Basic Theory and Implementations of Quantum Error Correction

Derek Rodriguez

Basic Theory and Implementations of Quantum Error Correction

Abstract

The introduction of quantum computing has presented algorithmic solutions to computationally difficult challenges that are far more efficient than those of classical computers. These algorithms leverage the properties of quantum mechanics to manipulate the quantum properties of subatomic particles, requiring immense precision and stability. Current quantum hardware, however, is too noisy and introduces too many errors for these algorithms to be useful in practice, necessitating the use of error correction algorithms. This field survey seeks to introduce various principles of quantum mechanics relevant to quantum computing and quantum error correction (QEC), detail the implementation and motivations of a basic QEC algorithm, and provide insight into the current research direction.

Document Type

Undergraduate Honors Thesis

Degree Name

B.S. in Mathematics

First Advisor

Petr Vojtechovsky

Keywords

Quantum error correction (QEC), Shor’s Factoring Algorithm, 9-qubit CSS code, Surface code, Toric code, Threshold theorem, Fault-tolerant quantum computation

Subject Categories

Computer Sciences | Mathematics | Other Computer Sciences | Other Mathematics | Physics | Quantum Physics

Basic Theory and Implementations of Quantum Error Correction

Derek Rodriguez

Department of Mathematics, University of Denver
Denver, CO

Abstract

The introduction of quantum computing has presented algorithmic solutions to computationally difficult challenges that are far more efficient than those of classical computers. These algorithms leverage the properties of quantum mechanics to manipulate the quantum properties of subatomic particles, requiring immense precision and stability. Current quantum hardware, however, is too noisy and introduces too many errors for these algorithms to be useful in practice, necessitating the use of error correction algorithms. This field survey seeks to introduce various principles of quantum mechanics relevant to quantum computing and quantum error correction (QEC), detail the implementation and motivations of a basic QEC algorithm, and provide insight into the current research direction.

Keywords: quantum error correction (QEC), Shor's Factoring Algorithm, 9-qubit CSS code, surface code, toric code, threshold theorem, fault-tolerant quantum computation.

1 Introduction

Quantum computing is a rapidly developing field that leverages the inherent properties of quantum mechanics to perform computations far more efficiently than classical computers. Through exclusively quantum properties such as superposition, entanglement, and interference, quantum computers have shown to be capable of performing a select few tasks exponentially faster than the most efficient known classical algorithms, and an even greater set of problems can be solved polynomially faster. While practical implementation of certain quantum algorithms such as Shor's [1994] Algorithm for factoring large integers has massive ramifications for cryptography and computation theory [1], current hardware is simply not up for the task. Current quantum computers are majorly limited by low fidelity: they have very high error rates that corrupt the data they work on. While the development of higher fidelity quantum computers is primarily an engineering concern under active research [2], many information-theoretic algorithms have been proposed to minimize the risk of error propagation at the cost of increased memory overhead. Adding additional memory to a quantum computer inevitably introduces new errors, however it has been shown that past a certain point, the addition of additional memory enables quantum error correction (QEC) algorithms to suffer fewer errors than they would have otherwise, which of course has major implications for the viability of QEC in general [3].

We seek to introduce various relevant concepts from quantum mechanics and quantum computing before detailing the reasoning behind one such exponentially faster algorithms. We then explain the limitations of implementing these algorithms in practice, introducing the fundamental concepts of QEC and explaining a basic QEC encoding scheme. We conclude with a discussion on the greater developments of QEC and the challenges the field faces.

2 Basic Principles of Quantum Mechanics

2.1 Quantum States

In classical computers, the smallest unit of information is the bit, which is always in one of the discrete states of 0 or 1. In contrast, quantum computers operate on *quantum bits*, or *qubits*, which also have a state, such as the corresponding $|0\rangle$ or $|1\rangle$. Note that we denote quantum states using the symbols '$|\ \rangle$' in what is called *Dirac notation*. What makes qubits different from bits is that they are able to exist 'in-between' the $|0\rangle$ and $|1\rangle$ states through a principle called *superposition*. Despite this, measuring a qubit will only result in one of the states $|0\rangle$ or $|1\rangle$, which results in some very unintuitive consequences for quantum algorithm design.

Superposition is the property that qubits are able to exist in a linear combination of other states, so a general qubit state $|\psi\rangle$ is a combination of the states $|0\rangle$ and $|1\rangle$:

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

where α and β are complex numbers called the *amplitudes* of their corresponding states. The short answer as to why the amplitudes of a quantum state are represented by complex, as opposed to real, numbers is that quantum particles are described as waves with amplitude and phase, and complex numbers are an efficient way to encode both at once[1]. A further restriction we place on the amplitudes of a superposition is that they must result in a normalized vector; that is, $|\alpha|^2 + |\beta|^2 = 1$. This restriction allows us to describe the effects of measuring qubits, in that we should expect to see state $|0\rangle$ with probability $|\alpha|^2$ and state $|1\rangle$ with probability $|\beta|^2$. The state of a qubit can thus be represented by a unit vector in a two dimensional complex vector space with orthonormal basis vectors denoted $|0\rangle$ and $|1\rangle$, known as the *Hilbert space* of the qubit.

Measurement of quantum information comes with some strange consequences. The ability for qubits to exist in superposition decouples our common intuition of states equating observations. If one was to query a classical computer for the value at a given memory address, the value they receive is a physical aspect of the computer's memory; there is a bijective mapping between the numbers representable in the computer and its underlying physicality. In contrast, when we query a quantum computer for the value of a given qubit, we can only observe the values $|0\rangle$ and $|1\rangle$. The underlying amplitudes α and β exist in the so-called "hidden world:" they are completely unmeasurable, and only determine the probabilities for which basis state we receive. Furthermore, the superposition *collapses* upon observation, meaning once we measure a qubit and receive, say, the $|1\rangle$ state, any following measurements that do not first reestablish superposition will also read $|1\rangle$. This means that for an arbitrary state, it is impossible to statistically approximate the values of α and β, since only a single measurement will destroy the hidden values.

Entanglement is yet another phenomenon that separates quantum mechanics from classical mechanics. It describes an apparent teleportation of information between two or more qubits, in which various physical properties such as position, momentum, spin, and polarization are *correlated*; that is, they cannot be measured independently.

Consider a two-qubit system. Just like the case for a single qubit, these two qubits can be in a superposition of states, but we will only ever measure one of $|00\rangle, |01\rangle, |10\rangle, |11\rangle$, corresponding to the four possible binary values of a two-bit system. Suppose our two-qubit system is in the following superposition:

$$\frac{|00\rangle + |11\rangle}{\sqrt{2}}.$$

Now, consider what the amplitudes of the two superpositions tell us: upon measuring the first qubit, we have a $\frac{1}{\sqrt{2}}^2 = 1/2$ probability of seeing $|0\rangle$ and an equal probability of seeing $|1\rangle$. When we measure the second qubit, however, the overall state of the system must be either $|00\rangle$ or $|11\rangle$, meaning that we will always measure the same state as we did for the first qubit. Fascinatingly, this holds true even if we physically move the qubits so far apart that measuring both simultaneously means not even light could travel quickly enough to 'inform' them what their entangled partner's state collapsed to. This principle does not, however, allow for communication faster than light: the states of entangled qubits are correlated in that they will always collapse to the same basis state when measured, but exactly which basis state they are measured at is fundamentally random, meaning we cannot rely on any deterministic communications protocol [4].

There are a number of ways that we can entangle qubits in the physical world, one such method applies a Hadamard gate followed by a CNOT gate, both of which are discussed in Section 2.2.

Interference results from the fact that the amplitudes of quantum states have associated *phases* as well as probability amplitudes. Recall how each basis state $|j\rangle$ in a superpositioned quantum state has an associated complex number a_j called an amplitude where $|a_j|^2$ equals the probability of measuring $|j\rangle$. Viewed as a vector in the complex plane, the probability is simply its magnitude. The rotation of the vector represents the phase of the state, which is physically unobservable, but does lead to some interesting consequences for quantum computing. Imagine we change a quantum state in such a way that the final amplitude of some basis state $|k\rangle$ is the sum of two amplitudes a_k and b_k. If these complex vectors have the same rotation, their sum will constructively interfere, resulting in a greater magnitude and thus greater probability of measuring $|k\rangle$. On the other hand, if the amplitudes have opposing phases, they will destructively interfere and reduce the probability of measuring $|k\rangle$. Thus, while amplitudes have phases that we cannot directly observe, they can be manipulated to increase our likelihood of measuring a useful result. This is the key principle of quantum algorithm design.

[1] Quantum waves also have a frequency, but it isn't particularly important for our model of quantum computation. Physical quantum computers do care about them, for instance by manipulating particles using lasers attuned to specific frequencies.

2.2 Quantum Computation

Similarly to individual qubits, the complete set of possible states that a quantum register (multiple qubits taken as a whole) can be in at any given time is called the *state space*, with the *state vector*, a unit vector in the state space, describing which specific state the machine is in. In general, a quantum computer with n many qubits will have a corresponding 2^n dimensional Hilbert space.

As an example, a two-qubit quantum system will have a four-dimensional state space with orthonormal basis vectors $\{|0\rangle, |1\rangle, |2\rangle, |3\rangle\}$. Note that these basis vectors can be denoted arbitrarily; typically decimal and boolean representations are the most common. A general state for our computer is then

$$|\psi\rangle = \sum_{k=0}^{3} a_k|k\rangle, \text{ where each } a_i \in \mathbb{C} \text{ and } \sum_{k=0}^{3} |a_i|^2 = 1.$$

Classical computers are built from electrical circuits comprised of wires to transmit information and logic gates that manipulate that information. In fact, all classical (finite and deterministic) algorithms can be expressed as a circuit comprised of Boolean logic gates [5]. In much the same way, a quantum computer is built from a *quantum circuit* comprised of wires to transmit quantum states and *quantum gates* to manipulate these states. Quantum algorithms can thus be viewed as an array of operations that cleverly 'massage' the hidden state of the machine in such a way that when we measure the state and its superpositions collapse, we have a greater likelihood of getting a useful result.

Unitary Operators are linear transformations whose adjoint is equal to its inverse. That is, an invertible complex square matrix U is *unitary* if its inverse U^{-1} equals its conjugate transpose $U^\dagger$, created by first transposing U and then applying complex conjugation to each entry (the complex conjugate of $a + bi$ being $a - bi$ for real a and b), so that $U^\dagger U = UU^\dagger = I$, the identity matrix.

In terms of algebra, unitary operators are simply automorphisms of Hilbert spaces, in that they preserve the topology of the space[2]. They thus form a group under composition, the isometry group of H, for some Hilbert space H. For finite dimensions, this group is isomorphic to the unitary group of degree n, the group of all unitary $n \times n$ matrices under matrix multiplication.

As it turns out, the laws of quantum mechanics only permit state changes to be represented by unitary matrices, as otherwise there is a possibility that summing the probabilities of obtaining each possible outcome results in something other than 1, leading to strange paradoxes such as time travel, faster-than-light communication, and violations of the second law of thermodynamics [4]. In fact, *any* unitary matrix specifies a valid quantum gate, meaning there are technically uncountably infinite valid quantum logic gates. A number of gates have been named [6] [7], but for our purposes we will only introduce the high-level properties of a few of the most commonly used in the literature. More information regarding their underlying mathematics can be found in tables 1 and 2 of Appendix A.1

Single Qubit Gates are quantum logic gates that act on a single qubit. Whereas the only nontrivial single-bit classical logic gate is the NOT gate, which maps $0 \mapsto 1$ and $1 \mapsto 0$, there are several useful single-qubit gates:

- There is a direct analogue to the classical NOT gate, known as the *Pauli-X gate* (denoted X), which maps $|0\rangle \to |1\rangle$ and $|1\rangle \to |0\rangle$, but since all operators must be unitary matrices (and thus linear transformations), it does this by simply swapping the amplitudes of the basis states.
- A complement to the probability-swapping X gate, the *Pauli-Z gate* (Z) swaps the relative phase of a qubit. As a reminder, this does not affect the probabilities of measuring $|0\rangle$ or $|1\rangle$, but can be used to cause interference that affects these probabilities in a greater circuit.
- The *Pauli-Y gate* (Y) is simply a combination of the X and Z gates, in that it performs a bit-flip and a phase-flip. These Pauli gates, when represented as 2×2 unitary matrices alongside the trivial identity matrix, form a group known as the Pauli group. These gates will be important in our discussion of quantum error correction in Section 4.
- Next, the *phase gate* (S) introduces a relative phase of $\pi/2$ to the $|1\rangle$ state of a qubit, resulting in unchanged probabilities but half the phase shift of the Z gate ($\pi/2$ instead of π).
- The $\pi/8$ gate (T) introduces half the relative phase of the S gate; that is, $\pi/4$.
- The final single-qubit gate we will discuss is the *Hadamard gate*, denoted H. The Hadamard gate acts on single qubits and can be thought of as a superpositioning gate, mapping the basis vectors to balanced superpositions:

$$\mathrm{H}|0\rangle = \frac{1}{\sqrt{2}}\big(|0\rangle + |1\rangle\big), \mathrm{H}|1\rangle = \frac{1}{\sqrt{2}}\big(|0\rangle - |1\rangle\big).$$

The gate is also sometimes used on states already in superposition in order to constructively

[2] in that are linear transformations mapping bounded subsets of the space to other bounded subsets, thus preserving the vector space structure, while also preserving the inner product.

or destructively interfere with its component amplitudes.

Multi-Qubit Gates provide access to conditional computation. Two-qubit gates typically do this by using the state of a *control* qubit to determine the activation of a single-qubit gate on a *target* qubit.

- The prototypical multi-qubit gate is the *controlled-NOT gate* (CNOT). If the control qubit is set to $|1\rangle$, the target qubit is flipped in the same action as an X gate. If the control qubit is $|0\rangle$, the target bit is unchanged. This gate is reminiscent of the classical XOR gate, since it maps the general state $|A, B\rangle$ to $|A, B \oplus A\rangle$, where $\oplus$ is addition modulo two.
- The *swap gate* (SWAP) exchanges the states of two qubits, so that $|\psi\rangle|\phi\rangle$ becomes $|\phi\rangle|\psi\rangle$.
- The *controlled-Z gate* (CZ) applies a Z gate to the target qubit if the control qubit is $|1\rangle$.
- The *controlled phase gate* (CP(θ)) is a generalization of the CZ gate, in that CZ=CP(π).

There are only a few common three-qubit gates, all of which add another control qubit to one of the listed two-qubit gates.

- The *Toffoli gate*, or controlled-controlled-NOT (CCNOT) flips the target qubit if both control qubits are $|1\rangle$.
- The *Fredkin gate*, or controlled-SWAP (CSWAP), swaps the two target qubits if the control qubit is $|1\rangle$.
- Finally, the *controlled-controlled-phase gate* (CCP(θ)) applies a phase shift of $e^{i\theta}$ to the $|111\rangle$ basis state if both control qubits are $|1\rangle$.

Just as all classical logic circuits can be rewritten using only NAND gates, a *universal* set of quantum gates can approximate any unitary operation to arbitrary accuracy. In other words, universal quantum gates are able to approximate any quantum circuit. One such set is {H, S, T, CNOT} [4]; another is {CCNOT, H} [8].

Reversible Logic refers to a result of our construction of quantum algorithms as arrays of quantum gates: each computation is completely reversible. That is, knowing the quantum state that has exited a gate uniquely implies a specific quantum state that must have entered it. In this sense, quantum gates can be thought of as bijections between the sets of input and output states, since any state we find ourselves in after a gate is applied must have one (surjective) and only one (injective) corresponding input state. An example of an *irreversible* gate is the classical XOR gate, since given an output of 0, we cannot determine if the input values were both 1 or both 0.

As a result, there are a few characteristics of classical circuits that are not possible on a quantum circuit. As there must be a bijective mapping from the input states to outputs states for a given quantum gate, we cannot join or split wires (known in classical circuitry as `FANIN` and `FANOUT`, respectively). This also prevents us from allowing feedback loops from one part of the circuit to another; any repetition must be programmed as physically repeating gates. As summary of the concepts covered so far, we explain an important quantum transformation below.

The Quantum Fourier Transform (QFT) is the quantum analogue to the discrete Fourier transform, an astonishingly applicable tool in digital signal processing, image processing, partial differential equations, convolution application, and a variety of numerical algorithms [9]. Originally discovered by Coppersmith [2002], the QFT is widely applicable in quantum algorithms such as quantum phase estimation, the hidden subgroup problem, and the order-finding problem in Shor's Algorithm [10], discussed in Section 3.2.

The transform itself acts on a quantum state $|\psi\rangle = \sum_{k=0}^{N-1} a_k|k\rangle$ with N many basis states (so $\log_2(N)$ qubits, where N is a power of two), and maps it to

$$\sum_{j=0}^{N-1} b_j|j\rangle, \text{ where } b_j = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k e^{2\pi ijk/N}.$$

In plain English, the QFT converts one quantum state into another by converting the unobservable phase information in the amplitudes of the first state into probability-affecting amplitudes in the second. It can generally be thought of as a frequency-extracting operation, as any amplitudes with similar facing phases (when considered as vectors in the complex plane) will constructively interfere, correlating to an increased probability amplitude for any basis states that are a multiple of N/r, where N is the number of basis states total and r is the period (the rate at which the bases with similarly-phased amplitudes occur in the superposition).

We have finally covered all the quantum machinery necessary to closely dissect an important quantum circuit, Shor's Algorithm.

3 Shor's Algorithm

Shor's Algorithm actually denotes a number of closely related algorithms, but for this analysis we will focus on factoring integers as a simpler case of the discrete logarithm problem [1].

The problem that Shor's Algorithm solves is: given a composite integer n, find two integers p and q

such that $n = pq$ and $p, q > 1$. Note that a solution to this problem permits a complete factorization of any n, since if either p or q are composite, the same algorithm can be used on them until a complete prime factorization is achieved.

The algorithm itself consists of two parts: a classical algorithm and a quantum subroutine. The quantum subroutine does not directly compute the factors of n, rather it finds the order of an element x in the multiplicative group $(\mathbb{Z}/n\mathbb{Z})^{\times}$; that is, the smallest positive integer r such that $x^r \equiv 1 \pmod{n}$. Using a bit of number theory, we show how the problem of factoring n is reduced to the problem of order finding using the classical reduction algorithm below.

3.1 Classical Reduction

Firstly, note that we are able to classically compute the greatest common divisor between two integers efficiently (that is, in polynomial time) using the Euclidean algorithm. In particular, we can compute $\gcd(2, n)$ efficiently. In the case that $\gcd(2, n) \neq 1$, n is even and so 2 and $n/2$ are trivially factors.

Further note that there exist efficient classical algorithms for determining whether an integer n is a prime power [11], as well as efficient classical factorization algorithms for prime powers.[3] Therefore, we may assume n is odd and not a prime power.

Let x be a randomly chosen integer such that $1 < x < n$. In the case that $\gcd(x, n) \neq 1$, then clearly $\gcd(x, n)$ and $n/\gcd(x, n)$ are factors of n and we are done. Otherwise, n and x are relatively prime, so that $x \in (\mathbb{Z}/n\mathbb{Z})^{\times}$; that is, x is an element of the multiplicative group of integers modulo n. Since there are countably infinite many powers of x and a finite number of elements of $(\mathbb{Z}/n\mathbb{Z})^{\times}$, by the pigeonhole principle there must be two integers s and t such that $x^s \equiv x^t \pmod{n}$. Without loss of generality let $s > t$. Since x has a multiplicative inverse x^{-1}, we have $x^s \equiv x^t \pmod{n} \Rightarrow x^{s-t} \equiv x^t x^{-t} \equiv 1 \pmod{n}$. x thus has a multiplicative order r modulo n, where r is the smallest positive integer such that

$$x^r \equiv 1 \pmod{n}.$$

At this point in the algorithm, the quantum subroutine would calculate the value of r. We assume the value is known for the remainder of the classical reduction, and investigate the quantum subroutine afterwards. Now, $x^r \equiv 1 \pmod{n}$ implies $x^r - 1 \equiv 0 \pmod{n}$, so that n divides $x^r - 1$. We then factor $x^r - 1$ by difference of squares, resulting in

$$n \mid (x^{r/2} - 1)(x^{r/2} + 1).$$

[3] If n is a prime power, it is of the form p^i for some $i \leq \log_2(n)$. It is then possible to compute each of the first $\log_2(n)$ roots of n with the Newton method and ensure each result is a prime integer using the AKS primality test [12].

If r is odd, $x^{r/2}$ is not an integer and cannot belong to $(\mathbb{Z}/n\mathbb{Z})^{\times}$, in which case we must randomly choose another $x \in (\mathbb{Z}/n\mathbb{Z})^{\times}$ and begin again. We should expect not to need to repeat this step very often; since n is an odd composite integer that is not a prime power, we have $n = p_1^{k_1} p_2^{k_2} ... p_m^{k_m}$ for odd primes p_i. By the Chinese Remainder Theorem then $(\mathbb{Z}/n\mathbb{Z})^{\times} \cong (\mathbb{Z}/p_1^{k_1}\mathbb{Z})^{\times} \times ... \times (\mathbb{Z}/p_m^{k_m}\mathbb{Z})^{\times}$. The order of $x \in (\mathbb{Z}/n\mathbb{Z})^{\times}$ is then the least common multiple of the orders r_i of $x \pmod{p_i^{k_i}}$; that is, $r = \text{lcm}(r_1, r_2, ..., r_m)$. Note that this implies that r is odd only if all r_i's are odd. For each odd prime power $p_i^{k_i}$, the corresponding factor $(\mathbb{Z}/p_i^{k_i}\mathbb{Z})^{\times}$ is cyclic of order

$$\phi(p_i^{k_i}) = p_i^{k_i} - p_i^{k_i - 1} = p_i^{k_i - 1}(p_i - 1),$$

where $\phi(n)$ is Euler's totient function. As each p_i is an odd prime, $p_i - 1$ is even, so each subgroup $(\mathbb{Z}/p_i^{k_i}\mathbb{Z})^{\times}$ has even order. Therefore, since r_i must divide $\phi(p_i^{k_i})$, it either divides an odd factor of an even number or is even itself. As $r = \text{lcm}(r_1, r_2, ... r_m)$, the likelihood that at least one r_i is itself even is very high.

We can therefore assume r is odd. Suppose $n \mid x^{r/2} - 1$. then $x^{r/2} \equiv 1 \pmod{n}$ and so $r/2$ is the order of x, a contradiction. Now, let $g = \gcd(x^{r/2} + 1, n)$. If g is nontrivial, then since $n \nmid x^{r/2} - 1$, g and n/g are nontrivial factors of n. If g is trivial, then $n \mid (x^{r/2} + 1)$ and we are unable to find nontrivial factors of n with this chosen x.

Thus, the classical reduction algorithm only fails to find a nontrivial factor of n if r is odd or
$x^{r/2} \equiv -1 \pmod{n}$. Shor [1997] showed that the reduction algorithm with this criterion for failure will find a factor of n with probability at least $1 - 1/2^{k-1}$, where k is the number of distinct prime factors of n [13], a full explanation of which we omit for brevity. Since n has more than 1 distinct prime factor, the probability for success is greater than zero, so we are guaranteed to find a suitable x eventually. In summary, the classical reduction algorithm works as follows:

1. Choose a random integer x such that $1 < x < n$.
2. Compute $k = \gcd(x, n)$ using the Euclidean Algorithm.
3. If $k \neq 1$, it and n/k are nontrivial factors of n and we are done.
4. Otherwise, the quantum subroutine finds the order r of x.
5. If r is odd, start again from step 1.
6. Otherwise, compute $g = \gcd(x^{r/2} + 1, n)$.
7. If $g \neq 1$, then it and n/g are nontrivial factors of n and we are done.
8. Otherwise, start again from step 1.

3.2 Quantum Subroutine

Given relatively prime integers n and x such that $1 < x < n$ and n is odd, the quantum subroutine finds the order r of x using a quantum circuit with two memory registers. The second register requires q qubits, where q is the power of 2 with $n^2 \leq q < 2n^2$; that is, $q = \lceil \log_2 n \rceil$, the number of bits in the binary representation of n. The size of the first register determines the accuracy of the overall circuit, with $2q$ qubits typically sufficient.

The first step of the algorithm is to put the first register in a uniform superposition of states representing the elements of $(\mathbb{Z}/q\mathbb{Z})^{\times}$; i.e., the integers $a \pmod q$. This is a relatively simple process, as it essentially just involves applying Hadamard gates in parallel to put each individual qubit in the first register into the superposition $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. This results in the machine state

$$\sum_{a=0}^{q-1} \frac{1}{\sqrt{q}} |a\rangle \otimes |1\rangle,$$

where the uniform superposition of every $|a\rangle$ denotes the state of the first register and the basis state $|1\rangle$ denotes that of the second. Note the symbol $\otimes$ refers to the *tensor product* of the states of the two registers, and denotes that the two are not entangled.

Modular Exponentiation is then applied to compute $x^a \pmod n$ in the second register, the circuit for which essentially follows the classical method. First, we repeatedly square $x \pmod n$ to find $x^{2^i} \pmod n$ for all $i < q$. Next, we iterate through the binary expansion of a; if 2^i appears, we multiply the second register by $x^{2^i} \pmod n$. Since we only need to compute $x^a \pmod n$ for fixed x and n, we can build the circuit such that they don't need to be stored in a register, and are rather built into the structure of our circuit. For this reason, providing a visual representation of the circuit is impossible in general, so we provide general pseudocode to summarize. We denote x_i as the ith bit of the binary expansion of x, indexed least-to-most significant:

```
1. $power \leftarrow 1$
2. for $i = 0$ to $q - 1$
3.     if $(a_i == 1)$ then
4.         $power \leftarrow power * x^{2^i} \pmod n$
5.     endif
6. endfor
```

This circuit is simple enough to build in practice: the second register begins in the $|1\rangle$ state, the loop can be unrolled and repeated physically, and the comparison with the bits of a is just a controlled unitary matrix, akin to the *CZ* or *CNOT* gates. The only difficult part to implement is line 4. To that end, $c = x^{2^i} \pmod n$ is computed classically, which can be done efficiently, and built into the structure of the quantum circuit to create a subroutine that takes some b as input and outputs $bc \pmod n$. Note that if $\gcd(c, n) \neq 1$; that is, if any repeated squaring of x shares a factor with n, then two distinct values of b will be mapped to the same value of $bc \pmod n$, making the multiplication noninjective. Thankfully, the case that $\gcd(x, n) \neq 1$ (and as an extension, any repeated squaring of x) results in trivial factors during the prior classical reduction, so such a circuit can be built reversibly. Remember from Section 2.2 that quantum circuits must necessarily use only reversible operations. The pseudocode for this circuit is as follows:

```
1. $result \leftarrow 0$
2. for $j = 0$ to $q - 1$
3.     if $(b_j == 1)$ then
4.         $result \leftarrow result + 2^j c \pmod n$
5.     endif
6. endfor
```

Here, we take b as input and return $(b, bc \pmod n)$ via repeated addition modulo n. No information from b was lost, which is the reason it remains in the output. Remember $c = x^{2^i} \pmod n$ is computed classically, so $2^j c \pmod n$ can be precomputed and built into the structure of this circuit. We now need some way of reversibly removing the state of b from our register, so we are left with only $bc \pmod n$. Since $\gcd(c, n) = 1$, $c \in (\mathbb{Z}/n\mathbb{Z})^{\times}$ and so there must be some $c^{-1} \pmod n$ such that $cc^{-1} \equiv c^{-1}c \equiv 1 \pmod n$. We can thus multiply $bc \pmod n$ by c^{-1} to get $(bc \pmod n, bcc^{-1} \pmod n) = (bc \pmod n, b)$, which is just the reverse of the operation we want. Since every computation we make is reversible, we can simply turn it around to erase b:

```
1. for $j = 0$ to $q - 1$
2.     if $(result_j == 1)$ then
3.         $b \leftarrow b - 2^j c^{-1} \pmod n$
4.     endif
5. endfor
```

Of course, we could not have just set b to 0 directly, as that would violate the reversibility requirement, but if everything went according to plan, b should now have the value 0. In fact, since we no longer need the superposition of b to be maintained, we can use it as a simple 'checksum' to verify that the rest of the computation succeeded: we simply measure b, and if it collapses to something other than 0 we know an error occurred. If we do measure 0, beyond knowing the state of b exactly, we also prevent any future computations using the qubits of b from magnifying errors, since any amplitude b had for being non-zero has been eliminated. This technique demonstrates what is known as the *quantum Zeno effect*, in which performing repeated measurements at known values has been shown to provide higher success rates than not [14].

Following Modular Exponentiation, our system is in the state

$$\sum_{a=0}^{q-1} \frac{1}{\sqrt{q}} |a\rangle |x^a \pmod{n}\rangle.$$

Critically, the two registers are now entangled[4], as the state of the second register was built based on the state of the first. We now measure the second register. Since the two are entangled, upon measuring some value k, the first register will be in a uniform superposition of all the states a that satisfy $k = x^a \pmod{n}$, with the states that do not having an amplitude of zero.

We then perform a quantum Fourier transform on the first register, described in Section 2.2. This effectively converts the implicit periodicity of the remaining nonzero amplitudes into an explicit and drastic increase in probabilities for measuring a multiple of $2^{2q}/r$. The reason we expect to measure a multiple of $2^{2q}/r$ and not r itself is that the dimensionality of the first register is 2^{2q}, and so for period r we should expect $2^{2q}/r$ basis states to remain after measuring the second register; this is why the size of the first register determines the accuracy of the subroutine.

Next, we measure the state of the first register. As discussed above, it will output a random integer m of the form $\frac{j}{r}2^{2q}$ for a random $j \in \{0, 1, ..., r-1\}$. Dividing m by 2^{2q}, we are left with some decimal value $\frac{m}{2^{2q}} \approx \frac{j}{r}$, from which we can attempt to identify r. We apply the continued-fraction algorithm using a classical computer, which can be done efficiently, to find relatively prime integers b and c such that b/c is the best fractional approximation for the measured value $m/2^{2q}$. The way this algorithm works is not particularly important for the purposes of the quantum subroutine; curious readers can find more information in [15]. Since the first register used $2q$ qubits, the values $b = j$, $c = r$ will give the best approximation for the value $m/2^{2q}$ that was actually measured. Note that the measurement of the first register could have resulted in a j that shared factors with r, meaning that the computed b and c may have lost factors originally in j and r, leading to an incorrect period recovered. In order to solve this, we start by running the entire quantum subroutine multiple times, resulting in a list of fractional approximations $\frac{b_1}{c_1}, \frac{b_2}{c_2}, ..., \frac{b_s}{c_s}$, where s is the number of times we rerun the subroutine. Now, these approximations will likely be for different values of j, as there was an equal probability of measuring any multiple of $2^{2q}/r$, but they should all be for the same value of r. We can thus take the least common multiple of each approximation, which, as it is based on the greatest common divisor, can be done efficiently: $\text{lcm}(\frac{b_1}{c_1}, \frac{b_2}{c_2}, ..., \frac{b_s}{c_s})$ will thus reintroduce any lost factors of r and should result in the order r of the original integer $x \pmod{n}$ with very high probability. In fact, Ekerå [2022] determined that as r tends to infinity, the probability of success for this quantum subroutine to compute r in a single run tends to 1, with more moderate values of r on the scale of around 128 bits having success rates exceeding $1 - 10^{-4}$ [16]. We summarize the quantum subroutine as follows:

1. Using Hadamard gates, set the first register to a uniform superposition of the integers $a \pmod{n}$.
2. Perform modular exponentiation to compute the values $x^a \pmod{n}$ in the second register.
3. Measure the second register, thus collapsing any values in the entangled second register that do not exponentiate to this measured value.
4. Perform a quantum Fourier transform on the first register, converting the implicit periodicity of the remaining values into high amplitudes for the multiples of $2^{2q}/r$.
5. Measure the first register and divide by 2^{2q} to receive some decimal value j/r.
6. Perform the continued fractions algorithm on j/r to find integers b and c such that $\gcd(b, c) = 1$ and $\frac{b}{c}$ is the best approximation of $\frac{j}{r}$.
7. If $c \neq r$, store the fraction b/c and repeat from step 1 s many times. Find the least common multiple of each computed b_i/c_i; this will equal r with very high probability.

3.3 Limitations

The ability to factor large integers in polynomial time has particular importance for cryptography, in which public-key encryption schemes, such as the widely-used RSA method [17], depend on the computational difficulty of factoring large numbers. If a sufficiently powerful quantum computer were built, Shor's Algorithm could be used to defeat these cryptographic systems, rendering current secure communications vulnerable to attack. This potential has spurred significant research into quantum-resistant cryptography [18].

However, despite its theoretical power, algorithm has yet to be implemented at scale. In fact, the great-

[4] Recall how in Section 2.1, a Hadamard gate followed by a CNOT gate, essentially the quantum subroutine up to this point, was a way to entangle qubits.

est number that has been successfully factorized is 21 [19]. The primary obstacle is the extreme sensitivity of current hardware to environmental noise; even slight errors in application of a quantum gate can compound and magnify until the entire circuit fails.

4 Quantum Error Correction

It would seem, then, that quantum computation should be impossible. How could we possibly hope to build a machine that relies on the precise manipulation of quantum states with such low fidelity hardware? Following Shor's algorithm, the field of quantum error correction emerged in an attempt to use algorithmic quantum encoding schemes to protect quantum information from noise and errors beyond what current hardware fidelity allows.

4.1 Basic Principles of QEC

Decoherence is the loss of information due to environmental interference. Unlike classical bits which are stored as thousands of electrons and thus relatively stable to the decoherence of individual particles, qubits are typically single subatomic particles, meaning any environmental interference has a high probability to create unintentional effects like superposition alterations and entanglement. This interaction is essentially noise in the classical sense and causes quantum states to unpredictably decohere, which can quickly amplify in a quantum circuit, destroying the information we were working on. As any error leading to decoherence must be a unitary operation, all errors can be reduced to bit-flips, where an erroneous X gate flips the values of $|0\rangle$ and $|1\rangle$, phase-flips in which a Z gate flips the phase of $|1\rangle$, or some combination of the two [20].

The No-Cloning Theorem states that an arbitrary unknown quantum state cannot be perfectly copied. More precisely, for any state $|s\rangle$ and unknown state $|v\rangle$, there is no unitary operator taking $|v\rangle \otimes |s\rangle$ to $|v\rangle \otimes |v\rangle$. This presents what was thought to be an insurmountable obstacle for QEC, since in classical computing, the simplest way to protect data from errors is by making redundant copies. If one copy gets corrupted, we can simply hold a majority vote among the others and, assuming fewer than half of our backups were corrupted, recover the original. However, the no-cloning theorem prevents this simple replication for quantum states; we cannot just make multiple identical copies of a qubit and then compare them to find the correct state if errors occur (not to mention how measuring these states would cause the collapse of their superpositions). It turns out there is a way to redundantly store quantum information, through the use of entanglement. QEC exploits the principles of entanglement to distribute this information across multiple physical qubits in such a way that local errors can be detected and corrected without directly measuring or copying the encoded quantum state [21].

4.2 Shor's 9-Qubit Repetition Code

As mentioned above, QEC was largely considered impossible due to the no-cloning theorem and the fragility of quantum states. Shor [1995] was the first to show that correcting arbitrary qubit errors is indeed possible through the use of a clever encoding scheme [22]. Shor's 9-qubit code, also known as Shor's code, is a foundational quantum error correction scheme that has since served as a basis for a more general class of codes known as the CSS codes (Calderbank-Shor-Steane codes).

Shor's 9-qubit code is designed to protect the information of a single qubit from the arbitrary single-qubit errors discussed above: bit-flips, phase-flips, and combinations of the two. It achieves this by encoding the state of a single qubit, known as the *logical qubit*, into nine total qubits, known as the *physical qubits*. We denote the state of the logical qubit as $|v_L\rangle$. The code itself works by combining two sub-codes: one that corrects bit-flip errors built on the work proposed by Peres [1985], and another that converts the problem of phase-flips to that of bit-flips [23].

The 3-Qubit Bit Flip Code seeks to correct bit-flip errors for a qubit with state $|v\rangle = \alpha|0\rangle + \beta|1\rangle$. We encode this logical state into three qubits by entangling them together as follows:

$$\begin{aligned} |0_L\rangle &= |000\rangle \\ |1_L\rangle &= |111\rangle. \end{aligned}$$

Note that we denote the physical qubits as $|q_i\rangle$, where i is the index of the qubit in the encoding from least-to-most significant. The key reason this code works is that instead of measuring values to form a majority vote, which would collapse any useful superpositions and eliminate any advantage of using quantum computing, we can measure the parity of the overall state without disturbing the superposition. Recall from Section 2.2 that the CNOT gate essentially acts as a parity-checking gate, since it will set the target qubit to $|0\rangle$ if both it and the control qubit have the same value, and $|1\rangle$ if they do not. To this end, we introduce two auxiliary qubits $|a_1\rangle$ and $|a_2\rangle$, called *ancilla qubits*, that record this parity as follows:

1. Apply a CNOT gate with $|q_1\rangle$ as the control and $|a_1\rangle$ as the target.

2. Apply another CNOT gate with $|q_2\rangle$ as the control and $|a_1\rangle$ as the target.
3. We can now measure $|a_1\rangle$ without destroying the superposition of $|v_L\rangle$. If $|a_1\rangle = |0\rangle$, both $|q_1\rangle$ and $|q_2\rangle$ must have had the same value. Otherwise, we must perform the same procedure to physical qubits $|q_2\rangle$ and $|q_3\rangle$ onto ancilla $|a_2\rangle$ to determine which specific qubit was flipped.

We can interpret the resulting values of $|a_1\rangle$ and $|a_2\rangle$, known as the *syndrome*, to find the location of the error:

- 00 indicates that no error occurred,
- 01 indicates that $|q_2\rangle$ and $|q_3\rangle$ are different, while $|q_1\rangle$ and $|q_2\rangle$ agree, so $|q_3\rangle$ has been flipped,
- 10 likewise indicates that $|q_1\rangle$ has been flipped,
- 11 indicates that $|q_2\rangle$ has been flipped.

As a bit-flip error is just an application of the X gate, which is its own inverse, once we determine the location of the error via the syndrome measurement above, we can simply apply another X to the offending qubit to reverse the error.

The 3-Qubit Phase Flip Code are handled in much the same way as bit-flip errors. The key insight is that a phase-flip on the basis states $|0\rangle$ and $|1\rangle$ is equivalent to a bit-flip after conjugation with a Hadamard gate; that is, $\text{HZH}^{-1} = \text{X}$. Thus, to check for phase-flip errors, we encode a single logical qubit into three physical qubits as above, perform Hadamard gates on each, use the exact same CNOT-based syndrome measurement circuit as bit-flips, apply an X gate to any offending qubits, then finally undo the Hadamard transformation by applying H to all qubits again.

Shor's Code simply combines the two 3-qubit codes. Starting with an arbitrary qubit state $|v\rangle = \alpha|0\rangle + \beta|1\rangle$, perform the following:

1. Initialize two ancilla qubits to $|0\rangle$, so that the overall state of the system is $(\alpha|0\rangle + \beta|1\rangle) \otimes |0\rangle \otimes |0\rangle$.
2. Apply a CNOT gate with the first qubit as control and second qubit as target, giving us $(\alpha|00\rangle + \beta|11\rangle) \otimes |0\rangle$.
3. Apply a second CNOT gate with the first qubit as control and third qubit as target, resulting in $\alpha|000\rangle + \beta|111\rangle$.

At this point, we have encoded $|v\rangle$ for bit-flip errors. To encode these three physical qubits for phase-flip errors, we must convert them to the *Hadamard basis*, which just entails applying a Hadamard gate to take $|0\rangle \to |+\rangle$ and $|1\rangle \to |-\rangle$, where

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

The encoding scheme for each physical qubit $|q_i\rangle = \alpha_i|0\rangle + \beta_i|1\rangle$ is thus:

1. Initialize two ancilla qubits to $|0\rangle$, so that the overall state of the system is $(\alpha_i|0\rangle + \beta_i|1\rangle) \otimes |0\rangle \otimes |0\rangle$.
2. Apply a Hadamard gate to $|q_i\rangle$, so that the state is then $(\alpha_i|+\rangle + \beta_i|-\rangle) \otimes |0\rangle \otimes |0\rangle$.
3. Apply a CNOT gate with the first qubit as control and second qubit as target, giving us $(\alpha_i|++\rangle + \beta_i|--\rangle) \otimes |0\rangle$.
4. Apply a second CNOT gate with the first qubit as control and third qubit as target, resulting in $\alpha_i|+++\rangle + \beta_i|---\rangle$.
5. Finally, apply another Hadamard gate to each qubit to return them to the standard computational basis, resulting in $\alpha_i|000\rangle + \beta_i|111\rangle$.

In short, we first encode a single logical qubit using the 3-qubit bit-flip code, then each of the three resulting physical qubits are encoded with the 3-qubit phase-flip code, resulting in a total of 9 physical qubits per logical qubit. The basis states are thus encoded as such:

$$|0_L\rangle = \frac{(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)}{2\sqrt{2}}$$

$$|1_L\rangle = \frac{(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)}{2\sqrt{2}}.$$

4.3 The Threshold Theorem

The threshold theorem is perhaps the most important result for the field of QEC. Originally proven by Aharonov and Ben-Or [1996], Knill, Laflamme, and Zurek [1998], and independently by Kitaev [1997], it states that if the error rate of individual quantum gates is below a certain threshold, then it is possible to perform arbitrarily long quantum computations with an arbitrarily low error rate, given a sufficient number of qubits and quantum gates [24] [25] [26]. In essence, it asserts that a *fault-tolerant* quantum computer can be built: a system that can perform any task without succumbing to decoherence, so long as the physical error rates of its operations are low enough. The existence of such a threshold implies that with enough gate fidelity, the additional qubits added by QEC will not introduce more errors than they cause, and will in fact reduce the overall error probability of the system. While the exact threshold value depends on the specific error correction code and architecture, estimates typically range from 10^{-3} to 10^{-4} per gate operation.

4.4 Surface Codes

Surface codes, originally introduced as toric codes by Kitaev [2003], are currently the most promising class of QEC schemes due to their relatively high error threshold (to satisfy the threshold theorem) and two-dimensional layout, which is well-suited for physical implementations [27]. Surface codes, as the name suggests, are a type of topological quantum error-correcting code, meaning they encode information in global properties of the system, making the encoded information robust against local errors, including the gate errors that Shor's code protects against

Similarly to Shor's, the surface codes encode a single logical qubit into many physical qubits. The key difference is that these physical qubits are arranged on a two-dimensional lattice, with toric codes having periodic bounding conditions that give this grid the shape of a torus, and planar codes having no such boundaries. The two have identical behavior in most cases.

The key to surface codes is their use of *stabilizer measurements,* which serve to measure the parity of specific regions of the lattice to detect errors without directly measuring the data qubits, in another similarity to Shor's code. These stabilizers are typically defined around the faces of the lattice, called *plaquettes*, and *vertices* of the grid, each involving four qubits. The measurements of these stabilizers yield error syndromes that indicate the location and type of errors. Errors manifest as anomalies on the lattice, and by tracking their movement, it is possible to infer and correct the underlying physical errors [20].

A significant advantage of surface codes over the earlier CSS codes is their higher error threshold, meaning they can tolerate higher physical error rates while still achieving fault-tolerant computation. This is partly due to the nature of encoding under a global system state, which makes them inherently more resilient to local noise while still being feasible to implement physically due to being two-dimensional.

Of course, surface codes also face challenges in their practical implementation. As it stands, the primary problem is the significant qubit overhead in encoding logical qubits. Encoding a single logical qubit reliably often requires hundreds to thousands of physical qubits, depending on gate fidelity and desired fault protection. Despite these challenges, ongoing research is focused on optimizing surface code implementations and exploring variants that might reduce the qubit overhead and simplify logical operations [28] [29].

5 Discussion

As it stands, the theory of quantum computation is far ahead of any physical implementations we can currently employ. Current hardware has been categorized by Preskill [2018] as belonging to the *Noisy Intermediate-Scale Quantum*, or *NISQ* era. The term "intermediate-scale" refers to the size of quantum computers possible to physically construct, which is on the scale of around 1000 qubits, with "noisy" referring to a lack of fault-tolerance. Modern hardware is highly susceptible to environmental interference, and while we are still largely in the NISQ era, there are ongoing developments pushing beyond its boundaries, with methods to exceed 1000 qubits currently being explored [30]. This signifies a gradual movement towards larger and potentially less noisy systems, though full fault-tolerance remains a significant engineering challenge. Furthermore, the threshold theorem relies on an assumption of independent errors, meaning that errors are assumed to appear independently of space (errors of a qubit do not influence the errors of another qubit), time (errors cannot be correlated through time), locality (errors of a qubit affect only a very small number of its neighbors), and type (bit-flip, phase-flip, or some combination). This highly simplified model of quantum errors is often challenged in real-world quantum hardware where errors are often apparently correlated. This has led to not-so-insignificant counterargument, for instance that offered by Dyakonov [2006], suggesting that fault-tolerant quantum computing may not be possible in reality [31].

While optimists of quantum fault-tolerance have had little more than the hope provided by the threshold theorem for the last two and a half decades, recent research by Acharya et. al [2023] has experimentally shown that encoding a logical qubit with more physical qubits actually decreases the observed error rates, empirically proving that while correlated errors are a challenge, sophisticated QEC codes can overcome them using current error models. While this result implies that fault-tolerant quantum computing may be possible soon, the authors admit that hardware error rates must improve by at least 20% to achieve scalability, with further experimentation required to validate this estimation against different code sizes and runtime durations [32]. In even more recent news, Bravji et. al [2024] have managed to create a QEC code that is a whopping 10 times more efficient than the surface codes previously discussed, protecting 12 logical qubits for nearly one million syndrome measurements using only 288 physical qubits in total [3]. This monumental discovery means that fault-tolerance may be possible to physically achieve before we even surpass NISQ-era quantum computers, the importance of which cannot be overstated.

To conclude, quantum computing has the spectacular potential to solve many problems more efficiently than the most powerful supercomputer could ever hope to. In this survey, we have introduced the principles of quantum mechanics that allow for this efficiency, as well as the unique restraints that they impose on quantum algorithm design. We then attempted to analyze Shor's Algorithm for factoring integers as closely as possible in order to demonstrate the application of these techniques, with a discussion on its feasibility and limitations. We then introduced the concept of quantum error correction, noting how classical error correction techniques are fundamentally impossible to adapt to quantum computers, and detailed the first code that was able to overcome this limitation. We then provided an overview of more recent research into the field, focusing on surface codes as the most widely studied class of error correcting schemes. We ended with a discussion on fault-tolerant quantum computation, asserting that despite decades of physical limitations, the recent verification of the benefits of QEC show that optimism for near-term fault-tolerance is absolutely warranted.

6 Acknowledgments

I would like to thank professor Petr Vojtechovsky for advising this work, including valuable discussion and refinements. This thesis was completed for undergraduate distinction in Mathematics at the University of Denver.

References

[1] P.W. Shor. "Algorithms for quantum computation: discrete logarithms and factoring". In: *Proceedings 35th Annual Symposium on Foundations of Computer Science*. 1994, pp. 124–134. DOI: 10.1109/SFCS.1994.365700.

[2] David A. Rower et al. "Suppressing Counter-Rotating Errors for Fast Single-Qubit Gates with Fluxonium". In: *PRX Quantum* 5 (4 Dec. 2024), p. 040342. DOI: 10.1103/PRXQuantum.5.040342. URL: https://link.aps.org/doi/10.1103/PRXQuantum.5.040342.

[3] Sergey Bravyi et al. "High-threshold and low-overhead fault-tolerant quantum memory". In: *Nature* 627.8005 (Mar. 2024), pp. 778–782. ISSN: 1476-4687. DOI: 10.1038/s41586-024-07107-7. URL: http://dx.doi.org/10.1038/s41586-024-07107-7.

[4] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[5] M. Sipser. *Introduction to the Theory of Computation*. Cengage Learning, 2005. ISBN: 9780534950972. URL: https://books.google.com/books?id=SV2DQgAACAAJ.

[6] Adriano Barenco et al. "Elementary gates for quantum computation". In: *Physical Review A* 52.5 (Nov. 1995), pp. 3457–3467. ISSN: 1094-1622. DOI: 10.1103/physreva.52.3457. URL: http://dx.doi.org/10.1103/PhysRevA.52.3457.

[7] N.S. Yanofsky and M.A. Mannucci. *Quantum Computing for Computer Scientists*. Cambridge University Press, 2008. ISBN: 9780521879965. URL: https://books.google.com/books?id=eTT0FsHA5DAC.

[8] Dorit Aharonov. *A Simple Proof that Toffoli and Hadamard are Quantum Universal*. 2003. arXiv: quant-ph/0301040 [quant-ph]. URL: https://arxiv.org/abs/quant-ph/0301040.

[9] William L. Briggs and Van Emden Henson. *The DFT: An Owner's Manual for the Discrete Fourier Transform*. Society for Industrial and Applied Mathematics, 1995. DOI: 10.1137/1.9781611971514. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9781611971514. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611971514.

[10] D. Coppersmith. *An approximate Fourier transform useful in quantum factoring*. 2002. arXiv: quant-ph/0201067 [quant-ph]. URL: https://arxiv.org/abs/quant-ph/0201067.

[11] Daniel J. Bernstein. "Detecting perfect powers in essentially linear time". In: *Math. Comput.* 67.223 (July 1998), pp. 1253–1283. ISSN: 0025-5718. DOI: 10.1090/S0025-5718-98-00952-1. URL: https://doi.org/10.1090/S0025-5718-98-00952-1.

[12] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. "PRIMES is in P". English. In: *Ann. Math. (2)* 160.2 (2004), pp. 781–793. ISSN: 0003-486X. DOI: 10.4007/annals.2004.160.781.

[13] Peter W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". In: *SIAM Journal on Computing* 26.5 (Oct. 1997), pp. 1484–1509. ISSN: 1095-7111. DOI: 10.1137/s0097539795293172. URL: http://dx.doi.org/10.1137/S0097539795293172.

[14] Varqa Abyaneh. *Harnessing the Quantum Zeno Effect: A New Approach to Ion Trapping*. 2024. arXiv: 2402.06398 [quant-ph]. URL: https://arxiv.org/abs/2402.06398.

[15] G. H. Hardy and E. M. Wright. *An introduction to the theory of numbers.* English. Oxford: Clarendon Press. xvi, 403 p. (1938). 1938.

[16] Martin Ekerå. "On the Success Probability of Quantum Order Finding". In: *ACM Transactions on Quantum Computing* 5.2 (May 2024), pp. 1–40. ISSN: 2643-6817. DOI: 10.1145/3655026. URL: http://dx.doi.org/10.1145/3655026.

[17] R.L. Rivest, A. Shamir, and L. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". In: *Communications of the ACM* 21.2 (1978), pp. 120–126.

[18] Linus Gasser. "Post-quantum Cryptography". In: *Trends in Data Protection and Encryption Technologies*. Ed. by Valentin Mulder et al. Cham: Springer Nature Switzerland, 2023, pp. 47–52. ISBN: 978-3-031-33386-6. DOI: 10.1007/978-3-031-33386-6_10. URL: https://doi.org/10.1007/978-3-031-33386-6_10.

[19] Enrique Martín-López et al. "Experimental realization of Shor's quantum factoring algorithm using qubit recycling". In: *Nature Photonics* 6.11 (Oct. 2012), pp. 773–776. ISSN: 1749-4893. DOI: 10.1038/nphoton.2012.259. URL: http://dx.doi.org/10.1038/nphoton.2012.259.

[20] D.A. Lidar and T.A. Brun. *Quantum Error Correction*. Cambridge University Press, 2013. ISBN: 9780521897877. URL: https://books.google.com/books?id=bM5KngEACAAJ.

[21] G.G.L. Guardia. *Quantum Error Correction: Symmetric, Asymmetric, Synchronizable, and Convolutional Codes*. Quantum Science and Technology. Springer International Publishing, 2020. ISBN: 9783030485504. URL: https://books.google.com/books?id=IsmGzQEACAAJ.

[22] Peter W. Shor. "Scheme for reducing decoherence in quantum computer memory". In: *Phys. Rev. A* 52 (4 Oct. 1995), R2493–R2496. DOI: 10.1103/PhysRevA.52.R2493. URL: https://link.aps.org/doi/10.1103/PhysRevA.52.R2493.

[23] Asher Peres. "Reversible logic and quantum computers". In: 32.6 (Dec. 1985), pp. 3266–3276. DOI: 10.1103/PhysRevA.32.3266.

[24] Dorit Aharonov and Michael Ben-Or. *Fault-Tolerant Quantum Computation With Constant Error Rate*. 1999. arXiv: quant-ph/9906129 [quant-ph]. URL: https://arxiv.org/abs/quant-ph/9906129.

[25] Emanuel Knill, Raymond Laflamme, and Wojciech H. Zurek. "Resilient quantum computation: error models and thresholds". In: *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 454.1969 (Jan. 1998), pp. 365–384. ISSN: 1471-2946. DOI: 10.1098/rspa.1998.0166. URL: http://dx.doi.org/10.1098/rspa.1998.0166.

[26] A.Yu. Kitaev. "Fault-tolerant quantum computation by anyons". In: *Annals of Physics* 303.1 (Jan. 2003), pp. 2–30. ISSN: 0003-4916. DOI: 10.1016/s0003-4916(02)00018-0. URL: http://dx.doi.org/10.1016/S0003-4916(02)00018-0.

[27] A. Yu. Kitaev. "Quantum computations: algorithms and error correction". In: *Russian Mathematical Surveys* 52.6 (1997), pp. 1191–1249.

[28] Craig Gidney et al. *Yoked surface codes*. 2023. arXiv: 2312.04522 [quant-ph]. URL: https://arxiv.org/abs/2312.04522.

[29] Rui Chao et al. "Optimization of the surface code design for Majorana-based qubits". In: *Quantum* 4 (Oct. 2020), p. 352. ISSN: 2521-327X. DOI: 10.22331/q-2020-10-28-352. URL: http://dx.doi.org/10.22331/q-2020-10-28-352.

[30] M. J. Weaver et al. *Scalable Quantum Computing with Optical Links*. 2025. arXiv: 2505.00542 [quant-ph]. URL: https://arxiv.org/abs/2505.00542.

[31] M. I. Dyakonov. *Is Fault-Tolerant Quantum Computation Really Possible?* 2006. arXiv: quant-ph/0610117 [quant-ph]. URL: https://arxiv.org/abs/quant-ph/0610117.

[32] Rajeev Acharya et al. "Suppressing quantum errors by scaling a surface code logical qubit". In: *Nature* 614.7949 (Feb. 2023), pp. 676–681. ISSN: 1476-4687. DOI: 10.1038/s41586-022-05434-1. URL: http://dx.doi.org/10.1038/s41586-022-05434-1.

A Appendix

A.1 Frequently Used Quantum Gates

Table 1: *Common single-qubit unitary transforms*

Name	Symbol	Matrix	Effect on $\lvert\psi\rangle = \alpha\lvert 0\rangle + \beta\lvert 1\rangle$
Hadamard	H	$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$	$\lvert\psi'\rangle = \frac{\alpha+\beta}{\sqrt{2}}\lvert 0\rangle + \frac{\alpha-\beta}{\sqrt{2}}\lvert 1\rangle$
Identity	I	$\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$	$\lvert\psi'\rangle = \alpha\lvert 0\rangle + \beta\lvert 1\rangle$
Pauli-X	X	$\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$	$\lvert\psi'\rangle = \beta\lvert 0\rangle + \alpha\lvert 1\rangle$
Pauli-Y	Y	$\begin{bmatrix}0 & -i\\ i & 0\end{bmatrix}$	$\lvert\psi'\rangle = -i\beta\lvert 0\rangle + i\alpha\lvert 1\rangle$
Pauli-Z	Z	$\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}$	$\lvert\psi'\rangle = \alpha\lvert 0\rangle - \beta\lvert 1\rangle$
Phase	S	$\begin{bmatrix}1 & 0\\ 0 & i\end{bmatrix}$	$\lvert\psi'\rangle = \alpha\lvert 0\rangle + i\beta\lvert 1\rangle$
$\pi/8$	T	$\begin{bmatrix}1 & 0\\ 0 & e^{i\pi/4}\end{bmatrix}$	$\lvert\psi'\rangle = \alpha\lvert 0\rangle + \beta e^{i\pi/4}\lvert 1\rangle$

Table 2: *Common dual-qubit unitary transforms*

Name	Symbol	Matrix	Effect on $\lvert\psi\rangle = a\lvert 00\rangle + b\lvert 01\rangle + c\lvert 10\rangle + d\lvert 11\rangle$
Controlled-NOT	CNOT	$\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{bmatrix}$	$\lvert\psi'\rangle = a\lvert 00\rangle + b\lvert 01\rangle + d\lvert 10\rangle + c\lvert 11\rangle$
Swap	SWAP	$\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$	$\lvert\psi'\rangle = a\lvert 00\rangle + c\lvert 01\rangle + b\lvert 10\rangle + d\lvert 11\rangle$
Controlled-Z	CZ	$\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & -1\end{bmatrix}$	$\lvert\psi'\rangle = a\lvert 00\rangle + b\lvert 01\rangle + c\lvert 10\rangle - d\lvert 11\rangle$
Controlled-phase	CP(θ)	$\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & e^{i\theta}\end{bmatrix}$	$\lvert\psi'\rangle = a\lvert 00\rangle + b\lvert 01\rangle + c\lvert 10\rangle + de^{i\theta}\lvert 11\rangle$

Note the matrices of the Toffoli gate (CCNOT), Fredkin gate (CSWAP), and controlled-controlled phase shift gate (CCP(θ) are not included in this appendix due to their size, but they are natural extensions of the respective CNOT, SWAP, and CP(θ) matrices, similarly to how the CNOT matrix is an extension of the Pauli-X matrix, etc.