

# Quantum Error Correction: An Introductory Guide

Joschka Roffe

*Department of Physics & Astronomy, University of Sheffield, Sheffield, S3 7RH, United Kingdom*

Quantum error correction protocols will play a central role in the realisation of quantum computing; the choice of error correction code will influence the full quantum computing stack, from the layout of qubits at the physical level to gate compilation strategies at the software level. As such, familiarity with quantum coding is an essential prerequisite for the understanding of current and future quantum computing architectures. In this review, we provide an introductory guide to the theory and implementation of quantum error correction codes. Where possible, fundamental concepts are described using the simplest examples of detection and correction codes, the working of which can be verified by hand. We outline the construction and operation of the surface code, the most widely pursued error correction protocol for experiment. Finally, we discuss issues that arise in the practical implementation of the surface code and other quantum error correction codes.

**Keywords:** Quantum computing; quantum error correction; stabilizer codes; surface codes

## 1. Introduction

In place of the bits in traditional computers, quantum computers work by controlling and manipulating quantum bits (qubits). Through the precise control of quantum phenomena such as entanglement, it is in principle possible for such qubit-based devices to outperform their classical counterparts. To this end, efficient quantum computing algorithms have been developed with applications such as integer factorisation [1], search [2], optimisation [3] and quantum chemistry [4].

There is currently no preferred qubit technology; a variety of physical systems are being explored for use as qubits, including photons [5,6], trapped ions [7–10], superconducting circuits [11–13] and spins in semiconductors [14–16]. A shortcoming shared by all of these approaches is that it is difficult to sufficiently isolate the qubits from the effects of external noise, meaning errors during quantum computation are inevitable. In contrast, bits in a classical computer are typically realised as the robust *on/off* states of transistor switches which are differentiated by billions of electrons. This provides classical bits with high error margins that near-eradicate failures at the physical level. For quantum computers, where qubits are realised as fragile quantum systems, there is no such security against errors. As such, any circuit-model quantum computer based on current and future qubit technologies will require some sort of active error correction.

Driven by the demands of high-performance communication networks and the Internet, there is a well-developed theory of classical error correction [17–19]. However, adapting existing classical methods for quantum error correction is not straightforward. Qubits are subject to the no-cloning theorem [20], meaning quantum information cannot be duplicated in the same way as classical information. Furthermore, it is not possible to perform arbitrary measurements on a qubit register due to the problem of wavefunction collapse. It was initially feared that these constraints would pose an insurmountable challenge to the viability of quantum computing. However, a breakthrough

was reached in 1995 by Peter Shor with a paper proposing the first quantum error correction scheme [21]. Shor’s method demonstrated how quantum information can be redundantly encoded by entangling it across an expanded system of qubits. Subsequent results then demonstrated that extensions to this technique can in principle be used to arbitrarily suppress the quantum error rate, provided certain physical conditions on the qubits themselves are met [22–26]. It was with these developments in quantum error correction that the field of quantum computing moved from a theoretical curiosity to a practical possibility.

Many reviews have been written covering quantum error correction and its associated subfields [27–33]. This work is intended as an introductory guide where we describe the essential concepts behind quantum error correction codes through the use of simple examples. The ultimate aim is to provide the reader with sufficient background to understand the construction and operating principles behind the surface code, the most widely pursued error correction scheme for experimental implementation [34]. Crucially, our descriptions of the surface code do not rely upon terminology from topology and homology, as is the case with many of the original sources. Whilst this review does not require prior knowledge of coding theory or error correction, we do assume an understanding of elementary quantum mechanics and the circuit model of quantum computing. The reader should be comfortable with quantum circuit notation, as seen for example in [35], and be familiar with standard gates such as the Hadamard gate ( $H$ ), the controlled-NOT gate (CNOT) and measurement operations in the computational basis. A brief outline of these gates, as well as the conventions we adopt for labelling quantum states and operators, can be found in appendices A-C.

In section 2, we begin by explaining the differences between bits and qubits, before describing the principal challenges in designing quantum error correction codes. Section 3 outlines how quantum information is redundantly encoded, and explains how errors can be detected by performing projective measurements. In section 4, we introduce the stabilizer framework which allows for the construction a large class of quantum error correction codes. Following this, the surface code is described in section 5. Finally, in section 6, we discuss some of the practical issues that arise when considering the implementation of quantum error correction codes on realistic hardware.

## 2. From classical to quantum error correction

Classical information technologies employ binary encodings in which data is represented as sequences of bits taking values ‘0’ or ‘1’. The basic principle behind error correction is that the number of bits used to encode a given amount of information is increased. The exact way in which this *redundant* encoding is achieved is specified by a set of instructions known as an *error correction code* [18,19].

The simplest example of an error correction code is the three-bit repetition code, the encoder for which duplicates each bit value  $0 \rightarrow 000$  and  $1 \rightarrow 111$ . More formally, we can define the three-bit encoder as a mapping from a ‘raw’ binary alphabet  $\mathcal{B}$  to a code alphabet  $C_3$

$$\mathcal{B} = \{0, 1\} \xrightarrow{\text{three-bit encoding}} C_3 = \{000, 111\}, \quad (1)$$

where the encoded bit-strings ‘000’ and ‘111’ are referred to as the *logical codewords* of the code  $C_3$ . As an example, consider the simple case where we wish to communicate a single-bit message ‘0’ to a recipient in a different location. Using the three bit encoding, the message that we would send would be the ‘000’ codeword.

Now, imagine that the message is subject to a single bit-flip error during transmission so that the bit-string the recipient receives is ‘010’. In this scenario, the recipient will be able to infer that the intended codeword is ‘000’ via a majority vote. The same will be true for all cases where the codeword is subject to only a single error. However, if the codeword is subject to two bit-flip errors, the majority vote will lead to the incorrect codeword. The final scenario to consider is when all

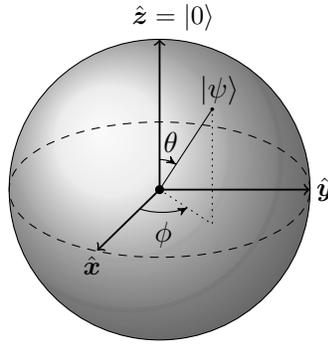


Figure 1. In the geometric representation, the state of a qubit  $|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle$  can be represented as a point on the surface of a Bloch sphere.

three bits are flipped so that the codeword ‘000’ becomes ‘111’. In this case, the corrupted message is also a codeword: the recipient will therefore have no way of knowing an error has occurred. The *distance* of a code is defined as the minimum number of errors that will change one codeword to another in this way. We can relate the distance  $d$  of a code to the number of errors it can correct as follows

$$d = 2t + 1. \quad (2)$$

where  $t$  is the number of errors the code can correct. It is clear that the above equation is satisfied for the three-bit code where  $t = 1$  and  $d = 3$ .

In general, error correction codes are described in terms of the  $[n, k, d]$  notation, where  $n$  is the total number of bits per codeword,  $k$  is the number of encoded bits (the length of the original bit-string) and  $d$  is the code distance. Under this notation, the three-bit repetition code is labelled  $[3, 1, 3]$ .

### 2.1. From bits to qubits

In place of bits in classical systems, the fundamental unit of quantum information is the *qubit*. The general qubit state can be written as follows

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \quad (3)$$

where  $\alpha$  and  $\beta$  are complex numbers that satisfy the condition  $|\alpha|^2 + |\beta|^2 = 1$ . Details regarding the notation we use to represent quantum states can be found in appendix A. Qubits can encode information in a superposition of their basis states, meaning quantum computers have access to a computational space that scales as  $2^n$  where  $n$  is the total number of qubits [35]. It is by exploiting superposition, in combination with other quantum effects such as entanglement, that it is possible to construct algorithms that provide a quantum advantage [1,2]. However, if such algorithms are ever to be realised on current or future quantum hardware, it will be necessary for the qubits to be error corrected.

### 2.2. The digitisation of quantum errors

In classical information, bits are either in the ‘0’ or ‘1’ state. Therefore, the only error-type to be considered is the bit-flip that takes  $0 \rightarrow 1$  and vice-versa. In contrast, the general qubit state defined in equation (3) can assume a continuum of values between its basis states. From the perspective of

developing error correction codes, this property is problematic as it means the qubit is subject to an infinite number of errors. To illustrate this more clearly, it is useful to rewrite the general qubit state in terms of a geometric representation given by

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle, \quad (4)$$

where the probability amplitudes maintain the condition that  $|\cos\frac{\theta}{2}|^2 + |e^{i\phi}\sin\frac{\theta}{2}|^2 = 1$ . In this form, the qubit state corresponds to a point, specified by the angles  $\theta$  and  $\phi$ , on the surface of a so-called Bloch sphere. An example state in this representation is shown in figure 1.

Qubit errors can occur by a variety of physical processes. The simplest case to examine are errors which cause the qubit to coherently rotate from one point on the Bloch sphere to another. Such qubit errors could, for example, arise from systematic control faults in the hardware with which the qubits are realised. Mathematically, coherent errors are described by a unitary operation  $U(\delta\theta, \delta\phi)$  which evolves the qubit state as follows

$$U(\delta\theta, \delta\phi)|\psi\rangle = \cos\frac{\theta + \delta\theta}{2}|0\rangle + e^{i(\phi + \delta\phi)}\sin\frac{\theta + \delta\theta}{2}|1\rangle, \quad (5)$$

where  $\theta + \delta\theta$  and  $\phi + \delta\phi$  are the new coordinates on the Bloch sphere. From this, we see that qubits are susceptible to a continuum of coherent errors obtained by varying the parameters  $\delta\theta$  and  $\delta\phi$ . It would therefore seem, at first glance, that quantum error correction protocols should have to be based on techniques from classical analogue computation for which the theory of error correction is not well developed. Luckily, however, it turns out that quantum errors can be digitised so that the ability to correct for a finite set of errors is sufficient to correct for any error [36]. To see how this is possible, we first note that coherent noise processes are described by matrices that can be expanded terms of a Pauli basis.<sup>1</sup> For example, the Pauli basis for two-dimensional matrices is given by

$$\mathbb{1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \quad (6)$$

The single-qubit coherent error process described in equation (5) can be expanded in the above basis as follows

$$U(\delta\theta, \delta\phi)|\psi\rangle = \alpha_I\mathbb{1}|\psi\rangle + \alpha_X X|\psi\rangle + \alpha_Z Z|\psi\rangle + \alpha_Y Y|\psi\rangle \quad (7)$$

where  $\alpha_{I,X,Y,Z}$  are the expansion coefficients. By noting that the Pauli  $Y$ -matrix is equivalent (up to a phase) to the product  $XZ$ , this expression can be further simplified to

$$U(\delta\theta, \delta\phi)|\psi\rangle = \alpha_I\mathbb{1}|\psi\rangle + \alpha_X X|\psi\rangle + \alpha_Z Z|\psi\rangle + \alpha_{XZ} XZ|\psi\rangle. \quad (8)$$

The above expression shows that any coherent error process can be decomposed into a sum from the Pauli set  $\{\mathbb{1}, X, Z, XZ\}$ . In the following sections, we will see that the error correction process itself involves performing projective measurements that cause the above superposition to collapse to a subset of its terms. As a result, a quantum error correction code with the ability to correct errors described the by the  $X$ - and  $Z$ -Pauli matrices will be able to correct any coherent error. This effect, referred to as the digitisation of the error, is crucial to the success of quantum error correction codes.

---

<sup>1</sup>For a more detailed definition of the Pauli group, and details of the Pauli notation used in this review, see appendix B.

### 2.3. Quantum error-types

As a result of the digitisation of the error there are two fundamental quantum error-types that need to be accounted for by quantum codes. Pauli  $X$ -type errors can be thought of as quantum bit-flips that map  $X|0\rangle = |1\rangle$  and  $X|1\rangle = |0\rangle$ . The action of an  $X$ -error on the general qubit state is

$$X|\psi\rangle = \alpha X|0\rangle + \beta X|1\rangle = \alpha|1\rangle + \beta|0\rangle. \quad (9)$$

The second quantum error type, the  $Z$ -error, is often referred to as a phase-flip and has no classical analogue. Phase-flips map the qubit basis states  $Z|0\rangle = |0\rangle$  and  $Z|1\rangle = -|1\rangle$ , and therefore have the following action on the general qubit state

$$Z|\psi\rangle = \alpha Z|0\rangle + \beta Z|1\rangle = \alpha|0\rangle - \beta|1\rangle. \quad (10)$$

So far, for simplicity, I have restricted discussion to coherent errors acting on single-qubits. However, the digitisation of the error result generalises to arbitrary quantum error processes, including those that describe incoherent evolution of the quantum state as a result of the qubits' interaction with their environment [36].

### 2.4. The challenges of quantum error correction

The digitisation of quantum errors means it is possible to reuse certain techniques from classical coding theory in quantum error correction. However, there remain a number of complications that prevent the straight-forward translation of classical codes to quantum codes. The first complication is the no-cloning theorem for quantum states [20], which asserts that it is not possible to construct a unitary operator  $U_{\text{clone}}$  which performs the following operation

$$U_{\text{clone}}(|\psi\rangle \otimes |0\rangle) \rightarrow |\psi\rangle \otimes |\psi\rangle, \quad (11)$$

where  $|\psi\rangle$  is the state to be cloned. In contrast, classical codes work under the assumption that data can be arbitrarily duplicated. For quantum coding, it is therefore necessary to find alternative ways of adding redundancy to the system.

The second complication in quantum coding arises from the fact that qubits are susceptible to both bit-flips ( $X$ -errors) and phase-flips ( $Z$ -errors). Quantum error correction codes must therefore be designed with the ability to detect both error-types simultaneously. In contrast, in classical coding, only bit-flip errors need to be considered.

The final complication specific to quantum error correction is the problem of wavefunction collapse. In a classical system, it is possible to measure arbitrary properties of the bit register without risk of compromising the encoded information. For quantum codes, however, any measurements of the qubits performed as part of the error correction procedure must be carefully chosen so as not to cause the wavefunction to collapse and erase the encoded information. In the next section, we will see how this is achieved through the use of a special type of projective measurement referred to as a stabilizer measurement [37].

## 3. Quantum redundancy & stabilizer measurement

As outlined in the previous section, quantum error correction is complicated by the no-cloning theorem, wavefunction collapse and the existence of a uniquely quantum error-type, the phase-flip. So, faced with these challenges, how is redundancy added to a quantum system to allow errors to

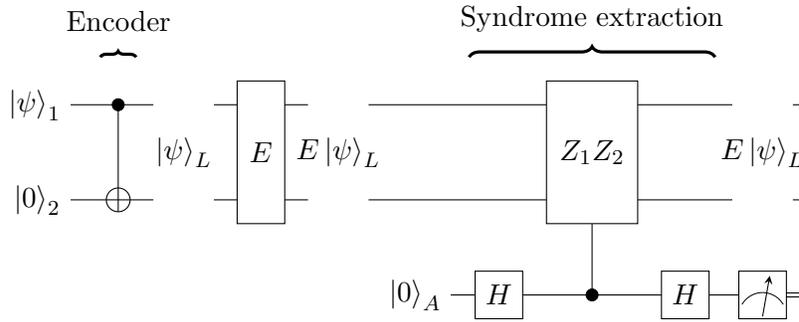


Figure 2. Circuit diagram for the two qubit code. Encode stage: the information contained in  $|\psi\rangle_1$  is entangled with a redundancy qubit  $|0\rangle_2$  to create a logical state  $|\psi\rangle_L$ . Error stage: during the error window (shown by the circuit element  $E$ ), the two code qubits are potentially subject to bit-flip errors. Syndrome extraction stage: the  $Z_1Z_2$  operator, controlled by the ancilla qubit  $A$ , is applied to the code qubits. The subsequent measurement of the ancilla gives the code syndrome  $S$ .

be detected in real time? Classical repetition codes work by increasing the resources used to encode the data beyond the theoretical minimum. Analogously, in quantum codes redundancy is added by expanding the Hilbert space in which the qubits are encoded [21]. To see how this is achieved in practice, we now describe the two-qubit code, a prototypical quantum code designed to detect a single-bit flip error. The encode stage of the two-qubit code, acting on the general state  $|\psi\rangle$ , has the following action

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle \xrightarrow{\text{two-qubit encoder}} |\psi\rangle_L = \alpha |00\rangle + \beta |11\rangle = \alpha |0\rangle_L + \beta |1\rangle_L, \quad (12)$$

where after encoding the logical codewords are  $|0\rangle_L = |00\rangle$  and  $|1\rangle_L = |11\rangle$ . Note that this does not correspond to cloning the state as

$$|\psi\rangle_L = \alpha |00\rangle + \beta |11\rangle \neq |\psi\rangle \otimes |\psi\rangle. \quad (13)$$

The effect of the encoding operation is to distribute the quantum information in the initial state  $|\psi\rangle$  across the entangled two-party logical state  $|\psi\rangle_L$ . This introduces redundancy to the encoding that can be exploited for error detection. To understand exactly how this works, it is instructive to consider the computational Hilbert spaces before and after encoding. Prior to encoding, the single qubit is parametrised within a two-dimensional Hilbert space  $|\psi\rangle \in \mathcal{H}_2 = \text{span}\{|0\rangle, |1\rangle\}$ . After encoding the logical qubit occupies a four-dimensional Hilbert space

$$|\psi\rangle \in \mathcal{H}_4 = \text{span}\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}. \quad (14)$$

More specifically the logical qubit is defined within a two-dimensional subspace of this expanded Hilbert space

$$|\psi\rangle_L \in \mathcal{C} = \text{span}\{|00\rangle, |11\rangle\} \subset \mathcal{H}_4, \quad (15)$$

where  $\mathcal{C}$  is called the codespace. Now, imagine that the logical qubit is subject a bit-flip error on the first qubit resulting in the state

$$X_1 |\psi\rangle_L = \alpha |10\rangle + \beta |01\rangle, \quad (16)$$

Table 1. The syndrome table for the two-qubit code. The syndrome  $S$  is bit a string where each bit corresponds to the outcome of a stabilizer measurement.

Error	Syndrome, $S$
$I_1 I_2$	0
$X_1 I_2$	1
$I_1 X_2$	1
$X_1 X_2$	0

where  $X_1$  is a bit-flip error acting on the first qubit. The resultant state is rotated into a new subspace

$$X_1 |\psi\rangle_L \in \mathcal{F} \subset \mathcal{H}_4, \quad (17)$$

where we call  $\mathcal{F}$  the error subspace. Notice that an  $X_2$ -error will also rotate the logical state into the  $\mathcal{F}$  subspace. If the logical state  $|\psi\rangle_L$  is uncorrupted, it occupies the codespace  $\mathcal{C}$ , whereas if it has been subject to a single-qubit bit-flip, it occupies the error space  $\mathcal{F}$ . As the  $\mathcal{C}$  and  $\mathcal{F}$  subspaces are mutually orthogonal, it is possible to distinguish which subspace the logical qubit occupies via a projective measurement without compromising the encoded quantum information. In the context of quantum coding, measurements of this type are called stabilizer measurements.

For the purposes of differentiating between the codespace  $\mathcal{C}$  and the error space  $\mathcal{F}$ , a projective measurement of the form  $Z_1 Z_2$  is performed. The  $Z_1 Z_2$  operator yields a (+1) eigenvalue when applied to the logical state

$$Z_1 Z_2 |\psi\rangle_L = Z_1 Z_2 (\alpha |00\rangle + \beta |11\rangle) = (+1) |\psi\rangle_L. \quad (18)$$

The  $Z_1 Z_2$  operator is said to *stabilize* the logical qubit  $|\psi\rangle_L$  as it leaves it unchanged [28]. Conversely, the  $Z_1 Z_2$  operator projects the errored states,  $X_1 |\psi\rangle_L$  and  $X_2 |\psi\rangle_L$ , onto the (-1) eigenspace. Notice that for either outcome, the information encoded in the  $\alpha$  and  $\beta$  coefficients of the logical state remains undisturbed.

Figure 2 shows the circuit implementation of the two-qubit code. In the encode stage, a CNOT gate is used to entangle the  $|\psi\rangle$  state with a redundancy qubit to create the logical state  $|\psi\rangle_L$ . Following this, we assume the logical qubit is subject to a bit-flip error  $E$ , applied during the stage of the circuit labelled ‘E’. Following the error stage, an ancilla qubit  $|0\rangle_A$  is introduced to perform the measurement of the  $Z_1 Z_2$  stabilizer. The syndrome extraction stage of the circuit transforms the quantum state as follows

$$E |\psi\rangle_L |0\rangle_A \xrightarrow{\text{syndrome extraction}} \frac{1}{2}(\mathbb{1}_1 \mathbb{1}_2 + Z_1 Z_2) E |\psi\rangle_L |0\rangle_A + \frac{1}{2}(\mathbb{1}_1 \mathbb{1}_2 - Z_1 Z_2) E |\psi\rangle_L |1\rangle_A, \quad (19)$$

where  $E$  is an error from the set  $\{\mathbb{1}, X_1, X_2, X_1 X_2\}$ . Now, consider the case where  $E = X_1$  so that the logical state occupies the error space  $E |\psi\rangle_L \in \mathcal{F}$ . In this scenario, it can be seen that the first term in equation (19) goes to zero. The ancilla qubit is therefore measured deterministically as ‘1’. Considering the other error patterns, we see that if the logical state is in the codespace (i.e., if  $E = \{\mathbb{1}, X_1 X_2\}$ ) then the ancilla is measured as ‘0’. Likewise, if the logical state is in the error subspace (i.e., if  $E = \{X_1, X_2\}$ ) then the ancilla is measured as ‘1’. The outcome of the ancilla qubit measurement is referred to as a *syndrome*, and tells us whether or not the logical state has

been subject to an error. The syndromes for all bit-flip error types in the two-qubit code are shown in table 1.

Up to this point, we have assumed that the error introduced by the circuit element labelled ‘ $E$ ’ is deterministic. We now demonstrate how the two qubit code works under a more general probabilistic error of the type discussed in section 2.2. For the purposes of this example, we will assume that each qubit in the two-qubit code is subject to a coherent error of the form

$$\mathcal{E} = \alpha_I \mathbb{1} + \alpha_X X, \quad (20)$$

where  $|\alpha_I|^2 + |\alpha_X|^2 = 1$ . Here we see that  $|\alpha_X|^2 = p_X$  is the probability of an  $X$ -error occurring on the qubit. The probability of no-error occurring is therefore equal to  $|\alpha_I|^2 = 1 - p_X$ . The combined action of the error operator  $\mathcal{E}$  acting on both qubits is given by

$$E = \mathcal{E}_1 \otimes \mathcal{E}_2 = \alpha_I^2 \mathbb{1}_1 \mathbb{1}_2 + \alpha_I \alpha_X (X_1 + X_2) + \alpha_X^2 X_1 X_2. \quad (21)$$

With the above error operator  $E$ , the syndrome extraction stage in figure 2 stage transforms the quantum state as follows

$$E |\psi\rangle_L |0\rangle_A \xrightarrow{\text{syndrome extraction}} (\alpha_I^2 \mathbb{1}_1 \mathbb{1}_2 + \alpha_X^2 X_1 X_2) |\psi\rangle_L |0\rangle_A + \alpha_I \alpha_X (X_1 + X_2) |\psi\rangle_L |1\rangle_A. \quad (22)$$

If the syndrome is measured as ‘0’, the state collapses to a subset of its terms

$$\frac{(\alpha_I^2 \mathbb{1}_1 \mathbb{1}_2 + \alpha_X^2 X_1 X_2) |\psi\rangle_L |0\rangle_A}{\sqrt{|\alpha_I|^2 + |\alpha_X|^2}} \quad (23)$$

where the the denominator ensures normalisation. By calculating the square-norm in the first term in the above, we can calculate the probability  $p_L$  that the logical state is subject to an error

$$p_L = \left| \frac{\alpha_X^2}{\sqrt{|\alpha_I|^2 + |\alpha_X|^2}} \right|^2 = \frac{p_x^2}{(1 - p_x)^2 + p_x^2} \approx p_x^2 \quad (24)$$

where the above approximation is made under the assumption that  $p_x$  is small. For the single qubit  $|\psi\rangle$ , the probability of error is  $p_x$  when it is subject to the error operator  $\mathcal{E}$ . For the logical qubit  $|\psi\rangle_L$  subject to the error operator  $\mathcal{E}_1 \otimes \mathcal{E}_2$ , the logical error rate is  $p_L = p_x^2$ . From this, we see that the two-qubit code suppresses the error rate relative to the un-encoded case.

### 3.1. The three-qubit error correction code

The syndrome produced by the two-qubit code informs us of the presence of an error, but does not provide enough information to allow us to infer which qubit the error occurred on. It is therefore a detection code. In order to create an error correction code with the ability to both detect and localise errors, multiple stabilizer measurements need to be performed.

We now describe the three-qubit code, the natural extension of the two-qubit code in which the encoding operation distributes the quantum information across an entangled three-party state to give a logical state of the form  $|\psi\rangle_L = \alpha |000\rangle + \beta |111\rangle$ . This logical state occupies an eight-dimensional Hilbert space that can be partitioned into four two-dimensional subspaces as follows

$$\begin{aligned} \mathcal{C} &= \text{span}\{|000\rangle, |111\rangle\}, & \mathcal{F}_1 &= \text{span}\{|100\rangle, |110\rangle\}, \\ \mathcal{F}_2 &= \text{span}\{|010\rangle, |101\rangle\}, & \mathcal{F}_3 &= \text{span}\{|001\rangle, |110\rangle\}, \end{aligned} \quad (25)$$

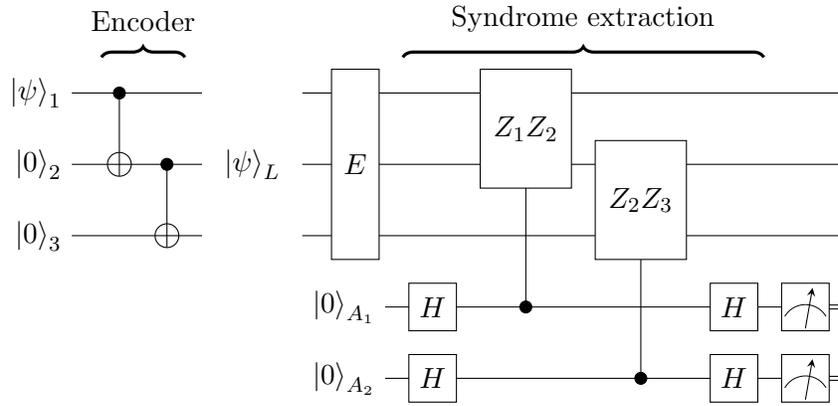


Figure 3. The circuit diagram of the three-qubit code. Encode stage: The information contained in a single qubit  $|\psi\rangle$  is entangled with two redundancy qubits  $|0\rangle_2$  and  $|0\rangle_3$  to create a logical qubit  $|\psi\rangle_L$ . The stabilizers  $Z_1Z_2$  and  $Z_2Z_3$  are measured on the logical qubit via two operations controlled on the ancilla qubits  $A_1$  and  $A_2$  respectively. The subsequent measurement of the ancilla qubits gives a two-bit syndrome  $S$ .

Table 2. The syndrome table for all bit-flip errors on the three qubit code. The syndrome  $S$  is a two-bit string formed by concatenating the results of the two stabilizer measurements.

Error	Syndrome, $S$	Error	Syndrome, $S$
$I_1I_2I_3$	00	$X_1X_2I_3$	01
$X_1I_2I_3$	10	$I_1X_2X_3$	10
$I_1X_2I_3$	11	$X_1I_2X_3$	11
$I_1I_2X_3$	01	$X_1X_2X_3$	00

where  $\mathcal{C}$  is the logical code space, and  $\mathcal{F}_{\{1,2,3\}}$  are the logical error spaces. We see that each single-qubit error from the set  $E = \{X_1, X_2, X_3\}$  will rotate the codespace to a unique error space so that  $X_i|\psi\rangle_L \in \mathcal{F}_i$ . In order to differentiate between these subspaces, we perform two stabilizer measurements  $Z_1Z_2$  and  $Z_2Z_3$  via the circuit shown in figure 3. The resultant syndrome table for single-qubit errors is given in table 2. From this we see that each single-qubit error produces a unique two-bit syndrome  $S = s_1s_2$ , enabling us to choose a suitable recovery operation.

### 3.2. Quantum code distance

As is the case for classical codes, the distance of a quantum code is defined as the minimum size error that will go undetected. Alternatively, this minimum size error can be viewed as a logical Pauli operator that transforms one codeword state to another. For the three-qubit code described in section 3.1, we see that the logical Pauli- $X$  operator is given by  $\bar{X} = X_1X_2X_3$ , so that

$$\bar{X}|0\rangle_L = |1\rangle_L \text{ and } \bar{X}|1\rangle_L = |0\rangle_L, \quad (26)$$

where  $|0\rangle_L = |000\rangle$  and  $|1\rangle_L = |111\rangle$  are the logical codewords for the three-qubit code. If it were the case that qubits were only susceptible to  $X$ -errors, then the three-qubit code would have distance  $d = 3$ . However, as qubits are also susceptible to phase-flip errors, it is also necessary to consider the logical Pauli- $Z$  operator  $\bar{Z}$  when determining the code distance. To do this, it is useful to switch

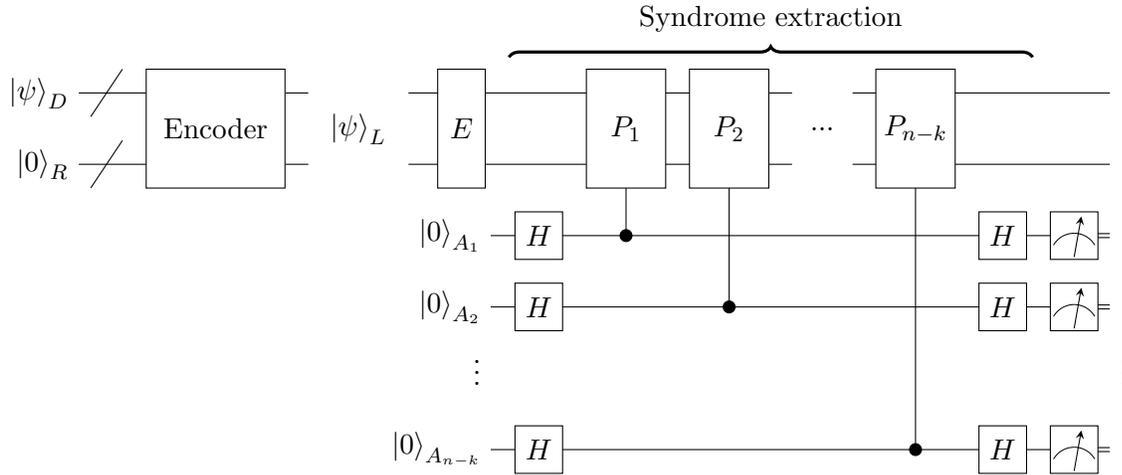


Figure 4. Circuit illustrating the structure of an  $[[n, k, d]]$  stabilizer code. A quantum data register  $|\psi\rangle_D = |\psi_1\psi_2\dots\psi_k\rangle$  is entangled with redundancy qubits  $|0\rangle_R = |0_10_2\dots0_{n-k}\rangle$  via an encoding operation to create a logical qubit  $|\psi\rangle_L$ . After encoding, a sequence of  $n - k$  stabilizer checks  $P_i$  are performed on the register, and each result copied to an ancilla qubit  $A_i$ . The subsequent measurement of the ancilla qubits provides an  $m$ -bit syndrome.

from the computational basis,  $\{|0\rangle, |1\rangle\}$ , to the conjugate basis,  $\{|+\rangle, |-\rangle\}$ , where we define

$$|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \quad \text{and} \quad |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \quad (27)$$

A  $Z$ -error maps the conjugate basis states as follows  $Z|+\rangle = |-\rangle$  and  $Z|-\rangle = |+\rangle$ . Now, encoding the conjugate basis states with the three-qubit code gives the logical states

$$|+\rangle_L = \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle) \quad \text{and} \quad |-\rangle_L = \frac{1}{\sqrt{2}}(|000\rangle - |111\rangle). \quad (28)$$

A weight-one logical Pauli- $Z$  operator  $\bar{Z} = Z_1$  will transform  $Z|+\rangle_L = |-\rangle_L$ , meaning the code is unable to detect the presence of single-qubit  $Z$ -errors. As a result, the three-qubit code has a quantum distance  $d = 1$ . In the next section, we outline the construction of general stabilizer codes capable of detecting both  $X$ - and  $Z$ -errors.

#### 4. Stabilizer codes

The three-qubit code works by de-localising the information in a single-qubit across three qubits. The resultant logical state is then encoded in a two-dimensional subspace (the codespace) of the expanded Hilbert space. The three-qubit code is designed such that if an  $X$ -error occurs, the logical state is rotated to an orthogonal error space, an event that can be detected via a sequence of two stabilizer measurements. This section describes how the procedure to can be generalised to create  $[[n, k, d]]$  stabilizer codes, where  $n$  is the total number of qubits,  $k$  is the number of logical qubits and  $d$  is the code distance. Note the use of double brackets to differentiate quantum codes from classical codes which are labelled with single brackets.

The circuit in figure 4 shows the basic structure of an  $[[n, k, d]]$  stabilizer code. A register of  $k$  data qubits,  $|\psi\rangle_D$ , is entangled with  $m = n - k$  redundancy qubits  $|0\rangle_R$  via an encoding operation to create a logical qubit  $|\psi\rangle_L$ . At this stage, the data previously stored solely in  $|\psi\rangle_D$  is distributed across the expanded Hilbert space. Errors can then be detected by performing  $m$  stabilizer measurements  $P_i$  as shown to the right of figure 4.

In the circuit in figure 4, each of the stabilizers is measured using the same syndrome extraction method that was used for the two-qubit code in figure 2. For each stabilizer  $P_i$ , the syndrome extraction circuit maps the logical state as follows

$$E|\psi\rangle_L|0\rangle_{A_i} \xrightarrow{\text{syndrome extraction}} \frac{1}{2}(\mathbb{1}^{\otimes n} + P_i)E|\psi\rangle_L|0\rangle_{A_i} + \frac{1}{2}(\mathbb{1}^{\otimes n} - P_i)E|\psi\rangle_L|1\rangle_{A_i}. \quad (29)$$

From the above, we see that if the stabilizer  $P_i$  commutes with an error  $E$  the measurement of ancilla qubit  $A_i$  returns ‘0’. If the stabilizer  $P_i$  anti-commutes with an error  $E$  the measurement returns ‘1’. The task of constructing a good code therefore involves finding stabilizers that anti-commute with the errors to be detected. In general, two Pauli operators will commute with one another if they intersect non-trivially on an even number of qubits, and anti-commute if otherwise. For specific examples of Pauli commutation relations, see appendix D.

The results of the  $m$  stabilizer measurements are combined to give an  $m$ -bit syndrome. For a well designed code, the syndrome allows us to deduce the best recovery operation to restore the logical state to the codespace.

#### 4.1. Properties of the code stabilizers

The stabilizers  $P_i$  of an  $[[n, k, d]]$  code must satisfy the following properties:

- (1) They must be Pauli-group elements,  $P_i \in \mathcal{G}_n$ . Here  $\mathcal{G}_n$  is the Pauli Group over  $n$ -qubits (see appendix B for the definition of the Pauli Group).
- (2) They must stabilize all logical states  $|\psi\rangle_L$  of the code. This means that each  $P_i$  has the action  $P_i|\psi\rangle_L = (+1)|\psi\rangle_L$  for all possible values of  $|\psi\rangle_L$ .
- (3) All the stabilizers of a code must commute with one another, so that  $[P_i, P_j] = 0$  for all  $i$  and  $j$ . This property is necessary so that the stabilizers can be measured simultaneously (or in a way independent of their ordering) as depicted in figure 4.

In the language of group theory, the stabilizers  $P_i$  of an  $[[n, k, d]]$  code form an Abelian subgroup  $\mathcal{S}$  of the Pauli Group. The stabilizer requirements listed above are incorporated into the definition of  $\mathcal{S}$  as follows

$$\mathcal{S} = \{P_i \in \mathcal{G}_n \mid P_i|\psi\rangle_L = (+1)|\psi\rangle_L \ \forall \ |\psi\rangle_L \wedge [P_i, P_j] = 0 \ \forall \ (i, j)\}. \quad (30)$$

An important point to note is that any product of the stabilizers  $P_i P_j$  will also be a stabilizer as  $P_i P_j |\psi\rangle_L = P_i (+1) |\psi\rangle_L = (+1) |\psi\rangle_L$ . Given this, it is important to ensure that the set of  $m = n - k$  stabilizers that are actually measured in the syndrome extraction process form a minimal set of the stabilizer group

$$\mathcal{S} = \langle G_1, G_2, \dots, G_m \rangle \quad (31)$$

In a minimal set it is not possible to obtain one stabilizer  $G_i$  as a product of any of the other elements  $G_j$ . As a simple example, consider the following set of stabilizers for the three-qubit code  $\mathcal{S} = \{Z_1 Z_2, Z_2 Z_3, Z_1 Z_3\}$ . This is not a minimal set, as it is possible to obtain the third stabilizer as a product of the first two. A possible minimal set is  $\mathcal{S} = \langle Z_1 Z_2, Z_2 Z_3 \rangle$ , which are the two stabilizers measured in the example in section 3.1.

#### 4.2. The logical operators of stabilizer codes

An  $[[n, k, d]]$  stabilizer code has  $2k$  logical Pauli operators that allow for logical states to be modified without having to decode then re-encode. For each logical qubit  $i$ , there is a logical Pauli- $X$  operator

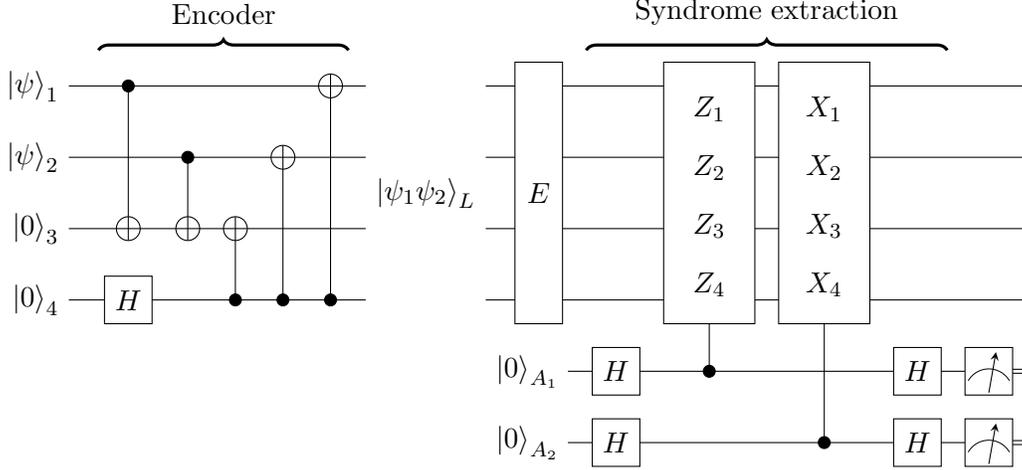


Figure 5. Circuit diagram for the four-qubit code. Encode stage: the information contained in two-qubit register  $|\psi\rangle_1 |\psi\rangle_2$  is distributed across two redundancy qubits,  $|0\rangle_3$  and  $|0\rangle_4$ , to create a logical state  $|\psi_1 \psi_2\rangle_L$  that encodes two qubits. Syndrome extraction stage: the code stabilizers,  $Z_1 Z_2 Z_3 Z_4$  and  $X_1 X_2 X_3 X_4$ , are measured on the code qubits and results copied to the ancilla qubits. The subsequent measurement of the ancilla qubits provides a two-bit syndrome  $S$  that informs of the occurrence of an error.

$\bar{X}_i$  and a logical Pauli- $Z$  operator  $\bar{Z}_i$ . Each pair of logical operators,  $\bar{X}_i$  and  $\bar{Z}_i$ , satisfy the following properties

- (1) They commute with all the code stabilizers in  $\mathcal{S}$ .
- (2) They anti-commute with one another, so that  $[\bar{X}_i, \bar{Z}_i]_+ = \bar{X}_i \bar{Z}_i + \bar{Z}_i \bar{X}_i = 0$  for all qubits  $i$ .

Any product of a logical operator  $\bar{L}_i$  and stabilizer  $P_j$  will also be a logical operator. This is clear from the fact that the stabilizer maps the logical state onto its (+1) eigenspace. Any product  $\bar{L}_i P_j$  therefore has the following action on the logical state  $\bar{L}_i P_j |\psi\rangle_L = \bar{L}_i |\psi\rangle_L$ .

### 4.3. Example: The $[[4,2,2]]$ detection code

The  $[[4,2,2]]$  detection code is the smallest stabilizer code to offer protection against a quantum noise model in which the qubits are susceptible to both  $X$ - and  $Z$ -errors [38,39]. As such, it provides a useful demonstration of the structure and properties of a stabilizer code.

An encoder for the  $[[4,2,2]]$  code is shown in figure 5. A two-qubit register  $|\psi\rangle_1 |\psi\rangle_2$  is entangled across four qubits to give the code state  $|\psi_1 \psi_2\rangle_L$ . As there are two encoded logical qubits in the  $[[4,2,2]]$  code, its codespace is four-dimensional and is spanned by

$$\mathcal{C}_{[[4,2,2]]} = \text{span} \left\{ \begin{array}{l} |00\rangle_L = \frac{1}{\sqrt{2}}(|0000\rangle + |1111\rangle) \\ |01\rangle_L = \frac{1}{\sqrt{2}}(|0110\rangle + |1001\rangle) \\ |10\rangle_L = \frac{1}{\sqrt{2}}(|1010\rangle + |0101\rangle) \\ |11\rangle_L = \frac{1}{\sqrt{2}}(|1100\rangle + |0011\rangle) \end{array} \right\}. \quad (32)$$

The stabilizers of the above logical basis states are  $\mathcal{S}_{[[4,2,2]]} = \langle X_1 X_2 X_3 X_4, Z_1 Z_2 Z_3 Z_4 \rangle$ . It is clear that these stabilizers commute with one another, as required by the definition in equation (30). These stabilizers can be measured using the syndrome extraction circuit shown to the right of figure 5. From equation (29), we know that for a syndrome measurement to be non-zero, the error  $E$  has to anti-commute with the stabilizer being measured. Considering first the single-qubit  $X$ -errors ( $E = \{X_1, X_2, X_3, X_4\}$ ), we see that they all anti-commute with the  $Z_1 Z_2 Z_3 Z_4$  stabilizer. Likewise, the single-qubit  $Z$ -errors ( $E = \{Z_1, Z_2, Z_3, Z_4\}$ ) anti-commute with the  $X_1 X_2 X_3 X_4$  stabilizer. Any

Table 3. The syndrome table for the  $[[4, 2, 2]]$  code for all single-qubit  $X$ -,  $Z$ - and  $Y$ -errors.

Error	Syndrome, $S$	Error	Syndrome, $S$	Error	Syndrome, $S$
$X_1$	10	$Z_1$	01	$Y_1$	11
$X_2$	10	$Z_2$	01	$Y_2$	11
$X_3$	10	$Z_3$	01	$Y_3$	11
$X_4$	10	$Z_4$	01	$Y_4$	11

single-qubit error on the  $[[4, 2, 2]]$  code will therefore trigger a non-zero syndrome. The syndrome table for all single-qubit errors in the  $[[4, 2, 2]]$  code is shown in table 3. For completeness, table 3 also includes the syndromes for single-qubit  $Y$ -errors, which are equivalent to the simultaneous occurrence of an  $X$ - and  $Z$ -error.

The  $[[4, 2, 2]]$  code has Pauli- $X$  and Pauli- $Z$  logical operators for each of its encoded logical qubits. A possible choice of these logical operators is given by

$$\mathcal{L}_{[[4,2,2]]} = \left\{ \begin{array}{l} \bar{X}_1 = X_1 X_3 \\ \bar{Z}_1 = Z_1 Z_4 \\ \bar{X}_2 = X_2 X_3 \\ \bar{Z}_2 = Z_2 Z_4 \end{array} \right\}. \quad (33)$$

Each logical operators commutes with the two stabilizers of the code so that  $[L_i, P_i] = 0$  for all  $L_i \in \mathcal{L}_{[[4,2,2]]}$  and  $P_i \in \mathcal{S}$ . Furthermore, it can be checked that the requirement  $[\bar{X}_i, \bar{Z}_i]_+ = 0$  is satisfied for each pair of logical operators. The minimum weight logical operator in  $\mathcal{L}_{[[4,2,2]]}$  is two, which sets the code distance to  $d = 2$ . As the distance of the  $[[4, 2, 2]]$  code is less than three, it is a detection code rather than a full correction code. In section 4.6, we introduce the Shor  $[[9, 1, 3]]$  code as an example of a code capable of both detecting and correcting errors.

#### 4.4. A general encoding circuit for stabilizer codes

The quantum codes presented this review have included bespoke encoding circuits to prepare the logical states. Special methods exist for constructing such circuits given a set of stabilizers [40,41]. In this section we describe a general method for preparing the logical states of stabilizer code using the same circuits that are used for syndrome extraction.

The  $|0\rangle_L$  codeword of any  $[[n, k, d]]$  stabilizer can be obtained via a projection onto the  $(+1)$  eigenspace of all of its stabilizers

$$|0\rangle_L = \frac{1}{N} \prod_{P_i \in \langle \mathcal{S} \rangle} (\mathbb{1}^{\otimes n} + P_i) |0^{\otimes n}\rangle, \quad (34)$$

where  $\langle \mathcal{S} \rangle$  is the minimal set of the code stabilizers and the  $1/N$  term is a factor that ensures normalisation. For example, the  $|00\rangle_L$  codeword of the four-qubit code defined in section 4.3 is given by

$$|00\rangle_L = \frac{1}{\sqrt{2}}(\mathbb{1}^{\otimes 4} + X_1 X_2 X_3 X_4)(\mathbb{1}^{\otimes 4} + Z_1 Z_2 Z_3 Z_4) |0000\rangle = \frac{1}{\sqrt{2}}(|0000\rangle + |1111\rangle). \quad (35)$$

The remaining codewords of the code can be obtained by applying logical operators to the  $|0\rangle_L$  codeword.

The  $|0\rangle_L$  codeword of any stabilizer code can be prepared via the projection in equation (34) by applying the general syndrome extraction circuit (shown on the right-hand-side of figure 4) to a  $|0\rangle^{\otimes n}$  state. As an example, consider the case where we apply the syndrome extraction circuit to the state  $|0\rangle^{\otimes 4}$  to prepare the  $|0\rangle_L$  codeword of the four-qubit code. The intermediary state immediately after the extraction of the  $X_1X_2X_3X_4$  stabilizer is given by

$$\frac{1}{2}(\mathbb{1}^{\otimes 4} + X_1X_2X_3X_4)|0000\rangle|0\rangle_A + \frac{1}{2}(\mathbb{1}^{\otimes 4} - X_1X_2X_3X_4)|0000\rangle|1\rangle_A. \quad (36)$$

When the ancilla is measured, the above state collapses to either the (+1) or (-1) projection with equal probability. In the case where the ‘1’ syndrome is measured, a correction needs to be applied to transform the state back onto the (+1) eigenspace of the stabilizer. Repeating this procedure for the remaining stabilizers leads to the preparation of the  $|0\rangle_L$  codeword.

#### 4.5. Quantum error correction with stabilizer codes

As is the case for classical codes, the distance of a quantum code is related to the number of correctable errors  $t$  via the relation  $d = 2t + 1$ . As a result, stabilizer codes with  $d \geq 3$  are error correction codes for which active recovery operations can be applied. In contrast, detection protocols such as the  $[[4, 2, 2]]$  code require a repeat-until-success approach.

Figure 6 shows the general error correction procedure for a single cycle of an  $[[n, k, d \geq 3]]$  stabilizer code. The encoded logical state  $|\psi\rangle_L$  is subject to an error process described by the circuit-element  $E$ . Next, the code stabilizers are measured (using the syndrome extraction method illustrated in figure 4), and the results copied to a register of  $m = n - k$  ancilla qubits  $|A\rangle^{\otimes m}$ . The ancilla qubits are then read out to give an  $m$ -bit syndrome  $S$ .

The next step in the error correction procedure is referred to as decoding, and involves processing the syndrome to determine the best unitary operation  $\mathcal{R}$  to return the logical state to the codespace. After this recovery operation has been applied, the output of the code-cycle is given by  $\mathcal{R}E|\psi\rangle_L \in \mathcal{C}_{[[n, k, d]]}$ . The decoding step is a success if the combined action of  $\mathcal{R}E$  on the code state is as follows

$$\mathcal{R}E|\psi\rangle_L = (+1)|\psi\rangle_L. \quad (37)$$

The above condition is trivially satisfied if  $\mathcal{R} = E^\dagger$  so that  $\mathcal{R}E = \mathbb{1}$ . However, this is not the only solution. Equation (37) is also satisfied for any product  $\mathcal{R}E$  that is an element of the code stabilizer such that  $\mathcal{R}E = P \in \mathcal{S}$ . In section 4.6, we will see that the fact that the solution for  $\mathcal{R}$  is not unique means it is possible to design *degenerate* quantum codes for which multiple errors can map to the same syndrome.

The decoding step fails if the recovery operation maps the code state as follows

$$\mathcal{R}E|\psi\rangle_L = L|\psi\rangle_L, \quad (38)$$

where  $L$  is a logical operator of the code. In this case, the state is returned to the codespace, but the recovery operation leads to a change in the encoded information.

#### 4.6. Example: The Shor $[[9, 1, 3]]$ code

The Shor nine-qubit code was the first quantum error correction scheme to be proposed. It is an example of a distance-three degenerate code for which it is possible to apply a successful recovery operation for any single-qubit error [21]. We now outline how the Shor code can be constructed via a method known as code concatenation.

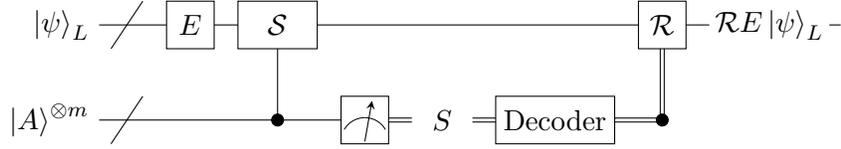


Figure 6. The general procedure for active recovery in a quantum error correction code. The logical qubit  $|\psi\rangle_L$  of an  $[[n, k, d]]$  stabilizer code is subject to an error process  $E$ . A generating set of stabilizers  $\mathcal{S}$  are measured on the logical state to yield an  $m$ -bit syndrome  $S$ . This syndrome is processed by a decoder to determine the best recovery operation  $\mathcal{R}$  to return the logical state to the codespace. After the recovery has been applied, the output of the error correction cycle is  $\mathcal{R}E|\psi\rangle_L$ . Double lines indicate classical information flow.

Code concatenation involves embedding the output of one code into the input of another. In the construction of the Shor nine-qubit code, the two codes that are concatenated are the three-qubit code for bit-flips and the three-qubit code for phase-flips [32]. The three-qubit code for bit-flips  $\mathcal{C}_{3b}$  was described in section 3.1 and is defined as follows

$$\mathcal{C}_{3b} = \text{span}\{|0\rangle_{3b} = |000\rangle, |1\rangle_{3b} = |111\rangle\}, \quad \mathcal{S}_{3b} = \langle Z_1 Z_2, Z_2 Z_3 \rangle, \quad (39)$$

where  $\mathcal{S}_{3b}$  are the code stabilizers. Similarly, the three-qubit code for phase-flips  $\mathcal{C}_{3p}$  is defined

$$\mathcal{C}_{3p} = \text{span}\{|0\rangle_{3p} = |+++ \rangle, |1\rangle_{3p} = |-- \rangle\}, \quad \mathcal{S}_{3p} = \langle X_1 X_2, X_2 X_3 \rangle, \quad (40)$$

To construct the nine-qubit code, the bit-flip code is embedded into the codewords of the phase-flip code. This concatenation maps the  $|0\rangle_{3p}$  codeword of the phase-flip code to a nine-qubit codeword  $|0\rangle_9$  as follows

$$|0\rangle_{3p} = |+++ \rangle \xrightarrow{\text{concatenation}} |0\rangle_9 = |+\rangle_{3b} |+\rangle_{3b} |+\rangle_{3b}, \quad (41)$$

where  $|+\rangle_{3b} = \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$  is a logical state of the bit-flip code. Similarly, the concatenation maps the  $|1\rangle_{3p}$  codeword of the phase-flip code to

$$|1\rangle_{3p} = |-- \rangle \xrightarrow{\text{concatenation}} |1\rangle_9 = |-\rangle_{3b} |-\rangle_{3b} |-\rangle_{3b}, \quad (42)$$

where  $|-\rangle_{3b} = \frac{1}{\sqrt{2}}(|000\rangle - |111\rangle)$ . The code defined by the codewords  $|0\rangle_9$  and  $|1\rangle_9$  is the nine-qubit Shor code with parameters  $[[9, 1, 3]]$ . Rewriting the right-hand-sides of Equations 41 and 42 in the computational basis, we get the following codespace for the Shor code

$$\mathcal{C}_{[[9,1,3]]} = \text{span} \left\{ \begin{array}{l} |0\rangle_9 = \frac{1}{\sqrt{8}}(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle) \\ |1\rangle_9 = \frac{1}{\sqrt{8}}(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle) \end{array} \right\}. \quad (43)$$

The stabilizers of the above code are given by

$$\mathcal{S}_{[[9,3,3]]} = \langle Z_1 Z_2, Z_2 Z_3, Z_4 Z_5, Z_5 Z_6, Z_7 Z_8, Z_8 Z_9, \\ X_1 X_2 X_3 X_4 X_5 X_6, X_4 X_5 X_6 X_7 X_8 X_9 \rangle. \quad (44)$$

Table 4. The syndrome table for single-qubit  $X$ - and  $Z$ -errors on the nine-qubit code. The nine-qubit code is a degenerate code, as certain  $Z$ -errors share the same syndrome.

Error	Syndrome, $S$	Error	Syndrome, $S$
$X_1$	10000000	$Z_1$	00000010
$X_2$	11000000	$Z_2$	00000010
$X_3$	01000000	$Z_3$	00000010
$X_4$	00100000	$Z_4$	00000011
$X_5$	00110000	$Z_5$	00000011
$X_6$	00010000	$Z_6$	00000011
$X_7$	00001000	$Z_7$	00000001
$X_8$	00001100	$Z_8$	00000001
$X_9$	00000100	$Z_9$	00000001

The first six terms are the stabilizers of the bit-flip codes in the three-blocks of the code. The final two stabilizers derive from the stabilizers of the phase-flip code.

Table 4 shows the syndromes for all single-qubit errors in the nine-qubit code. Each of the  $X$ -errors produce unique syndromes. In contrast,  $Z$ -errors that occur in the same block of the code have the same syndrome. Fortunately, this degeneracy in the code syndromes does not reduce the code distance. To see why this is the case, consider the single-qubit errors  $Z_1$  and  $Z_2$ , both of which map to the syndrome ‘00000010’. The decoder therefore has insufficient information to differentiate between the two errors, and will output the same recovery operation for either. For the purposes of this example, we will assume that the recovery operation the decoder outputs is  $\mathcal{R} = Z_1$ . For the case where the error is  $E = Z_1$ , the recovery operation trivially restores the logical state as  $\mathcal{R}E|\psi\rangle_9 = Z_1Z_1|\psi\rangle_9 = |\psi\rangle_9$ . In the event where  $E = Z_2$ , the recovery operation still restores the logical state as  $\mathcal{R}E = Z_1Z_2$  is in the stabilizer of  $\mathcal{C}_{[[9,1,3]]}$ , and therefore acts on the logical state as follows  $Z_1Z_2|\psi\rangle_9 = |\psi\rangle_9$ . The same arguments can be applied to the remaining degenerate errors of the code. As a result, the nine-qubit code has the ability to correct all single-qubit errors and has distance  $d = 3$ .

## 5. The surface code

The challenge in creating quantum error correction codes lies in finding commuting sets of stabilizers that enable errors to be detected without disturbing the encoded information. Finding such sets is non-trivial, and special code constructions are required to find stabilizers with the desired properties. In section 4.6 we saw how a code can be constructed by concatenating two smaller codes. Other constructions include methods for repurposing classical codes to obtain commuting stabilizer checks [42–45]. In this section, we outline a construction known as the surface code [46,47].

The realisation of a surface code logical qubit is key goal for many quantum computing hardware efforts [48–52]. Surface codes belong to a broader family of so-called *topological* codes [53]. The general design principle behind topological codes is that the code is built up by ‘patching’ together repeated elements. We will see that this modular approach ensures that the surface code can be straight-forwardly scaled in size whilst ensuring stabilizer commutativity. In terms of actual implementation, the specific advantage of surface code for current hardware platforms is that it requires only nearest-neighbour interactions. This is advantageous as many quantum computing platforms are unable to perform high-fidelity long-range interactions between qubits.

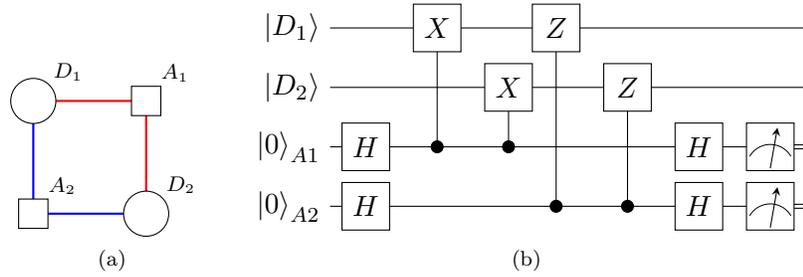


Figure 7. The surface code four-cycle. (a) Pictorial representation. The code qubits,  $D_1$  and  $D_2$ , are represented by the circular nodes. The ancilla qubits,  $A_1$  and  $A_2$ , are represented by the square nodes. The red and blue edges depict controlled- $X$  and controlled- $Z$  operations controlled on the ancilla qubits and acting on the code qubits. (b) An equivalent surface code four-cycle in circuit notation.

### 5.1. The surface code four-cycle

For surface codes it is beneficial to adopt a pictorial representation of the code qubits in place of the circuit notation we have used up to this point. Figure 7a shows a surface code four-cycle, the fundamental building block around which surface codes are constructed. The circles in figure 7a represent the code qubits and the squares the ancilla qubits. The red edges represent controlled- $X$  gates, each controlled on an ancilla qubit  $A$  and acting on a data qubit  $D$ . Likewise, the blue edges represent controlled- $Z$  operations, each controlled by an ancilla qubit and acting on a data qubit. These controlled operations are the gates with which the stabilizers of the four-cycle are measured. Ancilla qubit  $A_1$  connects to data qubits  $D_1$  and  $D_2$  via red edges, and therefore measures the stabilizer  $X_{D_1}X_{D_2}$ . Likewise, ancilla qubit  $A_2$  measures the stabilizer  $Z_{D_1}Z_{D_2}$ . For comparison, the four-cycle is shown in quantum circuit notation in figure 7b.

The stabilizers of the four-cycle,  $X_{D_1}X_{D_2}$  and  $Z_{D_1}Z_{D_2}$ , commute with one another as they intersect non-trivially on an even number of code qubits. This can easily be verified by inspection of figure 7b.

The  $|0\rangle_L$  codeword of the four-cycle can be prepared by setting the initial state of the code qubits to  $|D_1D_2\rangle = |00\rangle$ , and following the general encoding procedure outlined in section 4.4. However, as the four-cycle has two code qubits  $n = 2$  and two stabilizers  $m = 2$ , the number of logical qubits it encodes is equal to  $k = n - m = 0$ . As a result, the four-cycle is not in itself a useful code. However, we will see that working detection and correction codes can be formed by tiling together multiple four-cycles to form square lattices.

### 5.2. The $[[5, 1, 2]]$ surface code

Figure 8a shows the five-qubit surface code formed by tiling together four four-cycles in a square lattice [54]. By inspecting which data qubits each ancilla qubit connects to, the stabilizers of the code in figure 8 can be read off to give

$$\mathcal{S}_{[[5,1,2]]} = \langle X_{D_1}X_{D_2}X_{D_3}, Z_{D_1}Z_{D_3}Z_{D_4}, Z_{D_2}Z_{D_3}Z_{D_5}, X_{D_3}X_{D_4}X_{D_5} \rangle. \quad (45)$$

The first term in the above is the stabilizer measured by ancilla qubit  $A_1$ , the second by ancilla  $A_2$  etc. The stabilizers in  $\mathcal{S}_{[[5,1,2]]}$  commute with one another, as the  $X$ - and  $Z$ -type stabilizers all intersect on an even number of code qubits. From figure 8, we see that there are five code qubits and four stabilizers meaning the code encodes one logical qubit.

Figure 7b shows two examples of errors on the surface code and how they are detected. The  $Z_{D_1}$ -error on qubit  $D_1$  anti-commutes with the  $X_{D_1}X_{D_2}X_{D_3}$  stabilizer, and therefore triggers a ‘1’ syndrome. This is depicted by the red filling in the ancilla qubit  $A_1$ . Likewise, the  $X_{D_5}$ -error anti-commutes with the  $Z_{D_2}Z_{D_3}Z_{D_5}$  stabilizer and triggers a ‘1’ syndrome measurement in ancilla

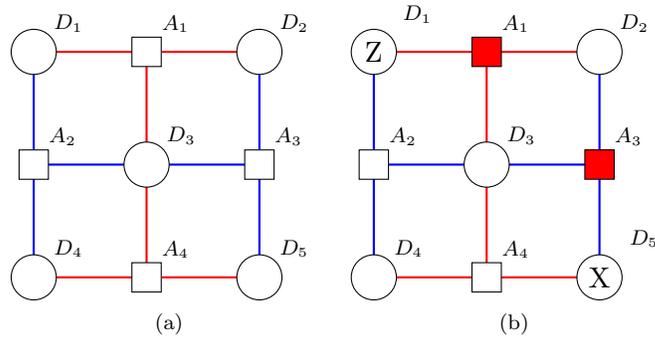


Figure 8. (a) The  $[[5, 1, 2]]$  surface code formed by tiling together four four-cycles in a square lattice. (b) Examples of error detection in the  $[[5, 1, 2]]$  surface code. The  $Z_{D_1}$  error on qubit  $D_1$  anti-commutes with the stabilizer measured by ancilla qubit  $A_1$ . The  $A_1$  qubit is coloured red to indicate it will be measured as a ‘1’. Likewise, the  $X_{D_5}$  error on qubit  $D_5$  is detected by the stabilizer measured by ancilla qubit  $A_3$ .

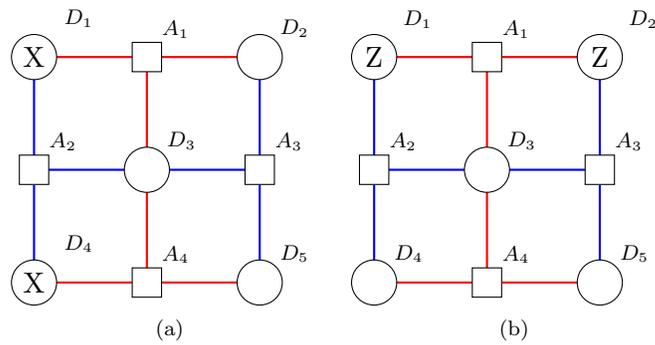


Figure 9. The logical operators of a surface code can be defined as chains of Pauli operations that act along the boundaries of the lattice. (a) The Pauli- $X$  logical operator  $\bar{X} = X_{D_1} X_{D_4}$  acts along the boundary along which  $Z$ -type stabilizers are measured. (b) The Pauli- $Z$  logical operator  $\bar{Z} = Z_{D_1} Z_{D_2}$  acts along the boundary along which  $X$ -type stabilizers are measured. The two logical operators anti-commute with one another.

qubit  $A_4$ .

From figure 7 it can be seen that the surface code is a square lattice with two types of boundaries. The vertical boundaries are formed of blue edges representing  $Z$ -type stabilizer measurements. The horizontal boundaries are formed of red-edges representing  $X$ -type stabilizer measurements. The logical operators of the surface code can be defined as chains of Pauli operators along the edges of these boundaries.

Figure 9a shows a two-qubit Pauli chain  $X_{D_1} X_{D_4}$  along the left-hand boundary of the five-qubit surface code. The  $X_{D_1} X_{D_4}$  operator commutes with all the stabilizers in  $\mathcal{S}_{[[5,1,2]]}$ , in particular the stabilizer  $Z_{D_1} Z_{D_3} Z_{D_4}$  with which it shares two qubits. Similarly, figure 9b shows an operator  $Z_{D_1} Z_{D_2}$  which acts across the top of the lattice. It can easily be checked that this operator also commutes with all the code stabilizers. Finally, we note that the operators  $X_{D_1} X_{D_4}$  and  $Z_{D_1} Z_{D_2}$  anti-commute. As outlined in section 4.2, the Pauli- $X$  and Pauli- $Z$  logical operators for each encoded qubit are pairs of operators that commute with all the code stabilizers but anti-commute with one another. A suitable choice for the logical operators of the  $[[5, 1, 2]]$  surface code would therefore be

$$\bar{X} = X_{D_1} X_{D_4} \text{ and } \bar{Z} = Z_{D_1} Z_{D_2}. \quad (46)$$

From the above we see that the minimum weight of the logical operators is 2, meaning the  $[[5, 1, 2]]$  code is a detection code with  $d = 2$ .

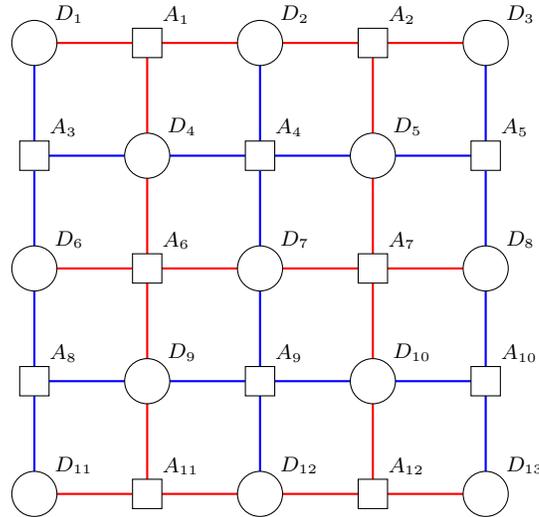


Figure 10. A distance-three surface code with parameters  $[[13, 1, 3]]$ . A possible choice for the logical operators of this code would be  $\bar{X} = X_{D_1}X_{D_6}X_{D_{11}}$  and  $\bar{Z} = Z_{D_1}Z_{D_2}Z_{D_3}$ .

### 5.3. Scaling the surface code

The distance of a surface code can be increased simply by scaling the size of lattice. In general, a surface code with distance  $d = \lambda$  will encode a *single* logical qubit and have code parameters given by

$$[[n = \lambda^2 + (\lambda - 1)^2, k = 1, d = \lambda]]. \quad (47)$$

For example, the distance-three  $[[13, 1, 3]]$  surface code is depicted in figure 10. The Pauli- $X$  logical operator of a surface code can be defined as a chain of  $X$ -Pauli operators along the boundary of the code along which the  $Z$ -stabilizers are applied (the blue boundary in our pictorial representation). Likewise, the  $Z$ -Pauli logical operator can be defined as a chain of  $Z$ -operators across the adjacent boundary along which the  $X$ -type stabilizers are applied (the red edges in our pictorial representation). For the distance-three code, a choice of logical operators would be  $\bar{X} = X_{D_1}X_{D_6}X_{D_{11}}$  and  $\bar{Z} = Z_{D_1}Z_{D_2}Z_{D_3}$ . The  $[[13, 1, 3]]$  code is the smallest surface code capable of detecting and correcting errors.

## 6. Practical considerations for quantum error correction

Up to this point, we have described stabilizer codes in an idealised, theoretical setting. In this section, we outline some of the practical issues that arise when considering the implementation of quantum error correction codes on actual hardware.

### 6.1. Efficient decoding algorithms

Given a code syndrome  $S$ , the role of the decoder is to find the best recovery operation  $\mathcal{R}$  to restore the encoded quantum information to the codespace. Measuring the stabilizers of an  $[[n, k, d]]$  code will produce an  $m$ -bit syndrome where  $m = n - k$ . As a result, there are  $2^m$  possible syndromes for each code. For the small code examples described in this review, it is possible to compute lookup-tables that exhaustively list the best recovery operation for each of the  $2^m$  syndromes. However, such a decoding strategy rapidly becomes impractical as the code size is increased. As

an example, consider the distance-five surface code which has parameters  $[[41, 1, 5]]$ . This code produces syndromes of length  $m = 40$ , and would therefore need a lookup table of size  $2^{40} \approx 10^{12}$ .

In place of lookup tables, large-scale quantum error correction codes use approximate inference techniques to determine the most likely error to have occurred given a certain syndrome  $S$ . Such methods allow for recovery operations to be chosen and applied in real-time between successive stabilizer code cycles. Unfortunately, there is no known universal decoder that can be efficiently applied to all quantum error correction codes. Instead, bespoke decoding algorithms need to be developed that are designed for specific code constructions. For surface codes, a technique known as minimum weight perfect matching (MWPM) can be used for decoding, which works by identifying error chains between positive syndrome measurements [55,56].

As outlined in section 4.5, the decode stage of an error correction cycle fails when  $\mathcal{R}E = L$ , where  $\mathcal{R}$  is the recovery operation output by the decoder,  $E$  is the error and  $L$  is a logical operator of the code. The frequency with which the decoder fails in this way gives a logical error rate  $p_L$ . As decoding algorithms are based on approximate inference techniques, some perform better than others. As such, the logical error rate of a quantum error correction code will depend heavily on the decoder used. The logical error rate can be determined by simulating stabilizer code cycles with errors sampled from a noise model. The specifics of the noise model are motivated by the physical device on which the code is to be run.

## 6.2. Code thresholds

A code construction provides a method for building a set of codes with a shared underlying structure. An example are the surface codes, for which the code distance can be increased by expanding the size of the qubit lattice. Given the increase in qubit overhead, scaling the code in this way is only ‘worthwhile’ if the resultant larger code has a lower logical error rate.

The *threshold theorem* for stabilizer codes states that increasing the distance of a code will result in a corresponding reduction in the logical error rate  $p_L$ , provided the physical error rate  $p$  of the individual code qubits is below a threshold  $p < p_{th}$ . The significance of this theorem is that it means that quantum error correction codes can in principle be used to arbitrarily suppress the logical error rate [22–26]. Conversely, if the physical error rate is above the threshold, the process of quantum encoding becomes self defeating. The threshold of a code therefore provides a minimum experimental benchmark that quantum computing experiments must reach before quantum error correction becomes viable.

Upper bounds on the threshold  $p_{th}$  for a code under a given noise model can be obtained using methods from statistical mechanics. Alternatively, more realistic thresholds can be numerically estimated by simulating code cycles and decoding using efficient inference algorithms as discussed in section 6.1. For the surface code, assuming  $X$ - and  $Z$ -errors are treated independently, the upper bound on the threshold is  $\approx 10.9\%$  [57]. In practice, decoders based on the MWPM algorithm can achieve thresholds as high as  $\approx 10.3\%$  [58].

## 6.3. Fault tolerance

In the discussion of quantum error correction codes so far, we have assumed that errors only occur in certain locations in the circuit. For example, in the circuit diagram for the two-qubit code shown in figure 2, errors are restricted to the region labelled ‘ $E$ ’. In doing this, we assume that all of the apparatus associated with encoding and syndrome extraction operates without error. However, in practice this is not the case. In fact, for many quantum computing technologies two-qubit gates, as well as measurement operations, can be dominant sources of error. As such, it is unrealistic to assume that any part of the circuit is error free.

A quantum error correction code is said to be *fault tolerant* if it can account for errors (of size

up to the code distance) that occur at any location in the circuit [26,59]. Various techniques exist for modifying quantum circuits to make them fault tolerant [60–62]. In the simplest terms, these methods ensure that small sub-distance errors do not spread uncontrollably through the circuit.

Modifying a quantum error correction circuit for fault tolerance can add considerable overhead in terms of the total number of additional ancilla qubits required. For example, a fault tolerant syndrome extraction procedure proposed by Shor requires  $\lambda$  ancilla qubits to measure each stabilizer, where  $\lambda$  is the number of non-identity elements in the stabilizer [60]. Under this scheme, eight ancilla qubits would be required to measure the two stabilizers of the four-qubit code depicted in figure 5. More efficient schemes exist [63], but a fault tolerant version of a code will always have increased overhead relative to the original circuit.

For a quantum circuit with noisy ancilla measurements, it is not always possible to decode the error correction code in a single round of syndrome extraction. To illustrate this, consider the case where the  $S = 10$  syndrome is measured in the three-qubit code outlined in section 3.1. From table 2, we see that this syndrome is triggered by an  $X_1$  error. However, if the ancillas themselves are subject to error, then the same syndrome could equally have resulted from an error on ancilla qubit  $A_1$ . To differentiate between these two possibilities, it is necessary to perform two (or more) rounds of stabilizer measurements and compare the syndromes over time. It should be noted that decoding over time in this way, in addition to any other modifications required for fault tolerance, will reduce the threshold for the code. For example, the threshold for the surface code with noisy ancilla measurements is  $\approx 1\%$ , compared to  $\approx 10\%$  in the ideal case [34,64].

#### 6.4. Encoded computation

A *universal quantum computer* is a device that can perform any unitary operation  $U$  that evolves a qubit register from one state to another  $U|\psi\rangle = |\psi'\rangle$ . It has been shown that any such operation  $U$  can be efficiently compiled from a finite set of elementary gates [65]. An example of a *universal gate set* is  $\langle \mathcal{U} \rangle = \langle X, Z, Y, H, \text{CNOT}, T \rangle$ , where  $T = \text{diag}(1, e^{i\pi/4})$ .

In this review, we have seen how Pauli logical  $\bar{X}$  and  $\bar{Z}$  operators can be defined for a variety of stabilizer codes. These gates allow computation to be performed directly on the encoded logical states, removing the need to decode then re-encode every time the state is to be evolved. However, the  $\bar{X}$  and  $\bar{Z}$  logical gates do not alone form a universal set, and additional logical operators need to be defined to achieve arbitrary computation on encoded states.

The major challenge in constructing a universal encoded gate set is to find ways in which the relevant gates can be performed fault tolerantly. For many codes, it is possible to fault tolerantly implement a subset of the gates in  $\langle \mathcal{U} \rangle$  without having to introduce additional qubits. This is achieved by defining the logical operators with a property known as *transversality* that guarantees errors will not spread uncontrollably through the circuit. However, a no-go theorem exists that prohibits the implementation of a full universal gate set in this way on a quantum computer [66]. As such, alternative techniques are required to perform universal encoded logic. Various methods have been proposed [67–70], but these typically impose a high cost in terms of the number of additional qubits required. To put this into context, it has been proposed that the surface code could realise a universal gate set using a method called magic state injection [67]. However, estimates suggest that the fault tolerant implementation of this technique could result in an order-of-magnitude increase in the total number of qubits required in the quantum computer [33].

#### 6.5. Experimental implementations of quantum error correction

The realisation of the first fault tolerant logical qubit will mark an important milestone in the journey to build a quantum computer. To this end, laboratories at places such as Google [49], IBM Research [71,72] and TU Delft [73] are currently building superconducting devices with the long-

term goal of realising a surface code logical qubit. Other efforts are currently underway pursuing qubit architectures based on ion-trap trap technology [48] and quantum optics [6].

The threshold for the surface code under a realistic noise assumptions is approximately 1% [64]. State-of-the-art qubit hardware has already been demonstrated with error rates below this level [11,74]. However, suppressing the logical error rate to the point where the logical qubit outperforms an un-encoded qubit will require levels of scalability that are not yet possible with current experiments. It is predicted that the first fault tolerant surface code logical qubits will require a lattice with over a thousand qubits [34]. To put the scale of the challenge that remains into context, the largest quantum computers to date have less than one hundred qubits. Furthermore, achieving this goal will only be the first step: a quantum computer with only a single logical qubit will be no more powerful than an abacus with one bead. In fact, it is currently estimated that a fault tolerant surface code quantum computer with the ability to outperform a classical device for a *useful* task will require over a million qubits in total [33,75].

The first quantum protocols to achieve fault tolerance will likely be quantum detection codes. As the smallest code capable of protecting against a quantum error model, the  $[[4, 2, 2]]$  code is a promising candidate. Several proof-of-concept implementations of the  $[[4, 2, 2]]$  code have already been demonstrated in [41,76–78]. Repetition codes (from the same family as the two- and three-qubit codes outlined in this review) have also been implemented on qubit hardware [79]. Over the past couple of years, several quantum computing hardware projects have developed cloud platforms to allow the public to program their devices. This makes it possible for interested readers to test some of the early proof-of-concept quantum codes. For example, a tutorial showing how a repetition code can be implemented on the IBM Q device can be found in the supplementary material of [79].

## 7. Outlook & Summary

A major hurdle in the realisation of a full-scale quantum computer stems from the challenge of controlling qubits in an error free way. Quantum error correction protocols offer a solution to this problem, in principle allowing for arbitrary suppression of the logical error rate provided certain threshold conditions on the physical qubits are met. However, there is a trade-off: quantum error correction protocols require large number of qubits to operate effectively. This will significantly increase the overheads associated with quantum computing.

Developing quantum codes is not straightforward. Complications arise due the no-cloning theorem, the problem of wavefunction collapse and necessity to deal with multiple error types. Stabilizer codes provide a formalism that allow quantum error correction codes to be constructed within these constraints. For stabilizer codes, quantum redundancy is achieved by entangling the quantum information in the initial register across an expanded space of qubits. Errors can then be detected by performing a series of projective stabilizer measurements, and the results interpreted to determine the best recovery operation to restore the quantum information to its intended state.

The surface code is currently the most widely pursued quantum error correction scheme for experiment. This is due to its comparatively high threshold combined with the fact it requires only nearest-neighbour interactions. However, there are drawbacks to the surface code, most notably its poor encoding density. The distance of the surface code can be increased simply by scaling the size of the qubit lattice, but this results in a vanishing code rate, where the rate is defined as the ratio of encoded qubits to physical qubits  $R = k/n$ . Another disadvantage to the surface code is that resource intensive methods are required to obtain a universal encoded gate set.

Alternatives to the surface code have been proposed based on different tilings of the qubit lattice [80], as well as extensions to higher dimensions [70,81]. These constructions typically have lower thresholds, but offer other advantages such as (potentially) easier access to universal encoded gate sets [70]. Efforts are also in progress to develop code constructions with non-vanishing rates based on principles from high-performance classical codes [45,59]. However, for these codes, it is often

necessary to perform arbitrary long range interactions between the code qubits.

In the quest to build a circuit model quantum computer there are many challenges to be overcome. At the hardware level, methods for the realisation and control of qubits need to be improved. In addition to this, a major theoretical challenge lies in finding better ways to achieve fault tolerant error correction. These two problems will have to be approached in parallel, with advances in either influencing the direction of the other.

## Acknowledgement(s)

JR acknowledges the support of the QCDA project which has received funding from the QuantERA ERA-NET Cofund in Quantum Technologies implemented within the European Unions Horizon 2020 Programme. Many thanks to Viv Kendon for useful discussions and help preparing and checking the manuscript. Special thanks to Benjamin Jones and Armanda Ottaviano Quintavalle for reading through early versions of the manuscript and providing valuable suggestions for improvement. Thanks also to Earl Campbell and Yingkai Ouyang for useful discussions, as well as to Jasminder Sidhu for helping with tikz drawings. The quantum circuit diagrams in this review were drawn using the QPIC package [82].

## Notes on contributor(s)

Joschka Roffe studied MPhys physics at The University Manchester, graduating in 2015. Following this, he studied a PhD in Quantum Computing at Durham University under the supervision of Viv Kendon. He now works as a research associate at the The University of Sheffield as part of the Quantum Codes Designs and Architectures (QCDA) project.

## References

- [1] Shor PW. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*. 1997;26(5):1484–1509.
- [2] Grover LK. A fast quantum mechanical algorithm for database search. In: *STOC*; 1996.
- [3] Schuld M, Sinayskiy I, Petruccione F. An introduction to quantum machine learning. *Contemporary Physics*. 2014;56(2):172–185. Available from: <https://doi.org/10.1080/00107514.2014.964942>.
- [4] Aspuru-Guzik A, Dutoi AD, Love PJ, et al. Simulated quantum computation of molecular energies. *Science*. 2005;309(5741):1704–1707. Available from: <https://science.sciencemag.org/content/309/5741/1704>.
- [5] Wang XL, Chen LK, Li W, et al. Experimental ten-photon entanglement. *Physical Review Letters*. 2016;117(21). Available from: <https://doi.org/10.1103/physrevlett.117.210502>.
- [6] Qiang X, Zhou X, Wang J, et al. Large-scale silicon quantum photonics implementing arbitrary two-qubit processing. *Nature Photonics*. 2018;12(9):534–539. Available from: <https://doi.org/10.1038/s41566-018-0236-y>.
- [7] Randall J, Weidt S, Standing ED, et al. Efficient preparation and detection of microwave dressed-state qubits and qutrits with trapped ions. *Physical Review A*. 2015;91(1).
- [8] Ballance C, Harty T, Linke N, et al. High-fidelity quantum logic gates using trapped-ion hyperfine qubits. *Physical Review Letters*. 2016;117(6).
- [9] Brandl MF, van Mourik MW, Postler L, et al. Cryogenic setup for trapped ion quantum computing. *Review of Scientific Instruments*. 2016;87(11):113103.
- [10] Debnath S, Linke NM, Figgatt C, et al. Demonstration of a small programmable quantum computer with atomic qubits. *Nature*. 2016;536(7614):63.
- [11] Chow JM, Gambetta JM, Córcoles AD, et al. Universal quantum gate set approaching fault-

- tolerant thresholds with superconducting qubits. *Physical Review Letters*. 2012;109(6). Available from: <https://doi.org/10.1103/physrevlett.109.060501>.
- [12] Chen Y, Neill C, Roushan P, et al. Qubit architecture with high coherence and fast tunable coupling. *Phys Rev Lett*. 2014;113:220502. Available from: <https://link.aps.org/doi/10.1103/PhysRevLett.113.220502>.
- [13] Wendin G. Quantum information processing with superconducting circuits: a review. *Reports on Progress in Physics*. 2017;80(10):106001. Available from: <https://doi.org/10.1088/1361-6633/aa7e1a>.
- [14] Kane BE. A silicon-based nuclear spin quantum computer. *Nature*. 1998;393(6681):133–137. Available from: <https://doi.org/10.1038/30156>.
- [15] Hill CD, Peretz E, Hile SJ, et al. A surface code quantum computer in silicon. *Science Advances*. 2015; 1(9):e1500707. Available from: <https://doi.org/10.1126/sciadv.1500707>.
- [16] van der Heijden J, Kobayashi T, House MG, et al. Readout and control of the spin-orbit states of two coupled acceptor atoms in a silicon transistor. *Science Advances*. 2018;4(12):eaat9199. Available from: <https://doi.org/10.1126/sciadv.aat9199>.
- [17] Shannon CE. A mathematical theory of communication. University of Illinois Press; 1949.
- [18] Hamming RW. Error detecting and error correcting codes. *Bell System Technical Journal*. 1950; 29(2):147–160. Available from: <https://doi.org/10.1002/j.1538-7305.1950.tb00463.x>.
- [19] MacKay DJ. Information theory, inference and learning algorithms. Cambridge University Press; 2003.
- [20] Wootters WK, Zurek WH. A single quantum cannot be cloned. *Nature*. 1982;299(5886):802–803.
- [21] Shor PW. Scheme for reducing decoherence in quantum computer memory. *Phys Rev A*. 1995;52:R2493.
- [22] Preskill J. Reliable quantum computers. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*. 1998;454(1969):385–410. Available from: <http://rspa.royalsocietypublishing.org/content/454/1969/385>.
- [23] Kitaev AY. Quantum computations: algorithms and error correction. *Russian Mathematical Surveys*. 1997;52(6):1191–1249. Available from: <https://doi.org/10.1070/rm1997v052n06abeh002155>.
- [24] Aharonov D, Ben-Or M. Fault-tolerant quantum computation with constant error. In: *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*; New York, NY, USA. ACM; 1997. p. 176–188; STOC '97. Available from: <http://doi.acm.org/10.1145/258533.258579>.
- [25] Knill E. Resilient quantum computation. *Science*. 1998;279(5349):342–345. Available from: <https://doi.org/10.1126/science.279.5349.342>.
- [26] Gottesman D. Theory of fault-tolerant quantum computation. *Physical Review A*. 1998;57(1):127.
- [27] Gaitan F. Quantum error correction and fault tolerant quantum computing. CRC Press; 2008.
- [28] Gottesman D. An introduction to quantum error correction and fault-tolerant quantum computation. arXiv:09042557. 2009;.
- [29] Quantum information processing and quantum error correction: An engineering approach. Elsevier; 2012. Available from: <https://doi.org/10.1016/c2010-0-66917-3>.
- [30] Devitt SJ, Munro WJ, Nemoto K. Quantum error correction for beginners. *Reports on Progress in Physics*. 2013;76(7):076001. Available from: <https://doi.org/10.1088/0034-4885/76/7/076001>.
- [31] Lidar D, Brun T. Quantum error correction. Cambridge University Press; 2013.
- [32] Terhal BM. Quantum error correction for quantum memories. *Reviews of Modern Physics*. 2015; 87(2):307–346. Available from: <https://doi.org/10.1103/revmodphys.87.307>.
- [33] Campbell ET, Terhal BM, Vuillot C. Roads towards fault-tolerant universal quantum computation. *Nature*. 2017;549(7671):172–179. Available from: <https://doi.org/10.1038/nature23460>.
- [34] Fowler AG, Mariantoni M, Martinis JM, et al. Surface codes: Towards practical large-scale quantum computation. *Physical Review A*. 2012;86(3). Available from: <https://doi.org/10.1103/physreva.86.032324>.
- [35] Nielsen M, Chuang I. Quantum computation and quantum information: 10th anniversary edition. Cambridge University Press, Cambridge, United Kingdom; 2010.
- [36] Knill E, Laflamme R. Theory of quantum error-correcting codes. *Physical Review A*. 1997;55(2):900–911.
- [37] Gottesman D. The Heisenberg representation of quantum computers. Group22: Proceedings of the XXII International Colloquium on Group Theoretical Methods in Physics, pp 32-43, Cambridge, MA, International Press. 1999;.
- [38] Vaidman L, Goldenberg L, Wiesner S. Error prevention scheme with four particles. *Physical Review A*.

- 1996;54(3):R1745–R1748.
- [39] Grassl M, Beth T, Pellizzari T. Codes for the quantum erasure channel. *Physical Review A*. 1997; 56(1):33–38.
- [40] Gottesman DE. Stabilizer codes and quantum error correction [dissertation]; 1997. Available from: <http://resolver.caltech.edu/CaltechETD:etd-07162004-113028>.
- [41] Roffe J, Headley D, Chancellor N, et al. Protecting quantum memories using coherent parity check codes. *Quantum Science and Technology*. 2018;3(3):035010.
- [42] Calderbank AR, Shor PW. Good quantum error-correcting codes exist. *Phys Rev A*. 1996;54:1098–1106.
- [43] Steane A. Error correcting codes in quantum theory. *Phys Rev Lett*. 1996;77:793–797.
- [44] Kovalev AA, Pryadko LP. Quantum kronecker sum-product low-density parity-check codes with finite rate. *Physical Review A*. 2013;88(1). Available from: <https://doi.org/10.1103/physreva.88.012311>.
- [45] Tillich JP, Zemor G. Quantum LDPC codes with positive rate and minimum distance proportional to the square root of the blocklength. *IEEE Transactions on Information Theory*. 2014;60(2):1193.
- [46] Bravyi SB, Kitaev AY. Quantum codes on a lattice with boundary. arXiv:quant-ph/9811052. 1998;.
- [47] Freedman MH, Meyer DA. Projective plane and planar quantum codes ; 1998.
- [48] Nickerson NH, Fitzsimons JF, Benjamin SC. Freely scalable quantum technologies using cells of 5-to-50 qubits with very lossy and noisy photonic links. *Physical Review X*. 2014;4(4):041041.
- [49] Kelly J, Barends R, Fowler AG, et al. Scalable in situ qubit calibration during repetitive error detection. *Physical Review A*. 2016;94(3).
- [50] Sete EA, Zeng WJ, Rigetti CT. A functional architecture for scalable quantum computing. In: 2016 IEEE International Conference on Rebooting Computing (ICRC). IEEE; 2016.
- [51] O’Gorman J, Nickerson NH, Ross P, et al. A silicon-based surface code quantum computer. *npj Quantum Information*. 2016;2(1). Available from: <https://doi.org/10.1038/npjqi.2015.19>.
- [52] Takita M, Cross AW, Córcoles A, et al. Experimental demonstration of fault-tolerant state preparation with superconducting qubits. *Physical Review Letters*. 2017;119(18).
- [53] Kitaev A. Fault-tolerant quantum computation by anyons. *Annals of Physics*. 2003;303(1):2–30. Available from: [https://doi.org/10.1016/s0003-4916\(02\)00018-0](https://doi.org/10.1016/s0003-4916(02)00018-0).
- [54] Horsman C, Fowler A, Devitt S, et al. Surface code quantum computing by lattice surgery. *New Journal of Physics*. 2012;14(12):123011.
- [55] Edmonds J. Paths, trees, and flowers. *Canadian Journal of Mathematics*. 1965;17:449–467.
- [56] Kolmogorov V. Blossom v: a new implementation of a minimum cost perfect matching algorithm. *Mathematical Programming Computation*. 2009;1(1):43–67.
- [57] Dennis E, Kitaev A, Landahl A, et al. Topological quantum memory. *Journal of Mathematical Physics*. 2002;43(9):4452–4505.
- [58] Criger B, Ashraf I. Multi-path Summation for Decoding 2D Topological Codes. *Quantum*. 2018;2:102. Available from: <https://doi.org/10.22331/q-2018-10-19-102>.
- [59] Gottesman D. Fault-tolerant quantum computation with constant overhead. *Quantum Info Comput*. 2014;14(15-16):1338–1372. Available from: <http://dl.acm.org/citation.cfm?id=2685179.2685184>.
- [60] Shor P. Fault-tolerant quantum computation. In: *Proceedings of 37th Conference on Foundations of Computer Science*. IEEE Comput. Soc. Press; 1996. Available from: <https://doi.org/10.1109/sfcs.1996.548464>.
- [61] Steane AM. Active stabilization, quantum computation, and quantum state synthesis. *Physical Review Letters*. 1997;78(11):2252.
- [62] DiVincenzo DP, Aliferis P. Effective fault-tolerant quantum computation with slow measurements. *Physical Review Letters*. 2007;98(2). Available from: <https://doi.org/10.1103/physrevlett.98.020501>.
- [63] Chao R, Reichardt BW. Quantum error correction with only two extra qubits. *Physical Review Letters*. 2018;121(5). Available from: <https://doi.org/10.1103/physrevlett.121.050502>.
- [64] Wang DS, Fowler AG, Stephens AM, et al. Threshold error rates for the toric and planar codes. *Quantum Info Comput*. 2010;10(5):456–469. Available from: <http://dl.acm.org/citation.cfm?id=2011362.2011368>.
- [65] Dawson CM, Nielsen MA. The solovay-kitaev algorithm. 2005;.
- [66] Eastin B, Knill E. Restrictions on transversal encoded quantum gate sets. *Physical Review Letters*. 2009;102(11). Available from: <https://doi.org/10.1103/physrevlett.102.110502>.

- [67] Bravyi S, Kitaev A. Universal quantum computation with ideal clifford gates and noisy ancillas. *Physical Review A*. 2005;71(2). Available from: <https://doi.org/10.1103/physreva.71.022316>.
- [68] Landahl AJ, Ryan-Anderson C. Quantum computing by color-code lattice surgery ; 2014.
- [69] Yoder TJ. Universal fault-tolerant quantum computation with bacon-shor codes. arXiv:170501686. 2017; .
- [70] Vasmer M, Browne DE. Universal quantum computing with 3d surface codes. arXiv:180104255. 2018;.
- [71] Gambetta JM, Chow JM, Steffen M. Building logical qubits in a superconducting quantum computing system. *npj Quantum Information*. 2017;3(1). Available from: <https://doi.org/10.1038/s41534-016-0004-0>.
- [72] Takita M, Cross AW, Córcoles A, et al. Experimental demonstration of fault-tolerant state preparation with superconducting qubits. *Physical Review Letters*. 2017;119(18).
- [73] Ristè D, Poletto S, Huang MZ, et al. Detecting bit-flip errors in a logical qubit using stabilizer measurements. *Nature Communications*. 2015;6(1). Available from: <https://doi.org/10.1038/ncomms7983>.
- [74] Harty T, Allcock D, Ballance C, et al. High-fidelity preparation, gates, memory, and readout of a trapped-ion quantum bit. *Physical Review Letters*. 2014;113(22).
- [75] O’Gorman J, Campbell ET. Quantum computation with realistic magic-state factories. *Physical Review A*. 2017;95(3). Available from: <https://doi.org/10.1103/physreva.95.032338>.
- [76] Linke NM, Gutierrez M, Landsman KA, et al. Fault-tolerant quantum error detection. *Science Advances*. 2017;3(10):e1701074. Available from: <https://doi.org/10.1126/sciadv.1701074>.
- [77] Vuillot C. Is error detection helpful on IBM 5q chips? *Quantum Information and Computation*. 2018;., Vol. 18, No. 11-12:0949–0964.
- [78] Harper R, Flammia ST. Fault-tolerant logical gates in the IBM quantum experience. *Phys Rev Lett*. 2019;122:080504. Available from: <https://link.aps.org/doi/10.1103/PhysRevLett.122.080504>.
- [79] Wootton JR, Loss D. Repetition code of 15 qubits. *Physical Review A*. 2018;97(5). Available from: <https://doi.org/10.1103/physreva.97.052313>.
- [80] Bombin H, Martin-Delgado MA. Optimal resources for topological two-dimensional stabilizer codes: Comparative study. *Physical Review A*. 2007;76(1). Available from: <https://doi.org/10.1103/physreva.76.012305>.
- [81] Breuckmann NP, Terhal BM. Constructions and noise threshold of hyperbolic surface codes. *IEEE Transactions on Information Theory*. 2016;62(6):3731.
- [82] Draper T, Kutin S. QPIC: Quantum circuit diagrams in LaTeX. ????; Available from: <https://github.com/qpic/qpic>.

## Appendix A. Notation for quantum states

In this review quantum states are represented using Dirac bra-ket notation. Unless otherwise stated, we use the computational basis, given by  $\{|0\rangle, |1\rangle\}$  in the single-qubit case. For example, the general qubit state is written

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle. \quad (\text{A1})$$

For multi-qubit systems, we adopt a labelling convention whereby qubits are implicitly labelled  $1, \dots, n$  from left-to-right, where  $n$  is the total number of qubits. For example, the three-qubit basis element  $|010\rangle$  is equivalent to  $|0\rangle_1 \otimes |1\rangle_2 \otimes |0\rangle_3$  in its full tensor product form.

## Appendix B. Pauli operator notation

The Pauli group on a single-qubit,  $\mathcal{G}_1$ , is defined as the set of Pauli operators

$$\mathcal{G}_1 = \{\pm \mathbb{1}, \pm i\mathbb{1}, \pm X, \pm iX, \pm Y, \pm iY, \pm Z, \pm iZ\}, \quad (\text{B1})$$

where the  $\pm 1$  and  $\pm i$  terms are included to ensure  $\mathcal{G}_1$  is closed under multiplication and thus forms a legitimate group. In matrix form, the four Pauli operators are given by

$$\mathbb{1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \quad (\text{B2})$$

The general Pauli group,  $\mathcal{G}$ , consists of the set of all operators that are formed from tensor products of the matrices in  $\mathcal{G}_1$ . For example, the operator

$$\mathbb{1} \otimes X \otimes \mathbb{1} \otimes Y \in \mathcal{G} \quad (\text{B3})$$

is an element of the four-qubit Pauli group. The *support* of a Pauli operator is given by the list of its non-identity elements. For example, the support of the Pauli operator in equation (B3) is  $X_2Y_4$  where the indices point to the qubit each element acts on. In this review, Pauli errors are always written in terms of their support. As an example, we would say that the bit-flip error  $X_2$  acts on the two-qubit basis element  $|00\rangle$  as follows  $X_2|00\rangle = |01\rangle$ .

## Appendix C. Quantum circuit notation

Quantum circuit notation provides a useful way of representing quantum algorithms. This appendix introduces the basic elements of quantum circuit notation necessary to understand quantum error correction circuits.

### C.1. Single qubit gates

In quantum circuit representation of quantum algorithms each qubit in the quantum register is assigned a wire. These wires are labelled with quantum gates from left-to-right in the order in which they are applied during the quantum computation. As an example, consider the quantum computation described by the application of the unitary operation  $U = X_1Z_1$  to a general qubit state,

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle \xrightarrow{U=X_1Z_1} X_1Z_1 |\psi\rangle = \alpha |1\rangle - \beta |0\rangle. \quad (\text{C1})$$

The quantum circuit for the above computation on a single-qubit is shown below

$$|\psi\rangle \text{---} \boxed{Z} \text{---} \boxed{X} \text{---} XZ|\psi\rangle \text{ ,}$$

where the input state is on the left and the output on the right. Note that the  $Z$ -gate is placed before the  $X$ -gate as it is applied first.

An important single-qubit gate for quantum error correction (and quantum algorithms in general) is the Hadamard gate which is defined in matrix-form as

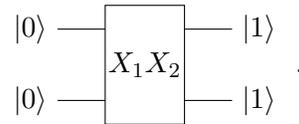
$$H = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \quad (\text{C2})$$

The Hadamard gate has the following effect on the computational basis states

$$\begin{aligned} H|0\rangle &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \\ H|1\rangle &= \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \end{aligned} \tag{C3}$$

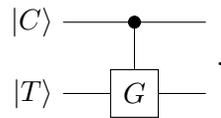
### C.2. Multi-qubit gates

A gate spanning two wires in a quantum circuit represents a multi-qubit operation. As an example, the quantum circuit for the operation  $U = X_1 X_2$  applied to the state  $|\psi\rangle = |00\rangle$  is given by



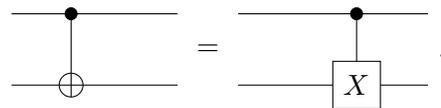
### C.3. Controlled-gates

A controlled gate is a gate whose action is conditional on the value of a ‘control’ qubit. In general, controlled gates are represented as follows in quantum circuit notation



In the above circuit, the top qubit  $C$  is the control and the lower qubit  $T$  is the target. The single-qubit  $G$ -gate is applied to the target if the control qubit is set to  $|C\rangle = |1\rangle$ . If the control qubit is set to  $|C\rangle = 0$ , the  $G$ -gate is not applied to the target.

A commonly occurring gate in quantum error correction is the controlled-NOT (CNOT) gate. In this review, we use the two equivalent symbols for the CNOT gate



### C.4. Measurement in the computation basis

In all the circuits in this review measurement is performed in the computation basis. As an example, consider the following quantum circuit for generating random numbers (the ‘Hello World’ of quantum computing)



where the computational basis measurement is depicted by the gate to the right. The above circuit outputs ‘0’ or ‘1’ with equal probability. The double lines at the end of the circuit indicate that the output is classical information.

## Appendix D. Commutation properties for Pauli operators

The elements of the Pauli group have eigenvalues  $\{\pm 1, \pm i\}$ . As a result, Pauli errors either commute or anti-commute with one another. In this appendix, we outline how to determine whether Pauli operators commute with one another.

First, recall that two operators,  $F_i$  and  $F_j$ , commute if  $F_i F_j = F_j F_i$ , and anti-commute if  $F_i F_j = (-1) F_j F_i$ . For single-qubit Pauli-operators, we see that all pairs of distinct operators anti-commute

$$X_1 Z_1 = -Z_1 X_1, \quad X_1 Y_1 = -Y_1 X_1, \quad Z_1 Y_1 = -Y_1 Z_1. \quad (\text{D1})$$

Now consider the operators  $Z_1 Z_2$  and  $X_1 X_2$ . These multi-qubit operators commute as

$$Z_1 Z_2 X_1 X_2 = Z_1 X_1 Z_2 X_2 = (-1) X_1 Z_1 (-1) X_2 Z_2 = X_1 X_2 Z_1 Z_2. \quad (\text{D2})$$

In general, two Pauli operators will commute with one another if they intersect non-trivially on an even number of qubits as above. Conversely, if the number of non-trivial intersections is odd, then the two operators anti-commute. As an example, consider the two operators  $X_1 Z_2 Z_3 Z_5 X_7$  and  $X_1 X_2 X_5 Z_7$ . These operators intersect on the qubits 1, 2, 5 and 7. However, the intersection on qubit 1 is trivial as both operators apply the same  $X$ -gate to that qubit. The number of non-trivial intersections that remain is therefore three. As the two operators intersect non-trivially an odd number of times, the two operators anti-commute.