuki Yoshioka* at Tokyo University. I feel extremely grateful for the time you invested in our collaborations, and hope we get a chance to collaborate again in the future.

Aside from them, I also had the chance to collaborate with various people in the almighty UK, either from the QTechTheory Group or from Quantum Motion: *Armands Strikis* who accompanied me in all of my QEC projects and who I learnt a lot from; *Zhenyu Cai*, *Hans Chan* and *Balint Koczor* who I exchanged frequently with on various projects; and *Fernando Gonzalez* and *Sofia Patomaki* when I ventured into more experimental grounds.

One other collaborator, but above all an 意外な友達 , deserves separate thanks however: *Hamza Jnane*. I guess he taught me that first impressions were not always accurate... and he quickly became one of my closest friends in Oxford. Thank you for bearing with me when I was judging things a bit too fast or when I was sending you too many travel pictures when you were just getting rained on. Cheers to many more adventures and hopefully less time-consuming projects as QMT's Senior Quantum Theorists. I hope that we'll make it to Naoshima together one day.

My thanks are then naturally directed at all my other friends, outside my quantum computing research. *Sasha Tinelli*, I'm not sure how I would have

survived without your constant and genuine love, that you sent straight to my stomach in the form of homemade pasta or mozzarella. *Lara Brudermüller*, *Peter Doohan* and *Tristram Walsh*, I am beyond grateful we got to share this beautiful house of ours together (even if it got taken away from us), sorry I got the smoke alarm to ring way too often.

Then come all my other friends, who supported me during these years of research: *Anton Rael*, *Elie Bermot*, *Misha Robert*, *Laura Kolcheva*, *Lyranne Rubinstein*, *Beth Wilkins*: you've all been wonderful friends and I'm so proud of what you've become and how you evolved in these past years. *Arsh Sahid*, *Tom Ho*, *Alex Wells* and *Ned Summers*: you are part of what made my experience abroad so rich and were invaluable supports when I was so far away from home.

Finally, I would like to properly thank my parents for supporting me in all of my decisions (except when it involved climbing the Mont Blanc). I know I do not always show the appreciation you would want me to, but I would not be here without you. You were a motor of my ambitions, you taught me to be free and only pursue the things that actually drive me. Even when I'm away, even if we don't chat every day, I am still grateful for everything you did and do for me.

Abstract

This thesis presents multiple advances in the construction of noise-resistant quantum computers. In order to bridge the gap between abstract theory and practical experiments, my research encompasses a wide range of subfields within quantum computing, including error correction, silicon spin qubit physics and error mitigation. Firstly, I demonstrate that large-scale defects affecting entire regions of an error correcting code, such as cosmic rays or manufacturing defects, do not necessarily constitute a fundamental obstacle. Conversely, bespoke protocols that employ code deformation can be utilised to mitigate their impact, at a moderate resource overhead. Secondly, I investigate the feasibility of building a fault-tolerant silicon spin qubit quantum computer in the near term. I demonstrate that arranging the qubits in a 2×N array represents an effective methodology for the construction of error correcting codes, provided that the qubits can be shuttled with high fidelity. This expected characteristic of silicon spins can thus be utilised to circumvent the construction of complex dense 2D grids of qubits. Subsequently, I build upon the existing literature on the implementation of low-noise gates in silicon spin qubits, at the physical level this time. Specifically, I design a protocol for the individual control of silicon spin qubits, without the need for local magnetic fields targeting every qubit in the device. This simplification enables efficient control of silicon spins, whose single-qubit addressability was an identified bottleneck. Finally, I complement the above research with an error mitigation scheme targeted at the evaluation of eigenvalues with an early-fault-tolerant quantum computers. This scheme promises to alleviate the stringent resource costs of error correction, thereby paving the way for fault-tolerance in a nearer future.

Contents

1 Introduction

In 1965, Gordon Moore posited that the power of our computers would be doubling every year, eventually making them able to solve any kind of problem. But this utopia is today halting, or at least declining, as our classical supercomputers start to face problems of unequalled complexity. If, for instance, modern chemistry is perfectly described by quantum mechanics, it is not conceivable to simulate a large molecule with a classical computer as the complexity of the wave function increases exponentially with the size of the system. From 1964, Richard Feynman was stating: "It always bothers me that, according to the laws as we understand them today, it takes a computing machine an infinite number of logical operations to figure out what goes on in no matter how tiny a region of space, and no matter how tiny a region of time." [1] In 1982, he proposed a new simulation platform to tackle this question: exploiting quantum mechanics itself to simulate quantum mechanics [2].

Since then, researchers have explored this new paradigm and showed that quantum computers could indeed run algorithms with a significant speed-up compared to their classical counterpart. For instance, Deutsch established in 1985 that a single evaluation of a function could show if it is constant or not [3]. Some years later, Shor [4] and Grover [5] suggested two quantum algorithms to solve problems that cannot be treated easily with a classical computer: finding the prime factors of an integer and searching an entry in a database.

However, even though Google claimed in 2019 they had reached *quantum supremacy* [6], the quantum devices built today are imperfect and cannot be used yet to implement aforementioned algorithms. Indeed, this would require a very high level of precision in the controls of the quantum bits, or so-called *qubits*, while guaranteeing minimal interaction with their environment, as this could lead to a loss of their quantum coherence thus ruining their quantum properties. Besides, the error rates one should be aiming for to run one of those algorithms should be at most 10^{-15} [7–9] which is far too low for a direct implementation on physical qubits, whose error rates are typically around 10^{-3} on a high-fidelity device [10–14].

In this document, I will explore solutions facilitating the implementation of such demanding algorithms on inherently unstable devices, and look at this task from different angles.

The first three chapters aim at reviewing important concepts used throughout the subsequent research chapters. First, the later parts of this chapter lays the foundation by providing a general introduction to quantum computing. It covers the basic concepts of qubits, gates and measurements, and discusses how these fundamental principles enable quantum computers to potentially outperform classical systems in solving specific types of problems.

Chapter 2 delves into one specific implementation of quantum computing using silicon spin qubits. Silicon, a material well-established in the classical semiconductor industry, offers promising prospects for scaling up quantum devices with already-existing manufacturing techniques. This review chapter aims at giving a general understanding of such qubit platform to the reader for subsequent research chapters 5 and 6.

Chapter 3 introduces the critical concept of quantum error correction (QEC), which promises to bridge the gap between achievable physical error rates (around 10^{-3}) and required logical error rates (around 10^{-15}). This third review chapter sets up the theoretical frameworks of quantum error correction, with a particular emphasis on protocols used in research chapters 4 and 5.

In particular, Chapter 4 extends on the usual noise models considered in QEC simulations, and tackles the correction of large-scale errors. Recent experiments have shown that many qubit platforms such as superconductors are sensitive to cosmic rays disrupting large areas of a code. Alternatively, some code regions may be defective right from the manufacturing stage. In this chapter, I show that tailored protocols allow for the suppression of such large-scale errors, and only at a moderate resource cost.

While Chapter 4 guarantees a more sustainable use of error correction in the longer run, in Chapter 5 I take a step back and show the practicality of QEC in a nearer term device, associated with yet another noise process. More specifically, one of the challenges of building error correcting codes today lies in the implementation of dense 2D grids of qubits. In this chapter, I show that a 2×N array of qubits equipped with shuttling can indeed be used to implement QEC despite the apparent constraints of this setup. Focusing on silicon spin qubits as a practical example of qubits believed to be suitable to such architectures, I provide a protocol for achieving full universal quantum computation with the surface code, while also addressing the additional constraints that are specific to a silicon spin qubit device.

The protocols of Chapter 5 are completed in Chapter 6, which tackles the question of single-qubit addressability in silicon spin qubit devices. This question arises as silicon spin qubits encoded with single electrons are manipulated via the use of magnetic fields, which can be difficult to localise at the qubit scale. While in Chapter 5 I show that some kind of global control of the electrons is sufficient for QEC protocols, these could be simplified if qubits could be locally addressed. Chapter 6 thus gives three different methods aiming at enhancing such individual control.

Finally, Chapter 7 completes the strategies presented in the previous chapters with a complementary approach: quantum error mitigation (QEM). Quantum error correction is a strong and sustainable solution to reduce errors. It however requires large resource overheads, which may not be available in the early-stage devices for instance depicted in Chapter 5. For this reason, it may be advantageous to couple the use of QEC with QEM, which relies on the classical post-processing

of the algorithm output. In this chapter, I focus on a particular example of QEM protocol for (partially) fault-tolerant devices, targeted at the estimation of the eigenvalues of an operator.

Finally, Chapter 8 concludes this thesis by summarising the key findings from each chapter and offering perspectives on the future of quantum computing.

1.1 Mathematical background for quantum computing

In the rest of this chapter, I give an overview of the mathematical background used in quantum computing.

1.1.1 Kets

While a classical bit value can only be 0 or 1, a quantum bit, noted $|\psi\rangle$, can be in any superposition of $|0\rangle$ and $|1\rangle$, the basis states of a Hilbert space of dimension 2:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \qquad \text{with} \quad |\alpha|^2 + |\beta|^2 = 1$$

Generalising it to n qubits, the state of the system lives in the tensor product of the Hilbert spaces of the individual qubits, hence reads:

$$|\psi\rangle = \sum_{(i_1,\ldots,i_n)\in\{0,1\}^n} \alpha_{i_1,\ldots,i_n}\,|i_1 \ldots i_n\rangle \qquad \text{with} \sum_{(i_1,\ldots,i_n)\in\{0,1\}^n} |\alpha_{i_1,\ldots,i_n}|^2 = 1$$

The system is thus in a superposition of 2^n states of the form $|i_1 \ldots i_n\rangle := |i_1\rangle \otimes \ldots \otimes |i_n\rangle$, hence described by $2^n - 1$ independent complex parameters. For $n > 10^3$, this quantity largely exceeds the number of atoms in the universe. This is an indication of the power of quantum computers compared their classical counterpart.

1.1.2 Gates

Qubits are operated on via quantum gates, which are unitary operations on the Hilbert space. Three categories of gates are of particular importance: Pauli gates, Clifford gates and non-Clifford gates.

First, single-qubit Pauli gates correspond to the following operations:

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

They respectively:

- trigger a bit flip: $X|0\rangle = |1\rangle$ and $X|1\rangle = |0\rangle$
- trigger a bit and phase flip: $Y(\alpha|0\rangle + \beta|1\rangle) = -i(\alpha|1\rangle - \beta|0\rangle)$ (the overall factor $-i$ does not matter)
- trigger a phase flip: $Z(\alpha|0\rangle + \beta|1\rangle) = \alpha|0\rangle - \beta|1\rangle$

n-qubit Pauli gates, obtained by taking the tensor products of single-qubit Paulis and identity matrices, form the n-qubit Pauli group $\mathcal{P}_n$. This group constitutes a basis of the n-qubit unitary operations $\mathcal{U}_n$.

Pauli gates are a part of a larger important class of gates called Clifford gates. They are defined as:

$$\mathcal{C}_n = \{V \in \mathcal{U}_n \mid V\mathcal{P}_n V^\dagger = \mathcal{P}_n\} \tag{1.1}$$

This group is generated by three gates only: the Hadamard H, the S gate and the CNOT gate, such that:

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}, \quad CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

The CNOT gate flips the second (target) qubit if and only if the first (control) qubit is in the $|1\rangle$ state. The evolution of a state under Clifford gates only has been shown to be efficiently simulable on a classical computer, in virtue of the Gottesman-Knill theorem [15]. Indeed, by definition, Clifford gates map Pauli operators onto different Pauli operators and can thus easily be tracked this way.

In order to reach full quantum power, it thus appears necessary to make use of non-Clifford gates. Bravyi and Kitaev proved that the addition of the T gate

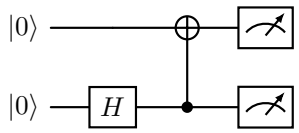


Figure 1.1: Example of a circuit preparing the 2-qubit state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle$ before measuring it. From left to right, the operations are a Hadamard gate, a CNOT targetting the first qubit and two measurements in the computational basis.

was enough to form a universal set of quantum gates [16], *i.e.* gates that can be used to decompose any unitary operation. The T gate reads:

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

Although the decomposition could be infinite, Solovay–Kitaev theorem proved that any n-qubit unitary operation (with n fixed) could actually be efficiently approximated by a finite sequence of gates from a universal set [17].

1.1.3 Measurements

Other than gates, measurements can be applied to qubits, either at the end or mid-computation. Following the principles of quantum mechanics (Born rule), measuring a quantum state $|\psi\rangle$ with some (Hermitian) operator $\hat{O}$ collapses the state onto an eigenstate $|\phi\rangle$ of the operator, with a probability given by $|\langle\phi|\psi\rangle|^2$. For instance: measuring Z collapses $\alpha|0\rangle + \beta|1\rangle$ onto $|0\rangle$ with probability $|\alpha|^2$ and onto $|1\rangle$ with probability $|\beta|^2$.

An important rule to have in mind (as it will be extensively used in the next chapters) is that only commuting operators can be simultaneously measured. More precisely, any set of commuting operators has a common eigenbasis, onto which a quantum state would be projected if all operators are measured. Conversely, if two non-commuting operators are successively measured, the outcome of the second one is in general random.

With all the above ingredients, quantum circuits can be formed by successively applying quantum gates and measurements. They are schematically represented as in Fig. 1.1.

2
Silicon spin qubits

Contents

Plan

2.1 Introduction

In Chapter 1, I presented a very general framework for quantum computing, irrespective of which physical platform is used to encode the qubits. Today, multiple physical systems are being considered for implementing a quantum computer, all of them responding to five criteria established in 2000 by Di Vincenzo [18]. In particular, the first criterion stipulates the scalability of the chosen physical system: as quantum computers develop to eventually run algorithms beyond the

classical regime, millions of qubits will be required, so as to build the resilient error correcting codes presented in Chapter 3. It is therefore paramount to search for systems offering opportunities for high-density qubit grids, and silicon spins are one of them. This is for two reasons: first their very small qubit and classical electronics footprints, around $100 \times 100\text{nm}^2$ per qubit, allows for such high densities; second they are compatible with already-established large-scale integration CMOS manufacturing techniques [19].

Many types of encoding exist for silicon spin qubits, however in this thesis I will focus — unless specified otherwise — on the case of a qubit represented by a single spin trapped in a quantum dot. This is the most natural and compact encoding as spins are inherently two-level systems, but they require the use of magnetic fields for their control (see Section 2.2). As these can be difficult to localise at the scale of a quantum dot and represent a leading source of dephasing [20], more complex qubit types have been suggested, based on compounds of multiple spins *e.g.* singlet-triplet [21, 22] or exchange-only qubits [23, 24]. These offer an advantageous electrical control of the qubits but come at the cost of an increased resource overhead. Alternatively to quantum-dot spin qubits, donor atoms such as ^{31}P in bulk Si may be used to form qubits [25]; I will not delve in further details into this implementation.

2.2 Physics of a quantum-dot silicon spin qubit

2.2.1 Defining a single-spin qubit

As explained above, I will focus on the case of a qubit encoded by a single electron trapped in a quantum dot, *i.e.* a 0D structure. However, electrons in bulk silicon naturally form a 3D electron gas. Several steps are thus followed to successively reduce the dimensionality. First, heterostructures are utilised to form a 2D electron gas, thanks to the conduction band difference between the chosen materials. More specifically, in the case of silicon, mainly two structures are being engineered today: Si metal-oxide-semiconductor (MOS) and Si/SiGe heterostructures. In the former, the large conduction band offsets between the silicon and the oxide layers allow for a significant electric field to be applied without electrons tunneling out of the

silicon. This effectively creates a triangular potential in which electrons are trapped. Alternatively, a thin silicon layer may be sandwiched between two layers of an SiGe alloy: the conduction band difference traps the electrons in the Si layer [26]. Once the electron has been confined in the out-of-plane (growth) direction, gate voltages are applied to trap it in the in-plane directions. This ultimately forms a quantum dot.

Electrons can thereon be loaded into the dots by tunneling between a source and a drain. The number of electrons in a dot is controlled the voltage applied to it. When the ground state energy of the dot is lower than the Fermi level of the source, one first electron can be loaded in the dot. Coulomb repulsion between electrons forbids a second electron to tunnel in, unless the dot potential is further lowered. When considering a double quantum dot (DQD) (Fig. 2.1 (a)), these phenomena can be experimentally observed in a stability diagram (Fig. 2.1 (b)). Each red line corresponds to the measurement of a current, *i.e.* the movement of an electron between the two dots, or between a dot and the source or the drain. The regions between the red lines thus correspond to stable configurations, each associated with a given electron distribution (N_1, N_2), where N_i $(i = 1, 2)$ is the number of electrons in dot i (Fig. 2.1 (c) and (d)). For the chosen qubit encoding (one qubit represented by one trapped spin), $N_i \leq 1$ should intuitively always be satisfied, but considering double-occupancy as well is necessary to describe phenomena such as exchange coupling. Note that in the case of single occupancy, the orbital part of the electron wavefunction should always be in the ground state of the quantum dots (which is not necessarily true in case of double occupancy, due to Pauli exclusion principle).

In order to use the spin part of the wavefunctions as qubits, one must lift their degeneracy. This is done via the application of a global magnetic field B_0 which splits the energy levels of the $|\uparrow\rangle$ and $|\downarrow\rangle$ states in virtue of the Zeeman effect:

$$\omega_q = g\mu_B B_0 \tag{2.1}$$

where ω_q is the qubit frequency, μ_B is the Bohr magneton and g is the electron's g-factor ($g \approx 2$). Note that we here set $\hbar = 1$.

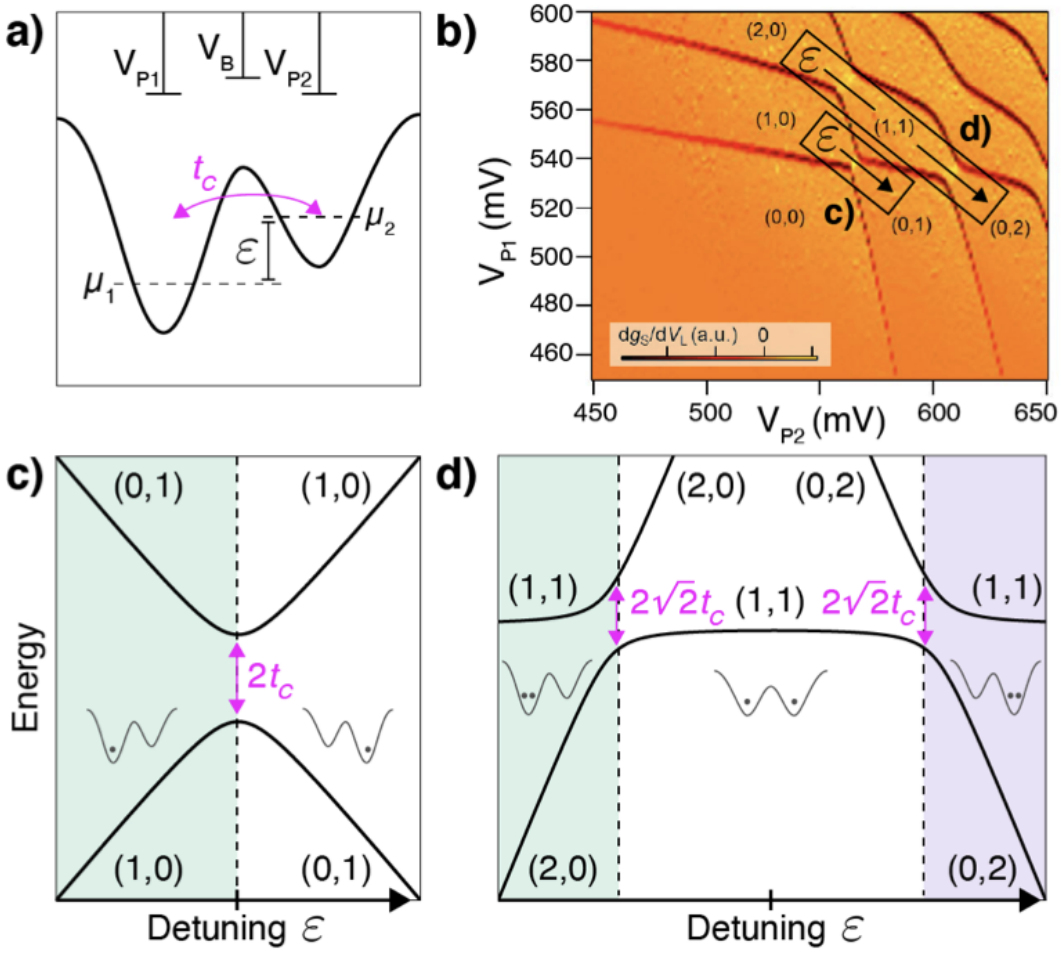


Figure 2.1: (a) Schematic representation of a double quantum dot (DQD), where the potential of each dot is controlled by the voltages V_{P1} and V_{P2}. $\varepsilon = V_{P2} - V_{P1}$ is the detuning between the dots. (b) Stability diagram of the DQD. Red lines coincide with the measurement of a current while orange regions between these lines correspond to stable configurations with given electron occupancies (N_1, N_2). (c-d) Energy levels as a function of ε in the single- and double-occupancy regimes. This figure was taken from [20].

2.2.2 Initialisation and measurement

So far, I have described how a quantum dot, *i.e.* a 0D structure, can be created, and how single electrons can be loaded in it. In this section, I will further explain how the $|\uparrow\rangle$ and $|\downarrow\rangle$ states can selectively be initialised and readout, which are two of Di Vincenzo's criteria for universal quantum computation [18].

The initialisation procedure relies on already-described ingredients: tunnelling from a reservoir and Zeeman splitting. Due to the latter, $|\downarrow\rangle$ is higher in energy than $|\uparrow\rangle$. By applying a negative voltage to the dot so as to place the Fermi energy

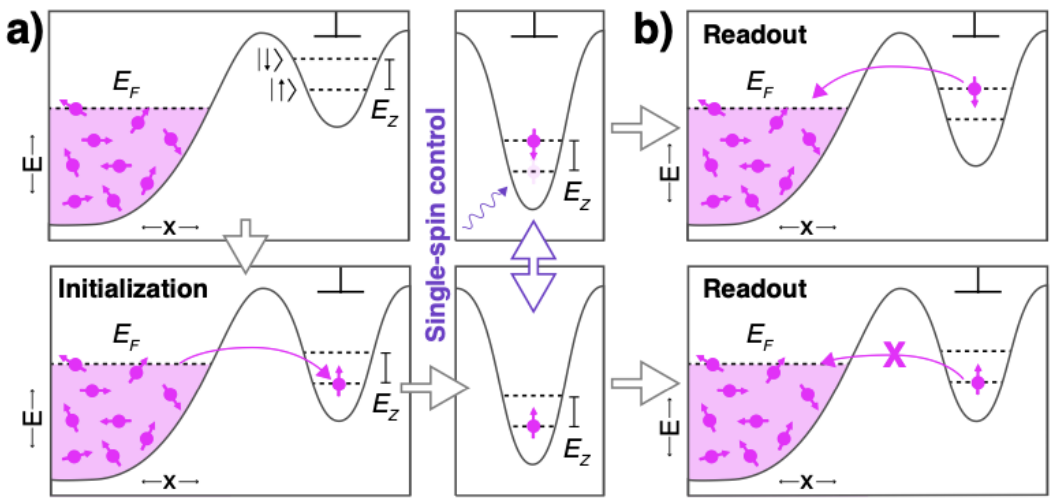


Figure 2.2: Initialisation and readout protocols for the single-electron encoding of a qubit. This figure was taken from [20].

of the reservoir between $E_\downarrow$ and $E_\uparrow$ (the respective energies of the qubit state), an electron can only tunnel to the ground state $|\uparrow\rangle$ (Fig. 2.2). Conversely, at the readout stage, an electron can only tunnel back to the reservoir if it is in the $|\downarrow\rangle$ state. Note this is only one example of measurement type: there exists others using *e.g.* Pauli Spin Blockade and single-triplet pairs.

2.2.3 Single-qubit gates

Between initialisation and measurement, qubits are operated on via the application of quantum gates, which can be divided into two groups: single- and two- qubit gates. At the physical level, one example of the implementation of the former relies on two mechanisms: (i) the application of a transverse AC magnetic field $\mathbf{B_1}$ and (ii) the electrical control of the local qubit g-factor via Stark shift. Without loss of generality, we can assume that $\mathbf{B_1}$ is parallel to the x-axis: it can therefore be further decomposed as $\mathbf{B_1} = (\Omega\cos(\omega t + \phi), 0, 0))$, with Ω the amplitude of the field, ω its oscillation frequency and ϕ its phase relative to a local oscillator. Let us additionally call ω_q^0 (resp. ω_q^1) the qubit original (resp. stark-shifted) frequencies. With these notations and before applying any Stark shift, the single-qubit Hamiltonian in

the laboratory frame of reference reads:

$$H = \frac{\omega_q^0}{2}\sigma_z + \Omega\cos(\omega t + \phi)\sigma_x \tag{2.2}$$

with σ_z and σ_x the Pauli operators.

By performing a frame change into the frame rotating at ω around the z-axis, one gets the following Hamiltonian [20]:

$$H = \frac{\Omega}{2}(\sigma_x\cos\phi + \sigma_y\sin\phi) + \frac{\omega_q^0 - \omega}{2}\sigma_z \tag{2.3}$$

This expression is obtained by making use of the rotating frame approximation, neglecting terms oscillating at 2ω as in practice $\Omega \ll \omega$.

We can then identify two useful operating regimes. First, if the qubit is at resonance with the drive ($\omega = \omega_q^0$), it will undergo a coherent rotation at frequency Ω around an axis in the xy-plane specified by the angle ϕ. Second, if the drive is turned off ($\Omega = 0$), Stark shift is applied ($\omega_q^0 \leftarrow \omega_q^1$) and the Hamiltonian is written in the frame rotating at the natural qubit frequency ($\omega = \omega_q^0$), one obtains a coherent rotation around the z-axis at frequency $\omega_q^1 - \omega_q^0$.

In addition, Z gates may be applied at strictly zero cost by simply updating the frame of reference and subsequently offsetting the classical phase ϕ in software. This technique is called *virtual Z gates* [**McKay_2017_efficient**]. Therefore, Z rotations are often discarded when evaluating the fidelity of a more complex gate (see Chapter 6).

Beyond the perfect cases described above, it will be relevant to further study the behaviour of a qubit under an off-resonant drive. For simplicity, let us assume that $\phi = 0$ and that the qubit initial state is $|0\rangle$. Let us additionally denote $\Delta = \omega_q^0 - \omega$. The Hamiltonian thus reads:

$$H = \frac{\Omega}{2}\sigma_x + \frac{\Delta}{2}\sigma_z \tag{2.4}$$

Simply solving Schrodinger equation leads to the Rabi model. The probability of being in the excited state evolves as:

$$P_e(t) = \frac{\Omega}{\sqrt{\Omega^2 + \Delta^2}}\sin^2\left(\frac{\sqrt{\Omega^2 + \Delta^2}}{2}t\right) \tag{2.5}$$

Therefore, the amplitude of the oscillations reduces as Δ increases and the drive becomes off-resonant. Note that the qubit does undergo Z-rotations as Δ increases though, but these can be absorbed into frame shifts and resulting virtual Z gates.

This well-known behaviour allows for the collective control of multiple qubits via a global drive. In a silicon spin qubit device, the qubits do not typically have the same g-factor due to the interface roughness [27], which allows for their selective control. By sending tones at resonance with target qubits only, these can be almost-perfectly driven while non-targets undergo marginal rotations due to off-resonant effects. The performance of such a technique depends on the frequency spacing between the qubits which intuitively decreases as the device is scaled – this issue is called *frequency crowding*. While Stark shift can be utilised to mitigate it [28], Chapter 6 will offer further insight and improvements into this.

2.2.4 Two-qubit gates

Beyond single-qubit gates, quantum computers must be able to perform two-qubit gates in order to form a universal set of gates. In silicon spin devices, these are enabled by the exchange coupling J. For a two-qubit system, assuming a distinct on-site g-factor and no transverse magnetic field, the Hamiltonian reads:

$$H = \frac{J}{4}(\sigma_x^1 \otimes \sigma_x^2 + \sigma_y^1 \otimes \sigma_y^2 + \sigma_z^1 \otimes \sigma_z^2) + \frac{\omega_q^1}{2}\sigma_z^1 + \frac{\omega_q^2}{2}\sigma_z^2 \tag{2.6}$$

The first term characterises the exchange interaction between the spins. It arises when the orbital wavefunctions of qubits in neighbouring dots overlap, resulting in their coupling. A microscopic study of the system, accounting for the tunnel coupling between the dots, their detuning, the onsite Coulomb repulsion and Pauli exclusion principle formally demonstrates the form of this term [20]. Besides, the exchange coupling is electrically adjustable and exponentially depends on the interdot barrier height, which makes it a natural choice for controlling two-qubit interactions.

When $\omega_q^1 = \omega_q^2$, the last two terms of Eq. 2.6 can be absorbed into a rotating frame, leaving the exchange term only. Alone, this term implements a SWAP gate

in a time $2\pi/J$ and a $\sqrt{\text{SWAP}}$ in a time π/J. The latter is an entangling gate and can be used to form a universal set of gates.

In practice however, due to g-factor disorder, the last two terms of Eq. 2.6 cannot be canceled simultaneously. When J is kept sufficiently below the qubits g-factors difference, the flip-flop term typically vanishes but phase still accumulates for two-qubit spin states with different orientations. This leads to the following Hamiltonian in the $(|\uparrow\uparrow\rangle, |\uparrow\downarrow\rangle, |\downarrow\uparrow\rangle, |\downarrow\downarrow\rangle)$ basis [29]:

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & e^{iJt/2} & 0 & 0 \\ 0 & 0 & e^{iJt/2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.7}$$

Modulo the addition of some single-qubit Z gates, this is can be used as a C-PHASE gate, which can as well be utilised to form a universal set of gates.

One last interesting two-qubit gate that can naturally be performed on spin qubits is the CROT (a controlled rotation around an axis in the xy plane). In the absence of J-coupling, the $|\uparrow\downarrow\rangle$ and $|\downarrow\uparrow\rangle$ levels are degenerate. However, the application of a finite J-coupling lifts this degeneracy, enabling the selective targeting of the $|\uparrow\uparrow\rangle$-$|\uparrow\downarrow\rangle$ transition. By applying a driving pulse at resonance with this energy splitting, a CROT gate (controlled by the first qubit) is implemented [30–33].

2.2.5 Shuttling

On top of the operations described above, silicon spin qubits have been demonstrated to exhibit excellent shuttling capabilities. While previously described quantum gates exclusively act on the spin state, shuttling rather displaces the qubit without altering their spin state. This operation can thus be used to change the physical configuration of the spins in the device and generate new pairs of nearest neighbours for subsequent two-qubit gates. In silicon spin qubits device, this kind of information transfer has been achieved using techniques in the form of bucket brigade [34] or conveyor belt approaches [35].

The first method makes use of tunneling to shuttle electrons, by successively lowering the potential of subsequent quantum dots along their way to generate

their movement. On the contrary, the second method aims at always keeping the wavefunction at a minimum of a smoothly moving sine wave, without tunneling out of it (see Fig. 2 of [36]). In this thesis, I will exclusively focus on the conveyor-belt mode of shuttling. Indeed, [36] demonstrated that the bucket brigade scheme may be rather difficult to scale up due to the inherent disorder in silicon spin devices. This makes rather challenging the fine-tuning of the detunings and tunnel couplings, which ultimately control the tunneling events leading to shuttling. Besides, the protocols I present in this thesis all rely on the collective and synchronous shuttling of the electrons, which makes the conveyor-belt mode attractively more suitable. Indeed, all shuttled electrons can simply be placed at different minima of the generated moving potential: no matter the size of the device, only three or four distinct signals need to be sent. The drawback is that each electron must occupy the space of three or four clavier gates rather than one. Shuttling is receiving growing attention in the silicon spin qubit community, and fast progress in its implementation enabled per-hop fidelities now reaching 99.99% [37].

The strength of this shuttling capability in the context of quantum error correction will be made evident in Chapter 5.

2.2.6 Valley degree of freedom

So far, we always assumed that electrons in singly-occupied quantum dots were sitting in the orbital ground state. While this should ideally be the case, non-adiabatic phenomena may excite electrons to the closest higher energy level: the excited valley state. Let us outline this notion.

The band structure of bulk silicon features six degenerate conduction bands: four in-plane and two out-of-plane. In the presence of gates in SiMOS or Si heterostructures, the six-fold degeneracy is lifted and the two out-of-plane bands become the lowest of the six: they are called the ground and excited valley states. The valley splitting typically varies between 10μeV and 200μeV in Si/SiGe [38–41]. Ideally always sitting in the ground valley state, an electron that is *e.g.* shuttled fast enough can non-adiabatically populate the excited valley state. This can

have a negative effect on the qubit control as these states have been shown to exhibit distinct g-factors [38]. Therefore, an excited-valley-state electron could start to precess in an uncontrolled manner, causing unwanted phase rotations. This phenomenon will be further explored and modelled in Chapter 5.

2.3 State-of-the-art

Electron spins in silicon quantum dots are therefore a promising physical platform to perform quantum computing: single- and two-qubit gates with fidelity well above 99% [30, 42, 43] or even beyond 99.9% [44] have been demonstrated, simple instances of quantum error correction have been shown to be feasible [45] or have already been implemented [46], and the technology has been scaled up to processors with up to 6 qubits [47]. Furthermore, as explained above, the compatibility with advanced manufacturing techniques [48, 49] and cryogenic classical electronics [50, 51] make them ideal candidates for large scale integration [19] and the implementation of dense two-dimensional grids of qubits [52, 53]. Finally, their strong shuttling capabilities offer superior opportunities for circuit compilation and error correcting code implementations. Today's challenges for scaling up silicon spins lie in the need for precise control, uniformity, and coherence across large arrays while maintaining high-fidelity qubit operations and scalable interconnects.

3 Quantum error correction

Contents

3.1 Classical error correction

In order to understand quantum error correction, it is relevant to first introduce classical error correction. Like their quantum counterpart, classical computers are prone to errors: bit flips. The aim of classical error correcting codes is to detect

these bit flips and correct them. The workflow is the following: logical bits are encoded using a combination of physical bits, which are subject to errors. Up to a certain amount of noise, these errors can be decoded and the initial logical bit can be retrieved [54]. The simplest example of this is the 3-bit repetition code, based on redundancy: each logical bit (or codeword) is encoded with 3 bits: 0 is encoded with 000 and 1 is encoded with 111. If one error happens on a physical bit, the logical bit can still be decoded by majority voting. However, if two or three errors happen, the process will fail and the logical bit will be decoded wrongly. More precisely, one could measure the operators $S_1(x, y, z) = x \oplus y$ and $S_2(x, y, z) = y \oplus z$, where $\oplus$ is the addition mod 2. Their values distinguishes single-bit-flip errors as follows, allowing for their detection and correction:

Errors	Physical bits after errors	Syndrome (S_1, S_2)
No error	000/111	(0,0)
Bit 1	100/011	(1,0)
Bit 2	010/101	(1,1)
Bit 3	001/110	(0,1)

The first column contains all errors with at most one flip; the second column represents the 0 and 1 codewords after corruption; the third column gives the *syndrome* (S_1, S_2) associated to a given error. One can see that there exists a one-to-one mapping between error patterns with at most one flip and syndrome values. In other words, knowing the syndrome is sufficient to infer the single-bit errors and correct them.

Let us define the above concepts more formally. A classical binary linear code $\mathcal{C}$ is a k-dimensional subspace of $\mathbb{F}_2^n$, where $\mathbb{F}_2^n$ is a binary field. n is the number of physical bits used for encoding and k the number of encoded codewords. Errors bring the codewords out of the code subspace which enables for the detection and subsequent correction. The weight of the minimum-weight logical operator is called the code distance d. Thus the maximum error weight for which the code is guaranteed to provide protection is $\lfloor \frac{d-1}{2} \rfloor$. Codes are often conveniently referred to as $[n, k, d]$ (*e.g.* $[3, 1, 3]$ for the 3-bit repetition code). These three parameters play a

vital role in understanding the performance of an error correcting code: the ultimate goal of error correction is to find codes with maximal rate k/n and distance d.

Rather than by its vectors, $\mathcal{C}$ is often described as the kernel of a parity check matrix H whose rows contains the parity checks S_i: $H_{i,j} = 1$ if and only if S_i acts on bit j. For instance, the 3-bit repetition code has parity check matrix:

$$H = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \tag{3.1}$$

For a given vector $x \in \mathbb{F}_2^n$, the syndrome is thus simply given by $s = Hx$. The value of the syndrome can then be used for decoding to infer the error pattern (*e.g.* with a lookup table as in the previous example).

3.2 Introduction to Quantum Error Correction

3.2.1 Limitations of classical error correction

Unfortunately, the simple principles explained above do not translate directly to quantum computers for a handful of reasons.

No-cloning theorem The redundancy principle used in the repetition code described above cannot be directly applied in quantum error correction as an arbitrary state $|\psi\rangle$ cannot be copied [55–57]:

$$\alpha|0\rangle + \beta|1\rangle \quad \text{cannot be encoded as} \quad (\alpha|0\rangle + \beta|1\rangle)^{\otimes 3}$$

However, the following encoding can instead be utilised to correct bit flips:

$$\alpha|0\rangle + \beta|1\rangle \quad \text{can be encoded as} \quad \alpha\left|\bar{0}\right\rangle + \beta\left|\bar{1}\right\rangle$$

with $\left|\bar{0}\right\rangle = |000\rangle$ and $\left|\bar{1}\right\rangle = |111\rangle$ being the logical states.

Quantum measurements While the value of a bit can be directly accessed in a classical context, it is not the case in quantum mechanics, where measurements are destructive, as explained in Chapter 1. A key challenge of quantum error correction is to be able to detect if the state of the system is in the logical subspace without

altering it. For instance for the 3-qubit repetition code, the logical subspace is spanned by $|000\rangle$ and $|111\rangle$. If an X error happens on the first qubit, the state will be in a superposition of $|100\rangle$ and $|011\rangle$ which are orthogonal to the logical subspace. It is then possible to find a set of measurements that will distinguish them from $|000\rangle$ and $|111\rangle$ without projecting onto the basis states: in this case the parity checks Z_1Z_2 and Z_2Z_3. In the example above, state with no error will yield $(+1, +1)$ for the outcomes of the parity checks, while an X error on the first qubit will yield $(-1, +1)$

Quantum errors Compared to their classical counterpart, qubits are prone to bit flips (X errors) but also phase flips (Z errors). The 3-bit repetition code is incapable of distinguishing phase flip errors as a Z error would keep the state within the logical subspace. However, phase flips can be corrected via a change of basis:

$$\left|\bar{0}\right\rangle = |+++\rangle \quad \text{and} \quad \left|\bar{1}\right\rangle = |---\rangle$$

The parity checks become X_1X_2 and X_2X_3. Naturally, this encoding now cannot detect X errors.

Additionally, it is worth noting that quantum errors are not limited to X and Z flips. First, Y errors can occur as well, but they simply reduce to a combined X and Z flip. Second, quantum errors are most generally linear combinations of X, Y and Z. While the study of general error mechanisms can be complicated, stabiliser measurements typically decohere the noise and project it onto simple Pauli errors [58, 59]. This is easily understood in the case of a single-data-qubit coherent error $R_z(\theta) = \cos(\theta/2)I + \sin(\theta/2)Z$: when measuring an X stabiliser involving the erroneous qubit, the error channel is projected onto I with probability $\cos^2(\theta/2)$ and Z with probability $\sin^2(\theta/2)$. In more complex cases where coherence might survive, twirling can be used to simplify error channels and transform them into Pauli noise [60].

Shor and Steane code Combining all the principles above, Shor proposed in 1995 a 9-qubit code capable of correcting both X and Z errors [61] by *concatenating* the bit-flip and phase-flip error correcting codes:

$$\left|\bar{0},\bar{1}\right\rangle = \frac{1}{2\sqrt{2}}(|000\rangle \pm |111\rangle) \otimes (|000\rangle \pm |111\rangle) \otimes (|000\rangle \pm |111\rangle)$$

A year later, Steane proposed a code capable of the same task with only 7 qubits and without concatenation [62]. In his code:

$$\begin{aligned}\left|\bar{0}\right\rangle = \frac{1}{8}(\,&|0000000\rangle + |1010101\rangle + |0110011\rangle + |1100110\rangle + |0001111\rangle + |1011010\rangle \\ &+ |0111100\rangle + |1101001\rangle)\end{aligned}$$

and a similar form for $\left|\bar{1}\right\rangle$.

3.2.2 Stabiliser formalism

As we can see in the previous examples, describing the logical states as a superposition of basis states quickly becomes unfeasible as the number of qubits is increased since the size of the Hilbert space grows exponentially. Fortunately, another formalism exists to describe quantum error correcting codes without having to explicitly derive the logical subspace: the stabiliser formalism [63–67].

The idea is to extend the concept of parity checks introduced in the previous section, using operators that can inform us about the quantum state without collapsing it. These operators form the *stabiliser group* $\mathcal{S}$ which:

- is a subgroup of the Pauli group on the n physical qubits
- is Abelian (all its elements commute)
- does not contain $-I$

Owing to the commutation property, it is possible to find a subspace of the total Hilbert space whose vectors will be $+1$ eigenvectors of all elements of $\mathcal{S}$: this is the logical subspace. $\mathcal{S}$ increases exponentially with the size of the code but

its elements can be listed easily via its generators. For instance, for the 3-qubit bit flip code, these generators are Z_1Z_2 and Z_2Z_3.

Basically, the stabilisers act like constraints that restrict the dimension of the logical subspace compared to the total Hilbert space. Similarly to classical linear error correction, quantum stabiliser codes can be described by parity check matrices, using the binary representation of Pauli operators. Given a n-qubit Pauli operator P, a unique binary vector $\mathbf{b} = (\mathbf{x}|\mathbf{z})$ of size $2n$ can be associated to it (up to a phase factor). $\mathbf{b}$ is defined such that the i-th component of $\mathbf{x}$ is 1 iff the i-th operator in P is X or Y (and similarly for $\mathbf{z}$ with Z or Y). The parity check matrix of a quantum code can then be defined as

$$H = (H_X|H_Z) \tag{3.2}$$

where the i-th row of H is the binary representation of the i-th stabiliser generator of the code. Contrary to classical error correction, quantum error correction thus effectively requires two check matrices H_X and H_Z, to account for bit flips as well as phase flips.

A quantum error correcting code $[\![n, k, d]\!]$ is thus defined as:

$$\mathcal{C} = \{|\psi\rangle \in \mathbb{C}^{2^n} | H|\psi\rangle = |\psi\rangle\} \tag{3.3}$$

n is the number of physical qubits, k the number of encoded logical qubits ($k = n - \text{rk } H$ where rk H is the rank of H) and d the code distance. The maximum number of correctable errors is $\lfloor \frac{d-1}{2} \rfloor$.

Stabiliser measurements on an uncorrupted state leave it unchanged as per Eq. 3.3. Errors however bring it outside the logical subspace, meaning that some stabiliser outcomes would flip from $+1$ to -1. The knowledge of all stabiliser outcomes, or syndrome, can easily be obtained from the parity check matrix as $s = HE$, where E is the binary representation of the Pauli errors. This syndrome can be used to infer the initial errors via a classical decoding algorithm. The simplest example of such decoder is a lookup table, which we used for the classical repetition code described earlier. Note that the size of the lookup table however

grows exponentially with the size of the code, making this decoder impractical for large codes: the design of high-performance decoders is a large area of research which I will review in Section 3.3.3.

3.2.3 Extension to subsystem codes and gauge operators

I here present an extension of stabiliser codes, namely subsystem codes, as these will play a vital role in Chapter 4. In subsystem codes, part of the encoded Hilbert space is treated as a *gauge subsystem* rather than as protected logical information. Instead of correcting all errors that act non-trivially on the full code space, subsystem codes focus on preserving only a designated logical subsystem, while degrees of freedom corresponding to the gauge subsystem may fluctuate without affecting the stored information. This relaxation allows for more flexible code constructions, often reducing the weight or number of required check operators [68]. Formally, a subsystem code partitions the code space as

$$\mathcal{H}_{\text{code}} = \mathcal{H}_L \otimes \mathcal{H}_G,$$

where $\mathcal{H}_L$ carries the logical qubits and $\mathcal{H}_G$ contains gauge qubits that are not required to remain fixed.

Gauge operators are those that act non-trivially only on the gauge subsystem and generate the *gauge group*, which extends the stabiliser group. The stabilisers of the code correspond to the centre of the gauge group, *i.e.* operators that commute with every other element of the group, and these stabilisers define the protected subspace in which the logical subsystem resides. Unlike stabiliser generators, gauge operators themselves need not commute, which enables the use of low-weight generators and can simplify syndrome extraction. In many subsystem codes—such as Bacon–Shor codes [69], subsystem surface codes [70], or subsystem colour codes [71] – the gauge operators are chosen to be geometrically local and of low weight, reducing measurement overheads and improving practical fault tolerance.

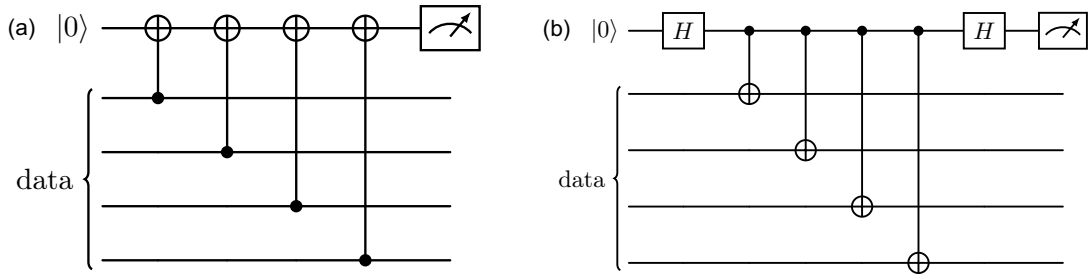


Figure 3.1: Possible circuit implementation of (a) a Z stabiliser measurement and (b) an X stabiliser measurement on four data qubits.

3.2.4 Fault-tolerance and threshold theorem

At this stage, we have considered the ideal case of perfect encoding and decoding of the quantum information at a fixed point in time. However, like the physical qubits, these processes are subject to errors that must be taken into account to properly describe our quantum device and hopefully achieve low logical error rates. For instance, stabiliser measurements are performed through a succession of quantum gates which would have a finite infidelity in any experimental implementation (Fig. 3.1). To get around this limitation, the idea is to adopt a non-static approach and perform repeated error correction over time: if errors are corrected faster than they appear, *fault-tolerance* is achieved.

A major step towards fault-tolerance is achieved via the threshold theorems. They state that below a given physical error rate p_{th}, one can reach a logical error rate as low as desired with an overhead scaling poly-logarithmically with the size of the ideal circuit. The hypotheses underlying these theorems depend on their version but generally suppose a low rate of correlated errors between qubits, that quantum gates on different qubits can be implemented in parallel and that classical post-processing is not a bottleneck (which will be reviewed in subsection 3.3.3).

The first versions of this theorem were proven in [72–75] for the case of concatenated codes, where the encoding explained before is just applied recursively, each logical qubit becoming a physical qubit of a higher-level structure (see Shor's code in Section 3.2.1).

Some later works considered different types of codes called *topological codes* for which a threshold was also proven [74, 76–79]. Particular examples of topological codes will be further explored Section 3.3.

Note that the existence of a threshold is not a *sine qua non* condition for a code family to be powerful. Some quantum codes, which I will review in Section 3.3.2 do not possess a threshold but still promise important reductions of the logical error rate, which is what ultimately matters.

3.2.5 Quantum computation with logical qubits

Threshold theorems explain how to detect and correct errors over time on a system that we do not act on, hence prove the feasibility of constructing a stable quantum memory over time. However, the ultimate goal is not only to store quantum information but also operate on it through quantum gates.

One first thing to note is that the Pauli corrections one infers from regular stabiliser measurements and decoding need not be physically applied on the device, or only rarely. This is because Pauli corrections can simply be stored in a classical software and updated as new errors are detected or as Clifford gates are applied (since Clifford gates map Paulis to other Paulis). Once at the end of the algorithm, the qubits measurement results are updated according to this so-called *Pauli frame*. When non-Clifford gates are applied however, the Pauli corrections may have to be physically implemented, or one may rather choose to update the non-Clifford operations themselves. This means that most of the time Pauli corrections on data qubits can be applied virtually. Hence, their implementation will not increase the logical error rate, or only marginally [80].

As for logical-level gates, they are effectively implemented via complex procedures targeting multiple data qubits at a time. To guarantee fault-tolerance, one must thus ensure that errors do not spread through them: a logical gate acting on physical qubits with x errors should not produce more than x errors. While Gottesman proved in 1997 [66] that this property can in theory be satisfied with any stabiliser code, the aim ultimately is to perform these fault-tolerant gates with minimal

resources while guaranteeing that the computational threshold is as close as possible to the memory threshold identified above.

An example of fault-tolerant gate with no overhead is a transversal CNOT for which CNOTs are applied in parallel to each pair of physical qubits of the two logical qubits. Transversal CNOTs can be performed in any Calderbank-Shor-Steane (CSS) codes [66], that is codes for which X and Z errors can be decoded separately. However, it was shown [81, 82] that a universal set of transversal quantum gates cannot be implemented on a nontrivial quantum error correcting code. Hence the need to further research a set of fault-tolerantly implementable gates. In 2013, Bravyi and Koenig [83] considered the case of gates implemented by constant-depth quantum circuits as they would also be fault-tolerant: an error on one qubit can only spread to a limited number of other qubits. However, they showed that 2D stabiliser codes could only implement Clifford gates this way, hence forcing one to use higher-dimensional structures or accept larger overheads for the implementation of the T gate for instance.

In turn, several techniques have been developed to implement these gates, like code switching [84] and magic-state distillation [16, 85, 86]. In the latter, multiple noisy logical magic states $|m\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + \mathrm{e}^{i\pi/4}|1\rangle\right)$ are prepared and *distilled* to produce a single low-noise magic state. One common way is through the 15-to-1 magic state distillation protocol. It is based on the [[15,1,3]] quantum Reed–Muller code, a stabiliser code which crucially has a transversal T gate, meaning that applying the physical T gate to each qubit implements a logical T gate without spreading errors between qubits. In the protocol, fifteen noisy copies of the magic state $|m\rangle$ are treated as encoded data; by measuring the code's stabilisers, any error syndromes are detected, and only outcomes consistent with the code space are accepted. This process effectively filters out correlated errors and suppresses the logical error rate from $O(p)$ to $O(p^3)$ where p is the input error probability. Importantly, one only performs error detection and post-selection – there is no error correction – which allows for the third-order error reduction at the cost of increased sampling. The output is a single, higher-fidelity magic state, and

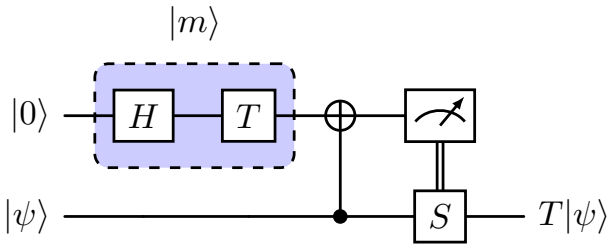


Figure 3.2: Implementation of a T gate on the state $|\psi\rangle$ through the preparation of a magic state $|m\rangle = T|+\rangle$ and subsequent gate teleportation. The double vertical bar indicates that the S gate should only be applied on state $|\psi\rangle$ if the ancilla was measured in state $|1\rangle$.

although the procedure consumes many noisy inputs per distilled state, it serves as a foundational primitive for achieving the ultra-low error rates required in large-scale, fault-tolerant quantum computation. Alternative magic state cultivation has recently been proposed [87] and has been shown to largely alleviate the resource requirements required to run algorithms such that RSA factoring [88].

In addition, while attractive for their simplicity and low overhead, transversal gates may sometimes not be the easiest to implement experimentally, especially if the physical qubits from two different logical qubits are far apart. This justifies the research on non-transversal implementations of normally-transversally-implementable gates. An example is the application of a CNOT between two surface codes via lattice surgery [89, 90].

3.2.6 Numerical simulation of quantum error correcting codes

Throughout my PhD, I have had to simulate the performance of error correcting codes on multiple occasions. Here I present the algorithm I used to calculate the logical error rate of a code used in memory mode (no logical gate applied). The first thing to note is that, contrary to most quantum systems, QEC codes are efficient to simulate. Indeed, the quantum circuits associated with the stabiliser measurements, as well as the considered quantum errors (Pauli operators) are all

Clifford gates. As mentioned in Chapter 1, this means that they can be simulated in polynomial time on a classical computer [15, 91].

After choosing a code to simulate, one must decide on a specific noise model. While I considered multiple ones throughout this thesis (which will be detailed in the relevant chapters), two of them are generic and consistently considered in the literature hence deserve to be described in this introductory chapter: phenomenological and circuit-level noise. In the former, X and Z errors are applied to data qubits at each round of error correction with probability p, and stabiliser measurement outcomes are classically flipped with the same probability p. This model is a convenient and simplistic way to estimate the performance of a code, but it does not capture the physical implementation of the stabilisers (via quantum gates). In turn, circuit-level noise offers this opportunity by rather considering that each process in the stabiliser circuits has a finite probability of failure. More concretely, this generally consists in applying depolarising channels after qubits initialisations and after quantum gates, and to classically flip measurement outcomes with a certain probability. The noise strength can be process-dependent, *e.g.* two-qubit gates may be faultier than single-qubit gates.

As mentioned previously, in the presence of measurements errors, rounds of stabiliser measurements must be repeatedly performed, effectively creating a space-time error syndrome. The number of error correction rounds is usually chosen to be equal to the code distance d. At each time step t, some new errors E_t may affect the data qubits, generating a syndrome $s_t = HE_tE_{t-1}...E_1$, where H is the parity check matrix of the code (we are here using the binary representation of the errors). This syndrome may itself be incorrectly measured, leaving the user with an inconsistent syndrome $\tilde{s}_t$. It is however common practice to consider that the last syndrome is perfectly measured, as at the end of a quantum computation, all data qubits can be measured. Hence the final syndrome can be inferred classically from the parity of data qubits measurements outcomes. After all d rounds of stabiliser measurements, the space-time syndrome $(\tilde{s}_1, ...\tilde{s}_{d-1}, s_d)$ is passed to a classical decoding algorithm, inferring a correction $\bar{E}$ explaining the final perfect

syndrome: $s_d = H\bar{E}$. The correction can then finally be applied to the logical state, which consequently returns to its logical subspace: $H\bar{E}E_d...E_1 = 0$. Note that the decoding algorithm generally uses the difference syndrome $(\tilde{s}_{t+1} - \tilde{s}_t)_{0\leq t\leq d-1}$ (with $\tilde{s}_0 = 0$), which is sparser than the original syndrome $(\tilde{s}_t)_{1\leq t\leq d}$.

To finish, one must verify if the accumulated errors over time and computed correction corrupted the state of the logical qubit. This operation is performed by checking if the total error $\bar{E}E_d...E_1$ commutes with all logical operators (which can easily be verified classically using known Pauli commutation rules).

The workflow to simulate a code's performance can then be summarised as follows:

1. generate errors on data and ancilla qubits according to the chosen noise model
2. compute the perfect syndrome at each time step $s_t = HE_tE_{t-1}...E_1$, then the erroneous syndrome $\tilde{s}_t$ due to measurement errors
3. decode the space-time syndrome via a classical decoding algorithm, and deduce a correction $\bar{E}$
4. add $\bar{E}$ to the accumulated errors $E_d...E_1$ and check for logical failure

3.3 Search for *good* codes

3.3.1 Criteria for good codes

In this section, I will briefly review techniques that lead to the construction of high-performance codes, in particular the ones used in Chapter 5. Multiple criteria should be considered when designing a quantum error correcting code:

(C1) Code distance d and code rate k/n For optimal error correction capabilities, an error correcting code should have a high rate k/n (meaning little overhead) and high distance (strong error reduction power). The rate is trivially bounded by 1 and the distance by n. Codes for which $k = \Theta(n)$ and $d = \Theta(n)$ are called *good* codes as they are, at least asymptotically, optimal.

(C2) Logical error rate, threshold and pseudo-threshold For code families that possess a threshold, the threshold value is an extremely important characteristic of the performance of the code. Indeed, it dictates the maximum physical error rate the code can tolerate to reach its exponential error suppression regime. As noted earlier however, not all code families considered in this thesis possess a threshold and this should not be viewed as an indication of a weak code: what ultimately matters is the achieved logical error rate at the device's working point. This should be far below the pseudo-threshold, defined as the point where the logical error rate equals the physical error rate for a given code.

(C3) Embedding the code in a physical device Codes are not just abstract mathematical constructs: they ultimately aim to be embedded in a physical chip, which raises multiple challenges. Firstly, it may be easier to design codes that can be laid out in a 2D plane, as 3D structures are inherently more challenging to engineer. Secondly, local operations between nearest neighbours may be favoured for they are more straightforward to implement. Yet, [92] showed that the performance of local 2D codes is upper-bounded by $kd^2 = O(n)$, meaning they cannot be *good* codes (in the sense defined two paragraphs earlier).

(C4) Set of implementable logical operations Importantly, a code must be able to implement a universal set of quantum gates efficiently *i.e.* with a minimal space-time cost and respecting the constraints imposed by the physical device.

(C5) Existence of an accurate and fast decoder Finally, a code is run jointly with its decoder, which must be (*i*) accurate (*i.e.* correct errors with high probability); (*ii*) fast (to keep up with the quantum computation) and (*iii*) scalable (even in a fault-tolerant regime with millions of physical qubits, the memory usage must be moderate).

With these principles in mind, I will review a few codes and their construction, which will aid the comprehension of Chapter 5.

3.3.2 qLDPC codes

The first quantum error correcting code, the Shor code, was built by concatenation, which consists of recursively encoding each physical qubit of a code with another code itself. While attractive for their conceptual simplicity, concatenated codes have one major downside: the increase of the stabiliser weights (*i.e.* number of data qubits involved in each stabiliser check) and of the non-locality with the levels of concatenation. This quickly led the QEC community to consider another paradigm: the quantum Low-Density-Parity-Check (qLDPC) code family. A code family is said to be LDPC if each data qubit is involved in a constant number of stabilisers, and each stabiliser acts on a constant number of data qubits. These codes are particularly attractive from a practical point of view as the low weight of the stabilisers implies (*i*) that errors are less likely to propagate and do not scale with the size of the system (*ii*) that the stabiliser cycles can be implemented in constant time.

These codes are most easily described with their parity check matrix. As explained in Section 3.2.2, a quantum stabiliser code $\mathcal{C}$ can be represented as the kernel of the parity check matrix $H = (H_X, H_Z)$. By definition, the parity check matrix of a qLDPC code is row- and column-sparse. Contrary to classical codes, this matrix cannot be chosen arbitrarily as the stabilisers defining the code must commute. Given known Pauli operators commutation relations, this translates into the following condition:

$$H_X H_Z^T + H_Z H_X^T = 0 \tag{3.4}$$

Code design therefore relies on the generation of *good* matrices H_X and H_Z satisfying the above relation. This is done by choosing *ansatzes* for H_X and H_Z. In this thesis, I will focus on a case that has been comprehensively addressed in the literature: CSS codes. These codes are characterised by stabilisers acting exclusively as X or Z checks, not a mix of both. Consequently, the overall parity check matrix has the following form:

$$H = \begin{pmatrix} H_X & 0 \\ 0 & H_Z \end{pmatrix} \tag{3.5}$$

and the condition of Eq. 3.4 now reads:

$$H_X H_Z^T = 0 \tag{3.6}$$

More simply put, X and Z checks must have an even number of data qubits in common. One further simplification generally consists of writing H_X and H_Z as block matrices:

$$H_X = (A \mid B) \tag{3.7}$$

$$H_Z = \left(B^T \mid A^T\right) \tag{3.8}$$

which transforms Eq. 3.6 into a simpler commutation relation $AB + BA = 0$, or equivalently since these matrices are in $\mathbb{F}_2^n$:

$$AB = BA \tag{3.9}$$

The objective now is to search for commuting matrices that would lead to the most optimal sets of parameters n, k and d. A natural way to do so is to consider large families of commuting matrices and explore the space they span. Several kinds of ansatzes for A and B have been suggested; in this thesis, in particular in Chapter 5 and 4, I will focus on two of them: hypergraph product codes (HGP) and generalised bicycle codes (GB). HGP codes are built by taking the product of two classical codes with parity check matrices C and D:

$$A = C \otimes I$$

$$B = I \otimes D^T$$

One famous example of hypergraph product code is the surface code, whose classical seed codes both are repetition codes. Its parameters are $[\![d^2 + (d-1)^2, 1, d]\!]$. For years, the surface code has been deemed as one of the most promising error correcting codes owing to its fully local implementation and high threshold, which come at the cost of a minimum code rate ($k = 1$). As it is at the heart of my research on QEC, I will describe it in further details in the next section. However attractive,

recent experimental improvements showed that non-local stabilisers may not be a deal breaker ([93] managed to implement non-local operations in QEC codes on reconfigurable atom arrays via shuttling). This possible relaxation of local constraints pushed research on qLDPC codes with better parameters from a purely theoretical realm to more practical grounds. GB codes are among them, where A and B are defined as circulant matrices. For instance, a $[128, 28, 8]$ GB code requires 128 physical qubits to encode 28 logical qubits of distance 8; this would take 3164 physical qubits using surface codes, so 24 times more.

In both cases of HGP and GB codes, the commutativity 3.9 is guaranteed and the code parameters can be estimated via mathematical theorems [94, 95].

Many more types of qLDPC codes and related constructions exist. Firstly, one may mention colour codes, as it is a vast class of codes that attracted a significant amount of research [96–100]. Steane's code is the smallest one. They are very similar to the surface code and are even equivalent up to a geometrical unitary [101, 102], but feature a few differences: colour codes have a lower threshold than surface codes [100] as their stabilisers act on more qubits (up to 6). However, if allowing an extra dimension, colour codes feature a transversal implementation of a universal set of gates. Even if a 3D implementation could appear particularly daunting for any experimentalist, some work I have been involved in suggested that theoretically-3D structures could be embedded in strictly 2D devices (see Section 5.6)

Recent efforts towards the construction of *good* qLDPC codes (in the sense defined in Section 3.3.1) led to the design of *fiber bundle codes* [103], followed by *balanced products* of *quantum expander codes* [104]. While unlikely to be experimentally practical, the latter code is characterised by an optimal asymptotic scaling $k = \Theta(n)$ and $d = \Theta(n)$.

3.3.3 Decoders

In order to be functional, all of the codes described above need an efficient decoding process so as to extract the errors on the data qubits from the stabiliser

measurements. Distinct codes may function with distinct decoders. I will review some of them in the following paragraphs.

Lookup tables This is the decoder I have been using in all the examples so far as it is the simplest but also the most applicable one (it applies to all kinds of the error correcting codes). The idea is simply to store the syndrome events associated with each error pattern. Once a certain syndrome is measured, it can be decoded by choosing the most likely error pattern that may have generated it. This decoder hence has an optimal accuracy — it is a *maximum likelihood decoder* — and it is fast, as all the decoding calculations can be done prior to the quantum computation, then stored in memory and just accessed. However, this decoder is only appropriate for small codes as its memory usage scales exponentially with the system size. Some suggestions have been made to make lookup tables more scalable, *e.g.* by only storing the most probable events [105].

Minimum Weight Perfect Matching (MWPM) To circumvent the scalability issues explained above, several code-specific decoders have been developed. MWPM is one of them and has been intensively studied in the case of the surface code. It is based on Edmonds' blossom algorithm [106, 107]. This decoder explains the syndrome events by the smallest total number of errors but does not take degeneracy into account, *i.e.* that certain syndrome events are more likely simply because multiple same-weight error patterns can explain them. Yet, this decoder yields excellent performance, very close to maximum likelihood decoders, and its best implementations are relatively fast, the algorithm running in time $O(n)$ [108–111]. This is the decoder I will be using in the rest of this thesis for surface code simulations.

Belief propagation (BP) While MWPM is tailored to the surface code, belief propagation decoding is very general and can be applied to any qLDPC code [95, 112]. As maximising the probability of a total error pattern E given a syndrome S is too hard to be solved exactly — it is an NP complete problem —, belief propagation attempts to maximise the marginals instead (*i.e.* the probability of

single-qubit errors). This is done by passing messages between cells corresponding to neighbouring data and ancilla qubits and updating them recursively until convergence is reached. If this algorithm has been known for a long time for classical codes, it first seemed intricate to apply it to quantum codes because of the presence of loops and degeneracy in the syndrome graph. More elaborate schemes have been developed since then to implement a quantum version of belief propagation, first using heuristic rules [113], then ordered statistics [95, 114] and memory belief propagation (a mix of BP and neural networks) [115]. BP also proves useful combined with MWPM as it allows to predict better weights for the matching graph, when correlations exist in the single-qubit errors [116, 117]. When simulating qLDPC codes other than the surface code, I will be using a BP-OSD decoder from Ref. [95].

3.4 Case study: the rotated surface code

3.4.1 Brief history of the surface code

In the previous section, I introduced the surface code with the hypergraph product code formalism. This is not the process that initially led to it: rather, Kitaev first invented the toric code, inspired by 2D systems with topologically protected properties [74, 78]. Analogous to this kind of systems, strings of errors forming trivial loops are topologically equivalent to an error-free state; non-trivial loops around the torus however lead to a change of the logical state. If laid out on a plane, this code is characterised by fully local stabilisers except at the edges — the toric code indeed supposes periodic boundary conditions. It was however shown shortly after that this periodicity could be discarded, which led to a much more implementable code: the surface (or planar) code [76, 77, 118]. Early works [119] showed that the performance of the surface code also largely surpassed that of initially-suggested concatenated codes, with a threshold of 3% under phenomenological noise. One further improvement of the surface code led to the rotated surface code, obtained by rotating the surface code by 45° (which roughly leads to a x2 overhead reduction

for large code distances at a very moderate cost on the performance). This is the code I will be further describing in this section, and using in the next chapters.

3.4.2 Code description and performance

The rotated surface code is a $[d^2, 1, d]$ code. It is a CSS code, meaning that each stabiliser is either Pauli X or Pauli Z but not a mix of both. X and Z errors can thus be decoded separately. A rotated surface code is represented in Fig. 3.3. Each red (resp. green) tile represents an X (resp. Z) stabiliser. As pictured in Fig. 3.1, X and and Z stabilisers values are obtained by measuring an ancilla qubit that previously interacted with corresponding data qubits. Such interactions are generally performed in the form of CNOT gates (targeting the ancilla for X stabilisers and the data qubits for Z stabilisers); they are schematically represented in Fig. 3.3 for one pair of X and Z stabilisers.

When Pauli errors (green stars) affect the code, the quantum state generally leaves the code subspace and some stabilisers outcomes switch from $+1$ to -1 (bright red squares). This syndrome s can then be fed to a classical decoding algorithm, whose purpose is to infer a correction (green circles) explaining the syndrome. This correction is finally applied to bring the code back to the logical subspace, ideally back to its error-free state, but potentially inducing a logical failure.

The total error (combined errors and correction) after a round of error correction can take several forms. The first possibility is a closed loop (orange loop in the figure), which can easily be seen as the product of the stabilisers inside the loop: as stabilisers act as identity of the logical state, closed loops correspond to the identity logical operator I_L. Alternatively, if the total error is a vertical (resp. horizontal) string of Z (resp. X) errors, a Z_L (resp. X_L) logical error occurs (green and red strings in the figure).

In order to reduce the time taken by each stabiliser round, CNOTs participating in X and Z stabilisers can be interleaved and implemented in parallel. Furthermore, the CNOT ordering in each individual stabiliser has to be chosen so as to avoid so-called hook errors: if an ancilla qubit is victim of a given error half-way through

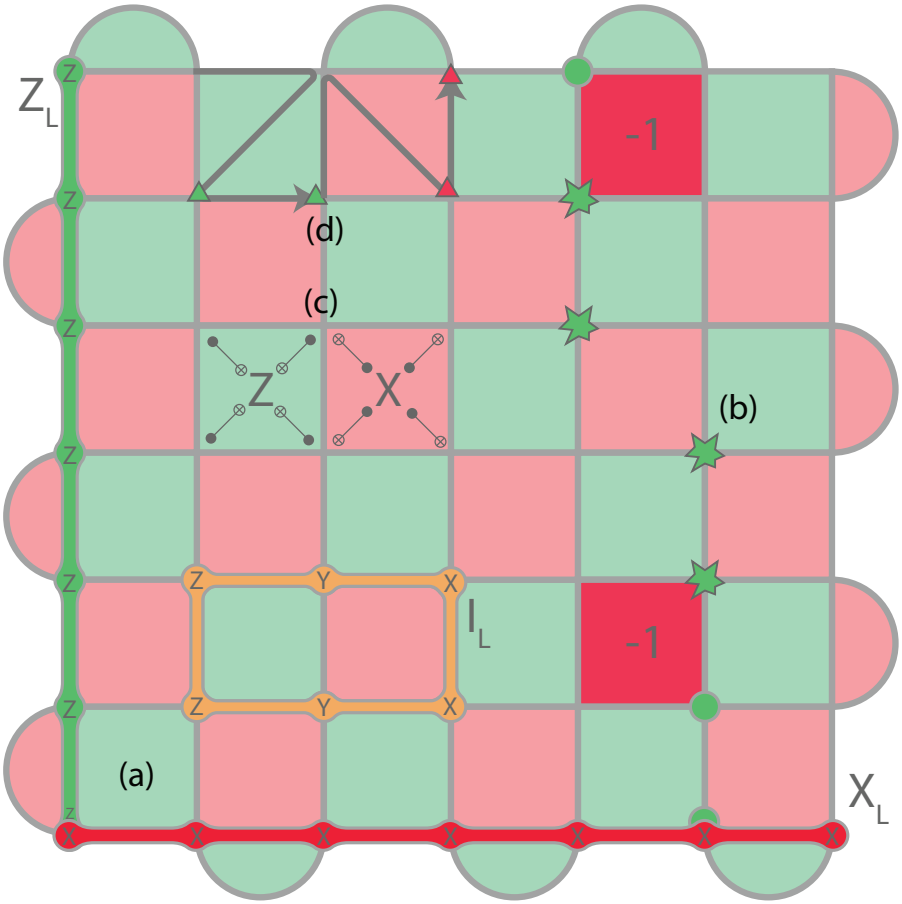


Figure 3.3: Rotated surface code. Data qubits lie at the vertices. Green (resp. red) tiles represent Z (resp. X) stabilisers. (a) Vertical strings of Z errors correspond to Z_L logical operators, while horizontal strings of X errors correspond to X_L logical operators. Loops (in orange) do not change the logical state as they can be written as a product of stabilisers. (b) When errors happen (green stars), some stabilisers may render a non-trivial measurement outcome (bright red X stabilisers). This information can be fed to a classical decoding algorithm to compute a correction (green circles) that will bring the state back to its operational subspace but may induce a logical failure (here a Z_L error is induced). (c) The stabilisers can be implemented at the physical level via CNOTs. (d) The ordering of the CNOTs must follow a given pattern so that hook errors (green and red triangles) are orthogonal to the corresponding logical operators.

a stabiliser circuit, errors can propagate to the subsequent data qubit through the CNOTs, effectively doubling the number of errors (see Fig. 3.4). It is thus paramount to order the CNOTs such that induced double errors are not parallel to the logical error strings (or the code distance would be halved). Note that one should not be worried about triple errors as, up to a closed loop, three errors around a stabiliser are equivalent to a single error. For both these reasons — interleaving X and Z stabiliser circuits and avoiding hook errors — the CNOT ordering responds

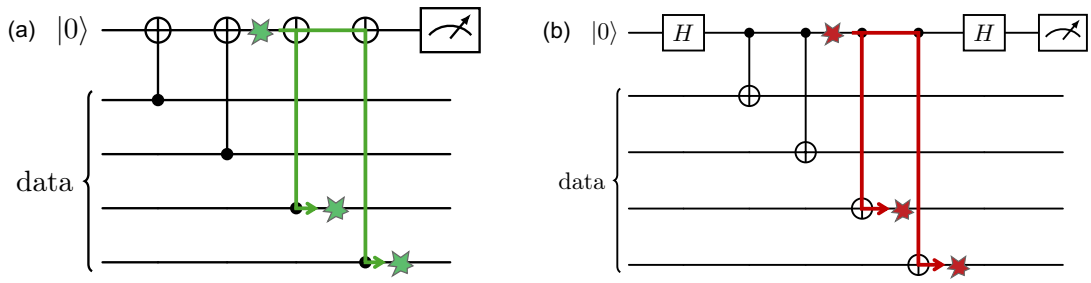


Figure 3.4: Schematic representation of hook errors for (a) a Z stabiliser and (b) an X stabiliser. Green (resp. red) stars represent a Z (resp. X) error occurring on the ancilla qubit half-way through the stabiliser measurement. The consequence is the propagation of the error to both subsequent data qubits.

to specific restrictions [120]. One choice satisfying both conditions is presented in Fig. 3.3 with the Z- and N-shaped arrows. The green (resp. red) triangles picture Z (resp. X) hook errors: they do not reduce the code distance as they sit perpendicularly to the logical operators.

Taking all these elements into account and using the simulation protocol presented in Section 3.2.6, I could simulate a rotated surface code under circuit-level noise using MWPM for decoding. The output was produced via an end-to-end pipeline that I implemented myself (apart from Edmond's algorithm for decoding). It reproduces well-known results with a relatively high threshold of 0.7% (Fig. 3.5).

3.4.3 Use in computation

Beyond its high threshold and conveniently local and 2D implementation, the surface code is a promising QEC platform for its demonstrated capabilities in computation. Three ingredients are required for such a task: state initialisation, state measurement and logical gates.

State initialisation

Procedures aiming at initialising a QEC code in a specified state are called *state injection*. Many of them have been engineered for the surface code, and they all rely on preparing each physical qubit in a given state before measuring the stabilisers.

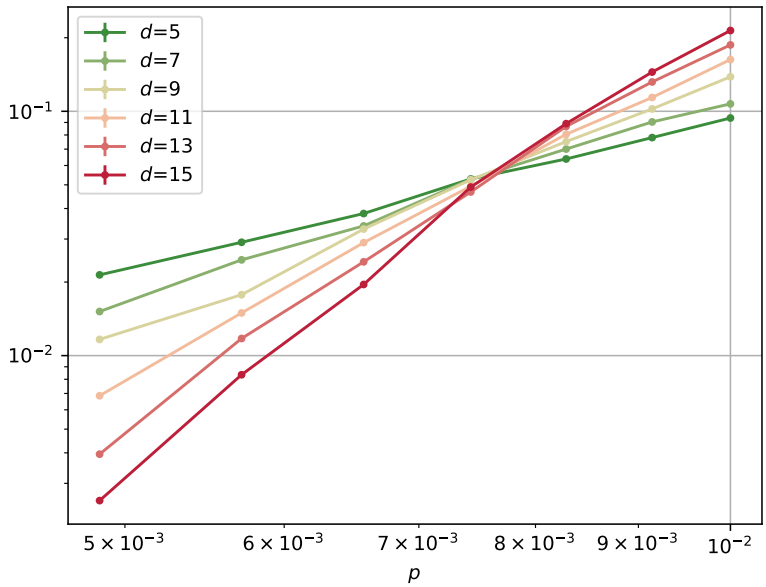


Figure 3.5: Z-memory experiment for the surface code under circuit-level noise. Depolarising channels of strength p are applied after every single- and two-qubit gates and after qubit initialisation. Measurement outcomes are classically flipped with probability p as well.

The latter action subsequently projects the total surface code state onto the desired logical state (assuming no error happened).

As an example, let us assume that one wants to initialise the surface code in the $|0_L\rangle$ logical state by preparing all physical qubits in the $|0\rangle$ state. The value of the Z_L operator from Fig. 3.3 is thus $+1$. Without additional errors, Z stabilisers should all render a $+1$ outcome as well, while X stabilisers would all be random. As Z_L commutes with all stabilisers, the action of measuring them thus projects the total state onto a common eigenstate of all stabilisers, while maintaining $Z_L = +1$: the logical state is correctly prepared in the $|0_L\rangle$ state. If any X error happened during the process, it can be detected by the Z stabilisers and corrected through regular error correction. Z errors here do not matter as they do not affect the $|0_L\rangle$ state. Note however that with this procedure the logical subspace does *not* correspond to the $+1$ eigenstate of all stabilisers, as roughly half of the X stabilisers measurements should yield -1 (but this is not an actual issue).

The preparation of more complex states, such as magic states, is similar yet a bit more sophisticated. Efficient protocols using a combination of regular error correction and classical post-processing have been designed for the preparation of logical magic states with output fidelity as high as physical gate noise [121, 122]. Magic state distillation or cultivation can then be used to boost the state fidelity.

State measurement

Single-logical-qubit Pauli measurements are easily implemented on the surface code: it is sufficient to measure all physical qubits of the code and classically take the product of the relevant qubits measurement outcomes to deduce the value of the logical operator.

A trickier problem is the measurement of multi-logical-qubit Pauli operators. This is performed via a well-known technique called *lattice surgery* [89], which respects the locality and 2D implementations of the surface code.

This technique traditionally allows for the measurement of the two-qubit logical operators $X_L^1 X_L^2$ and $Z_L^1 Z_L^2$ via local gates only, where P_L^i is a Pauli operator on logical qubit i. Let me describe an $X_L^1 X_L^2$ measurement in greater detail. The protocol can be decomposed into two steps: merge and split (Fig. 3.6).

The *merge* step is carried out by measuring all four-body stabilisers at the codes common boundary (rather than the separate two-body ones). Those that did not commute with the initial stabiliser group (in bright red) render a random $\pm$ outcome, which can be used to infer the two-logical-qubit operator $X_L^1 X_L^2$. Indeed, the product of these random operators is exactly equal to $X_L^1 X_L^2$ (and is thus non-random: it commutes with the stabiliser group). As their physical implementation can be faulty, they however need to be measured for d rounds in order to accurately measure $X_L^1 X_L^2$ (where d is the code distance). Thereon, the merged lattice can be split back by measuring the stabilisers of the original surface codes. Two-body Z stabilisers (bright green) will render random results as they anticommute with previously measured operators, but the values of facing pairs will be perfectly correlated. This

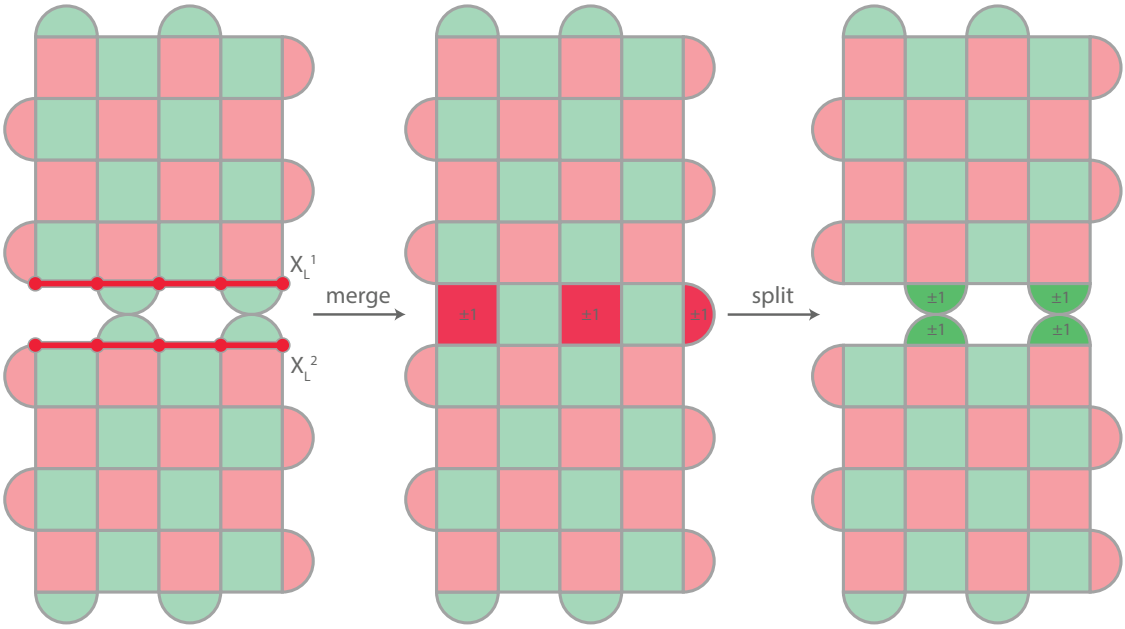


Figure 3.6: Schematic representation of an XX lattice surgery, inspired from [123]. (a) Two surface code patches as in Fig. 3.3. (b) Merge step of lattice surgery: bright red operators individually render random measurement outcomes, but their product is the $X_L^1 X_L^2$ two-qubit logical operator. (c) Split step of lattice surgery: initial stabilisers are measured again. Two-body boundary stabilisers are randomised but facing pairs should render the same outcome.

information can be used for subsequent decoding. $Z_L^1 Z_L^2$ measurements can be performed similarly if Z_L boundaries face each other instead of X_L boundaries.

Later on, Ref. [90] suggested that any multi-qubit Pauli measurement could be implemented via a variant of lattice surgery. Two-qubit Pauli measurements beyond $X_L^1 X_L^2$ and $Z_L^1 Z_L^2$ rely on the measurement of more complex stabilisers called *dislocations* and *twists* [124]. These stabilisers involve Pauli operators of distinct kinds, not all X or all Z. Any multi-qubit Pauli measurement can then be implemented by making use of long mediating ancilla qubits.

Gates

In the surface code, a convenient universal set of logical gates is Clifford+T. Multiple methods have been designed to implement them, but in this thesis, in particular in Chapter 5, I will mainly focus on a protocol designed by Daniel Litinski in [90, 124], that capitalises on the aforementioned augmented lattice surgery protocol.

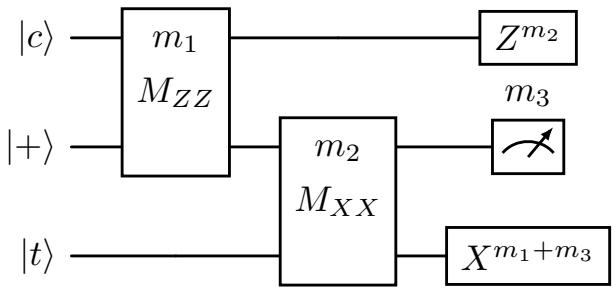


Figure 3.7: Decomposition of a CNOT gate on a control qubit $|c\rangle$ and a target qubit $|t\rangle$ via two-qubit measurements M_{XX} and M_{ZZ}. Their measurement outcomes m_1 and m_2, as well as the ancilla qubit measurement outcome m_3, are then used to perform corrective Pauli gates on $|c\rangle$ and $|t\rangle$.

A motivation for the development of lattice surgery was the attempt to find a fully local implementation of the logical CNOT gate. Indeed, while the surface code does support its transversal implementation, this procedure would require either to perform non-local operations between the physical qubits of the distinct logical patches, or to superpose two sheets of surface code in a higher-than-two-dimensional structure. Both these options would imply compromising on one of the surface code's greatest strengths: its full locality or 2D implementation. Rather, [89] suggested an implementation of the CNOT gate using the decomposition of Fig. 3.7 and lattice surgery for the two-qubit measurements. With this new implementation, the locality and low dimensionality of the surface code is preserved, at the cost of an increased time overhead, since the merge step requires d rounds of stabiliser measurements (while a transversal implementation would not).

Then [90] suggested that, beyond the virtual implementation of Pauli gates through the Pauli frame (see Section 3.2.5), Clifford gates could also be absorbed in such a frame-like picture. By commuting them through T gates, they could simply be absorbed in final measurements, at the cost of complexified $\pi/8$ rotations that would now span more than one qubit (see Fig. 3.8). In this picture, only such multi-qubit $\pi/8$ rotations have to be physically implemented. These can conveniently be carried out via the preparation of a distilled magic state and a multi-qubit measurement (similarly to Fig. 3.2 where the CNOT would be implemented via

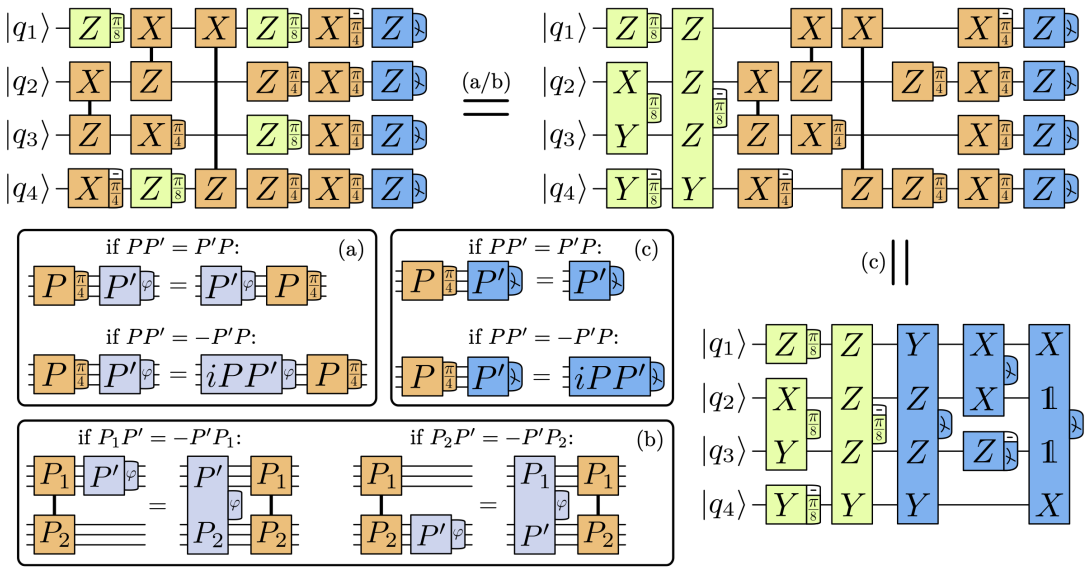


Figure 3.8: Implementation of a quantum circuit by commuting the Clifford gates (in orange) through the T gates (in light green). The commutation is performed using the rules in (a) and (b). This converts the T gates into more complex $\pi/8$ rotations, but removes the need to implement any of the Cliffords, which are absorbed in the final measurements (c).

the decomposition of Fig. 3.7). Coupled to the strong state injection protocols described above to prepare magic states, and to distillation protocols (that only require two-logical-qubit measurements implemented by lattice surgery), this scheme allows for universal quantum computation on the surface code.

3.5 Quantum error mitigation

While quantum error correction provides a rigorous and scalable framework for protecting quantum information against noise, its practical realisation remains beyond the capabilities of current noisy intermediate-scale quantum (NISQ) devices. Fault-tolerant QEC demands large numbers of physical qubits, high-fidelity operations, and substantial overhead in stabiliser or gauge measurements—resources that are not yet available at scale. In this context, quantum error mitigation (QEM) has emerged as a complementary strategy aimed at improving computational accuracy without encoding logical qubits or actively correcting errors during the computation. Instead of increasing the space overhead associated with QEC, QEM

requires a time overhead in the form of repeated sampling. I quickly review the main techniques enabling QEM in the last section of this chapter, as this will give important background for Chapter 7.

Quantum error mitigation operates by inferring the outcome of an ideal, noiseless quantum process from data obtained on a noisy device. Rather than suppressing or correcting errors at the hardware level, it relies on circuit modifications, repeated sampling, and classical post-processing to compensate for the effects of noise. Methods such as zero-noise extrapolation [125, 126] and probabilistic error cancellation [125] reconstruct ideal expectation values through noise amplification or quasi-probabilistic decompositions, while measurement error mitigation [127, 128] and symmetry verification [129, 130] target specific error channels by correcting classical readout biases or enforcing physically relevant constraints.

While promising for near-term architectures, QEM cannot achieve the long-term robustness or scalability promised by fault-tolerant QEC, as its associated sampling overhead grows exponentially with the circuit size. However, as explored in Chapter 7, it may be combined with QEC to alleviate its resource requirements in early fault-tolerance devices.

4

Adaptive surface code for quantum error correction in the presence of temporary or permanent defects

This chapter describes work leading to a 2022 paper (published in Quantum in 2023 [131]) with coauthors Armands Strikis and Simon Benjamin. It lies in the continuation of Armands' ideas [132] but was all produced by me. Throughout I benefitted from useful conversations with Armands and Simon.

All writing below is my own, in places using text verbatim from my published manuscript.

Contents

4.1 Introduction

In the previous chapter, I introduced the notion of quantum error correction, and showed how these techniques promise sufficient error reduction to run deep quantum algorithms. The drawback is a significant resource overhead, as considerable numbers of qubits must be assembled together to form error correcting codes.

In these large structures, it may be unrealistic to assume that all components will be working nominally after their fabrication. Even if this were the case, some events may temporarily or permanently disrupt their normal behaviour in the course of the computation: high-energy events leading to a surge of correlated errors, like cosmic rays in superconducting and silicon devices [133–135]; or leakage and loss of qubits, for instance in ion traps or neutral atom arrays [136–139]. In both cases, these events create *defects* on the lattice, either preexisting the computation or occurring while it is running. These defects must be distinguished from ones that can voluntarily be introduced in the surface code as a means to store logical information [140]. As experimentally identified in the repetition code [141], the uncontrolled defects studied in this chapter can alter the code's performance so profoundly that it eliminates the exponential suppression of errors upon which deep computations rely. For any hardware platform where this occurs, some protocol must be designed to deal with these defects.

The case of 'chip-level' errors was studied in [142], where it was shown that concatenating multiple codes over separate chips would effectively reduce the rate of catastrophic events. The research presented in this chapter is complementary as it studies smaller defects, that do not disrupt an entire chip. In this scenario, the goal is to retain the error correction capability in a code where some stabilisers cannot be measured or would just yield essentially random results. In both cases, my approach is to disregard all these *faulty qubits* manifesting the defect and remove them from the code.

At the time of this research, this picture had already been investigated for the surface code in [143–145]. These papers show that even when some defective zones of fixed size are removed from the code, effectively creating punctures in it, the code remains robust to errors as long as a modified set of stabilisers is measured. These punctures do have the effect of lowering the distance of the code. However, it remained unclear if defects of fixed density — rather than size — would disrupt the exponential suppression of errors of the surface code in the asymptotic limit. Recent work [132] introduced a method based on code deformation involving shells that isolate defects. That paper provided an analytic proof that a threshold must exist using their methodology. However, the proof involves a series of compounding worst-case assumptions so that a realistic estimate of the threshold could not be obtained. Moreover, obtaining a meaningful threshold requires the creation of bespoke decoders; in doing so there is the opportunity to generalise from factory defects to encompass defects happening on the fly.

In the present chapter, I tackle these questions and establish that a realistic high-defect-rate threshold does indeed exist. I numerically compare both of the previous existing approaches and show in which regime one is outperformed by the other, focusing on the case of memory storage — I do not study the resilience to errors when quantum gates are implemented. This knowledge, together with a novel defect detection algorithm that recognises events occurring during the execution of the stabiliser measurements, allowed the design of an *adaptive surface code* that deforms whenever a defect is detected so as to exclude it from the code. For such surface code, I then exhibit a threshold for defects detected on the fly during the stabiliser measurements, and compare it to that of a defect-free surface code. On top of this, I estimate the resource overhead needed to overcome these defects. Ultimately, I was able to numerically exhibit the existence of a *practical* threshold, *i.e.* one that would not impede the use of error correcting codes in the presence of defects, which, again, will be inevitable.

4.2 Methods

In this section, I present the general workflow of the adaptive surface code. It can be split in three distinct steps: first, the detection of the defective zone (if any); second, a code deformation allowing one to exclude the identified defect and thus store the logical information in the remaining qubits; third, the computation of the relevant stabiliser events and the deduction of a correction.

4.2.1 Formalism

First, I clarify the formalism that will be used in the rest of this section. The surface code is a stabiliser code made of d^2 data qubits that are repeatedly measured by $d^2 - 1$ independent stabilisers. This leaves a dimension-2 space for the logical qubit. If qubits are disabled to avoid a surge of errors in case of defect, the number of degrees of freedom in the code space increases, effectively adding *gauge operators* to the code: these operators still commute with all the stabilisers but can anticommute with each other. They are represented in brighter colours on the left-hand side of Fig. 4.2. Interestingly, [143–145] noted that, when puncturing a hole in a surface code, the product of the gauge operators around the hole however commutes with all the stabilisers and gauge operators, hence being itself a stabiliser of the code, called *superstabiliser* (see Fig. 4.2). In practice, the measurement of the superstabiliser is performed by measuring the individual gauge operators around the hole, and computing the product of these outcomes. As X- and Z-basis operators do not commute (only their product does), they have to be evaluated at different times. Measuring one subset of commuting gauge operators is called *gauge fixing* as — without errors — it fixes the otherwise random value of these operators when repeating their measurement.

4.2.2 Defect detection

Two different types of defects are distinguished in this chapter: defects that happen *before* the code is run and that are detected offline (*e.g.* fabrication defects), and defects that happen *while* the code is running and that must be identified on

the fly (*e.g.* cosmic rays). I emphasise that by 'defect' I refer to a cluster of qubits that are rendered effectively inoperable, either permanently or for a finite time; this is in contrast to the finite rate of transient errors which is presumed to afflict all qubits during the normal operation of the machine. In practice, it will be relatively straightforward to detect pre-existing defects: post-fabrication analysis by the manufacturer (quality control), and system calibration prior to a computation. The greater challenge is the real-time detection while the code is running. The approach obviously depends on the nature of the defect. I focus on a case known to occur: an instability in qubit(s) corresponding to noise far above the normal level, *i.e.* the behavior seen after a cosmic ray impact. In order to keep my method as general as possible, it is assumed that the only available information is the stabiliser measurements. Defects in more specific systems could however be detected more efficiently with tailored protocols [135]. A defect will be modelled via the appearance of a large density of correlated syndrome events in spacetime at a given location of the code (see Figure 4.1). The challenge is then to identify all the faulty qubits but to avoid misidentifying nominally-functioning qubits as faulty — and to do so in minimal time if the detection algorithm is run during the computation (rather than, say, in a calibration phase). To achieve this, stabilisers experiencing more than n_{flips} flips within a time Δt_{flips} are first identified, then gathered in clusters via the DBSCAN algorithm (density-based spatial clustering of applications with noise) [146]. This allows to group the previously identified stabilisers in clusters of defects (in cases where multiple defects have occurred) while removing any isolated stabiliser that could have flipped above n_{flips} but without being a part of the defective zone. All the hyper-parameters of the detection process (n_{flips}, Δt_{flips}, and the parameters of the DBSCAN algorithm) are chosen offline by the user to maximise the detection performance. For example, typically in my experiments I set $n_{\text{flips}} = 3$ and $\Delta t_{\text{flips}} = 6$, meaning a stabiliser is considered faulty if it flipped more than 3 times within 6 consecutive rounds.

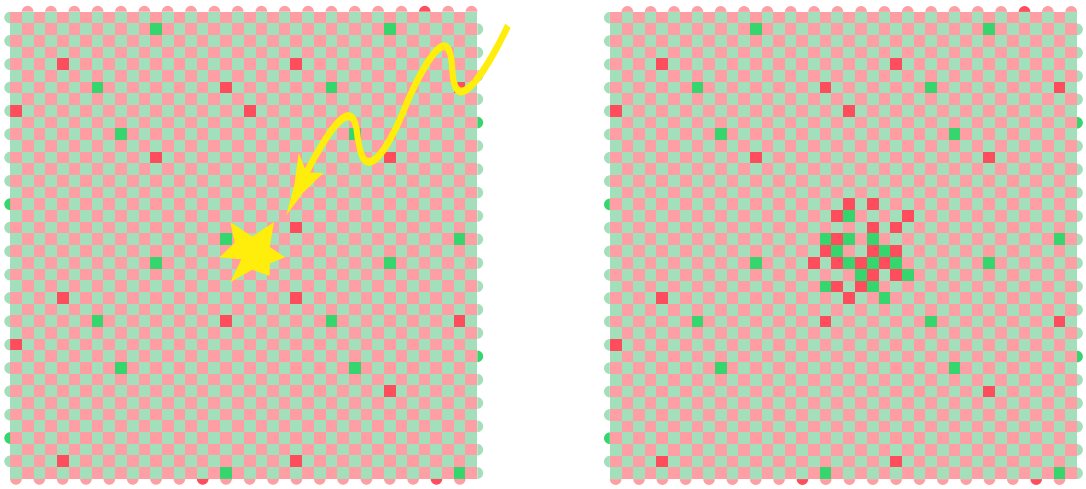


Figure 4.1: Rotated surface code before and after an abnormal event. The stabilisers are located on faces and a brighter color indicates a non-trivial syndrome measurement.

4.2.3 Code deformation

After the defect is detected, the second step consists of excluding the faulty data and ancilla qubits from the surface code. This is done by code deformation and two different pictures must be distinguished: defects located inside the code and defects touching the boundary of the code. For simplicity, I will consider the case of defects bounded by square regions in this chapter, although the concepts generalise to any shape.

Defects located inside the code This case was first reviewed in [143–145] and uses the concept of gauge operators [84, 147] and superstabiliser detailed in Section 4.2.1. When puncturing a hole in a surface code, operators around the hole cannot be used as stabilisers as they do not commute with each other anymore. Rather, a new stabiliser can be evaluated to avoid errors strings terminating at the hole without being detected: the superstabiliser. Its value is inferred from the individual value of each gauge operator. However, as X- and Z-basis operators do not commute (only their product does), they have to be evaluated at different times (see left panel of Figure 4.2). This is equivalent to switching to a subsystem code, as explained in Section 3.2.3. For example, when removing the central data qubit of a [5,1,2] surface code, one obtains a [4,1,2] subsystem code, where the superstabiliser is obtained

from the product of, say, the two X gauge operators. In this first approach, the two bases are measured on alternating rounds. Consequently there is no context allowing one to validate the individual measurements. Consider the measurement of a given X-basis operator; immediately after, the qubits involved will be measured according to Z-basis operator(s). When the X-basis operator is next measured, its value will be independent of its prior outcome due to the non-commutation of the operators – thus a faulty measurement has no signature or 'evidence' on the measurement outcome of an individual gauge operator. Instead, a faulty measurement changes the superstabiliser outcome that the gauge operator belongs to. In consequence, the inferred value of each superstabiliser becomes less reliable as the size of the puncture is increased, since more and more potentially-faulty measurements are involved in the inference of the superstabiliser value. Nevertheless, the approach is attractively straightforward. I will call this the *basic approach*.

In order to mitigate the increase of faulty superstabiliser measurements, it was recently proposed [132] to keep measuring the gauge operators for a number of consecutive rounds that scales with the size of the puncture, effectively creating alternating blocks of repeated measurements of X (and then, Z) gauge operators in spacetime, called *shells*. For instance, one could measure X gauge operators around a defect for, say, three rounds, then switch to measuring Z gauges for three rounds, and so forth. Normal stabilisers (away from a defect) are still measured at every round. In the absence of errors, the successive gauge operator measurements in the same basis will yield the same result; thus, errors on gauge operators become easily detectable simply by repeating and comparing the operator measurements. I call this the *shell approach*. It was proved to outperform the basic approach in the asymptotic limit, but finite code sizes were not studied. I report the conclusions of my numerical modeling of such scenarios in Section 4.3.

Defects located at the boundary The previous ideas do not apply to defects touching the boundary. Let me call *X-boundary* a boundary where X error strings can terminate without being detected (top and bottom boundary in Figure 4.2)

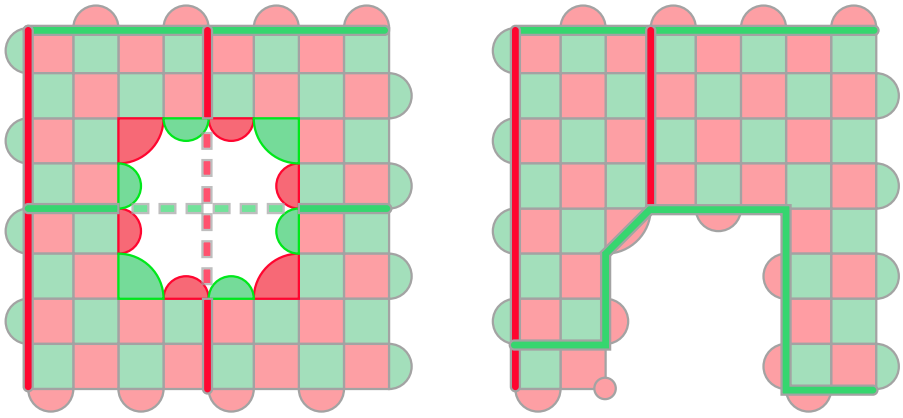


Figure 4.2: Surface code after defect identification and code deformation. Two cases are represented, depending on the defect's location with respect to the code. X (resp. Z) stabilisers and gauge operators are represented in red (resp. green). The gauge operators forming a superstabiliser are in brighter colors (left panel). Some logical operators are drawn with horizontal and vertical lines. Two of them are split in two halves connected by a dashed line. Note that in each case the code distance is $d = L - l$ where L and l are respectively the sizes of the code and the defect.

– and similarly for a *Z-boundary*. Along an X- (resp. Z-) boundary, the product of the Z (resp. X) gauge operators does not commute with some of the X (resp. Z) gauge operators. Hence, they cannot form a superstabiliser and need not be measured. After the deformation, the code just looks like a normal surface code with a deformed boundary (see Figure 4.2). Along an X-boundary, the Z gauge operators are not measured, which means that the X stabilisers of the initial code do not lose their commutation properties. This is why the distance associated with the logical Z operator is not lowered when an X-boundary is deformed. The same applies for the X distance and a Z-boundary. Note that when a defect hits a corner of the code, one can choose which boundary to deform.

4.2.4 Syndrome calculation and decoding

Once the code deformation has been determined, the quantum computation can proceed fault-tolerantly. The ongoing process consists of gathering a suitable set of stabiliser and gauge operator measurements after a period of computation, finding the relevant syndrome and decoding it. In the numerical modelling I presently describe, I simply decode using the entire syndrome *i.e.* without methods such as

parallel window decoding [148] (since I do not simulate codes so large, nor sequences of stabiliser cycles so long, that this is challenging for the decoder). Regarding the stabiliser and gauge operator measurements, they are obtained in the usual way as endpoints of error strings — excluding the superstabiliser measurements for now. This syndrome can then be input to any standard decoder. In this chapter, I use Minimum Weight Perfect Matching (MWPM) [7]. The weights in the matching graph are given by the shortest distance between two points in spacetime.

The only subtlety compared to the canonical surface code picture is how I handled the gauge operators and code deformation. Namely, at the time of such changes, the measurement outcomes of the gauge operators are randomised (recall that only the value of the superstabiliser is constant), because of their non-commutation. Let me describe how this is dealt with in each of the three cases I will consider in the rest of this chapter:

1. the basic static approach, where a defect is preexisting the computation thus the computation is started with the code already deformed, and gauge operators are measured on alternating rounds;

2. the shell static approach, where the computation is also started with the code already deformed, but gauge operators are repeatedly measured (in shells) before switching to the other type;

3. the adaptive approach, where the code deformation happens while the computation is running, and subsequent gauge operators are measured in shells.

Basic static approach. This is the simplest case. As gauge operators are measured on alternating rounds, their values are independent from one round to another. Hence, only the superstabilisers values are used, as the individual gauge operator measurements do not provide any useful information. The decoding is then the same as for a normal surface code, except that the X and Z superstabilisers replace the conventional stabilisers in the matching graph. However, since a superstabiliser is made of multiple gauge operators, its measurement is more likely

to be faulty than a normal stabiliser. I account for this in the decoder by setting the superstabiliser time-like edge weight to:

$$w_{\text{superstab}} = \log\left(\frac{1 - p_{\text{superstab}}}{p_{\text{superstab}}}\right) \tag{4.1}$$

where $p_{\text{superstab}}$ is the probability that the superstabiliser measurement is faulty. Assuming independent errors, this probability can be computed from the measurement error rate p as:

$$p_{\text{superstab}} = 1 - (1 - p)^{n_{\text{gauge}}} \tag{4.2}$$

where n_{gauge} is the number of gauge operators forming the superstabiliser. This weight is then used in the matching graph together with the usual space-like and time-like weights of the surface code.

Shell static approach. Here, the gauge operators are measured repeatedly for a number of rounds before switching to the other type. Within a shell, the difference syndrome can be computed from the individual gauge operator measurements as in a normal surface code, since their values are correlated from one round to another. At the time of a gauge change however, the gauge operator values are randomised, so no information can be extracted from their individual measurements (only their product is conserved). The approach I took is still to use the difference between these values and the previous gauge measurements in the matching graph. Weight-zero edges are however set between all gauge operators measured just after the gauge change, as shown in Figure 4.3. This way, MWPM can link them together with no cost, effectively ignoring their randomness. An equivalent approach would be to remove these gauge operators from the syndrome and only use the superstabiliser value at the time of the gauge change. However, by keeping them in the syndrome with weight-zero edges, the matching graph remains unchanged throughout the computation, which simplified my implementation of the decoder. In terms of output, both approaches are strictly equivalent as weight-zero edges will prioritise a matching between gauge operators, as if they formed a single vertex

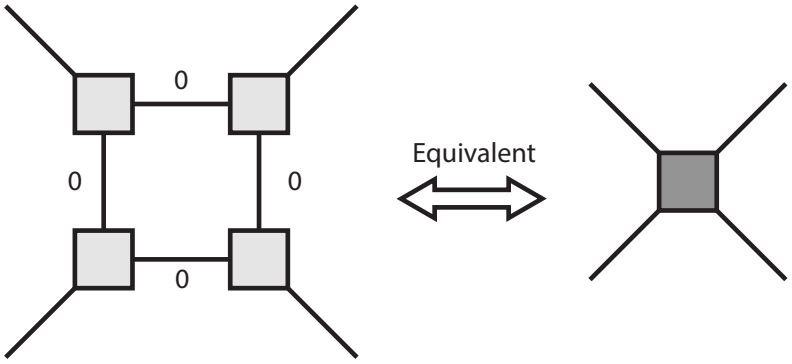


Figure 4.3: Two possible representations of the matching graph at the time of a gauge change in the shell approach. Left panel: gauge operators measured just after the gauge change are kept in the syndrome with weight-zero edges between them. Right panel: these gauge operators are removed from the syndrome and replaced by the superstabiliser value. Both approaches are equivalent in terms of output as weight-zero edges will prioritise a matching between gauge operators, as if they formed a single vertex in the matching graph. Outside the time of a gauge change, all gauge operators must be used in the matching graph in both cases.

in the matching graph (see Figure 4.3). The latter option is exactly what would be obtained with a detector-based decoder [147, 149] where no randomisation nor weight-zero edges are needed.

Adaptive approach. Here, the code deformation happens while the computation is running, namely we switch from the canonical surface code to a punctured one protected by the shell protocol. Except at the time of the code deformation, the syndrome calculation and decoding are the same as in the shell static approach. The main subtlety thus arises at the time of the code deformation itself. [123] proposes a general decoder for gauge changes but it supposes that the whole round of stabiliser measurements before the code deformation is perfect. This allows one to decode the syndrome before the gauge change, then change gauges, then keep the computation going in the new gauge and decode again at the end. This hypothesis is however not suitable to my study as the code deformation happens exactly when the highest amount of correlated errors is occurring. To circumvent this issue, one can note that, as the data qubits inside the defect are to be discarded after the code deformation, they can be measured out in the $(|0\rangle, |1\rangle)$ basis. By doing so one can infer the parity of a given pre-deformation Z stabiliser from the measurement of the

corresponding lower-weight gauge operator and the data qubits measurements. For instance, if a pre-deformation Z stabiliser acts on qubits q_1 to q_4 and q_1 and q_2 are deemed as defective, then these two are measured out. This results in converting the original stabiliser into a gauge operator that only acts on q_3 and q_4. The data qubits measurement outcomes are multiplied to the gauge operator's value before computing the difference syndrome with the original stabiliser. Once the Z gauge operators have been repeatedly measured (remember that the shell approach is being used now), one can switch to the X gauge operators (whose measurements are randomised). The process is now the same as for the static shell approach.

4.3 Results

To verify the performance of my method, I proceeded in several steps. First, I compared the basic and the shell approaches (as defined in Section 4.2.3) and explored which approach had better performance at a given finite scale. This allowed me to decide which one should be used in the adaptive method. With this knowledge, I was then indeed able to implement the adaptive method; as previously explained, I distinguish two types of defects: permanent defects identified before the code is run, solved by a so-called *static approach*, and defects happening while the code is running and that one must detect, tackled by the *adaptive approach*.

In all of the following simulations, I focus on X errors only – this is possible since the surface code can be regarded as a means to protect against Z and X errors as two separate (yet interlaced) tasks; for any homogeneous error model, the performance with respect to Z errors will be identical to the performance versus X errors. I adopt a simple phenomenological error model where errors are independent and identically distributed, and afflict data and ancilla qubits with the same probability p at each time step. When a defect occurs during the computation, it manifests as an abrupt increase in the phenomenological error rate to $q = 0.5$ for the affected qubit(s). For simplicity, I assume that these qubits form a square of side l centred at a random location of the code. Not more that one defect will be simulated in this chapter due to the increasing complexity of the decoder in

the presence of multiple defects. Finally, each data point in the following plots is obtained from Monte-Carlo simulations with a number of runs ranging between 10,000 and 100,000 (depending on the expected logical error rate). Error bars, when plotted, show the variance of the sampled data.

4.3.1 Basic vs. shell

An important parameter in the shell approach compared to the basic one is the number of measurements per shell n_{shell}, *i.e.* the number of rounds for which a type of gauge operator is measured before changing gauges and measuring the other type. Figure 4.4 shows how the logical error rate evolves with this parameter and for various defect sizes. The logical error of the basic approach is plotted in dashed lines for comparison. One can notice that when the size of the defect is increased, the shell method starts to outperform the basic one for the right choice of n_{shell}. There is an optimal choice of this parameter according to the likely defect size. A too-high rate of gauge switching does not give enough time to infer the value of the superstabiliser accurately. A rate set too low results in, for example, a poor capability to temporally localise the end of a chain of phase flip errors that terminates on the boundary while Z-gauge operators are being measured.

The data shown in Fig. 4.4 allows one to deduce the best-performing method and the optimal set of parameters: it is favourable to choose the shell approach for defect sizes $l \geq 3$ with a number of measurements per shell scaling with l. I thus set $n_{\text{shell}} = l$ in all the subsequent simulations. For this choice of parameters, I show that the difference between the shell and the basic approaches scales with the defect size, as one would expect (Figure 4.5).

4.3.2 Qubit overhead for quantum error correction on a defective lattice

To compensate for the loss of protective power due to the presence of defects, it will be necessary to increase the number of qubits in the code. However, for the presented methods to be of practical use, this overhead must be kept as low as

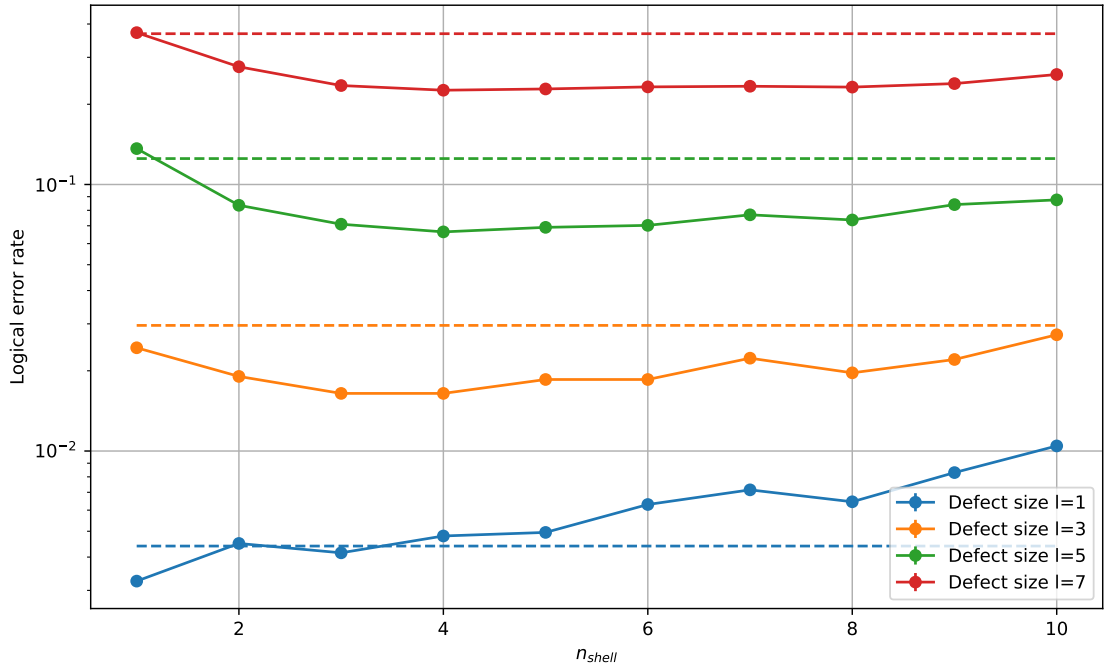


Figure 4.4: Comparison of the performance of the shell (solid lines) and basic (dashed lines) approaches over the number of measurements per shell n_{shell} and for various defect sizes l at $L = 15$ and $p = 1.5\%$.

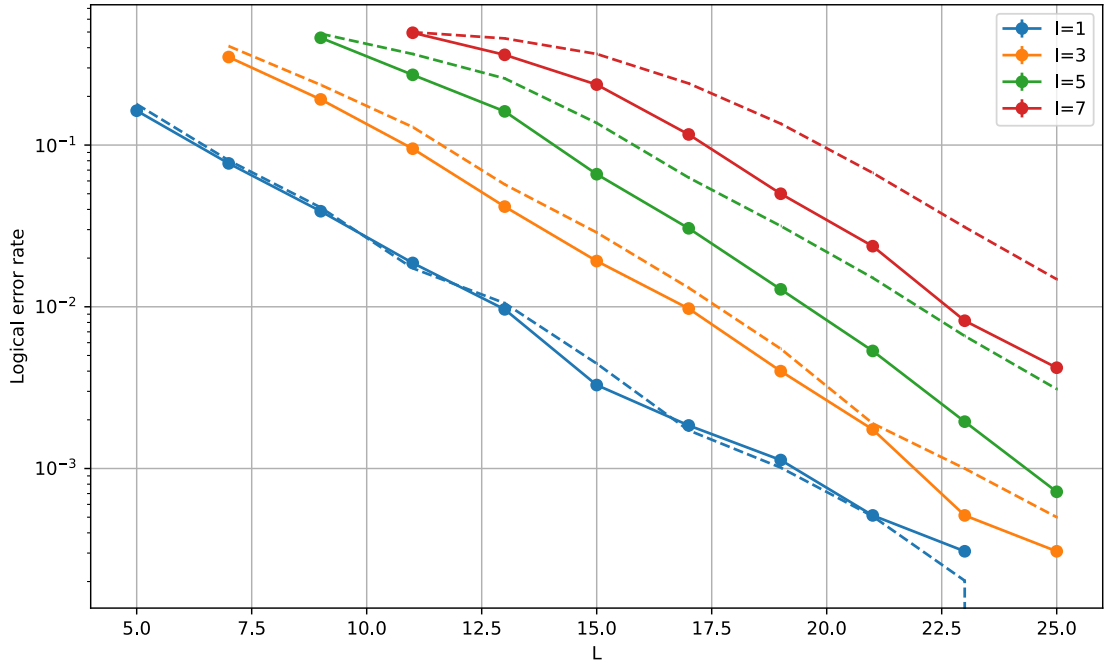


Figure 4.5: Comparison of the performance of the shell (solid lines) and basic (dashed lines) approaches over the size of the code L and for various defect sizes l at $p = 1.5\%$.

possible: it is estimated in Fig. 4.6, where the logical error rates of the different approaches presented in this chapter are compared to the logical error rate achieved by different sizes of regular surface code. The best form of mitigation strategy, as determined by the prior results, is applied in each case according to the defect size.

One can first observe that the performances of the static and adaptive approaches are very close, showing the efficiency of the latter to detect defects in real time. The divergence for large L stems from imperfect defect detection, specifically setting up larger shells than necessary around defects — if a qubit far away from the defect happens to 'flicker', due to random noise, it may be included within the inferred bounds of the defect, hence creating a significantly larger shell than necessary. Note that this is an artefact of the present simulations where at most one shell is employed; moreover the effect could certainly be mitigated by the use of more powerful detection and inference methods that are beyond the scope of this research.

A second observation is that the adaptive surface code still provides an exponential suppression of errors; this was not an obvious outcome given the existence of a finite period of highly correlated errors between the appearance of the defect and its detection and subsequent code deformation. Third, one can note that for any approach, the increase of code size that is necessary to equal the performance of an ideal surface code is of the order of the defect size, which is expected since it is the amount by which the code distance is lowered.

4.3.3 Performance of the adaptive surface code

The final step is to specifically analyse the performance of the adaptive approach: the capability to detect defects in real time via the DBSCAN algorithm, exclude them from the code, and keep the computation going. The logical error rate is plotted against the physical error rate p in Fig. 4.7 for different code sizes L. For any value of p, large codes appear to perform better than small codes: this is because of the additional burst of errors due to the defect to which small codes are less resilient.

However, in order to examine the existence of a threshold, one must incorporate an additional crucial factor: smaller codes are also less likely to suffer a defect, given

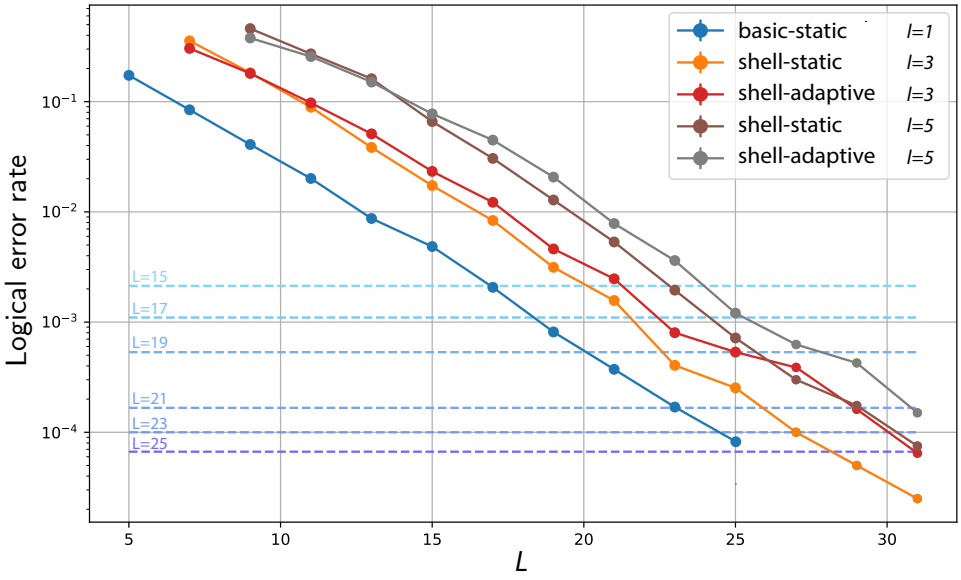


Figure 4.6: Logical error rate over L for three different defect sizes, 1×1, 3×3 and 5×5, with the high-performing protocols for each scenario selected (*i.e.* simple superstabiliser for the smallest defect, and the static or adaptive shell approaches otherwise). The dashed lines respectively correspond to the logical error rate achieved by a regular surface code of size 15, 17, 19, 21, 23 and 25. In all cases, qubits suffer 'normal' phenomenological noise at 1.5%. The intersection of a dashed line with a solid line gives the enlarged code size L needed to replicate the performance of the smaller code on a defect-free device.

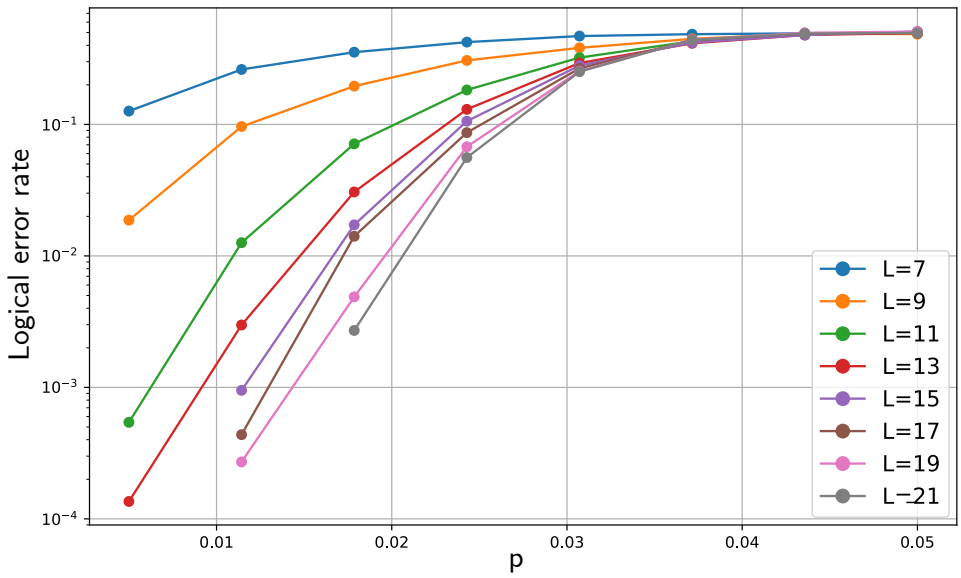


Figure 4.7: Logical error rate for the adaptive surface code under phenomenological noise for a defect of size 3×3 (ignoring that larger codes are more likely to be defective).

their smaller number of qubits. Over the course of a long quantum computation, the code may be hit by some defects, recover from them so that any required code deformation can be reversed, and subsequently suffer further defect events. Without any further assumption on the defects' nature, it is most reasonable to assume they occur uniformly in space and time — which is certainly the case for specific defects too, *e.g.* cosmic rays. As a result, I will define a constant rate of defects ρ per unit space and time, or equivalently, per qubit and per round of stabilisers. If supposing that defects survive in the code for T rounds, then the effective logical error over the whole computation will be given by:

$$p_{\log} = \sum_{k\geq 0} \langle t_k \rangle \times p(\text{logical error} | n_{\text{def}} = k) \tag{4.3}$$

$$\langle t_k \rangle = \frac{\mathrm{e}^{-2L^2\rho T}(2L^2\rho T)^k}{k!} \tag{4.4}$$

with $\langle t_k \rangle$ the proportion of the time spent suffering k defects and n_{def} the number of defects in the code. This simple probability model assumes that defects are uniformly distributed in time and mutually independent. In particular, it supposes that they can overlap. See details of Equations 4.3 and 4.4 in Appendix A.1.

As I simulate at most one defect in this chapter, the conditional probabilities are evaluated with a simple heuristics deduced from the previous subsection: for $k = 0$, a simple threshold calculation is performed for a rotated surface code without defects; for $k = 1$, the data from Fig. 4.7 is used; and for $k \geq 2$, I estimate from Figure 4.6 that the code distance is lowered by at most $2l$ in the presence of a defect of size l. Focusing on defects of a single size only, I deduce the following rule: the probability of a logical error in the presence of k defects of size l in a surface code of size L is given by the probability of a logical error in the presence of 1 defect of size l in a surface code of size $L - 2l(k-1)$. This assumption is more pessimistic than reality as defects overlapping actually involve fewer faulty qubits.

Figure 4.8 is obtained from this approach and manifests the existence of a threshold. The faded lines correspond to no defect ($\rho = 0$) and the opaque lines to a rate of defect nucleation, per qubit and per stabiliser round, of $\rho = 10^{-5}$.

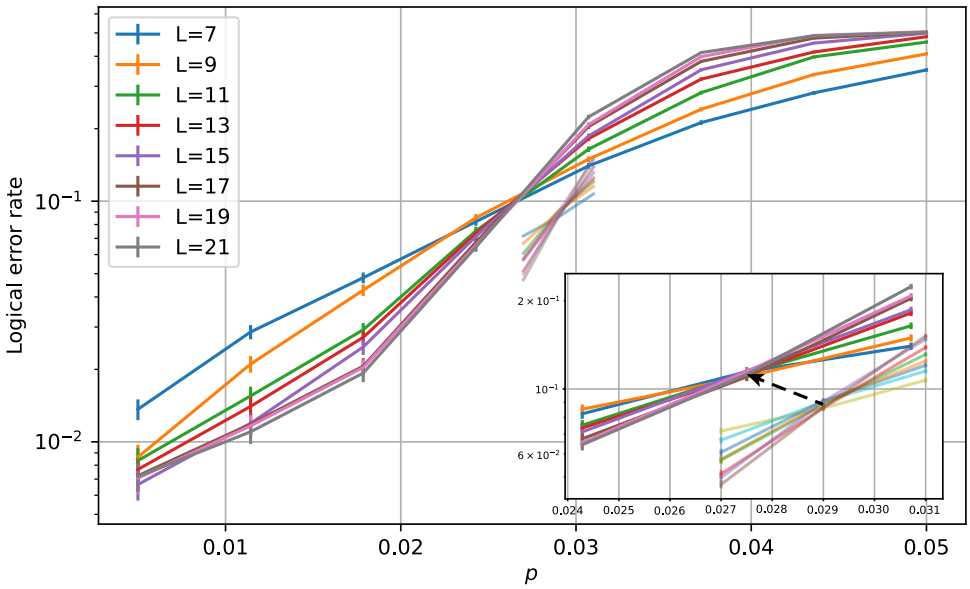


Figure 4.8: Opaque lines: threshold plots for the adaptive surface code under phenomenological noise for $\rho = 10^{-5}$ and $T = 100$ (see main text). Faded lines: threshold for the non-defective rotated surface code. Inset: zoom around the threshold region. The dashed arrow represents the evolution of the threshold when the rate of defects is increased.

For logical qubits of size 21×21, this is therefore one defect occurring every 110 stabiliser rounds, or around 8800 defects per second afflicting each logical qubit in a quantum computer with a 1MHz stabiliser cycle. The defect survival time T is set to 100, meaning that, each time a defect occurs, it takes 100 stabiliser cycles to reset and reintegrate the afflicted physical qubits. This means spending 37% of the total time with one defect, 16% with two defects and 5% with three: the values of ρ and T are hence not particularly conservative. For this value of T, the chosen ρ is also approximately the highest value one can pick for my heuristic model to apply: as logical error rates are evaluated with a relatively pessimistic rule for two defects or more (lowering the code size by twice the size of the defect), scenarios where more than three defects occur must remain quite rare.

In the inset of Fig. 4.8, the black dashed arrow shows the evolution of the threshold when ρ is increased (a more detailed plot can be found in Appendix A.2). It is worth noting that it ranges between 2.7% and 2.9% — the threshold of

the non-defective rotated surface code: in the presence of defects, the threshold is lowered but only by a small amount, which confirms the efficiency of my method, even under fairly pessimistic assumptions.

4.4 Discussion

I have thus established a procedure to deal with defects in the surface code, whether they are present and detected from the start, like fabrication defects, or occurring while the code is running, like cosmic ray impacts. This work is hence relevant to recent experimental studies that identified how damaging this latter kind of defect can be [135, 141]. The previous section has shown the performance of my method, in particular with the existence of a threshold that is only slightly lower than that of the non-defective surface code. A resource analysis also showed that the qubit overhead needed to overcome the presence of a defect is moderate, only scaling with the number of defective qubits.

However, my numerical simulations were performed under rather simplistic assumptions: phenomenological noise model for the lattice and the defect, defects are square, only one defect at a time can affect the code. These were only assumed out of simplicity for the numerical implementation, hence I do not expect that dropping these hypotheses would severely change my results. It would however be valuable to run further simulations relaxing them. In particular, the value of the defect rate ρ was chosen to be the highest my model could tolerate, by guaranteeing three defects or more remained relatively rare events. Further work could explore higher defect rates by properly implementing a decoder for multiple defects, without having to make use of the heuristic rule established in the previous section.

Moreover, the decoding algorithm used in this chapter is MWPM. While this is the most canonical choice for studies of the surface code, research is today ongoing to find potentially better decoders [114, 150], *i.e.* that would run faster, be scalable to very large codes and work as locally as possible to avoid memory and bandwidth issues between the quantum device and the classical processor

responsible for decoding [105, 151, 152]. The present work could thus be improved by taking these considerations into account and using an efficient local decoder that would be able to deal with defects.

Finally, since the time this research was undertaken (2022), the problem of error correction in the presence of defects has attracted a lot of attention, with multiple improvements showing that defects should not be a bottleneck in the implementation of fault-tolerant quantum computers in the long run. For instance, Ref. [153] reduces the impact of defects by minimising the necessary size of a puncture around a defect. In contrast, Ref. [154] takes a more radical approach and uses mid-cycle stabilisers to alternate the role of ancilla and data qubits. This allows one to periodically measure out all the qubits in the code (rather than just the ancillas in the conventional picture), thereby reducing the entropy increase generated by a potential defect.

5

Towards early fault tolerance on a 2×N array of qubits equipped with shuttling

The research presented in this chapter (up to the last section) was conducted jointly by Armands Strikis, Michael Fogarty and myself and is based on [155]. The idea that a 2×N array of qubits equipped with shuttling could be used to generate a variety of quantum codes, as well as the associated framework, came from Armands. I designed the precise protocols for universal quantum computation on the surface code, throughout useful brainstorming sessions with Michael. The noise modelling and subsequent numerical simulations were all generated by myself.

The last section of this chapter concerns a related research paper from Zhenyu Cai, Simon Benjamin and myself [156]. All fundamental ideas came from Cai and Simon. My contribution lay in the design of the noise model and subsequent threshold simulations of the system they advocated for.

All writing below is my own, in places using text verbatim from the manuscripts.

Contents

5.1 Introduction

In the previous chapter, I looked into a later-stage error correction problem: how to tackle rare but large-scale defects. The perspective I took in that chapter was quite abstract as it did not assume anything about the underlying physics of qubits composing the code. Furthermore, the protocols I designed should not be relevant before error correcting codes reach a more mature stage where these rare defects become limiting. In the present chapter, I take a step back and focus on a practical problem of current high relevance: simplifying the requirements of QEC codes at the device level.

As teams around the world are starting to build the very first error correcting codes, one obstacle they are facing lies in the complexity of building architectures requiring the entanglement of a very large number of qubits and the repeated measurement of a considerable number of stabilisers. To facilitate their physical

implementation, practical constraints are often integrated in the code design, such as locality of the interactions and planar structure of the code. The main example of error correcting code respecting these constraints is the surface code [74, 76], and its high threshold — that is the maximum error rate the code can tolerate to exponentially reduce errors — makes it one of the best candidates for achieving fault tolerance. However, more general quantum low-density parity-check codes have been constructed and demonstrate better performance in finite-size simulations [95, 112, 157–160]. Although these codes seem more challenging to realise experimentally due to their long-range interactions, they have the potential to considerably lower the qubit overhead due to their higher rate, *i.e.* the number of encoded logical qubits per physical qubit, and better distance scaling. Consequently, further research has been undertaken to demonstrate the experimental viability of such complex codes, and suggested solutions for the implementation of their long-range connections, either by swapping [161] or displacing the qubits [93, 156, 160].

It may seem natural to assume the use of fully 2D architectures to embed such codes, possibly with augmentations to enable some longer-range interactions (as in, e.g., [112]). However, such complex devices might not be available for some time; their first fault tolerant iteration may be more limited. For example, the size of an array realised on a quantum chip might be constrained in one direction. This motivated research towards the feasibility of 1D or quasi-1D error correction [162, 163]. Implementing 2D codes in this kind of devices can theoretically be solved by extending the length of the interactions, thereby increasing the non-locality of the operations. Albeit challenging, this can however be tackled in the same way as long-range interactions in the aforementioned qLDPC codes, *i.e.* by swapping or displacing the qubits.

In this chapter, I study the feasibility of quantum error correction in the most constrained experimental setup beyond 1D: a 2×N array of qubits. Long-range interactions are implemented by assuming that the qubits can be collectively shuttled along one of the two lines of the array. Several qubit platforms have been shown to be suitable to implement such an operation with high fidelity,

including atom arrays [93] or ion traps [164], but spins qubits [36] may be the most promising one for also implementing the proposed 2×N array. First, I detail a precise framework established by my co-author Armands Strikis to determine the classes of codes that can be efficiently implemented in our device. This study is then applied to two specific examples: the surface code and higher-rate qLDPC codes. Regarding the surface code, I demonstrate that universal quantum computation is possible in a practical setup based on silicon spin qubits, despite the additional experimental constraints that the geometry entails. With the help of numerical simulations accounting for additional shuttling errors, I estimate a resource cost that is moderate enough to run meaningful quantum algorithms in the classically intractable regime. As for higher-rate qLDPC codes, I evaluate their performance under the noise processes that are anticipated for relevant devices, and demonstrate that despite their complexity they do give an advantage over the surface code in a noise domain that is practically achievable.

For simplicity this analysis assumes a strictly 2×N array. In practice, given the long device length that would be required for meaningful post-classical tasks, it is reasonable to assume that a real device could incorporate a plurality of junctions – thus the overall geometry would be a lattice formed of long 2×N sections. The processes I analyse in this chapter would then occur along each long section, while the overall algorithm would benefit from the additional routing advantages of the lattice. I investigated this possibility in a recent research paper that was produced outside the scope of my PhD [165]. It will be further described in the Discussion section of this chapter.

5.2 Framework for finding codes within device restrictions

In this section, I present the framework introduced by my co-author Armands Strikis to describe codes that naturally embed in the 2×N architecture.

5.2.1 Architecture

In all the following, we assume that the device is composed of two parallel rails of evenly-spaced qubits, and that nearest-neighbour qubits from separate rails can interact through operations such as controlled-Z or controlled-X gates. Further interactions could evidently be supported, *e.g.* nearest-neighbour interactions between same-rail qubits, but these would induce an increased manufacturing complexity that will not be required.

In order to increase the connectivity of the device, we make use of a key additional ingredient: shuttling. More specifically, we assume that qubits from the top rail can collectively and synchronously be shuttled with respect to the bottom rail, in one direction or the other. After such shuttling event, different pairs of qubits will be facing each other and will be able to interact, thereby increasing the connectivity of the device. Note that shuttling the top row is sufficient as only the respective position of the top qubits compared to the bottom qubits matters. With the above assumptions, the generated connectivity graph of the device is thus bipartite as qubits from the top rail can only interact with qubits from the bottom rail. Moreover, any interaction can be implemented between the top and the bottom rows if accepting potentially long shuttles.

This piece of knowledge informs the placement of the data and ancilla qubits in the device. In a quantum error correcting code, data qubits need not interact: the stabilisers can always be measured by entangling the ancilla qubits to the data qubits. It is therefore natural to place all data qubits in, say, the top (moving) rail and the ancilla qubits in the bottom (static) rail. All device characteristics are summarised in Fig. 5.1.

5.2.2 Framework

Given the device description of the previous section, it appears that any quantum error correcting can theoretically be implemented, as *any* data qubit can be shuttled to the site opposite to *any* ancilla qubit, thereby enabling their interaction. It is nonetheless evident that most codes implemented this way would prove particularly

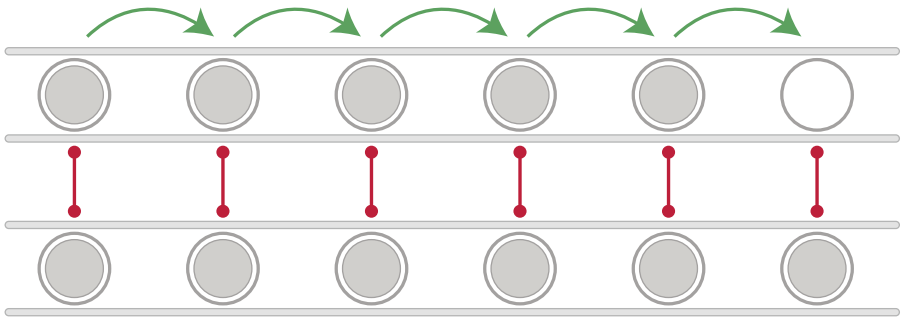


Figure 5.1: Representation of the 2×N architecture. The device is characterised by two parallel rails of evenly-spaced qubits (gray circles with gray disks inside). Adjacent qubits from different rails are allowed to interact via two-qubit gate operations (red vertical lines with disks on their ends). Finally, the qubits of the first row are allowed to shuttle along their rail (green arrows). Some empty locations are kept at the end of the device to leave space for the qubits to shuttle (empty gray circle).

disadvantageous as they might require many shuttles, that could potentially be device-long. As shuttling is an operation that is both finite-time and prone to errors, one would ideally want to minimise the total shuttling distance and the number of shuttles per stabiliser cycle. These two quantities will be the measures of the suitability of a given code to our architecture.

Thus, in order to understand what classes of codes naturally embed in the 2×N architecture, let us write down the biadjacency matrix of the qubit connectivity graph generated after a given sequence of shuttling operations. Let us assume that both rails have the same number N of qubits and that the i-th data qubit faces the i-th ancilla qubit ($i \in [\![1, N]\!]$). The initial biadjacency matrix is thus $H_0 = \mathrm{diag}(1, ..., 1)$. Now suppose that the data qubit row is shuttled by m increments to the right: then the new biadjacency matrix H_m has a diagonal of 1's that is m increments to the left with respect to the main diagonal. The same applies when shuttling data qubits to the left, with a matrix H_{-m} with a diagonal of 1's shifted to the right. For instance, shuttling the data qubits 1 increment to the left leads to the following transformation:

$$H_0 = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{pmatrix} \xrightarrow{shuttle} H_{-1} = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ 0 & 0 & 0 & \dots & 0 \end{pmatrix}$$

The total biadjacency matrix of the code after a sequence of shuttling events is thus given by a sum of the above H_j's. Using the two criteria identified above – number and length of shuttles – lowest-noise circuits are generated from biadjacency matrices with few non-trivial diagonals, that must additionally be located close to the main diagonal. Note that this description does not account for gate ordering for now. Besides, diagonals are not periodic here (the bottom-left element of H_{-1} is not a 1). Finally, diagonals may be incomplete typically if some operations are not required. For instance,

$$H = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \quad \text{and} \quad H' = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

have two and five non-trivial diagonals respectively.

This formalism makes it very convenient to read the complexity of the implementation of a given stabiliser circuit. To do so, it is appropriate to use the parity check matrix formalism described in Chapter 3. Besides, I will exclusively focus on CSS codes here although the discussion could be extended to further code classes. Instead of writing a parity check matrix with $2n$ columns (with n the number of data qubits) as in Eq. 3.2, Armands Strikis decided to use an alternative representation and write the parity check matrix as an $n \times n$ matrix, where each non-trivial row represents either an X or a Z stabiliser. As stabiliser codes typically have less ancilla than data qubits, certain dots of the bottom rail are kept empty, and are represented by a null row in the parity check matrix. With this representation, parity check matrices have the same structure as the previously described biadjacency matrices, where a 1 at row i and column j accounts for an interaction between a data and an ancilla qubits. This gives a necessary condition for a code to be efficiently-implementable on the proposed platform. Indeed, as explained before, suitable codes must be implementable in few short shuttles: the corresponding parity check matrix (or equivalently biadjacency matrix) must therefore contain few diagonals and these must be located close to the main diagonal. Note that the

indexation of the ancilla and data qubits can be reworked (or equivalently rows and columns of the parity check matrix can be swapped) so as to minimise the number of diagonals. See the next section for examples.

The above description however disregards gate ordering, which is a highly important matter in a stabiliser circuit. First X and Z stabilisers are implemented via CZ or CNOT gates that *do not* commute. Second even when respecting commutation rules certain gate orderings are more efficient than others. For instance, in any CSS code, all Z stabilisers can be implemented before all X stabilisers, but this incurs an increase of the stabiliser cycle time and subsequently of idling errors. In our specific architecture, this also increases (at most doubles) the number of shuttles and total shuttling distance, compared to a version of the stabiliser circuit where X and Z stabilisers are interleaved. Finally, the gate ordering impacts the output logical error rate, as a suboptimal ordering choice can be the cause of distance-reducing hook errors. All these mechanisms are thus highly important and to be considered separately when analysing the implementability of a stabiliser circuit. However, for general CSS codes (outside the surface code) the performance will be evaluated using the 'Z then X' gate ordering for simplicity.

In order to find suitable codes, the procedure Armands Strikis and I adopted is thus the following: look for high-performance codes whose parity check matrices either contain a small number of diagonals, or for which all diagonals are located close to the main diagonal. These conditions guarantee that the stabiliser circuit can be implemented either with few shuttles, or that all shuttles will be short. Typically, we consider that short shuttles have a length scaling as $O(\sqrt{n})$ and that a low number of shuttles is $O(1)$. The latter choice was made as a lower bound for the number of shuttles is the weight of the code's stabilisers, which is $O(1)$ for qLDPC codes. As for the shuttling distance, it is related to the locality of the operations of the code: when linearising a 2D code, say, along its rows, nearest-neighbour qubits within the same column become separated by a distance l, where l is the length of the rows and $l = O(\sqrt{n})$. Examples satisfying one of the conditions, or both, are presented in the next section.

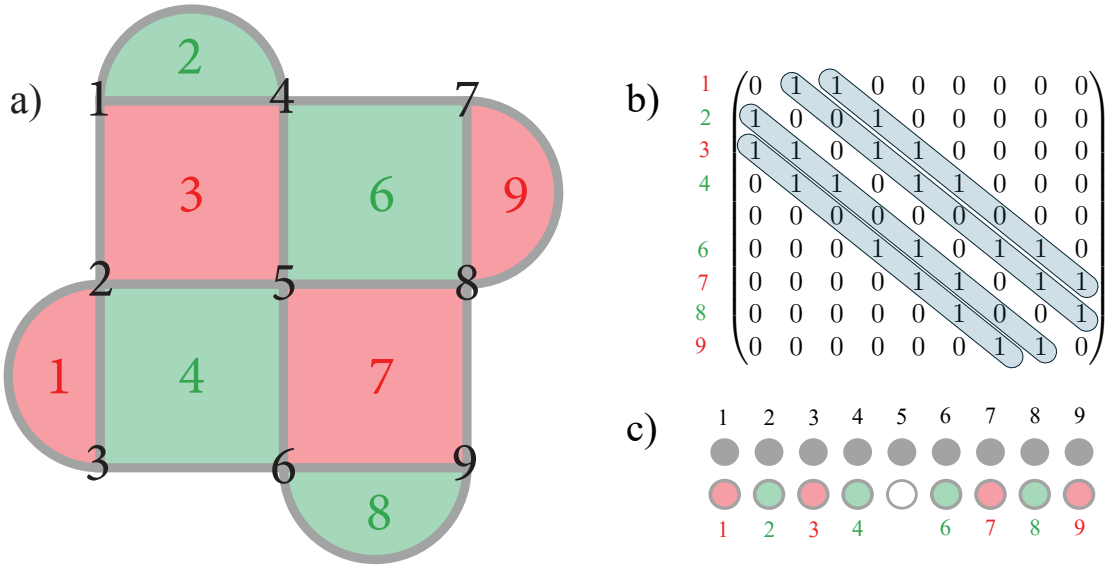


Figure 5.2: Three-way mapping for the rotated surface code. (a) The canonical 2D layout, (b) its parity check matrix with an additional trivial stabiliser and (c) a schematic for the layout on a 2×N architecture. Here, red/green colours indicate X/Z stabilisers respectively.

5.2.3 Code examples

Surface code

The first code my co-author and I proved to be efficiently embeddable in the 2×N architecture is the rotated surface code. As this code uses weight-four stabilisers, the parity check matrix must contain at least four diagonals. In fact, a clever choice of indexation of the stabilisers renders a parity check matrix with exactly four diagonals. A three-way mapping of the 3×3 rotated surface code is presented in Fig. 5.2. Note again that this neglects any gate ordering that should be chosen to minimise stabiliser cycle time and avoid hook errors. I however verified in Appendix B.1 that measuring the diagonals of Fig. 5.2(b) in the order 1-4-2-3 (where the diagonals are indexed from left to right) guarantees a minimum number of shuttles, a minimum shuttling distance and no distance-reducing hook errors. The rotated surface code thus satisfies both conditions of few and short shuttles, with 4 shuttles per stabiliser cycle, whose length scales with $d = \sqrt{n}$.

Hyper-graph product codes

Another interesting code class Armands and I came up with is hypergraph product codes (HGP), whose seed codes are a repetition code and a classical LDPC code. The surface code is a special case of them, where the classical LDPC code is another repetition code.

To understand this, let us lay such codes on a 2D plane and suppose that shuttling is performed in the horizontal direction. Let us additionally assume that the classical LDPC code is placed in the direction of shuttling, and the repetition code in the orthogonal direction. For simplicity, let us also suppose that both codes have distance d. It follows that vertical interactions have length at most 1 in the 2D grid (length of the repetition code's interactions), which translates into length d in the linearised 2×N architecture. Therefore, any ancilla qubit has to interact with data qubits that are at most d increments away, and compared to the surface code, the shuttling distance is not increased. Rather, the only difference with the latter is that the data qubit row has to stop more often, as stabilisers may have higher weights, but also as it may not be possible to synchronise the interactions as much as for the surface code (adopting the framework described before, the code's check matrix has more non-zero diagonals).

Armands formalised this as follows: let A be the $(n-1) \times n$ parity check matrix of a repetition code

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & \dots \\ 0 & 1 & 1 & 0 & \dots \\ 0 & 0 & 1 & 1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

and B to be a sparse $l \times k$ matrix representing some classical LDPC code. Then the HGP code of A and B has parity check matrices

$$H_X = \left(A \otimes \mathbb{I}_k, \mathbb{I}_{n-1} \otimes B^T\right)$$
$$H_Z = \left(\mathbb{I}_n \otimes B, A^T \otimes \mathbb{I}_l\right),$$

where $\mathbb{I}_m$ is an identity matrix of size $m \times m$. By cleverly rearranging the rows and columns of the total parity check matrix H (*i.e.* by the re-indexing the data and ancilla qubits), one can obtain

$$H = \begin{pmatrix} B & \mathbb{I}_l & 0 & 0 & 0 & 0 & 0 & \dots \\ \mathbb{I}_k & B^T & \mathbb{I}_k & 0 & 0 & 0 & 0 & \dots \\ 0 & \mathbb{I}_l & B & \mathbb{I}_l & 0 & 0 & 0 & \dots \\ 0 & 0 & \mathbb{I}_k & B^T & \mathbb{I}_k & 0 & 0 & \dots \\ 0 & 0 & 0 & \mathbb{I}_l & B & \mathbb{I}_l & 0 & \dots \\ 0 & 0 & 0 & 0 & \mathbb{I}_k & B^T & \mathbb{I}_k & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix},$$

whose non-trivial diagonals are at most $O(\sqrt{n})$ data qubits apart from the main diagonal. The number of diagonals however is suboptimal and scales as $O(\sqrt{n})$. Therefore, implementing such a HGP code comes at the cost of increased stabiliser cycle time and increased idling noise (but not increased shuttling noise). This additional noise can be compensated for if a *good* classical code is chosen for the product with the repetition code, as a higher-rate quantum code will be generated, thereby reducing the qubit overhead.

Generalised bicycle codes

Finally, one may naturally try to consider codes satisfying the opposite condition compared to HGP codes, *i.e.* with stabilisers implemented in a few shuttles but potentially long-distance. One code family that Armands identified to satisfy these criteria is generalised bicycle codes, which I described in Section 3.3.2. These codes are characterised by X and Z check matrices made from circulant matrices, *i.e.* of the form

$$C = \begin{pmatrix} a_0 & a_{l-1} & a_{l-2} & \dots & a_1 \\ a_1 & a_0 & a_{l-1} & \dots & a_2 \\ a_2 & a_1 & a_0 & \dots & a_3 \\ a_3 & a_2 & a_1 & \dots & a_4 \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix},$$

where $a_i \in \mathbb{F}_2$. These codes are thus particularly suitable as the number of diagonals is directly given by the number of non-trivial a_i's. Sparse circulant matrices hence guarantee a low number of shuttles. Besides, [95] showed that codes constructed

from such sparse circulant matrices were particularly powerful, making them a natural candidate for our proposed scheme. The length of the shuttles is however not restricted and may scale with n.

5.3 Universal quantum computation on a spin-qubit surface code

The previous section discussed how to embed an individual code block on a 2×N device by taking advantage of shuttling. This is only the first step towards the implementation of a fully functional quantum computer, as we still need to discuss how operations between logical qubits can be implemented despite the strong architectural constraints. In this section, I will demonstrate that universal quantum computation is indeed possible when embedding surface-code-like error correcting codes in a $2 \times N$ device equipped with shuttling. In order to ensure that our proposition is practical from the experimental point of view, I will focus on a silicon spin qubit device. Shuttling has been demonstrated to be implementable at high fidelity on such a platform [34, 35]. This motivates the introduction of additional constraints discussed below.

5.3.1 Silicon spin qubits

As detailed in Chapter 2, electron spins in silicon quantum dots (QDs) are a promising physical platform to perform quantum computing. While silicon architectures are expected to eventually provide dense two-dimensional grids of qubits [52, 53], early silicon quantum processor designs are particularly interesting since they could meet the requirements for the implementation of the protocol described in this chapter. These are (*i*) a bilinear qubit topology and (*ii*) information shuttling. Both these criteria have been met experimentally: devices with 2×N arrays of QDs have been demonstrated [166]; and information transfer has been achieved using spin shuttling techniques in the form of bucket brigade [34] or conveyor belt approaches [35]. As explained in Section 2.2.5, I will solely focus on the conveyor belt approach in this thesis, as it naturally allows for the collective and synchronous

shuttling of the electrons, which is the desired mode of shuttling here. Besides, this approach requires a moderate engineering complexity as only three or four distinct voltages must be sent to generate the moving potential.

There is of course a natural set of gates that one can directly implement with a class of silicon spin qubit device (here we assume single-spin representations of qubits). More precisely, local phase rotations [167] and two-qubit operations such as C-Phase gates [168] have been shown. However, general single-qubit rotations can be more challenging to localise, and may be more naturally realised on a wide scale, via the interaction of the spins with a global oscillatory field [28]. Alternative solutions for improved qubit addressability will be presented in Chapter 6.

For the present analysis, we will require only a modest refinement of unconditionally-global Hadamard gates. It will suffice to apply the operation to the entirety of one of the two linear arrays of the processor. This partial global, or 'semi-global' operation could be implemented in silicon devices using frequency engineering, for example by placing a micromagnet next to the array [169] or by using magnetic materials in the gate stack on one side of the array [170]. Such approaches could create a frequency difference between the two linear arrays that can be selectively addressed with known electron spin resonance techniques [171]. A schematic implementation of such frequency difference between the two lines of the array is pictured in Fig. 5.3, via the use of a micromagnet. Operating at a global magnetic field $B_0 = 1.4$T, one can expect the g-factor dispersion in the device to be around $\Delta f = 60$MHz [172]. In order to sufficiently separate the frequencies of both lines of the array, one could therefore set the micromagnet-induced magnetic field difference to $\Delta B = 5h\Delta f/g\mu_B = 10$mT. Here h is Planck's constant, $g = 2$ is the electron's Landé factor and μ_B is Bohr's magneton. Given a spacing of 100nm between the lines, this would mean a gradient of 0.1mT/nm, which is comfortably below demonstrated gradients, around 0.8mT/nm [173].

In our setup, the first and second one-dimensional arrays contain the data and ancilla qubits respectively, and the processor is subject to a magnetic field to polarise the spins. Each data qubit is encoded by a single quantum dot, initialised

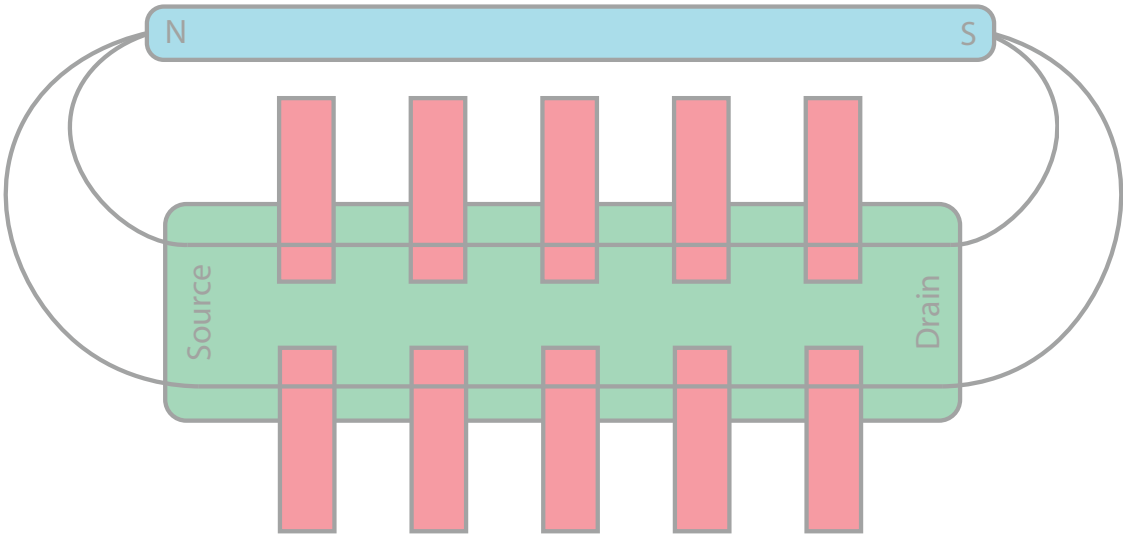


Figure 5.3: Schematic representation of a device where the g-factors in the first row are all higher than the g-factors in the second row. The green rectangle represents the source and drain, while the red and blue ones respectively represent the gates confining the electrons and the micromagnet. Two field lines are additionally drawn, showing the difference in magnetic field between both rows of the 2×N array. This is in turn responsible for a difference in Zeeman splitting, which can be utilised to selectively target either the spins of the first row or those of the second row.

in the $|0\rangle$ state (spin down) at the beginning of the computation. On the contrary, each ancilla qubit is encoded by two electrons in a singlet or triplet state. This is consistent with established techniques for preparation and measurement: at the start of each stabiliser cycle, the ancilla qubits are initialised in the singlet state by first applying a differential gate potential between QDs to produce an energy detuning that will relax the system in the singlet (02) [or (20)] state, followed by a ramp to the singlet (11) state (adiabatic with respect to the singlet anticrossing and diabatic with respect to the singlet-triplet anticrossing). At the end of the cycle, the stabiliser is measured by projecting the pair of spins to one of the QDs which, through Pauli spin blockade, reveals a different measurement outcome for singlet or triplet states [174, 175]. This is similar to the picture described in Section 2.2.2 but with singlet-triplet qubits instead of single-spin qubits. Finally, note that this choice of singlet-triplets requires the occupation of two quantum dots per ancilla qubit. Besides, measurement apparatuses cannot be included in the data row as they would block the shuttling, hence can only exist in the ancilla row. For these

reasons, the ancilla row has to be denser than the data row. This additional space between data qubits is not going to waste as it can be used to fit in the three or four clavier gates per electron required by the conveyor-belt mode of shuttling.

5.3.2 Logical gates and qubit layout

One of the major constraints discussed above is the absence of local single-qubit gates (apart from phase gates which can be performed virtually [176]). As such, if one wants to implement, say, a logical X gate on a given logical qubit, applying transversal X gates on the corresponding surface code would not meet our constraints, as these gates would have to be applied on all data qubits in the device, thereby implementing a global logical X. Local transversal single-logical-qubit gates are thus not permitted in our device (apart from Z gates). This motivates the choice of a protocol that would not make use of such transversal gates, but rather, of lattice surgery only. Indeed, this operation can be made local as our device does feature selective CZ gates. One such approach is described in [90, 124] (and in Section 3.4.3): using Clifford+T as a universal gate set, these papers show that Clifford gates can be commuted through T gates and incorporated in later measurements, thereby only leaving multi-qubit Pauli measurements and multi-qubit $\pi/8$ rotations to implement (Fig. 3.8). These rotations can in turn be performed by consuming a magic state through additional multi-qubit measurements. This protocol thus removes the necessity to implement local Clifford gates that are impractical in our device. However, it comes at the cost of complexified measurements which require a modified lattice surgery protocol. Importantly, while one does need to measure more exotic stabilisers (which I will show are implementable within the given constraints), the multi-qubit logical measurements still run in $O(d)$ rounds. In this sense, the adoption of this scheme is particularly well-suited to architectures that heavily rely on lattice surgery as it minimises the number of required operations (it would not necessarily be the case for architectures where transversal gates are easy to implement). As for the magic state, it can be prepared via state injection and subsequent magic state distillation. The latter only requires the

implementation of logical CNOTs, which can equally be performed via lattice surgery. Using this scheme, the remaining operations our device must be able to implement for universal quantum computation are thus: (*i*) stabiliser measurements, (*ii*) all variants of lattice surgery as described in [124] and (*iii*) state injection. An implementation of these three components within our architectural limitations is described in Section 5.3.3 and 5.3.4.

Before discussing the circuit-level implementation of these elements, it is first relevant to discuss the layout that I decided to adopt for the logical qubits, as this will have an impact on the ordering of the gates in each circuit (*e.g.* to avoid distance-reducing hook errors).

In order to embed an $n \times m$ grid of $d \times d$ tiles of data qubits (as in [90]) in a 2×N device, one must slice the whole grid, say, in the vertical direction. If we assume for now that each tile represents exactly one surface code, this means two consecutive data qubits within a row of a given patch are now separated by nd data qubits rather than d. The consequence is shuttle operations that are n times longer such that these two qubits can participate in the same stabiliser measurement. As shuttling is prone to errors, one way to minimise such long shuttles while permitting full quantum computation is to set $n = 2$ and consider a $2 \times m$ grid of tiles, where the logical information would only be stored in the first row of the grid (region A), while the second row (region B) would serve as a logical ancilla bus used for long-range interactions between the logical qubits of region A (Fig. 5.4). Additionally, a magic state factory is included at the right end of the device. In order to enable all types of interactions between the logical qubits of region A, both their logical X and Z operators should be adjacent to region B. This motivates the use of *wide qubits* as in Fig. 5.4, on top of an ancilla bus that can be initialised differently depending on the desired operation. The slicing is performed in the vertical direction, meaning that the first row of the 2×N device will contain an alternation of d data qubits from region A and d data qubits from region B. In order to entangle physical data qubits of consecutive columns with a physical ancilla qubit, one must therefore

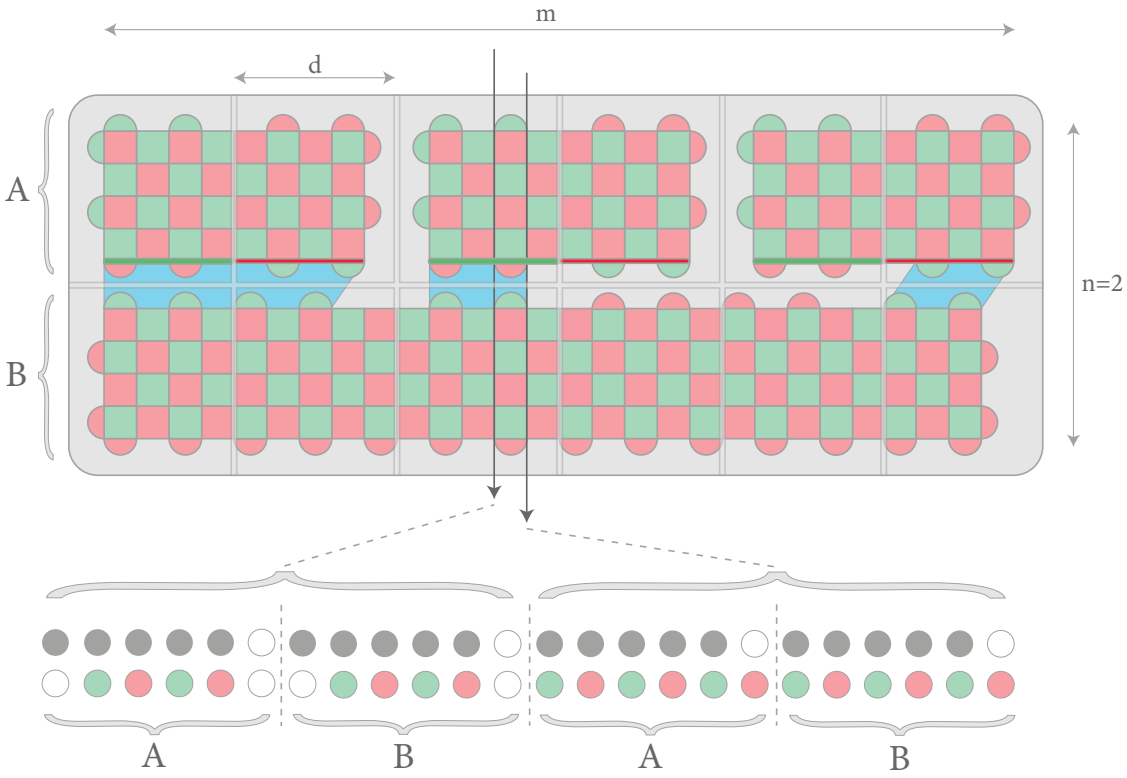


Figure 5.4: Qubit layout for universal quantum computation on the 2×N architecture. We choose to arrange the logical qubits on a grid of $n \times m$ square tiles of size $d \times d$, where d is the code distance [90]. We set $n = 2$ to minimise the shuttling distance. X and Z stabilisers are respectively represented with red and green squares or half-disks. The direction of shuttling is indicated by the vertical dark gray arrows. The top half (region A) contains the logical qubits storing the logical information, while the bottom half (region B) is a logical ancilla bus that can be used to perform long-range interactions between the logical qubits. Each logical qubit is a rectangular patch of surface code, whose X and Z logical operators are represented by red and green lines. Both these logical operators are adjacent to region B, so that they can interact with the logical ancilla. In this example, the ancilla is initialised so as to measure the operator $Y_1 \otimes Z_2 \otimes X_3$ with three lattice surgeries, represented with the blue boxes (see Fig. 44 of [90] for more details). Below, the portion of the layout corresponding to the qubits along the arrows is linearised, showing the alternation of regions A and B in the 2×N architecture.

shuttle by roughly $2d$ increments in one direction, and the same distance back (instead of d in the case of a single encoded surface code).

5.3.3 Stabilisers implementation

Gate ordering

The next step is to ensure that our three building blocks needed for fault tolerant quantum computation — stabiliser measurements, modified lattice surgery and state

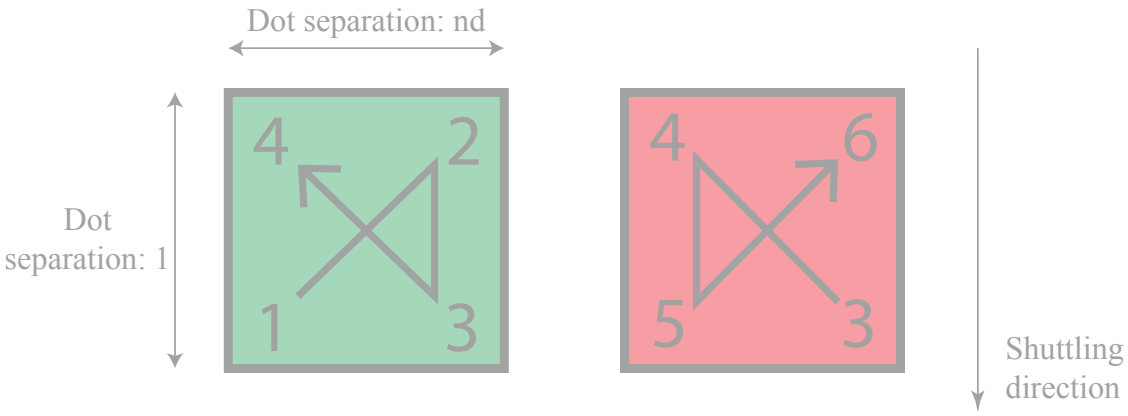


Figure 5.5: Gate ordering enabling one to measure X and Z stabilisers in an interleaved (but staggered) fashion. While X stabilisers perform steps 5 and 6, Z stabilisers undergo steps 1 and 2 again, and so on. The distance between data qubits (in terms of number of quantum dots) is indicated. This sequencing guarantees a minimum shuttling distance, a minimum number of shuttles and no distance-reducing hook errors.

injection — can indeed be implemented within our constraints. Before giving explicit circuits or procedures implementing these operations, one must first determine the order in which data qubits must be entangled within a given stabiliser measurement. This is an important matter as a suboptimal choice of ordering can lead to hook errors which effectively reduce the distance of the code. Additionally, a proper sequencing can help reduce the stabiliser circuit depth, number of shuttles and shuttling distance by interleaving the gates of the X and Z stabilisers. I verify in Appendix B.1 that the gate ordering depicted in Fig. 5.5 (*i*) does not create any hook error; (*ii*) allows one to interleave the gates of the X and Z stabilisers; (*iii*) is nearly optimal in terms of the number of shuttles and total shuttling distance.

In the case of modified lattice surgery however, the question of interleaving the dislocations and twists (see next section) with the regular stabilisers is much more complex. Not doing so is always an option, however, that comes at the cost of additional shuttles, and leaves some qubits idle while others are being measured. We leave a study similar to Fig. 16 of [124] to further research.

Stabiliser circuits with singlet-triplet ancilla qubits

Given the silicon spin platform which we are principally considering, we take it that the preferred embodiment of an ancilla qubit is via two spins, with measurement

occurring by differentiating between singlet and triplet states. After the ancilla is initialised in the singlet state, if there are an odd number of errors affecting the data qubits then the ancilla should transform to a triplet state, while an even number of data qubit errors should leave it in the singlet state. A Pauli spin blockade measurement then allows one to extract the value of the stabiliser.

Fig. 5.6 gives a circuit implementation of all types of stabilisers involved in regular error correction as well as lattice surgery, using CZ gates and semi-global Hadamards only (*i.e.* affecting all data qubits at the same time). These stabilisers fall into four categories: regular X stabilisers, regular Z stabilisers, dislocations (stabiliser checks involving both X and Z) and twists (stabiliser checks involving X, Z and Y). The two latter appear in the modified lattice surgery protocols of [90] when X and Z boundaries are facing (rather than both X or both Z in the conventional lattice surgery picture). One can check that, with this implementation, an error on a data qubit always transforms the singlet state $|S\rangle = (|01\rangle - |10\rangle)/\sqrt{2}$ into the triplet state $|T\rangle = (|01\rangle + |10\rangle)/\sqrt{2}$. This would not be the case if using CNOTs for the Z stabilisers (with data qubits as controls and one ancilla line as target), as it would transform a singlet state into a different triplet state $|T'\rangle = (|00\rangle - |11\rangle)/\sqrt{2}$.

This proves crucial when implementing dislocations and twists. Indeed, as an example, let us consider the measurement of the dislocation operator XZ on data qubits in the $|-\rangle \otimes |1\rangle$ state. This should lead to the final measurement of a singlet state. Let us first assume that the Z parity measurement is implemented with a CNOT targeting one qubit of the ancilla pair (rather than a CZ as in Fig. 5.6). The quantum state of the data/ancilla system evolves as follows:

$$\begin{aligned} |-\rangle\,|1\rangle\,(|01\rangle - |10\rangle)/\sqrt{2} &\longrightarrow |-\rangle\,|1\rangle\,(|01\rangle + |10\rangle)/\sqrt{2} \\ &\longrightarrow |-\rangle\,|1\rangle\,(|11\rangle + |00\rangle)/\sqrt{2} \end{aligned}$$

The final ancilla state is a triplet — this implementation gives the wrong measurement outcome. Let us now assume that the Z parity measurement is implemented

with a CZ gate, as prescribed in Fig. 5.6. This time, the quantum state of the data/ancilla system transforms as follows:

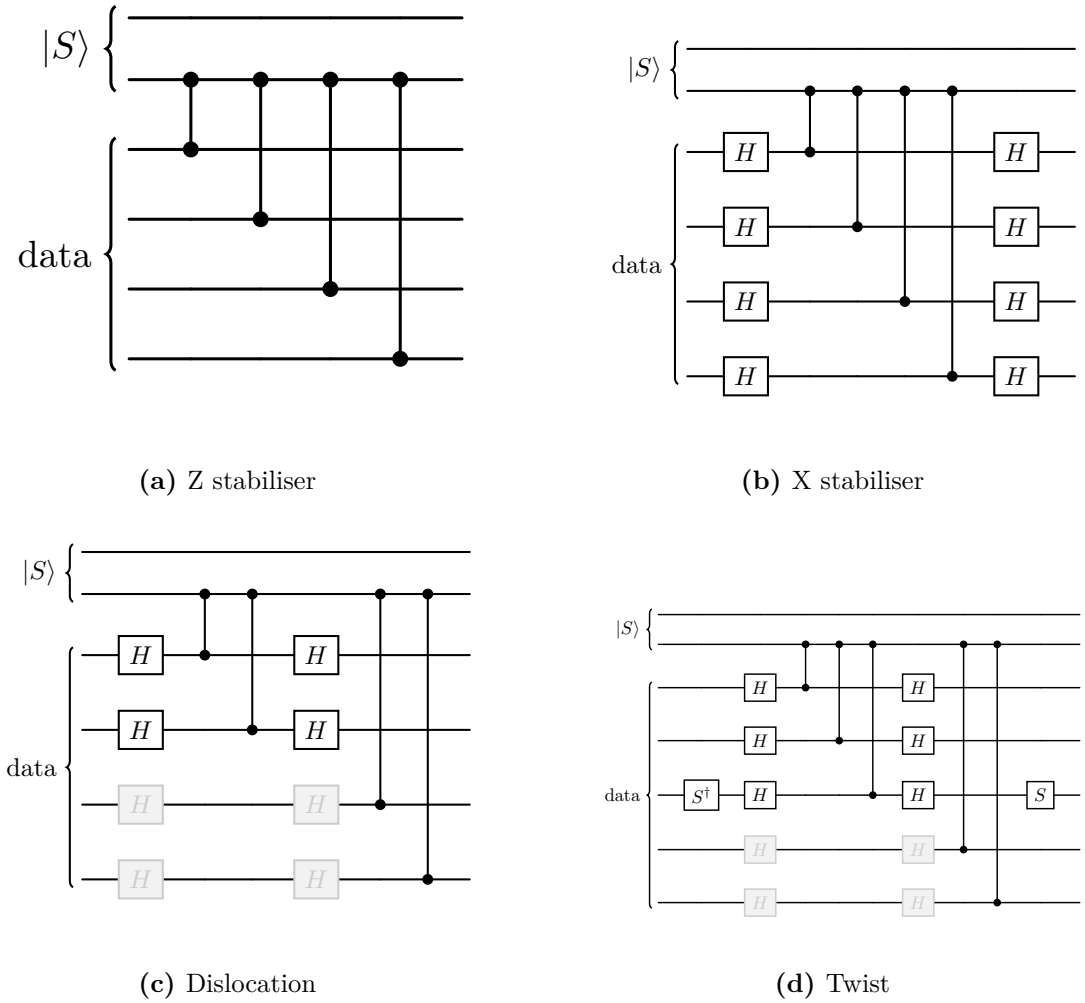
$$|-\rangle|1\rangle(|01\rangle - |10\rangle)/\sqrt{2} \longrightarrow |-\rangle|1\rangle(|01\rangle + |10\rangle)/\sqrt{2}$$
$$\longrightarrow |-\rangle|1\rangle(|01\rangle - |10\rangle)/\sqrt{2}$$

The ancilla state is thus back to the singlet state, which is the correct behaviour.

The twist measurement can be implemented in a similar way as the dislocation. Also, the presence of the S and $S^\dagger$ gates is not problematic as these are phase gates, which can be implemented locally.

Full circuit

The two previous sections respectively addressed the questions of gate ordering and individual implementation of all types of stabilisers. The last step is now to give a protocol to combine these two elements and obtain the full circuit implementation of a stabiliser cycle. At a given time step, the qubits are scheduled to undergo one of the following operations: shuttling; initialisation or measurement; or single- or two-qubit gates. In the latter case, gates can be separated in two sets: gates used for Z parity measurement (single CZ's) (set 1) and gates used for X parity measurement (H-CZ-H sequence) (set 2). We include the Y parity measurement of the twist in set 2 as the S gates can be implemented locally and are thus not problematic. The idea is then simply to perform these two sets of gates one after the other, and the undesired Hadamards will simplify. Note that the same technique was already used in Fig. 5.6. Additionally, by following the order 'set 1 - set 2 - set 2 - set 1' over two consecutive rounds of stabiliser measurements, one can further halve the number of Hadamards. This protocol is illustrated in Fig. 5.7 in the case of no dislocations or twists (but the same idea would apply if they had to be measured too). Gate set 1 is represented in green and gate set 2 in red.



(a) Z stabiliser

(b) X stabiliser

(c) Dislocation

(d) Twist

Figure 5.6: Circuit implementation of all kinds of stabilisers needed for regular error correction and modified lattice surgery [124], using a singlet-triplet ancilla qubit. This implementation also respects a semi-global implementation of Hadamard gates, meaning that they are globally applied on all data qubits. The light gray Hadamards are only included for this purpose and cancel each other.

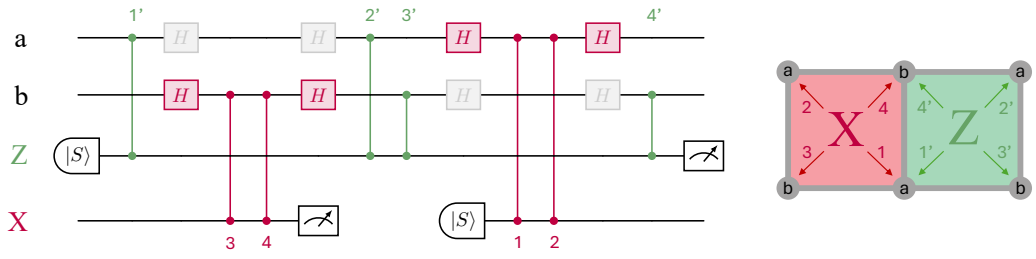


Figure 5.7: Circuit implementation of a cycle of X and Z stabilisers over one unit cell of the code. This implementation respects both the ordering avoiding hook errors, and makes use of semi-global Hadamard gates only. The a/b indexation for the data qubits is chosen so as to respect the periodicity of the lattice. The numbers (1 to 4 for X, 1' to 4' for Z) indicate the order in which gates should be implemented to avoid hook errors.

5.3.4 State injection

So far, I have shown how to implement all stabilisers involved in regular error correction as well as lattice surgery on the gate level. The final ingredient needed to enable universal quantum computation is state injection. The first step is to determine which logical states one would want to initialise.

The first one is the ancilla state used for multi-qubit measurements. Let us denote P_i (resp. $P_{i,ancilla}$) the Pauli operators applied to the i-th logical data (resp. ancilla) qubit. For the logical measurement of $P_1 \otimes ... \otimes P_n$, Fig. 44 of [90] prescribes the initialisation of the mediating logical ancilla in the $|+\rangle^{\otimes n-1}$ state, and measurement of the operators $Z_{i,ancilla} \otimes P_i$ via lattice surgery. In our case, as we can directly prepare $|0\rangle$ states only, preparing $|+\rangle$ states would require the application of an additional Hadamard. Therefore, it is simplest to initialise the logical ancilla in the $|0\rangle^{\otimes n-1}$ state and measure $X_{i,ancilla} \otimes P_i$ instead, which is strictly equivalent. Besides, as explained in [90], the ancilla is insensitive to Z errors, as it is prepared in Z eigenstates. This is particularly handy for us as our code is more prone to Z errors as a result of shuttling (whose precise noise model will be studied in the next section).

The second logical state one would want to prepare is the magic state $|m\rangle = (|0\rangle + e^{i\pi/4}|1\rangle)/\sqrt{2}$. To do so, I use a modified version of [121, 122]. The main obstacle in the direct implementation of these protocols on our device is the restriction on single-qubit gates. In particular, we cannot prepare a $|+\rangle$ state without applying a global Hadamard. Yet, these schemes require the preparation of the data qubits in both the $|0\rangle$ and $|+\rangle$ states. This issue can be circumvented by staggering the initialisation of the data qubits. More concretely, one would first prepare all qubits that should be initialised in the $|+\rangle$ state, as well as the physical magic state, in the $|0\rangle$ state. A global Hadamard is then applied, bringing all these qubits to the $|+\rangle$ state. Then a local phase gate can be applied on the physical magic state, thereby preparing it in the $|m\rangle$ state. All remaining qubits can then be prepared in the $|0\rangle$ state, and the state injection procedure of [121, 122] can be performed normally.

One important thing to note however is that whenever a global Hadamard is applied, it does not just target the patch of surface code one intends to prepare, but also all other data qubits in the device. As a result, a transversal H gate is applied to all logical qubits in the middle of the quantum computation. This results in the application of logical Hadamards, as well as the permutation of the X and Z stabilisers of each surface code. While this could seem problematic, these additional Hadamards can easily be undone by the application of another logical Hadamard on all qubits. This is no different than any other Clifford gate in the circuit and can thus be performed virtually, by changing the basis of the following gates and measurements.

Once a magic state has been injected into a surface code, it can be distilled via standard magic state distillation *i.e.* the 15-to-1 protocol. This protocol only requires logical CNOT gates, which can be implemented via lattice surgery.

5.3.5 Performance simulations under realistic noise

In the previous sections, I showed that universal error correction is possible from a theoretical point of view on a 2×N spin-qubit device with strong constraints, such as the access to semi-global single-qubit gates only (apart from phase gates). In the following, I argue that our proposition is also practical, *i.e.* that the required physical error rates and resource requirements for such a device are not experimentally out of reach.

Noise model

Let me first discuss the noise model I adopt to estimate the performance of our system. The main ingredient we extensively make use of here and that requires modelling is shuttling. When an electron is shuttled, its g-factor varies due to inhomogeneities in the device, which induces unwanted phase rotations in the laboratory reference frame. Systematic rotations can be calibrated away, so that we are effectively describing an electron in a shuttled reference frame [36, 177, 178]. When the phase is not exactly cancelled out owing to incorrect calibration, one can

expect the data qubits of the device to each undergo some small coherent rotation (remember that only data qubits are shuttled). As coherent errors are difficult to simulate classically and potentially more harmful than stochastic errors, twirling is generally used to transform them into Pauli errors [60]. However, this method is unfortunately not fully permitted in our device as some of the single-qubit gates can only be implemented semi-globally. As such, the only natural candidate for a twirling gate set tailored to phase rotations and respecting our constraints is $W = \{I, X_1 \otimes ... \otimes X_n\}$ (with I the identity gate, X_i the Pauli X gate on data qubit i and n the number of data qubits). Nonetheless, one can easily see that a Pauli Z error on an even number of data qubits commutes with both gates of W and will thus not be extracted from the coherent phase rotations by the twirling gates (another, more mathematical justification can be obtained by using Eq. 7 of [179]). Therefore, only odd-order errors (*i.e.* Pauli Z errors affecting an odd number of qubits) will correctly be twirled. If this poses some difficulties for the exact simulation of our system, it should however not be fatal in terms of performance, as the logical-level noise has been shown to behave similarly for coherent and random errors in large enough surface codes [58].

Therefore, even though the twirling process is not perfect, I will still approximate the remaining errors by some random dephasing of probability p_{sh} for simulation purposes, which is correct up to first order (as second-order Z errors are not correctly twirled). p_{sh} here represents the probability that a given data qubit is affected by a Pauli Z error somewhere within a given stabiliser cycle. To first order it is therefore the sum of four contributions, corresponding to the four shuttles that are carried out within a cycle. Ignoring non-adiabatic effects for now, Eq. 4 of [36] shows that the probability of a dephasing error $\delta\phi^2/2$ *for one shuttle* follows:

$$\delta\phi^2/2 = 2\frac{l_c L_s}{(vT_2^*)^2} \tag{5.1}$$

with L_s the shuttling distance, v the shuttling speed, T_2^* the characteristic dephasing time experienced by stationary spins and l_c the coherence length of the dephasing noise due to shuttling. The latter arises from imperfect calibration of the dynamic

reference frame used to absorb coherent rotations happening when electrons are shuttled. These calibrations errors stem from uncontrolled fluctuations caused by slow nuclear dynamics due to dipolar interaction or low-frequency $1/f$ charge noise affecting the spins' g-factors. Now summing the contributions of the four shuttles happening during a stabiliser cycle, one gets:

$$p_{adia} = 2\frac{l_c \times 4dl_{dd}}{(vT_2^*)^2} \tag{5.2}$$

where l_{dd} is the spacing between two data qubits. The total shuttling distance over a whole stabiliser cycle is roughly $4dl_{dd}$, as we need to shuttle by $2d$ increments and back (remember that we are interleaving logical data qubits and logical ancilla qubits, both of them of distance d, see Fig. 5.4).

Nonetheless, the above analysis would not be complete if I did not consider dominant non-adiabatic effects. The lowest energy splitting in silicon quantum dots is the valley splitting E_{VS}, which corresponds to the gap between the two out-of-plane conduction bands (or valleys). These two valley states are additionally characterised by distinct g-factors [38]. Consequently, an electron in the excited valley state, as opposed to the expected ground valley state, will precess at an unknown frequency, leading to unwanted phase rotations (see Section 2.2.6). One can estimate the amount of non-adiabaticity as follows: E_{VS} typically ranges between 10 and 200μeV for SiGe or can exceed 500μeV in SiMOS architectures [36]. Furthermore, the two valleys are site-dependent, meaning that they depend on the position $x(t)$ of the electron in the device. Assuming that the length scale of the electron wavefunction is roughly $l_x = 50$nm, one can assume that in the worst-case scenario the electron's valley state would switch every 50nm, leading to a characteristic energy $hv/l_x = 0.83\mu$eV, where h is Planck's constant. This is only 12 times smaller than 10μeV, *i.e.* the worst-case value for E_{VS}, justifying the inclusion of these non-adiabatic effects in my modelling. Further modelling and simulations found in Appendix B.2 lead to supplemental dephasing errors due to the valley state of $p_{dia}/4d = 1.4 \times 10^{-6}$. This value will be used in all the simulations of this chapter. The final shuttling-induced dephasing probability is then given by $p_{sh} = p_{adia} + p_{dia}$.

With these considerations in mind, we now have all necessary ingredients to simulate the performance of the device. The full noise model includes (*i*) dephasing channels of probability $p_{sh} = p_{adia} + p_{dia}$ on the data qubits at the beginning of each stabiliser cycle; (*ii*) depolarising channels of error rate p after each gate of the stabiliser circuit (including twirling gates and cancelled out Hadamards arising from their semi-global implementation); (*iii*) depolarising channels of error rate p after any initialisation; (*iv*) classical flip of any measurement outcome with probability p.

Note that here I ignore idling errors — these account for errors that idle qubits accumulate while others are being operated on. Indeed, spin-coherence times T_2 exceeding 20ms have been demonstrated for purified silicon electron spins [180], which is 4 orders of magnitude above initialisation, measurement and single- and two-qubit gate times (at most a few microseconds [47, 181–183]). These T_2 times are obtained via dynamical decoupling, which can be implemented by periodically flipping all the spins in the device. In our case, it is simplest to flip all spins at the same time, as this is permitted by the semi-global pulse I am assuming.

In the following simulations, I fix the gate noise p to 0.1%, so as to be below the usual circuit-level noise threshold of the surface code (around 0.7% [184]). This is consistent with experiments showing silicon spin qubit fidelities exceeding 99.9% [44]. The dephasing time T_2^* will range between 1μs and 8μs, which has been achieved [20, 44], and is well below demonstrated dephasing times of 100μs [183]. For the estimation of p_{sh}, I fix $v = 10$m/s, $l_c = 100$nm and $l_{dd} = 140$nm [36]. This choice of 140nm for the distance between two data qubits takes into account the additional spacing for the clavier gates required by the conveyor-belt mode of shuttling, and leaves extra space for singlet-triplet ancilla qubits and measurement apparatuses in the static row. These values in turn give a dephasing probability *per shuttling increment* $p_{sh}/4d$ between 5.8×10^{-6} and 2.7×10^{-4}.

The most optimistic end of this range corresponds to a more mature technology than the early demonstrations reported so far, which now approach 10^{-4} [34, 35, 37]. However the task of low-noise shuttling is receiving rapidly increasing attention, notably with recent proposals suggesting that g-factor disorder might help improve

Parameter	Notation	Value
Dephasing time	T_2^*	1μs to 8μs
Total shuttling error per increment	$p_{sh}/4d$	5.8×10^{-6} to 2.7×10^{-4}
Circuit-level noise	p	0.1%

Table 5.1: Relevant physical noise parameters for the subsequent QEC simulations.

shuttling fidelities rather than hinder them [185]. I will thus assume that these will keep improving. Important physical noise parameters are summarised in Table 5.1.

As the surface code is a CSS code, X and Z errors can be analysed separately. Since our code is more prone to Z type errors, I focus only on them. Therefore, I numerically evaluate the X logical error rate of a single distance-d wide surface code (*i.e.* with both logical operators facing in the same direction, see row A of Fig. 5.4) used in memory mode via N_{runs} runs of Monte Carlo simulations, with N_{runs} ranging between 10,000 and 1,000,000 depending on the target logical error rates. In each run, random errors are injected in the code according to the error rates p and p_{sh}, affecting both data and ancilla qubits for d rounds of stabiliser measurements. The syndrome they create is then decoded via Minimum Weight Perfect Matching [7]. The initial errors and the correction are then added to determine if the X logical operator value was flipped.

Simulations

I first plot in Fig. 5.8 the logical error rate for several code distances d, against $p_{sh}/4d$ *i.e.* the shuttling error per shuttling increment. The main observation of the plot is that the lines corresponding to surface codes of various distances do not cross at a single point. This is expected as lines crossing at the same point, equivalent to the existence of a threshold at this crossing point, should only appear if the noise model respects certain properties. In particular, the noise strength should not scale with the code size [75]. Yet, here I assumed that $p_{sh} \propto 4d$. As a result, the crossing points of subsequent-distance surface codes cross at lower and lower values of p. In other words, there does not seem to be any value of p below which increasing the code distance consistently reduces the error rate. Rather, there is a constant

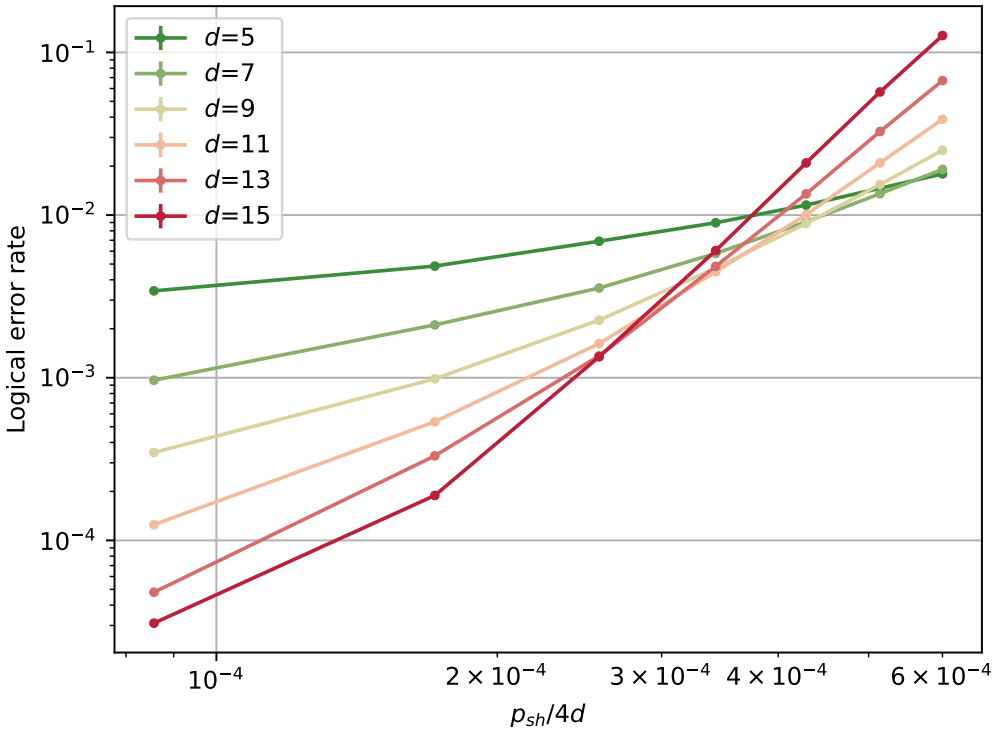


Figure 5.8: Logical error rate of a wide surface code against $p_{sh}/4d$, *i.e.* the probability that one shuttling increment introduces a dephasing error, plotted for several code distance d. The gate, measurement and initialisation noise p is set to 0.1%.

trade-off between increasing the code distance for more powerful error correction, and maintaining it low enough to keep the noise due to shuttling manageable.

The anticipated non-existence of a threshold does not present any basic issue for our approach. Since the ideas presented here are intended for the early fault-tolerant era, the relevant quantity is always the logical error rate that can be achieved for realistic experimental parameters and specific system sizes. In Fig. 5.9, I thus plot the logical error rate against d for T_2^* ranging between 1μs and 8μs. The simulation data, represented by small dots with error bars (quantifying the sampling error), is then fitted with the following trial function:

$$p_{\log}(d) = A(\alpha + \beta d)^{\gamma d+\delta}. \tag{5.3}$$

The motivation for this function is the following. If one denotes p_{tot} the total noise experienced by the system, the surface code promises an exponential suppression of

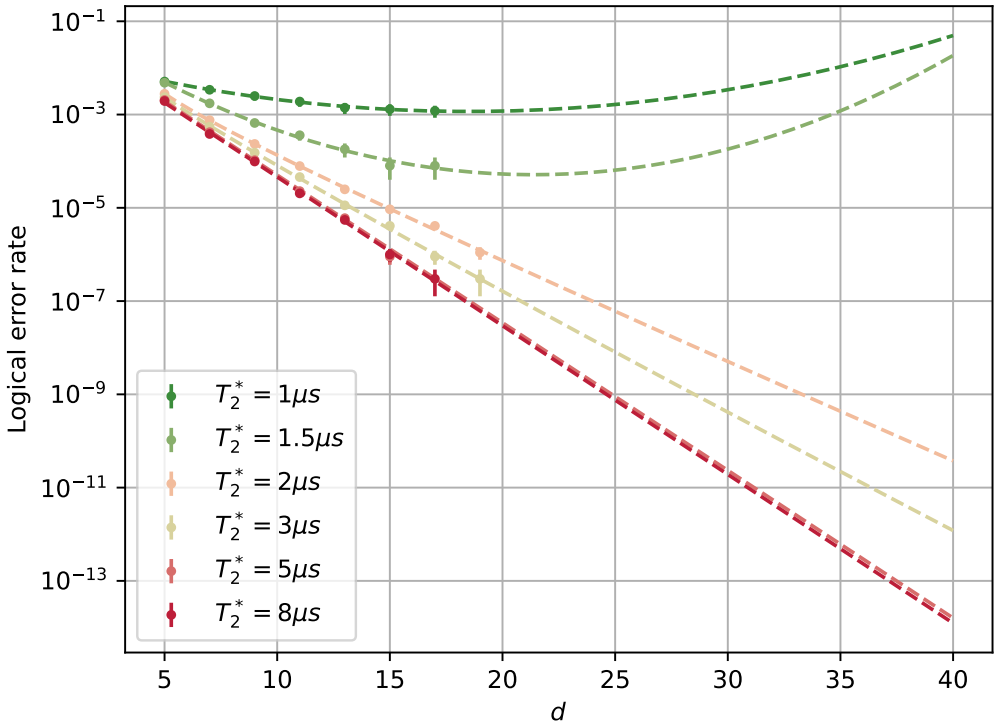


Figure 5.9: Logical error rate of a wide surface code against the code distance, plotted for several values of the stationary dephasing time T_2^*. The gate noise p is set to 0.1%. The dots represent the simulation data, their error bars quantifying the sampling error. The dashed lines correspond to a fit of the data according to the trial function of Eq. 5.3.

the errors of the form $p_{\log}(d) \propto (p_{tot}/p_{th})^{d/2}$ with p_{th} a constant. Then, up to first order, p_{tot} can be written as a sum of the gate noise p and the shuttling noise p_{sh}, where the latter is proportional to the code distance d. The consequence of p_{tot}'s dependence on d is a sub-exponential suppression of errors, or worse, an increase in the error rate when the distance of the code becomes too large. Fortunately, with dephasing time that is large enough yet experimentally plausible, low enough error rates (around 10^{-13}) can be reached before this phenomenon occurs.

5.3.6 Resource estimation for universal quantum computation

With all the above ingredients, I now estimate the resources required for our device to achieve two meaningful milestones. In all the following, I set a spins' stationary

dephasing time $T_2^* = 8\mu$s, a per-dot valley noise $p_{dia}/4d = 1.4 \times 10^{-6}$ and a gate noise $p = 0.1\%$. The first task I focus on is estimating the size N that would bring the 2×N architecture beyond the NISQ regime. More concretely, I aim to estimate N such that the device would contain 50 logical qubits, able to interact through a universal set of gates, and with logical error rates of at most 0.01% (ten times lower than physical error rates). Using Fig. 5.9, one can see that distance-9 wide surface codes are sufficient. Moreover, following our state injection protocol (Section 5.3.4), the probability of preparing a magic state in the wrong state is of the order of the physical error rate of the operations that created it, that is p and p_{sh}. To first order, the probability p_{mag} of initialising an erroneous logical magic state is thus of the order of $p + p_{sh} = 0.103\%$ (ignoring small constant factors, see Eq. 1 of [121]). This lowers to $35p_{\text{mag}}^3 = 4 \times 10^{-8}$ after a 15-to-1 distillation protocol — way below our target logical error rate of 0.01%. To meet our above requirements, one would therefore need $2 \times (50 + 15) = 130$ distance-9 surface code patches (including logical data qubits and magic state distillation factory, and doubling this number to account for the logical ancilla bus connecting all these logical qubits, see Fig. 5.4). This translates into an overall size of $N = 2 \times 10^4$ physical data qubits.

Beyond simply outperforming NISQ on paper, what one would actually want from a fault-tolerant quantum computer is to be able to run meaningful quantum algorithms. The second task I will thus focus on is estimating the ground-state energy of a 2D Hubbard model Hamiltonian. This can be solved by preparing the unitary $\mathcal{W}(H) = \mathrm{e}^{\mathrm{i}\arccos(H)}$ by qubitisation, then passing it to phase estimation. An optimised protocol to run this algorithm can be found in [186] along with the resource requirements for the smallest instance of this problem that is within the classically intractable regime, *i.e.* a 6×6 Hubbard model. For this grid size, 100 logical qubits and 10^8 logical T gates would be required.

Let us again set a spins' stationary dephasing time $T_2^* = 8\mu$s and a gate noise $p = 0.1\%$. Let us assume that both the probability of a logical error impacting *any* individual T gate, and the probability of an error impacting *any* of the logical qubits in the circuit, is below 10%. The algorithm thus fails with a probability of

around 20%. The quantum computation can then be run multiple times in parallel to exponentially increase the success probability. Following the procedure of [90], one can estimate the physical resources needed to obtain the correct outcome.

Magic state distillation The error rate of each individual T gate must be under 10^{-9} to ensure that the failure probability of any of the 10^8 gates is under 10%. Again, the probability of initialising an incorrect logical magic state before distillation is $p_{\text{mag}} \sim p + p_{sh}$. Assuming that the distance of the surface code is under 40 (which will be verified later on), one guarantees $p_{sh} < 0.02\%$. Adding the gate noise, one obtains a total noise $p_{\text{mag}} \lesssim 0.12\%$. Using the 116-to-12 distillation protocol, the probability of outputting incorrectly distilled magic states is $41.25 p_{\text{mag}}^4 \lesssim 10^{-10}$ [90], which is under our target per-gate error of 10^{-9}. 44 surface codes patches are needed to run the 116-to-12 distillation protocol, which outputs 12 logical magic states in $99d$ time steps with 89% success probability [90]. The initialisation time of one correct magic state is thus on average $9.27d$ time steps. The magic states can then be consumed one by one in the quantum computation, which requires d time steps for each state. The overall time cost of the quantum computation (to prepare and consume 10^8 magic states) is thus $10.27d \times 10^8$. It requires 100 logical qubits to store the logical information and 44 surface code patches in the distillation block (not counting the logical ancilla). Adding the long logical ancilla bus mediating all interactions, the logical qubit count doubles to 288.

Code distance Given the total number of logical qubits and the computation time, in order to set the probability of a logical error affecting any of the logical qubits during the computation under 10%, one requires the per-logical-qubit error rate p_L to be below

$$p_L < 0.1/(288 \times 10.27 \times 10^8) = 3 \times 10^{-13}. \tag{5.4}$$

By using the fitting function of Fig. 5.9, we can choose the code distance to be $d = 36$. This is indeed within the upper bound of 40 that I set earlier.

Summary Therefore, in order to run one instance of a 6×6 Hubbard model (which is within the classically intractable regime) on our device, one would need 288 wide qubits of distance $d = 36$, that is 1.4×10^6 physical (data and ancilla) qubits. The algorithm would run in $10.27d \times 10^8$ stabiliser cycles.

The total shuttling time over a stabiliser cycle is $t_{sh} = 4dl_{dd}/v$, where $l_{dd} = 140$nm is the distance between two consecutive data qubits, and $v = 10$m/s is the shuttling speed. Thus $t_{sh} = 2\mu$s. Besides, while two-qubit gates can be implemented much faster (around 0.1μs [181]), initialisation, measurement and single-qubit gates all take around a microsecond [47, 182, 183]. We can thus realistically assume an overall stabiliser cycle time of 6μs, which would bring the computation time to a total of 2.5 days for an algorithm that is only slightly beyond the classically tractable regime. For reference, it uses ten times less logical qubits and 60 times less T gates than the 2048-bit RSA factoring [88]. While this time may seem relatively daunting, one must remember that I exclusively used experimental parameters that have been achieved, ignoring any further optimisation that will surely happen in the following years before this kind of device can be implemented. My shuttling noise model, informed by earlier theoretical studies (esp. Ref. [36]), assumes levels of imperfection that are smaller than existing early demonstrations; I am confident that improving experimental techniques driven by rapidly increasing community interest will reach the domain that I have simulated. From Fig. 5.9, one observes that for the more optimistic coherence times we have a scenario where shuttling noise is among the least significant factors in the model (as there is very little variation in the logical error rate when T_2^* is decreased from 8μs to 5μs, *i.e.* when the shuttling error per increment varies between 6×10^{-6} and 1.3×10^{-5}). A limiting factor would thus be the gate infidelity $p = 0.1\%$. If this quantity could be improved by only a factor 2, logical error rates would drastically reduce due to almost exponentially scaling suppression (as long as the shuttling noise is kept low enough). Moreover, the assumed speed of the various operations corresponds to already-accomplished demonstrations, and does not begin to approach any

fundamental physical limitations. One might reasonably expect orders of magnitude improvement over generations of such devices.

5.4 Quantum memories with general qLDPC codes

In Section 5.2 I described the framework established by my co-author Armands Strikis, showing that our device is suitable for the implementation of various classes of qLDPC codes beyond the surface code. These codes are known for their high performance and can fully exploit the connectivity (beyond that required by the surface code) which is provided by our shuttling-based system. Therefore, in this section, I explore the potential of such non-local codes, constructed by either allowing more ancilla-data qubit interactions, or by increasing the shuttling distance. Here, I only study their use as memory codes.

The gate ordering I choose in the stabiliser circuits is not optimised. All X stabilisers will be implemented when the data qubits are shuttled one way, followed by all Z stabilisers when the data qubits are shuttled back. Regarding the noise model, I adopt the same as in the previous section — circuit-level noise plus additional dephasing due to shuttling (proportional to the shuttling distance). Due to the more complex data-ancilla interactions and to the non-optimal gate ordering, not all qubits perform their entangling operations synchronously (while it is the case for the surface code). Consequently, the data qubit rail might have to stop and start more often, and qubits that are not interacting will have to stay idle. To model this, I introduce an additional noise parameter p_{idle}, and add depolarising channels of probability p_{idle} every time a qubit is idle while others are interacting. Note that I neglected this in the previous section due to the high coherence time of silicon spin qubits.

The numerical experiments are run under various values of the gate noise p, the idling noise p_{idle} and the dephasing time T_2^*. The per-dot valley noise is still fixed to $p_{dia}/4d = 1.4 \times 10^{-6}$. Each code is simulated for d rounds of stabiliser cycles, where d is the code's estimated distance. The decoding is performed via *min-sum* BP-OSD, a limit of 32 iterations, a scaling factor of 0.625 and a serial schedule

[95]. The logical error rate *per round per logical qubit* p_{log} defined below is then plotted and used to compare the performance of various codes [112]:

$$p_{log} = 1 - (1 - P_L(k, d))^{1/kd}. \tag{5.5}$$

Here, $P_L(k, d)$ is the probability that a logical error happens on at least one of the k logical qubits of an $[n, k, d]$ code running for d rounds of stabiliser cycles.

5.4.1 Hypergraph product code with a repetition code

As explained in Section 5.2.3, one class of codes that is particularly suitable for our architecture are the hypergraph product (HGP) codes constructed with the repetition code as one of the seed codes.

In the following, I study the example of a [234,3,8] HGP code. The code is obtained by taking the product of a classical [17,3,8] LDPC code H_1 (generated randomly until good parameters were obtained, see Appendix B.3) and a distance-8 repetition code H_2. To limit the number of two-qubit gates, which are expected to be the leading source of noise, I generated H_1 by enforcing two 1's per row on average, thereby guaranteeing that the stabilisers of the quantum code are weight-four on average (attempts with higher-weight stabilisers led to poorer performance). This code is more compact than the surface code as it encodes $k = 3$ logical qubits with $(17 + 14) \times (8 + 7) = 465$ physical qubits (data and ancilla). On the contrary, one would require $k \times (2d - 1)^2 = 675$ physical qubits to encode the same amount of logical information with same-distance surface codes.

Fig. 5.10 compares the logical error rates per round per logical qubit p_{log} of the [234,3,8] HGP code and of surface codes of various distances. The shuttling noise is fixed by setting $T_2^* = 8\mu$s. As explained above, the HGP code experiences the same amount of shuttling error as the surface code, but higher idling noise. Indeed, its two-qubit gates cannot be implemented synchronously due to the randomness of the matrix H_1 (it contains many non-zero diagonals). Qubits that are not interacting thus have to stay idle, which induces depolarising errors at a rate p_{idle}. I thus plotted p_{log} for two levels of idling errors: $p_{idle} = 0$ and $p_{idle} = 0.1p$. One can observe

that for no idling errors and for a relevant range of gate noise p, the HGP code performs similarly as surface codes with comparable distances, yet requiring 30% less qubits. As expected however, when the idling noise is increased, the HGP code's performance falls dramatically while the surface code's is very moderately affected.

While observing this, one needs to take into account that I did not optimise the stabiliser circuit. An interleaved scheme of X and Z measurements could drastically reduce the overall idling noise. This could be implemented with the procedure presented in Ref. [187]. Besides, in the specific case of silicon spin qubits, p_{idle} is expected to be extremely small. Indeed, given the experimental values observed for the coherence time T_2, around 20ms [180], and the slower gates times, around $T_{gate} = 1\mu$s [47], one would obtain an idling error probability of

$$p_{idle} = 1 - \mathrm{e}^{-T_{gate}/T_2} = 5 \times 10^{-5}.$$

When using this value, one would obtain the same qualitative performance as for the $p_{idle} = 0$ case.

Furthermore, here I set a single parameter p_{idle} quantifying the idling error. It would be more comprehensive to distinguish different p_{idle}'s depending on the operation that is being waited for. Specifically, as two-qubit gates are one order of magnitude faster than single-qubit gates, they would be responsible for shorter wait times, thus lower idling errors. Indeed, it so happens that the difference between the HGP code and the surface code lies in the additional idling times due to asynchronous two-qubit gates, which are the fastest operations in the circuit.

Lastly, do note that the crossing point of the surface code lines around 1% is not a threshold in the common sense, as I am here plotting the logical error rate per round per logical qubit.

5.4.2 Generalised bicycle codes

In the previous section, I studied a type of code that did not require an increase of the shuttling distance compared to the surface code, but I did not set any constraint on the number of stops along the way. I therefore guaranteed both a shuttling and

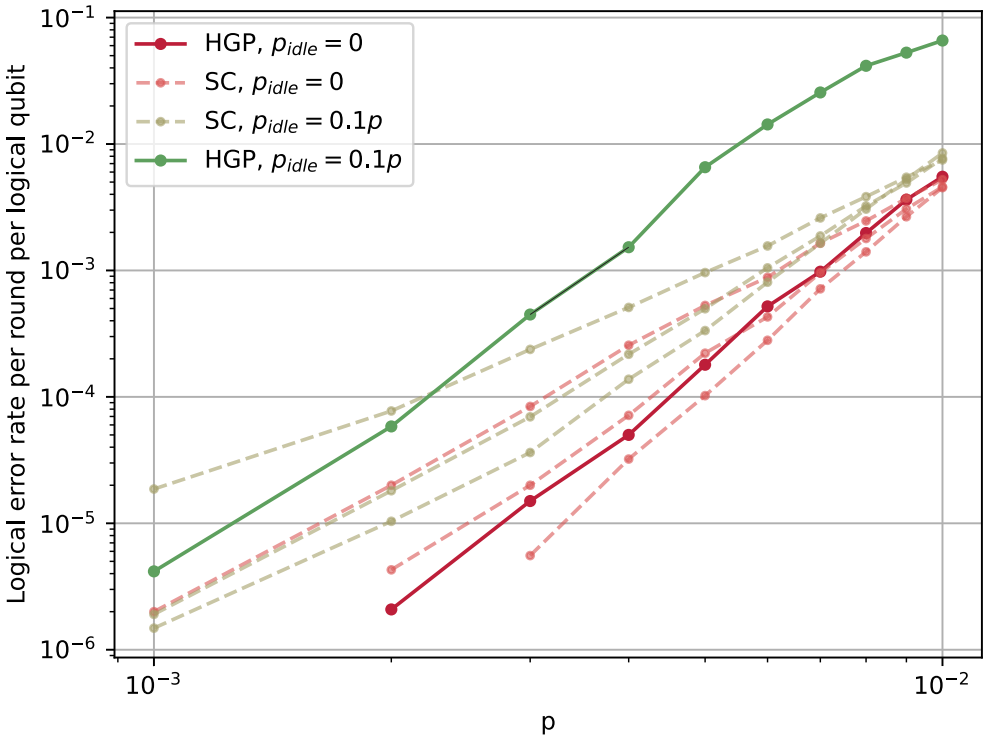


Figure 5.10: Logical error rate per round per logical qubit of a [234,3,8] hypergraph product code (HGP, solid lines) and surface codes (SC, dashed lines) of various distances d, against the circuit-level noise parameter p. The HGP code is built from the product of a repetition code and a classical code generated randomly. The dephasing time T_2^* is set to 8μs, and the logical performance of the HGP and surface codes are plotted for $p_{idle} = 0$ and $0.1p$. For both levels of idling noise, the three surface code lines correspond from top to bottom to a distance of 5, 7 and 9 respectively.

idling noise scaling as $O(\sqrt{n})$, where n is the number of data qubits in the code block. Here I study another paradigm, where I do not restrict the shuttling distance, but instead bound the number of stops: $p_{sh} = O(n)$ and $p_{idle} = O(1)$. Codes that satisfy this are generalised bicycle codes [95], as their check matrices by definition contain a constant number of non-zero diagonals (see Section 5.2.3).

In the following, I study the performance of a [126,28,8] generalised bicycle code (code A2 of [95]). This code contains contains 252 physical qubits (data and ancilla). Using the same number of physical qubits to encode $k = 28$ logical qubits, one would need to use 28 surface codes of distance

$$d_{sc} = \frac{\sqrt{252/k} + 1}{2} = 2.$$

This very small number stems from the relatively high rate of the code. However, tackling larger distances would require simulating much bigger codes which are beyond our numerical capabilities.

Similarly to the previous subsection, I plot in Fig. 5.11 the logical error rate per round per logical qubit of the [126,28,8] generalised bicycle code and compare it to a distance-3 surface code (the smallest surface code that can correct, not just detect, errors). I set $p_{idle} = 0.01p$ (with p the measurement, initialisation and gate noise) and plot the logical performance of the codes for different amounts of shuttling noise, which are parameterised by the static dephasing time T_2^*. Compared to the previous subsection idling noise does not dramatically impact the bicycle code, as it is formed from matrices with only 5 non-zero diagonals. However, the code is more prone to shuttling errors as the shuttling distance is now of the order of the number of qubits n (while it is of the order $d = \sqrt{n}$ for the surface code).

One can see that for low enough (yet experimentally achievable) shuttling noise, the [126,28,8] code outperforms surface codes with similar qubit overhead for gate noise as high as 0.3%, which is also within reach. If the gate noise can be further reduced to 0.05%, the general bicycle code would provide an advantage of around one order of magnitude compared to a distance-3 surface code (which also uses more qubits than the generalised bicycle code, as we were meant to compare with distance 2). Nevertheless, the performance of the generalised bicycle code is as expected more sensitive to the shuttling noise than the surface code, as it involves longer shuttles. But for the considered code size, the effect only becomes predominant for relatively low dephasing times. Of course, when considering larger codes, the demands on the dephasing time would have to be adjusted accordingly, to compensate for the even longer shuttles.

5.4.3 Comparative performance

I here confirm the observations of the above sections via additional simulations aiming at further understanding the trade-off between resource requirements and error correction performance. Specifically, I examine the performance of the three

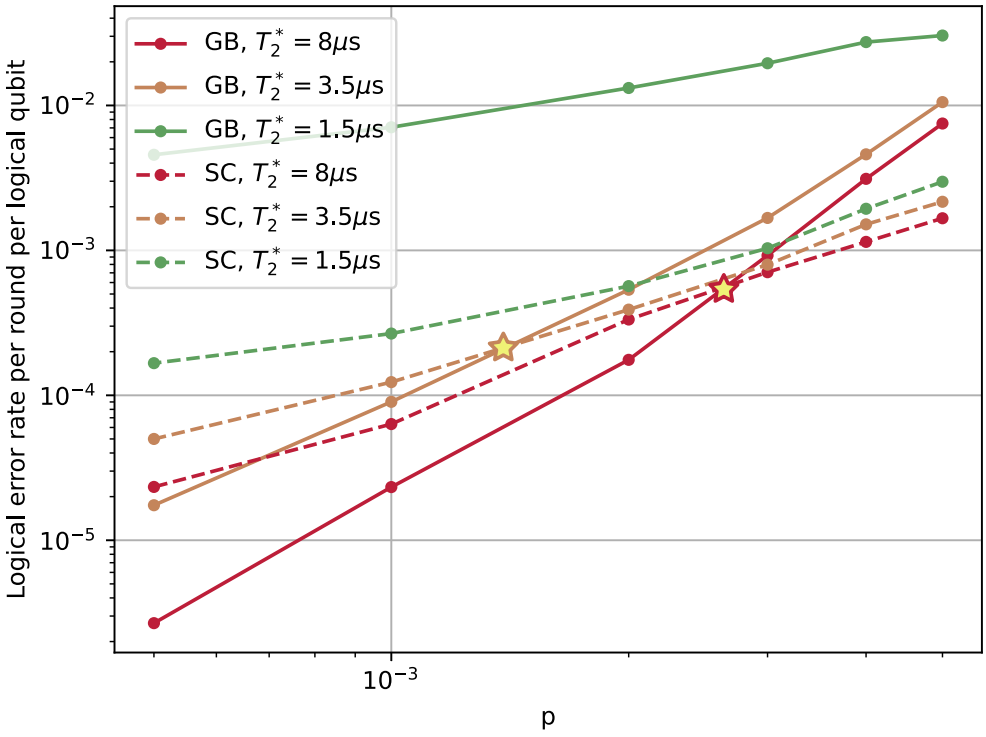


Figure 5.11: Logical error rate per round per logical qubit of a [126,28,8] generalised bicycle code [95] (GB, solid lines) and a distance-3 surface code (SC, dashed lines), against the circuit-level noise parameter p. The logical performance of both codes is plotted for $p_{idle} = 0.01p$ and different levels of shuttling noise, controlled by the dephasing time T_2^*. For $T_2^* = 5\mu$s and 10μs, the noise level where generalised bicycle codes start to beat surface codes is indicated by a star.

previously considered code families under three different noise regimes: (A) low idling noise but high shuttling noise; (B) low shuttling noise but high idling noise and (C) high idling noise and high shuttling noise. The results of such simulations are presented in Table 5.2, where the theoretical parameters of the code are given, along with the logical error rate per round per logical qubit in the three noise cases. I aim to compare codes with similar overhead, defined as the total number of physical qubit per encoded logical qubit *i.e.* $(n + n_{anc})/k$. In the first half of the table, I compare low-overhead codes: a distance-3 surface code (SC3), the [126,28,8] generalised bicycle code considered before (GB8) and a [40,3,4] hypergraph product code (HGP4). Its check matrix is given in Appendix B.3. In the second half of the table, I focus on higher-overhead codes: a distance-7 surface code (SC7)

	$\frac{n+n_{anc}}{k}$	d	Shuttling noise scaling	Idling noise scaling	A	B	C
SC3	25	3	$O(\sqrt{n})$	$O(1)$	2.10^{-4}	1.10^{-4}	$\mathbf{3.10^{-4}}$
GB	9	8	$O(n)$	$O(1)$	7.10^{-3}	$\mathbf{5.10^{-5}}$	7.10^{-3}
HGP4	26	4	$O(\sqrt{n})$	$O(\sqrt{n})$	$\mathbf{4.10^{-5}}$	1.10^{-3}	1.10^{-3}
SC7	169	7	$O(\sqrt{n})$	$O(1)$	1.10^{-5}	$\mathbf{3.10^{-6}}$	$\mathbf{1.10^{-5}}$
HGP8	155	8	$O(\sqrt{n})$	$O(\sqrt{n})$	$\mathbf{8.10^{-6}}$	1.10^{-5}	2.10^{-5}

Table 5.2: Comparative performance of three code families: surface code (SC), hypergraph product code (HGP) and generalised bicycle code (GB). See the main text for the precise definition of each code. The first three and last two rows respectively correspond to lower- and higher-overhead codes, where the overhead is defined as the number of physical qubits per logical qubits. The second column contains said overhead. The third column is the code distance. The fourth and fifth columns contain the shuttling and idling noise scalings of each code. Finally, in the last three columns the logical error rate per round per logical qubit is given for three different noise regimes: (A) $T_2^* = 1.5\mu$s, $p_{idle} = 0$; (B) $T_2^* = 5\mu$s, $p_{idle} = 0.1p$; and (C) $T_2^* = 1.5\mu$s, $p_{idle} = 0.1p$. In each of these columns, the highest-performance low-overhead code, as well as the highest-performance high-overhead code is highlighted with bold letters.

and the [234,3,8] hypergraph product code considered before (HGP8). Note that in this case, I could not simulate *good* generalised bicycle codes due to limited computational power. In each of the last three columns, the lowest error rate is highlighted, confirming the conclusions of the section: each code family considered in this chapter excels in distinct noise regimes. Specifically, GB codes perform best when the shuttling noise is kept low; HGP codes do when the idling noise is low; and the SC wins when neither of them is sufficiently small. In the earliest fault-tolerant regimes, the latter scenario might be the most relevant, which justifies the use of the surface code, for which I also proved full computational power in the previous section.

5.5 Discussion

In this chapter, I have investigated the feasibility of quantum error correction in a highly constrained experimental setup, specifically a 2×N array of qubits where long-range interactions are enabled by shuttling one of rows of the array. By establishing a precise framework for determining code classes that naturally embed

in our device, my co-author Armands Strikis showed that the architectural strong constraints should theoretically not be an obstacle to early fault-tolerance. I then completed this observation with the design of an entire protocol permitting full universal quantum computation with the surface code.

To show the practicality of our approach, I further tailored our protocol to silicon spin qubits, respecting their additional specificities and constraints, such as the difficulty to implement local Hadamard gates. This choice of platform was motivated by the high-fidelity shuttling capabilities that have been demonstrated [34, 35, 37], making it an ideal candidate for the implementation of our scheme. I then confirmed these theoretical protocols with extensive numerical simulations, showing that error rates as low as 10^{-13} can be reached with experimental parameters that have been achieved today, provided currently observed shuttling errors can be further suppressed to match our theoretical models. While this entails long quantum computations for running algorithms in the classically intractable regime, we are confident that the gate and shuttling performance will further improve in the coming years, making our proposal even more practical.

Furthermore, our exploration extends to the application of our device to more intricate qLDPC codes. Although more powerful than the surface code, their implementation comes at the cost of increased noise (either from longer shuttles or higher idling times). By simulating these codes and evaluating their performance in our constrained setup, I observed that this trade-off is in favour of the complex qLDPC codes when error rates are low enough (while still practically achievable). While I focused on specific code families that would naturally suit the 2×N device, my search was not exhaustive. For instance, lifted product codes might be another good fit to the architecture as they inherit the properties from both the bicycle and hypergraph-product codes. Further, later research showed that any qLDPC code could be implemented on the architecture, and gave a protocol for optimising the stabiliser circuit implementation [188]. This underscores the versatility of our platform, even within the limitations of the 2×N qubit array.

As future work, it would be interesting to engineer a slightly more complex architecture design that would let one include both the surface and better qLDPC codes in the same device and interface them. It would indeed prove advantageous to make use of the latter's strong error correction capabilities when used as memory codes. Whenever a logical qubit is idle for a significant amount of time during a quantum computation, one could take its logical information and store it in memory, before taking it out again when needed. If this store in-and-out scheme can theoretically be implemented via a modified lattice surgery protocol between a surface code and another type of qLDPC code [159], one would still need to understand how to efficiently embed this in our device.

More generally, throughout this paper we have chosen to work within quite severe limitations, *i.e.* the low-dimensional array and the restriction to semi-global Hadamard operations. I have established that even within these constraints computation is possible. Nonetheless, without further improvement, scaling up the current device to larger instances may prove very challenging. I identify two reasons for this. First, even for quantum algorithms slightly beyond the classically intractable regime, the run-times reach several days. This is partially due to my sub-optimal choice of logical qubit layout, where the logical information is stored in the first row of surface codes while the second row is only used as a logical ancilla bus (Fig. 5.4). While enabling all-to-all connectivity, this structure also introduces connectivity lockups that slow down the computation: for instance, when the leftmost and rightmost logical qubits are interacting through the ancilla bus, no other interaction can take place in parallel. On top of this, the second obstacle to large-scale computation in our current proposal is the sensitivity of the device to malfunctioning qubits. If a dot becomes inoperable, *e.g.* due to a fabrication defect, and qubits cannot shuttle through it anymore, the array will be cut in two halves that cannot communicate anymore. This would likely lead to a failure of the algorithm.

To circumvent both the aforementioned issues, I envisaged an extension of the strictly 2×N qubit array presented in this paper, with a piece of research that is outside the scope of this thesis [165]. This so-called Quantum Snakes on a

Plane architecture is constituted of a lattice or web-like structure where long 2×N filaments meet at occasional three- or four-way junctions through which qubits can shuttle. Such junctions can be well-spaced, so that the additional device complexity associated with each junction need not overlap with others. In addition to avoiding the problem of a single point of failure, such a geometry can afford a rich space of possibilities for compilation strategies and novel codes (it inherits the connectivity of the 2×N device with even higher design flexibility). They additionally offer superior opportunities to interface the aforementioned better qLDPC codes with surface codes.

5.6 Related work I was involved in: the looped pipeline architecture

In the final section of this chapter, I briefly describe a related project I was involved in before working on the 2×N architecture. The motivation for the research so far presented in this chapter was to simplify the implementation of error correcting codes and offer opportunities for their practical production in the near term. The key ingredient that enabled such simplifications is shuttling. Now, instead of using this strong capability to simplify codes for near-term usage, one could instead capitalise on it to build complex structures for a later stage. This is the reasoning behind the original idea of Zhenyu Cai and Simon Benjamin in [156]. In this paper where I contributed as the second author, shuttling is utilised as a means to turn 2D devices into 3D qubit lattices — while the research of this chapter focused on turning augmented 1D devices (2×N arrays) into 2D qubit lattices.

The principle of such procedure is summarised in Fig. 5.12. The 3D qubit lattice (Fig. 5.12(a)) can be fully embedded in a 2D device requiring nearest-neighbour interactions only if extensively making use of shuttling. The idea is to place qubits in looped shuttling tracks where they can be shuttled around. The number of qubits in each shuttling track is the number of layers of the original 3D lattice. All nearest-neighbour interactions in the 3D lattice can then be implemented in the 2D device by bringing the spins closer together. Let us focus on intra-layer

interactions first (Fig. 5.12(b)). If the movement is made synchronous, qubits from the same layer can periodically meet at interaction points (where shuttling tracks meet). For instance, at the time pictured in the figure, the gray and top left qubits of the purple layer can interact, as well as the gray of top right qubits of the brown layer, and so forth. As qubits move around, the gray and top right purple qubits will be allowed to interact etc. Similarly, inter-layer interactions can be implemented simply by bringing qubits from two different layers but belonging to the same shuttling track closer together (Fig. 5.12(c)).

These basic principles thus enable the implementation of a nearest-neighbour-interaction 3D qubit lattice within a strictly 2D device equipped with shuttling. If thinking of the gray qubits of Fig. 5.12 as ancillas, this method hence allows for the compact embedding of multiple error correcting codes within a 2D device by utilising the intra-layer (inter-loop) interactions (Fig. 5.13). The inter-layer (intra-loop) interactions can in turn be employed to perform transversal gates. Alternatively, a single 3D code could be embedded in this device by making use of the intra-layer as well as inter-layer interactions for stabilisers.

I contributed to this project by numerically showing the practicality of such an approach. The noise model we adopted was similar to the one I used in the rest of this chapter yet slightly more pessimistic. Three noise mechanisms were considered: first adiabatic shuttling causing dephasing errors with strength p_{sh}; second non-adiabatic shuttling giving rise to leakage to the excited valley state with probability p_{leak}; third Rashba spin-orbit interactions inducing X and Y errors with probability p_{rash} [189]. For the valley noise, instead of using the further modelling I later carried out in Appendix B.2, my co-author Zhenyu Cai and I decided to go for a worst-case assumption: a leaked qubit would cause the full depolarising of all the qubits it interacts with throughout a stabiliser cycle. Then the low valley lifetime (around 10ns [190, 191]) would bring the qubits back from the excited to the ground valley state by the end of a stabiliser cycle. The full noise model, including initialisation, measurement and gate noise, is summarised in Fig. 5.14. Like in the rest of this chapter, twirling gates are used to transform coherent rotations into

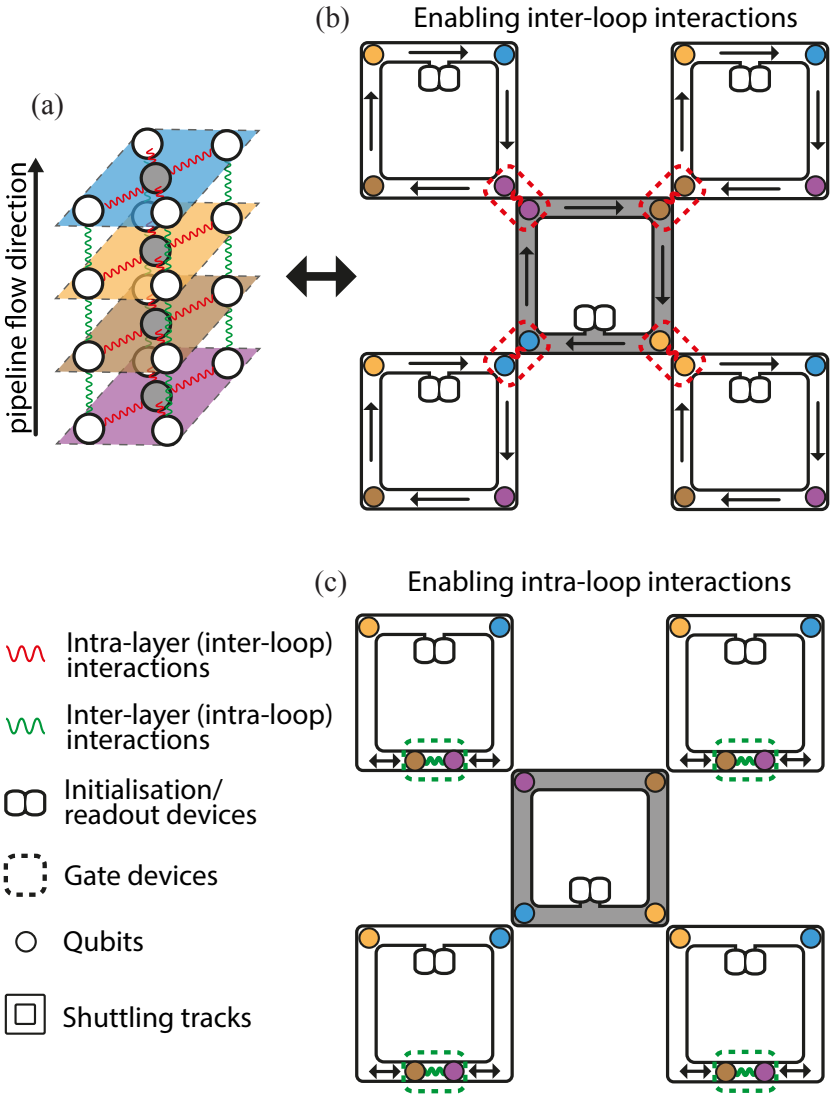


Figure 5.12: Embedding of a 3D qubit lattice (a) into a strictly 2D device by placing the spins in looped shuttling tracks. (b) Intra-layer interactions in the 3D lattice are enabled by inter-loop interactions in the 2D device. (c) Inter-layer interactions in the 3D lattice are enabled by intra-loop interactions in the 2D device. This figure was originally produced by Zhenyu Cai.

stochastic errors. The result of threshold simulations I carried out are presented in Fig. 5.15. Note that contrary to the study of the 2×N architecture, this proposal does lead to thresholds as the length of the loops is independent of the code distance d (the loop length is only related to the number of surface code patches one can load). For a gate noise of $p = 0.5\%$, one can read a shuttling noise threshold of 0.36%, which is above demonstrated shuttling infidelities now reaching 10^{-4} [37]. The leakage threshold is 0.04% which is also comfortably above the estimated leakage

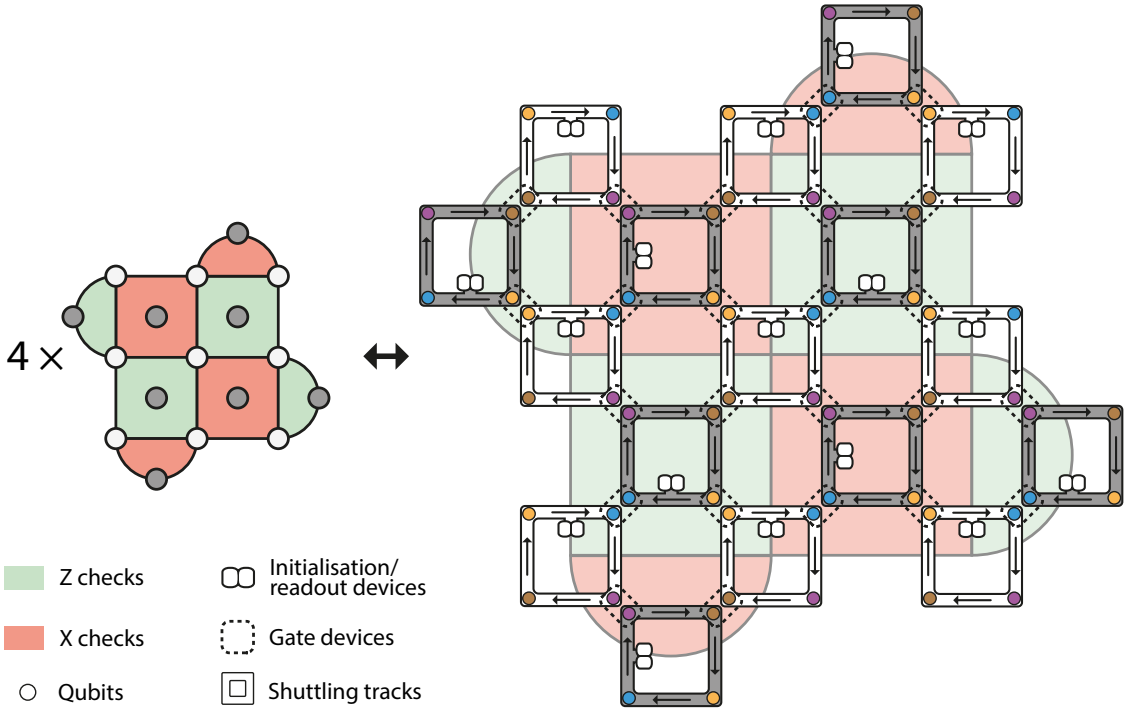


Figure 5.13: Embedding of four surface code patches in the looped pipeline architecture. This figure was originally produced by Zhenyu Cai.

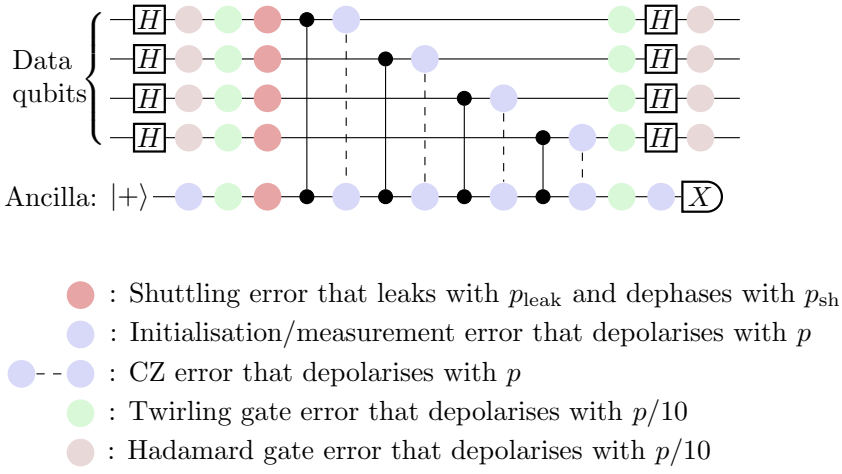
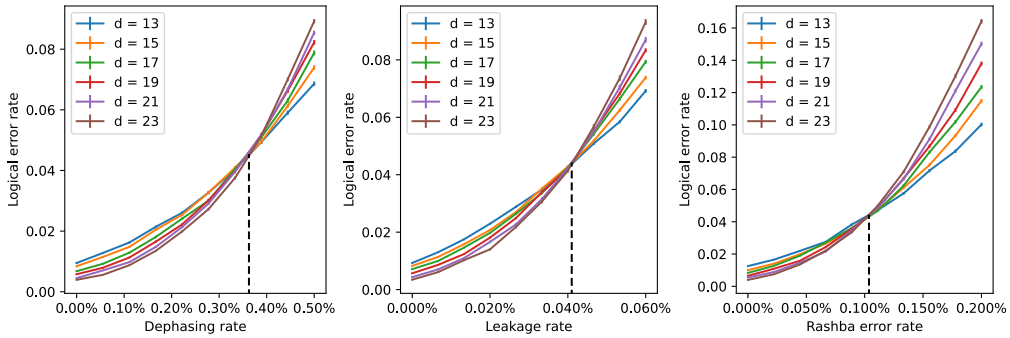


Figure 5.14: Noise model for the looped pipeline architecture. This figure was originally produced by Zhenyu Cai.

probabilities estimated in Appendix B.2. Finally, the spin-orbit-coupling-induced threshold is 0.1% which is of the same order as the shuttling noise; it should thus not pose fundamental limitations. If gate noise is further reduced towards $p = 0$, these thresholds in turn increase, leading to even higher error reduction.

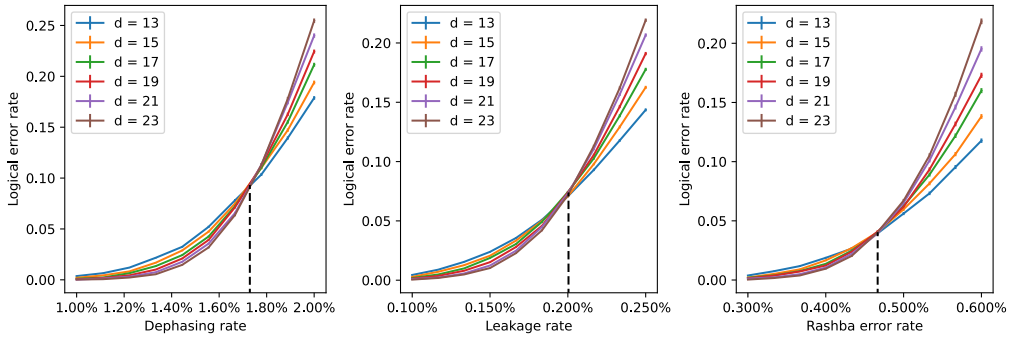
Therefore, the kind of architecture presented in this section lies on the other end



(a) Threshold for p_{sh} at $p = 0.5\%$, $p_{\text{leak}} = p_{\text{rash}} = 0$

(b) Threshold for p_{leak} at $p = 0.5\%$, $p_{\text{sh}} = p_{\text{rash}} = 0$

(c) Threshold for p_{rash} at $p = 0.5\%$, $p_{\text{sh}} = p_{\text{leak}} = 0$



(d) Threshold for p_{sh} at $p = p_{\text{leak}} = p_{\text{rash}} = 0$

(e) Threshold for p_{leak} at $p = p_{\text{sh}} = p_{\text{rash}} = 0$

(f) Threshold for p_{rash} at $p = p_{\text{sh}} = p_{\text{leak}} = 0$

Figure 5.15: Threshold plots for the dephasing errors (p_{sh}), leakage errors (p_{leak}) and the Rashba error (p_{rash}) due to shuttling when implementing surface codes in semiconductor spin qubit using shuttling-based architectures. Note that in order to investigate the tolerance of the surface code against a given specific type of shuttling noise, the other types shuttling noise are turned off in our simulation. Each noise type is evaluated both for a gate error rate p of zero, and for $p = 0.5\%$.

of the complexity spectrum, compared to the 2×N architecture. It is based on the same concept — periodically shuttling qubits — yet executes it in a different fashion. In the 2×N architecture, the density of the qubit lattice is reduced by shuttling data qubits throughout long distances: this enables a simpler implementation of codes of moderate complexity in the near term. On the contrary, in the looped pipeline setup, the density is increased by loading more qubits at each lattice site without the need to shuttle them far from their original position: this will allow for the implementation of more complex error correction protocols in the longer run.

6

Harnessing qubit transport for global spin qubit control

The research presented in this chapter was born from an original idea of Hamza Jnane consisting of swapping or shuttling qubits to effectively average their g-factors. All other theoretical background was jointly established by Hamza and me through many long discussions. As for the numerical results, I produced all of the code, that both Hamza and I ran. Fernando Gonzalez helped us by providing the useful outlook of an experimentalist.

All writing below is my own, in places using text verbatim from [192].

Contents

6.1 Introduction

In the previous chapter, I presented a scheme enabling full quantum computing power for nearer-term silicon spin qubit architectures (using a single-spin encoding of the qubits). In particular, one of the constraints that I considered in my analysis was the use of (semi) global single-qubit gates only. This relatively restrictive assumption arose as, for qubits encoded with a single spin, single-qubit gates are implemented via the application of a magnetic field: resonantly targeting a spin qubit with an AC field gives rise to Rabi oscillations of the qubit around the Bloch sphere (see Section 2.2.3). These fields can be applied locally through surface gate electrodes, but this approach, applied at scale, presents formidable challenges for signal delivery. Instead, a global control field can be applied and selected spins can be brought in and out of resonance by means of frequency tuning [171, 193–195]. Unfortunately, common frequency tuning techniques are unable to fully correct for the spread of qubit frequencies that are typically found in devices due to variations on the local spin-orbit fields [196–198].

To ameliorate the situation, recent studies have shown that the spin qubit frequency spread can be reduced by choosing the angle of the static magnetic field with respect to the crystallographic axes of the sample [189, 194]. Complementarily, one can tune the qubit frequencies using electric-field-induced Stark shifts, but its limited tuning range – one order of magnitude smaller than the typical frequency spread – forces further measures. A partial solution proposed tuning the g-factors using Stark shifts to place the qubits into a number of distinct-frequency bins, thereby mitigating the effect of frequency crowding. In this way, the minimum frequency spacing does not decrease with the system size [28, 195]. By sending tones at resonance with each bin, selected qubits can be driven and the desired global single-qubit gates can be performed. Reference [195] extended this scheme and suggested a way to drive not all but a single qubit, by adding SWAP gates. However, none of these works allows for the reliable implementation of single-qubit gates on any subset of qubits, without halving the already critical bin spacing.

Here, my co-author Hamza and I propose to solve the challenge of limited frequency tuning range by leveraging electron motion through the spin-qubit nanodevice. Namely, we show that one can reduce the number of distinct frequencies in the frequency spectrum by homogenising the electrons g-factors via spin shuttling or via exchange-driven two-qubit SWAPs. More specifically, if an electron experiences a collection of g-factors at a rate that is sufficiently fast compared to the Rabi frequency and the dispersion of the qubit frequencies, the electron will acquire an effective homogenised g-factor equal to the mean of all experienced g-factors. While both methods leverage the same simple idea, we find that the shuttling-based homogenisation is more scalable, and can additionally be utilised for the implementation of two-qubit gates. More than a key component for the development of fault-tolerant architectures [155, 156, 165, 199, 200], spin shuttling turns out to be an enabling tool in the search for homogeneous devices. It is worth mentioning that this idea is reminiscent of the frequency narrowing phenomenon [36]; however, this concept was never used to perform quantum gates. In this work, we concentrate on electron spin qubits in silicon because of their small spin-orbit fields. However, as we shall see later, this work can be extended to strong spin-orbit systems that present large variations in their spin quantisation axes.

The chapter is organised as follows. In Section 6.2, I introduce our model for the qubits. Thereafter, in Section 6.3 I abstractly describe the homogenisation principle before giving two methods to physically implement it, capitalising on the exchange coupling or shuttling. Focusing on the more promising shuttling-based protocol, I then explore its performance for two architectures, namely the 2×N [155, 188] and looped pipeline architectures [156] in Section 6.4. Finally, I present our conclusions in Section 6.5.

6.2 Preliminary

In all the following, I set $\hbar = \mu_B = 1$, thus identifying energies, frequencies and magnetic fields. The system can be described by the following Hamiltonian:

$$H = \sum_{<i,j>} J_{i,j} \vec{S}_i . \vec{S}_j + \sum_{i=1}^{n_q} g_i \vec{S}_i . \vec{B} \tag{6.1}$$

where n_q is the total number of qubits, $J_{i,j}$ is the tunable exchange interaction between neighbouring spins $\vec{S}_i$ and $\vec{S}_j$. The exchange coupling strength can be tuned in-situ via electric fields and whose typical on-state value ranges between 10 and 1000 MHz [30, 42, 201–204]. The field $\vec{B} = \sum_j \vec{B}_j$ is a global magnetic field, in the sense that it is applied across the entire device. The field is composed of a single static component $B_z = B_0$, and an oscillatory component $B_x = B_1 \sum_j \cos(\omega_j t)$. Typical experimental values for B_0 range from 0.1 T to 1.4 T. It is favourable to use values from the low end of the range to reduce the qubit energy spread (which will be beneficial for our homogenisation protocol). However, this complicates the task of single-qubit addressability owing to reduced frequency spreading, which justifies the design of more advanced schemes for handling frequency crowding. I therefore set $B_0 = 0.1$ T in the rest of this chapter unless otherwise specified. The oscillatory component of the drive will lead to magnetic dipole transitions with an amplitude Ω, whose value is usually around 1 to 5 MHz [30, 42, 44, 201, 205]. Additionally, the qubits can be shuttled along the dots at a speed v between 10 and 50 m/s [34, 35, 37]. I further assume an interdot spacing $d = 100$ nm, resulting in a shuttling frequency $v/d = 100$ up to 500 MHz. As for the g-factors, g_i, their distribution is complex to model and currently an active field of research. I thus elaborate on this matter in Appendix C.2. Note that in the common case of a single excitation frequency, the Hamiltonian of Eq. (6.1) is made time-independent by studying it in the rotating frame at the frequency of the drive. However, it is not possible to perform such transformation here as we are driving the spins with multiple tones ω_j.

Finally, it is important to note that the system is most accurately described by considering the full g-tensor in Eq. (6.1), which can effectively change the axes along which the effective static and oscillatory magnetic fields are applied [38]. I

Parameter	Notation	Order of magnitude
Time resolution	$1/\tau_r$	10 GHz
Zeeman splitting	$\omega_q/2\pi$	3 GHz
Shuttling speed	v/d	100-500 MHz
Exchange coupling	$J/2\pi$	10-1000 MHz
Average qubit splitting	$\Delta\omega_q/2\pi$	3-30 MHz
Drive strength	$\Omega/2\pi$	1-5 MHz
Stark-shift tunability	$\delta\omega_q/2\pi$	0.3-3 MHz

Table 6.1: Relevant parameters and frequency scaling at $B_0 = 0.1$ T.

show in Appendix C.3 that this has minimal impact on the fidelity of a single-qubit gate for electron spins in silicon, even when the static field includes components along the x and y axes. Therefore, in the rest of the paper, I will assume that the static and oscillatory fields are indeed applied along the z and x axes, respectively, and that g is a scalar. However, our scheme could be applied to more general species of spin qubits and semiconducting hosts. In these cases – where spin-orbit coupling effects are stronger – variations of the quantisation axes with the electron motion induce unwanted unitary gates. Fortunately, these could be calibrated away provided prior knowledge of the g-tensor [206]. I leave this study for future work and focus here on electron spins in silicon.

In Table 6.1, I summarise the parameters used in this study in decreasing order of magnitude considering $B_0 = 0.1$ T. The apparatus time resolution τ_r is added, as an upper (frequency) limit for all other parameters. The ratio between these parameters will be relevant for the characterisation of the performance of our scheme.

6.3 Motion-based single-spin control

If an electron explores a collection of g-factors at a rate that is much faster than the gate frequency and the qubits frequency dispersion, its g-factor will effectively be homogenised. Doing so, reduces the dispersion of the frequencies, thereby facilitating the implementation of global gates by sending a single driving tone to distinct target qubits.

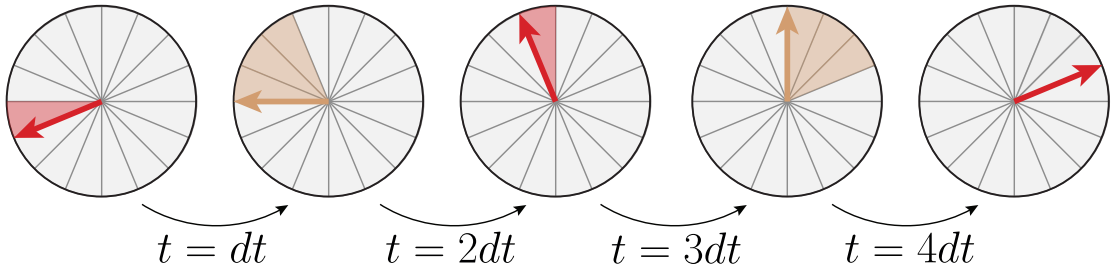


Figure 6.1: Schematic time-evolution of a single qubit oscillating between two quantum dots with distinct frequencies $\omega_{q,1}$ (red) and $\omega_{q,2}$ (yellow). We assume that the transfer rate between the dots is large enough that the electron instantaneously swaps position after each time step. Assuming that it starts in the first dot, the qubit rotates by one tile and ends up in the second dot after one time step. Next, it rotates by three tiles and comes back to the first dot. Finally at $t = 4dt$, the qubit is back to its original position and has covered eight tiles. This leads to the conclusion that the resulting dynamics is a precession of the qubit at a frequency $\bar{\omega}_q = (\omega_{q,1} + \omega_{q,2})/2$.

To provide a simple example, consider a single electron trapped in a double quantum dot (DQD). Suppose that the left and right dots have g-factors g_1 and g_2, respectively. In the laboratory frame of reference, the qubit would precess with a different frequency (set by the g-factor) depending on its location. Now suppose that the qubit can instantaneously be transferred from one dot to the other at every time step. The resulting time dynamics is schematically represented in Fig. 6.1. One can see that, because of the transfer, the qubit now rotates with an average frequency $\bar{\omega}_q = (\omega_{q,1} + \omega_{q,2})/2$. This leads to the conclusion that the qubit now has a modified resonant frequency equal to $\bar{\omega}_q$. Numerical simulations in the next section in the presence of a drive will support this hypothesis for finite transfer rates. Below, I report two different physical mechanisms (see Fig. 6.2) that could be leveraged to homogenise g-factors and hence qubit frequencies following the above intuition.

6.3.1 Exchange-based homogenisation

Single-qubit gates on two electrons spins

One way to implement the example presented in Fig. 6.1 is to make use of the exchange interaction as shown in Fig. 6.2(a). The transfer rate corresponds here to the exchange coupling J. By turning on J, qubits in neighbouring quantum dots

start to swap information. This process occurs at a finite rate and will thus lead to a non-perfect homogenisation of the qubits resonance frequencies.

Let us add an oscillatory magnetic field with frequency $\omega = \bar{\omega}_q$. If the intuition exposed previously is right, one would expect to perform a perfect X-rotation on both qubits. This is confirmed by the numerics of Fig. 6.3. Starting from the arbitrary initial state,

$$|\psi_0\rangle = |0\rangle \otimes \left(\sqrt{\frac{3}{4}}\,|0\rangle + \sqrt{\frac{1}{4}}\,|1\rangle\right), \tag{6.2}$$

I plot the expectation value of σ_z over time for both qubits, at a finite $J = 20\,\omega_q^{12}$ with $\omega_q^{12} = (\omega_{q,2} - \omega_{q,1})$, additionally choosing $\omega_q^{12} = \Omega$. These simulations are simply obtained by solving Schrödinger equation by time discretisation, using 10,000 time steps. The red and yellow faded lines respectively represent the full time-evolution of $\langle\psi(t)|\,\sigma_z \otimes I\,|\psi(t)\rangle$ and $\langle\psi(t)|\,I \otimes \sigma_z\,|\psi(t)\rangle$. These feature fast oscillations at a frequency J, which corresponds to the swapping of the qubits, and slower oscillations at frequency $\omega = \bar{\omega}_q$, which are triggered by the driving field. When sampling points from the red and yellow faded data every $2\pi/J$, one obtains the solid lines, which qualitatively look like an X gate on both qubits. This means that, at finite but large enough J, the individual qubits indeed acquire the same effective frequency $\bar{\omega}_q$ and are thus resonantly driven by a single drive at $\omega = \bar{\omega}_q$. Unfortunately, the exchange interaction can entangle the qubits. Therefore, in order to implement an X gate on both qubits without any additional entanglement creation, one must properly tune J such that the qubits go back to their original positions in the time taken by the X gates. This condition reads:

$$\frac{2p\pi}{J} = \frac{\pi}{\Omega}, \quad p \in \mathbb{N}$$

or equivalently:

$$J = 2p\Omega, \quad p \in \mathbb{N}. \tag{6.3}$$

In order to further estimate the closeness of the implemented gate U to the target unitary $U_{\text{target}} = X \otimes X$, one can evaluate the average single-qubit

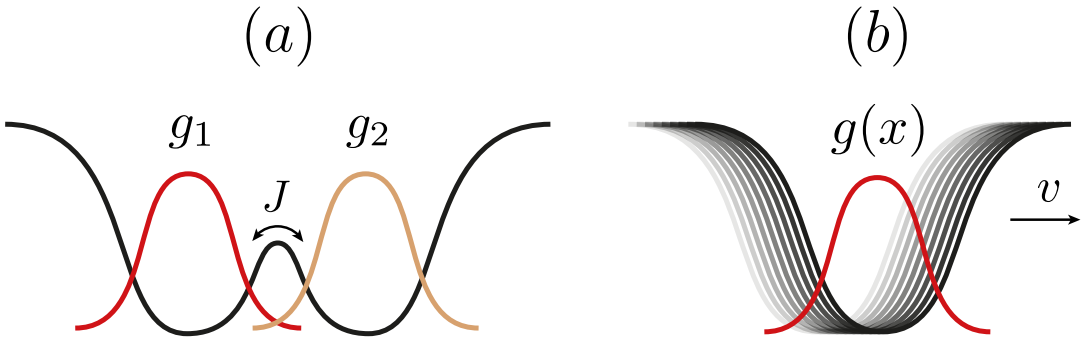


Figure 6.2: Physical implementations of the motion-based g-factor homogenisation. In (a), we consider two quantum dots with respective g-factors g_1 and g_2. The exchange coupling J is used to homogenise the frequencies of the two electrons. This allows to reduce the number of distinct frequencies in the spectrum and thus reduce crosstalk. In (b) we use a completely different method in which we physically move the electron to perform the homogenisation. Here, the g-factor evolves continuously along the shuttling path. We find that this method is the most powerful as it can be generalised to an arbitrary number of qubits and scales better. Hamza produced this figure.

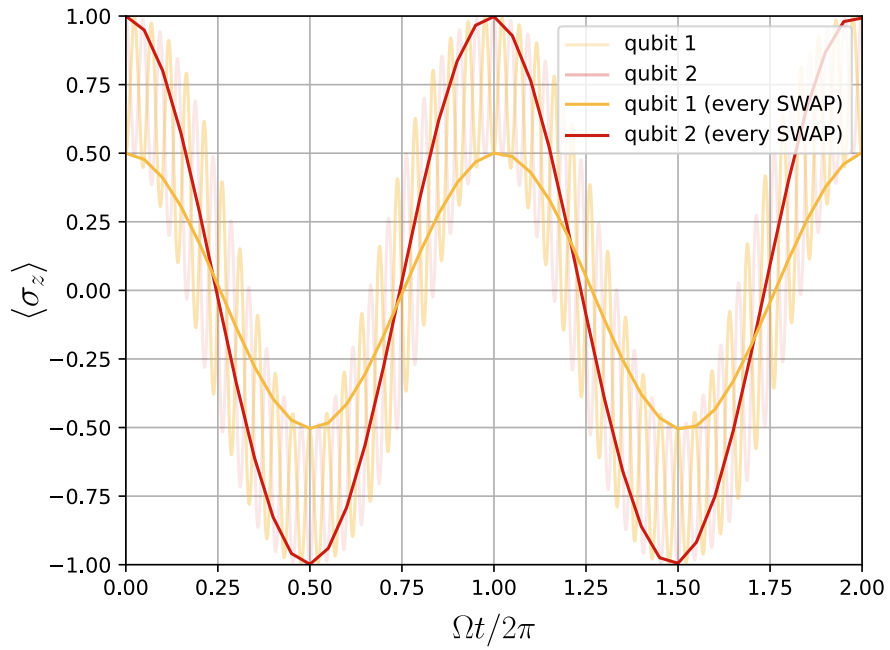


Figure 6.3: Time evolution of $\langle\sigma_z \otimes I\rangle$ and $\langle I \otimes \sigma_z\rangle$ under the two-qubit Hamiltonian of Eq. (6.1). I set $\omega_q^{12} = \Omega$ and $J = 20\,\omega_q^{12}$ where $\omega_q^{12} = (\omega_{q,2} - \omega_{q,1})$. The initial state is given in Eq. (6.2). The faded lines correspond to the full time evolution, while the solid lines are obtained from sampling the full data every $2\pi/J$.

fidelity $\mathcal{F}$, defined as

$$\mathcal{F} = \frac{1}{4}\left|\mathrm{tr}\left(U_{\mathrm{target}}^{\dagger}\tilde{U}\right)\right|^{2/n_q} \tag{6.4}$$

with $n_q = 2$ and $\tilde{U}$ is a slightly modified version of the time evolution operator U (where I removed *known but unwanted* Z rotations) as explained in Appendix C.4. This quantity is dimensionless and thus only depends on the ratios of the three parameters dictating the dynamics of the two-qubit system: J, Ω and ω_q^{12}. Only the difference between $\omega_{q,1}$ and $\omega_{q,2}$ matters, as their absolute values can be absorbed in a change of frame.

If the condition in Eq. (6.3) is always enforced, one can expect the infidelity $1-\mathcal{F}$ to decrease to 0 when J increases: this is confirmed by the numerics of Fig. 6.4, which I ran. One can also observe that when ω_q^{12} is decreased, the infidelity decreases as well: this is expected as the limiting case of $\omega_q^{12} = 0$ should yield a null infidelity. Now, plugging in the experimental values from Table 6.1, one approximately obtains $J/\Omega = 100$ and $\omega_q^{12}/\Omega \gtrsim 1$. This results in an infidelity between 0.01% and 0.1%.

In order to better understand what limits the fidelity here, I explore the Schmidt decomposition of U in Appendix C.5. In particular, I show that at finite transfer speed J, the time evolution operator U is indeed creating entanglement between the qubits, and cannot be decomposed into a product of two single-qubit gates.

Single-qubit gates on more than two electrons

The extension of the SWAP-based scheme to more than two qubits is not trivial. Indeed, one would naively think that turning on the J-coupling between all adjacent dots would produce the same desired outcome. Unfortunately, multi-qubit SWAP gates are not implemented by simply turning on all J-couplings. Rather, one could think of implementing a multi-qubit SWAP gate by decomposing it into two-qubit SWAPs, which could be performed by turning the J-couplings on and off in an alternate fashion. If this method theoretically works, it is however impractical.

One of the most significant sources of errors in implementing SWAP gates is charge noise, which arises from two-level fluctuators (TLFs) [207–209]. These TLFs

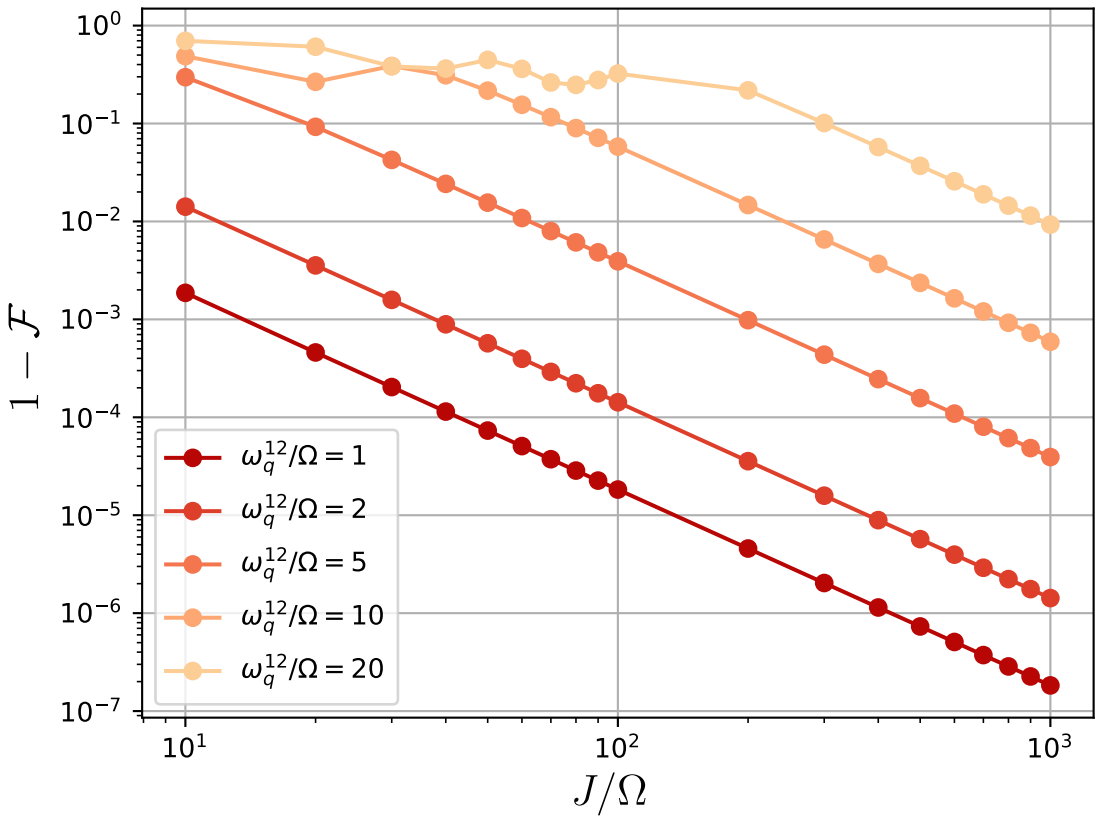


Figure 6.4: Infidelity of the swapping protocol. It is computed as the infidelity of implementing an $X \otimes X$ gate when driving two continuously-swapped qubits with frequencies $\omega_{q,1}$ and $\omega_{q,2}$ with a single driving tone at $\omega = \bar{\omega}_q$. The data is plotted against J/Ω and ω_q^{12}/Ω, where J is the exchange coupling and $\omega_q^{12} = \omega_{q,2} - \omega_{q,1}$. J is always chosen such that the condition in Eq. (6.3) is satisfied.

lead to unknown modifications in the exchange coupling [202–204], leading to timing errors that reduce the fidelity. When decomposing a multi-qubit SWAP gate into two-qubit SWAPs, a large number of sequential SWAPs between different qubits are required. This results in an exponential decrease in overall fidelity with the number of SWAPs, rendering the protocol impractical. However, in the two-qubit case, we only need to turn the exchange coupling on and off once, so the timing error only affects the process a single time. Moreover, since the exchange is active for an order of microseconds, charge fluctuations on this timescale should average out the variations of the exchange coupling, leading to higher fidelity [204]. Finally, it is important to note that as charge noise modifies the exchange coupling J, it will lead to a violation of the condition in Eq. (6.3). As this chapter focuses on

addressability, the impact of charge noise on our scheme is left for further study.

6.3.2 Shuttling-based homogenisation

Single-qubit gate on a single electron

In the previous section, I focused on a mechanism that allows for a qubit to oscillate between two regions of fixed g-factors. In this section, I advocate for the use of a more powerful mechanism, which leads to a broader exploration of the g-factor landscape: electron shuttling. Here, the electron is physically moved thanks to the electrical control of the trapping potential and can thus explore a continuous landscape of g-factors as illustrated in Fig. 6.2(b). Moreover, this scheme can easily be generalised to more than two qubits, as multiple electrons can be shuttled at the same time without any entanglement being generated between them.

There are two main ways to perform electron shuttling: the bucket brigade mode [34, 178, 210, 211], in which the electron is moved from one dot to another by altering the detuning between them and the conveyor-belt mode [35–37, 199, 212–214], in which it is the trapping potential itself that moves, carrying the electron with it. Here, I will focus on the latter as it is more advantageous in terms of scalability [37, 199], and recent experiments showed that coherent spin shuttling in Si/SiGe can be achieved with fidelities as high as 99.99% for gate-to-gate distance and around 99% for a distance of 10 μm [37]. As in the previous sections, an electron shuttled in a region of varying g-factor will acquire an average effective frequency if shuttled fast enough, with the difference that we are now considering a continuously varying g (see Eq. (C.3)). This frequency narrowing phenomenon had in fact already been observed in the context of shuttling, but had never been used for single-qubit addressability [36].

More precisely, during the gate time $T = \pi/\Omega$, an electron with trajectory $x(t)$ will obtain the homogenised g-factor $\bar{g}$ given by,

$$\bar{g} = \frac{1}{T}\int_0^T g(x(t))dt. \tag{6.5}$$

Contrary to the exchange-based protocol, it seems that a perfect knowledge of $g(x(t))$ is now required at each time t in order to find the drive frequency. While this would be quite challenging, one can rather directly determine the drive frequency by shuttling the electron along $x(t)$ and by sweeping the frequency of the drive ω until resonance with the qubit is obtained [215]. One can then deduce $\bar{g}$ from ω as $\omega = \bar{g}B_0$.

However, depending on the shuttling trajectory $x(t)$, such characterisation might not be needed. Suppose that we let the electron explore dots without repetitions, *e.g.* by shuttling it on a straight line for a time T. As the travelled distance increases, the homogenised g-factor $\bar{g}$ given by Eq. (6.5) will tend towards the device average g-factor g_0. For a sufficiently long distance, we expect that no calibration is needed and that the sole knowledge of g_0 is required which could be obtained through spin dependent scattering experiments on large samples [216]. This observation allowed Hamza and I to propose two different driving modes: mode I, for which we drive the qubit at a frequency $\omega = \bar{\omega}_q = \bar{g}B_0$; and mode II, where $\omega = \omega_0 = g_0 B_0$.

In the rest of the section, the evolution of the infidelity associated with both driving modes is presented. Let us suppose that we shuttle a qubit back and forth and that its movement is described by a triangle wave, recursively defined as:

$$x(t) = \begin{cases} d/2 - vt, & \text{if } t < d/v \\ -3d/2 + vt, & \text{if } d/v \leq t < 2d/v \\ x(t - 2d/v), & \text{if } t \geq 2d/v \end{cases} \tag{6.6}$$

with d representing the distance travelled by the electron for one way and v is its (constant) shuttling speed. Furthermore, I fix $\Delta g = 10^{-3}$ to address cases of higher frequency crowding.

To guide the analysis of the numerical simulations of the infidelity of our protocol, it is important to note that it suffers from two main sources of errors: the frequency homogenisation error and the error arising from off-resonant driving. While the latter is well understood and explained by the Rabi model, I need to describe the impact of v and d on the former. In Fig. 6.5, I use Gaussians centred around the mean g-factor to symbolise the quality of the shuttling-based frequency

homogenisation (in the spirit of frequency narrowing). Considering some distance d, and a constant speed v in Fig. 6.5 (a), the homogenisation is not perfect as illustrated by the wide distribution around the g-factor associated with the drive frequency $\omega = \bar{g}_d B_0$. While keeping d fixed, increasing v in Fig. 6.5 (b) yields a better homogenisation as explained in previous sections. The faded Gaussian, corresponding to a slower speed, is thus less peaked. Keeping v fixed, choosing a distance $d^{'} > d$ in Fig. 6.5 (c) reduces the quality of the protocol as the electron has to explore more dots. Moreover, as $d \to \infty$ we have $\bar{g}_d \to g_0$, which explains the shift of the mean frequency. The faded Gaussian, corresponding to a smaller distance, is therefore more peaked but its mean is farther from g_0.

We are now ready to study the infidelity of an X gate performed on a qubit as a function of d, for different speeds v and drive amplitudes Ω as plotted in Fig. 6.6 (which Hamza ran using my code). The distance d ranges from 100 nm (the interdot distance) to the maximum distance the electron can travel during the gate time *i.e.* $L = vT$. The infidelity measure is the same as for the exchange-coupling-based scheme – further details on this metric and its numerical simulation can be found in Appendix C.4. Each data point is obtained by averaging 20,000 instances of g-factors generated using the method presented in Appendix C.2.

For mode I (top of Fig. 6.6), in which we drive at the mean frequency $\omega = \bar{g}_d B_0$ ($\bar{g}_d$ being the mean g-factor for a distance d), the infidelity increases with d for fixed v and Ω. This is characteristic of a degradation of the homogenisation in the presence of a larger number of distinct g-factors, as explained before. Then, we notice that the curves can be grouped by their speed v, with the infidelities for $v = 50$ m/s (red) approximately an order of magnitude smaller than for $v = 10$ m/s (yellow), in agreement with our predictions. The larger the speed (or transfer rate) v, the better the homogenisation. Finally, we note that within each group, a change in Ω (solid versus dashed lines) does not have much impact on the infidelity, at least in the experimentally relevant parameter regime we study. Indeed, when looking at the Rabi model, the value of Ω does not affect the infidelity when driving on resonance.

For mode II (bottom of Fig. 6.6), in which we drive at $\omega = g_0 B_0$, the infidelity decreases with d unlike mode I. Indeed, as d gets larger, $\bar{g}_d$ tends to g_0 as showed before, thereby reducing the detuning between the drive and qubit frequency. While we can also group the curves in pairs here, it is now the driving amplitude that discriminates them. Within the Rabi model, this type of error solely depends on the ratio $|\omega - \omega_q|/\Omega$. The larger the ratio, the higher the fidelity. That is why the infidelity decreases as Ω increases. While increasing the speed should also improve the fidelity (for the same reasons as for mode I), the plot shows that the impact of v is minimal: this means that we are dominated by off-resonant driving errors here and that homogenisation errors are negligible. Only d and Ω thus matter in this regime. Finally, note that the infidelities obtained with mode II are not nearly as good as the ones from mode I. This is an artefact of these simulations where only one electron is driven. For this reason, it is obviously much simpler to leave a qubit idle and drive it at its own static resonant frequency rather than at the average device frequency. However, when considering multiple qubits, mode II will become more advantageous as it will allow one to drive many qubits at once without any calibration and with a single driving tone. I will confirm this statement in Section 6.4.

As our aim was to characterise the quality of the homogenisation, shuttling errors were not included in the previous simulations. If they were, it would be necessary to consider a trade-off between shuttling noise and homogenisation error for mode II. Indeed, as the distance increases, the infidelity of our protocol decreases while shuttling noise would increase. One would need to find a sweet spot balancing the impact of these two error sources. For mode I, increasing d decreases the infidelity, so including shuttling noise would only aggravate the situation. As a reminder, recent experiments in Si/SiGe showed per-increment shuttling fidelities being progressively reduced from 10^{-3} to 10^{-4} [34, 35, 37]. As one hop can be implemented in $\sim$10 ns (assuming an interdot distance of 100 nm and a shuttling speed of 10 m/s), the resulting overall shuttling noise over one X gate (lasting 1 μs for $\Omega = 1$ MHz) reaches at best 10^{-2} with current technologies. However, the task of low-noise shuttling is receiving rapidly increasing attention, notably with recent

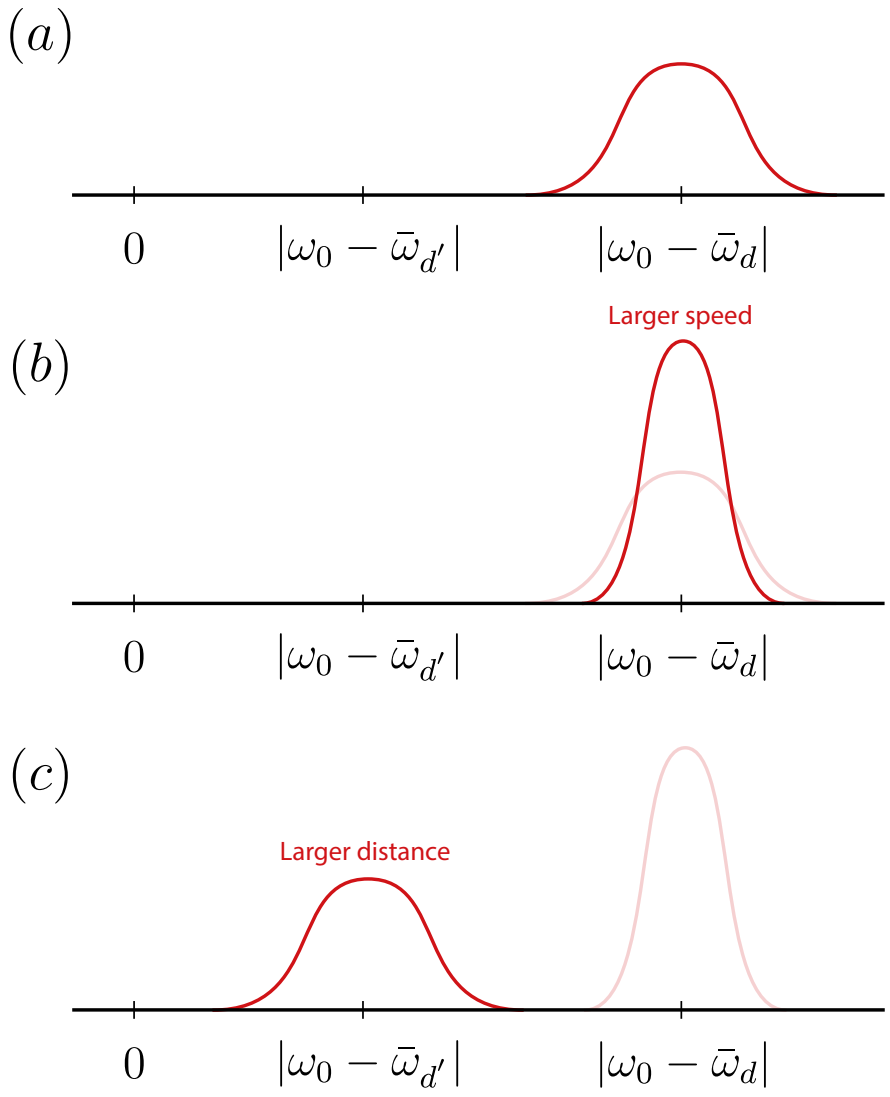


Figure 6.5: Illustration of the impact of speed and explored distance on the quality of the homogenisation process. The x-axis represents the difference between the electron's homogenised frequency $\bar{\omega}_d$ when shuttled for a distance d and the device-averaged one $\omega_0 = g_0 B_0$. (a) Wide distribution around the mean frequency representing an imperfect homogenisation. (b) Increasing the speed (*i.e.* the transfer rate) while keeping the explored distance fixed improves the quality of the homogenisation as explained before. This is represented by a more peaked Gaussian centred around $\bar{\omega}_d$, which is reminiscent of the frequency narrowing phenomenon. The faded Gaussian is plotted for comparison. (c) Increasing the shuttled distance while keeping the same speed deteriorates the homogenisation as the electron explores more dots. The shift of the mean frequency manifests the fact that $\bar{\omega}_d \to \omega_0$ as $d \to \infty$. Hamza produced this figure.

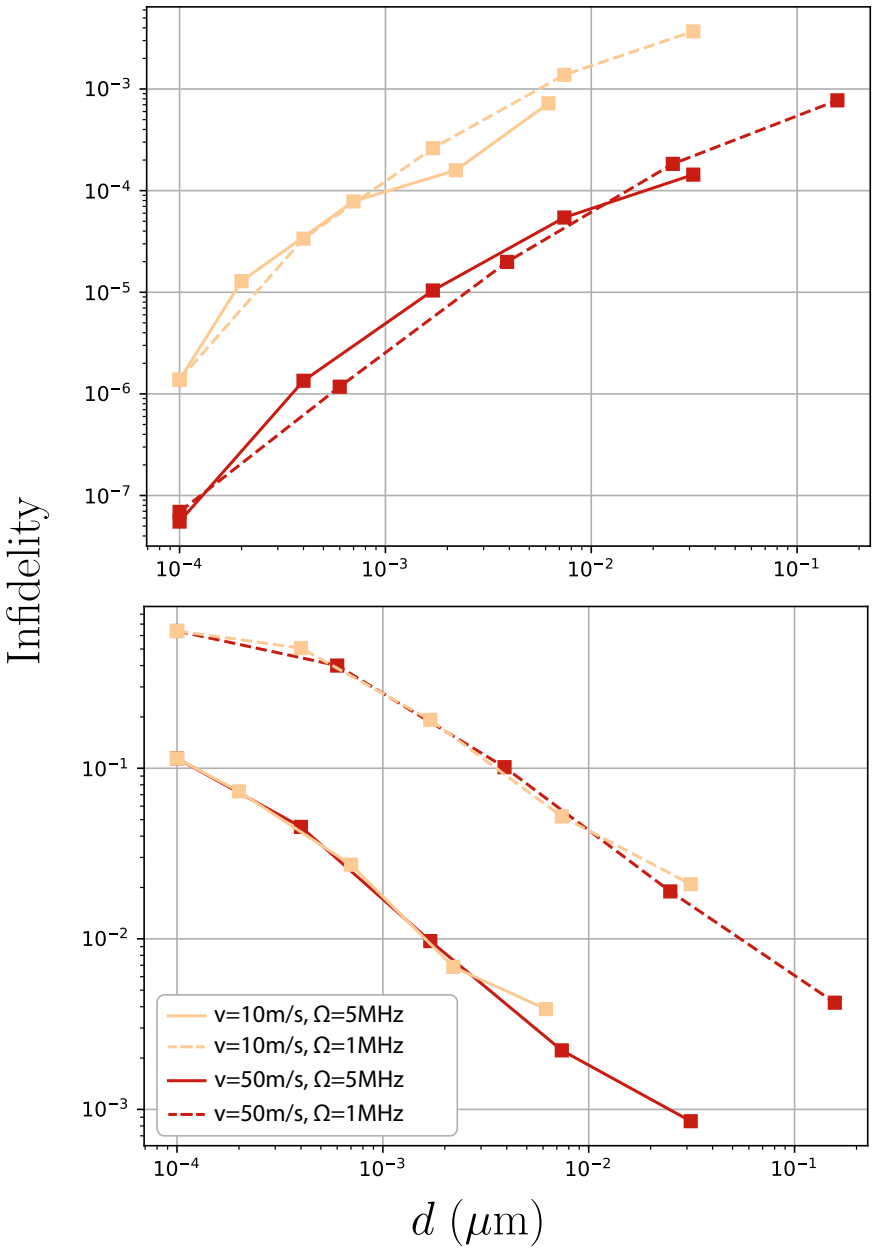


Figure 6.6: Infidelity of the shuttling-based protocol for modes I (top) and II (bottom) as a function of the distance d, the speed v and the drive amplitude Ω. These infidelities are obtained by performing 20,000 Monte-Carlo simulations with random instances of the g-factor landscape. The yellow and red lines correspond to $v = 10$ m/s and $v = 50$ m/s, respectively. The solid lines indicates that $\Omega = 5$ MHz, while $\Omega = 1$ MHz for the dashed lines. This set of data was produced by Hamza using the code I wrote.

proposals suggesting that g-factor disorder might help improve shuttling fidelities rather than hinder them [185]. I will thus assume that these will keep improving.

Additionally, while the results presented in this section are dependent on how the g-factor landscape is defined, the homogenisation principle should work for any model. However, it is worth noting that this scheme only performs well in the case of low frequency dispersion $\Delta\omega_q$ *i.e.* at low magnetic field B_0 and low g-factor dispersion Δg. This is analogous to the observations of Fig. 6.4 for the case of exchange oscillations, in which the infidelity rapidly degraded with the frequency spreading ω_q^{12}. I will confirm this statement for shuttling as well in Section 6.4.

Finally, note that I assumed here (and in the rest of the chapter) that the drive amplitude Ω is homogenous over the shuttling path, which can be quite challenging to achieve but not impossible [171]. I however confirmed with additional numerics that our scheme performs the same in the presence of a non-uniform driving field. Namely, the resonant frequency of a shuttled qubit would still be the average g-factor over the shuttling path and the resulting infidelity would be identical to the case of uniform Ω. The only distinction resides in the gate time, now defined as $T = \pi/\bar{\Omega}$, where $\bar{\Omega}$ is the average Rabi frequency over the shuttling path.

Single-qubit gates on more than one electron

Although we verified here that shuttling could advantageously be used to homogenise g-factors in the simplest case of a single qubit, this method can trivially be extended to higher qubit counts. Unlike the previously presented exchange-based scheme, conveyor-belt shuttling [35, 199] can conveniently be utilised to synchronously shuttle large groups of qubits with no increased resource overhead and without prohibitive fidelity reductions. Indeed, this mode of shuttling uses four input voltages only to create a moving sine wave, whose minima can hold the shuttled qubits and displace them altogether. Clusters of same-frequency qubits can thus be formed by shuttling qubits around loops, such that all qubits in the loop explore the same g-factor landscape.

Extension to two-qubit gates

While the focus of this chapter is on the use of frequency homogenisation to perform global single-qubit gates, the protocol I am presenting can be generalised to two-qubit gates as well. With silicon-spin qubits, these can be implemented by turning on the exchange interaction between neighbouring electrons. Depending on additional parameters *e.g.* the existence of a sufficient energy separation between the electrons or the application of a resonant drive, various kinds of gates can be implemented. These include $\sqrt{\text{SWAP}}$, CZ gates and CROT gates [20] (see Chapter 2). I will here focus on the latter as it requires the use of a drive, therefore making its description a natural extension of the single-qubit case.

More precisely, a CROT gate is a rotation around an axis in the xy plane of the spin of a first electron, conditional on the state of a second electron. To implement it, the exchange interaction J must be turned on but kept small compared to the qubits' energy separation, which I denote as GB_0. In this parameter regime, the degeneracy between the basis states of the two qubits is lifted, enabling the selective targetting of a specific energy transition with a driving pulse. By setting the drive frequency at the energy splitting of the $|10\rangle$-$|11\rangle$ transition, a CROT gate is implemented in a time π/Ω (when $J \ll GB_0$). This frequency is given by [30–33]:

$$\omega = \frac{1}{2}(\omega_{q,1} + \omega_{q,2} - J + \sqrt{J^2 + (\omega_{q,1} - \omega_{q,2})^2}) \tag{6.7}$$

where $\omega_{q,i} = g_i B_0$ is the intrinsic frequency of qubit $i \in \{1, 2\}$.

Like in the single-qubit-gate case, when many qubits are present in the device, it may prove challenging to selectively target pairs of qubits due to frequency crowding. Here I show that this can be mitigated by applying our shuttling-based homogenisation scheme in the same fashion as before. I thus consider two electrons evolving on two parallel shuttling tracks characterised by distinct g-factor profiles $g_1(x_1)$ and $g_2(x_2)$. A large frequency shift GB_0 is applied between the shuttling tracks *i.e.* by using a micromagnet or magnetic materials. Both electrons are synchronously shuttled back and forth on their respective shuttling track over a distance d (see Eq. (6.6)), with a small exchange interaction $J \ll GB_0$ always

turned on between them. The effect of shuttling is to homogenise the respective g-factors of the electrons, leading them to acquire the effective frequencies $\bar{\omega}_{q,1}$ and $\bar{\omega}_{q,2}$. In the light of the previous section, two driving modes are conceivable. Driving mode I consists of driving at the exact means:

$$\omega_I = \frac{1}{2}(\bar{\omega}_{q,1} + \bar{\omega}_{q,2} - J + \sqrt{J^2 + (\bar{\omega}_{q,1} - \bar{\omega}_{q,2})^2}) \tag{6.8}$$

which requires a precise knowledge of the g-factor landscape. In contrast, in driving mode II we drive at the device averages:

$$\omega_{II} = \frac{1}{2}((2g_0 + G)B_0 - J + \sqrt{J^2 + (GB_0)^2}) \tag{6.9}$$

which is agnostic of the details of the g-factor landscapes. Given the experimental challenge that implementing two-qubit gates while shuttling would represent, Hamza and I decided to solely focus on the more experimental friendly driving mode *i.e.* mode II.

The results of such an approach are presented in Fig. 6.7, which I produced. We observe like in Fig. 6.6 (bottom) that the infidelity decreases with the shuttling distance d. Indeed ω_{II} becomes a better and better estimate of the optimal driving frequency ω_I. Contrary to the single-qubit case however, the infidelity saturates at large d instead of keeping decreasing: this is because even the static and exactly-resonant implementation of the two-qubit gate is imperfect (*e.g* due to crosstalk with other energy transitions, see [30, 31] for details). This source of infidelity can be tamed by increasing G – since the condition $J \ll GB_0$ would become better enforced – or by carefully choosing Ω in order to synchronise the resonant and off-resonant rotations [32]. It is worth noting that increasing v reduces the infidelity too, since it improves the quality of the shutting-induced homogenisation. Finally, note that the scheme presented here only works when electrons are shuttled in separate tracks that are well separated in frequency (large G). If G was null or if the electrons were to be placed on the same shuttling track, the resulting operation would be an XX gate.

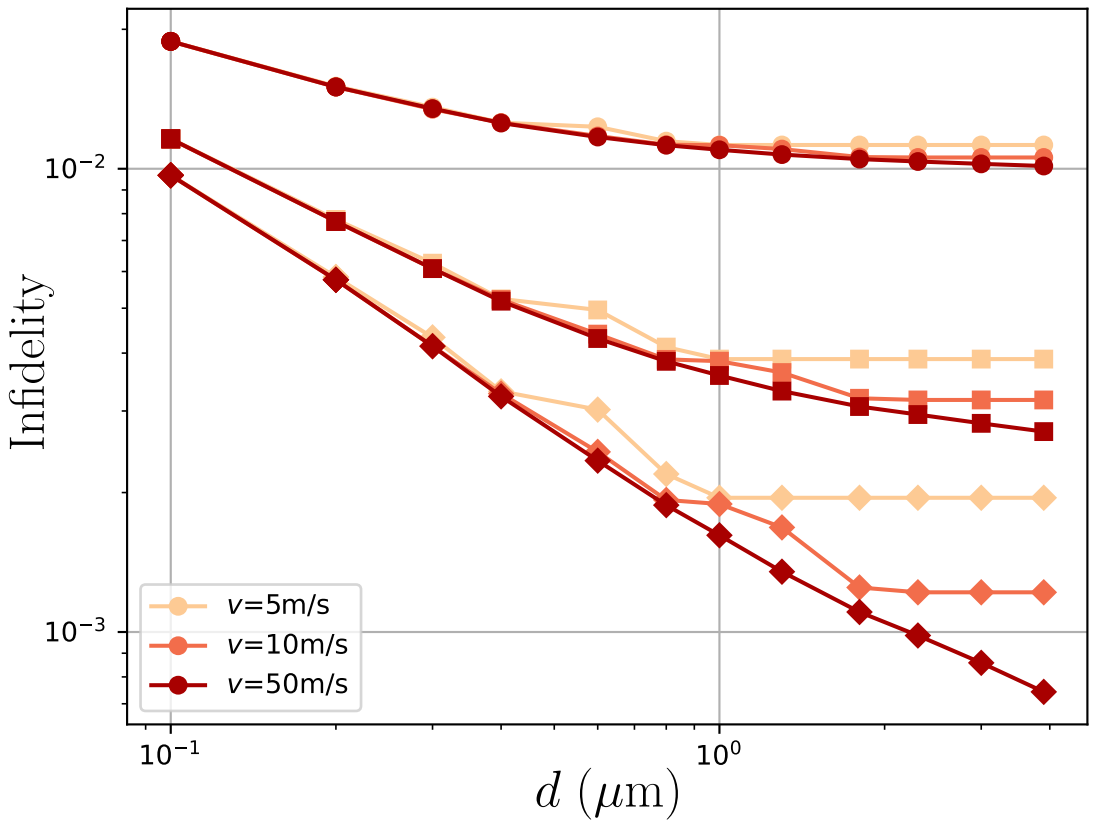


Figure 6.7: Infidelity of a CNOT gate implemented while shuttling the qubits so as to homogenise their respective g-factors. Lighter to darker colours correspond to increasing shuttling speeds. Each triplet of curve (from top to bottom: circles, squares, diamonds) is characterised by a distinct frequency gradient GB_0 between the shuttling rails, respectively 500 MHz, 1 GHz and 2 GHz.

6.4 Applications

In the previous Section, I showed that the shuttling-based protocol was the more scalable. Indeed, it can be applied to multiple electrons without generating entanglement between them, and it guarantees a high single-qubit gate fidelity while only using a limited number of electrodes. In this Section, I thus focus on this protocol and evaluate its performance for the global control of single-qubit gates in a realistic setting.

6.4.1 Problem statement

Given the model presented in Section 6.2, we wish to perform a high-fidelity X gate on n_t targets qubits while keeping the other $n_q - n_t$ qubits idle by using

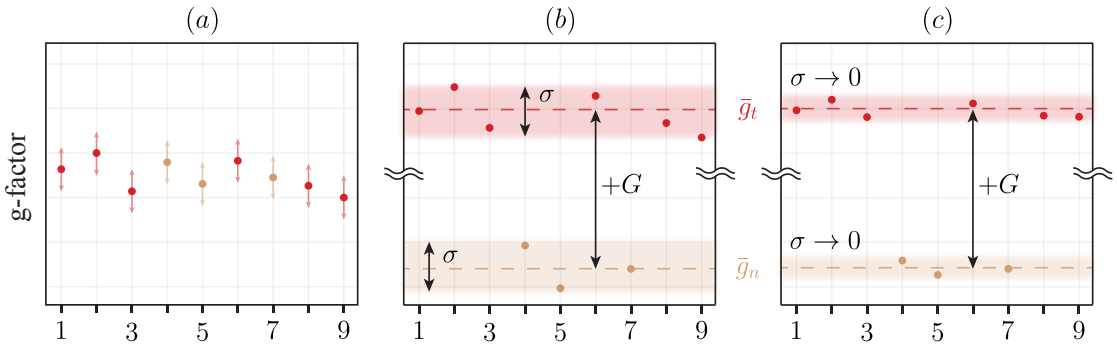


Figure 6.8: Illustration of the global control workflow. We wish to perform an X gate on the target qubits (red) while leaving the non-target qubits (yellow) idle. (a) Given an initial g-factor distribution, we shuttle qubits so as to homogenise their g-factors. The arrows describe the g-factor landscape explored by each electron and the central dots correspond to the resulting mean values. The quality of this process is in particular dependent on the shuttling speed v (see Fig. 6.5). (b) Using micromagnets or superconducting lines, we apply a magnetic field which effectively shifts the g-factors of the targets with respect to the non-targets by G as explained in the main text. The target and non-target qubits homogenised g-factors are now separated and spread around their means $\bar{g}_t$ and $\bar{g}_n$ (dashed lines). The standard deviation of this distribution is given by σ (red and yellow shaded areas). (c) Our shuttling-based homogenisation allows us to decrease σ by increasing the shuttling distance d (see Fig. 6.5(c)). In the limit of large d, every qubit within the red and yellow shaded areas acquire the same g-factor, thereby reducing the spectrum to only one target and one non-target frequencies. Alternatively, σ can be negated by further frequency engineering, namely by applying a distinct G to each qubit. Hamza produced this figure.

a global driving field.

The general workflow Hamza and I adopted is the following (see Fig. 6.8): starting from an initial distribution of g-factors $(g_i)_{1\leq i\leq n_q}$, we apply both our shuttling-based homogenisation protocol and additional electromagnetic fields (*e.g.* Stark shift, global magnetic field gradients) to reduce the frequency spectrum to essentially two g-factors $\bar{g}_t = g_0 + G$ and $\bar{g}_n = g_0$, corresponding to the mean target and non-target g-factors respectively after homogenisation. In some cases, there will remain a distribution of target and non-target g-factors around $\bar{g}_t$ and $\bar{g}_n$, but we ensure that the width σ of such a distribution is small compared to G. σ is controlled by the shuttling distance d: the larger the distance, the smaller σ is. In contrast, G is adjusted via the application of aforementioned electromagnetic fields. Note that the shuttling speed v has no impact on σ (which quantifies the distance between the homogenised g-factors and their means $\bar{g}_t$ and $\bar{g}_n$). v only

influences the quality of each homogenisation (see Fig. 6.8).

Thereon, we send a driving tone at (almost) resonance with the target qubits: $\omega = \bar{g}_t B_0$. Ideally, when G is large and σ is small, the non-targets are left almost idle while each target rotates at a rate $\Omega \approx \bar{g}_t B_1$. In reality, each target rotates around a slightly tilted axis as σ is finite. Besides, the non-targets may experience some crosstalk due to finite G. In order to mitigate it, we additionally tune Ω such that the driving tone induces a π rotation on the targets qubit, but a 2π rotation on the non-targets. This is explained in Appendix C.6.

I will compare this workflow to a state-of-the-art implementation of global single-qubit gates [28, 195], which leverages the same ingredients as above but without our shuttling-based homogenisation. For this reason, it is in general not possible in this case to reduce the spectrum to only two main frequencies like before (σ would be too large). Rather, frequency crowding and frequency separation are dealt with by the sole application of electromagnetic fields, namely by placing the qubits frequencies into (more than two) same-frequency bins. This way the minimum frequency spacing does not decrease with the system size, providing a solution to the frequency crowding issue. The driving pulse has to be composed of multiple driving tones, separated by a frequency distance equal to the bin spacing $2\delta\omega_q$. This relatively low distance can be a source of crosstalk, whose effect is mitigated by tuning the drive strength Ω like before (see Appendix C.6). I will refer to this general method as the *binning technique*.

More concretely, I will focus on two architectures suited for different eras: the near-term 2×N [155] and the long-term looped pipeline architectures [156]. Exploring these two architectures enables to survey the efficacy of our scheme for both the NISQ and fault-tolerant eras. Note however that 2×N arrays of qubits will also be relevant in the fault-tolerant regime as we explored in Chapter 5.

In all the following, I will assume a varying number of target qubits n_t and set the number of non-target qubits to $n_n = n_t - 2$ as an example. The g-factor dispersion Δg is set to $10^{-3} g_0$ or $10^{-2} g_0$, which I will denote as low and high interface roughness respectively. The maximum Stark shift δg is set to $\Delta g/10$. Besides I will

use a drive strength Ω around 5 MHz (taking into account the synchronisation of the target and non-target qubits rotations, as explained in the previous paragraphs and in Appendix C.6). When evaluating the performance of our proposal, I will assume a constant magnetic field $B_0 = 0.1$ T to minimise frequency spreading and facilitate our homogenisation protocol. In contrast, for the binning technique, I will set $B_0 = 1$ T to maximise the frequency shift allowed by Stark shift, which sets the bins separation. As such I am comparing both schemes when they are performing best. In addition, the targets and non-targets will be separated by a frequency shift $GB_0 = 300$ MHz. Assuming that targets and non-targets are on distinct shuttling tracks that are distant by 100 nm (as it will be the case in the following), this translates into a magnetic field gradient of 0.1 mT/nm. This is comfortably below demonstrated gradients induced by micromagnets, around 0.8 mT/nm [173]. As for the shuttling distance and speed, I will consider two distinct parameter regimes that will depend on the architecture: $v = 10$ m/s and $d = 3$ μm (2×N); or $v = 50$ m/s and $d = 20$ μm (looped pipeline). I will comment on the choice of these parameters in Appendix C.7. The performance of each technique is evaluated by running 10,000 Monte Carlo simulations with randomly-generated g-factor profiles. In each instance, I use the same fidelity measure as before: details on our numerics can be found in Appendix C.4. Shuttling noise will still be neglected in the present study in order to focus on qubit addressability.

6.4.2 Performance on the 2×N array

The first architecture Hamza and I considered corresponds to the easiest extension of a 1×N array of quantum dots. Despite its simplicity, I showed in Chapter 5 that one can reliably implement error correction on this architecture. However, here we are concerned with a more restricted version of the architecture in which we require the second row of the 2×N array to be empty as shown in Fig. 6.9.

As explained in the previous section, our first goal is to simplify the frequency spectrum, namely to only retain one main target g-factor $\bar{g}_t$ and one main non-target g-factor $\bar{g}_n$. To do so, we first spatially separate the targets from the non-targets by

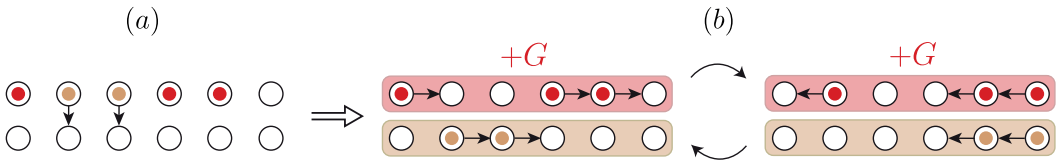


Figure 6.9: Illustration of the shuttling-based protocol for a 2×N array. Starting with all the qubits on the top row in (a), we move the non-targets to the bottom row to physically separate the two types of qubits. Then, thanks to superconducting lines or micromagnets we apply a large magnetic field gradient between the rows (shaded areas) in (b). This magnetic field gradient can also be seen as shifting the g-factors of the electrons on the top row by G. Finally, the electrons are shuttled back and forth in lines to homogenise their g-factors. Hamza produced this figure.

transferring the non-target qubits to the bottom row (Fig. 6.9(a)). In addition to this, we apply a magnetic field gradient between the rows, *e.g.* by placing micromagnet or a superconducting wire parallel to the array. This generates the desired frequency separation GB_0. At this stage however, the target and non-target g-factors are scattered around their mean values (respectively $\bar{g}_n = g_0$ and $\bar{g}_t = g_0 + G$). To lower the standard deviation σ of these dispersions, we shuttle both rows of qubits back and forth (Fig. 6.9(b)). This results in reducing σ by homogenising the g-factor landscape as illustrated in Fig. 6.8. Target qubits can now be driven close to resonance by sending a driving pulse at $\omega = (g_0 + G)B_0$. This is reminiscent of the driving mode II I presented in Section 6.3.2, where qubits are not driven at exact resonance, but rather at a known frequency that is agnostic of the details of the g-factor landscape (ω only depends on the device average g-factor g_0 and the set magnetic field gradient GB_0). As such, the infidelity is arising from two main contributions: the slightly off-resonant driving of the target qubits and the crosstalk between targets and non-targets. Note that in the present scheme, it is not feasible to use the driving mode I (where each target qubit is driven at exact resonance). Indeed, the leftmost and rightmost qubits of the top row do *not* experience the same g-factor landscape, yet their resulting effective g-factors are both too close to $\bar{g}_t$ to differentiate them and drive them with two distinct driving tones.

In Fig. 6.11, which I produced, I compare this protocol to the binning technique, as explained in the previous section. I will focus on the left panel, corresponding

to a shuttling speed $v = 10$ m/s and a distance $d = 3$ μm *i.e.* spanning 30 quantum dots. This scenario is realistic for the near term [37]. I find that in the case of low interface roughness (solid lines) our shuttling protocol (orange) quickly outperforms the binning scheme (yellow). More specifically, it yields a per-qubit infidelity around 0.3% *i.e.* under the surface code threshold, which the binning scheme fails to provide. One however observes an opposite behaviour in the case of high interface roughness (dashed lines), where the binning scheme yields a desirable sub-threshold single-qubit infidelity, far below the one obtained with shuttling. This is because the homogenisation of the g-factors becomes more demanding at large Δg, requiring unfeasible shuttling speeds or drive strengths to compensate for the wider dispersion when shuttling. In contrast, the binning scheme performs well at high interface roughness, owing to a greater interbin spacing arising from a larger Stark shift $\delta g = \Delta g/10$. Note however that contrary to our shuttling scheme, the binning scheme requires a perfect knowledge of the g-factor landscape and a reliable tunability via Stark shift. Besides, one can expect the interface roughness to decrease over time as manufacturing technologies improve.

An additional interesting feature one can note is that the average single-qubit fidelity as defined in Eq. (6.4) decreases when loading more qubits into the device with our shuttling scheme. In contrast, it grows with the binning scheme. This can be explained by noting that in the first case we are not increasing the number of driving tones with the number of qubits, while in the second case, a new driving tone should be added every time a new bin is created. This leads to an increase of the crosstalk between target qubits. At sufficiently large numbers of qubits, both infidelities become stationary however, meaning that both schemes can address the frequency crowding issue, albeit in different parameter regimes.

6.4.3 Performance on the looped pipeline architecture

The second architecture is the one I presented in the last section of Chapter 5, as a way to increase the logical-level connectivity between 2D error-correcting codes, or encode 3D error correction structures within a strictly 2D device [156]. This

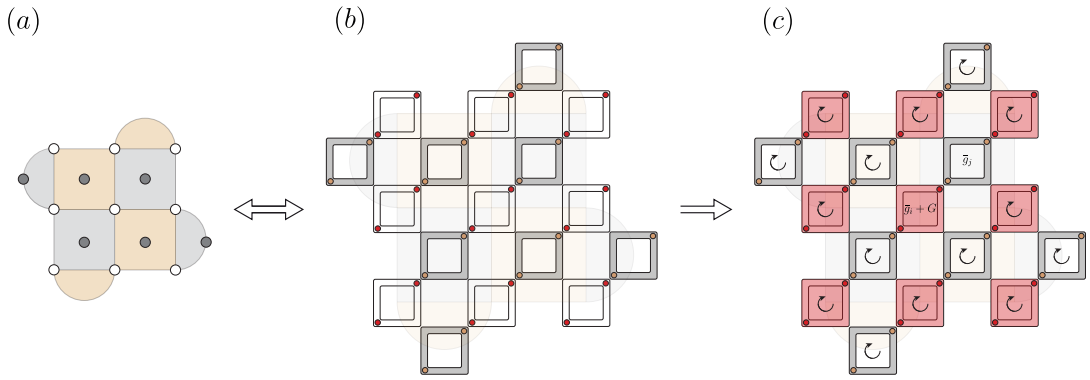


Figure 6.10: (a) Schematic representation of the 3×3 surface code with data and ancilla qubits represented in white and grey respectively. (b) Implementation of the 3×3 surface code within the looped-pipeline architecture with white and grey loops representing data and ancilla loops respectively. Note that n electrons ($n = 2$ here) can populate each loop in order to encode more than one surface code patch as explained in the main text. Red and yellow electrons represent target and non-target qubits respectively. (c) Implementation of single-qubit gates using our shuttling-based homogenisation protocol. During each stabiliser cycle, we wish to perform single-qubit gates on the data loops while leaving the ancillas idle. By shuttling the electrons around loops we homogenise their g-factors and reduce the number of distinct frequencies from $17n$ (with n being the number of electrons per loop) to 17 (number of loops). In order to further reduce crosstalk, we adopt the strategy presented in Fig. 6.8 leveraging superconducting lines/micromagnets to differentially shift the g-factors of the targets (red shaded area) with respect to the non-targets. The g-factor of the data qubits is denoted as $\bar{g}_i + G$ and that of the ancillas as $\bar{g}_j$. Note that, one could tailor the field generated by the micromagnets such that every target obtains the same g-factor G_0, which yields an even higher fidelity as seen in Fig. 6.11. Hamza produced this figure.

amelioration is carried out by shuttling data and ancilla qubits in separate loops that occasionally meet for the implementation of two-qubit gates. By placing multiple electrons in each loop, out-of-plane layers of error correcting codes are effectively stacked together (see Fig. 5.13). Throughout the normal execution of stabiliser cycles, data qubits frequently need to be collectively addressed by a single-qubit Hadamard gate while leaving ancilla qubits idle.

In Fig. 6.10, the looped pipeline architecture is represented in the case of two stacked 3×3 surface codes patches. Data and ancilla qubits (and loops) are represented in white and gray respectively. As qubits go around in loops, they acquire an effective frequency equal to the mean frequency of their respective shuttling track as per the observations of the previous section.

Contrary to the previous architecture I presented, target and non-target qubits are inherently spatially separated, which simplifies their addressability. In order to separate them in the frequency spectrum, we again apply a magnetic field gradient between the target and non-target shuttling tracks. Nonetheless, thanks to the less constrained architecture, we here have more flexibility in the choice of the applied field. Namely, we can distinguish two realistic scenarios. In the simplest one, the same frequency shift GB_0 is applied to all target loops (this is equivalent to the 2×N scheme): this can be implemented by placing regularly-spaced superconducting wires above the target loops. Note however that assuming a perfect loop-wise g-factor homogenisation, the number of frequencies remaining in the spectrum is given by the number of loops rather than the total number of qubits. Like before, the infidelity is here limited by the slightly off-resonant driving of the targets and the crosstalk between targets and non-targets.

In the second more intricate scenario, we apply a distinct field $(G - \bar{g}_i)B_0$ to the loop i, where $\bar{g}_i$ is its mean g-factor. This effectively negates the g-factor dispersion σ between the targets as well as between the non-targets, *i.e.* leaves the frequency spectrum with two frequencies only. This however requires a knowledge of all mean g-factors $\bar{g}_i$ and the capability to apply a loop-specific magnetic field gradient. One could envision this by inserting out-of-plane magnetic materials inside each target loop. Note that a constant monitoring of the g-factor landscape is required in this case, and it must be fed back so as to update the strength of their induced magnetic field accordingly. Therefore, the latter scheme represents a higher experimental challenge, but it is expected to yield significant infidelity reductions, owing to the total suppression of the dispersion σ of the homogenised g-factors around $\bar{g}_t$ and $\bar{g}_n$ (see Fig. 6.8). This means that all target qubits can be driven at exact resonance, and that the crosstalk with non-target qubits can be annihilated by tuning Ω such that they perform a 2π rotation during the gate time (see Appendix C.6). The only remaining source of infidelity in this case is thus the imperfect homogenisation of the g-factors.

The performance of these two variants of our scheme are presented in Fig. 6.11, along with the binning scheme. I will now focus on the right panel of the figure, corresponding to a shuttling distance $d = 20\mu$m and a shuttling speed $v = 50$ m/s. Such parameters are anticipated to be the working point of the looped pipeline architecture [156]. While the binning scheme is unaffected by the choice of d and v, we observe a general reduction of the infidelity of our shuttling scheme compared to the left panel. The main observations remain the same however: shuttling proves advantageous in the low-interface-roughness regime, while binning yields promising infidelities for high interface roughness. The main difference compared to the left panel is that the $\sigma = 0$ data (brown lines) outperforms the $\sigma > 0$ data (orange lines) by almost an order of magnitude, potentially justifying the use of this more experimentally-demanding strategy. The reason why it did not prove advantageous before is that the homogenisation error was just saturating the infidelity at low v. Moreover, as we set $n_n = n_t - 2$ the rightmost data points correspond to two stacked 3×3 rotated surface codes (two qubits per loop), for which we observe comfortable sub-threshold infidelities for the shuttling or binning schemes, respectively at low or high interface roughness. Since at this point all infidelities are stationary with the number of qubits, I expect this observation to hold for any code size and number of stacked surface codes.

6.5 Discussion and outlook

In this work, I considered the pressing issue of individual spin qubits control. When qubits are encoded with a single spin trapped in a quantum dot, single-qubit gates are implemented via the use of magnetic fields, which are hard to localise at the qubit scale. Rather, existing research advocated for the use of a global field with multiple driving tones at resonance with the target qubits frequencies, which are variable in essence, due to interface roughness. This generally induces frequency crowding thus unwanted crosstalk that needs to be addressed. In this piece of research, Hamza and I designed an elegant framework to globally control silicon spin qubits with high fidelity.

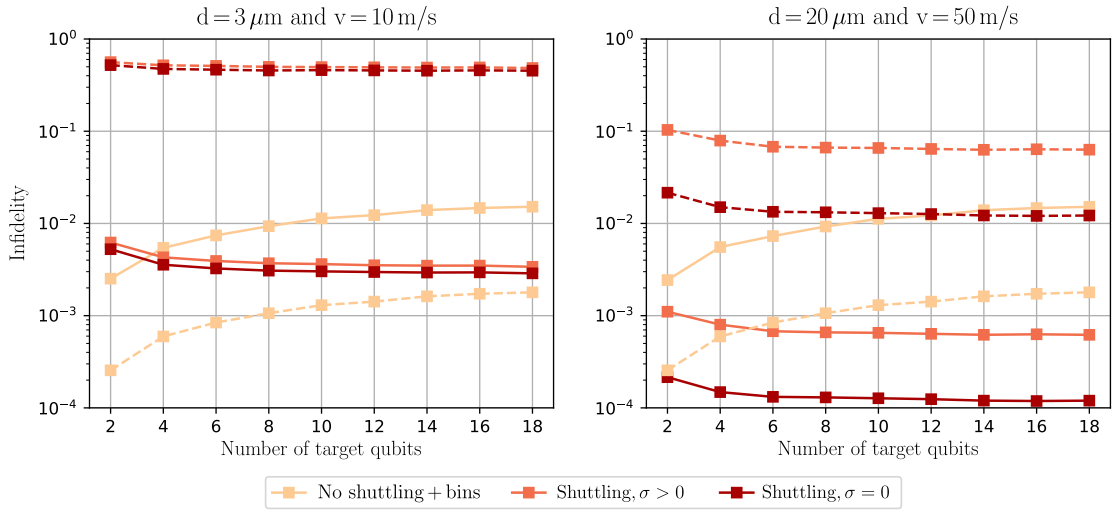


Figure 6.11: Single-qubit gate infidelity for the 2×N and looped pipeline architectures in the presence of n_t target qubits and $n_n = n_t - 2$ non-target qubits. The solid and dashed lines respectively correspond to a g-factor dispersion $\Delta g = 10^{-3} g_0$ and $10^{-2} g_0$. The targets and non-targets are separated by a magnetic field gradient GB_0. I consider three different strategies for the gate implementation: binning the frequencies without shuttling the qubits (yellow); shuttling the target and non-target qubits so as to reduce the spreading σ of their frequencies (orange); and the same as the latter with additional frequency engineering in order to negate any frequency spreading among the target qubits *i.e.* setting $\sigma = 0$ (brown). In the first case, the drive contains multiple tones at resonance with each bin. In the second and third cases however, we send a single tone either approximately or exactly at resonance with the target qubits (owing to σ being finite or not).

More specifically, we demonstrated that electron motion can be utilised to homogenise the g-factors it experienced. When the motion takes the form of exchange oscillations between a pair of qubits, both electrons g-factors become equal to the mean of the original g-factors. When the motion is implemented via spin qubit shuttling, all electrons travelling around a looped shuttling track obtain the same new effective g-factor equal to the loop mean g-factor. These examples make it clear that the above homogenisation method can be utilised to reduce the number of frequencies in the frequency spectrum. In this research, we put this observation to use for the implementation of single-qubit gates. While frequency narrowing had already been identified in the past [36], we here explored a new paradigm where it becomes an essential ingredient to the implementation of quantum gates.

The main result of this chapter is that when an electron is transferred between

multiple locations, a single-qubit gate can simply be implemented by sending a driving tone at resonance with the mean frequency felt by the electron. The quality of the gate is dependent on the transfer rate (exchange coupling, shuttling speed): highest fidelities are achieved when the transfer rate is large compared to the Rabi frequency and the qubits frequency dispersion. Via numerical simulations produced from my code, Hamza and I quantified the amount of error generated by such a protocol and obtain promising estimates in the case of transferring a single electron via shuttling. In particular, we demonstrated that when the shuttling distance was large enough, the qubit g-factor dispersion rapidly decreased, meaning that the sole knowledge of the device average g-factor g_0 sufficed to set the driving frequency. This alleviates the experimental requirements for a spin-qubit-based quantum computer, by removing the need for a constant monitoring and calibration of the detailed g-factor landscape. These promising observations led us to extend our analysis first to two-qubit gates, for which I still showed the suitability of our protocol. Then we aimed to tackle realistic experimental setups: a 2×N array of qubits and the looped pipeline architecture [156]. Even in these more complex contexts, I showed orders of magnitude improvements of the fidelity of single-qubit gates on selected subsets of qubits compared to previous state-of-the-art techniques. The use of shuttled two-qubits gates was not investigated but would certainly fit in this context as well, promising easier syndrome extraction for fault-tolerant devices.

As future work, it would be valuable to generalise our protocol to more general types of qubits and materials. Here, we focused on electron spins in silicon, where the g-tensor anisotropy is small. This means that we were able to only consider scalar g-factors. More generally, we believe that effects of stronger spin-orbit coupling can be calibrated away as long as they are known *a priori*.

Additionally, we envision extending our study to more kinds of architectures. One example is my novel Quantum Snakes on a Plane [165], where logical qubits consist of linearised objects than can be shuttled around large loops made of 2×N filaments. I believe that our protocol could greatly facilitate the implementation of single- and two-qubit cases in this paradigm.

Another direction we wish to explore is the inclusion of various physical and experimental limitations. First, this encompasses off-resonant driving, *e.g.* stemming from the use of finite-bandwidth driving tones. Second, non-adiabatic effects could arise from the abrupt switching-on and off of the driving pulse and exchange coupling. One could further evaluate if these effects can be tamed via known pulse shaping techniques. Finally, noise would inevitably be present in any experimental chip, *e.g.* charge noise or valley excitations (when shuttling). It would be valuable to run such simulations, which we overlooked for now in order to solely tackle the qubit addressability problem. These would further inform the user about the trade-offs between the use of our schemes and increased noise when qubits are swapped or shuttled.

7

Algorithmic error mitigation for quantum eigenvalues estimation

The research presented in this chapter originates from a collaboration with Kosuke Mitarai and Keisuke Fujii and is available online [217]. The problem at stake was imagined by Profs. Mitarai and Fujii. All ideas and research were however produced by myself, with the useful input and comments of my co-authors during our regular meetings.

All writing below is my own, in places using text verbatim from my manuscript.

Contents

7.1 Introduction

7.1.1 Quantum error mitigation for early fault-tolerance

In the past chapters, I delved into one of the most topical issues of quantum computing today: quantum error correction. Without it, quantum computers will not reach their full potential given the forbiddingly low error rate that are required to run deep quantum algorithms. Even so, while the first demonstrations of error correction are being experimentally implemented [9, 93, 218], it will be long before such devices can run the aforementioned algorithms: limited number of qubits or high physical noise are today among the experimental limitations that hinder the resulting logical error rates and number of implementable logical gates. Before we reach a regime of long-term stable fault-tolerance, there will likely be an extended period of time of *early fault-tolerance* where QEC is functional but does not provide low enough error rates or is limited in the number of logical gates that can be implemented. In this regard, I presented in Chapter 5 a solution to simplify the circuitry used to build error correction codes; this manifestly came at the cost of longer quantum computations, reaching several days for applications slightly beyond the classically tractable regime. Additionally, the increased physical error rates arising from the shuttling protocol implied higher code distances in order to reach low enough logical error rates, meaning increased qubit overhead.

In this specific case, I thus chose the simplest approach to counterbalance the higher physical error rates that are characteristic of the early fault-tolerant regime: scale up the code. While this is not believed to pose a significant challenge for silicon spin qubits [19], it may not be a viable option for all qubit platforms. This motivates the use of alternative techniques to further reduce errors, such as quantum error mitigation (QEM).

Originally, QEM was designed for Near Intermediate Scale Quantum (NISQ) devices, a generation of quantum computers that would precede fault-tolerance.

In this regime, the limitations stemming from noise and qubit and gate counts are too stringent to implement any error correcting code, and only low-depth (*e.g.* variational) algorithms can be run. Errors that accumulate from the implementation of a quantum circuit can then be *mitigated* rather than *corrected*. While QEC relies on a considerable qubit and gate overhead, QEM rather promises a reduction of errors by increasing the sampling overhead and classically post-processing the data. The drawback is that, for any QEM technique, the sampling cost exponentially increases with the qubit and gate counts [219], meaning that QEM alone is not a scalable technique and is not suitable to the deepest circuits. In concordance with QEC however, it could help alleviate the resource requirements associated with error correcting codes and extend the range of applications of early fault-tolerant devices. This perspective was for instance studied in details in [220], where it was shown that the use of QEM could effectively enhance error correction codes distances, thereby replacing qubit overhead with sampling overhead for equal error reduction performance.

7.1.2 Algorithmic error mitigation for quantum eigenvalues estimation

In this chapter, I further explore this paradigm and present an error mitigation technique targeted at the improved evaluation of quantum eigenvalues. The eigenvalues of observables characterising a quantum system serve as a critical benchmark for understanding the system's behaviour and properties. Accurate estimation of the eigenvalues, such as the ground-state energy of the Hamiltonian, enables researchers to unravel key phenomena like chemical reactions, material properties, and phase transitions. While the task of estimating the eigenvalues becomes exceedingly challenging for classical computers as the size of the quantum system increases, quantum computers offer a promising avenue for tackling this problem efficiently, in particular with the use of the well-known phase estimation algorithm (PE) [221–224]. No matter what variant of the algorithm is used, PE always takes as input a unitary operator $\mathcal{W}(A)$ whose eigenvalues are a function of

the eigenvalues of the target observable A. In the case of the Hamiltonian, most past works have focused on Hamiltonian Simulation, i.e. $\mathcal{W}(H) = \mathrm{e}^{-\mathrm{i}Ht}$ for a given time t, which can be implemented by Trotterisation [225, 226] or truncated Taylor series [227]. More recent papers [228, 229] however advocated for the implementation of $\mathcal{W}(H) = \mathrm{e}^{-\mathrm{i}\arccos(H/\lambda)}$ with λ a parameter greater than the spectral norm of H. This unitary can besides be implemented efficiently by qubitisation [230].

Nonetheless, as previously highlighted, early fault-tolerant computers may be too noisy to implement such deep algorithms: first due to residual gate errors stemming from insufficient error correction; second owing to compilation errors or *algorithmic errors* arising from approximations the algorithm uses to implement the unitary operator $\mathcal{W}(A)$. For instance, in the case of Trotterisation, the exponential of a sum is broken down into a product of exponentials, which is only exactly true when all terms of the sum commute; as for qubitisation, the protocol requires the preparation of a state $|G\rangle$ encoding information about the target Hamiltonian, which can only be executed up to a certain error [186]. These algorithmic errors can of course be tamed by augmenting the circuit complexity within the quantum algorithm, thereby improving its precision, but this may be unfeasible in early fault-tolerant quantum computers where the number of qubits and gates will have to remain moderate so as to avoid gate error accumulation (in particular non-Clifford gates [231]). The solution I instead advocate for in this chapter is thus to use error mitigation.

The specific protocol I present here targets algorithmic errors, and can be applied not only for the usual estimation of expectation values, but also for eigenvalues. To this end, my method expands on Richardson extrapolation [232], with key novel elements: if this extrapolation has already been identified as a promising candidate for error mitigation in the context of Hamiltonian Simulation [233, 234], or used in conjunction with error correction [220, 235, 236], my contribution lies in generalising it. More precisely, I first extend it to the estimation of eigenvalues, which requires the use of precise and non-trivial mathematical theorems. Second, I enable its application to the most optimal Hamiltonian Simulation algorithm, namely qubitisation, by employing a multi-parameter formalism introduced later in

the chapter. By implementing multiple imperfect but fully determined unitaries and extracting their eigenvalues, I show that one can build an estimate of the target eigenvalue up to any order p. To assess the effectiveness of this method, I conduct numerical tests and observe a significant reduction in errors, reaching several orders of magnitude, such as in the case of Trotter errors. The overall cost of the procedure is also evaluated and is shown to remain particularly modest as one only has to independently perform phase estimation a sufficient number of times. To correct up to p-th order, I estimate this number to grow as a polynomial of degree p with the number of terms of the Hamiltonian, or even linearly with p in some relevant cases such as Trotterisation.

7.2 Methods

In this section, I present my error mitigation scheme. After stating the assumptions it relies on, I describe the error mitigation protocol and theoretically estimate its performance.

7.2.1 Algorithmic errors of observables

We are interested in estimating the eigenvalues of a target observable A on a fault-tolerant computer. This can be done with the use of the phase estimation algorithm (PE), assuming that a unitary operator $\mathcal{W}(A)$ can be implemented. However, owing to limited circuit depth for the generation of such unitary operator, the implementation may only be performed up to some algorithmic (or compilation) errors. The consequence is the implementation of an actual unitary $\mathcal{W}(A')$, associated with an effectively implemented observable A', whose distance to the target observable A depends on the strength of the algorithmic errors.

One particularly important thing to note is that, because we are focusing on a fault-tolerant setup, A and A' are close to deterministic. Indeed, we can consider in a first approximation that the logical error rate is null in a FTQC and that only algorithmic errors exist (finite logical error rates will be studied later on). Consequently, one exactly knows the target observable A as well as the circuit that

generated the approximate observable A' (chosen by the user). A and A' can then be parameterised by a functional $\tilde{A}(\delta_1, ..., \delta_N)$ that satisfies:

$$A = \tilde{A}(0, ..., 0) \tag{7.1}$$

$$A' = \tilde{A}(\delta_1, ..., \delta_N) \tag{7.2}$$

for a specific choice of δ_i's which represent algorithmic errors. Importantly, these δ_i's are entirely known.

This can be illustrated with two examples in the case of the eigenvalue estimation of a Hamiltonian. I will note the target and implementable Hamiltonians H and H' respectively (instead of A and A'). When $H = \sum H_k$ and $\mathcal{W}(H) = \mathrm{e}^{-\mathrm{i}Ht}$ is implemented by Trotterisation, the algorithmic errors stem from the inexact splitting of the exponential of a sum into a product of exponentials. The effective Hamiltonian under first-order Trotterisation is:

$$H' = \frac{\mathrm{i}}{\delta t} \log\left(\prod \mathrm{e}^{-\mathrm{i}H_k \delta t}\right) \tag{7.3}$$

with δt a small fraction of the total time t. As a result, the functional describing H and H' can be chosen to depend on a single parameter: $N = 1$ and $\delta_1 = \delta t$.

Likewise, if H is expressed as a linear combination of unitaries $H = \sum c_i P_i$, and $\mathcal{W}(H) = \mathrm{e}^{-\mathrm{i}\arccos(H/\lambda)}$ is implemented by qubitisation [186, 228], then the protocol relies on the preparation of the state $|G\rangle = \sum \sqrt{c_i/\lambda}\,|i\rangle$ with $\lambda = \sum c_i$. More precisely, if the state $|G\rangle$ can be prepared, qubitisation enables an exact implementation of $\mathcal{W}(H)$ (assuming no gate errors). However, the state preparation itself is subject to algorithmic errors [186] and a state $|G'\rangle = \sum \sqrt{c'_i/\lambda'}\,|i\rangle$ with $\lambda' = \sum c'_i$, may instead be generated. As a result, the effectively implemented observable is:

$$H' = \sum_i c'_i P_i \tag{7.4}$$

The functional describing H and H' hence depends on multiple parameters $\delta_i = c'_i - c_i$ which, again, are known. For instance, in [186], c'_i is the μ-bit binary approximation of c_i, with μ related to the number of ancilla qubits used in the state preparation circuit.

7.2.2 Problem statement

The problem I consider here is the following: given a target observable A and an implementable observable A' that are both entirely known, how can one accurately estimate the eigenvalues of the target observable A from the eigenvalues of the implementable observable A'? I show in the next sections that this task can be fulfilled efficiently provided not one, but multiple and distinct, observables A'_k approximating A are implemented:

$$A'_k = \tilde{A}(\delta_{1,k}, ..., \delta_{N,k})$$

From now on, I will denote by m the number of implemented A'_k's. N will still designate the number of coefficients an individual A'_k depends on. In the above equation, the second subscript $k \in [\![1, m]\!]$ of $\delta_{i,k}$ will refer to the implemented observable A'_k while the first one $i \in [\![1, N]\!]$ will refer to the specific coefficient. Alternatively, $(\delta_{1,k}, ..., \delta_{N,k})$ may be noted in a more compact form as an N-dimensional vector $\vec{\delta}_k$.

There are various ways of generating these A'_k depending on the situation. Considering the two previous examples again: for Trotterisation, it is enough to vary the infinitesimal time step δt; for qubitisation, in particular within the state preparation protocol of [186], one can generate the closest μ-bit binary numbers $c'_{i,k}$ to each of the target coefficients c_i (rather than just one c'_i per c_i).

7.2.3 Original idea

In this section I present the pathway that led me to the main results of this chapter. Initially, I decided to solely focus on the ground state evaluation of a Hamiltonian $H = \sum c_i P_i$, approximated by a simulable Hamiltonian $H' = c'_i P_i$ (*e.g.* by qubitisation). Given the proximity of H and H', my initial idea was to use perturbation theory to approximate the unknown eigenvalues of H with the computable eigenvalues of H'. Noting $V = H - H' = \sum \delta_i P_i$, first-order perturbation theory gives:

$$E = E' + \sum_i \delta_i \left\langle \psi'^{(0)} \right| P_i \left| \psi'^{(0)} \right\rangle + O(\vec{\delta}^2) \tag{7.5}$$

Here the exponents in brackets refer to the index of an eigenstate *e.g.* $\left|\psi'^{(0)}\right\rangle$ for the ground state of H'. I simply note $E' = E'^{(0)}$ and $E = E^{(0)}$. This equation expresses E (that I am attempting to evaluate) as a function of quantities that can be estimated: E' and $\left|\psi'^{(0)}\right\rangle$ are the outputs of phase estimation applied to H' (which is implementable) and all δ_i's are known by assumption. All expectation values in the sum can thus be evaluated by sampling, eventually giving a first-order estimate of E.

While this is promising, sampling these N expectation values induces a sampling overhead one could actually avoid. To do so, let us suppose that we can prepare multiple H'_k in the vicinity of H (as explained in the previous section) and let us expand them as a function of H:

$$E'_k = E + \left\langle\psi^{(0)}\right| - \hat{V}_k \left|\psi^{(0)}\right\rangle + O(\vec{\delta}^2) \tag{7.6}$$

Now expressing E:

$$E = E'_k + \left\langle\psi^{(0)}\right| \hat{V}_k \left|\psi^{(0)}\right\rangle + O(\vec{\delta}^2) \tag{7.7}$$

$$= E'_k + \sum_i \delta_{i,k} \left\langle\psi^{(0)}\right| P_i \left|\psi^{(0)}\right\rangle + O(\vec{\delta}^2) \tag{7.8}$$

Importantly, instead of directly expressing E as a function of E'_k (as in Eq. 7.5), I rather did the opposite first then reversed the equation. This way, $\delta_{i,k}$ holds the only k-dependence (otherwise $\left|\psi^{(0)}\right\rangle$ would too). As $\vec{\delta}_k$ are vectors of an N-dimensional space, there exists $(\lambda_1, ..., \lambda_{N+1})$ such that

$$\sum_k \lambda_k \vec{\delta}_k = 0 \iff \forall i \quad \sum_k \lambda_k \delta_{i,k} = 0 \tag{7.9}$$

By combining the above equations:

$$\sum_k \lambda_k E = \sum_k \lambda_k E'_k + \sum_i \left(\sum_k \lambda_k \delta_{i,k}\right) \left\langle\psi^{(0)}\right| P_i \left|\psi^{(0)}\right\rangle + O(\vec{\delta}^2) \tag{7.10}$$

Then by dividing by $\sum_k \lambda_k$, one obtains:

$$E = \frac{\sum_k \lambda_k E'_k}{\sum_k \lambda_k} + \frac{\sum_i \left(\sum_k \lambda_k \delta_{i,k}\right) \left\langle\psi^{(0)}\right| P_i \left|\psi^{(0)}\right\rangle}{\sum_k \lambda_k} + O(\vec{\delta}^2) \tag{7.11}$$

$$= \frac{\sum \lambda_k E'_k}{\sum_k \lambda_k} + O(\vec{\delta}^2) \tag{7.12}$$

This result holds much more power than Eq. 7.5: with only $N+1$ uses of phase estimation on different H'_k's, one can exactly estimate a first-order approximation of E, without the need for any expectation values estimations. This requires the calculation of the coefficients (λ_k), which can easily be computed from the known $(\delta_{i,k})$ by solving a linear system. On the contrary, Eq. 7.5 required such expectation values estimations, leading to sampling errors ε and a total sampling cost N/ε.

Besides, this second method can easily be extended to higher orders:

$$E = E'_k + \left\langle \psi^{(0)} \right| \hat{V}_k \left| \psi^{(0)} \right\rangle - \sum_{l \neq 0} \frac{\left| \left\langle \psi^{(l)} \right| \hat{V}_k \left| \psi^{(0)} \right\rangle^2 \right|}{E^{(0)} - E^{(l)}} \tag{7.13}$$

$$= E'_k + \sum_i \delta_{i,k} \alpha_i + \sum_{i,j} \delta_{i,k} \delta_{j,k} \beta_{i,j} \tag{7.14}$$

where

$$\alpha_i = \left\langle \psi^{(0)} \right| P_i \left| \psi^{(0)} \right\rangle \quad \text{and} \quad \beta_{i,j} = \sum_{l \neq 0} \frac{\left\langle \psi^{(l)} \right| P_i \left| \psi^{(0)} \right\rangle \left\langle \psi^{(0)} \right| P_j \left| \psi^{(l)} \right\rangle}{E^{(0)} - E^{(l)}} \tag{7.15}$$

are both k-independent. Now by choosing $\vec{\lambda}$ of size at most $N^2 + N + 1$ such that

$$\forall i,j \quad \sum_k \lambda_k \delta_{i,k} = 0, \quad \sum_k \lambda_k \delta_{i,k} \delta_{j,k} = 0 \quad \text{and} \quad \sum_k \lambda_k = 1 \tag{7.16}$$

one gets:

$$E = \sum_k \lambda_k E'_k + \sum_i \left(\sum_k \lambda_k \delta_{i,k} \right) \alpha_i + \sum_{i,j} \left(\sum_k \lambda_k \delta_{i,k} \delta_{j,k} \right) \beta_{i,j} \tag{7.17}$$

$$= \sum_k \lambda_k E'_k + O(\vec{\delta}^3) \tag{7.18}$$

which gives an exact second-order estimate of E.

While writing these equations, I realised that the exact expression of the coefficients α_i and $\beta_{i,j}$ was unimportant as they were to be cancelled out anyway. Further, Eq. 7.14 can directly be written by Taylor-expanding E, rather than using step 7.13 and perturbation theory at all. Following this more agnostic procedure, the scheme could even be extended to any observable. This generalisation is presented in the next section.

7.2.4 Main theorem

My main result reads as follows:

Theorem 1. *Let A be an observable whose eigenvalues one wants to estimate up to a given order p, and (A'_k) be a set of observables, in the neighbourhood of A, whose eigenvalues one can estimate. Assume that the observables in the vicinity of A can be described by a smooth functional $\tilde{A}(\delta_1, ..., \delta_N)$ such that:*

$$A = \tilde{A}(0, ..., 0), \tag{7.19}$$

$$\forall k \quad A'_k = \tilde{A}(\delta_{1,k}, ..., \delta_{N,k}), \tag{7.20}$$

Assume that, for all i and k, the coefficients $\delta_{i,k}$ parameterising A'_k are known. It follows that one can construct coefficients $(\lambda_1, ..., \lambda_m) \in \mathbb{R}^m$ such that any non-degenerate eigenvalue a of A can be estimated up to order p from eigenvalues a'_k of A'_k and from the coefficients λ_k:

$$\exists m \in \mathbb{N} \quad \exists (\lambda_1, ...\lambda_m) \in \mathbb{R}^m \quad a = \sum_{k=1}^{m} \lambda_k a'_k + \delta a \tag{7.21}$$

where

$$\delta a = O\left(||\lambda||_1 ||\delta||_\infty^{p+1}\right) \tag{7.22}$$

where $||.||_1$ denotes the ℓ^1-norm and $||\delta||_\infty = \max_{i,k} |\delta_{i,k}|$. The required number of coefficients (or equivalently of observables A'_k) satisfies:

$$m = O(N^p) \tag{7.23}$$

Remark 1.1. *If the functional $\tilde{A}$ depends on one parameter only ($N = 1$), the eigenvalue a one is trying to estimate need not be non-degenerate.*

Proof. Let a be a non-degenerate eigenvalue of A and $p \in \mathbb{N}$ be the order up to which we want to estimate a. Since a is non-degenerate, it is a locally totally differentiable function of the operator A [237]. Alternatively, as per Remark 1.1, if $\tilde{A}$ depends on one parameter only, the eigenvalue a need not be non-degenerate to be a totally differentiable function of this parameter [237]. In both situations

and in less technical terms, this means that a varies smoothly when A is perturbed. Thus, if A' is an observable in the neighbourhood of A such that:

$$A' = \tilde{A}(\delta_1, ..., \delta_N)$$

then there exists an eigenvalue a' of A' in the neighbourhood of a, which can be expressed via a generalised Taylor expansion (using the notation $|i| = i_1 + ... + i_N$):

$$a' = a + \sum_{1 \leq |i| \leq p} \delta_1^{i_1} ... \delta_N^{i_N} \gamma_{i_1,...,i_N} + O(||\delta||_\infty^{p+1}),$$

with

$$\gamma_{i_1,...,i_N} = \frac{1}{i_1!...i_N!} \frac{\partial^{|i|} a}{\partial \delta_1^{i_1} ... \partial \delta_N^{i_N}} \Big|_{\delta_1 = ... = \delta_N = 0}. \tag{7.24}$$

For $m \in \mathbb{N}$ different observables A'_k, we have

$$a'_k = a + \sum_{1 \leq |i| \leq p} \delta_{1,k}^{i_1} ... \delta_{N,k}^{i_N} \gamma_{i_1,...,i_N} + O(||\delta||_\infty^{p+1}) \tag{7.25}$$

where

$$||\delta||_\infty = \max_{i \in [\![0, N-1]\!],\ k \in [\![0, m-1]\!]} |\delta_{i,k}|$$

Importantly, $\gamma_{i_1,...,i_N}$ does not depend on k. Now, consider, for $k \in [\![1, m]\!]$, the vector $\vec{x}_k$ consisting of the monomials appearing in the expansion:

$$\vec{x}_k = (\delta_{1,k}^{i_1} ... \delta_{N,k}^{i_N})_{0 \leq |i| \leq p} \tag{7.26}$$

Let us denote X the matrix of the $\vec{x}_k$'s (in columns) and $b = (1, 0, ..., 0)^T$ (of same size as $\vec{x}_k$). When m is greater than the size of $\vec{x}_k$, that is $m = O(N^p)$, one can find a solution $\vec{\lambda} = (\lambda_1, ..., \lambda_m)^T$ to the linear system

$$X\vec{\lambda} = b \tag{7.27}$$

The above system is equivalent to:

$$\sum_{k=1}^{m} \lambda_k \delta_{1,k}^{i_1} ... \delta_{N,k}^{i_N} = 0 \tag{7.28}$$

for all $i_1, ..., i_N$ such that $1 \leq |i| \leq p$, and:

$$\sum_k \lambda_k = 1 \tag{7.29}$$

when $i_1 = ... = i_N = 0$. As a result, combining Eqs. 7.25, 7.28 and 7.29, one gets:

$$\begin{aligned}\sum_k \lambda_k a'_k &= a + \sum_{1\leq|i|\leq p} \left(\sum_k \lambda_k \delta_{1,k}^{i_1} ... \delta_{N,k}^{i_N}\right) \gamma_{i_1,...,i_N} \\ &\qquad + \sum_k \lambda_k O\left(||\delta||_\infty^{p+1}\right) \\ &= a + O\left(||\lambda||_1 ||\delta||_\infty^{p+1}\right)\end{aligned}$$

□

This first theorem and its proof provide a deterministic and exact way to construct a p-th order estimate of any non-degenerate eigenvalue a of the target observable (or any eigenvalue if the A'_k's can be parameterised with a single variable).

Let us look at the simple example of a Hamiltonian implemented by Trotterisation $H'(\delta t)$ (Eq. 7.3). Assume that the minimal time step one can implement is $\delta t_{\min}$ and that one aims to correct up to first order. By implementing $H'(\delta t_{\min})$ and $H'(2\delta t_{\min})$, one can construct an estimate of any eigenvalue E of H as:

$$E \approx \lambda_1 E'(\delta t_{\min}) + \lambda_2 E'(2\delta t_{\min})$$

with (λ_1, λ_2) solution of the system:

$$\begin{cases} \lambda_1 \delta t_{\min} + \lambda_2 2\delta t_{\min} = 0 \\ \lambda_1 + \lambda_2 = 1 \end{cases}$$

giving $\lambda_1 = 2$ and $\lambda_2 = -1$. This estimate is accurate up to $O(\delta t_{\min}^2)$.

Importantly, the theorem's proof shows that the estimate we construct does not require any knowledge or evaluation of the complicated coefficients $\gamma_{i_1,...,i_p}$ of Eq. 7.24. It also does not matter if one has restrictions on the set of A'_k's they can implement (as long as it is large enough): there is indeed no condition on $(\delta_{i,k})$ to find $(\lambda_1, ..., \lambda_m)$ satisfying Eq. 7.28. There are however choices of $(\delta_{i,k})$ that are better than others in order to minimise the error, as will be expanded in the next subsection.

7.2.5 Bounding errors

Theorem 1 establishes the existence (and a construction in its proof) of the coefficients $(\lambda_1, ..., \lambda_m)$ that are involved in the estimate of a. However, the error bound I derived depends on $\|\vec{\lambda}\|_1$, hence on the set of implementable observables A'_k: if the A'_k's ($\vec{\delta}_k$'s) are almost all identical, the underlying linear system of Eq. 7.27 will be very poorly conditioned, and $\|\vec{\lambda}\|_1$ might explode. It is therefore important to control $\vec{\lambda}$ by noting that its choice is not unique, and that some $\vec{\delta}_k$'s may lead to better solutions.

Further, controlling $\vec{\lambda}$ appears necessary to tame fluctuations of the estimated eigenvalue a due to imperfect evaluations of the eigenvalues a'_k. It is particularly relevant in our problem setting as these eigenvalues will likely be measured via phase estimation, whose outcome statistically varies around the exact eigenvalue. If each a'_k is estimated with variance σ^2, that of a can be expressed as

$$\mathbb{V}\left(\sum_{k=1}^{m} \lambda_k a'_k\right) = \sigma^2 \sum_{k=1}^{m} \lambda_k^2 \tag{7.30}$$

where $\mathbb{V}$ denotes the variance. Thus, having some control over $\vec{\lambda}$ would both prevent target-estimate errors and amplification of PE statistical variations.

Consequently, Theorems 2 and 3 aim at bounding $\vec{\lambda}$ in two relevant cases: when $p = 1$ (first-order correction) and when $N = 1$ (mono-variate case).

Theorem 2. *Suppose that one wants to correct up to first order ($p = 1$) and that $\vec{0}$ is in the convex hull of all implementable $\vec{\delta}_k$'s. It follows that there exists $\vec{\lambda}$ satisfying both Eqs. 7.28 and 7.29, and:*

$$||\vec{\lambda}||_1 = 1 \tag{7.31}$$

Theorem 3. *Suppose that the implementable observables are parameterised with a single variable ($N = 1$, $\vec{\delta}_k := \delta_k$) and that the set Δ of implementable δ_k's satisfies:*

$$\exists c > 0 \quad \forall \delta \in \Delta \quad |\delta| \leq c \tag{7.32}$$

$$\exists d > 0 \quad \forall (\delta, \delta') \in \Delta^2 \quad \delta \neq \delta' \Longrightarrow |\delta - \delta'| \geq d \tag{7.33}$$

If follows that there exists $\vec{\lambda}$ satisfying both Eqs. 7.28 and 7.29, and:

$$||\vec{\lambda}||_2 = O(1) \tag{7.34}$$

Remark 3.1. *Note that the first result is on the ℓ^1-norm while the second one is on the ℓ^2-norm. For any $\vec{\lambda} = (\lambda_1, ..., \lambda_m)^T$, they are related by:*

$$||\vec{\lambda}||_2 \leq ||\vec{\lambda}||_1 \leq \sqrt{m}||\vec{\lambda}||_2 \tag{7.35}$$

The proof of these theorems can be found in Appendix D. Theorem 2 gives the configuration of the $\vec{\delta_k}$'s that provides a $\vec{\lambda}$ achieving minimal $\|\vec{\lambda}\|_1$. Indeed, given the normalisation condition $\sum \lambda_k = 1$:

$$||\vec{\lambda}||_1 \geq \sum_k \lambda_k = 1$$

and

$$||\vec{\lambda}||_1 = 1 \iff \forall k \ \lambda_k \geq 0$$

This exactly means that $\vec{0}$ is in the convex hull of all implementable $\vec{\delta_k}$'s. Also note that this configuration brings the ℓ^2-norm closer to its minimum $1/\sqrt{m}$, which is achieved when all λ_k's are equal and positive (using Cauchy-Schwarz inequality).

As for Theorem 3, it shows, in a simpler case than the general one, that the chosen $\vec{\delta_k}$'s must be separated enough to build the eigenvalue estimator with small variance. Otherwise, the linear system of Eq. 7.28 will have a high condition number, thus yielding high λ_k's.

It seems harder to assert anything for all the other cases than $p = 1$ or $N = 1$. In particular, the latter cannot be adapted to $N \geq 2$ as a much more complex matrix than the Vandermonde matrix would appear in the proof at Eq. D.3. It would not be clear where the singular values of that matrix vanish, compared to the well-known case of the Vandermonde matrix: it is thus harder to give an explicit condition such as the one of Eq. 7.33. In practice however, the numerical simulations of the next section show that the eigenvalue estimate remains robust even when $p \geq 2$ and $N \geq 2$.

7.3 Numerical results

In this section, I test my error mitigation protocol on one-dimensional Ising and Heisenberg XYZ Hamiltonians with n spins:

$$H_{\text{Ising}} = \sum_{i=1}^{n} a_i Z_i + \sum_{i=1}^{n} b_i X_i X_{i+1} \tag{7.36}$$

$$H_{\text{XYZ}} = \sum_{i=1}^{n} a_i Z_i + \sum_{i=1}^{n} (b_i X_i X_{i+1} + c_i Y_i Y_{i+1} + d_i Z_i Z_{i+1}) \tag{7.37}$$

where an index $n+1$ is taken mod n hence refers to spin 1. The coefficients a_i, b_i, c_i and d_i are chosen randomly from a uniform distribution between 0 and 1, but only once for each chain length. The objective is to estimate the ground-state energy of the target Hamiltonian H, via the measurement of the ground-state energy of implementable Hamiltonians H'_k. The set of implementable Hamiltonians varies from subsection to subsection, depending on the assumptions on the implementation of $\mathcal{W}(H)$. My numerical simulations do not use any quantum circuit simulator. The implementable Hamiltonians are directly diagonalised, which is achievable for small enough spin chains. Running full circuit simulations would require the compilation of phase estimation in terms of a universal set of gates that is implementable on a fault-tolerant quantum computer *e.g.* Clifford+T, which was outside the scope of my work.

7.3.1 p-th order mitigation of Trotter errors

Algorithmic errors only

In this subsection, I evaluate the efficiency of my protocol in the absence of phase estimation or logical errors, in order to solely focus on algorithmic errors reduction. Besides, I consider the simplest case of singly-parameterised Hamiltonians ($N = 1$). Such a situation is for instance relevant for trotterised Hamiltonians, where this parameter is the timestep δt (Eq. 7.3). I use the XYZ model in this demonstration: in this case the Hamiltonian can be split into two sub-Hamiltonians H_A and H_B obtained by respectively summing for even and odd i in Eq. 7.37 [226]. By compiling $\mathcal{W}(H') = \left(\mathrm{e}^{-\mathrm{i}H_A\delta t}\mathrm{e}^{-\mathrm{i}H_B\delta t}\right)^{t/\delta t} = \mathrm{e}^{-\mathrm{i}H't}$, the effectively implemented

Hamiltonian $H' = \tilde{H}(\delta t)$ thus reads:

$$H' = \tilde{H}(\delta t) = \frac{\mathrm{i}}{\delta t}\log\left(\mathrm{e}^{-\mathrm{i}H_A\delta t}\mathrm{e}^{-\mathrm{i}H_B\delta t}\right) \tag{7.38}$$

Let us denote $N_{\text{Trotter,max}}$ the maximum number of Trotter steps a device can tolerate. While the best possible H' is obtained by using $N_{\text{Trotter,max}}$ Trotter steps, multiple H'_k can be implemented by reducing the number of Trotter steps, effectively increasing the algorithmic errors [233]. Another possibility is to note that δt can be chosen to be negative:

$$\begin{aligned}\tilde{H}(-\delta t) &= -\frac{\mathrm{i}}{\delta t}\log\left(\mathrm{e}^{+\mathrm{i}H_A\delta t}\mathrm{e}^{+\mathrm{i}H_B\delta t}\right)\\ &= \frac{\mathrm{i}}{\delta t}\log\left(\left(\mathrm{e}^{+\mathrm{i}H_A\delta t}\mathrm{e}^{+\mathrm{i}H_B\delta t}\right)^{-1}\right)\\ &= \frac{\mathrm{i}}{\delta t}\log\left(\mathrm{e}^{-\mathrm{i}H_B\delta t}\mathrm{e}^{-\mathrm{i}H_A\delta t}\right)\end{aligned}$$

Hence, if $\tilde{H}(\delta t)$ is implementable, $\tilde{H}(-\delta t)$ surely is too, as it just involves swapping H_A and H_B. This trick allows one to halve the level of increase of algorithmic error that is needed to implement enough H'_k's. For a given level of error mitigation p, Eq. 7.27 is a system of $p+1$ linear equations (as $N = 1$ for vector $\vec{x}_k$ takes the form $(\delta_k^i)_{0\leq i\leq p}$ hence is of size $p+1$). Hence it has a non-zero solution $(\lambda_1, ..., \lambda_m)$ for $m \geq p+1$. I thus choose the set Δ of implemented δt_k to be:

$$\Delta = \{k \times \delta t_{\min},\ k \in [\![-p/2, p/2+1]\!] \setminus \{0\}\} \tag{7.39}$$

After obtaining $\vec{\lambda}$ corresponding to these δt_k's, I estimate the ground state energy of the original Hamiltonian via Eq. 7.21, using the ground state energies of the H_k's computed by exact diagonalisation.

The results of such an approach are presented in Fig. 7.1, where the error in the ground-state energy estimation is plotted against $N_{\text{Trotter,max}}$ for $p = 0$ (no error mitigation), $p = 2$ and $p = 4$ (second- and fourth-order error mitigation). The straight lines in log-log scale indicate a power law:

$$\text{error} = \left(\frac{1}{N_{\text{Trotter,max}}}\right)^{\alpha}$$

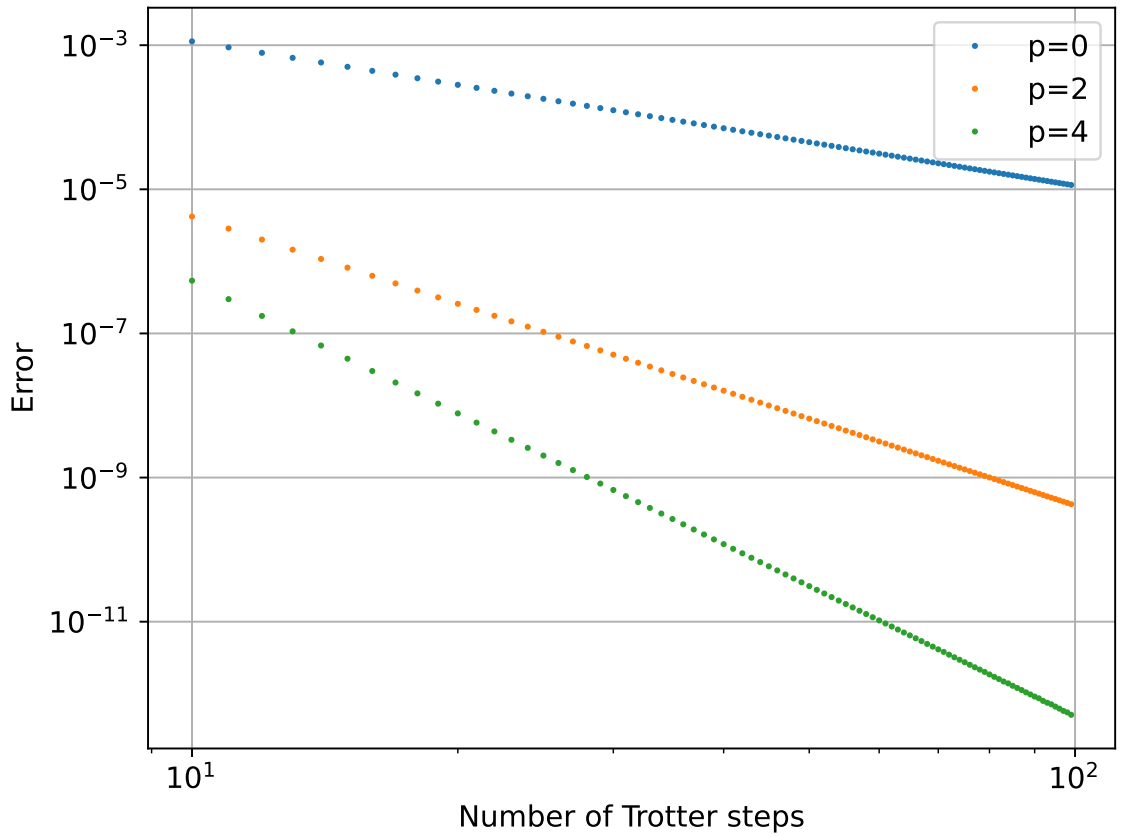


Figure 7.1: Error in the ground-state energy estimation of an XYZ Hamiltonian with $n = 6$ particles implemented by Trotterisation, plotted against the maximum number of Trotter steps. The coefficients of the Hamiltonian are randomly chosen before all the experiments. Each colour corresponds to a different order p of error mitigation, $p = 0$ meaning no error mitigation is performed.

For a given p, one would expect a p-th order estimate, hence $\alpha = p + 1$. However, a linear regression for each plot of Fig. 7.1, for $p = 0$, 2 and 4, respectively suggests $\alpha = 2$, 4 and 6. This indicates that the ground state has no odd order component in its Taylor expansion: correcting for, say, second order, one only retains a fourth order error. This feature was noted in [226] for the first-order component, given by $E^{(1)} = \langle 0| H' - H |0\rangle$. They noticed that after Trotterisation, $H' - H$ was off-diagonal for a wide category of Hamiltonians, including any Hamiltonian with nearest-neighbour interactions like H_{XYZ} employed here. The results of Fig. 7.1 hence suggest that this is not only true for first order, but for all odd orders. For even p, it thus becomes possible to obtain a $p + 1$-th order estimate of the ground state with only $p + 1$ approximate ground-state energy estimations.

Inclusion of random noise

Further, a natural question to ask is how robust this protocol is to residual random errors afflicting the logical qubits, and to statistical fluctuations due to the use of a finite number of ancilla qubits in phase estimation. Indeed, if my scheme does guarantee the suppression of algorithmic errors, one would not want other errors to accumulate in the process.

The description of phase estimation sampling errors is well understood. If the target eigenvalue is $\mathrm{e}^{-\mathrm{i}2\pi\phi}$, the output distribution is given by:

$$\mathbb{P}(y = y_0, \ldots, y_{n_{anc}-1}) = \frac{1}{2^{2n_{anc}}} \left| \frac{1 - \mathrm{e}^{2\mathrm{i}\pi\delta(y)2^{n_{anc}}}}{1 - \mathrm{e}^{2\mathrm{i}\pi\delta(y)}} \right|^2 \tag{7.40}$$

where n_{anc} is the number of ancillae used in phase estimation, $\mathbb{P}(y = y_0, ..., y_{n_{anc}-1})$ is the probability of observing the output state $|y_0, ..., y_{n_{anc}-1}\rangle$, and $\delta(y) = \phi - y = \phi - \sum_{i=0}^{n_{anc}-1} y_i/2^{i+1}$. I do not take into account additional fluctuations stemming from the sampling of other eigenvalues than the ground state (arising when the input state of PE is not exactly the ground state of the implemented Hamiltonian). This is a fair assumption if the ground and first excited states are separated enough: in this case, any outlier in the sampled data can be identified and ignored, and PE can be performed again.

As for logical gates errors, exactly modelling how they ultimately affect the output of the algorithm is quite complex. It would require compiling phase estimation and Trotterisation using a universal set of gates accessible to error-corrected quantum computers *e.g.* Clifford+T. Rather, I adopt a much simpler error model that would still give a good understanding of the impact of this noise type. Namely, I assume that every Trotter step $\mathrm{e}^{-\mathrm{i}H'\delta t_k} = \mathrm{e}^{-\mathrm{i}H_A\delta t_k}\mathrm{e}^{-\mathrm{i}H_B\delta t_k}$ (see Eq. 7.38) is implemented up to an error q_1. This means that the accumulated noise stemming from all these imperfect implementations will be of the form $q_1 N_{\mathrm{Trotter}} n_{anc}$, where n_{anc} is the number of ancilla qubits used in phase estimation. Indeed, n_{anc} different total unitary evolutions must be implemented within phase estimation. The output is then obtained by applying an inverse quantum Fourier transform, whose induced

noise is of the form $q_2 n_{anc}$ (no dependence on N_{Trotter}). For simplicity I will assume that $q_1 = q_2 := q$.

To quantify the impact of the errors, I will compute the average Euclidean distance between my estimate $\tilde{E}$ for the ground-state energy of H and its exact value E: $\sqrt{\mathbb{E}((\tilde{E}-E)^2)}$, where $\mathbb{E}$ denotes the expectation value. This quantity is evaluated by rerunning the same experiment as in Fig. 7.1 but with additional PE sampling errors and logical gate random noise. Namely, the output of running PE on H'_k is now sampled according to Eq. 7.40. Then noise is injected by using a Normal distribution centred around the obtained output, with width $\sigma = q n_{anc}(N_{\text{Trotter}} + 1)$. Because of the randomness of the current experiment, each data point is also averaged over $N_{\text{runs}} = 10,000$. The results are plotted in Fig. 7.2, for $n_{anc} = 16$ and different values of q.

For a low number of Trotter steps, *i.e.* when the level of the algorithmic errors is well above the level of the other noise sources, the data from Fig. 7.1 is unchanged. However, because of the presence of additional fluctuations that my protocol cannot correct, the error rate does not decrease towards 0 but now starts to increase as N_{Trotter} becomes larger. This is because the gate-induced noise grows with N_{Trotter}, which controls the depth of the phase estimation algorithm. Quite expectedly, the addition of random noise makes it apparent that there exists a tradeoff between algorithm errors (minimised at high N_{Trotter}) and gate noise (minimised at low N_{Trotter}). The Euclidean distance can be reworked as follows:

$$\begin{aligned}\sqrt{\mathbb{E}((\tilde{E}-E)^2)} &= \sqrt{\mathbb{V}(\tilde{E}-E) + (\mathbb{E}(\tilde{E}-E))^2} \\ &= \sqrt{||\lambda||_2^2 \sigma'^2 + (\bar{E}-E)^2}\end{aligned} \tag{7.41}$$

where $\bar{E}$ is the mean value of the estimate and σ' is the variance of a single eigenvalue evaluation (this parameter encapsulates both the gate noise σ and PE sampling errors). Eq. 7.30 was used between the first and second lines. In the regime of low algorithmic errors, the second term is negligible and one obtains

$$\sqrt{\mathbb{E}((\tilde{E}-E)^2)} \propto ||\lambda||_2 N_{\text{Trotter}} \tag{7.42}$$

In this regime of dominating random noise, one observes that as the order of error mitigation p increases, the error increases too, as $||\lambda||_2$ does. This growth is not dramatic though, as $||\lambda||_2 = 1$, 1.4 and 2.4 for respectively $p = 0$, 2 and 4. Besides, before this regime is reached, the error suppression promised by my error mitigation strategy is still achieved: for instance, in all plotted noise cases and on an early-fault tolerant device where only 10 Trotter steps may be reliably implementable, my error mitigation protocol always outperforms the no-mitigation case, still at the very low cost of 3 or 5 ground state energy evaluations.

Therefore, the addition of random noise to the simulations clearly shows the power of my error mitigation strategy: algorithmic errors are still suppressed, and random noise is only moderately amplified (at most multiplied by 2.4 for fourth-order mitigation). The effect of the random noise could even be lowered if $||\lambda||_2$ could be chosen to be smaller than 1: this is difficult in the case of singly-parameterised Hamiltonians as the parameter space is small, but will become easier in the next sections focusing on multi-parameter Hamiltonians. In the rest of the chapter, I will thus assume that we are in the regime where gate errors are well-below algorithmic errors, thereby neglecting them. Phase estimation statistical errors will be kept however. $||\lambda||_2$ will also always be engineered to be lower than 1, guaranteeing no error amplification.

7.3.2 p-th order mitigation of qubitised Hamiltonians

The previous section addressed the case of singly-parameterised observables, with the example of trotterised Hamiltonians. In this subsection, I tackle the case of multiple parameters ($N > 1$), with the example of qubitised Hamiltonians. As explained in Eq. 7.4, given a target Hamiltonian expressed as a linear combination of unitaries $H = \sum c_i P_i$, the effectively implemented Hamiltonians by qubitisation are $H'_k = \sum c'_{i,k} P_i$. I here absorb the renormalisation constants and assume $\sum_i c_i = \sum_i c'_{i,k} = 1$.

For all i and k, the coefficients c_i and $c'_{i,k}$ are between 0 and 1 and $\delta_{i,k} = c'_{i,k} - c_i$ is close to 0. Following the procedure designed in [186], $c'_{i,k}$ is chosen to be a μ-bit

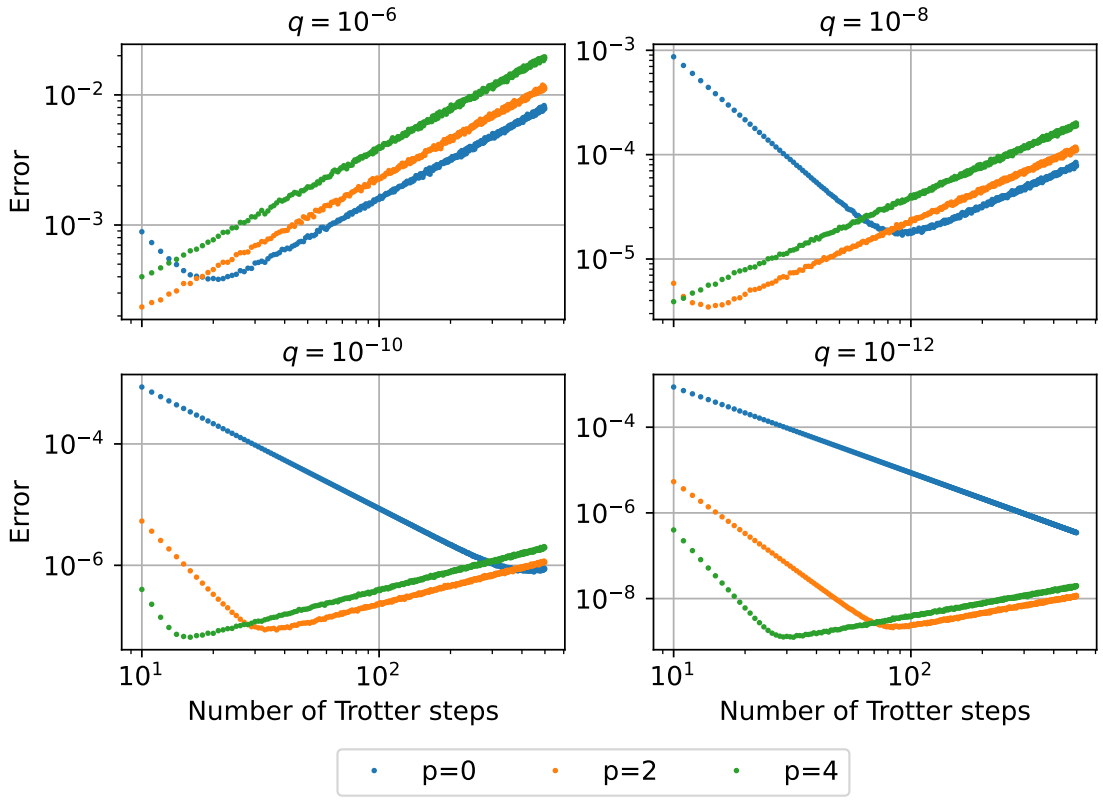


Figure 7.2: Average Euclidean distance between the estimate $\tilde{E}$ for the ground-state energy of H and its exact value E, $\sqrt{\mathbb{E}((\tilde{E}-E)^2)}$, in the presence of finite logical errors q (see main text for the exact definition of the parameter) and phase estimation sampling errors. The same set of parameters as in the experiment of Fig. 7.1 is used. Each data point corresponds to an average value over $N_{\text{runs}} = 10,000$ experiments. Each subplot corresponds to a different level q of logical noise, and the number of ancilla qubits for PE is set to $n_{anc} = 16$.

number close to c_i (either the closest, second closest or third closest, in order to have multiple implementable H'_k), before normalising all $c'_{i,k}$'s:

$$c'_{i,k} = \text{round}(2^\mu c_i)/2^\mu + \epsilon_{i,k}, \quad \epsilon_{i,k} \in \{-1/2^\mu, 0, 1/2^\mu\}$$
$$c'_{i,k} \leftarrow \frac{c'_{i,k}}{\sum_i c'_{i,k}}$$

where round(x) returns the closest integer to the float x. $(\epsilon_{i,k})$ should be chosen such as to minimise the norm of $\vec{\lambda} = (\lambda_1, ..., \lambda_m)^T$. In the present study, I did not try to optimise this choice: for a given set of $\epsilon_{i,k}$'s, one can obtain $c'_{i,k}$ and $\delta_{i,k}$, then compute a $\vec{\lambda}$ of minimum ℓ^2-norm satisfying both Eqs. 7.28 and 7.29. My approach is thus to repeatedly choose at random the set of $\epsilon_{i,k}$'s till this $\vec{\lambda}$ satisfies

a certain condition (norm less than 1 or positive coefficients). If the condition cannot be met, the number m of implemented Hamiltonians is incremented, thereby giving more freedom in the choice of $\vec{\lambda}$.

In addition, I here consider sampling errors arising from the use of the phase estimation algorithm. The difference compared to the previous section is that PE is applied to $\mathcal{W}(H) = \mathrm{e}^{-\mathrm{i}\arccos(H)}$ rather than $\mathrm{e}^{-\mathrm{i}H}$. Therefore, if PE outputs a value $y = y_0 \dots y_{n_{anc}-1}$ (in bit form) then the energy estimate is given by

$$E = \cos(2\pi y) \tag{7.43}$$

The procedure of the numerical experiment performed here is thus as follows. First, I repeatedly generate $(\epsilon_{i,k})$ randomly to construct sets $\{H'_k\}$ of implementable Hamiltonians, and stop when sufficiently good $\vec{\lambda}$ is obtained. Second, I compute the ground state energies of the retained H'_k's by exact diagonalisation. Third, I sample once the output of PE from 7.40 and compute the associated energies with 7.43, using $n_{anc} = 16$ ancillae. Fourth, I calculate the target eigenvalue estimator using $\vec{\lambda}$ and the sampled energies. Given the inherent randomness of this protocol, the four steps above are then repeated 10,000 times, hence giving 10,000 different estimates of the target energy, which are represented with histograms in Fig. 7.3.

The above protocol is applied to estimate the ground-state energy of an Ising chain with $n = 8$ particles. This means 16 unitaries in the decomposition of H, which translates into $N = 16$ parameters $\delta_i = c'_i - c_i$. Each subplot corresponds to a different level of algorithmic errors (high μ meaning low error) and the different colours to different error mitigation strategies: gray is first order and enforcing $||\vec{\lambda}||_2 < 1$, blue is first order and enforcing $\lambda_k > 0$ for all k (which leads to $\|\vec{\lambda}\|_1 = 1$), and yellow is second order and enforcing $||\vec{\lambda}||_2 < 1$. The target ground-state energy is indicated with a black vertical dashed line. The ground-state energy one would get by applying phase estimation to the best implementable Hamiltonian H'_0 ($\epsilon_{i,0} = 0$ for all i) is plotted in red: it follows the distribution given by Eqs. 7.40 and 7.43. Because of the algorithmic errors, there is a bias between the target value (vertical dashed line) and the data. This bias is corrected by the application of my error

mitigation strategies. As expected, first order correction (blue and gray) is sufficient to cancel the bias at low enough algorithmic errors ($\mu = 10$) while second order correction (yellow) offers greater correction at any plotted μ. Besides, one can observe that the variance of the corrected data increases with the strength of the errors (decreasing μ) as the fluctuations in $\epsilon_{i,k}$ increase. Note that even for the same p, the variance of the corrected data depends on the error mitigation strategy, as it is also proportional to $||\vec{\lambda}||_2$. As expected from Theorem 2, choosing the $\epsilon_{i,k}$'s such that all λ_k's are positive is most optimal: this translates into a minimum ℓ^1-norm and brings the ℓ^2-norm closer to its minimum. This results in a narrower distribution for the blue than for the gray data.

It hence naturally appears that increasing the complexity of the error mitigation method (gray to blue to yellow) improves the quality of the output. This obviously comes at a cost which is studied in the next section.

7.3.3 Cost of the proposed scheme

The cost of the protocol can be estimated by the number of times n_{PE} phase estimation must be performed. In order to obtain a p-th order estimate of a given eigenvalue a, one must be able to find coefficients $(\lambda_1, ..., \lambda_m)$ satisfying Eqs. 7.28 and 7.29. This is always possible if m is greater than the size s of the vector $\vec{x}_k = (\delta_{1,k}^{i_1} ... \delta_{N,k}^{i_N})_{0 \leq |i| \leq p}$, which can be calculated as:

$$s = \sum_{r=0}^{p} \binom{N+r-1}{N-1} \tag{7.44}$$

where I used that the number of tuples $(i_1, ..., i_N) \in \mathbb{N}^N$ of fixed sum r is $\binom{N+r-1}{N-1}$. Note that this exact number was not given in Theorem 1 where I only used that $s = O(N^p)$ to bound m. In some cases though, the $\vec{x}_k$'s may have some internal linear dependencies, thereby reducing the rank of their matrix: less λ_k's would thus be required. This is for example the case for normalised qubitised Hamiltonians, for which $\sum_i c_i = \sum_i c'_{i,k} = 1$, meaning that, for all k, $\sum_i \delta_{i,k} = 0$. This identity reduces the rank by 1 for $p = 1$, and creates even more linear dependencies for higher p.

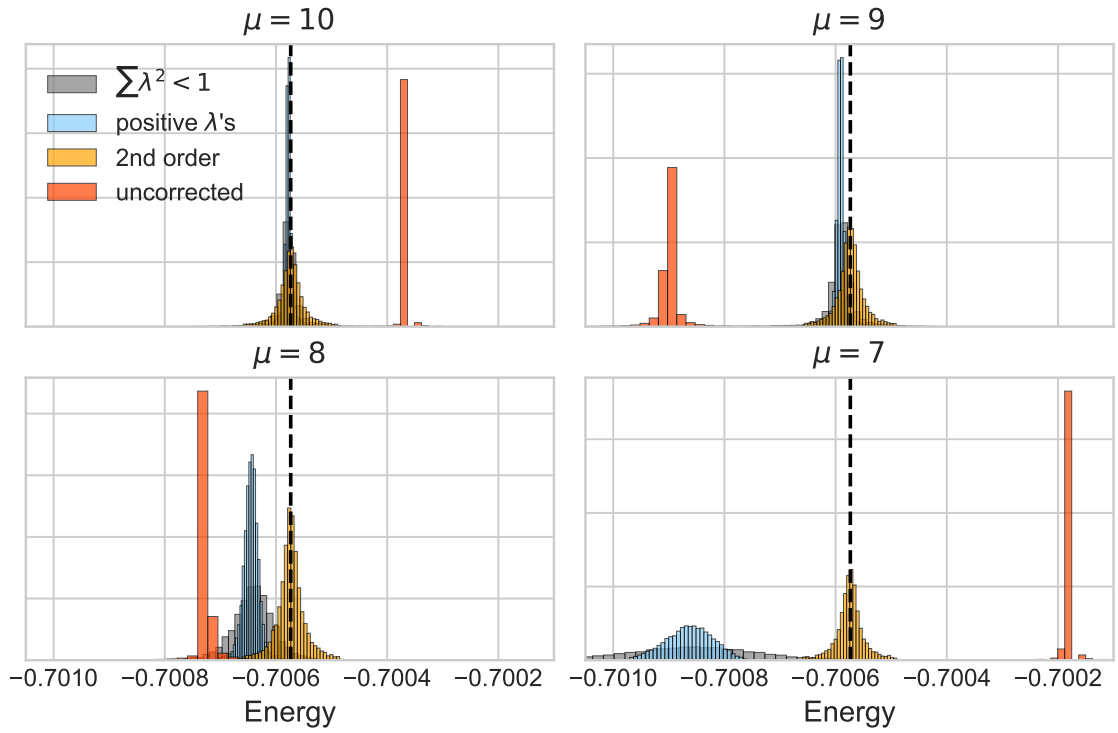


Figure 7.3: Distribution of the estimated ground-state energy of an Ising Hamiltonian with 8 particles and random coefficients, with phase estimation errors taken into account. PE is applied to $\mathcal{W}(H) = \mathrm{e}^{-\mathrm{iarccos}(H)}$, implemented by qubitisation. This requires the preparation of a state $|G\rangle$, which can only be done up to a certain precision proportional to $1/2^{\mu}$. The red data corresponds to the raw output obtained from PE, while the others are post-processed with my error mitigation scheme: gray and blue correspond to first order correction with two different conditions on $\vec{\lambda}$, and yellow corresponds to second order.

Besides, it is important to note that, for a given p, choosing the lowest possible m allowing one to construct $(\lambda_1, ..., \lambda_m)$, which I denote by $m_{p,\mathrm{min}}$, is not the most clever choice. Indeed, as the parameter m (*i.e.* the number of variables of the linear system $X\vec{\lambda} = b$ of Eq. 7.27) increases, the solution space also expands, potentially permitting a smaller minimum norm for $\vec{\lambda}$. Consequently, this leads to a reduced error since the bound on δa, as stated in Theorem 1, depends on $||\vec{\lambda}||_1$.

I thus conduct a final experiment to gain insights on the optimal choice of m. In Fig. 7.4, I plot the error in the ground-state energy estimation of a qubitised Ising Hamiltonian of varying length, using the same protocol as in the previous subsection but with only one random choice of $\epsilon_{i,k}$ (without trying to meet any additional condition on $\vec{\lambda}$). Each data point corresponds to the average distance

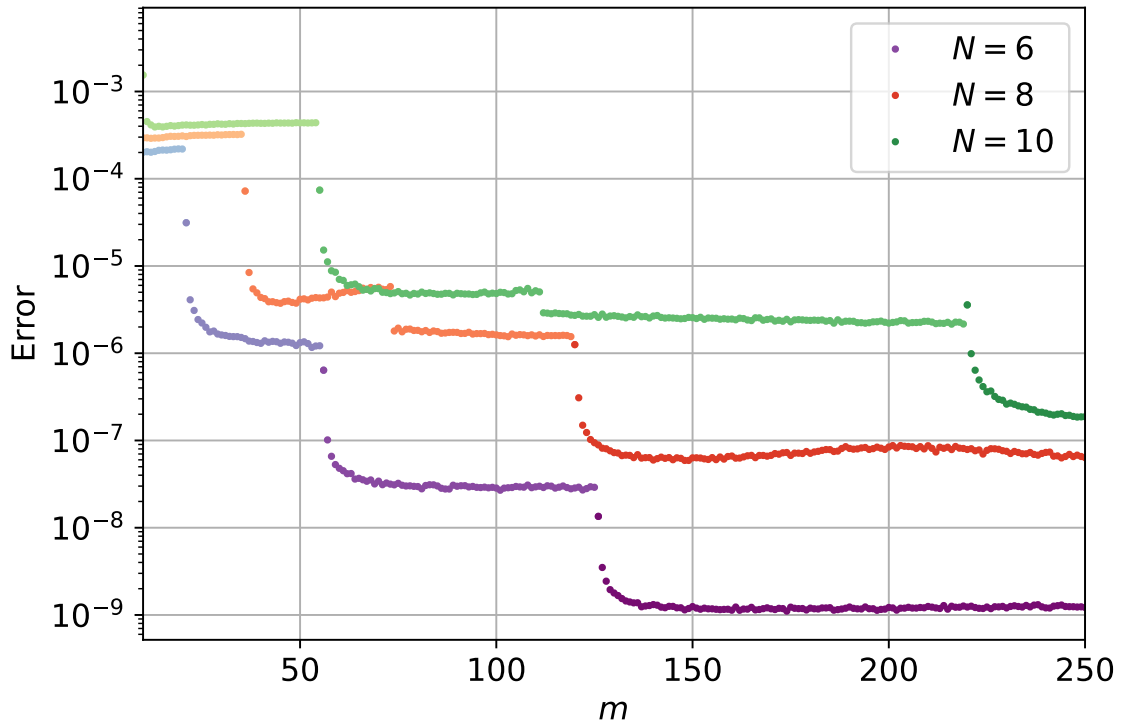


Figure 7.4: Error in the ground-state energy estimation of a qubitised Ising Hamiltonian with $N = 6$, 8 and 10 terms, plotted against the number m of distinct Hamiltonians implemented. Each distinct colour corresponds to a different number of terms in the Hamiltonian, while the slight changes of shade happening when the error drops coincide with a change in error mitigation order p.

between the estimator and the target eigenvalue, over 10,000 experiments. The highest possible order of error mitigation p is always chosen, *i.e.* whenever m exceeds the rank of the matrix X of the $\vec{x}_k$'s. An increase in p is, as expected, characterised by a drop in the error, and is highlighted by a change of shade of the same colour. This drop is however relatively smooth: Fig. 7.4 shows that choosing $m = m_{p,\min}$ fails to significantly improve the results of the $(p-1)$-th order mitigation. In turn, a slightly higher m yields results that faithfully achieve p-th order mitigation. This increase of m can however remain very moderate, only a few units (say, 10), as for a given p the error quickly stabilises. Therefore, to achieve minimum error for a given p, one can set:

$$m = m_{p,\min} + 10 = O(s) \tag{7.45}$$

Assuming that the initial state used for phase estimation has an overlap β with the real ground state and that the first excited state is separated enough from

the ground state to differentiate them, I conclude that the number of times n_{PE} phase estimation must be performed can be written as:

$$n_{\mathrm{PE}} = O(m/\beta) = O(s/\beta)$$

Thus:

$$n_{\mathrm{PE}} = \begin{cases} O(N^p/\beta), \text{ if } N \neq 1 \\ O(p/\beta), \text{ if } N = 1 \end{cases} \tag{7.46}$$

The number of times phase estimation must be performed is thus polynomial in the number of parameters the Hamiltonian depends on. In the case of singly-parameterised Hamiltonians, this number is even linear in the desired order of mitigation, enabling powerful error reduction at very low cost.

7.4 Discussion

In this chapter, I thus presented an extension of Richardson extrapolation, introducing several key novelties. Specifically, these include the mitigation of eigenvalues rather expectation values, and the use of a multi-parameter formalism without which Richardson extrapolation would fail to work for qubitised Hamiltonians.

My scheme targets algorithmic errors, that is errors arising from approximations in the quantum algorithm implementing the unitary passed to phase estimation (*e.g.* $\mathcal{W}(H) = \mathrm{e}^{-\mathrm{i}Ht}$). After proving the theoretical performance of the proposed method, I numerically confirmed its efficiency and low resource requirements. In particular, I showed that for some relevant cases, such as that of trotterised Hamiltonians, a p-th order estimate of the ground-state energy can be obtained with order of p (independent) uses of phase estimation, thereby drastically reducing the error at very low cost. On top of this, I conducted a novel theoretical and numerical study aiming at better understanding how the chosen noise parameters and number m of implemented observables impact the final error rate (via $||\vec{\lambda}||_2$). As such, I examined how these parameters can be fine-tuned to minimise the error. In particular, I extended the study of zero-noise extrapolation to a more general framework where negative parameters are allowed, showing that they can indeed be conducive to more optimal results.

Although my protocol can be applied to the estimation of the eigenvalues of any observable, it is however important to note that it is only effective against fully known errors. Indeed, this scheme entirely relies on the full knowledge of the parameters ($\delta_{i,k}$) the implementable Hamiltonians depend on. As it might be unrealistic to assume that random errors will entirely be suppressed in early-stage fault-tolerant quantum computers, I however showed that additional random noise, albeit non correctable by my protocol, will not accumulate. This justifies the robustness and usability of my scheme even in the very first fault-tolerant devices.

As for future directions, my method could be improved by giving a more systematic way to choose *good* $\delta_{i,k}$'s out of the set of implementable ones, so as to minimise the norm of $\vec{\lambda} = (\lambda, ..., \lambda_m)^T$. In the previous subsections, the $\delta_{i,k}$'s were chosen randomly till some conditions on $\vec{\lambda}$ were satisfied. If this yielded good results (in particular still guaranteed correction at the desired order p), a more optimised approach could help minimise the prefactor $||\vec{\lambda}||_1$ in the error term. As a result, one would understand more systematically the number m of λ_k's that are needed to obtain, for a given p, a minimal error.

8
Conclusion

In this thesis, I present several advancements towards the task of building a functional quantum computer. One of the cornerstones of reliable and long-term quantum computing is quantum error correction, which promises to significantly reduce the noise and reach sufficiently low error rates to run meaningful quantum algorithms. Not all noise processes are easy to correct however: in Chapter 4 I focused on one of the most daunting error sources, that is large-scale defects affecting entire regions of a code. I theoretically and numerically demonstrated that the use of tailored protocols could drastically tame these noise processes, and only at a moderate resource overhead. While the work of this chapter was fairly abstract, in Chapter 5 I zoomed in on one specific implementation of quantum error correction, targeted at near-term silicon spin qubit processors: a 2×N array of qubits equipped with shuttling. Less challenging to engineer compared to more common dense 2D grids, this architecture promises both an efficiently-implementable universal set of gates, and sufficient error correction to run quantum algorithms beyond the classically tractable regime. Further, the protocols I designed rely on the application of global single-qubit gates, as single-qubit addressability is highly challenging for spin qubits (at least in the case where qubits are encoded using one electron). While this complicated the above protocols, Chapter 6 offers an opportunity to reconsider

this statement: I there demonstrated that shuttling electrons while performing single-qubit gates would result in enhancing their global addressability. This promises to alleviate electronics requirements and simplify spin qubit control. Finally, in Chapter 7 I complement the previous chapters with an error mitigation scheme suited for early-stage fault-tolerant computers, where errors will be low yet finite.

Throughout the three years of my doctorate, I thus addressed the question of building stable quantum computers from various angles. In particular, I strived to always steer away from excessively theoretical considerations or to forget about the reality of an experiment. Asymptotically optimal quantum error correcting codes have been demonstrated: what matters now is the construction of practical ones and the design of new algorithms that prove advantageous in a practical setup. In my opinion, theoretical research has the power to orient and motivate experimental advances, by proposing schemes that are tailored to specific qubit platforms, or even to specific device layouts. On the one hand, this means understanding the strengths of a given qubit implementation, in order to capitalise on the right assets when designing new schemes (*e.g.* making use of shuttling in silicon devices). On the other hand, it means acknowledging the qubit weaknesses, and test new schemes with precise noise models, accounting for the specificities of the device. This kind of research, requiring intimate collaboration between theorists and experimentalists is, to me, what is likely to bring the greatest advances in the next years.

Throughout my work with Quantum Motion during my doctoral research, I had the opportunity to focus on one specific qubit type: silicon spins. While I believe that they hold enough potential to enable large-scale quantum computation, they are however a less mature technology than *e.g.* superconducting qubits. But this only means one thing: there is still a considerable margin for improvement. Silicon spin qubit design is receiving growing attention, and its fast progress again relies on the joint work of theoretical and experimental teams. It relies for instance on the refinement of existing qubit control protocols, or a deeper understanding of underlying noise processes. With these ingredients, I believe that silicon spins

will quickly progress to a point where they can indeed benefit from their greatest strength: a large-scale deployment to millions qubits over minimal footprint.

For all these reasons, quantum computing is a fast-paced and exciting field to be in at the moment. I am looking forward to participating in the next advancements it will bring.

Appendices

A

Appendices for Adaptive surface code for quantum error correction in the presence of temporary or permanent defects

Contents

A.1 Probability model for the logical error rate in the presence of multiple defects

In Section 4.3.3, the logical error rate is computed as follows:

$$p_{\log} = \sum_{k \geq 0} \langle t_k \rangle \times p(\text{logical error} | n_{\text{def}} = k) \tag{A.1}$$

$$\langle t_k \rangle = \frac{\mathrm{e}^{-2L^2 \rho T} (2L^2 \rho T)^k}{k!} \tag{A.2}$$

In Equation A.1, $\langle t_k \rangle$ is the proportion of the time spent suffering k defects and $p(\text{logical error} | n_{\text{def}} = k)$ is the probability of a logical error over L rounds of

stabiliser measurements for a surface code suffering k defects. Hence, $p_{\log}$ describes the average probability of a logical error for a surface code potentially hit by multiple defects over L rounds of stabiliser measurements. This is the appropriate quantity to evaluate if compared to more canonical threshold calculations where, in the presence of measurement errors, the surface code is run for a number of rounds proportional to L in order to obtain a neat threshold.

I then model $\langle t_k \rangle$ in Equation A.2. To do so, I suppose that defects are uniformly distributed in time and mutually independent. Realistically, the number of defects X_i hitting the code between stabiliser rounds i and $i+1$ is a random variable following a Poisson law of rate $2L^2\rho$, with ρ the rate of defects per qubit and per round of stabiliser measurements — as the rotated surface code contains roughly $2L^2$ qubits. Assuming that each defect survives for T rounds and denoting t_{tot} the total time of the simulation, N_i the number of defects at round i and $\mathbb{P}(...)$ a measure of probability, then:

$$\langle t_k \rangle = \frac{1}{t_{tot}} \sum_i \mathbb{P}(N_i = k) \tag{A.3}$$

$$= \frac{1}{t_{tot}} \sum_i \mathbb{P}\left(\sum_{j=i-T+1}^{i} X_j = k \right) \tag{A.4}$$

$$= \frac{1}{t_{tot}} \sum_i \frac{\mathrm{e}^{-2L^2\rho T}(2L^2\rho T)^k}{k!} \tag{A.5}$$

$$= \frac{\mathrm{e}^{-2L^2\rho T}(2L^2\rho T)^k}{k!} \tag{A.6}$$

where, between steps A.4 and A.5, I use that a sum of T mutually independent Poisson laws of rate λ is a Poisson law of rate λT.

A.2 Evolution of the threshold of the adaptive surface code with the defect rate ρ

Fig. A.1 is a more detailed version of Fig. 4.8, where more values of ρ are included. This confirms that the threshold follows a straight line from the right to the left when ρ increases, as shown by the black dashed arrow of Fig. 4.8.

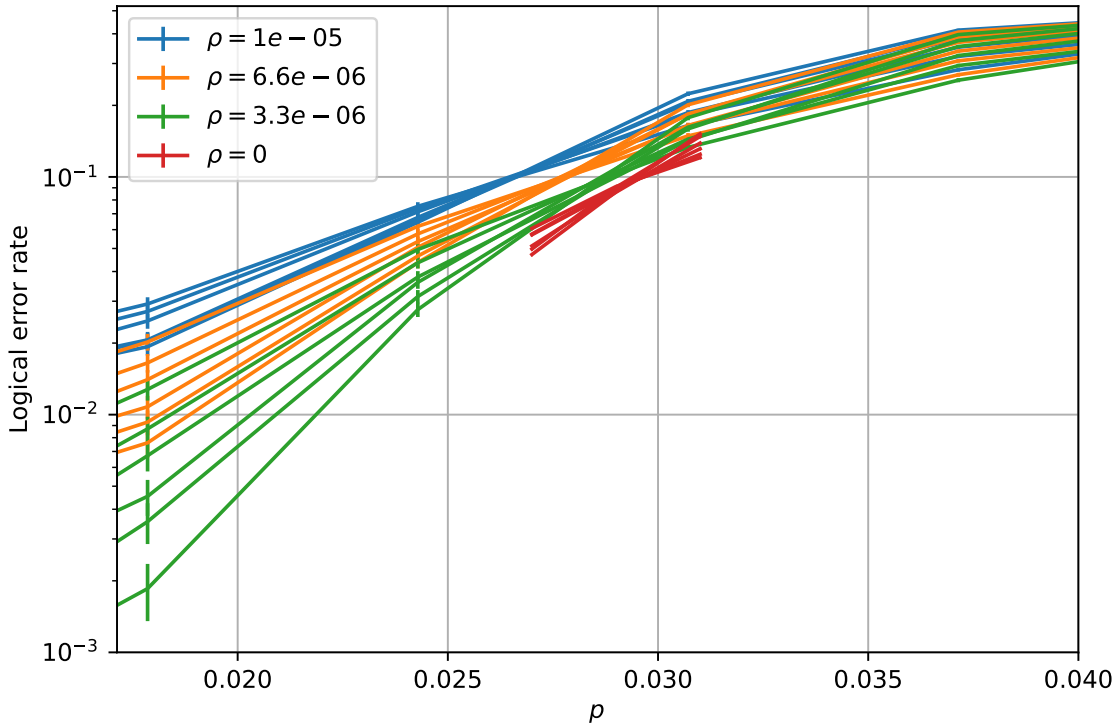


Figure A.1: Threshold plots for the adaptive surface code under phenomenological noise for $T = 100$ and various values of ρ. Lines of the same colour correspond to the same value of ρ but different code sizes (between $L = 9$ and $L = 21$).

B

Appendices for Towards early fault tolerance on a 2×N array of qubits equipped with shuttling

Contents

B.1 Surface code's stabiliser circuit gate ordering

Here I delve into the details of finding an optimal gate ordering for surface-code error correction on the 2×N architecture. Indeed, the usual pattern that is used for the regular rotated surface code with N- and Z-shaped orderings [120] leads to unnecessarily long shuttles. To see this, assume that the shuttling direction is vertical — two consecutive data qubits within the same column (resp. row) of the surface code are thus separated by 1 (resp. d) shuttling increment(s). Therefore, a Z-shaped ordering would require to shuttle four times along rows, and it would yield a total shuttling distance of roughly $4d$. My aim is to find an ordering that lets us

implement all gates of the stabiliser cycle with a total shuttling distance of roughly $2d$ (which corresponds to having the data qubits do one round trip, not two).

Furthermore, note that N- and Z- shaped orderings mentioned above do lead to distance-reducing hook errors in the wide surface code used in Section 5.3. Indeed, its X and Z logical operators both have a horizontal and a vertical representative. Instead, one can measure the stabilisers as shown in Fig. B.1. The shortest Z logical operators are horizontal, vertical or diagonal. The latter type passes through the centres of X stabilisers (red squares). However, using the measurement schedule represented by the gray arrow, Z hook errors are on either diagonal of the Z stabilisers (green squares), thus do not reduce the code distance. The same applies to X hook errors. Besides, with the same reasoning, one can prove that the regular rotated surface code is protected just as well from hook errors when this ordering is used. In summary, whether it is for the regular (Fig. 5.2) or wide (Fig. B.1) surface code, this cross-like sequence is the one I will consider.

Now, one could theoretically implement it by measuring the X and Z stabilisers separately on alternating rounds. However, this would unnecessarily increase the circuit depth and leave many qubits idle. Instead, a common solution is to interleave the gates of both stabilisers. This is possible provided the gates can be correctly commuted through each other so as to leave two neighbouring ancilla qubits disentangled (condition B.1); as well as gates can be implemented synchronously respecting the device layout and global shuttling of the data qubits (condition B.2).

Let me explain this more formally, and call time step k the interval between shuttles $k-1$ and k. At a given time step, some entangling gates must be implemented between certain ancilla-data qubit pairs. Let me use the notations of Fig. B.2, where each letter represents the time step when the ancilla and corresponding data qubit must be entangled.

Condition (B.1) is verified if and only if [124]

$$((b < e) \wedge (d < g)) \vee ((b > e) \wedge (d > g)). \tag{B.1}$$

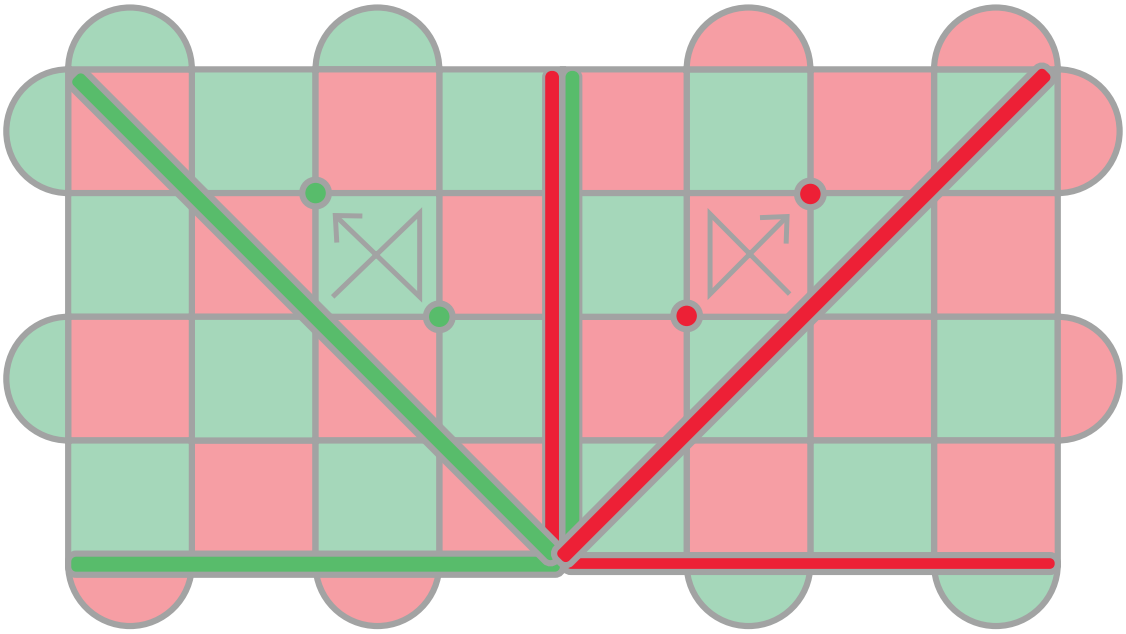


Figure B.1: Representation of the wide surface code. X and Z stabilisers are respectively represented with red and green squares or half-disks. Several shortest-length representatives of the logical X and Z operators are drawn with horizontal, vertical and diagonal lines. The gray arrows represent an order in which X and Z stabilisers can be measured to avoid hook errors reducing the distance of the code. Examples of X and Z hook errors are respectively drawn with red and green disks: as they do not coincide with the logical operators, they do not reduce the distance of the code. The same applies to the regular rotated surface code.

As for condition (B.2), it reads

$$(a \equiv e[s]) \wedge (b \equiv f[s]) \wedge (c \equiv g[s]) \wedge (d \equiv h[s]). \tag{B.2}$$

where s is the number of shuttles required to go back to the data qubits' initial position. This is because the device is laid out periodically and data qubits move as a whole along their shuttling track. This means that at any time step, ancilla qubits all face either their North-West, or North-East, or South-West, or South-East data qubit.

One last condition (B.3) can be added, enforcing the no-distance-reducing-hook-error ordering, which mathematically reads as

$$(c < b < d < a) \wedge (h < e < g < f). \tag{B.3}$$

Values for the time steps a to f respecting all three conditions are given in Fig. 5.5 (here $s = 4$). While their gates are indeed interleaved, X and Z stabilisers

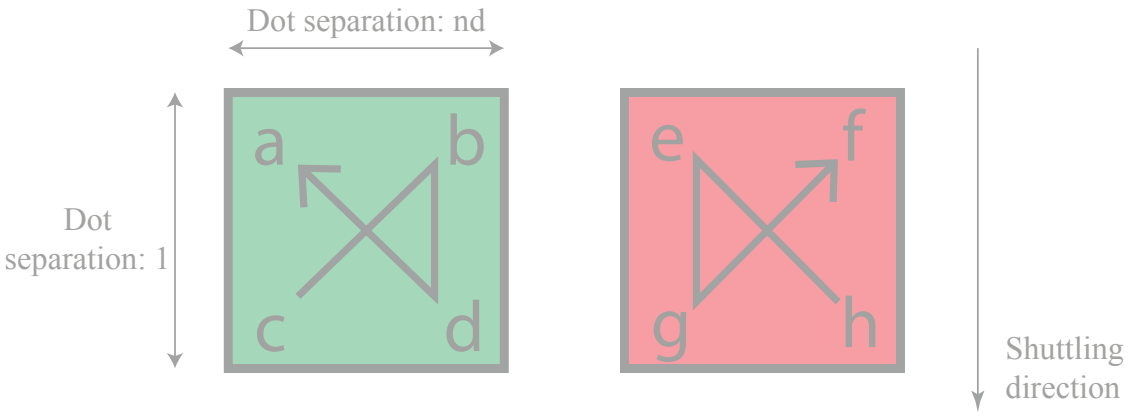


Figure B.2: Notations for conditions (B.1), (B.2) and (B.3).

are nonetheless operated on a staggered fashion. The whole operation sequence, including gates, measurements, initialisations and shuttles is the following:

1. entangle all ancilla qubits with their South-West data qubit
2. shuttle by $d-1$ increments forwards
3. entangle all ancilla qubits with their North-East data qubit; measure and reinitialise X ancilla qubits
4. shuttle by 1 increment forwards
5. entangle all ancilla qubits with their South-East data qubit
6. shuttle by $d+1$ increments backwards
7. entangle all ancilla qubits with their North-West data qubit; measure and reinitialise Z ancilla qubits
8. shuttle by 1 increment forwards
9. repeat

Note that the shuttling increments given here do not include the additional shuttling accommodating the ancilla bus of Region B in Fig. 5.4. Besides, for the first round of stabiliser measurements, X ancilla qubits should only start to undergo their

entangling gates from step 3. Similarly, for the last round, only X stabilisers should follow step 1 and 2.

With this protocol, in order to perform N_r rounds of X and Z stabilisers measurements, one thus needs $4N_r + 1$ shuttles and a total shuttling distance of $N_r(2d + 2) + d - 1$ (not including the logical ancilla bus of Fig. 5.4). One can also easily see that $4N_r$ and $N_r(2d + 2)$ are the respective lower bounds for the number of shuttles and total shuttling distance, no matter what gate ordering is chosen. Indeed, a given four-body stabiliser trivially requires four steps to entangle the ancilla with all the data qubits, thus four shuttles. Moreover, its North-West and South-East data qubits are separated by a distance $d + 1$, hence going back and forth between them requires a shuttling distance of $2d + 2$. Therefore, apart from a small correction arising at the last round due to the staggered implementation of the X and Z stabilisers, my solution is optimal.

B.2 Modelling for the valley degree of freedom

As explained in 2.2.6, fast shuttling can non-adiabatically excite the electron to the closest higher energy level: the excited valley state. As this state has been shown to exhibit a distinct g-factor from the ground valley state (where the electron normally sits) [38], the qubit could start to precess in a uncontrolled manner, causing unwanted phase rotations. This justifies the importance to control and estimate the impact of the excited valley state occupation. To model this, I will use an extension of the valley state modelling in [36].

Let me first introduce the position-dependent valley phase $\varphi_{VS}(x)$ and the bare valley splitting $E_{VS,0}$ (which models the valley splitting in the absence of the perturbations described below). The local two-level valley Hamiltonian can most generally be expressed as:

$$H_{loc}(x) = \frac{E_{VS,0}}{2} \left(\cos(\varphi_{VS}(x))\tau_x + \sin(\varphi_{VS}(x))\tau_y\right) \tag{B.4}$$

where τ_x and τ_y are Pauli operators in the valley subspace.

Assuming low coupling of the spatial and valley degrees of freedom, an electron of spatial probability distribution $\rho(x)$ would thus experience an average valley Hamiltonian:

$$H_v(x_0) = \int \rho(x - x_0) H_{loc}(x_0) \mathrm{d}x \tag{B.5}$$

Here I suppose that ρ is a Gaussian of standard deviation l_x. Two extreme cases can be considered for φ_{VS}: smoothly varying or abruptly changing owing to the presence of atomic steps. In the $v = 10\mathrm{m/s}$ regime, Fig. 1 of [36] shows that the smooth interface model leads to higher noise: this is thus the model I will adopt.

That paper focuses on the simplest case of a linear gradient model: $\varphi_{VS}(x) = a_x x$. I slightly refine it by assuming that the gradient is not constant but instead slowly varying at the scale of the electron wavefunction. This means that I can adopt all equations derived in [36], while assuming a slow variation of a_x to take disorder into account. In the instantaneous valley state basis, the final valley+orbital Hamiltonian is thus the following:

$$\tilde{H}_v(x_0) = \left(\frac{E_{VS}(x)}{2} \tau_z + \frac{\dot{\varphi}_{VS}(x)}{2} \tau_x \right) \otimes I + \frac{\Delta g(x)}{2} \frac{I - \tau_z}{2} \otimes \sigma_z \tag{B.6}$$

with:

$$E_{VS}(x) = E_{VS,0} \exp\left(-\frac{a_x(x)^2 l_x^2}{4} \right) \tag{B.7}$$

$$\dot{\varphi}_{VS}(x) = \dot{a}_x(x) x + a_x(x) v(x) \tag{B.8}$$

σ_z is the Pauli operator characterising the spin, $(I - \tau_z)/2$ is the projector onto the excited valley state and $\Delta g(x)$ models the difference in g-factor between the ground and excited valley states. I assume that the electron is smoothly shuttled back and forth along $2d$ dots separated by a distance l_{dd} (to account for the ancilla bus of Fig. 5.4), such that

$$v(x) = \begin{cases} v, & \text{if } x < 2dl_{dd} \\ -v, & \text{otherwise} \end{cases} \tag{B.9}$$

and

$$x(t) = \begin{cases} vt, & \text{if } t < 2dl_{dd}/v \\ -vt + 4dl_{dd}, & \text{if } 2dl_{dd}/v \leq t < 4dl_{dd}/v \end{cases} \tag{B.10}$$

To model the disorder, I describe $a_x(x)$ with a smooth random walk of the form:

$$a_x(x) = \sum_{k=1}^{n} \alpha_k \sin(\lambda_k x) \tag{B.11}$$

with $n = 20$ and λ_k chosen randomly between l_x and the maximum shuttled distance $2dl_{dd}$. These bounds guarantee a slow variation of the valley parameters on the scale of the electron wavefunction but over the whole landscape explored via shuttling. In the worst-case scenario, the ground and excited valley states should swap every l_x, meaning a variation of the valley phase of π. Thus a worst-case value for a_x is π/l_x. For an n-step random walk with unitary steps ($\alpha_k = 1$), the average maximum distance is $\langle \max_x a_x \rangle = \sqrt{\frac{n\pi}{2}}$. One can therefore randomly sample α_k between 0 and

$$\alpha_{\max} = \frac{\pi}{l_x}\sqrt{\frac{2}{n\pi}} = \frac{1}{l_x}\sqrt{\frac{2\pi}{n}} \tag{B.12}$$

so that with high probability a_x does not exceed π/l_x. The g-factor difference between ground and excited valley states $\Delta g(x)$ is modelled similarly with

$$\Delta g(x) = \sum_{k=1}^{n} \beta_k \sin(\mu_k x) \tag{B.13}$$

where $n = 20$ and μ_k is a random number between l_x and $2dl_{dd}$. A typical maximum value for Δg at a constant field of 1T is 100MHz [28], thus each β_k can randomly be sampled between 0 and $\sqrt{\frac{2}{n\pi}} \times 100\text{MHz}$.

Finally, in order to understand the phase accumulation over a whole stabiliser cycle, the valley+spin electron wavefunction is prepared in

$$|\psi_0\rangle = |0\rangle \otimes |+\rangle \tag{B.14}$$

I assume a shuttling speed $v = 10\text{m/s}$ and a dot separation $l_{dd} = 140\text{nm}$ as in the main text. The electron spatial distribution is set to $l_x = 50\text{nm}$, and I choose $E_{VS,0} = 15c0\mu\text{eV}$, so that in the worst-case of $a_x l_x = \pi$, one gets $E_{VS} = 15\mu\text{eV}$, which in the lowest possible range for E_{VS} [36] (therefore increasing non-adiabatic errors). I aim to estimate the influence of the code size d on the valley-induced dephasing noise, as it controls the overall shuttling distance. Further, I study the effect of additionally flipping the spin state via an X gate at the turning point

in the electron trajectory, in the spirit of dynamical decoupling. Indeed, ignoring any disorder, a flipped spin simply precesses back to its initial state when shuttled back. Hence one may expect that the same phenomenon happens in the presence of disorder and could therefore be used to reduce the noise. The evolution of the wavefunction over time is obtained by solving Schrodinger equation by time discretisation over 50,000 time steps. The final probability of a Z error is given by the overlap of the final state with the $|-\rangle$ spin state. In the presence of random disorder, I estimate the dephasing probability via Monte-Carlo simulations, by repeating the same experiment $N_{reps} = 10,000$ times.

Fig. B.3 shows the evolution of the valley-induced dephasing probability p_{dia} against the code distance d, both with and without flipping the spin at the turning point of the trajectory. One can see that the error rate is not lowered by such manipulation: I will therefore not execute this operation. By fitting the *no flip* data with a simple linear regression $y = \gamma x$, one can infer a per-dot error probability $p_{dia}/4d = 1.4 \times 10^{-6}$. As a comparison, the per-dot error induced by adiabatic shuttling for $T_2^* = 8\mu$s, *i.e.* in the most optimistic case considered in the main text, is $p_{adia}/4d = 4 \times 10^{-6}$. This justifies the addition of such non-adiabatic effects in my modelling.

B.3 Good matrix for HGP code

In Sections 5.4.1 and 5.4.3, we simulated the performance of a HGP constructed from the product of a [8,1,8] repetition code and a [17,3,8] classical code generated randomly. Its parity check matrix H_8 is

$$H_8 = \begin{pmatrix}
0&1&0&0&0&0&0&0&0&0&0&0&1&0&1&0&0&0\\
1&0&0&0&1&0&0&1&0&0&0&0&1&0&1&0&0&0\\
1&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&1&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&1&
\end{pmatrix}.$$

In Section 5.4.3, we followed the same protocol with a HGP code constructed from the product of a [4,1,4] repetition code and a [7,3,4] classical code generated

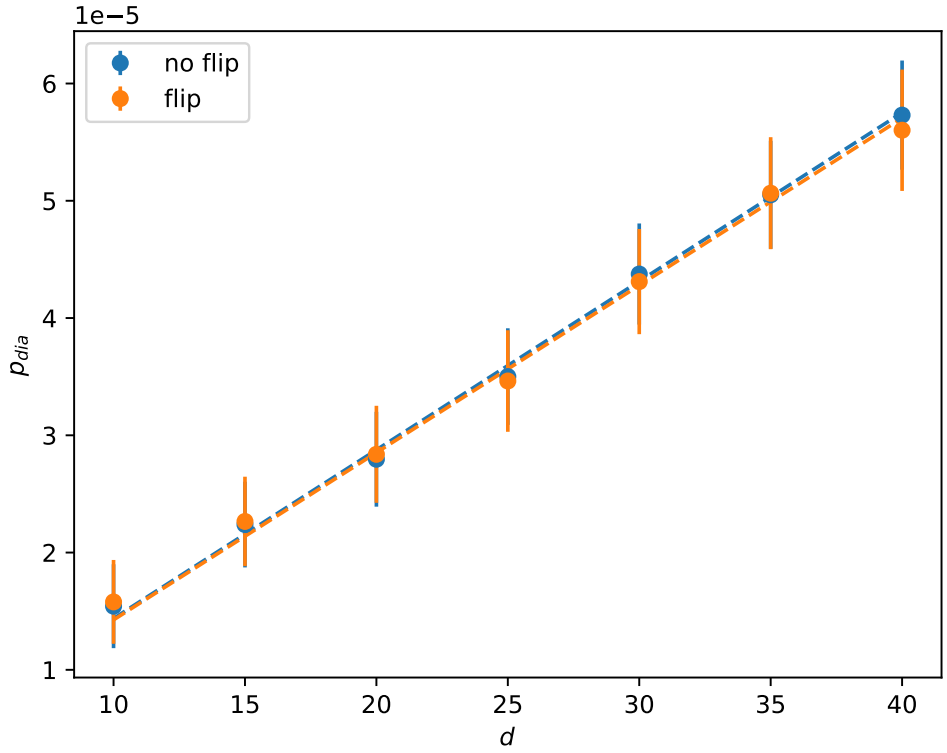


Figure B.3: Solid lines: valley-induced dephasing error p_{dia} against the code distance d, to which the shuttling distance is proportional. Two cases are plotted, depending on if the spin is flipped via an X before the electron is shuttled back. Dashed lines: linear regression of both cases.

randomly. Its parity check matrix H_4 is

$$H_4 = \begin{pmatrix} 1&0&0&1&1&0&1 \\ 0&0&1&0&1&0&1 \\ 0&1&1&0&0&1&0 \\ 1&0&1&0&1&1&0 \end{pmatrix}.$$

C

Appendices for Harnessing qubit transport for global spin qubit control

Contents

C.1 Unitary evolution ignoring Z rotations

C.2 Modelling the g-factor

The g-factor of an electron trapped in a quantum dot in silicon is related to the local spin-orbit fields caused by broken symmetries at the interface where dots form and can be linked to surface roughness, and is thus dependent on the precise position of the electron. As there exists no definitive microscopic description for

g-factor variations [36, 155], Hamza and I decided to model it with a 1D Ornstein-Uhlenbeck (OU) $g_{int}(x)$ process. The restriction to 1D is motivated by the fact that we are concerned by moving electrons along linear shuttling tracks. We chose this process as it allows for the modelling of a continuous random process with a certain correlation length, yet remaining relatively easy to simulate. This model still captures meaningful insights on the impact of the interface on our protocol [238].

The OU process is characterised by its mean g_0, its standard deviation Δg and its correlation length λ. In silicon, g_0 approaches 2 and Ref. [27] estimated that Δg lies between 10^{-3} and 10^{-2}. Both ends of this range are studied in Chapter 6. Here, λ represents the correlation length of the interface roughness which largely dictates the values of the g-factors. We set $\lambda = 20$ nm. To be more precise, $g_{int}(x)$ is given as the solution of the following stochastic differential equation,

$$dg_{int}(x) = \frac{1}{\lambda}(g_0 - g_{int}(x))dx + \sqrt{\frac{2dx}{\lambda}}\Delta g dw(x) \tag{C.1}$$

where w represents a Wiener process.

Eq. (C.1) corresponds to the g-factor at each position x, but does not yet describe the g-factor g_i (appearing in Eq. (6.1)) of an electron at the position x_i. One must indeed additionally take into account the spatial delocalisation of the electron around its central position x_i. To do so, the electron g-factor should rather be defined as the average of $g_{int}(x)$ weighted by the modulus squared of the wavefunction.

Consider a single quantum dot centred at $x = x_i$, the charge wavefunction of an electron trapped in this dot can be modelled as a Gaussian centred on x_i such that,

$$|\psi(x, x_i)|^2 = \frac{1}{\mathcal{N}} \exp\left(-\frac{(x - x_i)^2}{2\lambda_d^2}\right), \tag{C.2}$$

with $\lambda_d = 7$ nm nm which leads to a realistic size of the electron's wavefunction of approximately 40 nm (the electron is in the region $\pm 3\lambda_d$ with a probability of 99%) and $\mathcal{N} = \int_{\mathbb{R}} |\psi(x, x_i)|^2 dx$ is a normalisation constant. For simplicity, we only considered the wavefunction in 1D as we do not expect it to be heavily modified in

the other directions whilst the electron is being shuttled. Note however that our scheme works for any model of g. The averaged g-factor g_i is thus defined by,

$$g_i = g(x_i) \coloneqq \int_{\mathbb{R}} |\psi(x, x_i)|^2 g_{int}(x) dx. \tag{C.3}$$

In Fig. C.1, Hamza plotted one instance of $g_{int}(x)$ and the corresponding evolution of g_i.

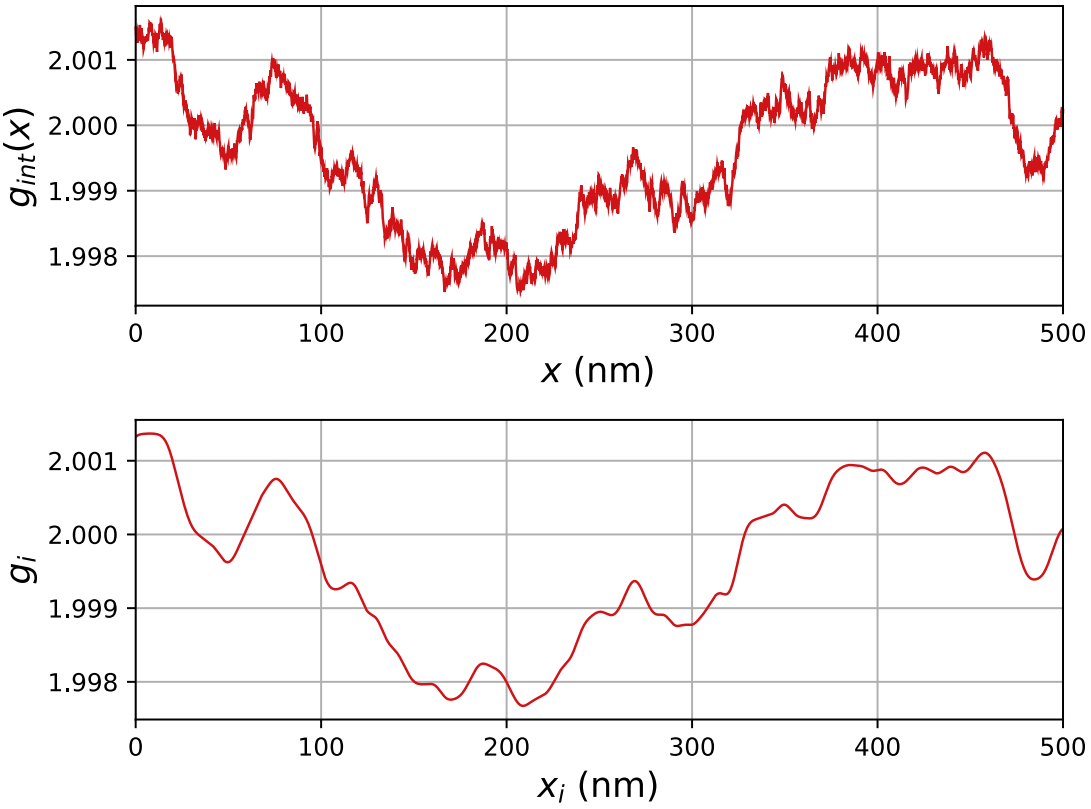


Figure C.1: Evolution of one instance of the random g-factor $g_{int}(x)$ and the corresponding averaged g_i. $g_{int}(x)$ is given by a 1D Ornstein-Uhlenbeck process of mean $g_0 = 2$ and standard deviation $\Delta g = 10^{-3}$. The averaged g-factor is obtained by averaging the position-dependent g-factor $g_{int}(x)$ with respect to the electron's wavefunction $\psi(x, x_i)$ as in Eq. (C.3). This data was produced by Hamza.

Note that in the main text I dropped the subscript i and use x to denote the position of the centre of the electron's wavefunction when the context is clear.

C.3 Impact of using the g-tensor

In Eq. (6.1), I use the following Hamiltonian to describe the impact of a magnetic field on the spin of an electron,

$$H_{mag} = g\vec{S}.\vec{B}, \tag{C.4}$$

where g and $\vec{S}$ represent the g-factor and spin of the electron respectively, and $\vec{B}$ the applied global magnetic field. However, as explained in the main text, the most precise way of describing this interaction requires the use of the full g-tensor $\mathbb{G}$. In this case, the Hamiltonian is now given by,

$$H_{mag} = \vec{S}^T\mathbb{G}\vec{B}. \tag{C.5}$$

Simply put, the effect of $\mathbb{G}$ is to change the axes of the applied magnetic field $\vec{B}$, ultimately leading to the modification of the single-qubit gates axes. As this change is position-dependent, this could then hinder our shuttling-based homogenisation protocol presented in Section 6.3.2.

Considering a driving field along a single axis and the matrix $\mathbb{G}$ associated to Si/SiO_2 heterostructures, I show here that using the g-tensor has minimal impact on the axes of single-qubit rotations. I even show that this impact could be negated, provided perfect knowledge of $\mathbb{G}$. Suppose that $\vec{B} = \vec{B}_{\text{osc}} + \vec{B}_{\text{stat}}$ with $B_{\text{osc}} = (B_1\cos(\omega t + \phi), 0, 0)^T$ and $B_{\text{stat}} = (B_x, B_y, B_0)^T$ its oscillatory and static parts. While the z-component of the Zeeman splitting field is dominant, smaller x- and y-components can arise from our use of micromagnets or superconducting lines. Moreover, I will use a matrix $\mathbb{G}$ obtained from atomistic simulations performed by Cifuentes et al. in [38],

$$\mathbb{G} = \begin{pmatrix} g_0 + \alpha & \beta & g_{13} \\ \beta & g_0 + \alpha & g_{23} \\ 0 & 0 & g_{33} \end{pmatrix}, \tag{C.6}$$

where $g_0 \approx 1.994$, $\alpha \approx -10^{-3}$, $\beta \approx \pm 10^{-2}$, $g_{13} \approx \pm 10^{-3}$, $g_{23} \approx \pm 10^{-3}$ and $g_{33} \approx 2.002$ are typical values of the different parameters for Si/SiO_2. One can

now expand H_{mag} as follows,

$$\begin{aligned} H_{mag} = \frac{1}{2}\Big(& (g_0+\alpha)(B_1\cos(\omega t+\phi)+B_x) \\ & + \beta B_y + g_{13}B_0\Big)\sigma_x \\ & + \frac{1}{2}\Big(\beta(B_1\cos(\omega t+\phi)+B_x) \\ & + (g_0+\alpha)B_y + g_{23}B_0\Big)\sigma_y \\ & + \frac{1}{2}g_{33}B_0\sigma_z, \end{aligned} \tag{C.7}$$

where σ_x, σ_y and σ_z are Pauli operators acting on the spin degree of freedom.

The following transformation is then performed to write H_{mag} in a frame rotating at the drive frequency ω:

$$H_{mag}(t) \rightarrow H'_{mag}(t) = U(t)H_{mag}(t)U^{\dagger}(t) + i\frac{dU}{dt}(t)U^{\dagger}(t), \tag{C.8}$$

with

$$U(t) = \exp\Big(i\frac{\omega}{2}\sigma_z t\Big). \tag{C.9}$$

Using the rotating-wave approximation one gets,

$$\begin{aligned} H'_{mag} = & \frac{1}{4}g_0\tilde{B}_1\cos(\theta+\phi)\sigma_x \\ & + \frac{1}{4}g_0\tilde{B}_1\sin(\theta+\phi)\sigma_y \\ & + \frac{1}{2}(g_{33}B_0-\omega)\sigma_z, \end{aligned} \tag{C.10}$$

where $\cos(\theta) = (1+(\beta/(g_0+\alpha))^2)^{-1/2}$ and $\tilde{B}_1 = \sqrt{(1+\alpha/g_0)^2+(\beta/g_0)^2}B_1$.

More precisely, supposing that $\phi = 0$, instead of obtaining a rotation along the x-axis with Rabi frequency g_0B_1 (as expected from the Rabi model), the use of the g-tensor leads to a rotation along the axis $(\cos(\theta), \sin(\theta), 0)^T$ with a slightly modified frequency $g_0\tilde{B}_1$. Using the values extracted from [38], one finds that $\sin(\theta) \approx O(10^{-3})$ and $|B_1 - \tilde{B}_1|/B_1 \approx O(10^{-4})$, which has a negligible impact on the gate fidelity. Finally, one can approximate H'_{mag} by,

$$H'_{mag} \approx \frac{1}{4}g_0B_1\sigma_x + \frac{1}{2}(g_{33}B_0-\omega)\sigma_z. \tag{C.11}$$

which validates our assumption that the oscillatory and static fields are respectively applied along the x and z axes. For commodity I will additionally assume that $g_0 \approx g_{33}$, which permits the use of a scalar g-factor.

Alternatively, if one has a perfect knowledge of $\mathbb{G}$, one can control the rotation axis and obtain a perfect rotation around the x-axis by setting $\phi = -\theta$. The rotation frequency $\tilde{B}_1$ would also be fully characterised.

C.4 Numerical simulations

To quantify the success of each technique presented in Chapter 6, I measure the fidelity of the implemented operation via the matrix norm:

$$\mathcal{F} = \frac{1}{4}\left|\mathrm{tr}\left(U_{\mathrm{target}}^{\dagger}\tilde{U}\right)\right|^{2/n_q} \tag{C.12}$$

where $U_{\mathrm{target}} = X^{\otimes n_t} \otimes I^{\otimes n_q - n_t}$ is the target unitary and $\tilde{U}$ is a slightly modified version of the time evolution operator U as explained in Eq. (C.15). Note that this expression is an average single-qubit fidelity over the n_q qubits, which is why an n_q-th root is taken. As H is time-dependent, I compute U by discretisation of the time, with time step $dt = T/N_{\mathrm{steps}}$, where $T = \pi/\Omega$ is the total gate time of an X gate with drive strength Ω. N_{steps} is set to $10,000$, such that the simulations converge.

When the qubits are non-interacting and as this time discretisation induces a computational overhead, I simplify the calculations by further decomposing the fidelity $\mathcal{F}$ into a product of single-qubit fidelities. These single-qubit fidelities are then slightly modified, by noting that *known but unwanted* Z-rotations can easily be cancelled out by a change of rotating frame (or equivalently via the implementation of a virtual Z gate) [176]. Therefore, when evaluating the success of a single-qubit gate, one should remove the Z components of the implemented unitary U.

A general representation of any $SU(2)$ gate is:

$$U = \begin{pmatrix} \cos(\theta/2) & -i\mathrm{e}^{i\lambda}\sin(\theta/2) \\ -i\mathrm{e}^{i\phi}\sin(\theta/2) & \mathrm{e}^{i(\lambda+\phi)}\cos(\theta/2) \end{pmatrix} \tag{C.13}$$

$$= Z_\phi X_\theta Z_\lambda \tag{C.14}$$

with $P_\alpha = \mathrm{e}^{-i\alpha P/2}$ and X and Z the Pauli operators. Removing the Z-rotations from U thus amounts to setting $\phi = \lambda = 0$. As $\theta \in [0, \pi[$, this is equivalent to considering the modified time-evolution operator

$$\tilde{U} = \begin{pmatrix} |u_{0,0}| & -i|u_{0,1}| \\ -i|u_{1,0}| & |u_{1,1}| \end{pmatrix} \tag{C.15}$$

with $u_{i,j}$ the coefficients of the matrix U.

The case of two-qubit gates (for the exchange-coupling scheme) is more complex and was not implemented here, which means that I will not attempt in this case to remove unwanted Z rotations.

Finally, note that I will only study unitary evolutions in Chapter 6, thus not simulating noise processes such as charge noise or shuttling noise. If I were to include them in my simulations, the resulting infidelities would only saturate at a level given by the strength of the dominant noise source.

C.5 Residual entanglement in the exchange-based homogenisation

In the exchange-based homogenisation scheme, I noticed that the achieved fidelities were somehow limited. In an attempt to better understand the unitary operation U resulting in the simultaneous driving and swapping, I decided to extend my study to its Schmidt decomposition. Mathematically, any vector in the tensor product of two vector spaces can be expressed as a sum of tensor products. Applied to U, this reads:

$$U = \sum_{i=1}^{m} \alpha_i V_i \otimes W_i \tag{C.16}$$

where for all $i \in [\![1, m]\!]$, V_i and W_i are single-qubit unitary operations on the first and second qubit respectively, and $\alpha_i \in \mathbb{C}$. This decomposition is thus a very convenient way to deduce if a two-qubit unitary is a product of two single-qubit unitaries, in which case only one α_i should be non-zero. In my setup, I verified that the ideal case of instantaneous swapping led to a single non-vanishing α_i with the expected unitaries: two single-qubit rotations around the x-axis at frequency Ω. I then aimed to understand if beyond this ideal case, the resulting unitary U could

be written as the product of two single-qubit rotations, potentially around slightly off axes or with slightly different Rabi frequencies. I however found that reducing the ratios J/Ω or Ω/ω_q^{12} inevitably led, even to first order, to more than one non-vanishing α_i. Finite values of such ratios thus necessarily, but also quite expectedly, increase the entanglement power of the operation. This results in an unwanted mixing of the single-qubit states, that seems to be the first source of infidelity: the only solution against this thus is to operate with higher values of the above ratios. The easiest way to do so is to increase the exchange coupling J. However, it might be experimentally challenging to operate with large exchange couplings.

C.6 Crosstalk suppression via synchronisation

In all the applications I present in Section 6.4, we send global pulses aiming to drive a subset of the qubits, and wish to minimise their impact on all other qubits. This is in general enforced by maximising the frequency spacing between drives and non-targetted qubits (thereby minimising off-resonant effects). Here I show that additionally tuning the drive strength Ω can help further mitigate these crosstalk effects. Namely, a clever choice of Ω can allow for the synchronisation of the oscillations of the electrons, such that the non-target qubits perform a 2π rotation (identity gate) while the target qubits perform a π rotation (X gate).

Let us first focus on the case where only one target frequency and one non-target frequency are present in the frequency spectrum. This essentially describes all the shuttling-based applications from Section 6.4 (although in general there subsists a small dispersion σ around these main frequencies). In this case, one can equate the times taken by one X gate on the target qubits and a $2p\pi$ ($p \in \mathbb{N}$) rotation of the non-target qubits by enforcing the condition,

$$\frac{\pi}{\Omega} = \frac{2p\pi}{\sqrt{\Omega^2 + (\omega_q^{12})^2}}, \tag{C.17}$$

with ω_q^{12} the difference between the effective target and non-target frequencies. The target qubits theoretically oscillate at a frequency Ω, while off-resonant effects

induce oscillations of the non-target qubits at frequency $\sqrt{\Omega^2 + (\omega_q^{12})^2}$ as inferred from the Rabi model. This results in setting:

$$\Omega = \frac{\omega_q^{12}}{\sqrt{4p^2 - 1}}, \quad p \in \mathbb{N} \tag{C.18}$$

By reducing the spectrum to only one target and one non-target frequencies, setting the above condition *totally* removes any crosstalk infidelity as non-target qubits exactly perform an identity gate while target qubits perform an X rotation.

When using the binning scheme, it is in general not possible to reduce the frequency spectrum to only two frequencies. In this case, one can generalise the above condition as follows (see Section II of [195] with a bin width $2\delta\omega_q$):

$$\Omega = \frac{\delta\omega_q}{p}, \quad p \in \mathbb{N} \tag{C.19}$$

where p is a free parameter that can be adjusted according to the achievable values of Ω and $\delta\omega_q$. This is a generalisation of Eq. (C.18), where the existence of only two frequencies in the frequency spectrum permitted the complete removal of crosstalk effects. Here the extension to more than two frequencies only allows for a partial reduction of the crosstalk.

Note that in both cases $\Omega \leq \omega_q^{12}$, meaning that if the frequencies are close to each other, slower gates will be obtained.

C.7 Estimation of the right parameter regime

C.7.1 Shuttling-based scheme

In Section 6.4.1, I stated the values of experimental parameters which *a posteriori* led to a high performance of our shuttling-based protocol. In this section, I further justify this choice with some analytical arguments.

Firstly, our shuttling-based homogenisation performs best at low frequency dispersion, justifying to work in the regime $\Delta g = 10^{-3} g_0$ and $B_0 = 0.1$ T. Second, the dispersion of target frequencies around the driving frequency (due to finite σ) means that target qubits are not exactly driven at resonance. This undesired effect

is minimised at large Ω, which is also synonym of faster gates. This is also observed in the second panel of Fig. 6.6. For this reason, I set $\Omega = 5$ MHz.

Let me now evaluate the target values of the last key experimental parameters enabling the full power of our protocol *i.e.* the required g-factor shift G between targets and non-targets, the shuttling length d and the shuttling speed v. When qubits are shuttled back and forth in a line, d is the one way distance; when qubits are shuttled in a loop, d is the loop length. I will first focus on the case of a uniformly applied frequency shift GB_0 between the targets and non-targets. In this scenario $\sigma > 0$. I will estimate the above parameters by distinguishing two main contributions to the infidelity: the slightly off-resonant driving of the homogenised target qubits and the crosstalk with the non-target qubits. I wish to bring each of these contributions below 2×10^{-3} for at least 95% of the qubits, so as to be comfortably below the surface code threshold. If the locations of the 5% outliers are known, the error they generate can be mitigated by suitable QEC protocols as erasure errors [239].

The first contribution reads:

$$I_1 = \frac{1}{1 + \left(\frac{\Omega}{2\sigma B_0}\right)^2} \tag{C.20}$$

from the Rabi model, where I use $2\sigma B_0$ for the frequency mismatch between the drive and a given target qubit so as to cover 95% of the qubits. σ is the standard deviation of the homogenised g-factors

$$\bar{g} = \frac{1}{d}\int_0^d g(x)\mathrm{d}x. \tag{C.21}$$

In a first approximation, one can write

$$\sigma = \frac{\Delta g}{\sqrt{d/\lambda}} \tag{C.22}$$

where λ is the coherence length of the g-factor landscape. This equation simply states that the standard deviation of the average of independent random variables decreases as $1/\sqrt{N}$, where N is the number of samples. This leads to:

$$I_1 = \frac{1}{1 + \frac{\Omega^2}{4\Delta\omega_q^2}\frac{d}{\lambda}} \tag{C.23}$$

with $\Delta\omega_q = \Delta g B_0$. By setting $I_1 = 2 \times 10^{-3}$ I find a minimum shuttling distance $d \geq 14$ μm. From this I can deduce the minimum shuttling speed enabling the exploration of such a distance during the gate time $T = \pi/\Omega$:

$$v = \frac{\Omega d}{\pi} \tag{C.24}$$

It follows that $v \geq 23$ m/s. I can now evaluate the infidelity arising from the crosstalk with non-target qubits:

$$I_2 = \frac{1}{1 + \left(\frac{GB_0}{\Omega}\right)^2} \tag{C.25}$$

I here neglect the dispersion σ of the non-target qubits around their mean g-factor as it is negligible compared to G. I here find $G \geq 160$ MHz. Note that for simplicity, I do not describe here the crosstalk reductions offered by the technique of Appendix C.6. This would permit for lower values of G.

In the case where the magnetic field gradient can be tuned so as to annihilate the frequency dispersion σ, the infidelity is only limited by the performance of our homogenisation protocol (which I neglected in the previous case). Indeed, I_1 vanishes as $\sigma = 0$ (all targets are resonantly driven) and I_2 as well as Eq. (C.18) guarantees a total suppression of the crosstalk in the strict presence of two frequencies in the spectrum. Therefore, G can be chosen arbitrarily, as long as $GB_0 \geq 5$ MHz (remember that Eq. (C.18) enforces $\Omega \leq GB_0$). As for v and d, their values can be inferred from the top panel of Fig. 6.6, which describes a driving at exact resonance with the homogenised target frequency. At $\Omega = 5$ MHz, the resulting infidelity (from imperfect homogenisation) never exceeds 0.4%, meaning no value of d and no value of $v \geq 10$ m/s would be an impediment to surface-code-enabled error correction.

Note however that these models are highly simplified for the ease of parameter estimation, using worst-case approximations. Consequently, although the first parameter regime I considered in Section 6.4 (left panel of Fig. 6.11) does not rigorously respect the conditions I here set, it still yields desirable performances.

C.7.2 Binning scheme

I proceed similarly for the binning scheme. Its performance is maximised at large interbin spacing, which is given by $2\delta\omega_q = 2\Delta g B_0/10$. It is thus optimal to work at higher interface roughness and higher magnetic field in this case: $\Delta g = 10^{-2} g_0$ and $B_0 = 1$ T. Another reason for this choice is that enforcing Eq. (C.19) for crosstalk reduction (which is crucial here) implies $\Omega \leq \delta\omega_q$. $\delta\omega_q$ must therefore be kept relatively high to ensure that gates are fast. With the above choice of parameters, I obtain $\delta\omega_q = 30$ MHz, which would not be a bottleneck. Besides, the previous analysis leading to $G \geq 160$ MHz is still valid here.

C.7.3 Hybrid shuttling-binning scheme

I briefly mention here that one further protocol I envisioned was to combine the shuttling and binning schemes. Once qubits are shuttled, rather than targetting them at a mean slightly off-resonant frequency $\omega = GB_0$, one could envisage to place the post-shuttling effective frequencies into bins and resonantly target them with multiple driving tones. While this two-step frequency factorisation protocol sounds powerful on paper, it would not lead to desirable fidelities as the shuttling and binning schemes are not operated in the same parameter regime (they respectively work best at low and high $\Delta\omega_q$).

D

Appendices for Algorithmic error mitigation for quantum eigenvalues estimation

Contents

D.1 Proof of Theorem 2

Proof. The case ($p = 1$) is quite straightforward. First note that for any $\vec{\lambda}$ verifying Eq. (7.29), $||\vec{\lambda}||_1 \geq \sum_k \lambda_k \geq 1$. Besides, if 0 is in the convex hull of all implementable $\vec{\delta_k}$'s, then:

$$\exists(\lambda_1, ..., \lambda_m) \in \mathbb{R}_+^m \quad \sum_k \lambda_k \delta_k = 0 \tag{D.1}$$

with:

$$\sum_k \lambda_k = 1$$

These two equations are what Eqs. (7.28) and (7.29) reduce to in the case $p = 1$. This choice of $\vec{\lambda}$ achieves the lower bound:

$$||\vec{\lambda}||_1 = \sum_k |\lambda_k| = \sum_k \lambda_k = 1 \tag{D.2}$$

□

D.2 Proof of Theorem 3

Proof. This requires more work. For $N = 1$, Eq. (7.28) becomes:

$$\forall i \in [\![1, p]\!] \quad \sum_{k=1}^{m} \lambda_k \delta_k^i = 0$$

Thus, one can note that both Eqs. (7.28) and (7.29) can be combined using the rectangular Vandermonde matrix $V = (\delta_k^i)_{i,k}$ and the vector $b = (1, 0, ..., 0)^T$ with $m - 1$ zeros:

$$V\vec{\lambda} = b \tag{D.3}$$

The first equation of this linear system is Eq. (7.29) and the others are Eq. (7.28). A solution $\vec{\lambda}^*$ of minimum ℓ^2-norm of this system satisfies:

$$||\vec{\lambda}^*||_2 = ||V^+ b||_2 \tag{D.4}$$

with V^+ the Moore–Penrose inverse of V. In particular:

$$||\vec{\lambda}^*||_2 \leq ||V^+||_2 ||b||_2 = ||V^+||_2 \tag{D.5}$$

Using the singular value decomposition $V = U\Sigma V^\dagger$, we obtain:

$$||V^+||_2 = ||\Sigma^+||_2 = \max\{1/\mu, \mu \in \mathcal{S} \setminus \{0\}\} \tag{D.6}$$

where $\mathcal{S}$ denotes the singular values of V. Besides, it is well-known that the square Vandermonde matrix $\tilde{V}(\delta_1, ..., \delta_p)$ is invertible if and only if for all $i \neq j$, $\delta_i \neq \delta_j$. As a result, the rectangular Vandermonde matrix V is full-rank, hence has non-zero singular values only, if and only if for all $i \neq j$, $\delta_i \neq \delta_j$. By continuity of the singular values over $(\delta_1, ..., \delta_m)$ on the compact space Δ, it follows that $||V^+||_2$ is bounded, hence $||\vec{\lambda}^*||_2$ too. □

References

[1] R. P. Feynman, *The character of physical law* (1964) Chap. 2.

[2] R. P. Feynman, "Simulating physics with computers", Int J Theor Phys (1982).

[3] D. Deutsch, "Quantum theory, the church-turing principle and the universal quantum computer", Royal Society (London), Proceedings, Series A - Mathematical and Physical Sciences, 10.1098/rspa.1985.0070 (1985).

[4] P. W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer", arXiv:quant-ph/9508027, 10.48550/arXiv.quant-ph/9508027 (1996).

[5] L. K. Grover, "A fast quantum mechanical algorithm for database search", arXiv:quant-ph/9605043, 10.48550/arXiv.quant-ph/9605043 (1996).

[6] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell, B. Burkett, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, A. Dunsworth, E. Farhi, B. Foxen, A. Fowler, C. Gidney, M. Giustina, R. Graff, K. Guerin, S. Habegger, M. P. Harrigan, M. J. Hartmann, A. Ho, M. Hoffmann, T. Huang, T. S. Humble, S. V. Isakov, E. Jeffrey, Z. Jiang, D. Kafri, K. Kechedzhi, J. Kelly, P. V. Klimov, S. Knysh, A. Korotkov, F. Kostritsa, D. Landhuis, M. Lindmark, E. Lucero, D. Lyakh, S. Mandrà, J. R. McClean, M. McEwen, A. Megrant, X. Mi, K. Michielsen, M. Mohseni, J. Mutus, O. Naaman, M. Neeley, C. Neill, M. Y. Niu, E. Ostby, A. Petukhov, J. C. Platt, C. Quintana, E. G. Rieffel, P. Roushan, N. C. Rubin, D. Sank, K. J. Satzinger, V. Smelyanskiy, K. J. Sung, M. D. Trevithick, A. Vainsencher, B. Villalonga, T. White, Z. J. Yao, P. Yeh, A. Zalcman, H. Neven, and J. M. Martinis, "Quantum supremacy using a programmable superconducting processor", en, Nature **574**, 505 (2019).

[7] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, "Surface codes: towards practical large-scale quantum computation", Physical Review A **86**, 10.1103/physreva.86.032324 (2012).

[8] C. Gidney and M. Ekerå, "How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits", Quantum **5**, 433 (2021).

[9] Z. Chen, K. J. Satzinger, J. Atalaya, A. N. Korotkov, A. Dunsworth, D. Sank, C. Quintana, M. McEwen, R. Barends, P. V. Klimov, S. Hong, C. Jones, A. Petukhov, D. Kafri, S. Demura, B. Burkett, C. Gidney, A. G. Fowler, A. Paler, H. Putterman, I. Aleiner, F. Arute, K. Arya, R. Babbush, J. C. Bardin, A. Bengtsson, A. Bourassa, M. Broughton, B. B. Buckley, D. A. Buell, N. Bushnell, B. Chiaro, R. Collins, W. Courtney, A. R. Derk, D. Eppens, C. Erickson, E. Farhi, B. Foxen, M. Giustina, A. Greene, J. A. Gross, M. P. Harrigan, S. D. Harrington, J. Hilton, A. Ho, T. Huang, W. J. Huggins,

L. B. Ioffe, S. V. Isakov, E. Jeffrey, Z. Jiang, K. Kechedzhi, S. Kim, A. Kitaev, F. Kostritsa, D. Landhuis, P. Laptev, E. Lucero, O. Martin, J. R. McClean, T. McCourt, X. Mi, K. C. Miao, M. Mohseni, S. Montazeri, W. Mruczkiewicz, J. Mutus, O. Naaman, M. Neeley, C. Neill, M. Newman, M. Y. Niu, T. E. O'Brien, A. Opremcak, E. Ostby, B. Pató, N. Redd, P. Roushan, N. C. Rubin, V. Shvarts, D. Strain, M. Szalay, M. D. Trevithick, B. Villalonga, T. White, Z. J. Yao, P. Yeh, J. Yoo, A. Zalcman, H. Neven, S. Boixo, V. Smelyanskiy, Y. Chen, A. Megrant, J. Kelly, and Google Quantum AI, "Exponential suppression of bit or phase errors with cyclic error correction", en, Nature **595**, 383 (2021).

[10] C. Monroe, "High-fidelity quantum logic gates using trapped-ion hyperfine qubits", Physical Review Letters **117**, 10.1103/physrevlett.117.060504 (2016).

[11] W. Huang, C. H. Yang, K. W. Chan, T. Tanttu, B. Hensen, R. C. C. Leon, M. A. Fogarty, J. C. C. Hwang, F. E. Hudson, K. M. Itoh, A. Morello, A. Laucht, and A. S. Dzurak, "Fidelity benchmarks for two-qubit gates in silicon", Nature **569**, 532 (2019).

[12] M. A. Rol, F. Battistel, F. K. Malinowski, C. C. Bultink, B. M. Tarasinski, R. Vollmer, N. Haider, N. Muthusubramanian, A. Bruno, B. M. Terhal, and L. DiCarlo, "Fast, High-Fidelity Conditional-Phase Gate Exploiting Leakage Interference in Weakly Anharmonic Superconducting Qubits", en, Physical Review Letters **123**, 120502 (2019).

[13] P. Jurcevic, A. Javadi-Abhari, L. S. Bishop, I. Lauer, D. F. Bogorin, M. Brink, L. Capelluto, O. Günlük, T. Itoko, N. Kanazawa, A. Kandala, G. A. Keefe, K. Krsulich, W. Landers, E. P. Lewandowski, D. T. McClure, G. Nannicini, A. Narasgond, H. M. Nayfeh, E. Pritchett, M. B. Rothwell, S. Srinivasan, N. Sundaresan, C. Wang, K. X. Wei, C. J. Wood, J.-B. Yau, E. J. Zhang, O. E. Dial, J. M. Chow, and J. M. Gambetta, "Demonstration of quantum volume 64 on a superconducting quantum computing system", arXiv:2008.08571, 10.48550/arXiv.2008.08571 (2020).

[14] B. Foxen, C. Neill, A. Dunsworth, P. Roushan, B. Chiaro, A. Megrant, J. Kelly, Z. Chen, K. Satzinger, R. Barends, F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, S. Boixo, D. Buell, B. Burkett, Y. Chen, R. Collins, E. Farhi, A. Fowler, C. Gidney, M. Giustina, R. Graff, M. Harrigan, T. Huang, S. V. Isakov, E. Jeffrey, Z. Jiang, D. Kafri, K. Kechedzhi, P. Klimov, A. Korotkov, F. Kostritsa, D. Landhuis, E. Lucero, J. McClean, M. McEwen, X. Mi, M. Mohseni, J. Y. Mutus, O. Naaman, M. Neeley, M. Niu, A. Petukhov, C. Quintana, N. Rubin, D. Sank, V. Smelyanskiy, A. Vainsencher, T. C. White, Z. Yao, P. Yeh, A. Zalcman, H. Neven, J. M. Martinis, and Google AI Quantum, "Demonstrating a Continuous Set of Two-qubit Gates for Near-term Quantum Algorithms", en, Physical Review Letters **125**, 120504 (2020).

[15] D. Gottesman, *The heisenberg representation of quantum computers*, 1998.

[16] S. Bravyi and A. Kitaev, "Universal quantum computation with ideal clifford gates and noisy ancillas", Physical Review A **71**, 10.1103/physreva.71.022316 (2005).

[17] C. M. Dawson and M. A. Nielsen, "The Solovay-Kitaev algorithm", arXiv:quant-ph/0505030, 10.48550/arXiv.quant-ph/0505030 (2005).

[18] D. P. DiVincenzo, "The physical implementation of quantum computation", Fortschritte der Physik **48**, 10.1002/1521-3978(200009)48:9/11<771::aid-prop771>3.0.co;2-e (2000).

[19] M. F. Gonzalez-Zalba, S. de Franceschi, E. Charbon, T. Meunier, M. Vinets, and A. S. Dzurak, "Scaling silicon-based quantum computing using cmos technology", Nature electronics **4**, 872 (2021).

[20] G. Burkard, T. D. Ladd, A. Pan, J. M. Nichol, and J. R. Petta, "Semiconductor spin qubits", Rev. Mod. Phys. **95**, 025003 (2023).

[21] J. Levy, "Universal quantum computation with spin-1/2 pairs and heisenberg exchange", Phys. Rev. Lett. **89**, 147902 (2002).

[22] 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 (2005).

[23] D. P. DiVincenzo, D. Bacon, J. Kempe, G. Burkard, and K. B. Whaley, "Universal quantum computation with the exchange interaction", Nature **408**, 10.1038/35042541 (2000).

[24] T. D. Ladd and C. Jones, "Gauge-invariant Adiabatic Two-Qubit Gates for Exchange-Only Qubits", arXiv:1712.05849, 10.48550/arXiv.1712.05849 (2017).

[25] G. Feher, "Electron spin resonance experiments on donors in silicon. i. electronic structure of donors by the electron nuclear double resonance technique", Phys. Rev. **114**, 1219 (1959).

[26] P. W. Deelman, L. F. Edge, and C. A. Jackson, "Metamorphic materials for quantum computing", eng, MRS bulletin **41**, 224 (2016).

[27] 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", Phys. Rev. B **97**, 241401 (2018).

[28] S. M. Patomäki, M. F. Gonzalez-Zalba, M. A. Fogarty, Z. Cai, S. C. Benjamin, and J. J. L. Morton, "Pipeline quantum processor architecture for silicon spin qubits", en, npj Quantum Information **10**, 31 (2024).

[29] T. Meunier, V. E. Calado, and L. M. K. Vandersypen, "Efficient controlled-phase gate for single-spin qubits in quantum dots", Phys. Rev. B **83**, 121403 (2011).

[30] A. Noiri, K. Takeda, T. Nakajima, T. Kobayashi, A. Sammak, G. Scappucci, and S. Tarucha, "Fast universal quantum gate above the fault-tolerance threshold in silicon", Nature **601**, 10.1038/s41586-021-04182-y (2022).

[31] R. Kalra, A. Laucht, C. D. Hill, and A. Morello, "Robust Two-Qubit Gates for Donors in Silicon Controlled by Hyperfine Interactions", Physical Review X **4**, 021044 (2014).

[32] M. De Smet, Y. Matsumoto, A.-M. J. Zwerver, L. Tryputen, S. L. de Snoo, S. V. Amitonov, S. R. Katiraee-Far, A. Sammak, N. Samkharadze, Ö. Gül, R. N. M. Wasserman, E. Greplová, M. Rimbach-Russ, G. Scappucci, and L. M. K. Vandersypen, "High-fidelity single-spin shuttling in silicon", en, Nature Nanotechnology **20**, 866 (2025).

[33] D. M. Zajac, A. J. Sigillito, M. Russ, F. Borjans, J. M. Taylor, G. Burkard, and J. R. Petta, "Resonantly driven cnot gate for electron spins", Science **359**, 439 (2018).

[34] J. Yoneda, W. Huang, M. Feng, C. H. Yang, K. W. Chan, T. Tanttu, W. Gilbert, R. C. C. Leon, F. E. Hudson, K. M. Itoh, A. Morello, S. D. Bartlett, A. Laucht, A. Saraiva, and A. S. Dzurak, "Coherent spin qubit transport in silicon", Nature Communications **12**, 10.1038/s41467-021-24371-7 (2021).

[35] I. Seidler, T. Struck, R. Xue, N. Focke, S. Trellenkamp, H. Bluhm, and L. R. Schreiber, "Conveyor-mode single-electron shuttling in si/sige for a scalable quantum computing architecture", npj Quantum Information **8**, 10.1038/s41534-022-00615-2 (2022).

[36] V. Langrock, J. A. Krzywda, N. Focke, I. Seidler, L. R. Schreiber, and Ł. Cywiński, "Blueprint of a scalable spin qubit shuttle device for coherent mid-range qubit transfer in disordered si/sige/sio 2", PRX Quantum **4**, 10.1103/prxquantum.4.020305 (2023).

[37] M. D. Smet, Y. Matsumoto, A.-M. J. Zwerver, L. Tryputen, S. L. de Snoo, S. V. Amitonov, A. Sammak, N. Samkharadze, Ö. Gül, R. N. M. Wasserman, M. Rimbach-Russ, G. Scappucci, and L. M. K. Vandersypen, "High-fidelity single-spin shuttling in silicon", (2024).

[38] J. D. Cifuentes, T. Tanttu, W. Gilbert, J. Y. Huang, E. Vahapoglu, R. C. C. Leon, S. Serrano, D. Otter, D. Dunmore, P. Y. Mai, F. Schlattner, M. Feng, K. Itoh, N. Abrosimov, H.-J. Pohl, M. Thewalt, A. Laucht, C. H. Yang, C. C. Escott, W. H. Lim, F. E. Hudson, R. Rahman, A. S. Dzurak, and A. Saraiva, "Bounds to electron spin qubit variability for scalable cmos architectures", Nature Communications **15**, 10.1038/s41467-024-48557-x (2024).

[39] D. Culcer, X. Hu, and S. Das Sarma, "Interface roughness, valley-orbit coupling, and valley manipulation in quantum dots", Phys. Rev. B **82**, 205315 (2010).

[40] M. Friesen, S. Chutia, C. Tahan, and S. N. Coppersmith, "Valley splitting theory of SiGe/Si/SiGe quantum wells", Phys. Rev. B **75**, 115318 (2007).

[41] M. G. Borselli, R. S. Ross, A. A. Kiselev, E. T. Croke, K. S. Holabird, P. W. Deelman, L. D. Warren, I. Alvarado-Rodriguez, I. Milosavljevic, F. C. Ku, W. S. Wong, A. E. Schmitz, M. Sokolich, M. F. Gyure, and A. T. Hunter, "Measurement of valley splitting in high-symmetry si/sige quantum dots", Applied Physics Letters **98**, 10.1063/1.3569717 (2011).

[42] X. Xue, M. Russ, N. Samkharadze, B. Undseth, A. Sammak, G. Scappucci, and L. M. K. Vandersypen, "Quantum logic with spin qubits crossing the surface code threshold", Nature **601**, 10.1038/s41586-021-04273-w (2022).

[43] A. R. Mills, C. R. Guinn, M. J. Gullans, A. J. Sigillito, M. M. Feldman, E. Nielsen, and J. R. Petta, "Two-qubit silicon quantum processor with operation fidelity exceeding 99%", Science Advances **8**, 10.1126/sciadv.abn5130 (2022).

[44] J. Yoneda, K. Takeda, T. Otsuka, T. Nakajima, M. R. Delbecq, G. Allison, T. Honda, T. Kodera, S. Oda, Y. Hoshi, N. Usami, K. M. Itoh, and S. Tarucha, "A quantum-dot spin qubit with coherence limited by charge noise and fidelity higher than 99.9%", Nature Nanotechnology **13**, 10.1038/s41565-017-0014-x (2017).

[45] 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**, 10.1038/npjqi.2015.19 (2016).

[46] K. Takeda, A. Noiri, T. Nakajima, T. Kobayashi, and S. Tarucha, "Quantum error correction with silicon spin qubits", Nature **608**, 10.1038/s41586-022-04986-6 (2022).

[47] S. G. J. Philips, M. T. Madzik, S. V. Amitonov, S. L. de Snoo, M. Russ, N. Kalhor, C. Volk, W. I. L. Lawrie, D. Brousse, L. Tryputen, B. P. Wuetz, A. Sammak, M. Veldhorst, G. Scappucci, and L. M. K. Vandersypen, "Universal control of a six-qubit quantum processor in silicon", Nature **609**, 10.1038/s41586-022-05117-x (2022).

[48] R. Maurand, X. Jehl, D. Kotekar-Patil, A. Corna, H. Bohuslavskyi, R. Laviéville, L. Hutin, S. Barraud, M. Vinet, M. Sanquer, and S. De Franceschi, "A cmos silicon spin qubit", Nature Communications **7**, 10.1038/ncomms13575 (2016).

[49] A. M. J. Zwerver, T. Krähenmann, T. F. Watson, L. Lampert, H. C. George, R. Pillarisetty, S. A. Bojarski, P. Amin, S. V. Amitonov, J. M. Boter, R. Caudillo, D. Correas-Serrano, J. P. Dehollain, G. Droulers, E. M. Henry, R. Kotlyar, M. Lodari, F. Lüthi, D. J. Michalak, B. K. Mueller, S. Neyens, J. Roberts, N. Samkharadze, G. Zheng, O. K. Zietz, G. Scappucci, M. Veldhorst, L. M. K. Vandersypen, and J. S. Clarke, "Qubits made by advanced semiconductor manufacturing", Nature Electronics **5**, 10.1038/s41928-022-00727-9 (2022).

[50] X. Xue, B. Patra, J. P. G. van Dijk, N. Samkharadze, S. Subramanian, A. Corna, B. Paquelet Wuetz, C. Jeon, F. Sheikh, E. Juarez-Hernandez, B. P. Esparza, H. Rampurawala, B. Carlton, S. Ravikumar, C. Nieva, S. Kim, H.-J. Lee, A. Sammak, G. Scappucci, M. Veldhorst, F. Sebastiano, M. Babaie, S. Pellerano, E. Charbon, and L. M. K. Vandersypen, "CMOS-based cryogenic control of silicon quantum circuits", en, Nature **593**, 205 (2021).

[51] A. Ruffino, T.-Y. Yang, J. Michniewicz, Y. Peng, E. Charbon, and M. F. Gonzalez-Zalba, "A cryo-cmos chip that integrates silicon quantum dots and multiplexed dispersive readout electronics", Nature electronics **5**, 53 (2022).

[52] 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**, 10.1038/s41467-017-01905-6 (2017).

[53] J. M. Boter, J. P. Dehollain, J. P. van Dijk, Y. Xu, T. Hensgens, R. Versluis, H. W. Naus, J. S. Clarke, M. Veldhorst, F. Sebastiano, and L. M. Vandersypen, "Spiderweb array: a sparse spin-qubit array", Physical Review Applied **18**, 10.1103/physrevapplied.18.024053 (2022).

[54] F. J. MacWilliams and N. J. A. Sloane, *The theory of error-correctiong codes.* (1977).

[55] D. Dieks, "Communication by epr devices", Physics Letters A **92** (1982).

[56] J. L. Park, "The concept of transition in quantum mechanics", Found Phys **1** (1970).

[57] W. K. Wootters and W. H. Zurek, "A single quantum cannot be cloned", en, Nature **299**, 802 (1982).

[58] S. Bravyi, M. Englbrecht, R. König, and N. Peard, "Correcting coherent errors with surface codes", npj Quantum Information **4**, 10.1038/s41534-018-0106-y (2018).

[59] S. J. Beale, J. J. Wallman, M. Gutiérrez, K. R. Brown, and R. Laflamme, "Quantum error correction decoheres noise", Phys. Rev. Lett. **121**, 190501 (2018).

[60] J. J. Wallman and J. Emerson, "Noise tailoring for scalable quantum computation via randomized compiling", Phys. Rev. A **94**, 052325 (2016).

[61] P. W. Shor, "Scheme for reducing decoherence in quantum computer memory", **125**, 10.1103/physrevlett.125.120504 (1995).

[62] A. M. Steane, "Error correcting codes in quantum theory", Phys. Rev. Lett. **77**, 793 (1996).

[63] A. R. Calderbank and P. W. Shor, "Good quantum error-correcting codes exist", Physical Review A **54**, 1098 (1996).

[64] A. R. Calderbank, E. M. Rains, P. W. Shor, and N. J. A. Sloane, "Quantum error correction and orthogonal geometry", Physical Review Letters **78**, 405 (1997).

[65] D. Gottesman, "Class of quantum error-correcting codes saturating the quantum hamming bound", Physical Review A **54**, 1862 (1996).

[66] D. Gottesman, *Stabilizer Codes and Quantum Error Correction*, arXiv:quant-ph/9705052, May 1997.

[67] "Multiple-particle interference and quantum error correction", Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences **452**, 2551 (1996).

[68] D. Poulin, "Stabilizer formalism for operator quantum error correction", Physical Review Letters **95**, 230504 (2005).

[69] D. Bacon, "Operator quantum error-correcting subsystems for self-correcting quantum memories", Physical Review A **73**, 012340 (2006).

[70] S. Bravyi, A. Cross, and B. Terhal, "Subsystem surface codes with three-qubit check operators", Quantum Information Computation **13**, 963 (2013).

[71] H. Bombin, "Subsystem color codes", Physical Review A **81**, 032301 (2010).

[72] D. Aharonov and M. Ben-Or, "Fault-Tolerant Quantum Computation With Constant Error Rate", arXiv:quant-ph/9906129, 10.48550/arXiv.quant-ph/9906129 (1999).

[73] P. Aliferis, D. Gottesman, and J. Preskill, "Quantum accuracy threshold for concatenated distance-3 codes", 10.48550/ARXIV.QUANT-PH/0504218 (2005).

[74] A. Y. Kitaev, "Quantum computations: algorithms and error correction", Russian Mathematical Surveys **52**, 1191 (1997).

[75] E. Knill, R. Laflamme, and W. H. Zurek, "Resilient quantum computation: error models and thresholds", Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences **454**, 365 (1998).

[76] S. B. Bravyi and A. Y. Kitaev, *Quantum codes on a lattice with boundary*, 1998.

[77] M. H. Freedman and D. A. Meyer, *Projective plane and planar quantum codes*, 1998.

[78] A. Kitaev, "Fault-tolerant quantum computation by anyons", Annals of Physics **303**, 2 (2003).

[79] J. Preskill, *Fault-tolerant quantum computation*, 1997.

[80] L. Riesebos, X. Fu, S. Varsamopoulos, C. Almudever, and K. Bertels, "Pauli frames for quantum computer architectures", in 2017 54th acm/edac/ieee design automation conference (dac) (2017), pp. 1–6.

[81] B. Eastin and E. Knill, "Restrictions on transversal encoded quantum gate sets", Physical Review Letters **102**, 10.1103/physrevlett.102.110502 (2009).

[82] B. Zeng, A. Cross, and I. L. Chuang, "Transversality Versus Universality for Additive Quantum Codes", IEEE Transactions on Information Theory **57**, 6272 (2011).

[83] S. Bravyi and R. König, "Classification of topologically protected gates for local stabilizer codes", Physical Review Letters **110**, 10.1103/physrevlett.110.170503 (2013).

[84] A. Paetznick and B. W. Reichardt, "Universal fault-tolerant quantum computation with only transversal gates and error correction", Physical Review Letters **111**, 10.1103/physrevlett.111.090505 (2013).

[85] E. Knill, "Quantum computing with realistically noisy devices", Nature **434**, 39 (2005).

[86] B. W. Reichardt, "Quantum universality by state distillation", 10.48550/ARXIV.QUANT-PH/0608085 (2006).

[87] C. Gidney, N. Shutty, and C. Jones, "Magic state cultivation: growing T states as cheap as CNOT gates", arXiv:2409.17595, 10.48550/arXiv.2409.17595 (2024).

[88] C. Gidney, *How to factor 2048 bit rsa integers with less than a million noisy qubits*, 2025.

[89] C. Horsman, A. G. Fowler, S. Devitt, and R. V. Meter, "Surface code quantum computing by lattice surgery", New Journal of Physics **14**, 123011 (2012).

[90] D. Litinski, "A game of surface codes: large-scale quantum computing with lattice surgery", Quantum **3**, 128 (2019).

[91] S. Aaronson and D. Gottesman, "Improved simulation of stabilizer circuits", Phys. Rev. A **70**, 052328 (2004).

[92] S. Bravyi, D. Poulin, and B. Terhal, "Tradeoffs for reliable quantum information storage in 2d systems", Physical Review Letters **104**, 10.1103/physrevlett.104.050503 (2010).

[93] D. Bluvstein, S. J. Evered, A. A. Geim, S. H. Li, H. Zhou, T. Manovitz, S. Ebadi, M. Cain, M. Kalinowski, D. Hangleiter, J. P. B. Ataides, N. Maskara, I. Cong, X. Gao, P. S. Rodriguez, T. Karolyshyn, G. Semeghini, M. J. Gullans, M. Greiner, V. Vuletić, and M. D. Lukin, "Logical quantum processor based on reconfigurable atom arrays", Nature, 10.1038/s41586-023-06927-3 (2023).

[94] A. A. Kovalev and L. P. Pryadko, "Improved quantum hypergraph-product ldpc codes", in 2012 ieee international symposium on information theory proceedings (July 2012).

[95] P. Panteleev and G. Kalachev, "Degenerate quantum LDPC codes with good finite length performance", Quantum **5**, 585 (2021).

[96] H. Bombin and M. A. Martin-Delgado, "Topological quantum distillation", Phys. Rev. Lett. **97**, 180501 (2006).

[97] H. Bombin, "Gauge color codes: optimal transversal gates and gauge fixing in topological stabilizer codes", 10.48550/ARXIV.1311.0879 (2013).

[98] B. J. Brown, N. H. Nickerson, and D. E. Browne, "Fault-tolerant error correction with the gauge color code", Nature Communications **7**, 10.1038/ncomms12302 (2016).

[99] A. Kubica and M. E. Beverland, "Universal transversal gates with color codes: a simplified approach", Phys. Rev. A **91**, 032330 (2015).

[100] A. J. Landahl, J. T. Anderson, and P. R. Rice, "Fault-tolerant quantum computing with color codes", arXiv:1108.5738, 10.48550/arXiv.1108.5738 (2011).

[101] A. Kubica, B. Yoshida, and F. Pastawski, "Unfolding the color code", New Journal of Physics **17**, 083026 (2015).

[102] A. B. Aloshious and P. K. Sarvepalli, "Projecting three-dimensional color codes onto three-dimensional toric codes", Physical Review A **98**, 10.1103/physreva.98.012302 (2018).

[103] M. B. Hastings, J. Haah, and R. O'Donnell, "Fiber Bundle Codes: Breaking the $N^{1/2}$polylog(N) Barrier for Quantum LDPC Codes", arXiv:2009.03921, 10.48550/arXiv.2009.03921 (2020).

[104] N. P. Breuckmann and J. N. Eberhardt, "Balanced product quantum codes", IEEE Transactions on Information Theory **67**, 6653 (2021).

[105] P. Das, A. Locharla, and C. Jones, "LILLIPUT: A Lightweight Low-Latency Lookup-Table Based Decoder for Near-term Quantum Error Correction", arXiv:2108.06569, 10.48550/arXiv.2108.06569 (2021).

[106] J. Edmonds, "Maximum matching and a polyhedron with 0,1-vertices", Journal of Research of the National Bureau of Standards Section B Mathematics and Mathematical Physics, 125 (1965).

[107] J. Edmonds, "Paths, trees, and flowers", Canadian Journal of Mathematics **17**, 449 (1965).

[108] A. G. Fowler, A. C. Whiteside, and L. C. L. Hollenberg, "Towards practical classical processing for the surface code", Physical Review Letters **108**, 10.1103/physrevlett.108.180501 (2012).

[109] A. G. Fowler, "Minimum weight perfect matching of fault-tolerant topological quantum error correction in average $O(1)$ parallel time", 10.48550/ARXIV.1307.1740 (2013).

[110] A. G. Fowler, "Optimal complexity correction of correlated errors in the surface code", arXiv:1310.0863, 10.48550/arXiv.1310.0863 (2013).

[111] O. Higgott, "PyMatching: A Python package for decoding quantum codes with minimum-weight perfect matching", arXiv:2105.13082, 10.48550/arXiv.2105.13082 (2021).

[112] S. Bravyi, A. W. Cross, J. M. Gambetta, D. Maslov, P. Rall, and T. J. Yoder, "High-threshold and low-overhead fault-tolerant quantum memory", en, Nature **627**, 778 (2024).

[113] D. Poulin and Y. Chung, "On the iterative decoding of sparse quantum codes", 10.48550/ARXIV.0801.1241 (2008).

[114] J. Roffe, D. R. White, S. Burton, and E. Campbell, "Decoding across the quantum low-density parity-check code landscape", Physical Review Research **2**, 10.1103/physrevresearch.2.043423 (2020).

[115] K.-Y. Kuo and C.-Y. Lai, "Exploiting degeneracy in belief propagation decoding of quantum codes", en, npj Quantum Information **8**, 111 (2022).

[116] O. Higgott, T. C. Bohdanowicz, A. Kubica, S. T. Flammia, and E. T. Campbell, "Improved Decoding of Circuit Noise and Fragile Boundaries of Tailored Surface Codes", en, Physical Review X **13**, 031007 (2023).

[117] B. Criger and I. Ashraf, "Multi-path summation for decoding 2d topological codes", Quantum **2**, 102 (2018).

[118] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, "Topological quantum memory", Journal of Mathematical Physics **43**, 4452 (2002).

[119] C. Wang, J. Harrington, and J. Preskill, "Confinement-higgs transition in a disordered gauge theory and the accuracy threshold for quantum memory", Annals of Physics **303**, 31 (2003).

[120] Y. Tomita and K. M. Svore, "Low-distance surface codes under realistic quantum noise", Physical Review A **90**, 10.1103/physreva.90.062320 (2014).

[121] L. Lao and B. Criger, "Magic state injection on the rotated surface code", in Proceedings of the 19th acm international conference on computing frontiers, CF '22 (2022), pp. 113–120.

[122] Y. Li, "A magic state's fidelity can be superior to the operations that created it", New Journal of Physics **17**, 10.1088/1367-2630/17/2/023037 (2015).

[123] C. Vuillot, L. Lao, B. Criger, and C. Garcí, "Code deformation and lattice surgery are gauge fixing", New Journal of Physics **21**, 033028 (2019).

[124] D. Litinski and F. v. Oppen, "Lattice surgery with a twist: simplifying clifford gates of surface codes", Quantum **2**, 10.22331/q-2018-05-04-62 (2018).

[125] K. Temme, S. Bravyi, and J. M. Gambetta, "Error mitigation for short-depth quantum circuits", Physical Review Letters **119**, 180509 (2017).

[126] Y. Li and S. C. Benjamin, "Efficient variational quantum simulator incorporating active error minimisation", Physical Review X **7**, 021050 (2017).

[127] S. Bravyi, J. M. Gambetta, A. Mezzacapo, and K. Temme, "Mitigating measurement errors in quantum computers", npj Quantum Information **7**, 173 (2021).

[128] P. D. Nation, B. Kang, P. Lougovski, and K. Temme, "Scalable mitigation of measurement errors on quantum computers", PRX Quantum **2**, 040326 (2021).

[129] X. Bonet-Monroig, R. Sagastizabal, D. Singh Roy, and T. E. O'Brien, "Low-cost error mitigation by symmetry verification", Physical Review A **98**, 062339 (2018).

[130] S. McArdle, S. Endo, A. Aspuru-Guzik, S. C. Benjamin, and X. Yuan, "Error-mitigated quantum computation with symmetry verification", npj Quantum Information **5**, 75 (2019).

[131] A. Siegel, A. Strikis, T. Flatters, and S. Benjamin, "Adaptive surface code for quantum error correction in the presence of temporary or permanent defects", Quantum **7**, 1065 (2023).

[132] A. Strikis, S. C. Benjamin, and B. J. Brown, "Quantum Computing is Scalable on a Planar Array of Qubits with Fabrication Defects", en, Physical Review Applied **19**, 064081 (2023).

[133] J. M. Martinis, "Saving superconducting quantum processors from qubit decay and correlated errors generated by gamma and cosmic rays", 10.1038/s41534-021-00431-0 (2020).

[134] C. D. Wilen, S. Abdullah, N. A. Kurinsky, C. Stanford, L. Cardani, G. D'Imperio, C. Tomei, L. Faoro, L. B. Ioffe, C. H. Liu, A. Opremcak, B. G. Christensen, J. L. DuBois, and R. McDermott, "Correlated charge noise and relaxation errors in superconducting qubits", Nature **594**, 369 (2021).

[135] "Resolving catastrophic error bursts from cosmic rays in large arrays of superconducting qubits", Nature Physics **18**, 107 (2021).

[136] J. Vala, K. B. Whaley, and D. S. Weiss, "Quantum error correction of a qubit loss in an addressable atomic system", Phys. Rev. A **72**, 052318 (2005).

[137] A. Bermudez, X. Xu, R. Nigmatullin, J. O'Gorman, V. Negnevitsky, P. Schindler, T. Monz, U. Poschinger, C. Hempel, J. Home, et al., "Assessing the progress of trapped-ion processors towards fault-tolerant quantum computation", Physical Review X **7**, 041061 (2017).

[138] I. Cong, H. Levine, A. Keesling, D. Bluvstein, S.-T. Wang, and M. D. Lukin, "Hardware-efficient, fault-tolerant quantum computation with rydberg atoms", 10.48550/ARXIV.2105.13501 (2021).

[139] N. C. Brown, M. Newman, and K. R. Brown, "Handling leakage with subsystem codes", New Journal of Physics **21**, 073055 (2019).

[140] B. J. Brown, K. Laubscher, M. S. Kesselring, and J. R. Wootton, "Poking holes and cutting corners to achieve clifford gates with the surface code", Phys. Rev. X **7**, 021029 (2017).

[141] "Suppressing quantum errors by scaling a surface code logical qubit", Nature **614**, 676 (2023).

[142] Q. Xu, A. Seif, H. Yan, N. Mannucci, B. O. Sane, R. Van Meter, A. N. Cleland, and L. Jiang, "Distributed quantum error correction for chip-level catastrophic errors", 10.48550/ARXIV.2203.16488 (2022).

[143] T. M. Stace, S. D. Barrett, and A. C. Doherty, "Thresholds for topological codes in the presence of loss", Physical Review Letters **102**, 10.1103/physrevlett.102.200501 (2009).

[144] S. Nagayama, A. G. Fowler, D. Horsman, S. J. Devitt, and R. V. Meter, "Surface code error correction on a defective lattice", New Journal of Physics **19**, 023050 (2017).

[145] J. M. Auger, H. Anwar, M. Gimeno-Segovia, T. M. Stace, and D. E. Browne, "Fault-tolerance thresholds for the surface code with fabrication errors", Physical Review A **96**, 10.1103/physreva.96.042316 (2017).

[146] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise", in Proceedings of the second international conference on knowledge discovery and data mining, KDD'96 (1996), pp. 226–231.

[147] O. Higgott and N. P. Breuckmann, "Subsystem codes with high thresholds by gauge fixing and reduced qubit overhead", Physical Review X **11**, 10.1103/physrevx.11.031039 (2021).

[148] L. Skoric, D. E. Browne, K. M. Barnes, N. I. Gillespie, and E. T. Campbell, "Parallel window decoding enables scalable fault tolerant quantum computation", Nature Communications **14**, 10.1038/s41467-023-42482-1 (2023).

[149] M. McEwen, D. Bacon, and C. Gidney, "Relaxing hardware requirements for surface code circuits using time-dynamics", 10.48550/arXiv.2302.02192 (2023).

[150] N. Delfosse and G. Zé, "Linear-time maximum likelihood decoding of surface codes over the quantum erasure channel", Physical Review Research **2**, 10.1103/physrevresearch.2.033042 (2020).

[151] P. Das, C. A. Pattison, S. Manne, D. Carmean, K. Svore, M. Qureshi, and N. Delfosse, "A Scalable Decoder Micro-architecture for Fault-Tolerant Quantum Computing", arXiv:2001.06598, 10.48550/arXiv.2001.06598 (2020).

[152] S. S. Tannu, Z. A. Myers, P. J. Nair, D. M. Carmean, and M. K. Qureshi, "Taming the instruction bandwidth of quantum computers via hardware-managed error correction", in Proceedings of the 50th annual ieee/acm international symposium on microarchitecture, MICRO-50 '17 (2017), pp. 679–691.

[153] Z. Wei, T. He, Y. Ye, D. Wu, Y. Zhang, Y. Zhao, W. Lin, H.-L. Huang, X. Zhu, and J.-W. Pan, "Low-overhead defect-adaptive surface code with bandage-like super-stabilizers", en, npj Quantum Information **11**, 75 (2025).

[154] D. M. Debroy, M. McEwen, C. Gidney, N. Shutty, and A. Zalcman, "LUCI in the Surface Code with Dropouts", arXiv:2410.14891, 10.48550/arXiv.2410.14891 (2024).

[155] A. Siegel, A. Strikis, and M. Fogarty, "Towards Early Fault Tolerance on a 2 × N Array of Qubits Equipped with Shuttling", en, PRX Quantum **5**, 040328 (2024).

[156] Z. Cai, A. Siegel, and S. Benjamin, "Looped pipelines enabling effective 3d qubit lattices in a strictly 2d device", PRX Quantum **4**, 020345 (2023).

[157] N. P. Breuckmann, C. Vuillot, E. Campbell, A. Krishna, and B. M. Terhal, "Hyperbolic and semi-hyperbolic surface codes for quantum storage", Quantum Science and Technology **2**, 035007 (2017).

[158] O. Higgott and N. P. Breuckmann, "Improved single-shot decoding of higher-dimensional hypergraph-product codes", PRX Quantum **4**, 020332 (2023).

[159] Q. Xu, J. P. Bonilla Ataides, C. A. Pattison, N. Raveendran, D. Bluvstein, J. Wurtz, B. Vasić, M. D. Lukin, L. Jiang, and H. Zhou, "Constant-overhead fault-tolerant quantum computation with reconfigurable atom arrays", en, Nature Physics **20**, 1084 (2024).

[160] J. Viszlai, W. Yang, S. F. Lin, J. Liu, N. Nottingham, J. M. Baker, and F. T. Chong, "Matching Generalized-Bicycle Codes to Neutral Atoms for Low-Overhead Fault-Tolerance", arXiv:2311.16980, 10.48550/arXiv.2311.16980 (2024).

[161] C. A. Pattison, A. Krishna, and J. Preskill, "Hierarchical memories: Simulating quantum LDPC codes with local gates", arXiv:2303.04798, 10.48550/arXiv.2303.04798 (2025).

[162] Y. Li and S. C. Benjamin, "One-dimensional quantum computing with a 'segmented chain' is feasible with today's gate fidelities", npj Quantum Information **4**, 10.1038/s41534-018-0074-2 (2018).

[163] A. T. E. Shaw, M. J. Bremner, A. Paler, D. Herr, and S. J. Devitt, "Quantum computation on a 19-qubit wide 2d nearest neighbour qubit array", arXiv:2212.01550, 10.48550/arXiv.2212.01550 (2022).

[164] J. M. Pino, J. M. Dreiling, C. Figgatt, J. P. Gaebler, S. A. Moses, M. S. Allman, C. H. Baldwin, M. Foss-Feig, D. Hayes, K. Mayer, C. Ryan-Anderson, and B. Neyenhuis, "Demonstration of the trapped-ion quantum ccd computer architecture", Nature **592**, 10.1038/s41586-021-03318-4 (2021).

[165] A. Siegel, Z. Cai, H. Jnane, B. Koczor, S. Pexton, A. Strikis, and S. Benjamin, "Snakes on a Plane: mobile, low dimensional logical qubits on a 2D surface", arXiv:2501.02120, 10.48550/arXiv.2501.02120 (2025).

[166] L. Hutin, B. Bertrand, E. Chanrion, H. Bohuslavskyi, F. Ansaloni, T.-Y. Yang, J. Michniewicz, D. J. Niegemann, C. Spence, T. Lundberg, A. Chatterjee, A. Crippa, J. Li, R. Maurand, X. Jehl, M. Sanquer, M. F. Gonzalez-Zalba, F. Kuemmeth, Y.-M. Niquet, S. De Franceschi, M. Urdampilleta, T. Meunier, and M. Vinet, "Gate reflectometry for probing charge and spin states in linear Si MOS split-gate arrays", in 2019 IEEE International Electron Devices Meeting (IEDM), ISSN: 2156-017X (Dec. 2019), pp. 37.7.1–37.7.4.

[167] L. C. Camenzind, S. Geyer, A. Fuhrer, R. J. Warburton, D. M. Zumbühl, and A. V. Kuhlmann, "A hole spin qubit in a fin field-effect transistor above 4 kelvin", Nature electronics **5**, 178 (2022).

[168] X. Xue, T. F. Watson, J. Helsen, D. R. Ward, D. E. Savage, M. G. Lagally, S. N. Coppersmith, M. A. Eriksson, S. Wehner, and L. M. K. Vandersypen, "Benchmarking gate fidelities in a si/sige two-qubit device", Physical Review X **9**, 10.1103/physrevx.9.021011 (2019).

[169] 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**, 10.1038/nnano.2014.153 (2014).

[170] O. Crawford, J. R. Cruise, N. Mertig, and M. F. Gonzalez-Zalba, "Compilation and scaling strategies for a silicon quantum processor with sparse two-dimensional connectivity", npj quantum information **9**, 13 (2023).

[171] E. Vahapoglu, J. P. Slack-Smith, R. C. C. Leon, W. H. Lim, F. E. Hudson, T. Day, T. Tanttu, C. H. Yang, A. Laucht, A. S. Dzurak, and J. J. Pla, "Single-electron spin resonance in a nanoelectronic device using a global field", Science Advances **7**, 10.1126/sciadv.abg9158 (2021).

[172] 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**, 10.1103/physrevx.8.021058 (2018).

[173] B. Klemt, V. Elhomsy, M. Nurizzo, P. Hamonic, B. Martinez, B. Cardoso Paz, C. Spence, M. C. Dartiailh, B. Jadot, E. Chanrion, V. Thiney, R. Lethiecq, B. Bertrand, H. Niebojewski, C. Bäuerle, M. Vinet, Y.-M. Niquet, T. Meunier, and M. Urdampilleta, "Electrical manipulation of a single electron spin in cmos using a micromagnet and spin-valley coupling", eng, npj quantum information **9**, 107 (2023).

[174] 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**, 10.1021/acs.nanolett.5b01306 (2015).

[175] A. West, B. Hensen, A. Jouan, T. Tanttu, C.-H. Yang, A. Rossi, M. F. Gonzalez-Zalba, F. Hudson, A. Morello, D. J. Reilly, and A. S. Dzurak, "Gate-based single-shot readout of spins in silicon", Nature Nanotechnology **14**, 10.1038/s41565-019-0400-7 (2019).

[176] D. C. McKay, C. J. Wood, S. Sheldon, J. M. Chow, and J. M. Gambetta, "Efficient z gates for quantum computing", en, Physical Review A **96**, 022330 (2017).

[177] 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**, 10.1088/2058-9565/aaf3c4 (2019).

[178] B. Buonacorsi, B. Shaw, and J. Baugh, "Simulated coherent electron shuttling in silicon quantum dots", Physical Review B **102**, 10.1103/physrevb.102.125406 (2020).

[179] Z. Cai and S. C. Benjamin, "Constructing smaller pauli twirling sets for arbitrary error channels", Scientific Reports **9**, 10.1038/s41598-019-46722-7 (2019).

[180] M. Veldhorst, J. C. C. Hwang, C. H. Yang, A. W. Leenstra, B. de Ronde, J. P. Dehollain, J. T. Muhonen, F. E. Hudson, K. M. Itoh, A. Morello, and A. S. Dzurak, "An addressable quantum dot qubit with fault-tolerant control-fidelity", Nature Nanotechnology **9**, 10.1038/nnano.2014.216 (2014).

[181] B. M. Maune, M. G. Borselli, B. Huang, T. D. Ladd, P. W. Deelman, K. S. Holabird, A. A. Kiselev, I. Alvarado-Rodriguez, R. S. Ross, A. E. Schmitz, M. Sokolich, C. A. Watson, M. F. Gyure, and A. T. Hunter, "Coherent singlet-triplet oscillations in a silicon-based double quantum dot", Nature **481**, 344 (2012).

[182] 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**, 10.1038/s41565-019-0488-9 (2019).

[183] K. Takeda, A. Noiri, T. Nakajima, L. C. Camenzind, T. Kobayashi, A. Sammak, G. Scappucci, and S. Tarucha, "Rapid single-shot parity spin readout in a silicon double quantum dot with fidelity exceeding 99%", npj quantum information **10**, 22 (2024).

[184] R. Raussendorf, J. Harrington, and K. Goyal, "Topological fault-tolerance in cluster state quantum computation", New Journal of Physics **9**, 199 (2007).

[185] S. Bosco, J. Zou, and D. Loss, "High-fidelity spin qubit shuttling via large spin-orbit interactions", PRX Quantum **5**, 10.1103/prxquantum.5.020353 (2024).

[186] R. Babbush, C. Gidney, D. W. Berry, N. Wiebe, J. McClean, A. Paler, A. Fowler, and H. Neven, "Encoding electronic spectra in quantum circuits with linear t complexity", Physical Review X **8**, 10.1103/physrevx.8.041015 (2018).

[187] M. A. Tremblay, N. Delfosse, and M. E. Beverland, "Constant-overhead quantum error correction with thin planar connectivity", Phys. Rev. Lett. **129**, 050504 (2022).

[188] A. Micciche, F. A. Mian, A. Chatterjee, A. McGregor, and S. Krastanov, "Optimizing compilation of error correction codes for 2xN quantum dot arrays and its NP-hardness", arXiv:2501.09061, 10.48550/arXiv.2501.09061 (2025).

[189] T. Tanttu, B. Hensen, K. W. Chan, C. H. Yang, W. W. Huang, M. Fogarty, F. Hudson, K. Itoh, D. Culcer, A. Laucht, A. Morello, and A. Dzurak, "Controlling spin-orbit interactions in silicon quantum dots using magnetic field direction", Phys. Rev. X **9**, 021028 (2019).

[190] 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**, 10.1038/ncomms3069 (2013).

[191] C. Tahan and R. Joynt, "Relaxation of excited spin, orbital, and valley qubit states in ideal silicon quantum dots", Phys. Rev. B **89**, 075302 (2014).

[192] H. Jnane, A. Siegel, and M. F. Gonzalez-Zalba, "Harnessing electron motion for global spin qubit control", arXiv:2503.12767, 10.48550/arXiv.2503.12767 (2025).

[193] B. E. Kane, "A silicon-based nuclear spin quantum computer", Nature **393**, 133 (1998).

[194] I. Hansen, A. E. Seedhouse, S. Serrano, A. Nickl, M. Feng, J. Y. Huang, T. Tanttu, N. Dumoulin Stuyck, W. H. Lim, F. E. Hudson, K. M. Itoh, A. Saraiva, A. Laucht, A. S. Dzurak, and C. H. Yang, "Entangling gates on degenerate spin qubits dressed by a global field", Nature Communications **15**, 10.1038/s41467-024-52010-4 (2024).

[195] A. Fayyaz and J. P. Kestner, "Enhanced local addressability of a spin array with local exchange pulses and global microwave driving", Physical Review B **108**, 10.1103/physrevb.108.l081303 (2023).

[196] M. Veldhorst, R. Ruskov, C. H. Yang, J. C. C. Hwang, F. E. Hudson, M. E. Flatté, C. Tahan, K. M. Itoh, A. Morello, and A. S. Dzurak, "Spin-orbit coupling and operation of multivalley spin qubits", Phys. Rev. B **92**, 201401 (2015).

[197] W. Huang, M. Veldhorst, N. M. Zimmerman, A. S. Dzurak, and D. Culcer, "Electrically driven spin qubit based on valley mixing", Phys. Rev. B **95**, 075403 (2017).

[198] R. Ferdous, "Valley dependent anisotropic spin splitting in silicon quantum dots", npj Quantum Information **4**, 26 (2018).

[199] R. Xue, M. Beer, I. Seidler, S. Humpohl, J.-S. Tu, S. Trellenkamp, T. Struck, H. Bluhm, and L. R. Schreiber, "Si/SiGe QuBus for single electron information-processing devices with memory and micron-scale connectivity function", Nature Communications **15**, 2296 (2024).

[200] R. Li, L. Petit, D. P. Franke, J. P. Dehollain, J. Helsen, M. Steudtner, N. K. Thomas, Z. R. Yoscovits, K. J. Singh, S. Wehner, L. M. K. Vandersypen, J. S. Clarke, and M. Veldhorst, "A crossbar network for silicon quantum dot qubits", Science Advances **4**, eaar3960 (2018).

[201] A. R. Mills, C. R. Guinn, M. J. Gullans, A. J. Sigillito, M. M. Feldman, E. Nielsen, and J. R. Petta, "Two-qubit silicon quantum processor with operation fidelity exceeding 99%", Science Advances **8**, eabn5130 (2022).

[202] M. M. E. K. Shehata, G. Simion, R. Li, F. A. Mohiyaddin, D. Wan, M. Mongillo, B. Govoreanu, I. Radu, K. De Greve, and P. Van Dorpe, "Modeling semiconductor spin qubits and their charge noise environment for quantum gate fidelity estimation", Phys. Rev. B **108**, 045305 (2023).

[203] J. D. Cifuentes, P. Y. Mai, F. Schlattner, H. E. Ercan, M. Feng, C. C. Escott, A. S. Dzurak, and A. Saraiva, "Path-integral simulation of exchange interactions in cmos spin qubits", Phys. Rev. B **108**, 155413 (2023).

[204] H. Jnane and S. C. Benjamin, "*Ab initio* modeling of quantum dot qubits: Coupling, gate dynamics, and robustness versus charge noise", en, Physical Review Applied **24**, 044042 (2025).

[205] C. H. Yang, K. W. Chan, R. Harper, W. Huang, T. Evans, J. C. C. Hwang, B. Hensen, A. Laucht, T. Tanttu, F. E. Hudson, S. T. Flammia, K. M. Itoh, A. Morello, S. D. Bartlett, and A. S. Dzurak, "Silicon qubit fidelities approaching incoherent noise limits via pulse engineering", Nature Electronics **2**, 10.1038/s41928-019-0234-1 (2019).

[206] C.-A. Wang, V. John, H. Tidjani, C. X. Yu, A. S. Ivlev, C. Déprez, F. van Riggelen-Doelman, B. D. Woods, N. W. Hendrickx, W. I. L. Lawrie, L. E. A. Stehouwer, S. D. Oosterhout, A. Sammak, M. Friesen, G. Scappucci, S. L. de Snoo, M. Rimbach-Russ, F. Borsoi, and M. Veldhorst, "Operating semiconductor quantum processors with hopping spins", Science **385**, 447 (2024).

[207] M. Kpa, N. Focke, Ł. Cywiński, and J. A. Krzywda, ""Simulation of 1/f charge noise affecting a quantum dot in a Si/SiGe structure"", Applied Physics Letters **123**, 034005 (2023).

[208] J. Yoneda, J. S. Rojas-Arias, P. Stano, K. Takeda, A. Noiri, T. Nakajima, D. Loss, and S. Tarucha, "Noise-correlation spectrum for a pair of spin qubits in silicon", Nature Physics **19**, 1793 (2023).

[209] J. Rojas-Arias, A. Noiri, P. Stano, T. Nakajima, J. Yoneda, K. Takeda, T. Kobayashi, A. Sammak, G. Scappucci, D. Loss, and S. Tarucha, "Spatial noise correlations beyond nearest neighbors in ^{28}Si/si-ge spin qubits", Phys. Rev. Appl. **20**, 054024 (2023).

[210] F. Ginzel, A. R. Mills, J. R. Petta, and G. Burkard, "Spin shuttling in a silicon double quantum dot", Phys. Rev. B **102**, 195418 (2020).

[211] A. R. Mills, D. M. Zajac, M. J. Gullans, F. J. Schupp, T. M. Hazard, and J. R. Petta, "Shuttling a single charge across a one-dimensional array of silicon quantum dots", Nature Communications **10**, 1063 (2019).

[212] M. Künne, A. Willmes, M. Oberländer, C. Gorjaew, J. D. Teske, H. Bhardwaj, M. Beer, E. Kammerloher, R. Otten, I. Seidler, R. Xue, L. R. Schreiber, and H. Bluhm, "The SpinBus architecture for scaling spin qubits with electron shuttling", Nature Communications **15**, 4977 (2024).

[213] M. Jeon, S. C. Benjamin, and A. J. Fisher, "Robustness of electron charge shuttling: Architectures, pulses, charge defects, and noise thresholds", en, Physical Review B **111**, 195302 (2025).

[214] R. Nagai, T. Takemoto, Y. Wachi, and H. Mizuno, "Digital-Controlled Method of Conveyor-Belt Spin Shuttling in Silicon for Large-Scale Quantum Computation", arXiv:2502.20955, 10.48550/arXiv.2502.20955 (2025).

[215] F. H. L. Koppens, C. Buizert, K. J. Tielrooij, I. T. Vink, K. C. Nowack, T. Meunier, L. P. Kouwenhoven, and L. M. K. Vandersypen, "Driven coherent oscillations of a single electron spin in a quantum dot", Nature **442**, 766 (2006).

[216] C. C. Lo, V. Lang, R. E. George, J. J. L. Morton, A. M. Tyryshkin, S. A. Lyon, J. Bokor, and T. Schenkel, "Electrically detected magnetic resonance of neutral donors interacting with a two-dimensional electron gas", Phys. Rev. Lett. **106**, 207601 (2011).

[217] A. Siegel, K. Mitarai, and K. Fujii, "Algorithmic error mitigation for quantum eigenvalues estimation", arXiv:2308.03879, 10.48550/arXiv.2308.03879 (2024).

[218] L. Postler, S. Heuen, I. Pogorelov, M. Rispler, T. Feldker, M. Meth, C. D. Marciniak, R. Stricker, M. Ringbauer, R. Blatt, P. Schindler, M. Müller, and T. Monz, "Demonstration of fault-tolerant universal quantum gate operations", en, Nature **605**, 675 (2022).

[219] R. Takagi, S. Endo, S. Minagawa, and M. Gu, "Fundamental limits of quantum error mitigation", npj Quantum Information **8**, 10.1038/s41534-022-00618-z (2022).

[220] Y. Suzuki, S. Endo, K. Fujii, and Y. Tokunaga, "Quantum error mitigation as a universal error reduction technique: applications from the nisq to the fault-tolerant quantum computing eras", PRX Quantum **3**, 10.1103/prxquantum.3.010345 (2022).

[221] A. Y. Kitaev, *Quantum measurements and the abelian stabilizer problem*, 1995.

[222] C. J. O'Loan, "Iterative phase estimation", Journal of Physics A: Mathematical and Theoretical **43**, 015301 (2009).

[223] S. Kimmel, G. H. Low, and T. J. Yoder, "Robust calibration of a universal single-qubit gate set via robust phase estimation", Physical Review A **92**, 10.1103/physreva.92.062315 (2015).

[224] K. Wan, M. Berta, and E. T. Campbell, "Randomized quantum algorithm for statistical phase estimation", Physical Review Letters **129**, 10.1103/physrevlett.129.030503 (2022).

[225] N. Hatano and M. Suzuki, in *Quantum annealing and other optimization methods* (Springer Berlin Heidelberg, Nov. 2005), pp. 37–68.

[226] C. Yi and E. Crosson, "Spectral analysis of product formulas for quantum simulation", en, npj Quantum Information **8**, 37 (2022).

[227] D. W. Berry, A. M. Childs, R. Cleve, R. Kothari, and R. D. Somma, "Simulating hamiltonian dynamics with a truncated taylor series", Physical Review Letters **114**, 10.1103/physrevlett.114.090502 (2015).

[228] D. W. Berry, M. Kieferová, A. Scherer, Y. R. Sanders, G. H. Low, N. Wiebe, C. Gidney, and R. Babbush, "Improved techniques for preparing eigenstates of fermionic hamiltonians", npj Quantum Information **4**, 10.1038/s41534-018-0071-5 (2018).

[229] D. Poulin, A. Kitaev, D. S. Steiger, M. B. Hastings, and M. Troyer, "Quantum algorithm for spectral measurement with a lower gate count", Physical Review Letters **121**, 10.1103/physrevlett.121.010501 (2018).

[230] G. H. Low and I. L. Chuang, "Hamiltonian simulation by qubitization", Quantum **3**, 163 (2019).

[231] Y. Akahoshi, K. Maruyama, H. Oshima, S. Sato, and K. Fujii, "Partially Fault-Tolerant Quantum Computing Architecture with Error-Corrected Clifford Gates and Space-Time Efficient Analog Rotations", en, PRX Quantum **5**, 010337 (2024).

[232] K. Temme, S. Bravyi, and J. M. Gambetta, "Error mitigation for short-depth quantum circuits", Physical Review Letters **119**, 10.1103/physrevlett.119.180509 (2017).

[233] S. Endo, Q. Zhao, Y. Li, S. Benjamin, and X. Yuan, "Mitigating algorithmic errors in a hamiltonian simulation", Physical Review A **99**, 10.1103/physreva.99.012334 (2019).

[234] A. C. Vazquez, R. Hiptmair, and S. Woerner, "Enhancing the quantum linear systems algorithm using richardson extrapolation", ACM Transactions on Quantum Computing **3**, 10.1145/3490631 (2022).

[235] A. Gonzales, A. M. Babu, J. Liu, Z. Saleem, and M. Byrd, "Fault Tolerant Quantum Error Mitigation", arXiv:2308.05403, 10.48550/arXiv.2308.05403 (2024).

[236] M. A. Wahl, A. Mari, N. Shammah, W. J. Zeng, and G. S. Ravi, "Zero noise extrapolation on logical qubits by scaling the error correction code distance", arXiv:2304.14985, 10.48550/arXiv.2304.14985 (2023).

[237] T. Kato, *Perturbation theory for linear operators* (Springer Berlin, Heidelberg, 1995) Chap. 2.

[238] A. S. Mokeev, Y.-N. Zhang, and V. V. Dobrovitski, "Modeling of decoherence and fidelity enhancement during transport of entangled qubits", arXiv:2409.04404, 10.48550/arXiv.2409.04404 (2024).

[239] K. Sahay, J. Jin, J. Claes, J. D. Thompson, and S. Puri, "High-threshold codes for neutral-atom qubits with biased erasure errors", Phys. Rev. X **13**, 041013 (2023).