9-2022

Quantum Error Detection Without Using Ancilla Qubits

Nicolas Guerrero

QUANTUM ERROR DETECTION WITHOUT USING ANCILLA QUBITS

DISSERTATION

Nicolas Guerrero, Captain, USAF

AFIT-ENP-DS-22-S-044

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENP-DS-22-S-044

QUANTUM ERROR DETECTION WITHOUT USING ANCILLA QUBITS

DISSERTATION

Presented to the Faculty

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy in Applied Physics

Nicolas Guerrero, M.S.A.P.

Captain, USAF

September 15, 2022

AFIT-ENP-DS-22-S-044

QUANTUM ERROR DETECTION WITHOUT USING ANCILLA QUBITS

DISSERTATION

Nicolas Guerrero, M.S.A.P.
Captain, USAF

Committee Membership:

Dr. David Weeks, Ph.D
Chair

Dr. Laurence Merkle, Ph.D
Member

Dr. Anil Patnaik, Ph.D
Member

AFIT-ENP-DS-22-S-044

Abstract

Quantum computers are beset by errors from a variety of sources. Although quantum error correction and detection codes have been developed since the 1990s, these codes require mid-circuit measurements in order to operate. In order to avoid these measurements we have developed a new error detection code that only requires state collapses at the end of the circuit, which we call no ancilla error detection (NAED). We investigate some of the mathematics behind NAED such as which codes can detect which errors. We then ran NAED on three separate types of circuits: Greenberger–Horne–Zeilinger circuits, phase dependent circuits, and a quantum approximate optimization algorithm running the max cut problem. In total, we used the IBMQ quantum computers over 325 million times and were able to show that NAED can be used to improve the performance of the quantum computers. Additionally, we present generalized logical encodings and gates as well as proofs of the fidelity of these gates.

Table of Contents

Page

Page

List of Figures

List of Tables

QUANTUM ERROR DETECTION WITHOUT USING ANCILLA QUBITS

I. Introduction, background, and literature review

1.1 Introduction

Quantum computing promises to be a natural extension to classical computational architecture. While these exotic machines can exponentially speed up key algorithms over classical machines, they are highly prone to various errors that do not effect classical machines. Even more disturbingly, classical error correction does not work for correcting qubit errors. Luckily, quantum error detection and correction codes have been developed since the 1990s and can theoretically correct any quantum error.

Initially, we wished to investigate these codes using the IBMQ quantum systems. However, when we started our experiments the IBM systems were unable to perform mid-circuit measurements, a key part of all previous quantum error correction and detection codes. Thus, we developed a new error detection code which does not require these measurements. We experimented with this code in various different situations/circuits and were able to improve the overall performance of the algorithms.

1.2 The early history of quantum computing

In 1980, Benioff published an unassuming paper relating Turing machines to Hamiltonians of certain quantum systems [1]. He showed that for any Turing machine, and any N steps of this machine, that there exists an initial state $\psi(0)$ and some Hamiltonian H such that

$$\psi(t) = \exp(-itH)\psi(0) \tag{1}$$

describes the Turing machine. At times $t_1, t_2, ..., t_N$ the state $\psi(t_i)$ matches the state of the Turing machine at the ith step. Additionally, this Hamiltonian and initial state can be made in such a way that the quantum system is stable for an arbitrary amount of time around each t_i. This paper introduced a key idea: that qubits (although they were not called that then) represent classical data. In his paper each qubit state corresponded to a particular letter from an arbitrary alphabet, although we now use qubits with two spins where each spin corresponds to a binary bit. By combining these two seemingly separate fields of study, computer science and quantum mechanics, Benioff had laid the foundation for the nascent field of quantum computing.

The next significant event for quantum computing occurred during the 1981 conference on the physics of computation. During his keynote lecture, Feynman discussed how one could model a quantum system using a classical computer [2]. The issue he presented was that many particle systems could not be efficiently simulated on classical computers. Thus, if one hoped to simulate a quantum system, a classical computer could only determine macroscopic properties from other macroscopic properties. To get around this, Feynman proposed a 'quantum computer' which would be able to efficiently simulate a quantum system. A researcher would set up their quantum computer to model a desired system, run the quantum computer, and then tabulate outputs in order to learn something about the original problem. From this talk it is clear that Feynman envisioned quantum computers not as a generalization of classical computers but rather exotic experimental setups that would only act as an additional tool for researchers.

The jump from experiment to quantum computation happened in 1985 when Deutsch published a paper on a theoretical quantum computer [3]. He showed that

such a machine would not only be able to model quantum systems but also perform any computation that a classical computer could. He also introduced many terms/practices that are still in use today: computational basis states are the set of eigenvectors which span the Hilbert space over the physical qubits and the spectrum of two state particles is defined to be $|0\rangle$ and $|1\rangle$ rather than $|-1/2\rangle$ and $|1/2\rangle$. Several years later, he and Jozsa would produce one of the first demonstrative quantum algorithms.

The Deutsch-Jozsa algorithm [4] was the first quantum algorithm to demonstrate an advantage over classical computing. Imagine the following: suppose there is some function $f : \{0, 1, ..., 2n-1\} \to \{0, 1\}$ with the condition

$$f(0) = f(1) = ... = f(2n-1) \text{ or } |\{k : f(k) = 0\}| = n \tag{2}$$

Can we determine whether f is the constant function? On a classical computer, we can query $n+1$ different values of $f(n)$ to determine if it is constant or not. However, using the quantum algorithm we can determine the answer with a single query. That is, after running the algorithm we will definitively know whether or not the function is constant. Of course, the algorithm does not tell us whether the function is the constant 0 or 1 function, just that it is or is not constant. While interesting, this problem did not fundamentally change anything in the field as it is a 'toy problem' with no practical application.

The most important algorithm in all of quantum computing is undoubtedly Shor's algorithm [5], which factors a large inter into two smaller integers in polynomial time. Such an equivalent algorithm does not currently exist for classical computers, and the problem is suspected to be NP-intermediate. Although no proof of this claim exists, indeed such a proof would prove $P \neq NP$ [6], it is widely held to be true. It is not hyperbolic to say that if Shor's algorithm is ever implemented at a useful size,

then the vast majority of digital encryption on Earth would be in danger. Of course, the algorithm is not yet implementable for any meaningful size. For example, to factor a number with 2048 bits (RSA-2048) would take around 4000 qubits and over 100 million gates [7]. Without quantum error correction, a useful implementation of Shor's algorithm is a near certain impossibility although small instances have been run with some success [8, 9, 10, 11].

1.3 Errors in quantum computing

In his 1995 paper *Is Quantum Mechanics Useful?* [12], Rolf Landauer describes two main issues with quantum computing. First, there are "manufacturing" issues, or things that can theoretically be controlled by a researcher in a closed system. In quantum computers, qubits are controlled by some outside, classical source described by a Hamiltonian operator on the quantum system. In the world of pure theoretical mathematics/physics, these qubits and operators can be exactly defined and generally provide a full description of the quantum system. Of course, real life is not so tidy. Even in a completely isolated system, we cannot produce these Hamiltonians such that the difference between the theoretical operator acting on theoretical qubits and experimentally applied operator acting on physical qubits is negligible for anything other than the smallest circuits.

This was an issue in 1995 and it is still an issue today. In this 2019 paper *Not all qubits are created equal* [13] the authors discuss variable error rates for different single and multi-qubit gates on the IBMQ systems. They show that the variability in these error rates, not just the error rates themselves, can have a large effect on the reliability of the quantum computer.

Material issues are also a source of errors in quantum computers. In 2013 Eckstein and Levy published a paper describing the various material issues facing several

common types of quantum computation [14]. Quantum dots are highly sensitive to charge fluctuation noise which is mainly mitigated through more precise manufacturing techniques. Superconducting qubits are sensitive to a host of material defects which can eventually dephase/decohere a quantum state [15]. Trapped ions suffer from defects in the electrodes which hold the qubits. These defects can be errantly charged, simulating an operation on the ion [16]. Of course, there has also been significant progress in addressing these material issues. For example, a 2019 paper in Nature [17] discusses the effects that material defects have on quantum computing and propose a potential mitigation technique during the manufacturing process. Additionally, they claim that their electric field spectroscopy technique can also be used to positively alter circuits that may have suffered degradation over long term use.

The second issue Landauer describes is "restandardization", or the correction of errors accumulated over the course of a quantum computation. In a classical computer, this process is easily achievable. However, an issue arises for error correction of a quantum system. Correction implies the erasure of the incorrect information, erasure implies energy loss, which in turns contradicts the idea of a Hamiltonian system. Landauer was very familiar with this idea of a reversible computation as he had previously written a seminal paper on the classical analogue as well as the drawbacks such a system might encounter [18]. Of course, we now know that quantum error correction codes do exist, and the vast majority of this document will be used to explore these and similar codes.

In fact, there is a third issue that Landauer nor anyone else considered until relatively recently: systemic environmental errors. In an ideal universe, we would hope that we could sufficiently isolate a quantum computer so as to ignore all other possible outside sources of error. Unfortunately, this is not the case. Vacuum fluctuations, gravity waves, and cosmic rays all have the potential to produce seemingly random

errors in quantum computations. Although these types of errors are only beginning to be understood [19, 20, 21, 22], they create a problem without an easy solution for quantum computing.

1.4 Error correction codes

According to Landauer's principle, error correction (whether in a classical or quantum system) requires the dissipation of some energy from the system [18]. This would seem to suggest that quantum error correction is impossible, as any energy released from the system contradicts the assumptions of a Hamiltonian system. Of course, we now know there are ways of bypassing such restrictions. Quantum error correction and detection work by assigning large groups of physical qubits to logical qubits. These logical qubits are operated on by logical gates (see section 2.1.7 for complete definition) and produce desired logical states. In this section, we will describe several different error correction and detection schemes which utilize mid-circuit measurement. These mid-circuit measurements produce a syndrome while simultaneously collapsing the wave function into a certain set of possible states. The syndrome can then be used to declare an error has occured or correct the error (in the case of detection or correction respectively).

1.4.1 The Shor code

The first error correction code was Shor's 9-qubit error correction code. The idea behind this code was to combine two other correction codes: a bit flip code and a phase code, each using three physical qubits. The bit code can be described with the following logical states

$$|0\rangle_L = |000\rangle \text{ and } |1\rangle_L = |111\rangle \tag{3}$$

Obviously, if a bit flip were to occur then the three qubits will be in a mixed state of $|0\rangle$s and $|1\rangle$s. Interestingly, this code was actually first outline by Peres in 1985 [23]. Although this code does not detect phase errors, it suited his purposes as he was interested in performing classical computations on a quantum computer. In a similar manner, the phase code is given by

$$|0\rangle_L = \frac{1}{\sqrt{8}}(|0\rangle + 1|\rangle)^{\otimes 3} \text{ and } |1\rangle_L = \frac{1}{\sqrt{8}}(|0\rangle - 1|\rangle)^{\otimes 3} \tag{4}$$

Unfortunately, this code cannot detect bit flip errors.

Combining these two codes gives us the full 9 qubit Shor code. As the name implies, this error correction code uses 9 physical qubits to encode the states

$$|0\rangle_L = \frac{1}{\sqrt{8}}(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle) \tag{5}$$

$$|1\rangle_L = \frac{1}{\sqrt{8}}(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle) \tag{6}$$

These logical states can be encoded and decoded using the circuit in Fig. 1. This encoding circuit takes a qubit in some state $|\psi\rangle$ and 8 qubits in the state $|0\rangle$ and outputs a single logical state $|\psi\rangle_L$. To decode this state, simply perform the circuit in reverse. We can create logical gates using these encoding and decoding circuits in the following manner: decode the logical state to a single qubit, perform whatever gate we desire on the single qubit (or multiple qubits if it is a multi-qubit gate), and recode the logical state using the encoding circuit. Of course, it is probable that producing logical gates in this manner is inefficient: the encoding and decoding alone takes at least 16 physical C_x gates.

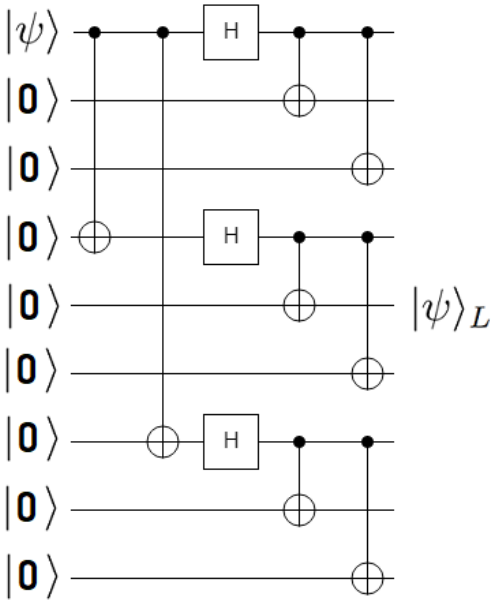

Figure 1: The encoding circuit for a single logical qubit. The input is 1 qubit in some state $|\psi\rangle$ and 8 qubits in the $|0\rangle$ state. The output is $|\psi\rangle_L$. To decode a logical state, simply perform this circuit in reverse.

Performing error correction is a four step process: three bit flip error checks and one phase error check. To begin, label the qubits in Fig. 1 from 0 to 8 from top to bottom. Now implement the bit flip correction circuit on the following blocks of qubits: q_0 through q_2, q_3 through q_5, and q_6 through q_8. For example, for the first block the circuit is shown in Fig. 2

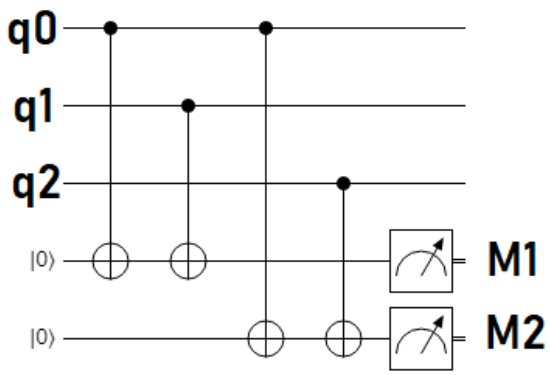

Figure 2: The bit flip error correction circuit for qubits q_0 through q_2. Using this circuit we can extract a syndrome and after referencing Table 1 we can correct any error that might have occured.

After getting measurements M_1 and M_2, we can consult the Table 1 to see what corrections we need to perform.

M_1	M_2	Correction
0	0	Nothing
0	1	σ_x on q_2
1	0	σ_x on q_1
1	1	σ_x on q_0

Table 1: Bit flip corrections for qubits q_0 through q_2.

The same procedure is performed on blocks q_3 through q_5 and q_6 through q_8. A different circuit is used to correct for phase errors and is shown in Fig. 3.

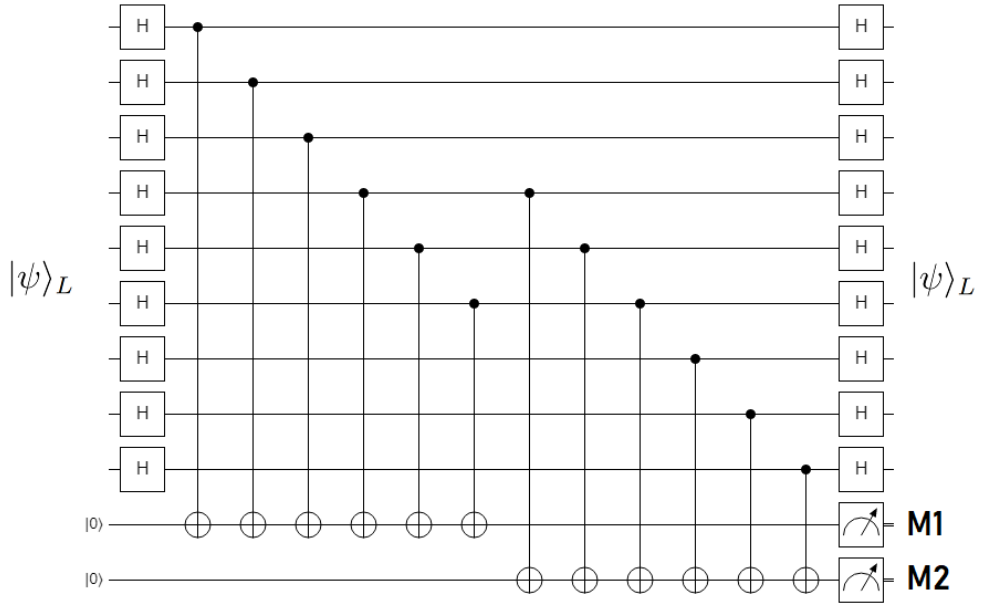

Figure 3: The phase error correction circuit. We can correct any phase errors after extracting the syndrome and consulting Table 2.

After using this circuit to extract a syndrome, consult Table 2 for correction.

M_1	M_2	Correction
0	0	Nothing
0	1	σ_z on $q_6q_7q_8$
1	0	σ_z on $q_0q_1q_2$
1	1	σ_z on $q_3q_4q_5$

Table 2: Bit flip corrections for qubits q_0 through q_2.

Overall, each cycle of error correction will take at least 24 physical C_x gates, 18 Hadamard gates, and 8 mid-circuit measurements.

1.4.2 The 5-qubit correction code

Developed in 1996, the DiVincenzo-Shor [24] 5-qubit code is the smallest possible error correction code [25]. It is impossible to perform full error correction with less than five physical qubits per logical qubit. The encoding is defined by the code words

$$4|0\rangle_L = |00000\rangle + |11000\rangle + |01100\rangle + |00110\rangle$$
$$+|00011\rangle + |10001\rangle - |10100\rangle - |01010\rangle$$
$$-|00101\rangle - |10010\rangle - |01001\rangle - |11110\rangle$$
$$- |01111\rangle - |10111\rangle - |11011\rangle - |11101\rangle \quad (7)$$

$$4|1\rangle_L = |11111\rangle + |00111\rangle + |10011\rangle + |11001\rangle$$
$$+|11100\rangle + |01110\rangle - |01011\rangle - |10101\rangle$$
$$-|11010\rangle - |01101\rangle - |10110\rangle - |00001\rangle$$
$$- |10000\rangle - |01000\rangle - |00100\rangle - |00010\rangle \quad (8)$$

To encode these states we use the complicated circuit in Fig. 4 [26]

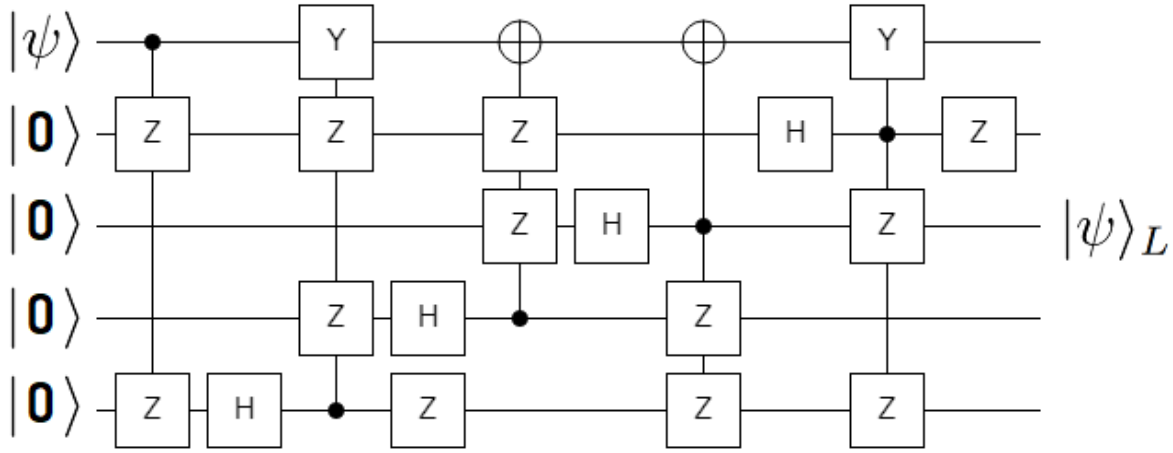

Figure 4: The encoding circuit for the 5-qubit DiVincenzo-Shor code. To decode a logical state perform this circuit in reverse.

We can use this encoding circuit in a similar manner as the 9-qubit Shor code to make

logical gates. Decode a logical state by performing the encoding circuit in reverse, perform whatever gates you want on the single qubit states, then recode using the circuit above. In Fig. 5, we present the syndrome extraction circuit.

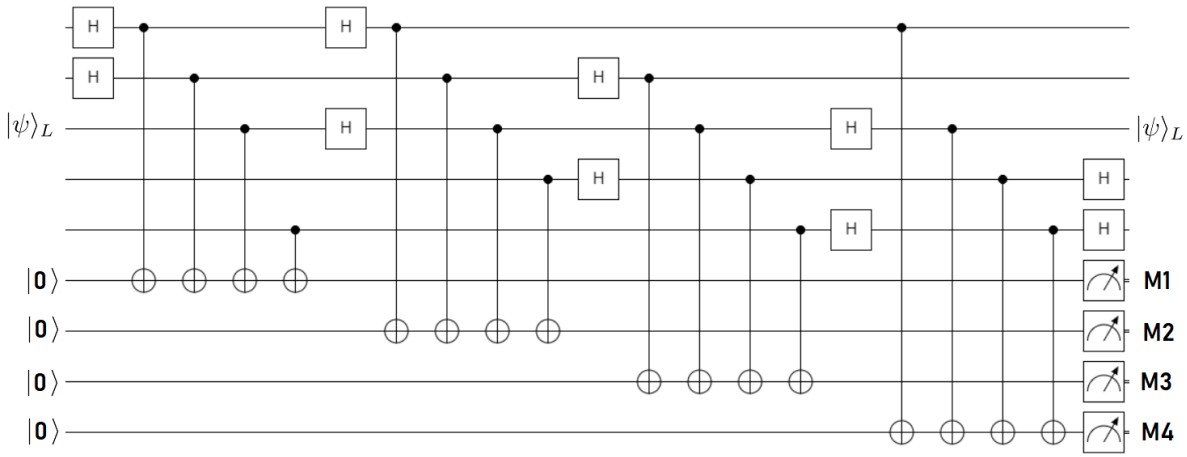

Figure 5: The syndrome extraction circuit for the 5-qubit DiVincenzo-Shor code. Consult Table 3 for any possible corrections.

After running this circuit, consult Table 3 for any possible corrections. Note that we are assuming the qubits are labeled top to bottom from q_0 to q_4 and that the measurements are arranged (M_1, M_2, M_3, M_4).

(M_1, M_2, M_3, M_4)	Correction
0000	Nothing
0001	σ_z on q_4
0010	σ_x on q_1
0011	σ_z on q_3
0100	σ_x on q_3
0101	σ_x on q_0
0110	σ_z on q_2
0111	σ_y on q_3

(M_1, M_2, M_3, M_4)	Correction
1000	σ_z on q_0
1001	σ_x on q_2
1010	σ_x on q_4
1011	σ_y on q_4
1100	σ_z on q_1
1101	σ_y on q_0
1110	σ_y on q_1
1111	σ_y on q_2

Table 3: Reference table for required corrections for the 5-qubit DiVincenzo-Shor code.

This code has been experimentally verified in small instances [27, 28, 29].

1.4.3 Distance 3 surface correction code

Surface codes represent a different type of error correction scheme than the two previously described. Initially described by Kitaev in 1997 [30], they represent one of the most promising error correction codes for noisy intermediate-scale quantum (NISQ) computers [31, 32, 33]. It is illuminating to present the logical qubit topology and only after describe the code. This topology is given in Fig. 6 below:

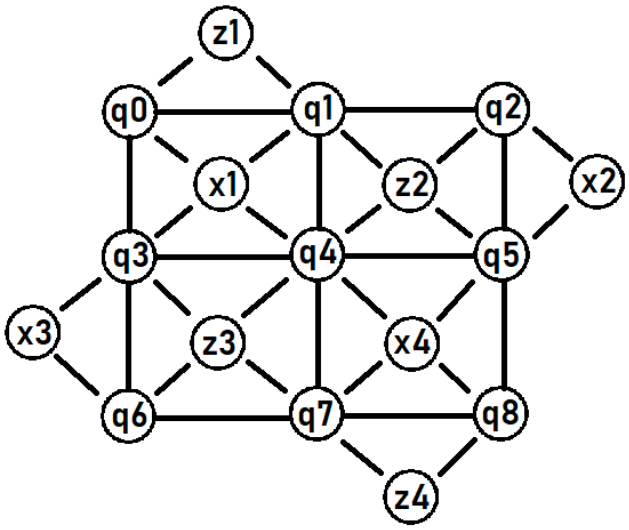

Figure 6: This code uses nine physical qubits (qubits q_0 through q_8) to encode a single logical qubit as well as two sets of four ancilla qubits each: z_1 through z_4 and x_1 through x_4.

This code uses nine physical qubits (q_0 through q_8) to encode a single logical qubit. There are four circuits used to extract syndromes, examples of which are presented below:

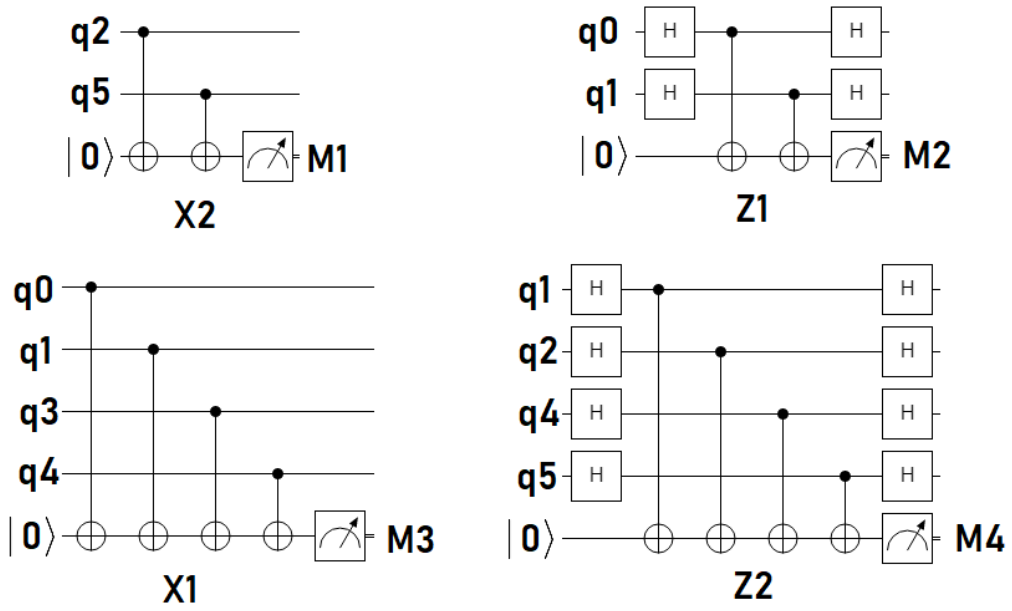

Figure 7: Example syndrome extraction for the x_1, x_2, z_1, and z_2 ancilla qubits. The other four ancilla qubits are associated with the same circuits with different physical qubits attached. If any of M_1 through M_4 are measured in the $|1\rangle$ state then an error has occured. Correcting the error is dependent on which M_i are in the $|1\rangle$ state.

These syndrome extraction circuits are only powerful enough to tell us if an error has occured, which happens when any of the ancilla qubits are measured in the $|1\rangle$ state. To get full error correction, we need to analyze which ancilla qubits are in the $|1\rangle$ state. The following example is illuminating:

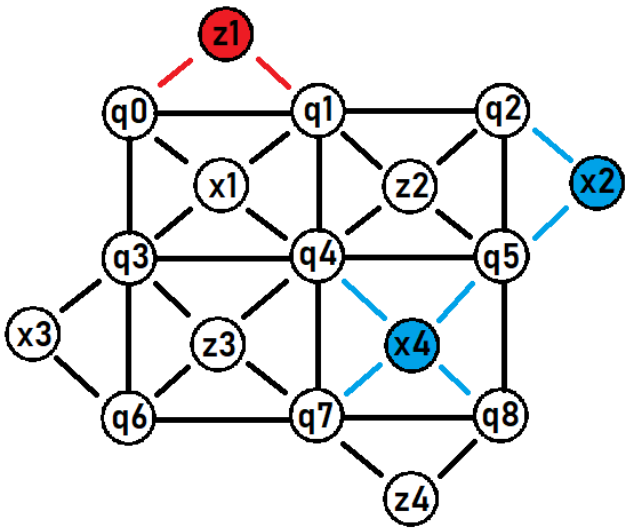

Figure 8: Suppose that z_1 is measured in the state $|1\rangle$ and z_2, z_3, and z_4 are measured in the state $|0\rangle$. We can conclude we need to operate on q_0 with σ_z as this is the most likely source of σ_z error. A similar argument works for the blue errors x_2 and x_4. With this syndrome, we may conclude we need to correct q_5 with σ_x.

If we measure z_1 to be in the $|1\rangle$ state, then we can conclude we need to operate on q_0 with σ_z as this is the most likely source of σ_z error. In a similar manner, we know that q_5 had a σ_x error since both x_2 and x_4 were measured in the $|1\rangle$ state and x_1 and x_3 were measured in the $|0\rangle$ state. These interaction have been experimentally verified in a variety of circumstances [34, 35].

It is valuable to discuss generalizing this type of surface code in two ways: larger distances and more exotic topologies. The distance 3 surface code is so named since the nine physical qubits are arranged in a 3×3 grid. An obvious generalization is to expand this idea to larger grids (see 1.5.2 for the distance 2 code). In fact, this generalization provides a nice bonus: larger surface codes are more resistant to physical errors. The main reason this occurs is that larger distances are less likely to produce misidentified qubit errors [36, 37]. So far, we have only discussed errors in the context of single error occurrences. Of course, this is powerful assumption to

make as it would seem natural that multiple errors could occur near instantaneously. If this were to occur, it is possible that our error correction codes would recognize an incorrect error or not recognize any error had occured. These misidentified errors would then be corrected (or ignored) and the desired qubit state would be destroyed. For example, suppose that the three qubit surface code reports errors on the x_1, x_3, and x_4 ancilla qubits (see Fig. 9). Then the possible σ_x errors occurred either on q_3 and q_7 or q_4 and q_6.

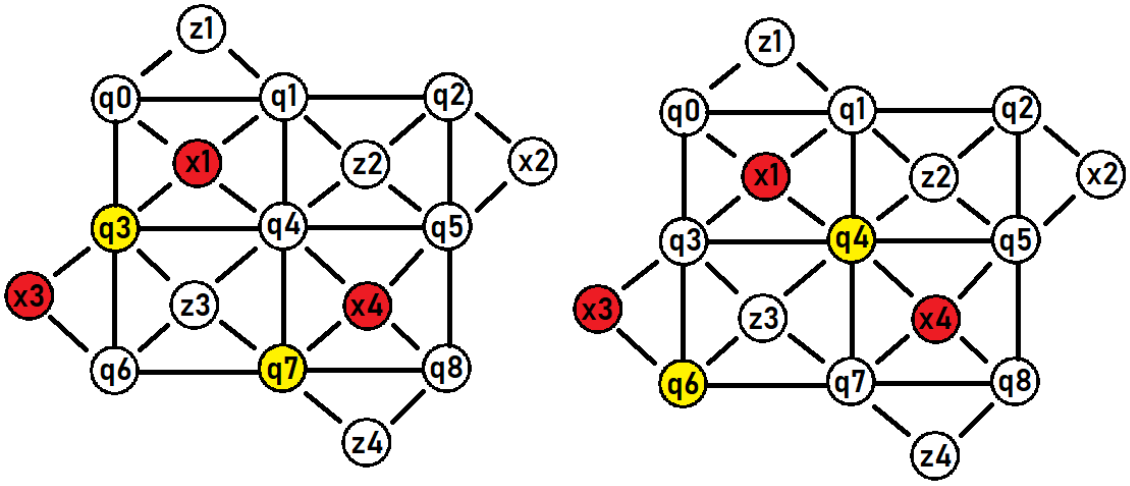

Figure 9: If the x_1, x_3, and x_4 ancilla qubits measure an error (red qubits), then either configuration is possible for double σ_x errors (yellow qubits).

The second way to generalize surface codes is with more exotic topologies. The distance 3 code discussed above (indeed, all distance n codes) are referred to as planar topologies. Perhaps the simplest exotic code is the toric code. This code takes a planar topology and then with the use of boundary conditions creates a topological torus. For example, one can turn a distance 3 planar code into a toric code by connecting the ancilla qubits in a certain way:

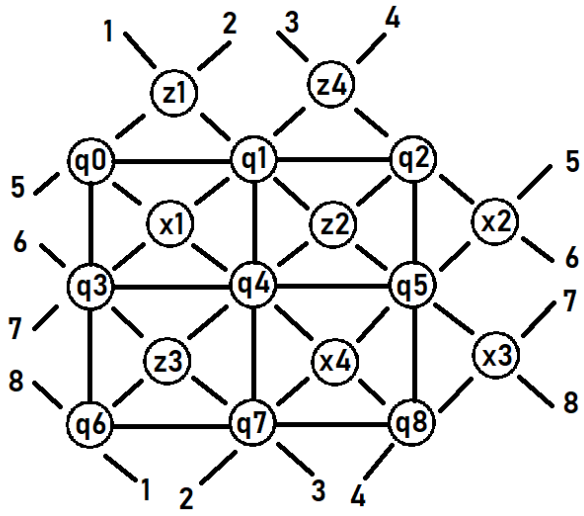

Figure 10: We can turn a distance 3 planar code into a toric code by connecting z_1 to q_6 and q_7, z_4 to q_7 and q_8, x_2 to q_0 and q_3, and x_3 to q_3 and q_6.

Wrap each side of the planar code and connect it to the opposite side. This toric code represents two logical qubits (instead of the single logical qubit for the planar code). These toric codes have been simulated [38] and experimentally verified [39, 40, 41] at small scales.

1.5 Error detection codes

In many ways, error detection codes are similar to error correction codes. They both use groups of physical qubits to define logical qubits and both utilize mid-circuit measurement of ancilla qubits. Error detection uses fewer physical qubits though and obviously cannot tell one how to correct a given error. It is useful to present a simple example to demonstrate the differences between correction and detection. Consider the encoding given by

$$|0\rangle_L = |01\rangle \text{ and } |1\rangle_L = |10\rangle \tag{9}$$

To see whether a bit-flip has occurred, we can run the circuit in Fig. 11

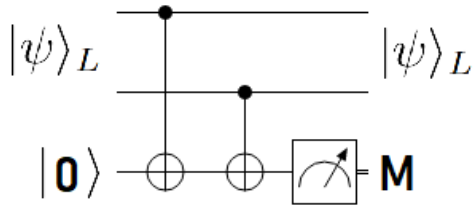

Figure 11: A simple bit flip error detection code for the code words in equation 9. If M is measured $|1\rangle$ then no error has occured, if it is measured $|0\rangle$ then a bit flip error has occured although we cannot say on which qubit.

If M is measured in the $|1\rangle$ state then no bit flip occured, if it is measured in the $|0\rangle$ state then a bit flip occured although we cannot say on which qubit.

1.5.1 The Knill code

The foundational paper on error detection was written by Knill in 2005 [42]. He popularized the term "post-selection" to mean measuring qubits and either accepting or rejecting a run based off of the measurement outcome. In his original paper, he described the smallest (in terms of physical qubits per logical qubit) [43] error detection code possible. The code words for this encoding are given by

$$|00\rangle_L = \frac{1}{\sqrt{2}}(|0000\rangle + |1111\rangle) \tag{10}$$

$$|01\rangle_L = \frac{1}{\sqrt{2}}(|1100\rangle + |0011\rangle) \tag{11}$$

$$|10\rangle_L = \frac{1}{\sqrt{2}}(|1010\rangle + |0101\rangle) \tag{12}$$

$$|11\rangle_L = \frac{1}{\sqrt{2}}(|0110\rangle + |1001\rangle) \tag{13}$$

Note that this error detection code works with pairs of logical qubits rather than single logical qubits. To encode this state we can use the circuit in Fig. 12.

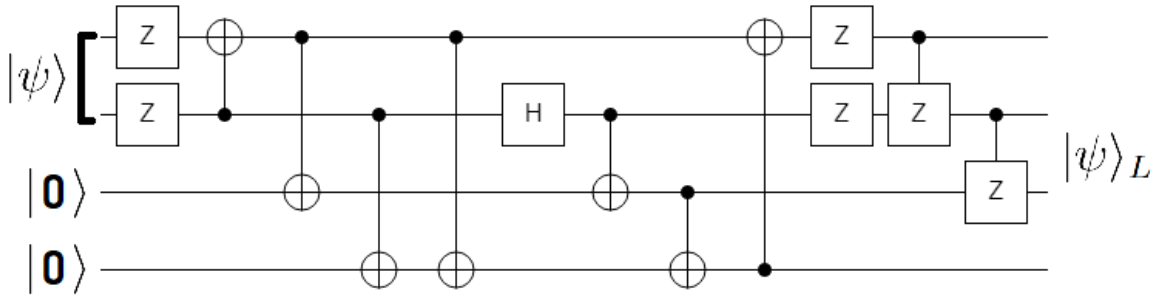

Figure 12: The encoding circuit for the 4-qubit Knill error detection encoding. To decode this state simply run this circuit in reverse.

To perform logical gates, one can decode the logical state, perform gates on the unencoded state, and reencode to a new logical state. We can extract a syndrome and perform error detection using the circuit in Fig. 13 below

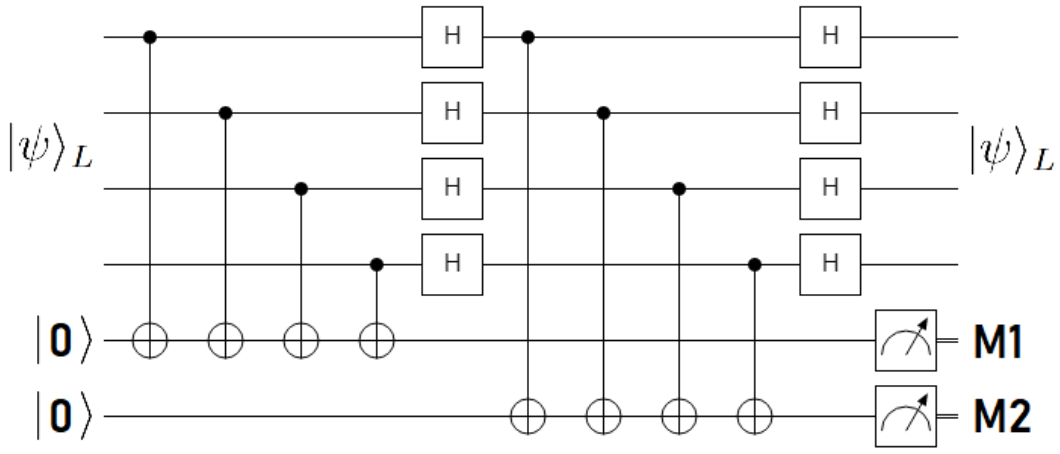

Figure 13: The syndrome extraction circuit for the 4-qubit Knill code. If either M_1 or M_2 measure in the state $|1\rangle$, then an error has occurred.

If either M_1 or M_2 measure in the $|1\rangle$ state, then an error has occured and the run is discarded. This code has been verified for instances of a single logical qubit [44].

1.5.2 Distance 2 surface code

The distance 2 surface code is very similar to distance 3 surface code. Here, four physical qubits create one logical qubit with code words

$$|0\rangle_L = \frac{1}{\sqrt{2}}(|0000\rangle + |1111\rangle) \tag{14}$$

$$|1\rangle_L = \frac{1}{\sqrt{2}}(|0101\rangle + |1010\rangle) \tag{15}$$

These physical qubits are accompanied by three ancilla qubits used for error detection. These physical qubits are arrange in the following manner:

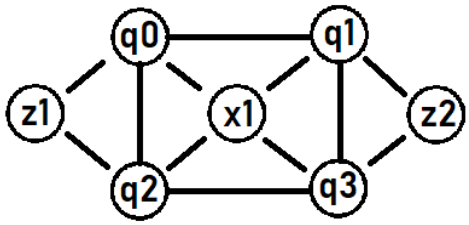

Figure 14: The four physical qubits and three ancilla qubits which compose the distance 2 surface code.

The syndrome extraction is also very similar to the distance 3 code. The extraction circuits are given in Fig. 15.

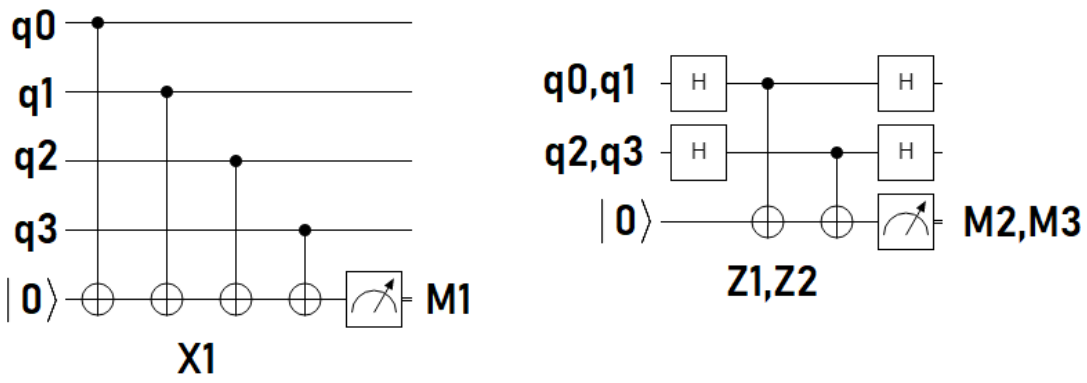

Figure 15: The syndrome extraction circuits for the distance 2 surface code. If either M_1, M_2, or M_3 are measured in the $|1\rangle$ state, then an error has occured.

After measuring the ancilla qubits, if any of them were measured in the $|1\rangle$ state then an error has occured and the run is discarded.

Encoding logical states and creating logical gates is a highly non-trivial problem. A recent Nature article [45] describes an algorithm to prepare any initial logical state $|\psi\rangle_L = \alpha|0\rangle_L + \beta|1\rangle_L$. To begin, hit q_0 and q_3 with the unitary matrices

$$\begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \text{ on } q_0 \tag{16}$$

$$\begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ e^{i\phi}\sin(\theta) & e^{i\phi}\cos(\theta) \end{pmatrix} \text{ on } q_3 \tag{17}$$

This produces the state

$$|\psi\rangle = [\cos(\theta)|0\rangle + \sin(\theta)|1\rangle]|0\rangle[\cos(\theta)|0\rangle + e^{i\phi}\sin(\theta)|1\rangle] \tag{18}$$

After implementing the syndrome extraction circuits from Fig. 15 we collapse the state and have a certain probability of being in the logical state

$$|\psi\rangle_L = \frac{1}{\sqrt{N}} \left(\cos(\theta)^2|0\rangle_L + e^{i\phi}\sin(\theta)^2|1\rangle_L\right) \tag{19}$$

(here N is the normalizing constant). This probability is given by

$$P = \frac{1}{2}\left(\cos(\theta)^4 + \sin(\theta)^4\right) \tag{20}$$

We know we are in the final logical state if the measurements M_2 and M_3 agree. this same article also describes how to perform single qubit logical gates [45]. They are implemented in the much the same way as the initial states: perform specific single qubit gates on the physical qubits q_0 through q_3. Perform the syndrome measurements circuits and collapse the wave function into the desired state.

1.5.3 No ancilla error detection (NAED)

No ancilla error detection (NAED) uses groups of physical qubits to represent logical qubits in much the same way as the previously described error correction and detection. It differs in the fact that there are no ancilla qubits or mid-circuit measurements. This is important as the IBMQ systems did not allow mid-circuit measurements at the start of this dissertation. This meant that it was impossible to experimentally verify any of the error correction or detection schemes detailed above. At this point though IBM has introduced mid-circuit measurement [46] rendering the whole issue moot.

As we will come to see, NAED can perform error detection with fewer physical qubits. However, the main downside to NAED is that it only gives a probability of detecting an error. Without ancilla qubits, it is possible to have an error occur, complete the error detection scheme, and then not learn that said error occured. In this document we will discuss NAED in detail: its encodings and decodings, logical

gates, and present experimental evidence of it in action.

1.6 Mathematical notation and similarity measure

1.6.1 Important mathematical notation

A word on the notation used throughout this document: let σ_x be the standard 2×2 Pauli x matrix. We will use the notation σ_i to represent the $2^Q \times 2^Q$ matrix

$$\sigma_i = I_{2^i} \otimes \sigma_x \otimes I_{2^{Q-i-1}} \tag{21}$$

where I_n is the $n \times n$ identity matrix. For example, for $Q = 3$ we have

$$\sigma_0 = \sigma_x \otimes I_4 \tag{22}$$

$$\sigma_1 = I_2 \otimes \sigma_x \otimes I_2 \tag{23}$$

$$\sigma_2 = I_4 \otimes \sigma_x \tag{24}$$

Note that qubits are counted starting from 0 rather than 1.

For any two physical qubits in a given circuit, the gate $C_x(i, j)$ is defined to be a C_x gate with control qubit i and target qubit j. Additionally, qubits other than i and j are assumed to be operated on by identity gates. The gate $C_x(i, j)$ represents a $2^n \times 2^n$ matrix, rather than a 4×4 matrix, where n will be obvious from context. As an example, for $n = 3$ physical qubits

$$C_x(1, 2) = I_2 \otimes C_x \tag{25}$$

$$C_x(0,2) = (I_2 \otimes SWAP)(C_x \otimes I_2)(I_2 \otimes SWAP) \tag{26}$$

$$C_x(1,0) = (SWAP \cdot C_x \cdot SWAP) \otimes I_2 \tag{27}$$

Here, both C_x and $SWAP$ are the standard 4×4 controlled-not and swap gates respectively. Finally, when using product notation for matrix multiplication, we will use the convention

$$\prod_{n=1}^{N} A_n = A_N A_{N-1} \cdots A_2 A_1 \tag{28}$$

1.6.2 Similarity measure

In order to measure how well a quantum algorithm works, we will employ a modified version of the total variation distance [47, 48] between probability measure denoted $\tau(A, B)$ for finite probability distribution functions (PDFs) A and B. Define the similarity between two finite PDFs to be

$$0 \leq \mu(A, B) = 100(1 - \tau(A, B)) = 100 - 50 \sum_{i=1}^{|A|} |A_i - B_i| \leq 100 \tag{29}$$

The similarity is equal to 0 if two PDFs are completely dissimilar and 100 if they are identical.

II. Mathematical foundations of NAED

Preamble

NAED provides not only an experimental framework with which to work with, but also a rich background of mathematical tools and discoveries. In this chapter, we will present some basic definitions, examples of useful concepts, and proofs of foundational facts.

2.1 Definitions

In this section we define several useful concepts for NAED as well as provide examples of each definition.

2.1.1 Logical states and encodings

In the most general case we use q physical qubits to define a single logical qubit. These encodings are defined as follows

$$|0\rangle_L = \sum_{n=0}^{2^q-1} a_n |n\rangle \tag{30}$$

$$|1\rangle_L = \sum_{n=0}^{2^q-1} b_n |n\rangle \tag{31}$$

with the stipulations that

$$\sum_{n=0}^{2^q-1} |a_n|^2 = \sum_{n=0}^{2^q-1} |b_n|^2 = 1 \tag{32}$$

$$\sum_{n=0}^{2^q-1} a_n \overline{b_n} = 0 \tag{33}$$

for some $a_i, b_i \in \mathbb{C}$.

Notation: Normally, we write out the logical states to define an encoding. For example

$$|0\rangle_L = \frac{|00\rangle + |01\rangle + |11\rangle}{\sqrt{3}} \tag{34}$$

$$|1\rangle_L = \frac{|00\rangle + |10\rangle + |11\rangle}{\sqrt{3}} \tag{35}$$

2.1.2 The code, error, and orthogonal sets

For a given logical encoding over q qubits and n logical qubits, define the set

$$(H^+)^n = \left\{ \sum_{i=0}^{2^n-1} a_i |i\rangle_L : \sum_{i=0}^{2^n-1} |a_i|^2 = 1 \right\} \tag{36}$$

and define the set

$$(H^-)^n = \mathcal{H}_{nq} - (H^+)^n \tag{37}$$

where $\mathcal{H}_{nq}$ is the Hilbert space over all nq-qubits. We say that $(H^+)^n$ is the set of code states for n logical qubits and $(H^-)^n$ is the set of error states. Additionally, define the orthogonal set

$$(H^\dagger)^n = \mathcal{H}_{nq}/(H^+)^n = \{|\phi\rangle \in \mathcal{H}_{nq} : \langle\phi|\psi\rangle = 0 \text{ for all } |\psi\rangle \in (H^+)^n\} \tag{38}$$

For the sake of notation, for $n = 1$ we will let $H^+ = (H^+)^1$, $H^- = (H^-)^1$, and $H^\perp = (H^\perp)^1$.

Example: Consider the encoding $|0\rangle_L = |01\rangle$ and $|1\rangle_L = |10\rangle$. Then

$$H^+ = \left\{ (0, a, b, 0)^T : |a|^2 + |b|^2 = 1 \right\} \tag{39}$$

$$H^- = \left\{(\epsilon, a, b, \delta)^T : |\epsilon| + |\delta| > 0\right\} \tag{40}$$

$$H^\dagger = \left\{(\epsilon, 0, 0, \delta)^T : |\epsilon|^2 + |\delta|^2 = 1\right\} \tag{41}$$

See Theorem 1 and Theorem 2 (sections 2.2.1 and 2.2.2) respectively for qualifiers for code and error states.

2.1.3 Flawless Encodings

We say that an encoding over Q physical qubits is k-flawless (for $k \leq Q$) if for all $|\psi\rangle \in H^+$

$$U_{(i_1,i_2,...,i_k)}|\psi\rangle \in H^+ \tag{42}$$

(where $U_{(i_1,i_2,...,i_k)}$ is some $2^k \times 2^k$ unitary matrix acting on the k qubits $i_1, i_2, ..., i_k$) implies $U_{(i_1,i_2,...,i_k)} = e^{i\theta} I_{2^k}$. That is, the $2^k \times 2^k$ identity matrix multiplied by some phase θ. The phase θ appears as we are allowed to multiple any quantum state by a global phase and get the same measurement out when we are done. As discussed in chapter I we generally assume that errors encountered during actual quantum computation effect single qubits at a time. Thus, for the rest of this paper, we will drop the prefix and simply let flawless denote 1-flawless and only consider single qubit errors. Then the definition of flawless becomes: for all $|\psi\rangle \in H^+$ and all $0 \leq n < Q$

$$I_{2^n} \otimes U \otimes I_{2^{Q-n-1}}|\psi\rangle \in H^+ \tag{43}$$

implies $U = e^{i\theta} I_2$ for some phase θ.

Example: An example of the smallest flawless encoding is given by

$$|0\rangle_L = \frac{1}{\sqrt{3}}\Big(|001\rangle + |010\rangle + |100\rangle\Big) \tag{44}$$

$$|1\rangle_L = \frac{1}{\sqrt{3}}\Big(|011\rangle + |101\rangle + |110\rangle\Big) \tag{45}$$

See Theorem 5 and Corollary 1 (sections 2.2.5 and 2.2.7) for proof.

2.1.4 Near-flawless encodings

Let E be some flawed encoding and let Ω be the set of 2×2 unitary matrices U such that

$$I_{2^n} \otimes U \otimes I_{2^{q-n-1}}|\psi\rangle \in H^+ \tag{46}$$

for some $|\psi\rangle \in H^+$ and $0 \leq n \leq q-1$. We say that $U \in \Omega$ is ignorable if either of the following conditions hold

1. $U = e^{i\phi} I_2$ for some $\phi \in \mathbb{R}$ (the trivial error)
2. $\mu(\{|\psi\rangle \in H^+ : I_{2^n} \otimes U \otimes I_{2^{q-n-1}}|\psi\rangle \in H^+ \text{ for any } 0 \leq n \leq q-1\}) = 0$

Here $\mu(S)$ denote the Lebesgue measure of the set S in the set H^+. Using this definition of ignorable, we define a near flawless encoding: an encoding E is near flawless if the following two conditions hold

1. E is flawed
2. Every $U \in \Omega$ (as defined above) is ignorable

Example: The following encoding is near flawless

$$|0\rangle_L = \frac{1}{2\sqrt{3}}\Big(3|00\rangle + |01\rangle + |10\rangle + |11\rangle\Big) \tag{47}$$

$$|1\rangle_L = \frac{1}{2}\Big(- |00\rangle + |01\rangle + |10\rangle + |11\rangle \Big) \tag{48}$$

See Theorem 7 and Theorem 8 (section 2.2.8 and 2.2.9) for details.

2.1.5 Natural representations

For a given encoding E over Q physical qubits, define the set of natural representations by

$$\chi(E) = \left\{ \begin{pmatrix} a_0 & b_0 & \dots \\ a_1 & b_1 & \dots \\ \vdots & \vdots & \ddots \\ a_{2^q-1} & b_{2^Q-1} & \dots \end{pmatrix} \in U(2^Q) \right\} \tag{49}$$

$\chi(E)$ is the set of $2^Q \times 2^Q$ unitary matrices with the property that the first and second columns are the $|0\rangle_L$ and $|1\rangle_L$ vectors respectively.

Example: Consider the encoding E defined by $|0\rangle_L = |01\rangle$ and $|1\rangle_L = |10\rangle$. Then

$$\chi(E) = \left\{ \begin{pmatrix} 0 & 0 & e^{i\delta}\cos(\theta) & -e^{i(\phi+\delta)}\sin(\theta) \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & e^{i(\lambda+\delta)}\sin(\theta) & e^{i(\lambda+\phi+\delta)}\cos(\theta) \end{pmatrix} : \delta, \theta, \lambda, \phi \in \mathbb{R} \right\} \tag{50}$$

2.1.6 Final States

We define a pair of valid final states $|0\rangle_F = \sum_{i=0}^{2^Q-1} c_i|i\rangle$ and $|1\rangle_F = \sum_{i=0}^{2^Q-1} d_i|i\rangle$ as orthonormal vectors (similar to our logical states) with the added stipulations that $\sum_{i=0}^{2^Q-1} |c_i d_i| = 0$ and for at least one $0 \leq i \leq 2^Q - 1$ we have $c_i = d_i = 0$.

2.1.7 Logical Gates

We say a gate U is a logical gate V if for all possible states $|\psi\rangle$

$$V|\psi\rangle = |\phi\rangle \Rightarrow U|\psi\rangle_L = |\phi\rangle_L \tag{51}$$

where $|\psi\rangle_L$ and $|\phi\rangle_L$ are the logical versions of $|\psi\rangle$ and $|\phi\rangle$ respectively.

Example: Consider the encoding given by $|0\rangle_L = |01\rangle$ and $|1\rangle_L = |10\rangle$. Then the logical σ_x is given by $L(\sigma_x) = \sigma_x \otimes \sigma_x$. This is obvious as

$$L(\sigma_x)(\alpha|0\rangle_L + \beta|1\rangle_L) = (\sigma_x \otimes \sigma_x)(\alpha|01\rangle + \beta|10\rangle) = \alpha|10\rangle + \beta|01\rangle = \alpha|1\rangle_L + \beta|0\rangle_L \tag{52}$$

2.1.8 Encoding and decoding matrices

For a given encoding E over Q physical qubits we define the following sets of encoding and decoding matrices: define $\mathcal{A}$ to be the set of matrices $A \in U(2^Q)$ such that $A|0...00\rangle = e^{i\theta}|0\rangle_L$ for some $\theta \in \mathbb{R}$ and define $\mathcal{B}$ to be the set of matrices $B \in U(2^Q)$ such that $B(\alpha|0\rangle_L + \beta|1\rangle_L) = \alpha|0\rangle_F + \beta|1\rangle_F$ for some pair of valid final states.

2.1.9 Robust encodings

Let $\Omega = \{M_1, M_2, ..., M_k\}$ be the set of base gates used for a particular quantum computing architecture. We say that a encoding over Ω is robust if for all $M \in \Omega$ and $|\phi\rangle \in (H^\dagger)^n$, there is some logical M' such that

$$M'|\phi\rangle \in (H^\dagger)^n \tag{53}$$

Usually (and for the rest of this document) we will take $\Omega = \{U, C_x\}$ where U is the most general single qubit unitary gate and C_x is the controlled-not gate. Thus, when we say an encoding is robust from here on, we mean that the encoding is robust over $\{U, C_x\}$.

Example: Consider the encoding $|0\rangle_L = |01\rangle$ and $|1\rangle_L = |10\rangle$. The first step to showing that this encoding is robust is to designate our logical gates for U and C_x. Let

$$L(U) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & 0 & 0 & -e^{i\phi}\sin\left(\frac{\theta}{2}\right) \\ 0 & \cos\left(\frac{\theta}{2}\right) & -e^{i\phi}\sin\left(\frac{\theta}{2}\right) & 0 \\ 0 & e^{i\lambda}\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right)e^{i\lambda+i\phi} & 0 \\ e^{i\lambda}\sin\left(\frac{\theta}{2}\right) & 0 & 0 & \cos\left(\frac{\theta}{2}\right)e^{i\lambda+i\phi} \end{pmatrix} \tag{54}$$

$$L(C_x) = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \tag{55}$$

On a single qubit, every $|\phi\rangle \in H^\dagger$ is of the form $|\phi\rangle = (\epsilon_0, 0, 0, \epsilon_3)^T$ which implies

$$L(U)|\phi\rangle = \begin{pmatrix} \epsilon_1 \cos\left(\frac{\theta}{2}\right) - \epsilon_2 e^{i\phi} \sin\left(\frac{\theta}{2}\right) \\ 0 \\ 0 \\ \epsilon_2 \cos\left(\frac{\theta}{2}\right) e^{i\lambda + i\phi} + e^{i\lambda} \epsilon_1 \sin\left(\frac{\theta}{2}\right) \end{pmatrix} \in H^\dagger \tag{56}$$

For two logical qubits, $|\phi\rangle \in (H^\dagger)^2$ has the form

$$|\phi\rangle = (\epsilon_0, \epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4, 0, 0, \epsilon_7, \epsilon_8, 0, 0, \epsilon_{11}, \epsilon_{12}, \epsilon_{13}, \epsilon_{14}, \epsilon_{15})^T \tag{57}$$

When operated on by the logical C_x, this turns into

$$L(C_x)|\phi\rangle = (\epsilon_1, \epsilon_0, \epsilon_3, \epsilon_2, \epsilon_4, 0, 0, \epsilon_7, \epsilon_{11}, 0, 0, \epsilon_8, \epsilon_{14}, \epsilon_{15}, \epsilon_{12}, \epsilon_{13})^T \in (H^\dagger)^2 \tag{58}$$

2.1.10 Stable endings

We say that an encoding has a stable end if there exist final states $|0\rangle_F$, $|1\rangle_F$, and some gate M such that the following conditions hold

1. For all $|\psi\rangle = \alpha|0\rangle_L + \beta|1\rangle_L \in H^+$, $|\langle 0|_F M|\psi\rangle|^2 = |\alpha|^2$ and $|\langle 1|_F M|\psi\rangle|^2 = |\beta|^2$
2. For all $|\phi\rangle \in H^-$, $|\langle 0|_F M|\phi\rangle|^2 + |\langle 1|_F M|\phi\rangle|^2 < 1$

Example: For a nontrivial example, consider

$$|0\rangle_L = \frac{|00\rangle + |11\rangle}{\sqrt{2}} \tag{59}$$

$$|1\rangle_L = \frac{|01\rangle + |10\rangle}{\sqrt{2}} \tag{60}$$

the matrix

$$M = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \end{pmatrix} \tag{61}$$

and final states $|0\rangle_F = |00\rangle$ and $|1\rangle_F = |01\rangle$. Then

$$M(\alpha|0\rangle_L + \beta|1\rangle_L) = \frac{1}{2}\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \end{pmatrix}\begin{pmatrix} \alpha \\ \beta \\ \beta \\ \alpha \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \\ 0 \\ 0 \end{pmatrix} = \alpha|0\rangle_F + \beta|1\rangle_F \quad (62)$$

Thus, the first condition is satisfied. To show the second condition is satisfied, let $|\phi\rangle$ be any state in the whole computational space such that

$$|\langle 0|_F M|\phi\rangle|^2 + |\langle 1|_F M|\phi\rangle|^2 = 1 \quad (63)$$

Since $|0\rangle_F = |00\rangle$ and $|1\rangle_F = |01\rangle$, this statement is equivalent to saying

$$M|\phi\rangle = \begin{pmatrix} \gamma \\ \delta \\ 0 \\ 0 \end{pmatrix} \quad (64)$$

for some $|\gamma|^2 + |\delta|^2 = 1$. But then

$$|\phi\rangle = M^{-1}M|\phi\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 1 & 0 & -1 & 0 \end{pmatrix}\begin{pmatrix} \gamma \\ \delta \\ 0 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} \gamma \\ \delta \\ \delta \\ \gamma \end{pmatrix} \in H^+ \quad (65)$$

We conclude the second condition is fulfilled.

2.2 Theorems

2.2.1 Theorem 1: When a state is valid

For a given encoding E and a state $|\psi\rangle$, $|\psi\rangle$ is in H^+ if and only if

$$|\psi\rangle = M(\alpha, \beta, 0, ..., 0)^T \tag{66}$$

(for some $\alpha, \beta \in \mathbb{C}$) for any $M \in \chi(E)$.

Proof: $\Leftarrow$ This is obvious as

$$|\psi\rangle = M(\alpha, \beta, 0, ..., 0)^T = \alpha \sum_{n=0}^{2^q-1} a_n|n\rangle + \beta \sum_{n=0}^{2^q-1} b_n|n\rangle = \alpha|0\rangle_L + \beta|1\rangle_L \in H^+ \tag{67}$$

$\Rightarrow$ In the other direction, $|\psi\rangle \in H^+$ implies

$$|\psi\rangle = \alpha|0\rangle_L + \beta|1\rangle_L = \alpha \sum_{n=0}^{2^q-1} a_n|n\rangle + \beta \sum_{n=0}^{2^q-1} b_n|n\rangle = M(\alpha, \beta, 0, ..., 0)^T \tag{68}$$

QED

2.2.2 Theorem 2: When a state is invalid

For a given encoding E and a state $|\phi\rangle$, $|\phi\rangle$ is in $H^\perp$ if and only if

$$|\phi\rangle = M(0, 0, x_1, x_2, ..., x_n)^T \tag{69}$$

(for some $x_i \in \mathbb{C}$) for any $M \in \chi(E)$.

Proof: $\Leftarrow$ Let $|\psi\rangle = \alpha|0\rangle_L + \beta|1\rangle_L$ be an arbitrary vector in H^+. Then

$$\langle\phi|\psi\rangle = \begin{pmatrix} 0 \\ 0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}^\dagger M^\dagger M \begin{pmatrix} \alpha \\ \beta \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}^\dagger \begin{pmatrix} \alpha \\ \beta \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = 0 \tag{70}$$

Thus, $|\phi\rangle \in H^\dagger$.

$\Rightarrow$ Suppose that $|\phi\rangle \in H^\dagger$ and consider the vector components of

$$M^\dagger|\phi\rangle = (y_1, y_2, ..., y_n)^T \tag{71}$$

where M is any element of $\chi(E)$. If either y_1 or y_2 is non-zero, let $|\psi\rangle = \alpha|0\rangle_L + \beta|1\rangle_L \in H^+$ where

$$\alpha = \frac{y_1}{\sqrt{|y_1|^2 + |y_2|^2}} \tag{72}$$

$$\beta = \frac{y_2}{\sqrt{|y_1|^2 + |y_2|^2}} \tag{73}$$

Note that these are valid choices for α and β since $|y_1|^2 + |y_2|^2 > 0$. But then

$$\langle\phi|\psi\rangle = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{pmatrix}^\dagger M^\dagger M \begin{pmatrix} \alpha \\ \beta \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{pmatrix}^\dagger \begin{pmatrix} \alpha \\ \beta \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \frac{|y_1|^2 + |y_2|^2}{\sqrt{|y_1|^2 + |y_2|^2}} > 0 \tag{74}$$

Since this contradicts the fact that $|\phi\rangle \in H^\dagger$, it must be the case that $y_1 = y_2 = 0$ and we are done.

QED

2.2.3 Theorem 3: Every encoding is a robust encoding

Every encoding is a robust encoding. Further, any logical gates $L(U)$ and $L(C_x)$ will satisfy the conditions that

$$|\phi\rangle \in H^\dagger \Rightarrow L(U)|\phi\rangle \in H^\dagger \tag{75}$$

$$|\phi\rangle \in (H^\dagger)^2 \Rightarrow L(C_x)|\phi\rangle \in (H^\dagger)^2 \tag{76}$$

Proof: By definition, we have that

$$L(U)(\alpha|0\rangle_L + \beta|1\rangle_L) = \tau|0\rangle_L + \delta|1\rangle_L \tag{77}$$

for some complex τ and δ. But then equation 77 becomes

$$L(U)M(\alpha, \beta, 0, ..., 0)^T = M(\tau, \delta, 0, ..., 0)^T \tag{78}$$

$$M^\dagger L(U)M(\alpha, \beta, 0, ..., 0)^T = (\tau, \delta, 0, ..., 0)^T \tag{79}$$

Since α and β are arbitrary, it must be the case that

$$M^\dagger L(U)M(0, 0, x_1, ..., x_n)^T = (0, 0, y_1, ..., y_n)^T \tag{80}$$

for all $x_i \in \mathbb{C}$. But this implies that for an arbitrary

$$|\phi\rangle = M(0,0,z_1,...,z_n)^T \in H^\dagger \tag{81}$$

we have

$$L(U)|\phi\rangle = L(U)M(0,0,z_1,...,z_n)^T = M(0,0,w_1,...,w_n)^T \in H^\dagger \tag{82}$$

for some $w_i \in \mathbb{C}$. The case for $L(C_x)$ can be proven a similar manner.

QED

2.2.4 Theorem 4: Every encoding has a stable ending

Every encoding has a stable ending.

Proof: Let E be an arbitrary encoding, M to be the conjugate transpose of any element in $\chi(E)$, $|0\rangle_F = |00...00\rangle$, and $|1\rangle_F = |00...01\rangle$. Then for $|\psi\rangle = \alpha|0\rangle_L + \beta|1\rangle_L \in H^+$ we have

$$|\langle 0|_F M|\psi\rangle|^2 = \left|(1,0,...,0)(\alpha,\beta,0,...,0)^T\right|^2 = |\alpha|^2 \tag{83}$$

$$|\langle 1|_F M|\psi\rangle|^2 = \left|(0,1,...,0)(\alpha,\beta,0,...,0)^T\right|^2 = |\beta|^2 \tag{84}$$

For any $|\phi\rangle \in H^-$, note that

$$|\phi\rangle = M(\alpha,\beta,x_1,...,x_n)^T \tag{85}$$

with the condition that for some $1 \leq i \leq n$, $|x_i| > 0$ (else $|\phi\rangle$ would be a member of H^+). This then implies that

$$|\alpha|^2 + |\beta|^2 < |\alpha|^2 + |\beta|^2 + |x_i|^2 \leq 1 \tag{86}$$

Therefore

$$|\langle 0|_F M|\phi\rangle|^2 + |\langle 1|_F M|\phi\rangle|^2 = |\alpha|^2 + |\beta|^2 < 1 \tag{87}$$

QED

2.2.5 Theorem 5: The existence of a flawless encoding

The encoding given by

$$|0\rangle_L = \frac{1}{\sqrt{3}}\Big(|001\rangle + |010\rangle + |100\rangle\Big)$$

$$|1\rangle_L = \frac{1}{\sqrt{3}}\Big(|011\rangle + |101\rangle + |110\rangle\Big) \tag{88}$$

is flawless.

Proof: Consider the state

$$|\psi\rangle = \alpha|0\rangle_L + \beta|1\rangle_L = \frac{1}{\sqrt{3}}\begin{pmatrix} 0 \\ \alpha \\ \alpha \\ \beta \\ \alpha \\ \beta \\ \beta \\ 0 \end{pmatrix} \tag{89}$$

Now, observe what happens in each of the following three situations:

$$U_0 = U \otimes I_4 \tag{90}$$

$$U_1 = I_2 \otimes U \otimes I_2 \tag{91}$$

$$U_2 = I_4 \otimes U \tag{92}$$

Here, U is the most general 2×2 unitary matrix (up to a phase)

$$U = \begin{pmatrix} \cos(\theta) & -e^{i\phi}\sin(\theta) \\ e^{i\lambda}\sin(\theta) & \cos(\theta)\, e^{i\lambda+i\phi} \end{pmatrix} \tag{93}$$

and U_i is U applied to qubit i. For $i = 0$, the outcome is

$$U_0|\psi\rangle = \begin{pmatrix} -\alpha e^{i\phi}\sin(\theta) \\ \alpha\cos(\theta) - \beta e^{i\phi}\sin(\theta) \\ \alpha\cos(\theta) - \beta e^{i\phi}\sin(\theta) \\ \beta\cos(\theta) \\ \alpha\cos(\theta)\, e^{i\lambda+i\phi} \\ \alpha e^{i\lambda}\sin(\theta) + \beta\cos(\theta)\, e^{i\lambda+i\phi} \\ \alpha e^{i\lambda}\sin(\theta) + \beta\cos(\theta)\, e^{i\lambda+i\phi} \\ \beta e^{i\lambda}\sin(\theta) \end{pmatrix} \in H^+ \tag{94}$$

Since the first entry and the last entry must both be zero (and at least one of α and β is nonzero) we may conclude that $\sin(\theta) = 0$ and therefore $\cos(\theta) = \pm 1$. The vector then simplifies to

$$U_0|\psi\rangle = \pm \begin{pmatrix} 0 \\ \alpha \\ \alpha \\ \beta \\ \alpha e^{i\lambda+i\phi} \\ \beta e^{i\lambda+i\phi} \\ \beta e^{i\lambda+i\phi} \\ 0 \end{pmatrix} \in H^+ \tag{95}$$

Again, since at least one of α and β is non-zero, we may conclude that $e^{i\lambda+i\phi} = 1$. But what is this unitary matrix? It is easy to calculate that

$$U = \begin{pmatrix} \cos(\theta) & -e^{i\phi}\sin(\theta) \\ e^{i\lambda}\sin(\theta) & \cos(\theta)\, e^{i\lambda+i\phi} \end{pmatrix} = \pm \begin{pmatrix} 1 & 0 \\ 0 & e^{i\lambda+i\phi} \end{pmatrix} = \pm \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \pm I_2 \tag{96}$$

In the same manner, we can prove that $U_1|\psi\rangle \in H^+$ and $U_2|\psi\rangle \in H^+$ implies U is $\pm I_2$. Thus, we have shown that the encoding given above is a flawless encoding.

QED

2.2.5.1 Lemma 1

If X is any matrix in $\mathbb{C}^{2\times 2}$, then there exists a unitary matrix U, with the stipulation that $U \neq e^{i\phi} I_2$ for any angle ϕ, such that

$$\det([U, X]) = |UX - XU| = 0$$

Proof: By definition there exists a magnitude 1 vector $\vec{v}_1$ such that $X\vec{v}_1 = \tau\vec{v}_1$ for

some $\tau \in \mathbb{C}$. Let $\vec{v}_2$ be any vector orthonormal to $\vec{v}_1$. Now, let U be the unitary matrix defined by

$$U\vec{v}_1 = \vec{v}_1 \text{ and } U\vec{v}_2 = -\vec{v}_2 \tag{97}$$

Note that we are assured that U is not the identity matrix times a phase since its eigenvalues have different phases. But then

$$(UX - XU)\vec{v}_1 = UX\vec{v}_1 - XU\vec{v}_1 = \tau U\vec{v}_1 - X\vec{v}_1 = \tau\vec{v}_1 - \tau\vec{v}_1 = \vec{0} \tag{98}$$

We conclude that there exists a non-trivial U such that $\det([U, X]) = 0$.

QED

2.2.6 Theorem 6: The existence of a matrix with determinant zero

Let M be an arbitrary 4×4 unitary matrix and consider the following two matrices

$$M^\dagger(U \otimes I_2)M = \begin{pmatrix} Q_0 & R_0 \\ S_0 & T_0 \end{pmatrix} \tag{99}$$

$$M^\dagger(I_2 \otimes U)M = \begin{pmatrix} Q_1 & R_1 \\ S_1 & T_1 \end{pmatrix} \tag{100}$$

Here, U is a 2×2 unitary matrix and $Q_i, R_i, S_i, T_i \in \mathbb{C}^{2\times 2}$. There exists a non-trivial 2×2 unitary matrix U (non-trivial in the sense that it is not the identity times some phase) such that either $\det(S_0) = 0$ or $\det(S_1) = 0$.

Proof: Let M be an arbitrary 4×4 unitary matrix

$$M = \begin{pmatrix} a_0 & b_0 & c_0 & d_0 \\ a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ a_3 & b_3 & c_3 & d_3 \end{pmatrix} \tag{101}$$

There are 7 possible cases for M that we will investigate:

1. $\begin{vmatrix} a_0 & b_0 \\ a_1 & b_1 \end{vmatrix} \neq 0$

2. $\begin{vmatrix} a_2 & b_2 \\ a_3 & b_3 \end{vmatrix} \neq 0$

3. $\begin{vmatrix} a_1 & b_1 \\ a_3 & b_3 \end{vmatrix} \neq 0$

4. $\begin{vmatrix} a_0 & b_0 \\ a_2 & b_2 \end{vmatrix} \neq 0$

5. $M = \begin{pmatrix} e^{i\delta_1}\cos(\theta_1) & -e^{i(\delta_1+\phi_1)}\sin(\theta_1) & 0 & 0 \\ 0 & 0 & e^{i\delta_2}\cos(\theta_2) & -e^{i(\delta_2+\phi_2)}\sin(\theta_2) \\ 0 & 0 & e^{i(\delta_2+\lambda_2)}\sin(\theta_2) & e^{i(\delta_2+\lambda_2+\phi_2)}\cos(\theta_2) \\ e^{i(\delta_1+\lambda_1)}\sin(\theta_1) & e^{i(\delta_1+\lambda_1+\phi_1)}\cos(\theta_1) & 0 & 0 \end{pmatrix}$
for some $(\delta_1, \theta_1, \lambda_1, \phi_1, \delta_2, \theta_2, \lambda_2, \phi_2) \in \mathbb{R}^8$

6. $M = \begin{pmatrix} 0 & 0 & e^{i\delta_2}\cos(\theta_2) & -e^{i(\delta_2+\phi_2)}\sin(\theta_2) \\ e^{i\delta_1}\cos(\theta_1) & -e^{i(\delta_1+\phi_1)}\sin(\theta_1) & 0 & 0 \\ e^{i(\delta_1+\lambda_1)}\sin(\theta_1) & e^{i(\delta_1+\lambda_1+\phi_1)}\cos(\theta_1) & 0 & 0 \\ 0 & 0 & e^{i(\delta_2+\lambda_2)}\sin(\theta_2) & e^{i(\delta_2+\lambda_2+\phi_2)}\cos(\theta_2) \end{pmatrix}$
for some $(\delta_1, \theta_1, \lambda_1, \phi_1, \delta_2, \theta_2, \lambda_2, \phi_2) \in \mathbb{R}^8$

7. M does not fall into cases $1-6$

We will start with case 7) as this case is actually impossible:

Case 7: In order to show that this case is impossible, we will assume that we are not in cases $1-4$ and from there show how this implies either of case 5, case 6, or a contradiction. To start, note that every 2×2 matrix with determinant 0 has the form

$$\begin{pmatrix} a & b \\ xa & xb \end{pmatrix} \text{ or } \begin{pmatrix} 0 & 0 \\ a & b \end{pmatrix} \tag{102}$$

for some $a, b, x \in \mathbb{C}$. Then with matrices from cases $1-4$, we are left with 16 subcases, each one corresponding to a different form from equation 102. For example, the first subcase is

Case 7.1: Assume that the matrices are of the form

$$\begin{pmatrix} a_0 & b_0 \\ a_1 & b_1 \end{pmatrix} = \begin{pmatrix} a & b \\ xa & xb \end{pmatrix} \tag{103}$$

$$\begin{pmatrix} a_2 & b_2 \\ a_3 & b_3 \end{pmatrix} = \begin{pmatrix} c & d \\ yc & yd \end{pmatrix} \tag{104}$$

$$\begin{pmatrix} a_1 & b_1 \\ a_3 & b_3 \end{pmatrix} = \begin{pmatrix} e & f \\ ze & zf \end{pmatrix} \tag{105}$$

$$\begin{pmatrix} a_0 & b_0 \\ a_2 & b_2 \end{pmatrix} = \begin{pmatrix} g & h \\ wg & wh \end{pmatrix} \tag{106}$$

for some $a, b, c, d, e, f, g, h, x, y, z, w \in \mathbb{C}$. Right off the bat, we get the following equalities: $a = g$, $b = h$, $c = wg$, $d = wh$, $e = xa$, and $f = xb$. With these equations, we can rewrite the first two columns of M as

$$\begin{pmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{pmatrix} = \begin{pmatrix} a & b \\ xa & xb \\ wa & wb \\ zxa & zxb \end{pmatrix} \tag{107}$$

But this is a contradiction as these columns are not linearly independent. Thus, this subcase is impossible.

Case 7.2: We will show the work in this case as it diverges enough from the work in case 7.1 to be worth writing out in full. Assume that the matrices are of the form

$$\begin{pmatrix} a_0 & b_0 \\ a_1 & b_1 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ a & b \end{pmatrix} \tag{108}$$

$$\begin{pmatrix} a_2 & b_2 \\ a_3 & b_3 \end{pmatrix} = \begin{pmatrix} c & d \\ xc & xd \end{pmatrix} \tag{109}$$

$$\begin{pmatrix} a_1 & b_1 \\ a_3 & b_3 \end{pmatrix} = \begin{pmatrix} e & f \\ ye & yf \end{pmatrix} \tag{110}$$

$$\begin{pmatrix} a_0 & b_0 \\ a_2 & b_2 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ g & h \end{pmatrix} \tag{111}$$

for some $a, b, c, d, e, f, g, h, x, y \in \mathbb{C}$. Again, we get a set of easy equations:
$c = g$, $d = h$, $a = e$, $b = f$, $xc = ye$, and $xd = yf$. But then we can rewrite the first

two columns of M as

$$\begin{pmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ e & f \\ c & d \\ xc & xd \end{pmatrix} \tag{112}$$

Now, if $y \neq 0$ then we can further manipulate this as

$$= \begin{pmatrix} 0 & 0 \\ \frac{x}{y}c & \frac{x}{y}d \\ c & d \\ xc & xd \end{pmatrix} \tag{113}$$

But again, these columns are not linearly independent. Thus, y must be 0 (implying that $xc = xd = 0$) and therefore the first two columns of M are of the form

$$\begin{pmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ a & b \\ c & d \\ 0 & 0 \end{pmatrix} \tag{114}$$

Since these columns are orthonormal, the four non-zero elements must form a 2×2 unitary matrix, and we are therefore firmly planted in case 6.

Cases 7.3-7.16: The remaining cases are all dealt with in a similar manner as the previous two cases.

Case 1: Write equation 99 in block matrix form:

$$\begin{pmatrix} Q_1 & R_1 \\ S_1 & T_1 \end{pmatrix} = M^\dagger (I_2 \otimes U) M = \begin{pmatrix} A^\dagger & C^\dagger \\ B^\dagger & D^\dagger \end{pmatrix} \begin{pmatrix} U & \hat{0} \\ \hat{0} & U \end{pmatrix} \begin{pmatrix} A & B \\ C & D \end{pmatrix} \tag{115}$$

Note that by our assumption we know that A is invertible and thus by the Nullity Theorem [49] we also know that D is invertible. But then we can rewrite

$$M^\dagger = M^{-1} = \begin{pmatrix} (A - BD^{-1}C)^{-1} & \hat{0} \\ \hat{0} & (D - CA^{-1}B)^{-1} \end{pmatrix} \begin{pmatrix} I_2 & -BD^{-1} \\ -CA^{-1} & I_2 \end{pmatrix} \tag{116}$$

which comes from the block matrix inversion formula [50]. Using this form of $M^\dagger$ in equation 115 give us

$$\begin{pmatrix} Q_1 & R_1 \\ S_1 & T_1 \end{pmatrix} = \begin{pmatrix} (A - BD^{-1}C)^{-1} & \hat{0} \\ \hat{0} & (D - CA^{-1}B)^{-1} \end{pmatrix} \begin{pmatrix} I & -BD^{-1} \\ -CA^{-1} & I \end{pmatrix} \begin{pmatrix} U & \hat{0} \\ \hat{0} & U \end{pmatrix} \begin{pmatrix} A & B \\ C & D \end{pmatrix} \tag{117}$$

Multiplying this out, we get that

$$S_1 = (D - CA^{-1}B)^{-1}(UCA^{-1} - CA^{-1}U)A \tag{118}$$

We may now appeal to the lemma proved earlier with $X = CA^{-1}$. Thus, we conclude there is a non-trivial U such that

$$\det(S_1) = \det(D - CA^{-1}B)^{-1} \det(UCA^{-1} - CA^{-1}U) \det(A) = 0 \tag{119}$$

Case 2: For this case, define

$$P = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \tag{120}$$

Then we write

$$\begin{pmatrix} Q_1 & R_1 \\ S_1 & T_1 \end{pmatrix} = M^\dagger (I_2 \otimes U) M = \begin{pmatrix} A^\dagger & C^\dagger \\ B^\dagger & D^\dagger \end{pmatrix} \begin{pmatrix} U & \hat{0} \\ \hat{0} & U \end{pmatrix} \begin{pmatrix} A & B \\ C & D \end{pmatrix}$$

$$= \begin{pmatrix} A^\dagger & C^\dagger \\ B^\dagger & D^\dagger \end{pmatrix} P^\dagger P \begin{pmatrix} U & \hat{0} \\ \hat{0} & U \end{pmatrix} P^\dagger P \begin{pmatrix} A & B \\ C & D \end{pmatrix} \tag{121}$$

But this simplifies as

$$PM = \begin{pmatrix} C & D \\ A & B \end{pmatrix} \tag{122}$$

$$P \begin{pmatrix} U & \hat{0} \\ \hat{0} & U \end{pmatrix} P^\dagger = \begin{pmatrix} U & \hat{0} \\ \hat{0} & U \end{pmatrix} \tag{123}$$

Then equation 121 simplifies to

$$= \begin{pmatrix} C^\dagger & A^\dagger \\ D^\dagger & B^\dagger \end{pmatrix} \begin{pmatrix} U & \hat{0} \\ \hat{0} & U \end{pmatrix} \begin{pmatrix} C & D \\ A & B \end{pmatrix} \tag{124}$$

This is the same as case 1 (as we assumed that C was invertible) except we now have that

$$S_1 = (B - AC^{-1}D)^{-1}(UAC^{-1} - AC^{-1}U)C \tag{125}$$

Again appealing to the lemma, we conclude that there is a non-trivial U such that $\det(S_1) = 0$.

Case 3: This case can be worked through in the same manner as the last two cases except we start with equation 100 instead of equation 99. First, define

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{126}$$

Then

$$\begin{pmatrix} Q_0 & R_0 \\ S_0 & T_0 \end{pmatrix} = M^\dagger (U \otimes I_2) M = M^\dagger P^\dagger P (U \otimes I_2) P^\dagger P M \tag{127}$$

If we write

$$U = \begin{pmatrix} e^{i\delta}\cos(\theta) & -e^{i(\delta+\phi)}\sin(\theta) \\ e^{i(\delta+\lambda)}\sin(\theta) & e^{i(\delta+\lambda+\phi)}\cos(\theta) \end{pmatrix} \tag{128}$$

(the most general 2×2 unitary matrix), we see that

$$P^{\dagger}(U \otimes I_2)P = \begin{pmatrix} e^{i(\delta+\lambda+\phi)}\cos(\theta) & e^{i(\delta+\lambda)}\sin(\theta) & 0 & 0 \\ -e^{i(\delta+\phi)}\sin(\theta) & e^{i\delta}\cos(\theta) & 0 & 0 \\ 0 & 0 & e^{i(\delta+\lambda+\phi)}\cos(\theta) & e^{i(\delta+\lambda)}\sin(\theta) \\ 0 & 0 & -e^{i(\delta+\phi)}\sin(\theta) & e^{i\delta}\cos(\theta) \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} e^{i(\delta+\lambda+\phi)}\cos(\theta) & e^{i(\delta+\lambda)}\sin(\theta) \\ -e^{i(\delta+\phi)}\sin(\theta) & e^{i\delta}\cos(\theta) \end{pmatrix} = I_2 \otimes U^{'} \tag{129}$$

where $U^{'}$ is simply another way to write the most general 2×2 unitary matrix. Importantly, note that if $U = e^{i\phi}I_2$ then $U^{'} = e^{i\phi}I_2$ (and vice-versa). We can also write

$$PM = \begin{pmatrix} A^{'} & B^{'} \\ C^{'} & D^{'} \end{pmatrix} \tag{130}$$

where $A^{'} = \begin{pmatrix} a_1 & b_1 \\ a_3 & b_3 \end{pmatrix}$ (note that by our assumption $A^{'}$ is invertible). But this leads us to the same situation we had in cases 1 and 2. Here, we get that

$$S_0 = (D^{'} - C^{'}A^{'-1}B^{'})^{-1}(U^{'}C^{'}A^{'-1} - C^{'}A^{'-1}U^{'})A^{'} \tag{131}$$

Thus, there is a nontrivial matrix U (related to $U^{'}$ in a complicated manner) such that $\det(S_0) = 0$.

Case 4: This final case is proved in much the same way as the previous cases. We simply take

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{132}$$

and the logical train of thought will lead us to

$$S_1 = (B' - A'C'^{-1}D')^{-1}(U'A'C'^{-1} - A'C'^{-1}U')C' \tag{133}$$

(where C' is matrix we assumed was invertible). Thus, there exists non-trivial U such that $\det(S_0) = 0$.

Case 5 and Case 6: These cases will be proved in a different manner than the previous cases. Simply note that

$$\begin{pmatrix} Q_0 & R_0 \\ S_0 & T_0 \end{pmatrix} = M^\dagger \left(I_2 \otimes \begin{pmatrix} 1 & 0 \\ 0 & e^{i\tau} \end{pmatrix} \right) M = \begin{pmatrix} A & \hat{0} \\ \hat{0} & B \end{pmatrix} \tag{134}$$

$$\begin{pmatrix} Q_1 & R_1 \\ S_1 & T_1 \end{pmatrix} = M^\dagger \left(\begin{pmatrix} 1 & 0 \\ 0 & e^{i\tau} \end{pmatrix} \otimes I_2 \right) M = \begin{pmatrix} C & \hat{0} \\ \hat{0} & D \end{pmatrix} \tag{135}$$

for some 2×2 unitary A, B, C, D. In both cases, it is obvious that $S_0 = S_1 = \hat{0}$ for all angles τ. Since these clearly have determinant 0, we are done.

QED

2.2.7 Corollary 1: There is no flawless encoding over two qubits

There are no flawless encodings over two physical qubits.

Proof: Let E be an arbitrary encoding over two physical qubits and let $M \in \chi(E)$. From Theorem 6 we know there exists a 2×2 unitary matrix U such that either S_0 or S_1 in

$$M^\dagger(U \otimes I_2)M = \begin{pmatrix} Q_0 & R_0 \\ S_0 & T_0 \end{pmatrix}$$

$$M^\dagger(I_2 \otimes U)M = \begin{pmatrix} Q_1 & R_1 \\ S_1 & T_1 \end{pmatrix} \tag{136}$$

has determinant zero and U is not the identity matrix times some phase. Suppose that $\det(S_0) = 0$ (the work is identical if $\det(S_1) = 0$) and let $\alpha, \beta \in \mathbb{C}$ be complex numbers such that $(\alpha, \beta)^T$ has unit magnitude and is in the nullspace of S_0. Then for $|\psi\rangle = M(\alpha, \beta, 0, 0)^T \in H^+$ we have

$$(U \otimes I_2)|\psi\rangle = MM^\dagger(U \otimes I_2)M \begin{pmatrix} \alpha \\ \beta \\ 0 \\ 0 \end{pmatrix}$$

$$= M \begin{pmatrix} Q_0 & R_0 \\ S_0 & T_0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ 0 \\ 0 \end{pmatrix} = M \begin{pmatrix} \tau \\ \delta \\ 0 \\ 0 \end{pmatrix} \in H^+ \tag{137}$$

where the last equality happens from our definition of α and β. Thus, E is not a flawless encoding and we are done.

QED

2.2.8 Theorem 7: Near-flawless encodings

A 2-qubit encoding E is near flawless if and only if the matrices S_0 and S_1 (as defined in Theorem 6 in equations 99 and 100 for any $M \in \chi(E)$) are never the $\hat{0}$ matrix except for the trivial error $e^{i\phi} I_2$.

Proof: ($\Rightarrow$) We will prove the contrapositive. Suppose there is a non-trivial error U such that S_0 or S_1 is the $\hat{0}$ matrix. If it is the former, then

$$M^\dagger (U \otimes I_2) M \begin{pmatrix} \alpha \\ \beta \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} R_0 & Q_0 \\ S_0 & T_0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} R_0 & Q_0 \\ \hat{0} & T_0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \tau \\ \delta \\ 0 \\ 0 \end{pmatrix} \tag{138}$$

That is, for any $\alpha|0\rangle_L + \beta|1\rangle_L \in H^+$ we have

$$(U \otimes I_2)(\alpha|0\rangle_L + \beta|1\rangle_L) = \delta|0\rangle_L + \tau|1\rangle_L \in H^+ \tag{139}$$

We conclude that the encoding is not near flawless.

($\Leftarrow$) Suppose that S_0 and S_1 are never the $\hat{0}$ (except for the case $U = e^{i\phi} I_2$). Now, if U is an error matrix on the 0th qubit we know

$$(U \otimes I_2)|\psi\rangle \in H^+ \tag{140}$$

If such a U does not exist then it must exist for qubit 1 since no 2-qubit encoding is flawless (and the arguement is the same). This then implies

$$\begin{pmatrix} \tau \\ \delta \\ 0 \\ 0 \end{pmatrix} = M^{\dagger}(U \otimes I_2)M \begin{pmatrix} \alpha \\ \beta \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} R_0 & Q_0 \\ S_0 & T_0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ 0 \\ 0 \end{pmatrix} \tag{141}$$

It is clear that $\det(S_0) = 0$ since

$$S_0 \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{142}$$

As stated before, this implies S_0 is of the form

$$\begin{pmatrix} a & b \\ xa & xb \end{pmatrix} \text{ or } \begin{pmatrix} 0 & 0 \\ a & b \end{pmatrix} \tag{143}$$

We will now prove that $\mu(\Omega) = 0$ where

$$\Omega = \left\{ \begin{pmatrix} y \\ z \end{pmatrix} : S_0 \begin{pmatrix} y \\ z \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \text{ and } |y|^2 + |z|^2 = 1 \right\} \tag{144}$$

Case 1.1: Assume $a = 0$, $b \neq 0$, and S_0 is the first type of non-invertible matrix. Then

$$\begin{pmatrix} 0 & b \\ 0 & xb \end{pmatrix} \begin{pmatrix} y \\ z \end{pmatrix} = \begin{pmatrix} zb \\ zxb \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{145}$$

Since $b \neq 0$ we conclude $z = 0$ and thus

$$\Omega = \left\{ \begin{pmatrix} y \\ 0 \end{pmatrix} : |y| = 1 \right\} \tag{146}$$

Thus, $\mu(\Omega) = 0$ and we are done.

Case 1.2 Assume $a = 0$, $b \neq 0$, and S_0 is the second type of non-invertible matrix. This case follows in the same way as the previous case.

Case 2: Assume $b = 0$ and $a \neq 0$. Then this case follows in the same was as Case 1.

Case 3.1: Assume $ab \neq 0$ and S_0 is the first type of non-invertible matrix. Then

$$\begin{pmatrix} a & b \\ xb & xb \end{pmatrix} \begin{pmatrix} y \\ z \end{pmatrix} = \begin{pmatrix} ya + zb \\ x(ya + zb) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{147}$$

We then have

$$ya + zb = 0 \Rightarrow y = -\frac{zb}{a} \tag{148}$$

Thus, the set Ω is

$$\Omega = \left\{ \begin{pmatrix} -z\frac{a}{b} \\ z \end{pmatrix} : |z| = \frac{1}{1 + \left|\frac{a}{b}\right|^2} \right\} \tag{149}$$

Then $\mu(\Omega) = 0$ and we are done.

Case 3.2: Assume $ab \neq 0$ and S_0 is the second type of non-invertible matrix. By the same logic as above, we have $\mu(\Omega) = 0$.

Case 4: The last case is impossible as $a = b = 0$ implies $S_0 = \hat{0}$.

Having covered all our cases, we conclude $\mu(\Omega) = 0$.

2.2.9 Theorem 8: The existence of a near-flawless encoding

The encoding given by

$$|0\rangle_L = \frac{1}{2\sqrt{3}}\Big(3|00\rangle + |01\rangle + |10\rangle + |11\rangle\Big)$$

$$|1\rangle_L = \frac{1}{2}\Big(-|00\rangle + |01\rangle + |10\rangle + |11\rangle\Big) \tag{150}$$

is near flawless.

Proof: Let $M \in \chi(E)$ be given by

$$M = \begin{pmatrix} \frac{\sqrt{3}}{2} & -\frac{1}{2} & 0 & 0 \\ \frac{1}{2\sqrt{3}} & \frac{1}{2} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{2}} \\ \frac{1}{2\sqrt{3}} & \frac{1}{2} & \frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{2\sqrt{3}} & \frac{1}{2} & -\sqrt{\frac{2}{3}} & 0 \end{pmatrix} \tag{151}$$

We can then calculate S_0 and S_1 explicitly with

$$U = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -e^{i\phi}\sin\left(\frac{\theta}{2}\right) \\ e^{i\lambda}\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right)e^{i\lambda+i\phi} \end{pmatrix} \tag{152}$$

to get

$$S_0 = \begin{pmatrix} \frac{\sin\left(\frac{\theta}{2}\right)\left(e^{i\lambda}-e^{i\phi}\right)-\cos\left(\frac{\theta}{2}\right)\left(-1+e^{i(\lambda+\phi)}\right)}{6\sqrt{2}} & \frac{\cos\left(\frac{\theta}{2}\right)\left(-\left(-1+e^{i(\lambda+\phi)}\right)\right)-\sin\left(\frac{\theta}{2}\right)\left(3e^{i\lambda}+e^{i\phi}\right)}{2\sqrt{6}} \\ \frac{\cos\left(\frac{\theta}{2}\right)\left(-\left(-1+e^{i(\lambda+\phi)}\right)\right)-\sin\left(\frac{\theta}{2}\right)\left(3e^{i\lambda}+e^{i\phi}\right)}{2\sqrt{6}} & \frac{\sin\left(\frac{\theta}{2}\right)\left(e^{i\lambda}-e^{i\phi}\right)-\cos\left(\frac{\theta}{2}\right)\left(-1+e^{i(\lambda+\phi)}\right)}{2\sqrt{2}} \end{pmatrix} \tag{153}$$

$$S_1 = \begin{pmatrix} \frac{\sin\left(\frac{\theta}{2}\right)\left(e^{i\lambda}-e^{i\phi}\right)-\cos\left(\frac{\theta}{2}\right)\left(-1+e^{i(\lambda+\phi)}\right)}{6\sqrt{2}} & \frac{\cos\left(\frac{\theta}{2}\right)\left(-\left(-1+e^{i(\lambda+\phi)}\right)\right)-\sin\left(\frac{\theta}{2}\right)\left(3e^{i\lambda}+e^{i\phi}\right)}{2\sqrt{6}} \\ \frac{\sin\left(\frac{\theta}{2}\right)\left(3e^{i\lambda}+e^{i\phi}\right)+\cos\left(\frac{\theta}{2}\right)\left(-1+e^{i(\lambda+\phi)}\right)}{2\sqrt{6}} & \frac{\sin\left(\frac{\theta}{2}\right)\left(e^{i\phi}-e^{i\lambda}\right)+\cos\left(\frac{\theta}{2}\right)\left(-1+e^{i(\lambda+\phi)}\right)}{2\sqrt{2}} \end{pmatrix} \tag{154}$$

Now, define the function

$$f\left[\begin{pmatrix} a & b \\ c & d \end{pmatrix}\right] = |a|^2 + |b|^2 + |c|^2 + |d|^2 \tag{155}$$

Note that this function is zero if and only if the input matrix is $\hat{0}$. We can then write out

$$f(S_0)f(S_1) = \frac{1}{81}(3\cos(\theta)\cos(\lambda)\cos(\phi) - (\cos(\theta)+3)\sin(\lambda)\sin(\phi)$$

$$-\sin(\theta)\cos(\lambda) + \sin(\theta)\cos(\phi) + 3\cos(\theta) + \cos(\lambda)\cos(\phi) - 7)^2 \tag{156}$$

Thus, the matrices S_0 and S_1 can only ever be $\hat{0}$ if the function

$$g(\theta,\lambda,\phi) = 3\cos(\theta)\cos(\lambda)\cos(\phi) - (\cos(\theta)+3)\sin(\lambda)\sin(\phi)$$

$$-\sin(\theta)\cos(\lambda) + \sin(\theta)\cos(\phi) + 3\cos(\theta) + \cos(\lambda)\cos(\phi) - 7 \tag{157}$$

is equal to 0. We will show that this function is only zero on the set

$$(\theta,\lambda,\phi) \in \{(2\pi k, \lambda, -\lambda + 2\pi m) : \lambda \in \mathbb{R} \text{ and } k,m \in \mathbb{Z}\} \tag{158}$$

Note that for any element of this set we have $U = e^{i\tau}I_2$ for some $\tau \in \mathbb{R}$ which is why these solutions can be safely ignored. To prove that this is the only solution set, we will use the half-tangent angle substitution. That is

$$\tan\left(\frac{\theta}{2}\right) = x \Rightarrow \cos(\theta) = \frac{1-x^2}{1+x^2} \text{ and } \sin(\theta) = \frac{2x}{1+x^2} \tag{159}$$

where $x \in \mathbb{R}$ (we can do the same for λ and ϕ with y and z). Then transforming $0 = g(\theta, \lambda, \phi)$ into a function of x, y, z gives us

$$0 = -\frac{4\left(x^2\left(y^2\left(3z^2+2\right)+2yz+2z^2+3\right)+x\left(z^2-y^2\right)+2(y+z)^2\right)}{\left(x^2+1\right)\left(y^2+1\right)\left(z^2+1\right)} \tag{160}$$

This simplifies to

$$0 = x^2\left(y^2\left(3z^2+2\right)+2yz+2z^2+3\right)+x\left(z^2-y^2\right)+2(y+z)^2 \tag{161}$$

Solving this quadratic in x gives us

$$x = \frac{\pm\sqrt{-3(y+z)^2(8+6yz+5z^2+y^2(5+8z^2))}+y^2-z^2}{2\left(3y^2z^2+2y^2+2yz+2z^2+3\right)} \tag{162}$$

It is easy to show that both

$$8+6yz+5z^2+y^2(5+8z^2) > 0 \tag{163}$$

$$3y^2z^2+2y^2+2yz+2z^2+3 > 0 \tag{164}$$

for all $(y, z) \in \mathbb{R}^2$. Thus, x can only be a real number if $y+z = 0$ (else the discriminant would be negative). This implies that

$$\tan\left(\frac{\lambda}{2}\right) = -\tan\left(\frac{\phi}{2}\right) \Rightarrow \lambda = -\phi + 2\pi m \tag{165}$$

for some $m \in \mathbb{Z}$. Additionally, plugging $y = -z$ back into x gives us

$$x = 0 \Rightarrow \theta = 2\pi k \tag{166}$$

for some $k \in \mathbb{Z}$. Thus, the only matrices U which can set either of S_0 or S_1 to be $\hat{0}$ are the matrices $U = e^{i\tau} I_2$ for some $\tau \in \mathbb{R}$. By Theorem 7 we conclude the encoding is near flawless.

QED

2.2.10 Theorem 9: Encoding a near-flawless code takes at least two controlled not gates

If E is a near-flawless encoding and $M \in \chi(E)$, then M cannot be created using zero or one physical C_x gate.

Proof: The following proof will be slip into two cases: the case where E is an encoding over two physical qubits and the case where E is an encoding over more than two physical qubits. As the possibility of M being the product of single qubit gates is obviously impossible, we shall focus on the situation where M is constructed with a single C_x gate.

Case 1: Suppose that E is an encoding over two physical qubits but there exists $M \in \chi(E)$ which takes a single physcial C_x to create. Without loss of generality, suppose that this M has the form

$$M = (U_3 \otimes U_4) C_x (U_1 \otimes U_2) \tag{167}$$

which in circuit form is

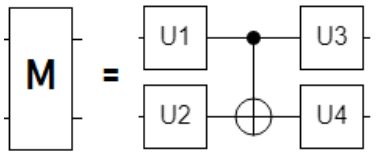

Figure 16: The most general way to describe a 2-qubit gate created with one physical C_x. The gates U_1-U_4 are general single qubit gates.

Here, the gates U_1-U_4 are general single qubit gates. Now, consider an error on the second qubit (the one being controlled by the C_x gate) of the form

$$Error = U_4 \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -i \\ i & -1 \end{pmatrix} U_4^\dagger \tag{168}$$

Plugging these matrices into M and U in equation 100 gives us

$$M^\dagger (I_2 \otimes Error) M = \begin{pmatrix} A & \hat{0} \\ \hat{0} & B \end{pmatrix} \tag{169}$$

for some unitary matrices A and B. Thus, $S_1 = \hat{0}$ and therefore by Theorem 7 E is not a near-flawless encoding.

Case 2: Suppose that E is an encoding over $r \geq 3$ physical qubits. Since there exists $M \in \chi(E)$ that can be constructed with a single C_x, without loss of generality suppose this gate is between the first two qubits. Then M is of the form

$$M = D \otimes U \tag{170}$$

where D is a $2^{r-1} \times 2^{r-1}$ unitary matrix and U is some single qubit gate. But then for any $|\psi\rangle = \alpha|0\rangle_L + \beta|1\rangle_L \in H^+$ and error

$$Error = U\sigma_x U^\dagger \tag{171}$$

on the final qubit we have

$$M^\dagger(I_2^{r-1} \otimes Error)M(\alpha, \beta, 0, ..., 0)^T = (D^\dagger \otimes U^\dagger)(I_2^{r-1} \otimes U\sigma_x U^\dagger)(D \otimes U)(\alpha, \beta, 0, ..., 0)^T$$

$$(D^\dagger I_{2^{r-1}} D)\otimes(U^\dagger U\sigma_x U^\dagger U)(\alpha, \beta, 0, ..., 0)^T = (I_{2^{r-1}}\otimes\sigma_x)(\alpha, \beta, 0, ..., 0)^T = (\beta, \alpha, 0, ..., 0)^T \in H^+ \tag{172}$$

Since $|\psi\rangle$ was arbitrary, we conclude E is not a near-flawless encoding.

QED

2.2.11 Corollary 2: Two physical controlled not gates is sufficient for a near-flawless encoding

Two physical C_x gates is sufficient for a near-flawless encoding.

Proof: Consider the encoding E given in Theorem 8 defined by

$$|0\rangle_L = \frac{1}{2\sqrt{3}}\Big(3|00\rangle + |01\rangle + |10\rangle + |11\rangle\Big)$$

$$|1\rangle_L = \frac{1}{2}\Big(- |00\rangle + |01\rangle + |10\rangle + |11\rangle\Big) \tag{173}$$

with $M \in \chi(E)$ be given by

$$M = \begin{pmatrix} \frac{\sqrt{3}}{2} & -\frac{1}{2} & 0 & 0 \\ \frac{1}{2\sqrt{3}} & \frac{1}{2} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{2}} \\ \frac{1}{2\sqrt{3}} & \frac{1}{2} & \frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{2\sqrt{3}} & \frac{1}{2} & -\sqrt{\frac{2}{3}} & 0 \end{pmatrix} \tag{174}$$

By the work of Vatan and Williams [51], we know that M can be implemented in two physical C_x gates. Since E is a near-flawless encoding, we are done.

QED

III. The Bit-Flip Encoding

Preamble

In this chapter, we will describe a procedure to implement a simple NAED code. This includes the operators required to encode the initial states as well as methods to create logical gates out of the standard base set of U_3 and C_x gates. These logical gates are then used to produce logical Greenberger–Horne–Zeilinger (GHZ) states with differing amounts logical qubits and physical qubits per logical qubit. In general, constructing these logical gates remains an active area of research for all error mitigation schemes [52, 53, 54, 55].

The unencoded GHZ circuit can be thought of as generalized bell circuit and produces the state

$$GHZ(N,1) = \frac{|00...0\rangle + |11...1\rangle}{\sqrt{2}} \tag{175}$$

Here, there are N 0s and N 1s in each ket and the 1 in $GHZ(N,1)$ represents the fact that there is no encoding with one physical qubit per logical qubit. We then run these circuits and compare the results with and without error detection.

3.1 Bit-flip error detection

The bit-flip encoding (based on classical Hamming Codes [56]) is the simplest NAED code. In general, a bit-flip encoding on Q physical qubits can be defined by a set $S \subseteq \{0, 1, ..., Q-1\}$. We then define the two integers

$$x = \begin{cases} 0 & S = \emptyset \\ \sum_{i \in S} 2^i & \text{otherwise} \end{cases} \tag{176}$$

$$y = 2^Q - 1 - x \tag{177}$$

The code words are then defined as $|0\rangle_L = |x\rangle$ and $|1\rangle_L = |y\rangle$ where x and y are written in binary. From this, it is apparent that for Q physical qubits, there are 2^Q encodings that we might utilize. For example, a simple set of codewords for $Q = 3$ physical qubits is $|0\rangle_L = |001\rangle$ and $|1\rangle_L = |110\rangle$ as determined by the set $S = \{0\}$.

3.1.1 Encoding, logical gates, decoding, and detectable errors

In order use the bit-flip encoding, three things are required: a way to encode the initial $|0\rangle_L$ state, a method to construct logical gates that operate on $|0\rangle_L$ and $|1\rangle_L$, and a way to decode the states to some set of final states. However, since the logical states also function as final states, we do not need to perform any decoding for NAED to function. To encode $|0\rangle_L$ for a given set S (with Q physical qubits starting in the state $|00...0\rangle$) we simply need to operate on each qubit q_i with a σ_x if $i \in S$. This can be written mathematically as

$$M_S = \prod_{i \in S} \sigma_i \tag{178}$$

where M_S is identified as the encoding matrix.

The construction of logical gates for the bit-flip encoding is slightly more complicated. For Q physical qubits and $S = \emptyset$ we have

$$L_\emptyset(U_3) = \left[\prod_{i=1}^{Q-1} C_x(0, Q-i)\right] U_3 \otimes I_{2^{Q-1}} \left[\prod_{i=1}^{Q-1} C_x(0, i)\right] \tag{179}$$

Note that this logical gate uses $2(Q-1)$ physical C_x gates. For a general set S we have

$$L_S(U_3) = \begin{cases} L_\emptyset(\sigma_x U_3 \sigma_x) & 0 \in S \\ L_\emptyset(U_3) & 0 \notin S \end{cases} \tag{180}$$

Finally, to define the logical C_x gate we must introduce some notation describing the physical qubits of each logical qubit: let $\{r_0, r_1, ..., r_{Q-1}\}$ be the Q physical qubits that correspond to the logical control qubit and let $\{p_0, p_1, ..., p_{Q-1}\}$ be the Q physical qubits that correspond to the logical qubit being operated on. Then for an arbitrary S the logical C_x gate is given by

$$L_S(C_x) = \prod_{i=0}^{Q-1} C_x(r_i, p_i)(I_{2^Q} \otimes M_S) \tag{181}$$

Here, M_S is the encoding matrix defined in equation 178. For proofs that equations 180 and 181 are indeed logical gates, see the next section (section 3.1.2). An example of bit-flip encoding and the associated logical gates is given by encoding a bell state with $Q = 2$ and $S = \{1\}$. In this example $x = 2$ and $y = 1$ with $|0\rangle_L = |10\rangle$ and$|1\rangle_L = |01\rangle$. The circuit includes an encoding step, a logical Hadamard $L_{\{1\}}(H)$, and a logical controlled-not $L_{\{1\}}(C_X)$ as shown in Fig. 17. The final state will be given by $|\psi\rangle = 1/\sqrt{2}(|00\rangle_L + |11\rangle_L) = 1/\sqrt{2}(|0101\rangle + |1010\rangle)$.

After measurement, there are 16 possibilities: 4 states in H^+ and 12 states in H^-. If we measure $|0101\rangle$ then we obtain $|00\rangle_L$. In a similar fashion $|1010\rangle$ yields $|11\rangle_L$, $|0110\rangle$ yields $|01\rangle_L$, and $|1001\rangle$ yields $|10\rangle_L$. If we measure any other of the 12 combinations of 0s and 1s, then we are in H^- and an error is declared.

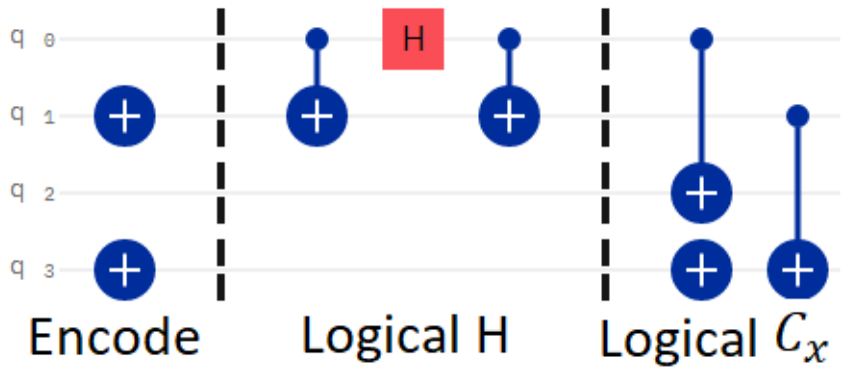

Figure 17: This circuit encodes a bell state for $Q = 2$ physical qubits per logical qubit with codewords $|0\rangle_L = |01\rangle$ and $|1\rangle_L = |10\rangle$. The first set of gates to the left of the first barrier transforms $|0000\rangle$ to $|0101\rangle = |00\rangle_L$. The second set of gates is the logical Hadamard gate $L_{\{1\}}(H)$. The third set of gates is the logical C_x gate $L_{\{1\}}(C_X)$. At the end of this circuit, the qubits will be in the linear combination $|\psi\rangle = 1/\sqrt{2}(|00\rangle_L + |11\rangle_L) = 1/\sqrt{2}(|0101\rangle + |1010\rangle)$.

It is possible to describe the errors that we have a chance to detect while using the bit-flip code. If we model these errors by 2×2 unitary gates, then any error not of the form

$$P(\theta, \phi) = e^{i\phi} \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix} \tag{182}$$

for $\phi, \theta \in \mathbb{R}$, will force any state $|\psi\rangle \in H^+$ into some state $|\psi'\rangle \in H^-$. These detectable errors include σ_x, σ_y, and any other non-phase error that might occur.

3.1.2 Proofs of logical gates

In order to prove the validity of the logical gates described in equations 180-181, we will use the following four identities:

$$(I_2 \otimes \sigma_x)C_x = C_x(I_2 \otimes \sigma_x) \tag{183}$$

$$(\sigma_x \otimes I_2)C_x(\sigma_x \otimes I_2) = (I_2 \otimes \sigma_x)C_x \tag{184}$$

$$\alpha|0\rangle_L + \beta|1\rangle_L = M_S(\alpha|00...0\rangle + \beta|11...1\rangle) \tag{185}$$

where M_S is the encoding matrix given in equation 178. The first two of these equations are easily checked while the third equation follows from the definition of M_S. For the rest of these proofs, we will drop the S subscript and simply refer to the encoding matrix as M. After noting that $M = M^\dagger = M^{-1}$, this also gives us

$$M(\alpha|0\rangle_L + \beta|1\rangle_L) = \alpha|00...0\rangle + \beta|11...1\rangle \tag{186}$$

3.1.2.1 Logical U_3 gate

We will start with the $S = \emptyset$ case (with codewords $|0\rangle_L = |00...0\rangle$ and $|1\rangle_L = |11...1\rangle$) and from there prove the general case. To prove that $L_\emptyset(U_3)$ is a logical U_3 gate, we must show that if $U_3(\alpha|0\rangle + \beta|1\rangle) = \tau|0\rangle + \delta|1\rangle$ then $L_\emptyset(U_3)(\alpha|0\rangle_L + \beta|1\rangle_L) = \tau|0\rangle_L + \delta|1\rangle_L$. With the first part of $L_\emptyset(U_3)$ we have

$$|\psi_1\rangle = \left[\prod_{i=1}^{Q-1} C_x(0,i)\right](\alpha|0\rangle_L + \beta|1\rangle_L)$$

$$= \alpha|00...0\rangle + \beta|10...0\rangle = (\alpha|0\rangle + \beta|1\rangle) \otimes |00...0\rangle \tag{187}$$

where the state $|00...0\rangle$ in the resulting expression has $Q-1$ zeros. Then applying the U_3 gate gives us

$$|\psi_2\rangle = U_3 \otimes I_{2^{q-1}}|\psi_1\rangle = U_3 \otimes I_{2^{q-1}}(\alpha|0\rangle + \beta|1\rangle) \otimes |00...0\rangle$$

$$= (\tau|0\rangle + \delta|1\rangle) \otimes |00...0\rangle \tag{188}$$

The final part of $L_\emptyset(U_3)$ gives us

$$|\psi_3\rangle = \left[\prod_{i=1}^{Q-1} C_x(0, Q-i)\right] |\psi_2\rangle$$

$$= \left[\prod_{i=1}^{Q-1} C_x(0, Q-i)\right] (\tau|0\rangle + \delta|1\rangle) \otimes |00...0\rangle$$

$$= \tau|00...0\rangle_L + \delta|11...1\rangle_L = \tau|0\rangle_L + \delta|1\rangle_L \tag{189}$$

as desired.

For the general case, if $0 \notin S$ then

$$L_S(U_3)(\alpha|0\rangle_L + \beta|1\rangle_L) = L_\emptyset(U_3)(\alpha|0\rangle_L + \beta|1\rangle_L)$$

$$= L_\emptyset(U_3)MM(\alpha|0\rangle_L + \beta|1\rangle_L) = L_\emptyset(U_3)M(\alpha|00...0\rangle + \beta|11...1\rangle) \tag{190}$$

We also know that $[L_\emptyset(U_3), M] = \hat{0}$ since every C_x in $L_\emptyset(U_3)$ is controlled by q_0, the physical U_3 gate in $L_\emptyset(U_3)$ acts on q_0, and $0 \notin S$ which allows us to use equation 183. Thus, equation 190 becomes

$$= L_\emptyset(U)M(\alpha|00...0\rangle + \beta|11...1\rangle)$$

$$= ML_\emptyset(U)(\alpha|00...0\rangle + \beta|11...1\rangle)$$

$$= M(\tau|00...0\rangle + \delta|11...1\rangle) = \tau|0\rangle_L + \delta|1\rangle_L \tag{191}$$

For the case where $0 \in S$, let $M^{'} = \sigma_0 M$. Then using the fact $\sigma_i = \sigma_i^{-1}$, we have

$$L_\emptyset(\sigma_x U_3 \sigma_x)M = \sigma_0\sigma_0 L_\emptyset(\sigma_x U_3 \sigma_x)\sigma_0 M^{'} \tag{192}$$

Then using equation 184 $2(Q-1)$ times (one for each physical C_x in $L_\emptyset(\sigma_x U \sigma_x)$) and simplifying using equation 183 we get

$$= \sigma_0 M^{'} M^{'} L_\emptyset(U_3) M^{'} = M L_\emptyset(U_3) \tag{193}$$

Using this relation we may now conclude

$$L_S(U_3)(\alpha|0\rangle_L + \beta|1\rangle_L) = L_\emptyset(\sigma_x U \sigma_x)(\alpha|0\rangle_L + \beta|1\rangle_L)$$

$$= L_\emptyset(\sigma_x U \sigma_x) M M(\alpha|0\rangle_L + \beta|1\rangle_L) = M L_\emptyset(U)(\alpha|00...0\rangle + \beta 0|11...1\rangle)$$

$$= M(\tau|00...0\rangle + \delta|11...1\rangle) = \tau|0\rangle_L + \delta|1\rangle_L \tag{194}$$

3.1.2.2 Logical C_x gate

In a similar manner to the previous proof, We will start with the $S = \emptyset$ case and from there prove the general case. To prove that $L_\emptyset(C_x)$ is a logical C_x gate, we must show that $L_\emptyset(C_x)(\alpha|00\rangle_L + \beta|01\rangle_L + \tau|10\rangle_L + \delta|11\rangle_L) = \alpha|00\rangle_L + \beta|01\rangle_L + \delta|10\rangle_L + \tau|11\rangle_L$. As in the main paper, let $\{r_0, r_1, ..., r_{Q-1}\}$ be the Q physical qubits the make up the logical control qubit and let $\{p_0, p_1, ..., p_{Q-1}\}$ be the Q physical qubits that make up the logical target bit. For ease of notation, the full ket corresponding to these $2Q$

physical qubits shall be written as $|r_0 r_1 ... r_{Q-1}; p_0 p_1 ... p_{Q-1}\rangle$, note the semi-colon used to distinguish between both sets of qubits. We then have

$$C_x(r_0, p_0)(\alpha|00\rangle_L + \beta|01\rangle_L + \tau|10\rangle_L + \delta|11\rangle_L)$$

$$= C_x(r_0, p_0)(\alpha|00...0; 00...0\rangle + \beta|00...0; 11...1\rangle$$

$$+\tau|11...1; 00...0\rangle + \delta|11...1; 11...1\rangle)$$

$$= \alpha|00...0; 00...0\rangle + \beta|00...0; 11...1\rangle$$

$$+ \tau|11...1; 10...0\rangle + \delta|11...1; 01...1\rangle \tag{195}$$

Repeating this process for the other $Q-1$ physical C_x gates in $L_\emptyset(C_x)$ gives us the desired result. To generalize to all S, note that by equations 183 and 184 we have

$$L_S(C_x) = L_\emptyset(C_x)(I_{2^Q} \otimes M) = (I_{2^Q} \otimes M)L_\emptyset(C_x)$$

$$= (M \otimes I_{2^Q})L_\emptyset(C_x)(M \otimes I_{2^Q})$$

$$= (M \otimes MM)L_\emptyset(C_x)(M \otimes I_{2^Q}) = (M \otimes M)L_\emptyset(C_x)(M \otimes M) \tag{196}$$

Using this equivalent definition for $L_S(C_x)$, we get

$$L_S(C_x)(\alpha|00\rangle_L + \beta|01\rangle_L + \delta|10\rangle_L + \tau|11\rangle_L)$$

$$= (M \otimes M) L_{\emptyset}(C_x)(M \otimes M)(\alpha|00\rangle_L + \beta|01\rangle_L + \delta|10\rangle_L + \tau|11\rangle_L)$$

$$= (M \otimes M) L_{\emptyset}(C_x)(\alpha|00...000...0\rangle + \beta|00...011...1\rangle)$$

$$+ \delta|11...100...0\rangle + \tau|11...111...1\rangle) \tag{197}$$

by equation 186. But this is precisely the relationship we just showed (the $S = \emptyset$ encoding). Thus, it simplifies to

$$= (M \otimes M)(\alpha|00...000...0\rangle + \beta|00...011...1\rangle)$$

$$+\tau|11...100...0\rangle + \delta|11...111...1\rangle)$$

$$= \alpha|00\rangle_L + \beta|01\rangle_L + \tau|10\rangle_L + \delta|11\rangle_L \tag{198}$$

as desired.

3.2 An application of the bit-flip encoding

The logical U_3 and C_x gates are used to construct $GHZ(N, Q)$ circuits where N is the number of logical qubits and Q is the number of physical qubits per logical qubit. Gates corresponding to this circuit are given by

$$GHZ(N, 1) = \left[\prod_{i=0}^{N-1} C_x(i, i+1)\right](H \otimes I_{2^{N-1}}) \tag{199}$$

which uses $N - 1$ physical C_x gates and a single Hadamard gate. Note that this is not the only way to create the $GHZ(N, 1)$ state [57].

In order to turn the $GHZ(N, 1)$ circuit into the $GHZ(N, Q)$ circuit (using bit-flip

encoding) we simply replace all physical gates in the $GHZ(N,1)$ circuit with their logical equivalents from equations 180 and 181. Of course, we also have to decide which set S we will use for the encoding. In order to balance the number of 0s and 1s that make up $|0\rangle_L$ and $|1\rangle_L$, we will use $S = \{0, 1, ..., \lceil Q/2 \rceil\}$. The last thing we need to do is slightly simplify the resulting circuit before implementation. For example, for the circuit $GHZ(2,2)$ we have the the circuit given in Fig. 17, however the first C_x gate as well as the two σ_x gates on q_3 are redundant. Removing these gives us the circuit in Fig. 18, and a similar simplification will be used for all experimental runs.

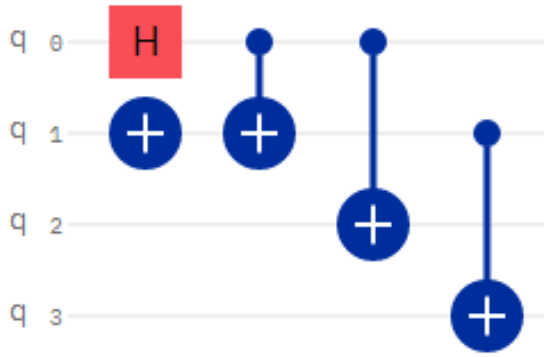

Figure 18: The simplified $GHZ(2,2)$ circuit. This circuit is identical to the circuit in Fig. 17 except that the first C_x gate and bottom σ_x gates have been removed as redundant. This does not change the overall state $|\psi\rangle = 1/\sqrt{2}(|0101\rangle + |1010\rangle)$ that this circuit produces.

3.2.1 Experimental design, results, and discussion

For our experiments we will use the similarity measure (equation 29 in section 1.6.2) and compare the experimentally determined PDF for the $GHZ(N,Q)$ circuit with theoretical expected PDF. For the simple $GHZ(N,Q)$ circuit, this theoretical PDF is easily computed and will always be an equally split between two states in the full $\mathcal{H}_{NQ}$ circuit space.

For each $(N, Q) \in \{2, 3, 4, 5\} \times \{1, 2, 3, 4, 5\}$, the $GHZ(N, Q)$ circuit was submitted at optimization level 1 for 2^{13} shots. Two similarity measures are computed using equation 29. The first retains all of the experimental results and represents the full similarity measure of the encoded circuit, μ_{Full}. The second omits all experimental results for which an error is detected and is the NAED result, denoted μ_{NAED}. This process was repeated between 220 and 230 times for each circuit, and averaged to yield μ_{Full} and μ_{NAED} for each (N, Q) pair.

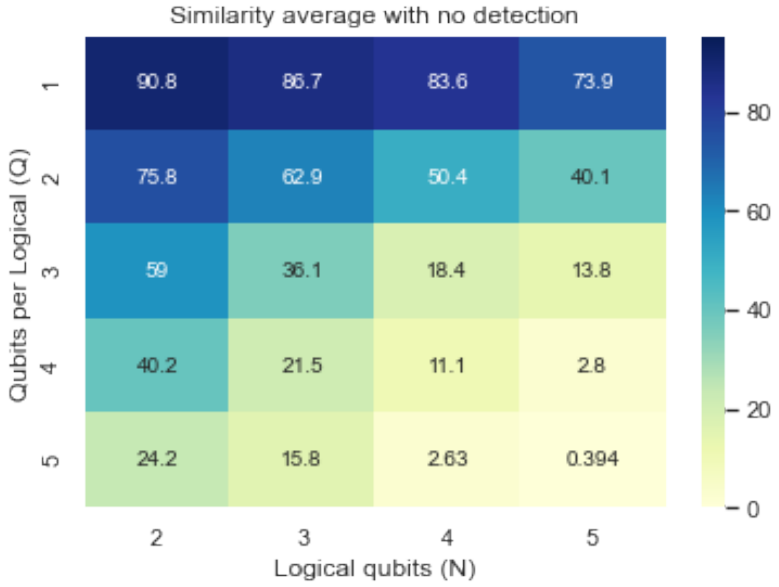

Figure 19: The similarity measure μ_{Full} of the $GHZ(N, Q)$ circuits over the input space $(N, Q) \in \{2, 3, 4, 5\} \times \{1, 2, 3, 4, 5\}$. Not surprisingly, the best results occur at $GHZ(2, 1)$ with a similarity measure of 90.8. The similarity decreases as both N and Q increase, with the worst similarity of 0.4 for $N = Q = 5$.

Values of μ_{Full} are shown in Fig. 19 where the best run overall is the $GHZ(2, 1)$ circuit with a similarity measure of 90.8. The similarity then decreases as both N and Q increase. Values of μ_{NAED} are shown in Fig. 20 and

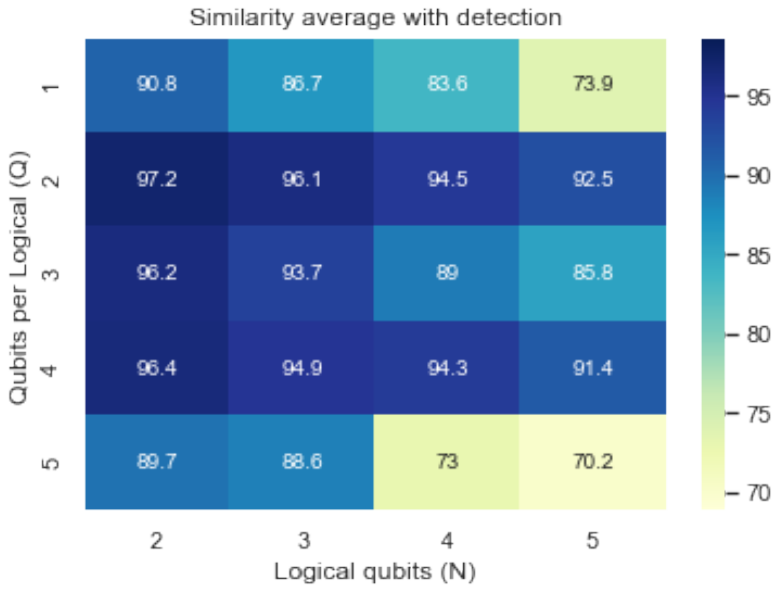

Figure 20: The similarity measure μ_{NAED} of the $GHZ(N,Q)$ circuits over the input space $(N,Q) \in \{2,3,4,5\} \times \{1,2,3,4,5\}$. The highest similarity is now 97.2 for $GHZ(2,2)$ with the circuit from Fig. 18. while the greatest incr ase in similarity from the unencoded circuit occurs between $GHZ(5,1)$ and $GHZ(5,2)$.

demonstrate that NAED is a viable option for improving quantum computation. As seen in Fig. 20, values of μ_{NAED} for $GHZ(N,Q)$ where $2 \leq Q \leq 4$ are all greater than μ_{NAED} for the unencoded $GHZ(N,1)$ circuit. It is not until $Q = 5$ that NAED produced lower values of μ_{NAED} than $Q = 1$ where no error detection is performed.

Also seen in Fig. 20 is evidence that even values of Q outperform odd values of Q. For example, the similarity of $GHZ(N,4)$ is greater than the similarity of $GHZ(N,3)$ for all N. This behavior is caused by an assymmetry in the number of 0's and 1's that make up the codewords for odd values of Q. Since super conducting qubits will naturally decohere towards the $|0\rangle$ state [58, 59], this leads to an asymmetry in measured states for odd Q.

A primary cost of NAED is the number of runs that are discarded when an error is

declared. The total number of runs that are not included in the calculation of μ_{NAED} increases with N and Q as seen in Fig. 21 where only a few percent of the runs are retained for larger N and Q.

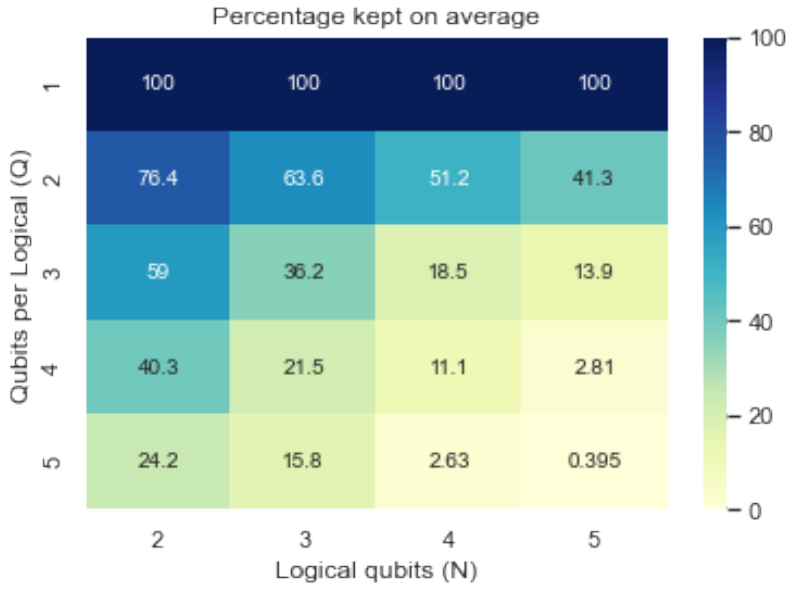

Figure 21: The percentage of runs retained for each $GHZ(N, Q)$ circuit over the input space $(N, Q) \in \{2, 3, 4, 5\} \times \{1, 2, 3, 4, 5\}$ after error detection has been performed. For $Q = 1$, there are no runs removed. The next highest percentage is for the $GHZ(2, 2)$ circuit (given by Fig. 18) at 76.4% kept. From here, the percentage retained decreases as both N and Q increase.

It is interesting to note the values of μ_{Full} in Fig. 19 and percentage retained in Fig. 21 approach each other in value as both N and Q increase. This observation is used to show that the ratio of false positives to the total number of measurements decreases as N and Q increase. To illustrate why this is the case, define the following

- $T =$ the total number of measurements
- $r_0 =$ the total number of $|00...0\rangle_L$ measured

- r_1 = the total number of $|11...1\rangle_L$ measured
- r_a = the total number of logical states other than $|00...0\rangle_L$ or $|11...1\rangle_L$ measured
- r_b = the total number of states which don't fall into any logical state measured

Here, r_b are errors that NAED would catch while r_a are errors that NAED would not catch. For example, with $N = 3$ and $Q = 2$ the state $|010110\rangle = |001\rangle_L$ is an invalid GHZ state but will not be caught by error detection. Also note that $T = r_0 + r_1 + r_a + r_b$. Then the percentage of measurements kept is given by

$$P_{Kept} = 1 - \frac{r_b}{T} \tag{200}$$

Without detection, μ_{Full} can be written as

$$\mu_{Full} = 1 - \frac{1}{2}\left(\left|\frac{1}{2} - \frac{r_0}{T}\right| + \left|\frac{1}{2} - \frac{r_1}{T}\right| + \frac{r_a}{T} + \frac{r_b}{T}\right) \tag{201}$$

The absolute values may be ignored since the chance that $\frac{r_0}{T} > \frac{1}{2}$ or $\frac{r_1}{T} > \frac{1}{2}$ is negligible for higher N and Q. Then equation 201 becomes

$$\mu_{Full} = 1 - \frac{1}{2T}(T - r_0 - r_1 + r_a + r_b)$$

$$= 1 - \frac{1}{2T}(T - r_0 - r_1 - r_a - r_b + 2r_a + 2r_b)$$

$$= 1 - \frac{r_a}{T} - \frac{r_b}{T} = P_{Kept} - \frac{r_a}{T} \tag{202}$$

As N and Q increase, the value of r_a/T in equation 202 goes to zero.

3.3 Conclusion

As an initial endeavour, these experiments have shown that NAED is a viable option for increasing the fidelity of quantum circuits. At its best, it was able to improve the similarity measure of a $GHZ(5,1)$ circuit from $\mu_{Full} = 73.9$ to $\mu_{NAED} = 92.5$ using a $GHZ(5,2)$ circuit. Additionally, we have shown that the ratio of false positives to the total number of measurements decreases as the number of logical qubits and the number of physical qubits per logical qubits increase. One drawback to this experiment however is that the GHZ circuit does not utilize phase in any way. In the next chapter we will implement NAED with a circuit in which phase is a critical part of the final result.

IV. The XY, YZ, and ZX encodings

Preamble

In the previous chapter, we described and experimentally verified the bit-flip encoding using GHZ states. While useful as a demonstration that NAED is a viable possibility for quantum error detection, the experiments were nonetheless lacking due to the fact that the qubit phases were completely ignored. The bit-flip encoding is unable to catch phase errors and the GHZ states do not use phase anywhere in the algorithm.

4.1 Motivation

The Gottesman-Knill Theorem [60] states that any quantum circuit composed of gates entirely from the Clifford group can be efficiently simulated on a classical computer. The Clifford group can be generated by the following set of gates [61]

$$C_x = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \quad H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \tag{203}$$

(this includes the Pauli matrices). Thus, any quantum circuit that uses only these gates is unable to provide anything other than a polynomial speedup versus a classical computer. While this may suffice for some applications, it illustrates the importance general phase gates have for quantum algorithms.

In order to further develop NAED, we found it necessary to investigate codes which can detect phase errors. In this chapter, we will describe three different codes,

the process we went through in creating them, and the types of errors they can and cannot detect. Additionally, we will implement these codes on the circuit in Fig. 22

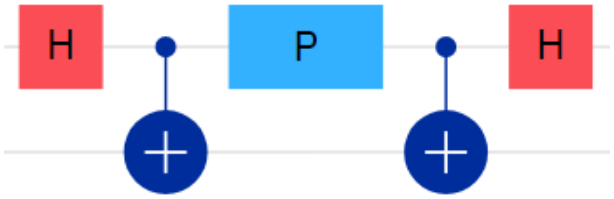

Figure 22: The P gate is a phase gate of phase ϕ and the probabilities of measurement are given by $P(|00\rangle) = \cos(\phi/2)^2$ and $P(|10\rangle) = \sin(\phi/2)^2$.

This circuit is important because unlike the GHZ circuits used in the bit-flip encoding, this circuit utilizes phase in an appreciable manner. Performing the matrix math and simplifying, we find that the probability of measuring $|00\rangle$ is given by $\cos(\phi/2)^2$ while the probability of measuring $|10\rangle$ is given by $\sin(\phi/2)^2$. Here, ϕ is the phase of the phase gate P.

4.2 XY, YZ, and ZX encodings

The bit-flip error detection code can detect any error except for phase errors. Although simple, it easily expanded to a large number of physical qubits per logical qubit and has a well designed process for implementation. In fact, the bit-flip error detection can be thought of as a generalization of a 2-qubit code we call the XY code. This code is named to symbolize that it can detect σ_x errors and σ_y errors but not σ_z errors. In a similar manner, we have also developed YZ and ZX encodings.

4.2.1 The XY encoding

The XY encoding is a specific form of the bit-flip encoding where $Q = 2$ and $S = \{0\}$. This gives the logical states

$$|0\rangle_L = |01\rangle \text{ and } |1\rangle_L = |10\rangle \tag{204}$$

In order to show which errors this encoding can and cannot detect, we complete the following procedure:

- Let $M \in \chi(XY)$ be given by $M = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$.
- Compute both S_0 and S_1 (equations 99 and 100) in

$$M^\dagger (U \otimes I_2) M = \begin{pmatrix} Q_0 & R_0 \\ S_0 & T_0 \end{pmatrix} \tag{205}$$

$$M^\dagger (I_2 \otimes U) M = \begin{pmatrix} Q_1 & R_1 \\ S_1 & T_1 \end{pmatrix} \tag{206}$$

  where U is the most general 2×2 (up to a phase) unitary matrix given by

$$U = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -e^{i\phi} \sin\left(\frac{\theta}{2}\right) \\ e^{i\lambda} \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) e^{i\lambda + i\phi} \end{pmatrix} \tag{207}$$

- Recall the function $f(A)$ (equation 208) defined as

$$f\left[\begin{pmatrix} a & b \\ c & d \end{pmatrix}\right] = |a|^2 + |b|^2 + |c|^2 + |d|^2 \tag{208}$$

  Compute

$$f(S_0)f(S_1) = 4\sin\left(\frac{\theta}{2}\right)^4 \tag{209}$$

We call this the functional equation of the encoding XY and it is equal to 0 if and only if at least one of S_0 and S_1 is the zero matrix. That is, the functional equation is equal to 0 if and only if U is an error of XY.

- Solving the functional equation for 0 in terms of (θ, λ, ϕ) we find

$$Error_{XY} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\tau} \end{pmatrix} \tag{210}$$

for any $\tau \in \mathbb{R}$.

We can also describe how to encode, decode, and create logical gates for the XY encoding. In Fig. 23 we show circuit diagrams for each of these objects.

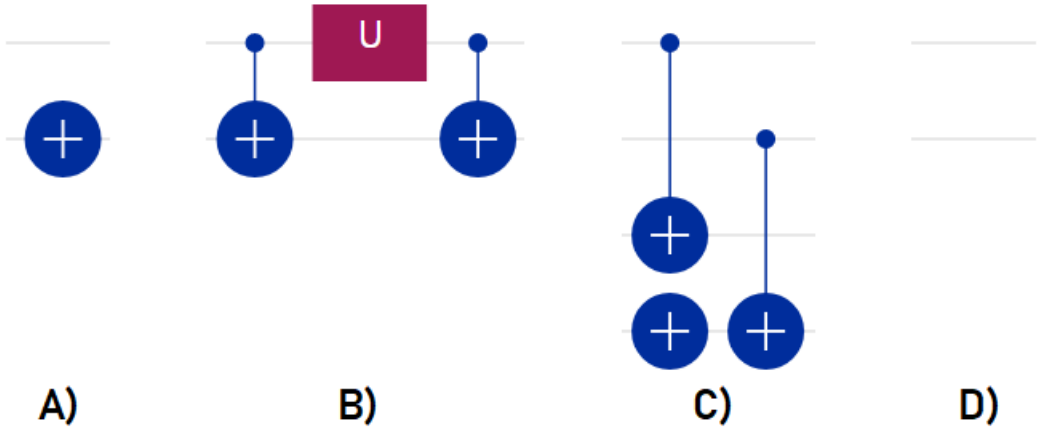

Figure 23: The physical gates used for encoding (A), $L(U)$ (B), $L(C_x)$ (C), and decoding (D) for the XY encoding. Notice that the decoding step is blank as the logical states are valid final states.

The encoding step A) is a σ_x gate on the second physical qubit, the $L(U)$ gate B) is

given by a physical U gate and two C_x gates, the $L(C_x)$ gate is given by a σ_x gate and two C_x gates, and the decoding step is blank as the logical states given in equation 204 also function as final states.

4.2.2 The YZ encoding

As the name suggests, the YZ encoding can detect σ_y and σ_z errors but not σ_x errors. It is given by the code words

$$\begin{aligned} |0\rangle_L &= \frac{1}{2}\left(|00\rangle - |01\rangle + |10\rangle - |11\rangle\right) \\ |1\rangle_L &= \frac{1}{2}\left(|00\rangle + |01\rangle - |10\rangle - |11\rangle\right) \end{aligned} \tag{211}$$

Working through a similar set of steps as above reveals that the functional equation is given by

$$0 = \left(1 - \cos(\theta)\cos(\lambda)\cos(\phi) + \sin(\lambda)\sin(\phi)\right)^2 \tag{212}$$

Solving this in terms of (θ, λ, ϕ) give two solution sets: $(0, -\tau, \tau)$ and $(\pi, \tau, \tau + \pi)$ for any $\tau \in \mathbb{R}$. The first of these gives the matrix I_2, an ignorable error. The second one gives us our error

$$Error_{YZ} = e^{i\tau} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = e^{i\tau}\sigma_x \tag{213}$$

In Fig. 24 we also display the circuit diagrams for the various elements needed for the YZ encoding. The encoding step A) uses two Hadamard gates and a σ_x gate, the logical U gate B) uses two physical C_x gates and various U_3 gates, the logical C_x uses two physical C_x gates and various single qubit gates, and the decoding step D) uses two Hadamard gates.

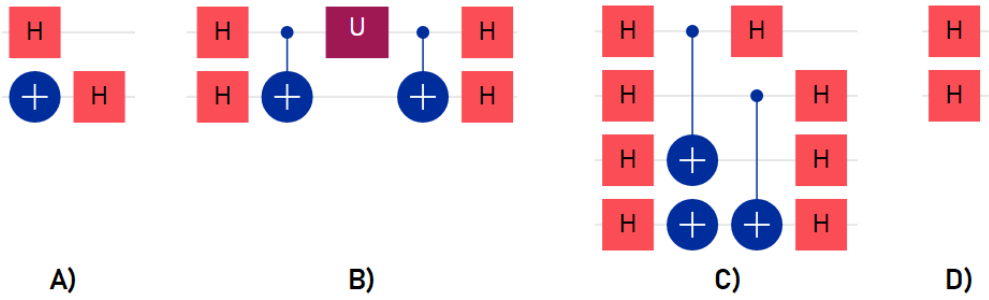

Figure 24: The physical gates used for encoding (A), $L(U)$ (B), $L(C_x)$ (C), and decoding (D) for the YZ encoding.

Of particular note, the 4×4 matrix defined for the encoding step above is not in $\chi(YZ)$. Written out, it is given by

$$Encoding_{YZ} = (H \otimes H)(I_2 \otimes \sigma_x) = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \end{pmatrix} \notin \chi(YZ) \tag{214}$$

Still, this is a valid encoding matrix since the first column is equal to $|0\rangle_L$ and each logical qubit starts in the $|00\rangle$ state.

4.2.3 The ZX encoding

The final encoding we shall investigate is the ZX encoding given by

$$\begin{aligned} |0\rangle_L &= \frac{1}{2}\left(|00\rangle - i|01\rangle + i|10\rangle + |11\rangle\right) \\ |1\rangle_L &= \frac{1}{2}\left(|00\rangle + i|01\rangle - i|10\rangle + |11\rangle\right) \end{aligned} \tag{215}$$

The functional equation for this encoding is similar to the YZ encoding

$$0 = (1 - \cos(\lambda)\cos(\phi) + \cos(\theta)\sin(\lambda)\sin(\phi))^2 \tag{216}$$

Solving this equation gives us the error

$$Error_{ZX} = e^{i\tau} \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} = e^{i\tau}\sigma_y \tag{217}$$

In Fig. 25 we display the circuit diagrams for the encoding, decoding, and logical gates of the ZX encoding. The matrix S is defined as

$$S = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -i \\ 1 & i \end{pmatrix} \tag{218}$$

but otherwise the elements are identical to the YZ encoding.

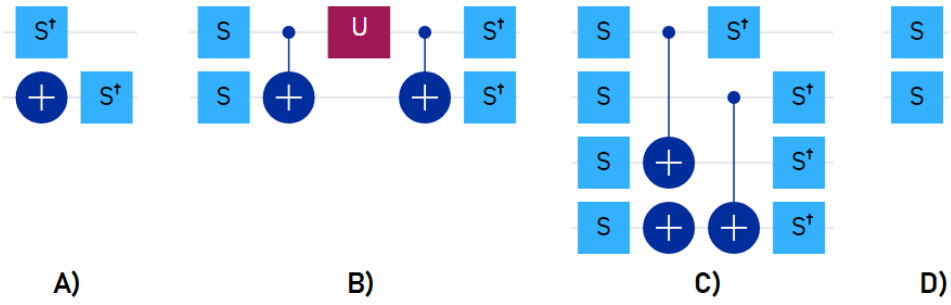

Figure 25: The physical gates used for encoding (A), $L(U)$ (B), $L(C_x)$ (C), and decoding (D) for the ZX encoding. S is the matrix given in equation 218.

In fact, all three of these encodings can be described in much the same way: if we replace S in Fig. 25 with Hadamard gates, we get the YZ encoding. If we replace it with identity gates, we get the XY encoding.

4.2.4 General $U \otimes U$ encodings

So far we have presented three two qubit encodings, ways to encode and decode them, logical gates associated with them, and which errors they are unable to detect. Generalizing this idea, let us define a general encoding G with logical states

$$|0\rangle_L = \frac{1}{2}e^{-i\lambda_0}\sin(\theta_0)|00\rangle + \frac{1}{2}(\cos(\theta_0)+1)e^{-i(\lambda_0+\phi_0)}|01\rangle$$
$$\frac{1}{2}(\cos(\theta_0)-1)e^{-i(\lambda_0+\phi_0)}|10\rangle - \frac{1}{2}\sin(\theta_0)e^{-i(\lambda_0+2\phi_0)}|11\rangle \quad (219)$$

$$|1\rangle = \frac{1}{2}e^{-i\lambda_0}\sin(\theta_0)|00\rangle + \frac{1}{2}(\cos(\theta_0)-1)e^{-i(\lambda_0+\phi_0)}|01\rangle$$
$$\frac{1}{2}(\cos(\theta_0)+1)e^{-i(\lambda_0+\phi_0)}|10\rangle - \frac{1}{2}\sin(\theta_0)e^{-i(\lambda_0+2\phi_0)}|11\rangle \quad (220)$$

To get the circuit elements, let S in Fig. 25 be given by the general matrix

$$S = \begin{pmatrix} \cos\left(\frac{\theta_0}{2}\right) & -e^{i\phi_0}\sin\left(\frac{\theta_0}{2}\right) \\ e^{i\lambda_0}\sin\left(\frac{\theta_0}{2}\right) & \cos\left(\frac{\theta_0}{2}\right)e^{i\lambda_0+i\phi_0} \end{pmatrix} \quad (221)$$

Then the functional equation of G is given by

$$0 = \frac{1}{16}\left[4\sin^2(\theta_0)\sin(\phi_0-\phi_1)\sin(\lambda_1+\phi_0)+\cos(\theta_1)\right.$$
$$\left(4\sin^2(\theta_0)\cos(\phi_0-\phi_1)\cos(\lambda_1+\phi_0)+2\right)+2\sin(2\theta_0)\sin(\theta_1)(\cos(\phi_0-\phi_1)-\cos(\lambda_1+\phi_0))$$
$$\left.+\cos(2\theta_0-\theta_1)+\cos(2\theta_0+\theta_1)-4\right]^2 \quad (222)$$

with a non-detectable error of

$$Error_G = \begin{pmatrix} \cos\left(\frac{\theta_1}{2}\right) & -e^{i\phi_1}\sin\left(\frac{\theta_1}{2}\right) \\ e^{i\lambda_1}\sin\left(\frac{\theta_1}{2}\right) & \cos\left(\frac{\theta_1}{2}\right)e^{i\lambda_1+i\phi_1} \end{pmatrix} \quad (223)$$

Interestingly, the variable λ_0 does not appear in the functional equation.

4.2.5 Catastrophic cancellation and barriers

Although mathematically sound, the YZ and ZX encodings present an issue when used for error detection. When implemented without any extra precautions, the native qiskit back end will force catastrophic cancellation. Catastrophic cancellation is less an issue with NAED than the native IBM qiskit compiler. Unless one manually implements every physical gate and timing in a circuit, the compiler will attempt to perform simple cancellations where possible. Although normally good practice, this can be detrimental when trying to implement NAED. For example, in Fig. 26 we have implemented a bell circuit using the YZ encoding.

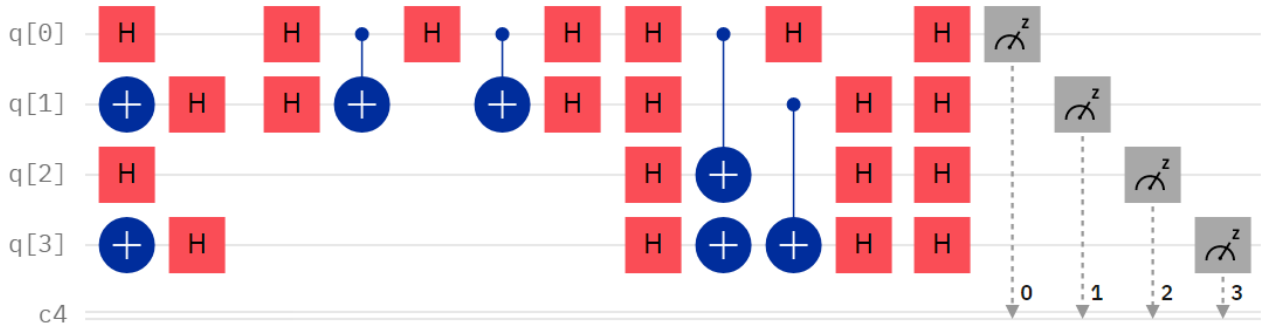

Figure 26: The bell circuit implemented using the YZ encoding without simplification.

The problem is that qiskit will naturally cancel the Hadamard gates, leaving the circuit in Fig. 27.

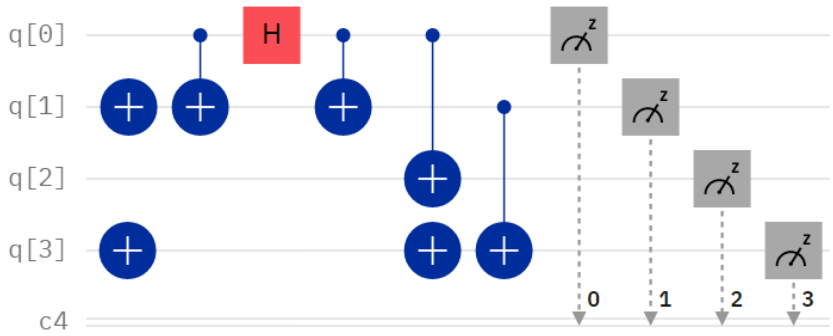

Figure 27: The bell circuit implemented using the YZ encoding with catastrophic cancellations of the Hadamards.

This circuit is exactly the same as if we had implemented the XY encoding, removing any possible advantages the YZ encoding might have given us. The solution to the issue of catastrophic cancellation is to add barriers into the circuit between circuit elements. This stops the compiler from performing catastrophic cancellation and allows us to test different NAED encodings. Taking the YZ encoding of a bell circuit from above, we now have the circuit diagram in Fig. 28

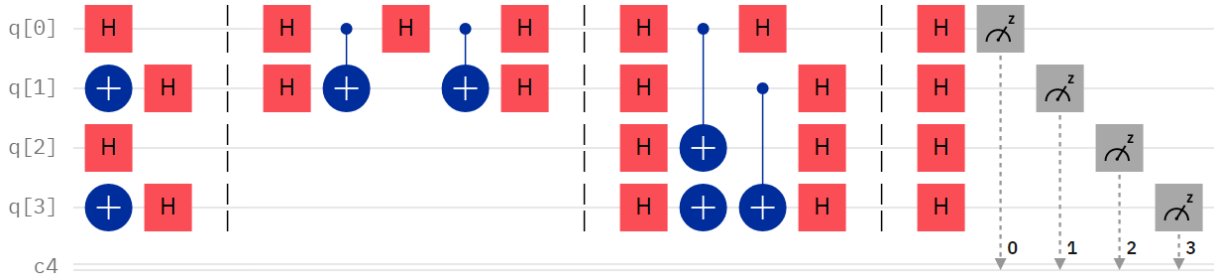

Figure 28: Barriers have been added between every circuit element. These include logical gates, encoding, and decoding.

Barriers have been placed between the encoding step, the logical gates, and the decoding step. In the next section, we will use this technique to test the XY, YZ, and ZX encodings on a phase dependent circuit.

4.3 Experiment design

As previously stated, we will implement the XY, YZ, and ZX encodings on the circuit in Fig. 22 as well as an unencoded circuit as a control. Without any encodings, this circuit in matrix form is

$$Circuit = (H \otimes I_2)C_x(P(\phi) \otimes I_2)C_x(H \otimes I_2) \tag{224}$$

Here, $P(\phi)$ is a phase gate of angle ϕ which is our independent variable and will be varied from 0 to 2π. Specifically, ϕ will be taken from the set

$$S = \left\{ \frac{2\pi \cdot 0}{200}, \frac{2\pi \cdot 1}{200}, \frac{2\pi \cdot 2}{200}, ..., \frac{2\pi \cdot 199}{200} \right\} \tag{225}$$

To implement any of the XY, YZ, or ZX encodings, replace every gate in Fig. 22/Equation 224 with the corresponding logical gates from Figs. 23, 24, and 25. Additionally, add barriers between every layer in the circuit to avoid catastrophic cancellation. For example, for the YZ encoding we have the Fig. 29

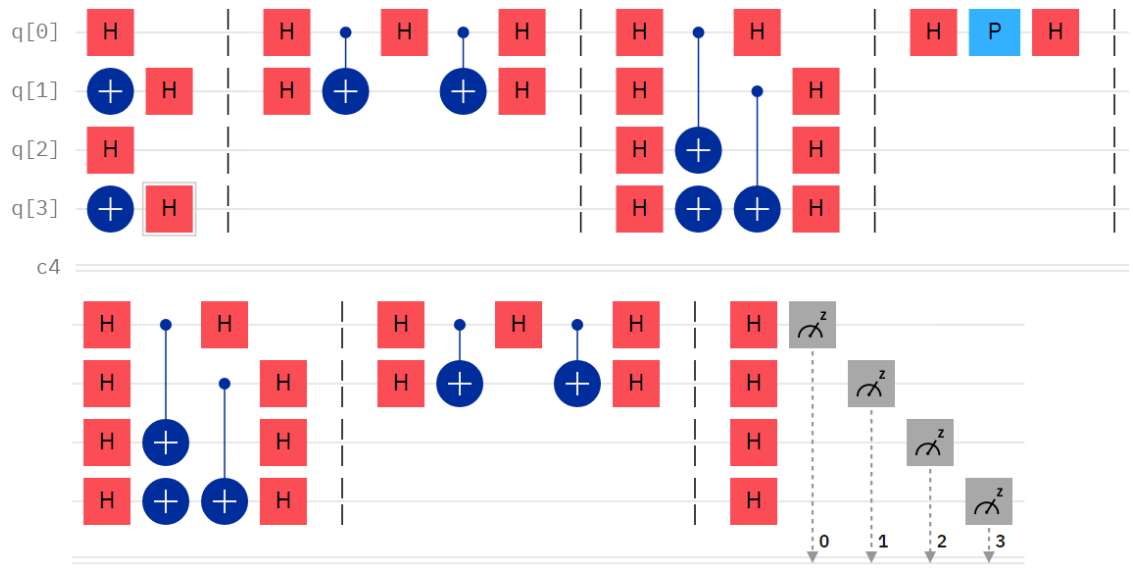

Figure 29: The full YZ circuit for implementation. The P gate is a phase gate with input $\phi \in S$.

where the P gate is a phase gate with input $\phi \in S$. We then implement these circuits on *ibmq_montreal* [62] according to the following procedure

1. Encode the circuit using the XY encoding
2. For each angle in S, submit the circuit at optimization level 3 for 2^{13} shots
3. Tabulate the similarity measure before and after applying NAED
4. Repeat $1) - 3)$ for the YZ and ZX encodings
5. Repeat $1) - 2)$ for the unencoded circuits
6. Tabulate the similarity measure
7. Repeat $1) - 6)$ 20 times and average the results

Overall, this experiment uses the quantum computer over 130 million times.

4.4 Results and analysis

Before performing NAED, the average similarities came out to

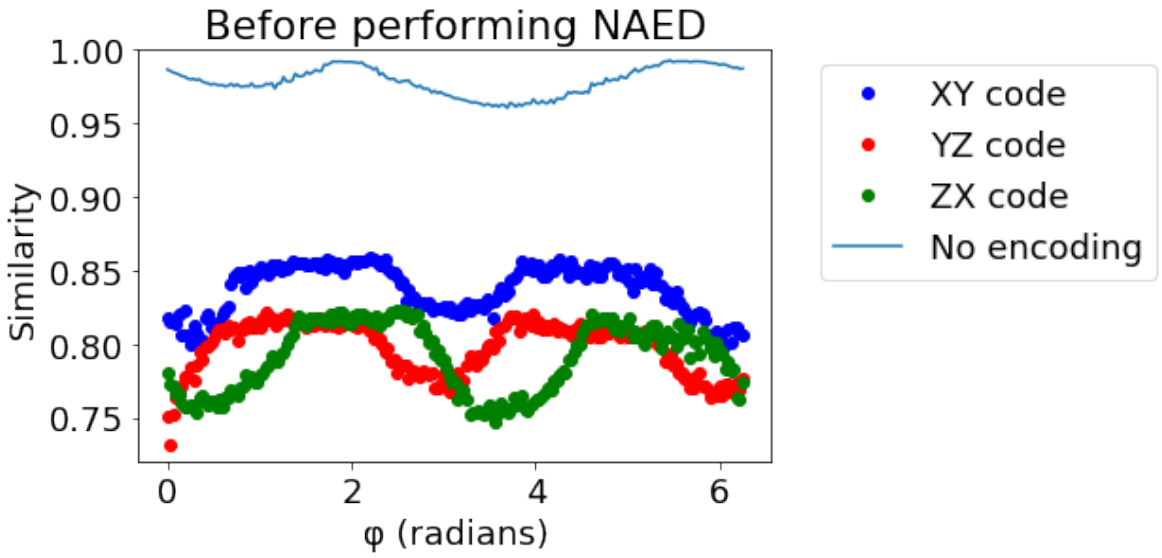

Figure 30: The average similarities before performing NAED compared to the unencoded circuit.

Taking another average can give us a picture of the overall performance of each experiment. These averages of averages are presented in Table 4:

Encoding	Average
XY	0.837
YZ	0.798
ZX	0.791
None	0.978

Table 4: The mean of the average similarities for the XY, YZ, ZX, and unencoded circuits without performing NAED.

After performing NAED, the average similarities were

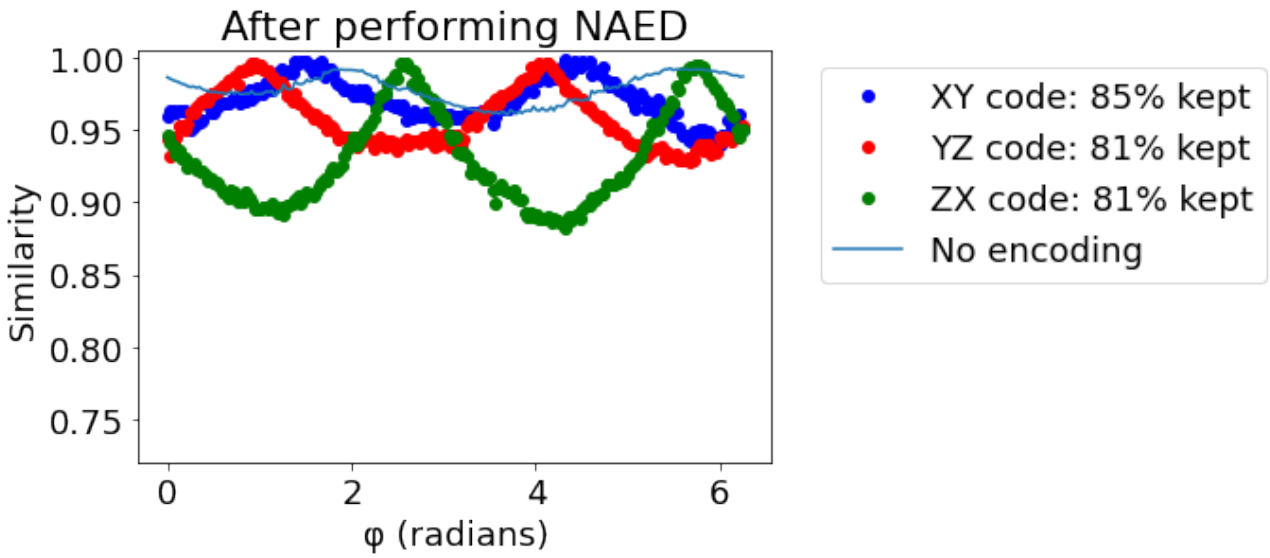

Figure 31: The average similarities after performing NAED compared to the unencoded circuit. The amount of runs kept per encoded circuit is also shown.

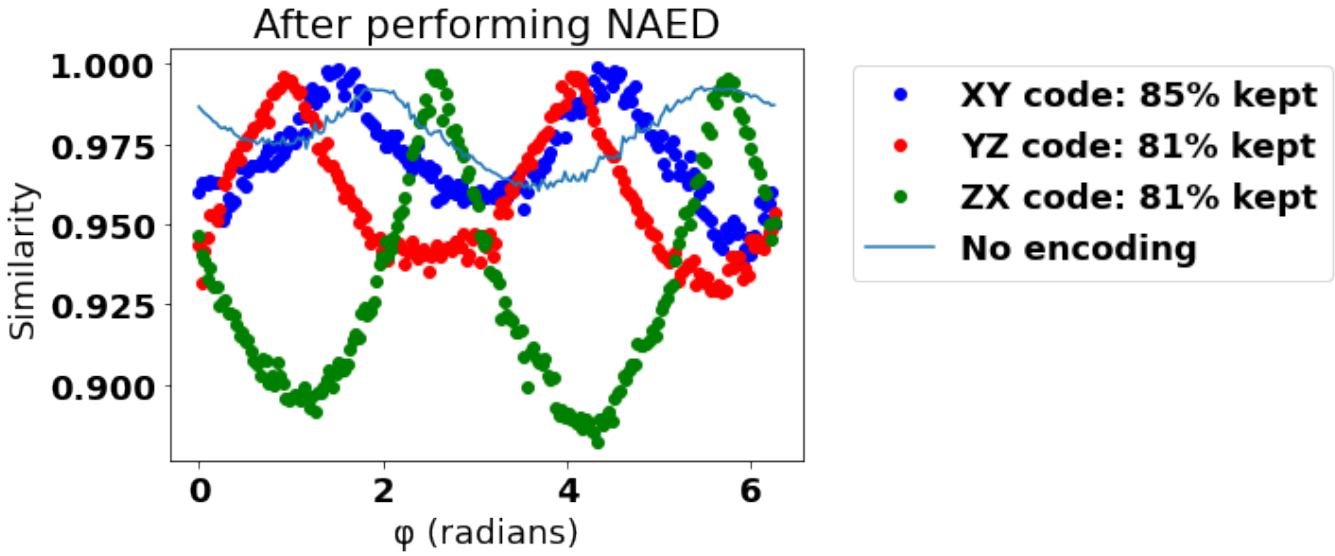

Figure 32: The previous plot with a smaller y-axis.

There has clearly been a large increase in similarity although we had to sacrifice 15% to 20% of the total number of runs. Computing the averages again gives use the data in Table 5:

Encoding	Average
XY	0.971
YZ	0.958
ZX	0.931
None	0.978

Table 5: The mean of the average similarities for the XY, YZ, ZX, and unencoded circuits after performing NAED.

For these encoded circuits, the average increase in similarity is .144.

The plots in Fig. 32 seem to be following some 2π periodic curve: the function

$$f(\phi) = 1 - |a \sin(\phi - b)| \tag{226}$$

seems to be a good fit. We can find a and b by minimizing a linear L_1 fitting which produces the following four functions:

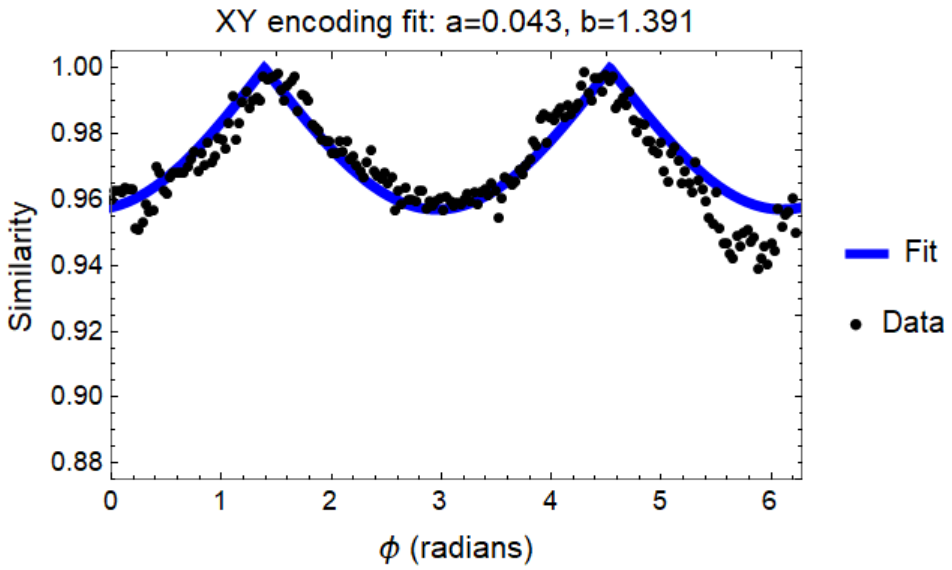

Figure 33: The function $f(\phi) = 1 - |.043 \sin(\phi - 1.391)|$ fitted against the XY experimental data (with NAED).

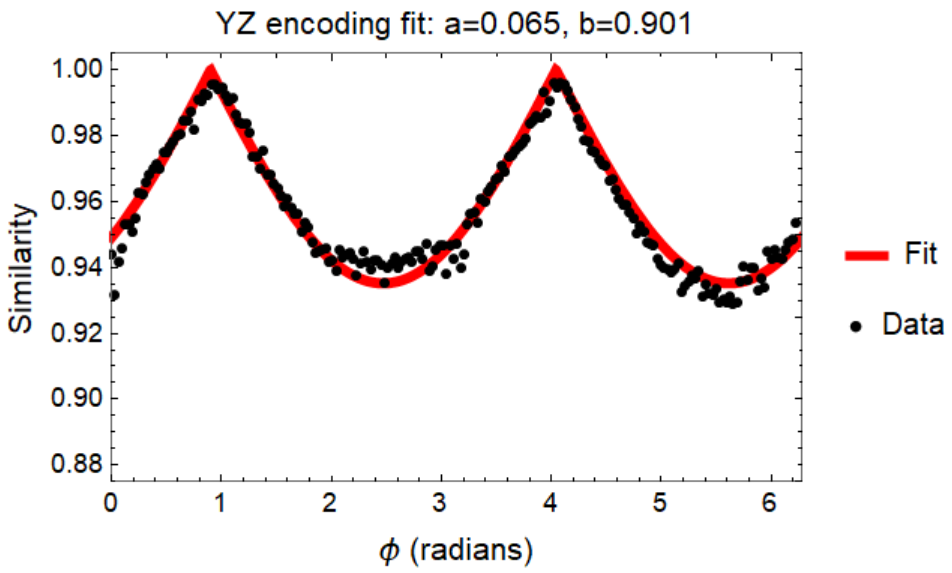

Figure 34: The function $f(\phi) = 1 - |.065 \sin(\phi - 4.042)|$ fitted against the YZ experimental data (with NAED).

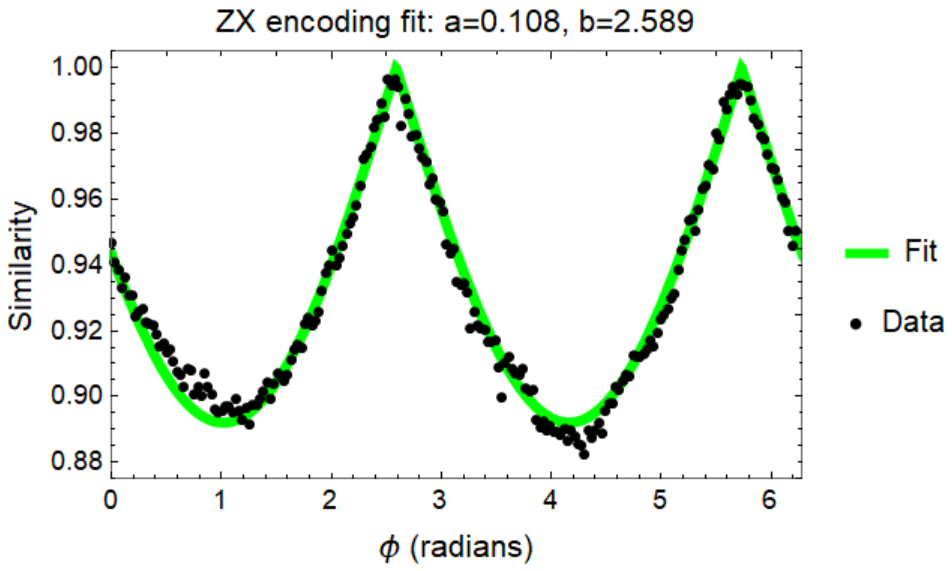

Figure 35: The function $f(\phi) = 1 - |.108 \sin(\phi - 2.589)|$ fitted against the ZX experimental data (with NAED).

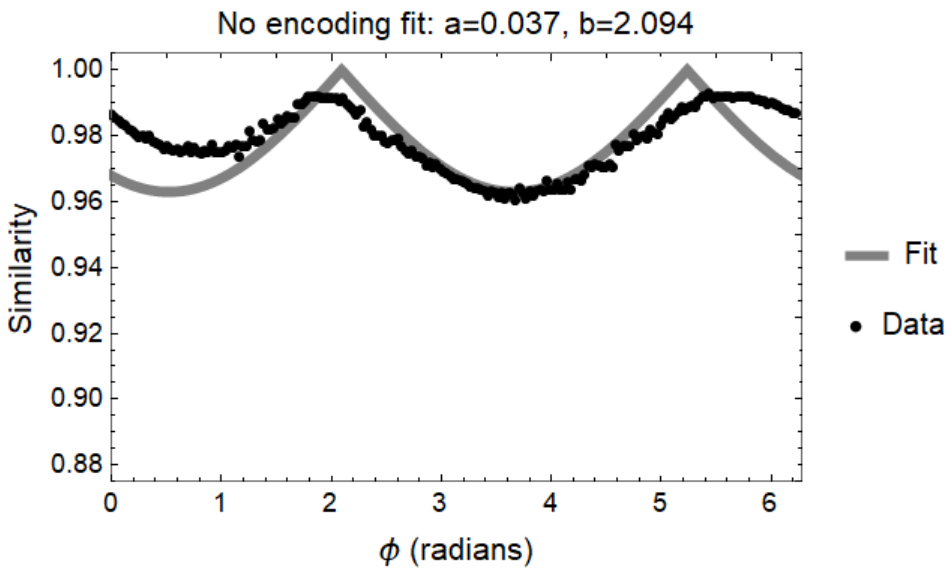

Figure 36: The function $f(\phi) = 1 - |.037 \sin(\phi - 2.094)|$ fitted against the unencoded experimental data.

While the XY, YZ, and ZX fits work relatively well, it seems that the unencoded data may be described by some other function. The average distance between the fitted function and the data is presented below:

Encoding	Average distance ($\times 10^3$)
XY	4.82
YZ	3.20
ZX	3.61
None	6.49

Table 6: The average absolute value between the fitted function and the experimental data.

Obviously, none of the encoded circuits best the unencoded circuit on average. However, it is clear that for certain angles, the encoded circuits with NAED produce better similarities than the unencoded circuits. For example, at 4 radians both the XY and YZ encodings perform better than the unencoded control circuit. This suggests that while NAED is circuit dependent, at some angles there is phase error detection occurring. Further study is needed to determine which angles correspond to which

phase error detections.

4.5 Conclusion

We experimented with a circuit which had phase hard wired into the final measurement probabilities. Although the XY, YZ, and ZX encodings did not perform as well as the bit-flip encoding did with the GHZ circuits in the previous chapter, we still managed to show an increase in the similarities for certain angles. This suggests that NAED can be viable but care must be taken to choose the right encoding for the right circuit. Overall, these experiments showed that NAED can work for more complicated circuits as well as correct phase errors in addition to bit-flip errors.

V. Improving QAOA using NAED

5.1 The quantum approximate optimization algorithm (QAOA)

The quantum approximate optimization algorithm (QAOA) is a noisy intermediate-scale quantum algorithm designed to give approximate solutions to a wide range of combinatorial optimization problems. First introduced in 2014 by Farhi and Goldstone [63], the algorithm takes advantage of both classical and quantum resources to arrive at a desired answer. Although it is unable to solve NP-complete problems (that anyone knows of), it is able to provide answers which approximate optimum solutions. As a relatively simple algorithm, it has been studied a number of times by a large set of researchers. These include many different foundational studies [64, 65, 66], simulations [67, 68, 69, 70], and experiments on physical machines [71, 72]. In this section, we will provide a brief overview of the algorithm as well as the specific problem we will be implementing for our experiments.

5.1.1 Combinatorial optimization problems

Consider the set $S = \{0, 1, ..., 2^n - 1\}$ for some $n \in \mathbb{N}$ and some finite set of functions $C_i : S \to \{0, 1\}$ (for $1 \leq i \leq m$). In general, the goal in combinatorial optimization is to find the maximum of

$$C(s) = \sum_{i=1}^{m} C_i(s) \tag{227}$$

This function is called the cost function and the individual C_i functions are called the individual clauses of the cost function. For example, one might have $n = 3$ and the clauses

$$C_1 = \begin{cases} 1 & \text{if } s_2 = 1 \\ 0 & \text{if } s_2 = 0 \end{cases} \tag{228}$$

$$C_1 = \begin{cases} 1 & \text{if } s_1 s_0 = 1 \\ 0 & \text{if } s_1 s_0 = 0 \end{cases} \tag{229}$$

where the s_i are the binary digits of s ($s = 2^2 s_2 + 2^1 s_1 + 2^0 s_0$). Writing out all possibilities in the following cost function table

Input s (in binary)	$C_1(s)$	$C_2(s)$	$C(s)$
000	0	0	0
001	0	0	0
010	0	0	0
011	0	1	1
100	1	0	1
101	1	0	1
110	1	0	1
111	1	1	2

Table 7: All possible inputs and outputs for the cost function. This function is maximized at $s = 7$.

it is clear that $s = 7$ maximizes $C(s)$.

Of course, any problem with a set number of clauses is ultimately uninteresting. Normally, as the problem size increases the number of possible inputs will increase exponentially while the number of clauses will increase polynomially. One such example is the dominating set problem (DSP). The DSP is an NP-complete problem [73] which asks: what is the minimum number of vertices required to 'dominate' a given graph. We say that a vertex set W dominates a graph if every vertex in the graph is connected to a vertex in W (every vertex is self-connected). Fig. 37 and 38 provide a visual explanation of this concept.

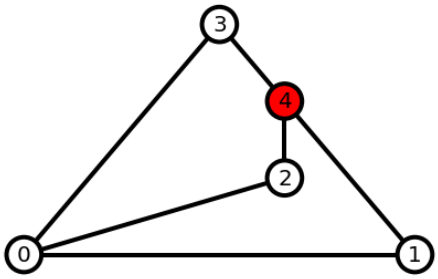

Figure 37: $W = \{4\}$ does not dominate this graph since 0 is not connected to 4.

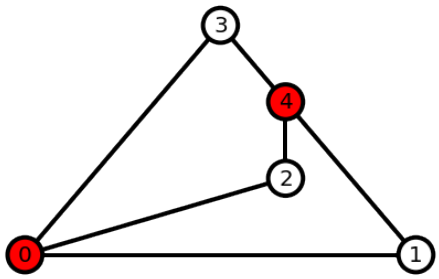

Figure 38: $W = \{0, 4\}$ does dominate this graph since $1, 2, 3$ are connected to 4 and 0 is connected to 0.

For the sake of notation, we will split the clauses associated with the DSP into two categories: $T_i(s)$ and $D_i(s)$. For n nodes, the input space is $s \in \{0, 1, ..., 2^n - 1\}$ and the clauses are defined as

$$T_i(s) = \begin{cases} 1 & \text{if the } i\text{th nodes is connected to some } k\text{th node where } s_k = 1 \\ 0 & \text{if otherwise} \end{cases} \tag{230}$$

$$D_i(s) = \begin{cases} 1 & \text{if } s_i = 0 \\ 0 & \text{if } s_i = 1 \end{cases} \tag{231}$$

From these clauses, we see that the number of inputs is given by 2^n while the number of clauses is given by $2n$.

5.1.2 Implementing QAOA

It is instructive to present QAOA from a purely theoretical perspective. Suppose that we have a combinatorial optimization problem with m clauses $C_k(s)$ on 2^n inputs. The independent variables are a set of p angles $\{\gamma_i : 1 \leq i \leq p\}$ and a set of p angles

$\{\beta_i : 1 \leq i \leq p\}$. However, since increasing p simply refines any answer one might get, we will stick with $p = 1$ for the rest of this chapter. For the sake of notation, denote $\gamma_1 = \gamma$ and $\beta_1 = \beta$. To begin, define

$$U_C(\gamma) = \prod_{k=1}^{m} \exp(-i\gamma C_k) \tag{232}$$

where C_k is an $n \times n$ diagonal matrix where the rth diagonal entry is 1 if $C_k(r) = 1$ and 0 otherwise. With the product over every clause, the matrix $U_C(\gamma)$ becomes a diagonal matrix where the rth diagonal is given by $e^{-i\gamma C(r)}$. For the angle β, define

$$U_B(\beta) = \begin{pmatrix} \cos(\beta) & -i\sin(\beta) \\ -i\sin(\beta) & \cos(\beta) \end{pmatrix}^{\otimes n} \tag{233}$$

With these matrices defined, we can fully describe QAOA. In matrix notation, it is given by

$$|\psi\rangle = U_B(\beta)U_C(\gamma)H^{\otimes n}|0\rangle^{\otimes n} \tag{234}$$

where $(\gamma, \beta) \in [0, 2\pi] \times [0, \pi/2]$. As an aside, note that we have not described how to implement matrices of the form $\exp(-i\gamma C_k)$ as in general it is problem dependent. However, since the general case is not needed for our experiments, we shall describe how to implement our specific problem instance in the next section.

Run the circuit described in Equation 234 for some angles (γ, β) for a 'large' number of shots. At measurement, each qubit corresponds to a bit of a number and each possible measurement result over all n qubits corresponds to a value in $\{0, 1, ..., 2^n - 1\}$. Using these results, calculate

$$f_N(\gamma, \beta) = \frac{1}{N} \sum_{i=0}^{2^n-1} a_i C(i) \tag{235}$$

where N is the total number of shots run and a_i is the total number of times i (in binary) is measured. While this may appear to take an exponential number of operations, the vast majority of a_i will be 0 and as such this quantity can be calculated in polynomial time.

In the limit as the number of shots goes to infinity, this function $f_N(\gamma, \beta)$ becomes

$$\lim_{N\to\infty} f_N(\gamma,\beta) = \lim_{N\to\infty} \sum_{i=0}^{2^n-1} \frac{a_i}{N} C(i)$$

$$= \sum_{i=0}^{2^n-1} \lim_{N\to\infty} \frac{a_i}{N} C(i) = \sum_{i=0}^{2^n-1} P(|i\rangle) C(i) = f_\infty(\gamma,\beta) \tag{236}$$

which is continuous over the space $(\gamma, \beta) \in [0, 2\pi] \times [0, \pi/2]$. The goal of the classical computation portion of QAOA is to find angles (γ, β) which maximize this function. Once such a maximum has been found (or a set of angles that one believes to be the maximum) at some angles (γ, β), then QAOA is run one more time at these (γ, β). After the desired number of shots have been performed, every measured outputs is fed into the original cost function. Whichever measured r produces the largest $C(r)$ is declared to be the s which maximizes $C(s)$ for $s \in \{0, 1, ..., 2^n - 1\}$.

5.1.3 The maximum cut problem (MCP)

Suppose you have some graph for which you split up the vertices into two vertex sets. The maximum cut problem (MCP) asks how many edges is possible between the two vertex sets and the decision problem version is know to be NP-complete [74]. For example, suppose we color some vertices red (to represent being in one of the vertex sets) in the graphs in Fig. 39 and 40:

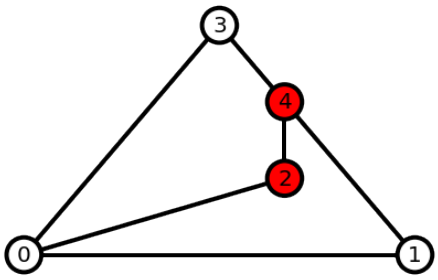

Figure 39: Choosing nodes 2 and 4 results in a score of three: edges $0-1$, $1-4$, and $3-4$.

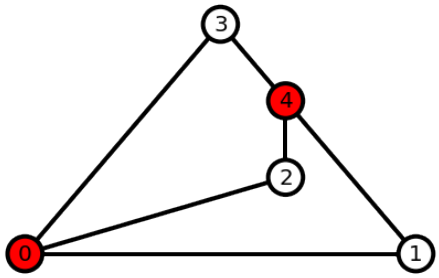

Figure 40: Choosing nodes 0 and 4 results in a score of six as all edges are between vertices of different colors.

The clauses associated with the MCP are very simply: there are exactly as many clauses as edges and they are defined as

$$C_{(u,v)}(s) = \begin{cases} 1 & \text{if } s_u \neq s_v \\ 0 & \text{if } s_u = s_v \end{cases} \tag{237}$$

Basically, each clause is 1 if the two nodes an edge connects are in different vertex sets and 0 otherwise.

Finally, we need to implement the $U_C(\gamma)$ gate associated with the MCP (from equation 232). To this end, we will demonstrate how to implement $\exp(-i\gamma C_{(u,v)})$ and from there the general gate follows. Consider the circuit given by

Figure 41: The physical gates used to encode $\exp(-i\gamma C_{(u,v)})$. Here, the top qubit is qubit u and the bottom qubit is qubit v. The P gate has phase $-\gamma$.

Here, the top qubit represents qubit u, the bottom qubit represents qubit v, and the

phase gate has phase $-\gamma$. In matrix notation, this circuit is given by

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \left[\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & e^{-i\gamma} \end{pmatrix} \right] \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & e^{-i\gamma} & 0 & 0 \\ 0 & 0 & e^{-i\gamma} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \exp\left[-i\gamma \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \right] = \exp(-i\gamma C_{(u,v)}) \quad (238)$$

This analysis can be extended for the general $U_C(\gamma)$ gate. For example, for the graph given in Fig. 42

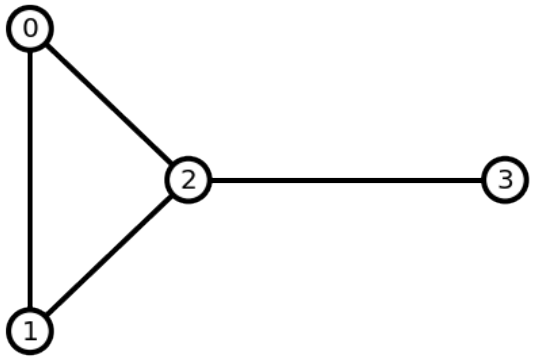

Figure 42: A graph with four nodes.

the $U_C(\gamma)$ gate is given by Fig. 43

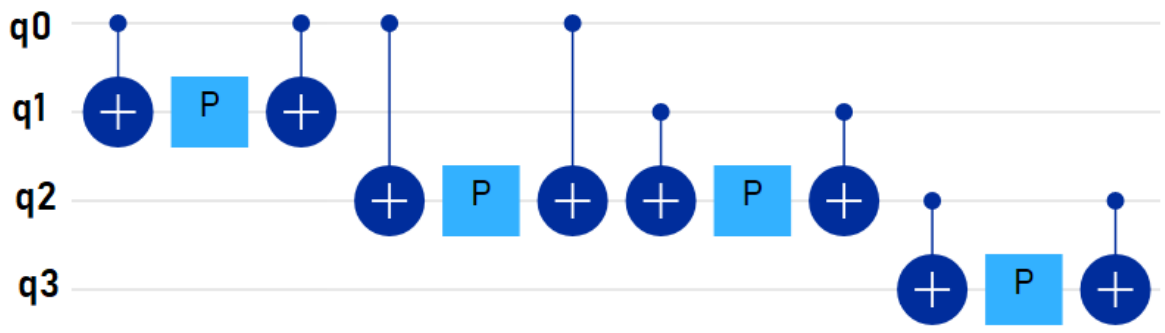

Figure 43: The $U_C(\gamma)$ gates associated with the graph in Fig. 42. The phase gates have phase $-\gamma$. Each qubit q_i corresponds to node i in the graph.

Putting everything together, the QAOA circuit for the graph in Fig. 42 is given by

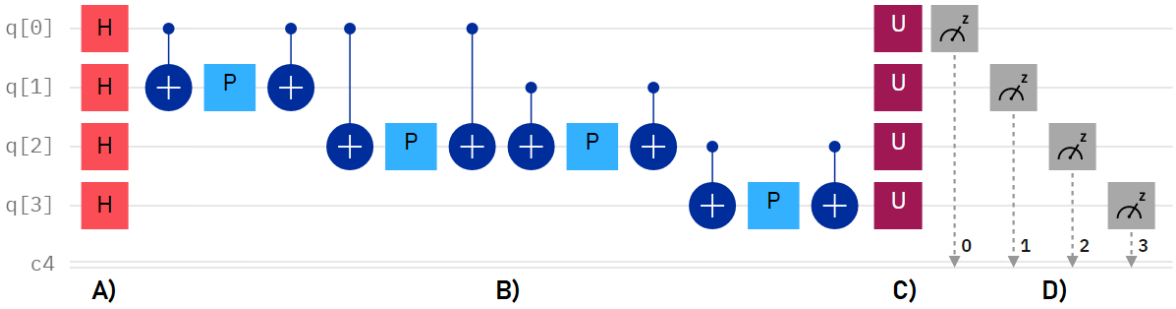

Figure 44: The full QAOA circuit for the graph in Fig. 42. A) is the $H^{\otimes 4}$ gates, B) is the $U_C(\gamma)$ gate, C) is the $U_B(\beta)$ gate where each U gate is given in equation 233, and D) are the measurements.

Here, section A) is the $H^{\otimes}4$ gates, B) is the $U_C(\gamma)$ gate, C) is the $U_B(\beta)$ gate (given in equation 233), and D) are the measurements.

5.1.4 Review of masters thesis results

In this section we will review work previously done testing QAOA with the max cut problem. These experiments were run on the 20 qubit *ibmq_poughkeepsie* [62] machine represented in Fig. 45

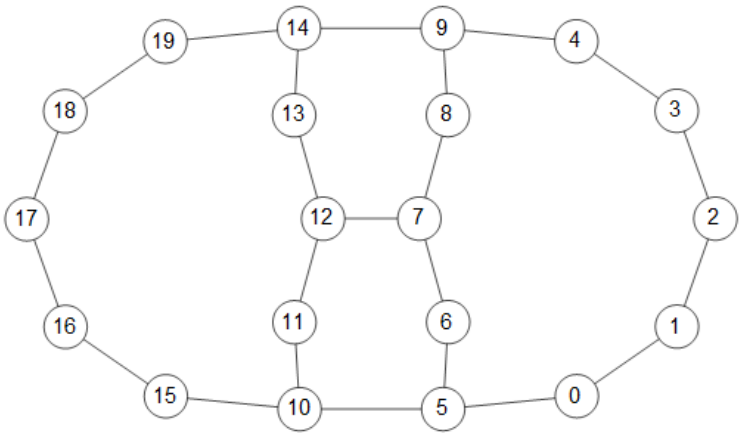

Figure 45: The 20 qubit machine *ibmq_poughkeepsie*. Before being decommissioned in 2020, it had a quantum volume of 8.

Before being decommissioned in 2020, it had a quantum volume of 8 [75] representing a space of three working qubits. 26 graphs were investigated in these experiments: all connected graphs with 5 or less edges and four connected graphs with 6 edges. For example, the 13th graph we tested is presented in Fig. 46. For diagrams of all 26 graphs, see Appendix A.

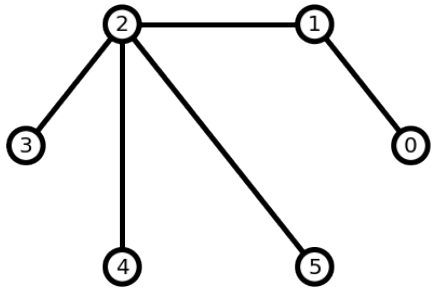

Figure 46: Graph 13

To test QAOA, the input space was subdivided into 121 angles sets

$$(\gamma, \beta) \in W = \left\{ 2\pi \cdot \frac{0}{10}, 2\pi \cdot \frac{1}{10}, ..., 2\pi \cdot \frac{10}{10} \right\} \times \left\{ \frac{\pi}{2} \cdot \frac{0}{10}, \frac{\pi}{2} \cdot \frac{1}{10}, ..., \frac{\pi}{2} \cdot \frac{10}{10} \right\} \quad (239)$$

For each of these elements in W the associated QAOA circuit was run on Poughleepsie for 2^{13} shots at optimization level 3, producing a PDF $P_{Ex}(\gamma, \beta)$. Additionally, this QAOA circuit was also run on the IMBQ simulator producing a PDF $P_{Sim}(\gamma, \beta)$. These PDFs were then compared and a final similarity was found for each graph and pair of angles. After getting a similarity for every pair of angles, the average is taken for a final similarity. These similarities are presented in Fig. 47.

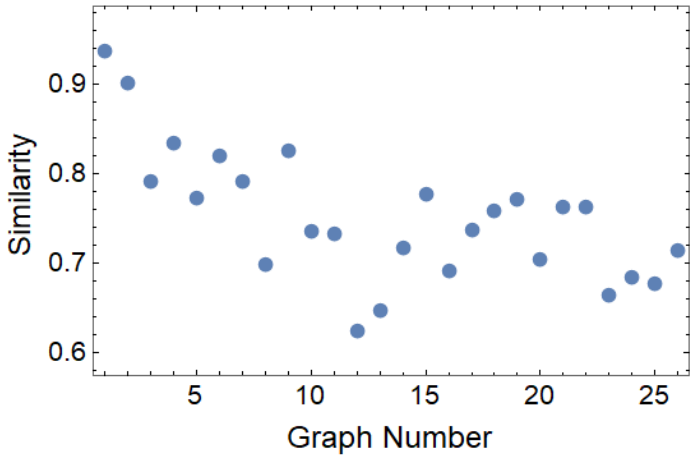

Figure 47: The average similarities for the QAOA circuits run on Poughkeepsie as compared to the QAOA circuits run on the IBMQ simulator. Note that graph number corresponds to the graph numbers presented in Appendix A.

The graph number in the x-axis denotes the graph numbers presented in Appendix A. Although this is not indicative of any particular graph property increasing other than the number of edges, it is clear that as the graph number increases the average similarity $\mu(P_{Ex}, P_{Sim})$ generally decreases.

5.2 Experiments run

The QAOA algorithm is susceptible to both phase and bit-flip errors. As such, it is a useful circuit to fully test NAED. This section will detail the encodings used,

the different experiments ran, as well as some experiments that could not be completed. All experiments were run on *ibmq_montreal*. This computer has 27 physical qubits, a quantum volume of 128 (corresponding to 7 working qubits) and a topology represented in Fig. 48 [62].

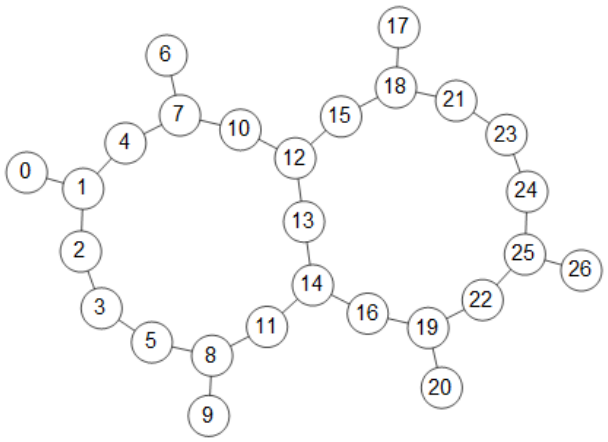

Figure 48: The topology of *ibmq_montreal*. This machine has 27 qubits and a quantum volume of 128.

Like the previous chapter, we will encode QAOA circuits using the XY, YZ, and ZX encodings. The following sections will follow a set pattern: a specific experiment will be described, any interesting notes will be detailed, and the final average similarities will be shown. Overall, a grand total of 150 million shots were run on the quantum computer

5.2.1 Comparing 2019 and 2022 results

The first thing we did was repeat the experiments run on Poughkeepsie on Montreal. With an increase in quantum volume from 8 to 128, we expected to see an overall increase in the average similarity measure of the QAOA circuits. Indeed, this is exactly what we found

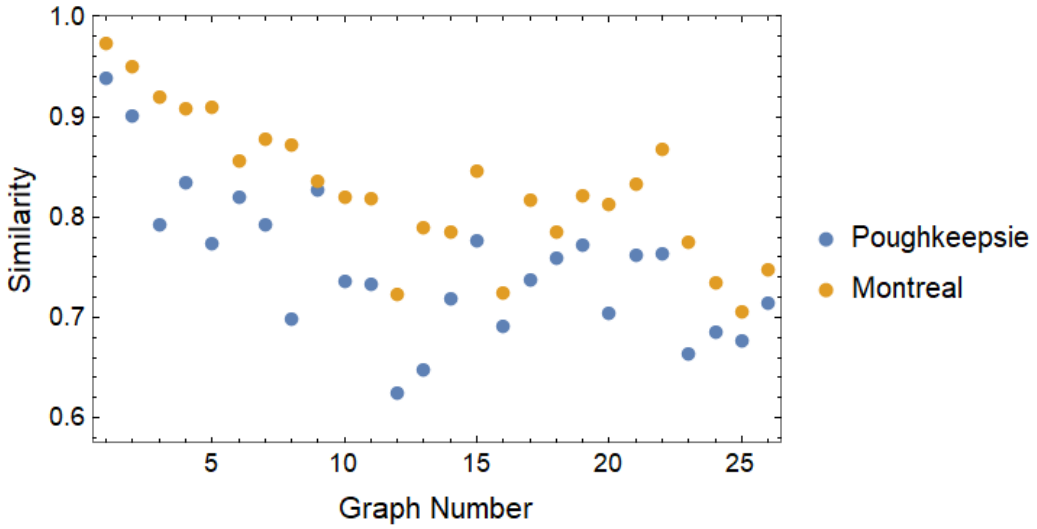

Figure 49: A comparison of average similarities between 2019 Poughkeepsie (quantum volume 8) and 2022 Montreal (quantum volume 128). Montreal outperformed Poughkeepsie on every graph.

Montreal outperformed Poughkeepsie on every graph and the average similarity increase was 0.076. This is evidence that the IBMQ machines have indeed improved over subsequent generations of machines.

5.2.2 Implementing XY, YZ, and ZX encodings

In a previous chapter, we detailed the XY, YZ, and ZX encodings (sections 4.2.1, 4.2.2, and 4.2.3). These encodings were then implemented for every graph and associated QAOA circuit. To do this, the gates in the unencoded circuit were replaced with logical versions from each encoding (Fig. 23 through 25) and barriers were added between each logical gate to prevent catastrophic cancellation. These circuits were then run at each pair of angles in W (equation 239) at optimization level 3. The average similarity between the experimental PDF and the simulated PDF without error detection is shown in Fig. 50.

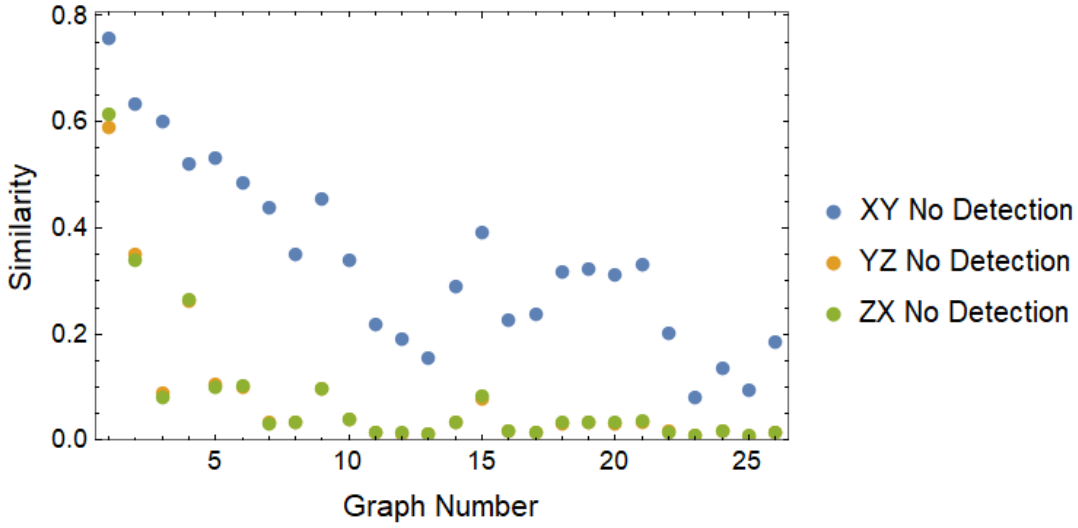

Figure 50: The average similarity for the encoded QAOA circuits without performing error detection.

While it is rather interesting that the XY encoding does the best overall, each one of the three encodings does extremely poorly. However, after performing error detection, the average similarities becomes

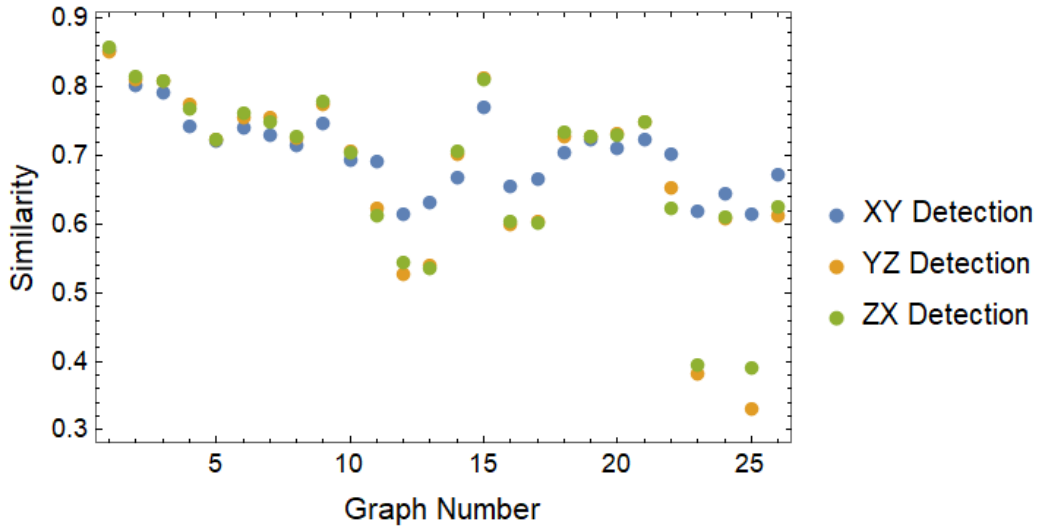

Figure 51: The average similarity for the encoded QAOA circuits after performing error detection.

It seems that the YZ and ZX encodings perform roughly the same. Indeed, the average distance between the similarities is $8.2 \cdot 10^{-3}$. Unfortunately, there is not an immediate pattern apparent that describes when the XY encoding beats the other two encodings. See section 5.3.2 for more analysis of this phenomenon.

5.2.3 No encodings with barriers

Finally, the QAOA circuits were run without encoding but with barriers every layer of the circuit. For example, the circuit in Fig. 52 becomes

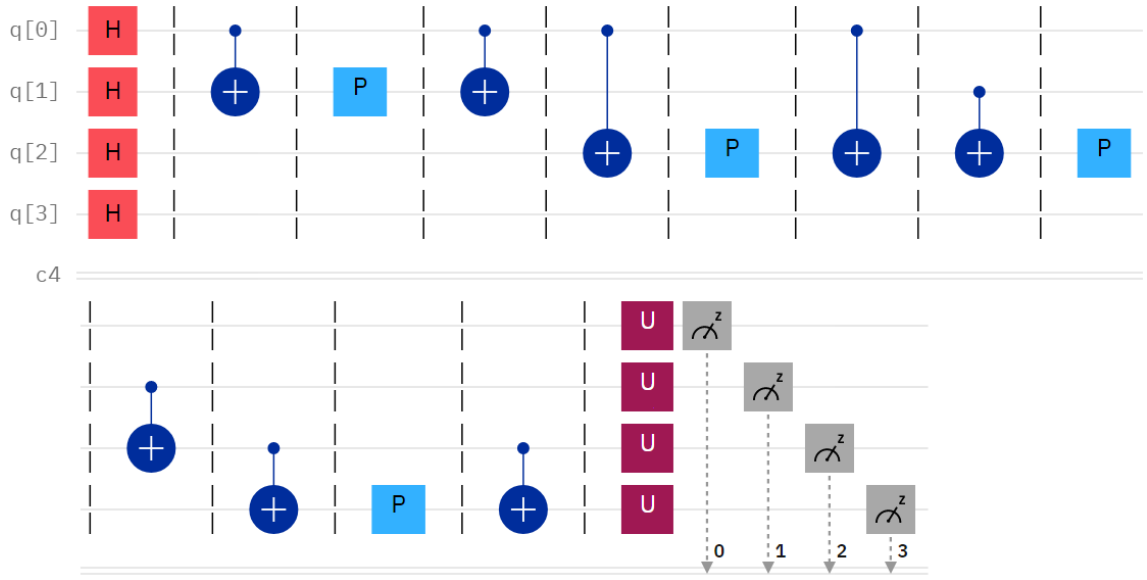

Figure 52: The circuit in Fig. 52 with added barriers in each layer.

The reason for these barriers is to 'level the playing field' between the unencoded circuits and the encoded circuits since the encoded circuits have barriers each layer to prevent catastrophic cancellation. However, this advantage is purely an artifact of the IBMQ compiler and does not represent an inherent advantage that the unencoded circuit enjoys over an encoded circuit. We believe that for this reason this is a reasonable experiment to run to get a better understanding of the differences between the encoded and unencoded circuits. Running these QAOA circuits at optimization

level 3 an averaging over the angle pairs produces Fig. 53

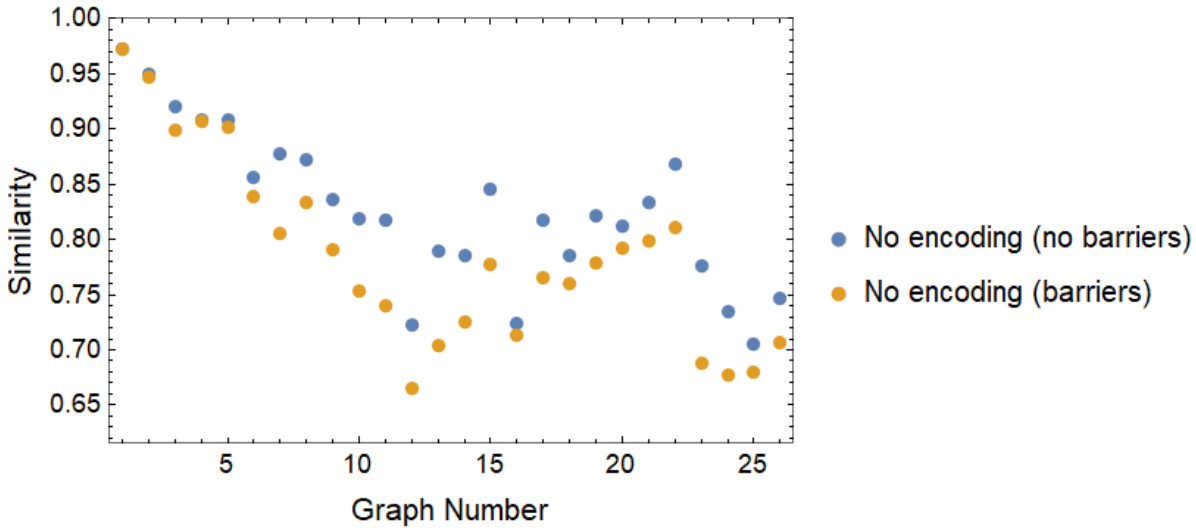

Figure 53: The unencoded QAOA circuits, both with and without barriers every layer.

Every QAOA circuit saw a decrease in the similarity, with an average decrease of 0.041. This is to be expected as adding barriers at every layer forces a less efficient optimization during the compilation of the circuit.

5.2.4 Failed experiments

Unfortunately, there were several experiments which were planned but could not be fully executed due to compiler issues. The first such experiment was attempting to simplify QAOA circuits encoded with the XY encoding. As an example, consider the QAOA circuit associated with the K_2 graph

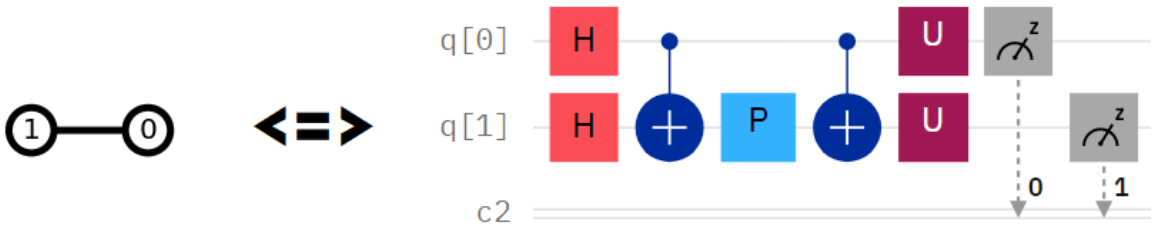

Figure 54: The QAOA circuit associated with the K_2 graph.

Encoding the circuit using the XY encoding transforms this into the circuit in Fig. 55

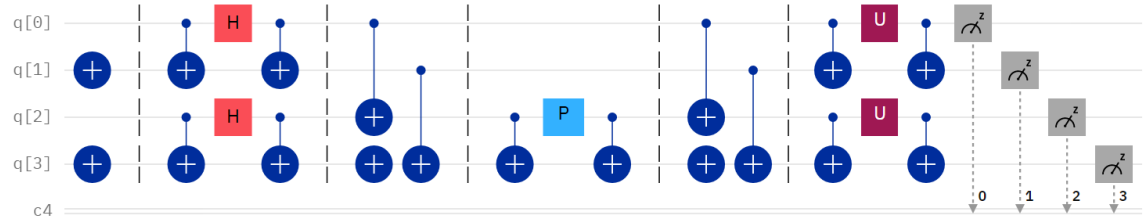

Figure 55: The QAOA circuit from Fig. 54 with the XY encoding.

The idea to simplify the circuit would proceed thusly: remove the barriers and combine any sets of gates which simplify to the identity gate. After such a simplification, the circuit became

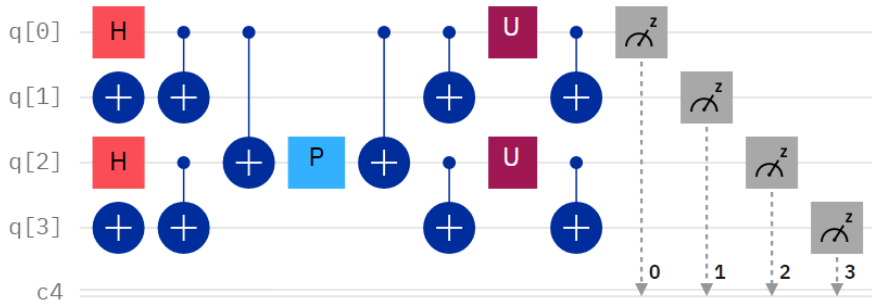

Figure 56: The QAOA circuit from Fig. 55 simplified.

This would obviously greatly reduce the number of overall gates. Just in this example, the C_x count went from 14 C_x gates to 8 C_x gates.

Unfortunately we were not able to run these circuits for all the graphs. Optimization level 3 represents the most intensive optimization that IBM offers for quantum circuits and in order to prevent the wasting of computing resources, IBM puts a hard cap on the number of optimizations allowed by the compiler. Graphs with 4 nodes (8 nodes when encoded) began to hit this hard cap and could not be implemented by the compiler. It seems that removing barriers and simplifying creates a large additional overhead that would otherwise not appear. Overall, we were only able to actually experiment on 5 of the 26 total graphs.

There were some additional graphs we wished to add to the experiments described above. These graphs included various graphs with 8, 10 and 12 node graphs. Unfortunately, we encountered the same problem as the simplified XY code. For both the unencoded QAOA circuits and the encoded QAOA circuits (with barriers) we hit the hard limit on the compiler and were unable to implement the circuits. It would seem that any further testing has to take place with at least optimization level 2 as 7-8 qubit QAOA circuits seems to be about the compiler limit.

5.3 Results and analysis

5.3.1 Analysing by angle

Above, we presented results after averaging over all 121 angle pairs in the input space. It is useful though to analyse the data for every individual input. Below in Fig. 57 and Fig. 62 we provide the results for graph 9 by angle. The left figure is a comparison between the XY, YZ, and ZX encodings, where each square represents which encoding had the highest similarity measure. The right figure is a comparison between the XY, YZ, ZX, and no encoding (with added barriers). Additionally, recall that the goal of QAOA is to maximize the function in equation 236. The red dots on each plot represent where this function is maximized for this particular graph. For

all results by graph, see Appendix B.

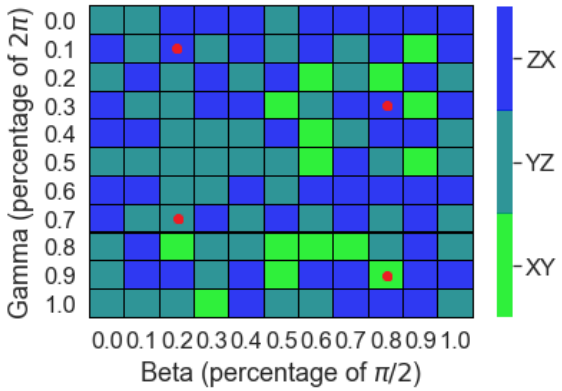

Figure 57: Graph 9 results (just encodings). The red dot represents an approximate maximum of equation 236.

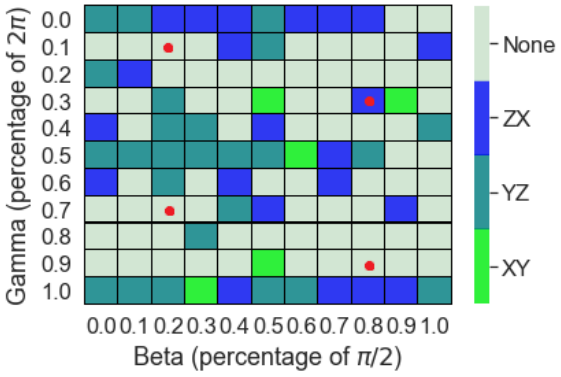

Figure 58: Graph 9 results (with unencoded circuits). The red dot represents an approximate maximum of equation 236.

Without the addition of the unencoded circuits 22 of the maximums occur where XY performs the best, 18 occur where YZ performs the best, and 24 occur where ZX performs the best. Including the unencoded circuits produces a dramatically different result: the XY covers a single maximum, the YZ covers 3 maximums, the ZX covers 2 maximums, while the circuits without encodings performed best at 58 maximums. However, we caution that this is not exhaustive proof that no encodings should ever be used. Indeed, 22 out of the 26 graphs had the peak lie in domain corresponding to $(\gamma, \beta) = (0.62, 0.31)$. It is reasonable to suspect that this is simply an artifact of the small graphs we have used. If not, then solving the max cut problem would become possible rather than just approximating a solution. Simply start the search near $(0.62, 0.31)$ and you would be guaranteed to maximize equation 236. Unfortunately, most believe that this is not possible [76]. Although it is not known whether quantum computers can solve NP-complete problems, the general consensus is that they probably cannot. Of course, the lack of proofs one way or the other does

not deter some and this topic is an active area of research [77].

5.3.2 XY versus YZ/ZX encodings

After discussing in the last section why one pattern can probably be ignored for these small graphs, we will now present a different pattern that we hope might continue to larger graphs. Consider the results and associated graphs below:

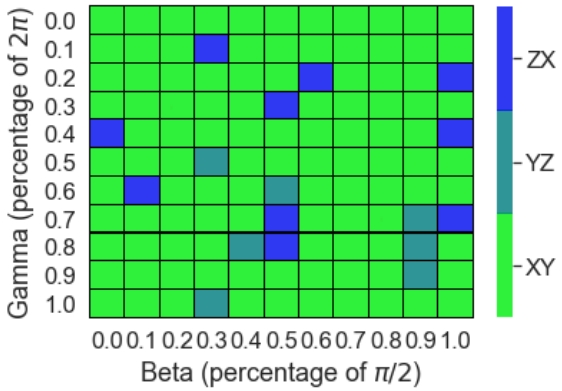

Figure 59: For graph 21 XY beats out YZ and ZX.

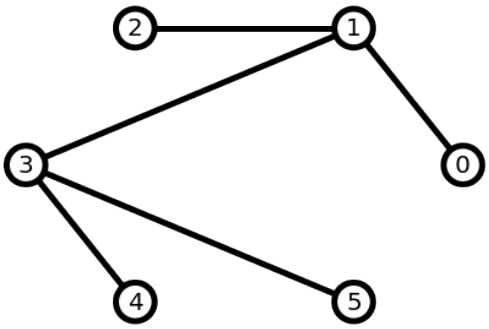

Figure 60: Graph 21 has no cycles.

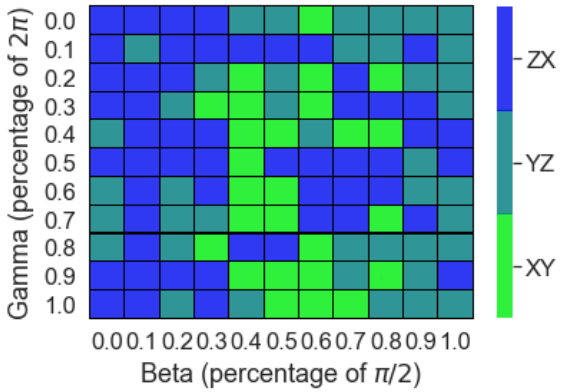

Figure 61: For graph 22 XY loses to YZ and ZX.

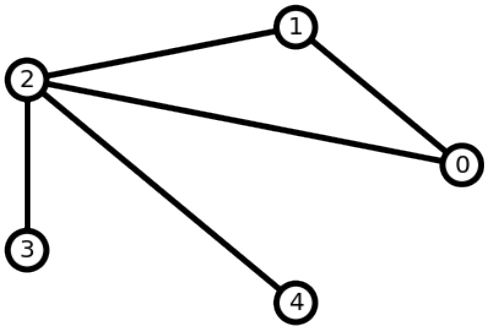

Figure 62: Graph 22 has one cycle.

From a glance, it is obvious that for graph 21, the XY encoding greatly outperforms

the YZ and ZX encodings. In the same way, for graph 22 the YZ and ZX encodings outperform the XY encoding. A quick count confirms this: in graph 21 the XY encoding performs the best for 104 of the inputs (out of 121), for graph 22 it performs the best for 26 of the inputs. We see this type of pattern repeated throughout the experiments where highly connected graphs see greater success with YZ and ZX encodings while tree graphs see greater success with XY encodings.

To formalize this notion count the number of K_3 subgraphs each graph has (see Fig. 65). Next, compare the number of input angles (γ, β) for which XY has the highest similarity versus the number of inputs for which either of YZ or ZX have the highest similarity. This produces Fig. 63.

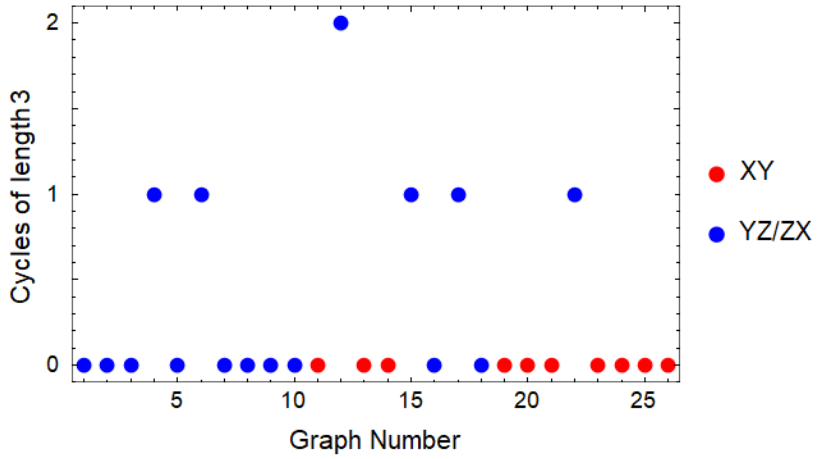

Figure 63: The y-axis is the number of K_3 subgraphs in each graph (the x-axis). The color is red if XY has the most input angles (γ, β) with the highest similarity and blue otherwise.

The XY encoding only performed best on the majority of inputs for graphs without any K_3 subgraphs. The YZ or ZX encodings performed best on the smallest graphs as well as graphs with K_3 subgraphs.

What could be the cause of this behavior? Be tentatively believe it might be due to the encoded states resulting from 3 cycles versus the encoded states resulting from

tree graphs. Consider the following graphs:

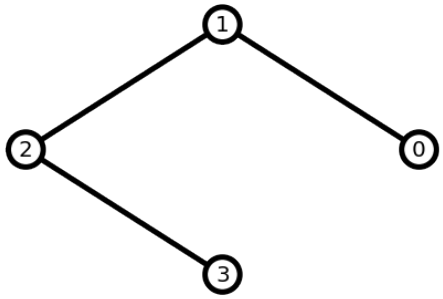

Figure 64: Graph 3, a graph with no 3 cycles.

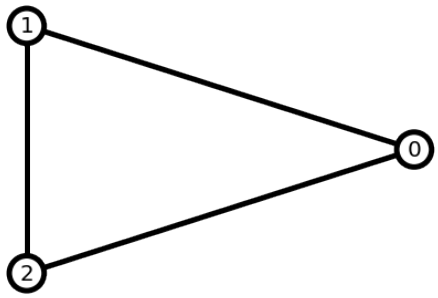

Figure 65: Graph 4, the K_3 graph with a single cycle of length 3

In QAOA, these graphs implement the $U_C(\gamma)$ matrix (equation 238) as follows:

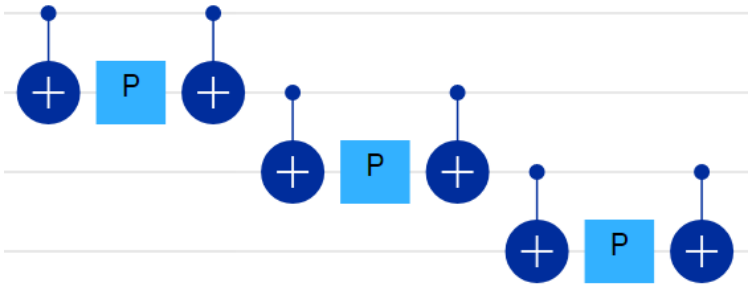

Figure 66: The QAOA implementation of the $U_C(\gamma)$ matrix from equation 238 for graph 3.

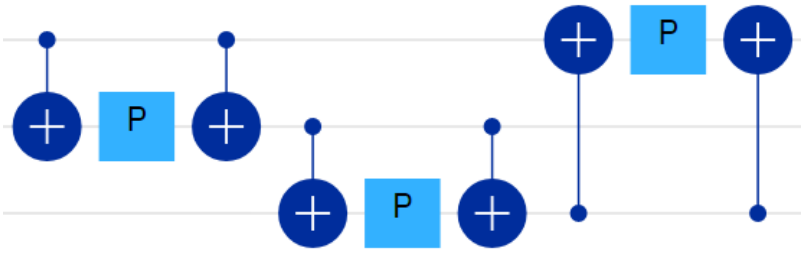

Figure 67: The QAOA implementation of the $U_C(\gamma)$ matrix from equation 238 for graph 4.

The implementation for graph 3 is a cascading chain of gates while the implementation for graph 4 loops around back on itself. This pattern repeats for any implementation of any graph, and it is this feature which we tentatively believe leads to the results seen in Fig. 63.

Unfortunately, this behavior dissapears when using a different perspective. If instead of counting how many input angles each encoding excels at we instead average the similarities over all inputs, we get Fig. 68 below:

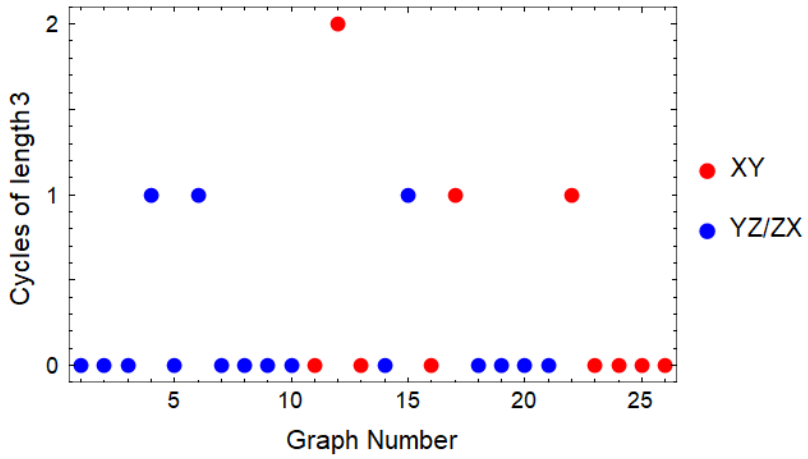

Figure 68: The y-axis is the number of K_3 subgraphs in each graph (the x-axis). The color is red if the XY encoding has a higher average similarity over the YZ and ZX encodings (blue otherwise).

With this, we cannot say what pattern (if any) in this data may or may not continue to larger graphs. To investigate this, one would need to experiment on graphs with a much higher number of K_3 subgraphs as well as many more tree graphs. At this point, we are unable to continue these experiments due to the limitations of the IBMQ compiler at optimization level 3.

5.4 Combing XY encoding with unencoded results

So far, the unencoded circuit has bested the encoded circuits for almost every graph as shown in Fig. 69.

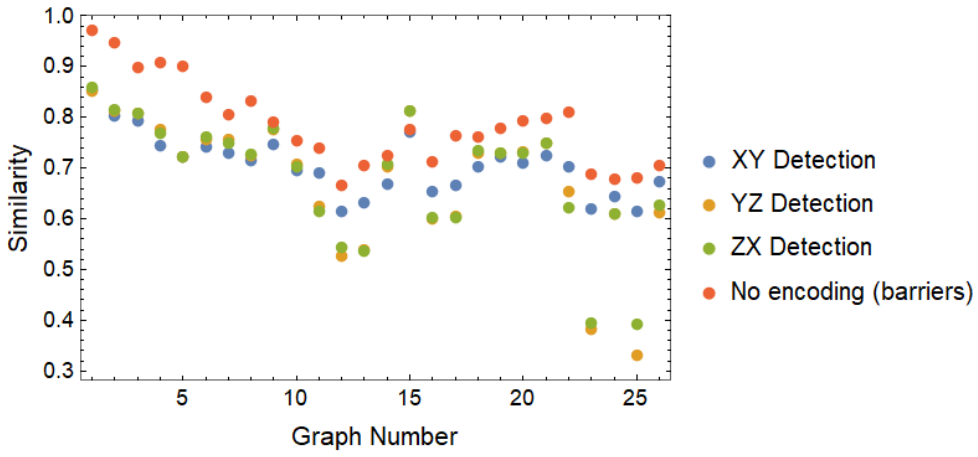

Figure 69: A comparison of the average similarities between the XY, YZ, ZX encodings and no encoding (with barriers). No encoding is greater for every graph except graph 15.

Ignoring graph 15, the average increase from the maximum average similarity of the encoded circuits to the unencoded circuits is 0.071. However, it is possible to increase the average similarity using an encoded circuit if we only take specific angles. Suppose for each angle input we do the following: check which region (γ, β) lies in Fig. 70 below

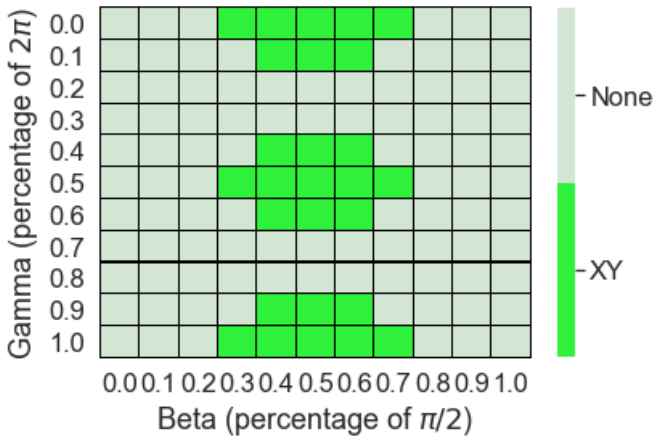

Figure 70: When combining the XY and unencoded circuits, if an input (γ, β) is in the green regions then use the XY encoding. If not, then use no encoding.

If a particular (γ, β) lies in any of the XY regions above, then use the XY encoding for the QAOA circuit. If not, then do not encode the circuit. Afterwards, perform NAED on any results from encoded circuits and combine the results of the encoded and unencoded circuits. Doing this for every graph and computing the average similarity gives us Fig. 71

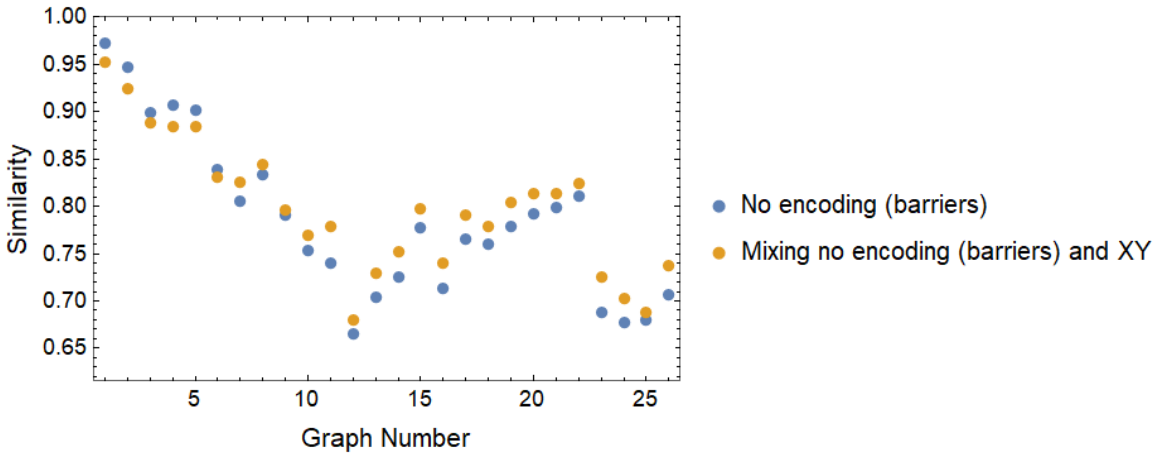

Figure 71: The average similarity for the combined XY and unencoded circuits. After graph 7, the combination performs better than the unencoded circuit on its own.

After graph 7, the combination of the XY and unencoded circuits outperforms the unencoded circuit on its own. The average increase in similarity is 0.021.

5.5 Conclusions

Throughout these QAOA experiments we have sent over 150 million shots to the IBMQ quantum computers, including 6 different experiments over 2 different machines. First, it is plain to see that the quantum computers have indeed gotten better with upgrades. The initial experiments were run on *ibmq_poughkeepsie* (quantum volume 8) while the later experiments were run on *ibmq_montreal* (quantum volume 128). This sizable increase in quantum volume only increased the similarity on average by 0.076. Next, we attempted to increase the average similarities using the XY, YZ and ZX encodings. There might be some behavior regulating which of these encodings is most desirable to use for a particular graph but further testing on much larger graphs is needed to confirm this. Overall, the encoded circuits did not best the unencoded circuit. This held true even if barriers were added to the unencoded circuit to level the playing field in the compiler. However, we were able to get an improvement through a combination of the XY encoding and the unencoded circuits. Through a clever selection based on input angle, we were able to increase the average similarity by 0.021 for all but the seven smallest graphs. This shows that while angle dependent, NAED can still provide a boost over circuits run without error detection.

VI. Overall results, conclusions, and future work

We began this document off by defining NAED and investigating some interesting mathematics associated with this error detection scheme. We showed that in the strictest sense, it is impossible to encode two physical qubits to detect any possible error and that all flawless encodings must necessarily use three physical qubits (and provided an example of such an encoding). However, if one relaxes assumptions slightly, then it is possible to encode two physical qubits such that the resulting encoding can detect all possible errors except for a set of measure zero. Unfortunately, any such encoding requires at least two or more C_x gates. In fact, any near-flawless encoding regardless of the number of physical qubits requires at least two C_x for implementation.

Of course, many interesting mathematical problems still exist. For example, do there exist "flawless" logical gates. These logical gates would have the property that for any error that might occur in the implementation of the logical gate, the output state is sent to the error set instead of the code set. Realistically, it seems difficult to imagine that such a set of gates exists but this could be due to the small sizes of the encodings investigated in this document. If such a set of logical gates does exist it is likely that the size of the encoding is large, perhaps on the order of dozens of physical qubits.

The first main experiment we ran was NAED on GHZ circuits. We described generalized encodings and logical gates, and used these encodings to run five GHZ circuits with four different sized encodings (as well as an unencoded control). Overall, over 45 million shots were sent to the quantum computers. We found that NAED could greatly increase the similarities of the GHZ circuit, with an improvement for all encodings except the case where one logical qubit was encoded by five physical qubits. These results reassured us that NAED could be a viable method for improv-

ing quantum circuits. In future experiments, it might be interesting to rerun this experiment using a different encoding. Obviously, the GHZ states do not depend on phase, so we would hypothesize that phase detection encodings would perform worse overall.

Since phase is an integral part of quantum computing, it was necessary to test NAED on circuits for which phase played an important role. Thus, we developed a toy circuit with output probabilities directly dependent on an interior phase gate. We also developed XY, YZ, and ZX encodings which provided two advantages: they were simple to implement requiring no C_x gates and the errors that each encoding could not detect were easy to describe. We ran our circuit for over 130 million shots using the XY, YZ, and ZX encodings as well as an unencoded control circuit. After performing NAED, the encoded circuits each beat the unencoded circuits at certain angles of the input space. Although less convincing than the GHZ results, these results showed that NAED could be used to improve circuits with phase dependence. Care should be taken however as it seems that these encodings are highly circuit dependent. If one wished, one could further explore these encodings and implement arbitrary $U \otimes U$ encodings instead of $I \otimes I$ for the XY, $H \otimes H$ for the YZ, and $S \otimes S$ for the ZX. These arbitrary encodings admit three degrees of freedom and would presumably work best for specific angles depending on the encoding. Ideally, one would eventually be able to infer at exactly which angles based off of the initial matrix U chosen.

The final experiment we performed was to improve the results from my masters thesis. QAOA produces highly entangled circuits for solving combinatorial optimization problems and originally we approximated solutions to the MCP for 26 graphs using QAOA implemented on *ibmq_poughkeepsie*. Continuing these experiments, we ran the same circuits using the XY, YZ, and ZX encodings as well as two versions of the unencoded circuit on the *ibmq_montreal* machine. All together, over 150 mil-

lion shots were submitted in total. The XY encoding and YZ/ZX encodings showed some interesting graph dependent results: the XY appears to perform better for tree graphs while the YZ/ZX encodings appear to perform better for graphs with cycles. Additionally, we were able to combine the XY encoding and unencoded circuit in a particular way as to improve the overall similarity. While the results were not as encouraging as the previous experiments, we were still able to use NAED to increase the performance of the quantum computers. In future experiments, it would be interesting to expand QAOA to larger and more complicated graphs. Of course, as the graphs get larger the compiler begins to run into compilation issues, so it is likely that these experiments would have to be ran at optimization level 2 or even 1.

Overall, we performed three main experiments while using the quantum computer over 325 million times. Each experiment showed that NAED could be utilized in a way that improved the overall performance of the quantum computer. However, the second two experiments also showed that choosing the correct encoding is generally circuit dependent. While interesting, more work is required to deduce how to include these dependencies in future encodings.

VII. Appendix

7.1 Appendix A: Graph topologies

Figure 72: Graph 1

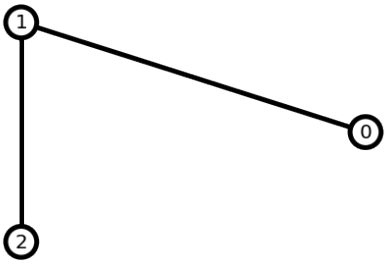

Figure 73: Graph 2

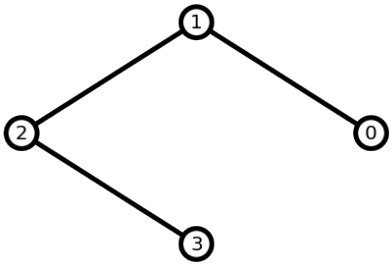

Figure 74: Graph 3

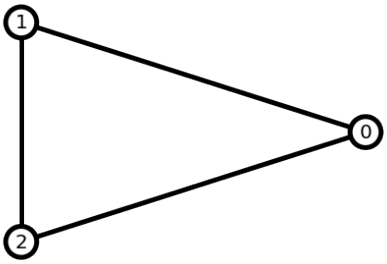

Figure 75: Graph 4

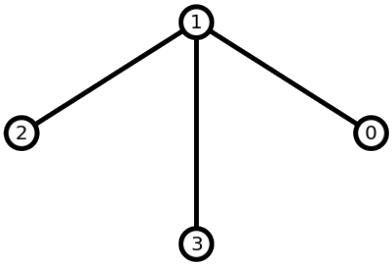

Figure 76: Graph 5

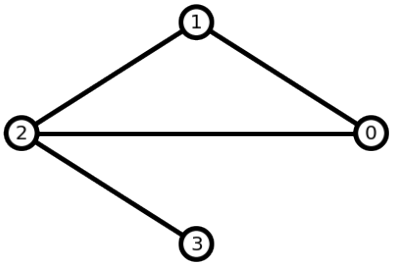

Figure 77: Graph 6

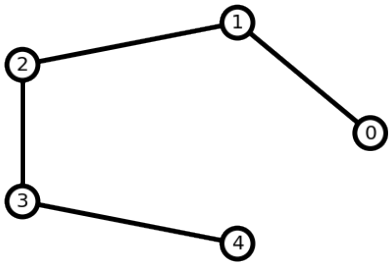

Figure 78: Graph 7

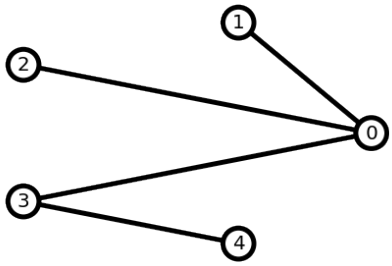

Figure 79: Graph 8

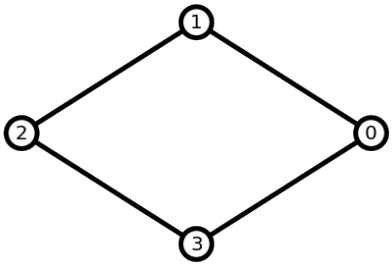

Figure 80: Graph 9

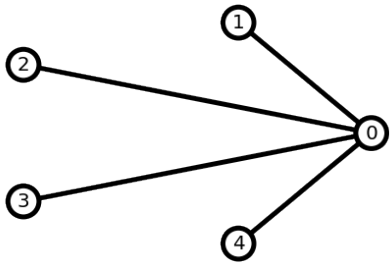

Figure 81: Graph 10

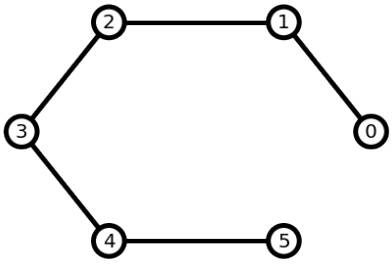

Figure 82: Graph 11

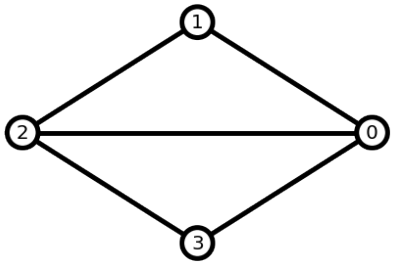

Figure 83: Graph 12

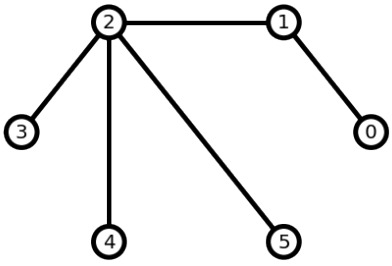

Figure 84: Graph 13

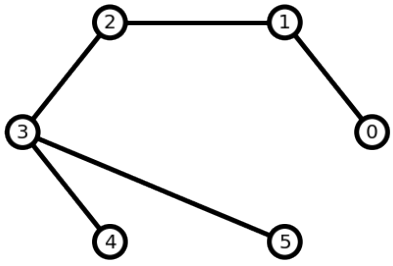

Figure 85: Graph 14

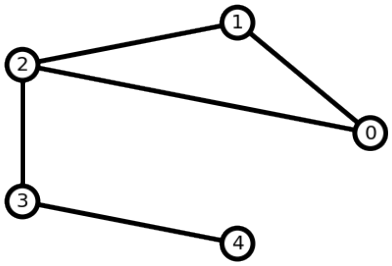

Figure 86: Graph 15

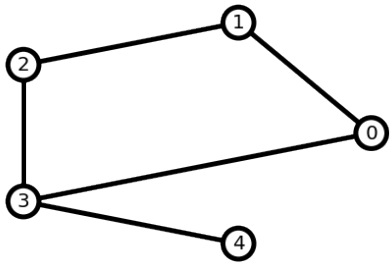

Figure 87: Graph 16

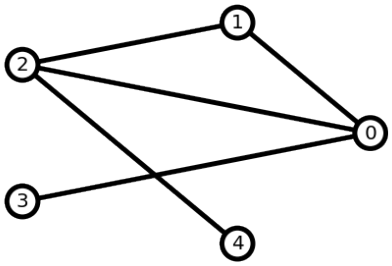

Figure 88: Graph 17

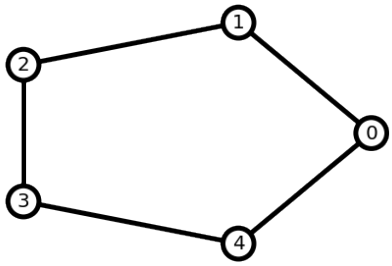

Figure 89: Graph 18

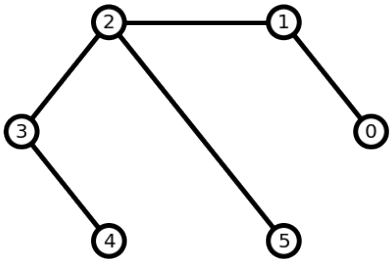

Figure 90: Graph 19

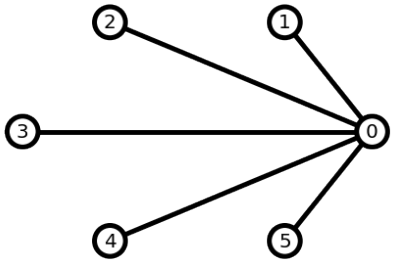

Figure 91: Graph 20

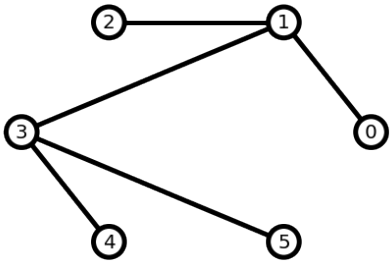

Figure 92: Graph 21

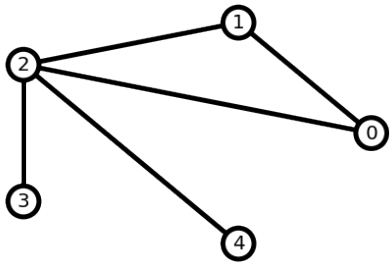

Figure 93: Graph 22

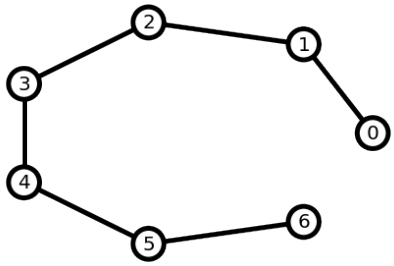

Figure 94: Graph 23

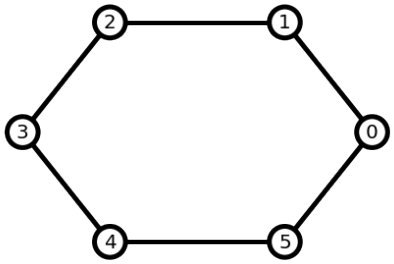

Figure 95: Graph 24

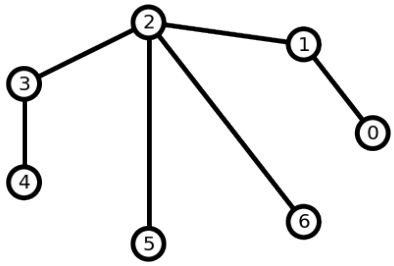

Figure 96: Graph 25

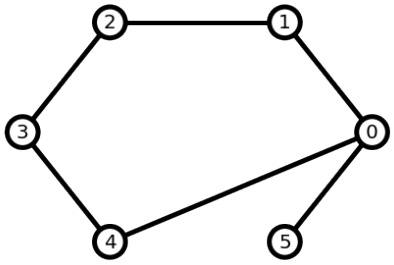

Figure 97: Graph 26

7.2 Appendix B: QAOA results by angle

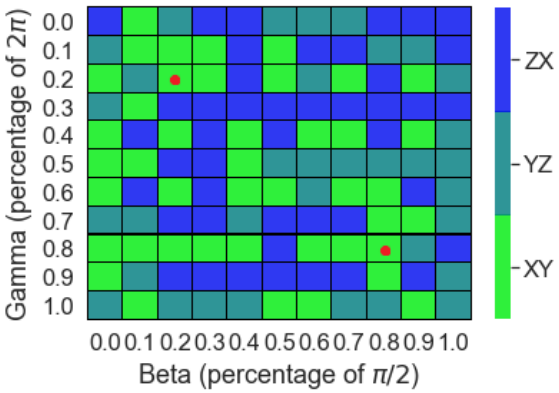

Figure 98: Graph 1 results (just encodings). The red dot represents an approximate maximum of equation 236.

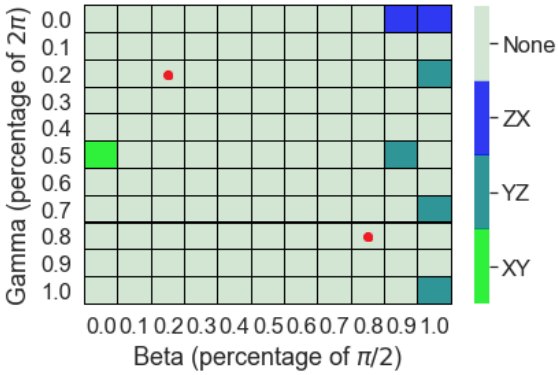

Figure 99: Graph 1 results (with unencoded circuits). The red dot represents an approximate maximum of equation 236.

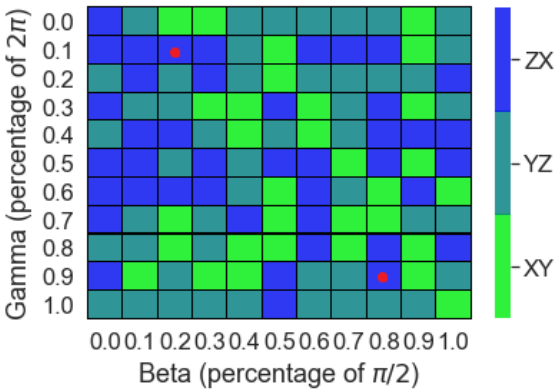

Figure 100: Graph 2 results (just encodings). The red dot represents an approximate maximum of equation 236.

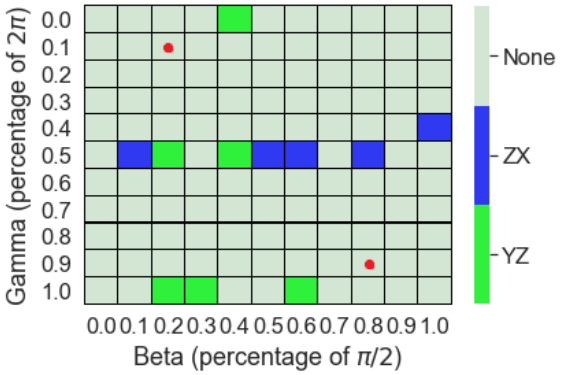

Figure 101: Graph 2 results (with unencoded circuits). The red dot represents an approximate maximum of equation 236.

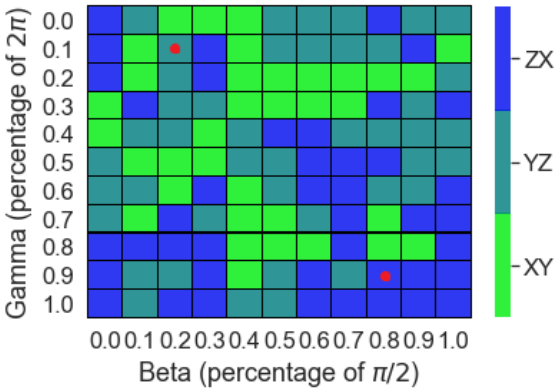

Figure 102: Graph 3 results (just encodings). The red dot represents an approximate maximum of equation 236.

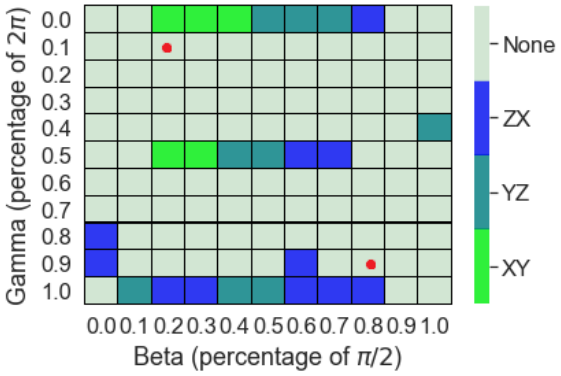

Figure 103: Graph 3 results (with unencoded circuits). The red dot represents an approximate maximum of equation 236.

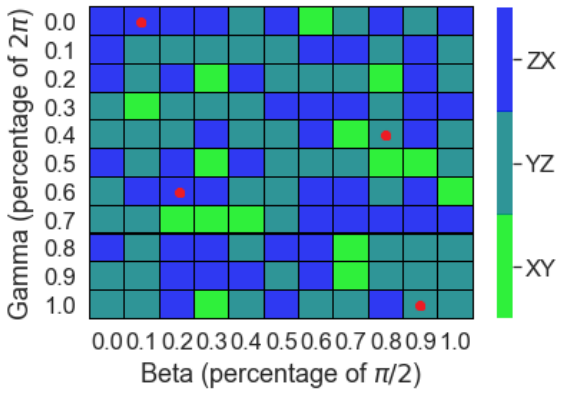

Figure 104: Graph 4 results (just encodings). The red dot represents an approximate maximum of equation 236.

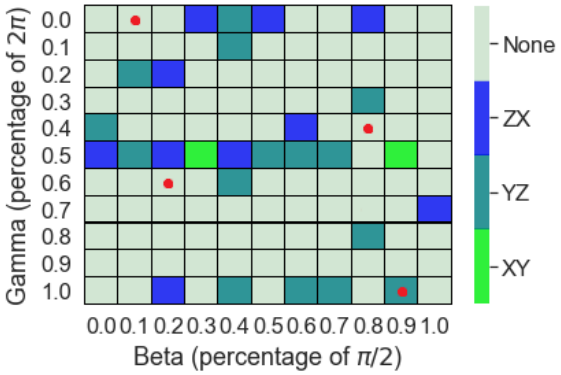

Figure 105: Graph 4 results (with unencoded circuits). The red dot represents an approximate maximum of equation 236.

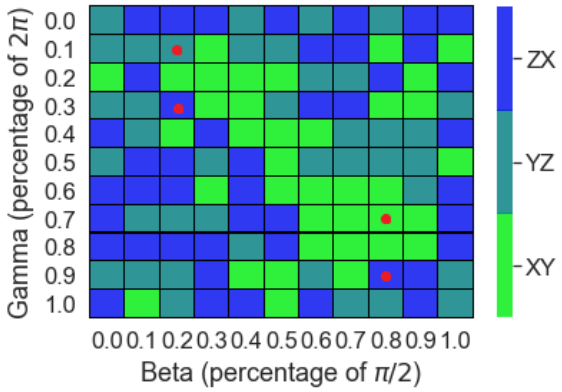

Figure 106: Graph 5 results (just encodings). The red dot represents an approximate maximum of equation 236.

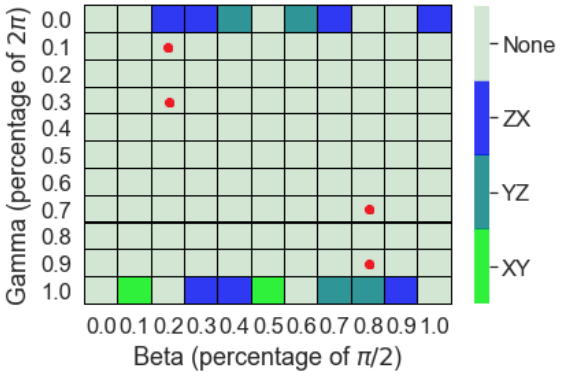

Figure 107: Graph 5 results (with unencoded circuits). The red dot represents an approximate maximum of equation 236.

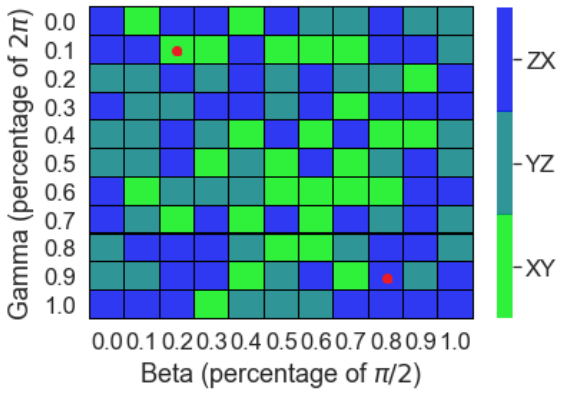

Figure 108: Graph 6 results (just encodings). The red dot represents an approximate maximum of equation 236.

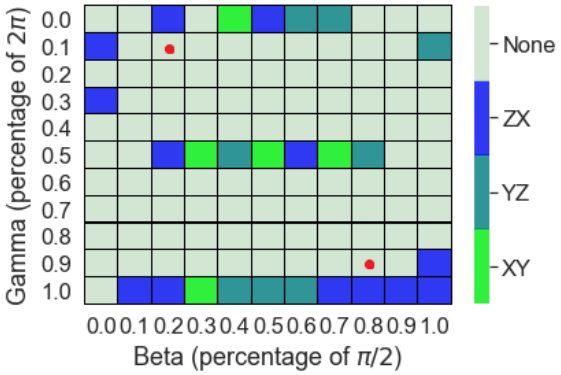

Figure 109: Graph 6 results (with unencoded circuits). The red dot represents an approximate maximum of equation 236.

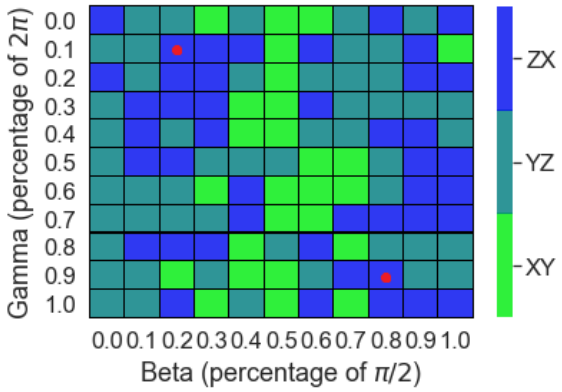

Figure 110: Graph 7 results (just encodings). The red dot represents an approximate maximum of equation 236.

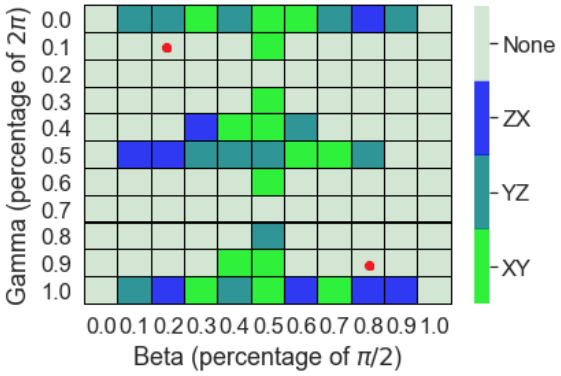

Figure 111: Graph 7 results (with unencoded circuits). The red dot represents an approximate maximum of equation 236.

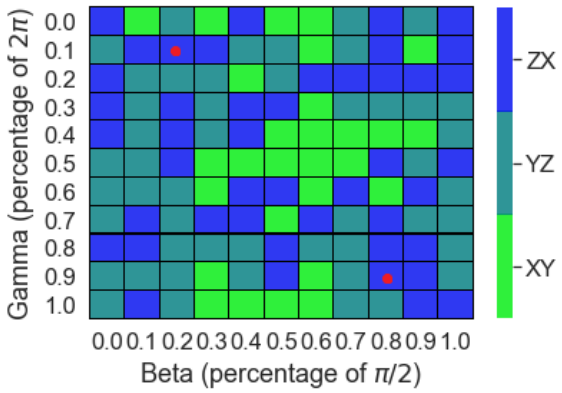

Figure 112: Graph 8 results (just encodings). The red dot represents an approximate maximum of equation 236.

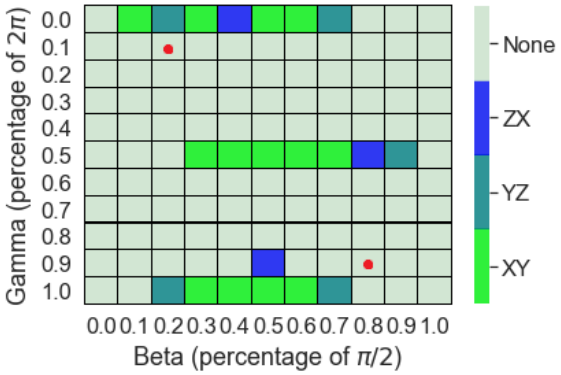

Figure 113: Graph 8 results (with unencoded circuits). The red dot represents an approximate maximum of equation 236.

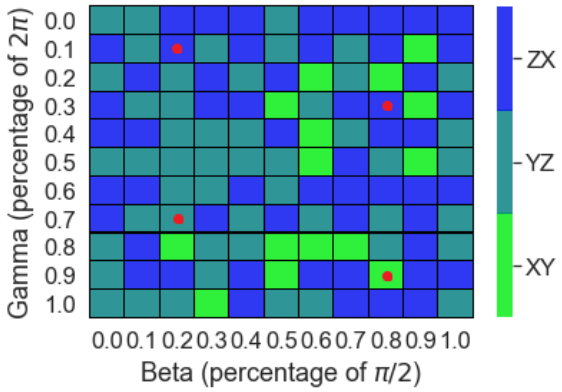

Figure 114: Graph 9 results (just encodings). The red dot represents an approximate maximum of equation 236.

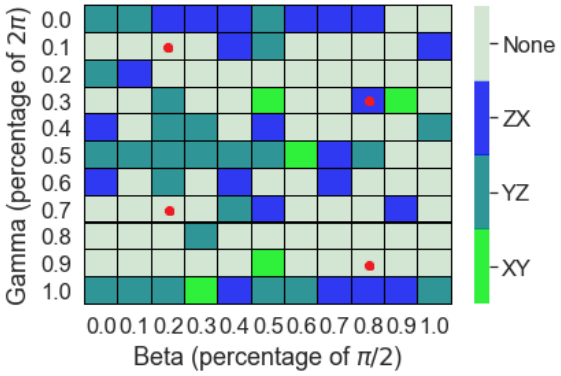

Figure 115: Graph 9 results (with unencoded circuits). The red dot represents an approximate maximum of equation 236.

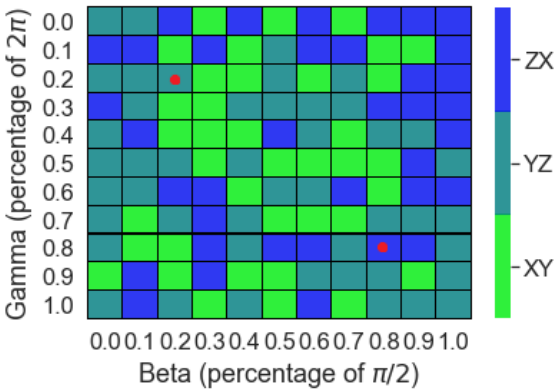

Figure 116: Graph 10 results (just encodings). The red dot represents an approximate maximum of equation 236.

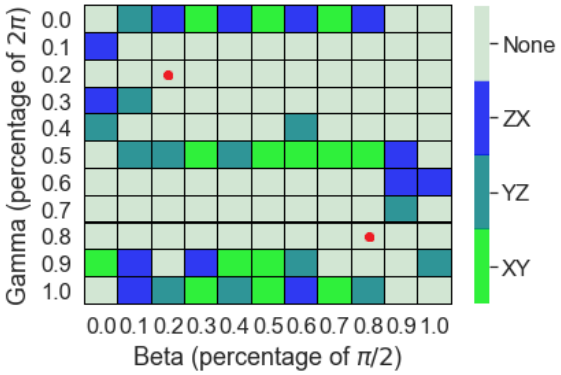

Figure 117: Graph 10 results (with unencoded circuits). The red dot represents an approximate maximum of equation 236.

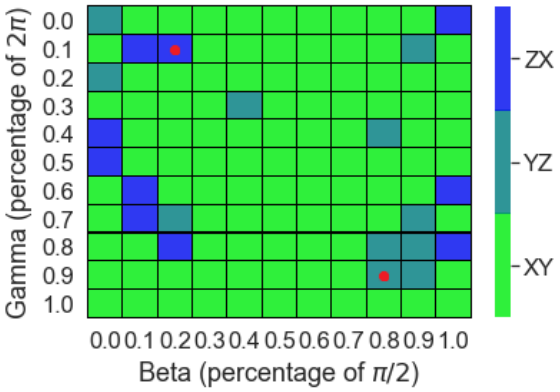

Figure 118: Graph 11 results (just encodings). The red dot represents an approximate maximum of equation 236.

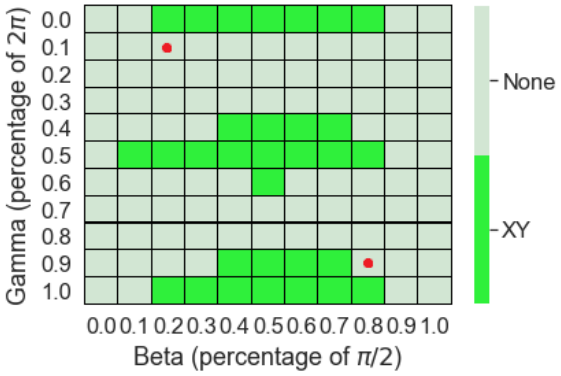

Figure 119: Graph 11 results (with unencoded circuits). The red dot represents an approximate maximum of equation 236.

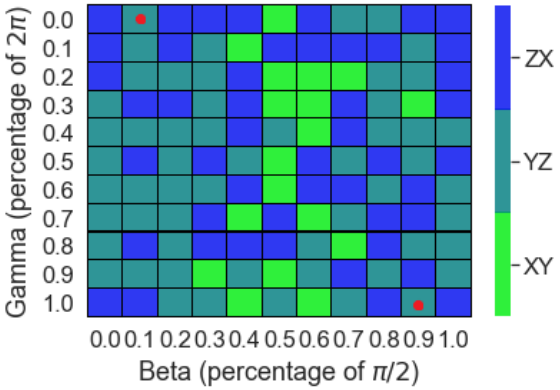

Figure 120: Graph 12 results (just encodings). The red dot represents an approximate maximum of equation 236.

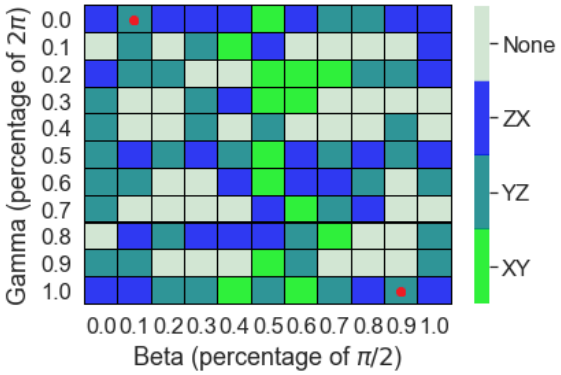

Figure 121: Graph 12 results (with unencoded circuits). The red dot represents an approximate maximum of equation 236.

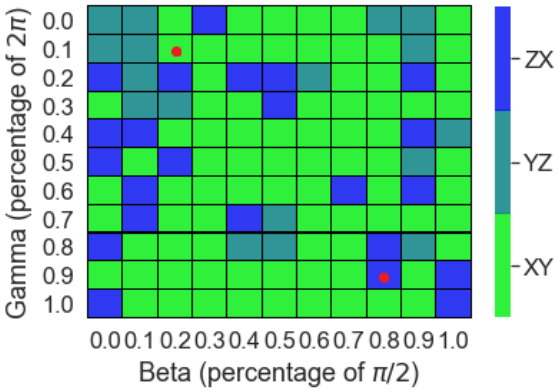

Figure 122: Graph 13 results (just encodings). The red dot represents an approximate maximum of equation 236.

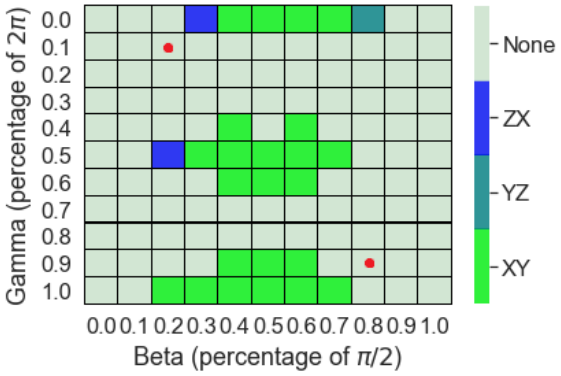

Figure 123: Graph 13 results (with unencoded circuits). The red dot represents an approximate maximum of equation 236.

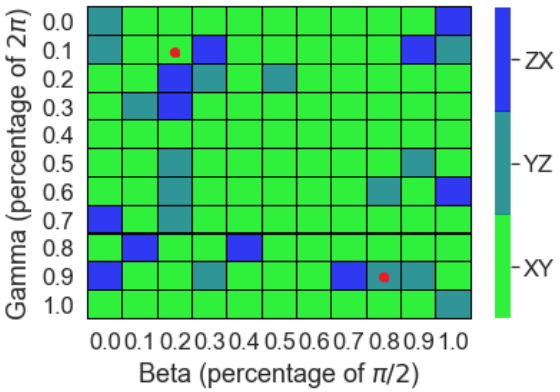

Figure 124: Graph 14 results (just encodings). The red dot represents an approximate maximum of equation 236.

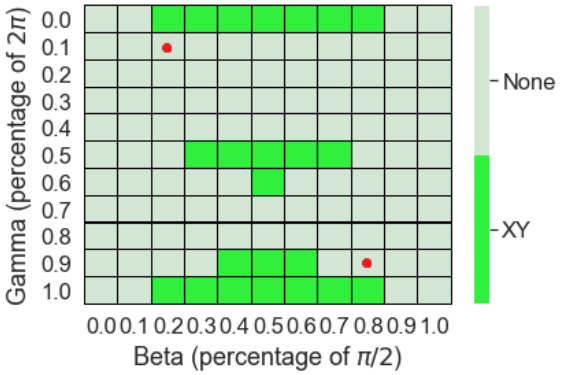

Figure 125: Graph 14 results (with unencoded circuits). The red dot represents an approximate maximum of equation 236.

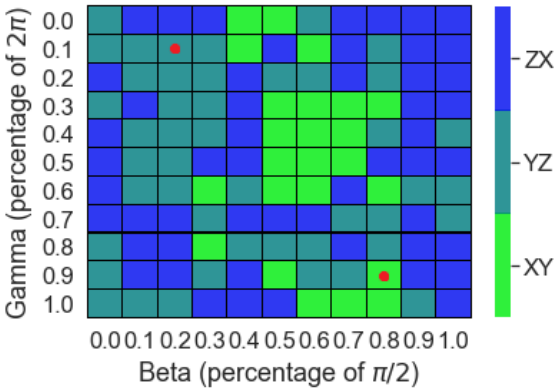

Figure 126: Graph 15 results (just encodings). The red dot represents an approximate maximum of equation 236.

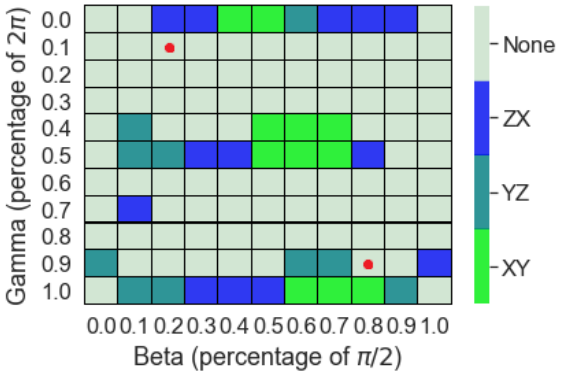

Figure 127: Graph 15 results (with unencoded circuits). The red dot represents an approximate maximum of equation 236.

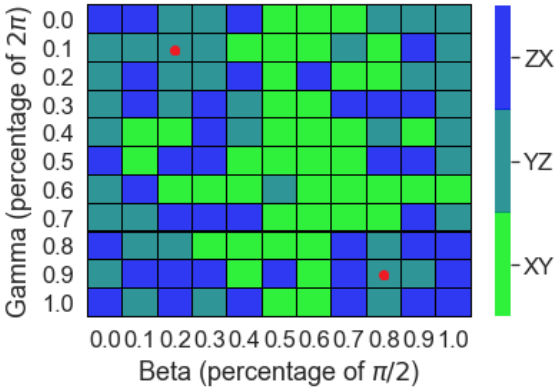

Figure 128: Graph 16 results (just encodings). The red dot represents an approximate maximum of equation 236.

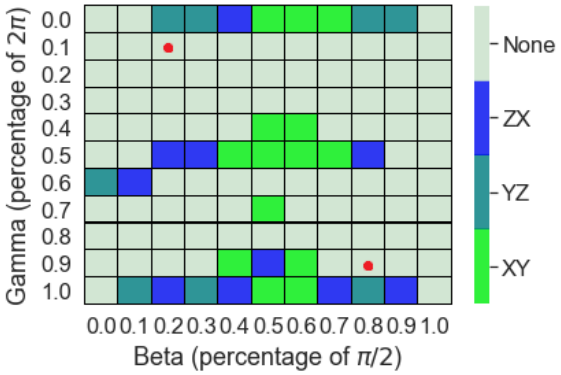

Figure 129: Graph 16 results (with unencoded circuits). The red dot represents an approximate maximum of equation 236.

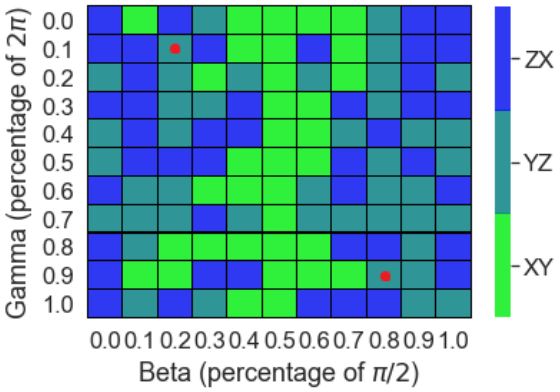

Figure 130: Graph 17 results (just encodings). The red dot represents an approximate maximum of equation 236.

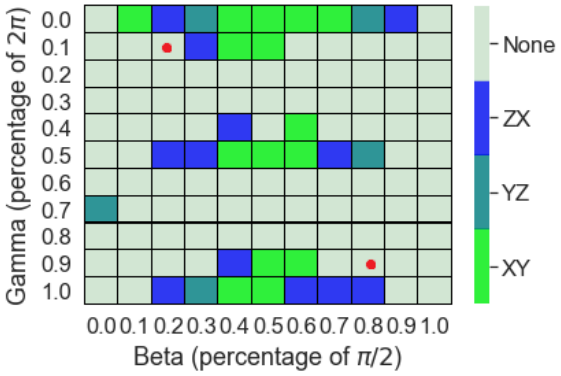

Figure 131: Graph 17 results (with unencoded circuits). The red dot represents an approximate maximum of equation 236.

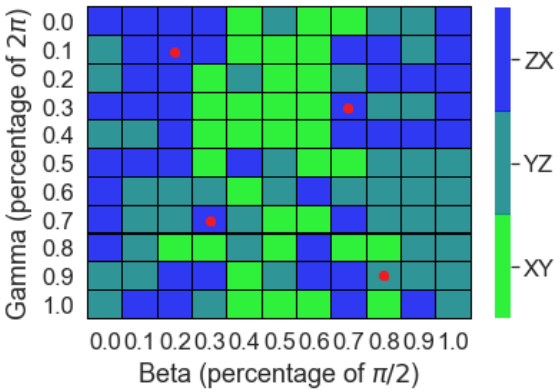

Figure 132: Graph 18 results (just encodings). The red dot represents an approximate maximum of equation 236.

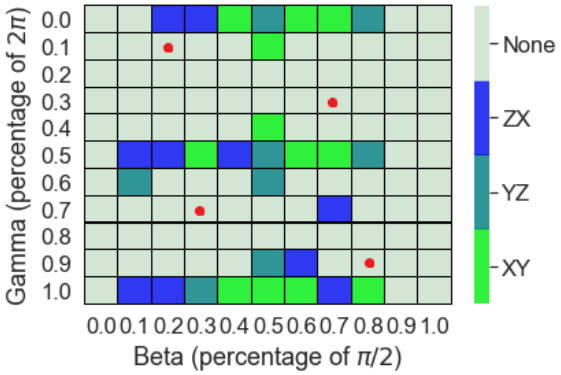

Figure 133: Graph 18 results (with unencoded circuits). The red dot represents an approximate maximum of equation 236.

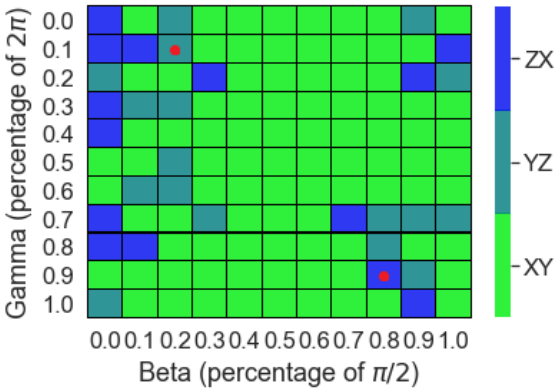

Figure 134: Graph 19 results (just encodings). The red dot represents an approximate maximum of equation 236.

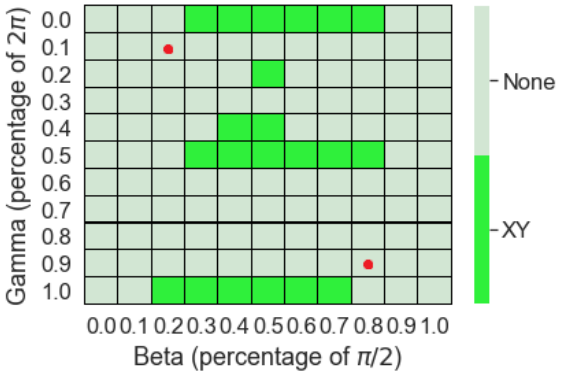

Figure 135: Graph 19 results (with unencoded circuits). The red dot represents an approximate maximum of equation 236.

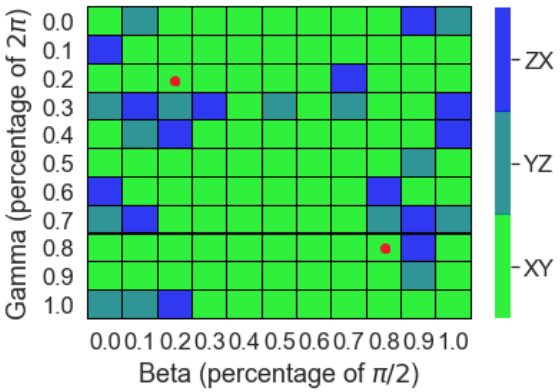

Figure 136: Graph 20 results (just encodings). The red dot represents an approximate maximum of equation 236.

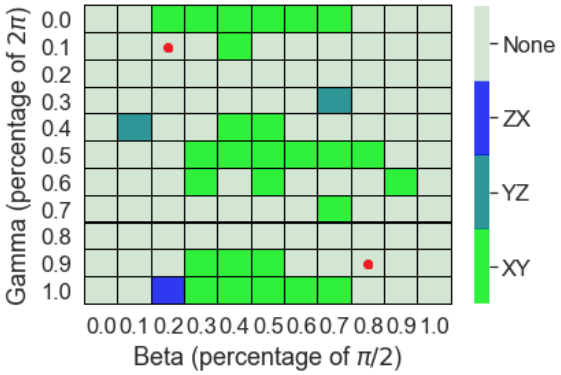

Figure 137: Graph 20 results (with unencoded circuits). The red dot represents an approximate maximum of equation 236.

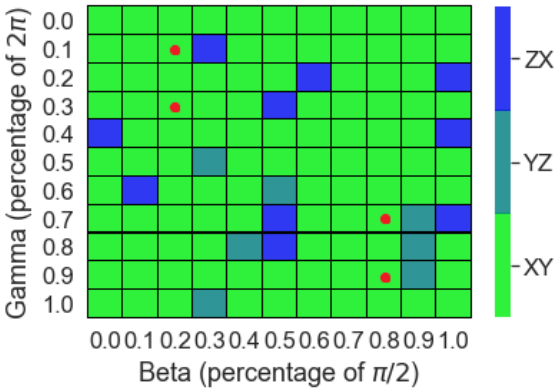

Figure 138: Graph 21 results (just encodings). The red dot represents an approximate maximum of equation 236.

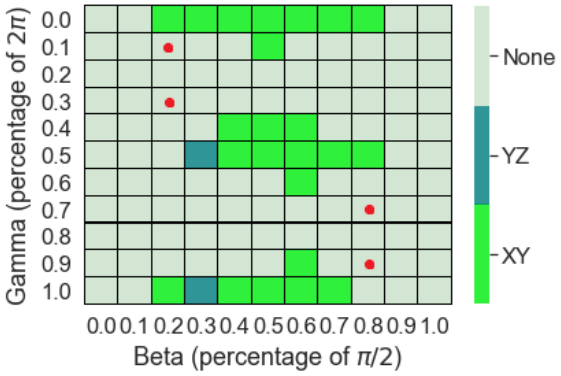

Figure 139: Graph 21 results (with unencoded circuits). The red dot represents an approximate maximum of equation 236.

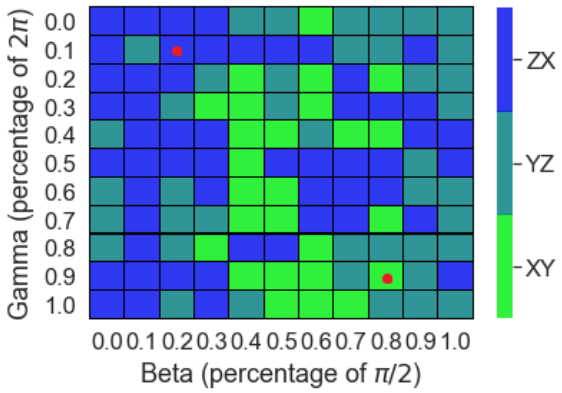

Figure 140: Graph 22 results (just encodings). The red dot represents an approximate maximum of equation 236.

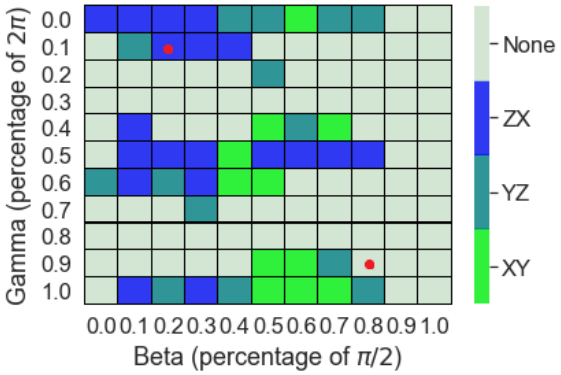

Figure 141: Graph 22 results (with unencoded circuits). The red dot represents an approximate maximum of equation 236.

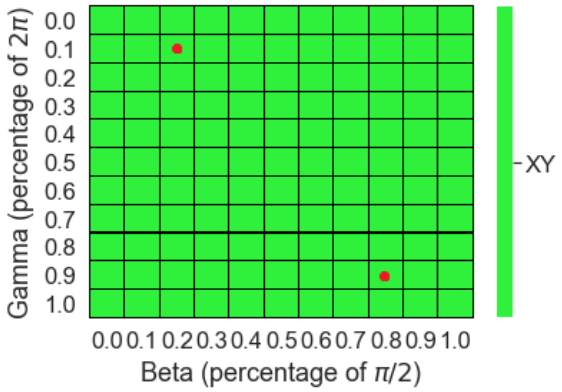

Figure 142: Graph 23 results (just encodings). The red dot represents an approximate maximum of equation 236.

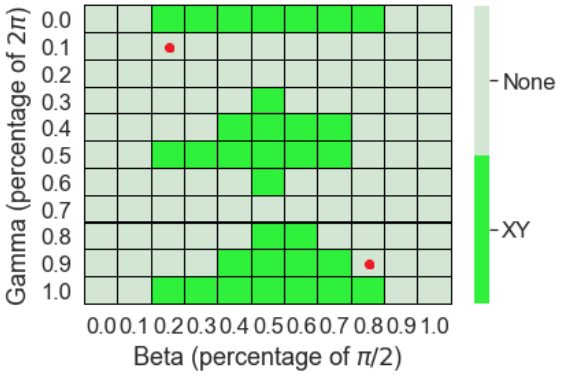

Figure 143: Graph 23 results (with unencoded circuits). The red dot represents an approximate maximum of equation 236.

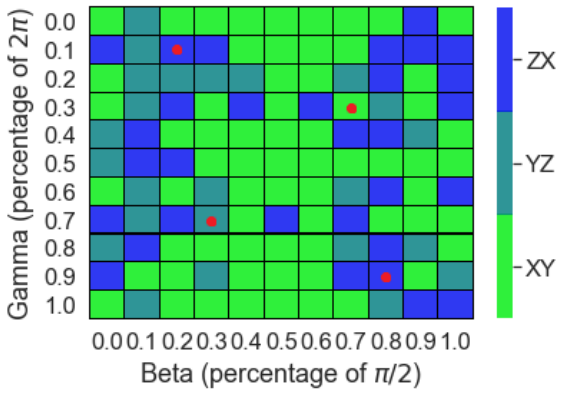

Figure 144: Graph 24 results (just encodings). The red dot represents an approximate maximum of equation 236.

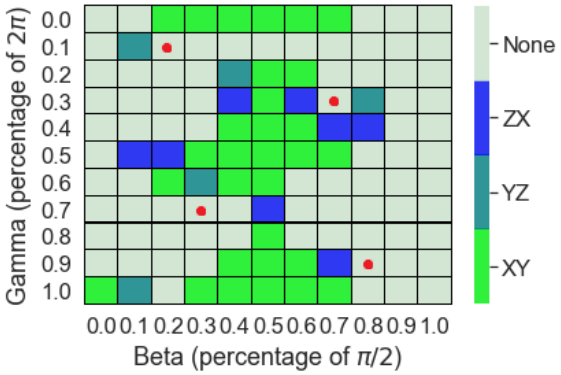

Figure 145: Graph 24 results (with unencoded circuits). The red dot represents an approximate maximum of equation 236.

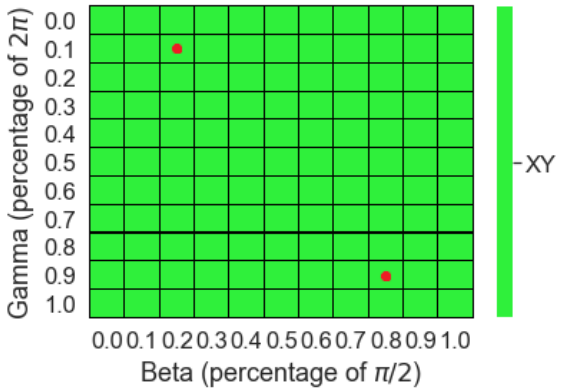

Figure 146: Graph 25 results (just encodings). The red dot represents an approximate maximum of equation 236.

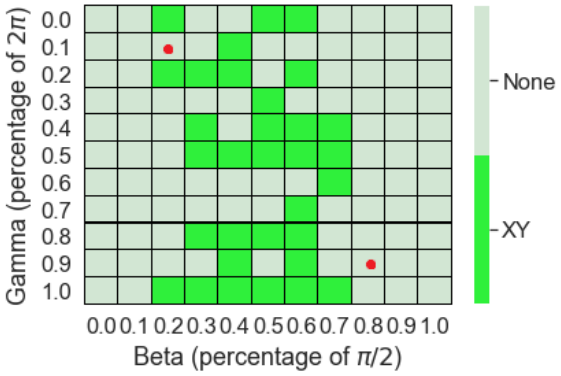

Figure 147: Graph 25 results (with unencoded circuits). The red dot represents an approximate maximum of equation 236.

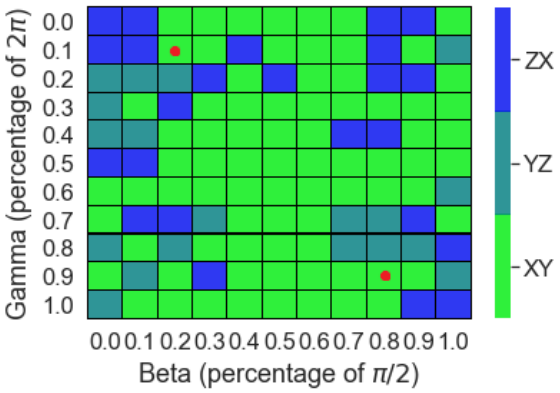

Figure 148: Graph 26 results (just encodings). The red dot represents an approximate maximum of equation 236.

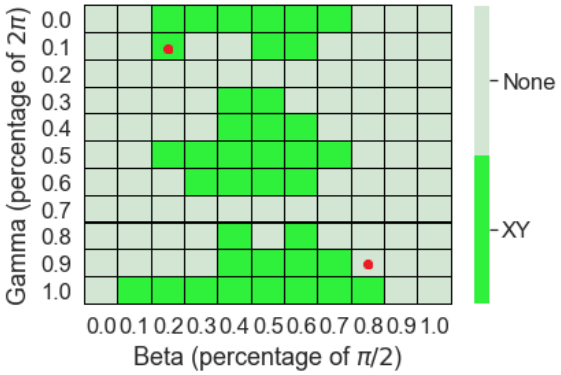

Figure 149: Graph 26 results (with unencoded circuits). The red dot represents an approximate maximum of equation 236.

Bibliography

1. Paul Benioff. The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by turing machines. *Journal of Statistical Physics*, 22(5):563–591, 1980.

2. Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6-7):467–488, 1982.

3. David Deutsch. Quantum theory, the church–turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 400(1818):97–117, 1985.

4. David Deutsch and Richard Jozsa. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 439(1907):553–558, Dec 1992.

5. Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.

6. Clay Math Institute, https://www.claymath.org/millennium-problems, 2022.

7. Matthew Campagna, Lidong Chen, Özgür Dagdelen, Jintai Ding, Jennifer K. Fernick, Nicolas Gisin, Donald Hayford, Thomas Jennewein, Norbert Lütkenhaus, Michele Mosca, Brian Neill, Mark Pecen, Ray Perlner, Grégoire Ribordy, John M. Schanck, Douglas Stebila, Nino Walenta, William Whyte, and Zhenfei Zhang. Quantum safe cryptography and security: An introduction, benefits, enablers and challengers. Technical report, ETSI (European Telecommunications Standards Institute), June 2015.

8. Y. S. Nam and R. Blümel. Performance scaling of shor's algorithm with a banded quantum fourier transform. *Phys. Rev. A*, 86:044303, Oct 2012.

9. Alex Bocharov, Martin Roetteler, and Krysta M. Svore. Factoring with qutrits: Shor's algorithm on ternary and metaplectic quantum architectures. *Phys. Rev. A*, 96:012306, Jul 2017.

10. Implementing shor's algorithm on josephson charge qubits. *Phys. Rev. A*, 70:012319, Jul 2004.

11. Austin G. Fowler and Lloyd C. L. Hollenberg. Scalability of shor's algorithm with a limited set of rotation gates. *Phys. Rev. A*, 70:032329, Sep 2004.

12. Rolf Landauer. Is quantum mechanics useful? *Ultimate Limits of Fabrication and Measurement*, page 237–240, Dec 1995.

13. Swamit S. Tannu and Moinuddin K. Qureshi. Not all qubits are created equal: A case for variability-aware policies for nisq-era quantum computers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '19, page 987–999, New York, NY, USA, 2019. Association for Computing Machinery.

14. James N. Eckstein and Jeremy Levy. Materials issues for quantum computation. *MRS Bulletin*, 38(10):783–789, 2013.

15. John Clarke and Frank K. Wilhelm. Superconducting quantum bits. *Nature*, 453(7198):1031–1042, 2008.

16. D.J. Wineland, C. Monroe, W.M. Itano, D. Leibfried, B.E. King, and D.M. Meekhof. Experimental issues in coherent quantum-state manipulation of trapped atomic ions. *Journal of Research of the National Institute of Standards and Technology*, 103(3):259, 1998.

17. Jürgen Lisenfeld, Alexander Bilmes, Anthony Megrant, Rami Barends, Julian Kelly, Paul Klimov, Georg Weiss, John M. Martinis, and Alexey V. Ustinov. Electric field spectroscopy of material defects in transmon qubits. *npj Quantum Information*, 5(1):105, Nov 2019.

18. R. Landauer. Irreversibility and heat generation in the computing process. *IBM Journal of Research and Development*, 5(3):183–191, 1961.

19. Carlos Sabín, Borja Peropadre, Marco del Rey, and Eduardo Martín-Martínez. Extracting past-future vacuum correlations using circuit qed. *Phys. Rev. Lett.*, 109:033602, Jul 2012.

20. C. D. Wilen, S. Abdullah, N. A. Kurinsky, C. Stanford, L. Cardani, G. D'Imperio, C. Tomei, L. Faoro, L. B. Ioffe, C. H. Liu, A. Opremcak, B. G. Christensen, J. L. DuBois, and R. McDermott. Correlated charge noise and relaxation errors in superconducting qubits. *Nature*, 594(7863):369–373, Jun 2021.

21. Antti P. Vepsäläinen, Amir H. Karamlou, John L. Orrell, Akshunna S. Dogra, Ben Loer, Francisca Vasconcelos, David K. Kim, Alexander J. Melville, Bethany M. Niedzielski, Jonilyn L. Yoder, Simon Gustavsson, Joseph A. Formaggio, Brent A. VanDevender, and William D. Oliver. Impact of ionizing radiation on superconducting qubit coherence. *Nature*, 584(7822):551–556, Aug 2020.

22. Antonio D. Córcoles, Jerry M. Chow, Jay M. Gambetta, Chad Rigetti, J. R. Rozen, George A. Keefe, Mary Beth Rothwell, Mark B. Ketchen, and M. Steffen. Protecting superconducting qubits from radiation. *Applied Physics Letters*, 99(18):181906, October 2011.

23. Asher Peres. Reversible logic and quantum computers. *Phys. Rev. A*, 32:3266–3276, Dec 1985.

24. David DiVincenzo and Peter Shor. Fault-tolerant error correction with efficient quantum codes. *Physical Review Letters*, 77(15):3260–3263, Oct 1996.

25. Daniel Gottesman. Class of quantum error-correcting codes saturating the quantum hamming bound. *Physical Review A*, 54(3):1862–1868, Sep 1996.

26. Rainer Baumann. Quantum error correction (qec), 2003. Semesterthesis 2003.

27. E. Knill, R. Laflamme, R. Martinez, and C. Negrevergne. Benchmarking quantum computers: The five-qubit error correcting code. *Phys. Rev. Lett.*, 86:5811–5814, Jun 2001.

28. C. Ryan-Anderson, J. G. Bohnet, K. Lee, D. Gresh, A. Hankin, J. P. Gaebler, D. Francois, A. Chernoguzov, D. Lucchetti, N. C. Brown, T. M. Gatterman, S. K. Halit, K. Gilmore, J. A. Gerber, B. Neyenhuis, D. Hayes, and R. P. Stutz. Realization of real-time fault-tolerant quantum error correction. *Phys. Rev. X*, 11:041058, Dec 2021.

29. Ming Gong, Xiao Yuan, Shiyu Wang, Yulin Wu, Youwei Zhao, Chen Zha, Shaowei Li, Zhen Zhang, Qi Zhao, Yunchao Liu, and et al. Experimental exploration of five-qubit quantum error-correcting code with superconducting qubits. *National Science Review*, 9(1), 2021.

30. A.Yu. Kitaev. Fault-tolerant quantum computation by anyons. *Annals of Physics*, 303(1):2–30, 2003.

31. A. Holmes, M. Jokar, G. Pasandi, Y. Ding, M. Pedram, and F. T. Chong. Nisq+: Boosting quantum computing power by approximating quantum error correction. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, pages 556–569, Los Alamitos, CA, USA, jun 2020. IEEE Computer Society.

32. Eric Dennis, Alexei Kitaev, Andrew Landahl, and John Preskill. Topological quantum memory. *Journal of Mathematical Physics*, 43(9):4452–4505, 2002.

33. Robert Raussendorf and Jim Harrington. Fault-tolerant quantum computation with high threshold in two dimensions. *Phys. Rev. Lett.*, 98:190504, May 2007.

34. Colin J Trout, Muyuan Li, Mauricio Gutiérrez, Yukai Wu, Sheng-Tao Wang, Luming Duan, and Kenneth R Brown. Simulating the performance of a distance-3 surface code in a linear ion trap. *New Journal of Physics*, 20(4):043038, apr 2018.

35. Sebastian Krinner, Nathan Lacroix, Ants Remm, Agustin Di Paolo, Elie Genois, Catherine Leroux, Christoph Hellings, Stefania Lazar, Francois Swiadek, Johannes Herrmann, and et al. Realizing repeated quantum error correction in a distance-three surface code. *Nature*, 605(7911):669–674, 2022.

36. Austin Fowler, David Wang, and Lloyd Hollenberg. Surface code quantum error correction incorporating accurate error propagation. *Quantum information and computation*, 11, 04 2010.

37. Austin G. Fowler, Matteo Mariantoni, John M. Martinis, and Andrew N. Cleland. Surface codes: Towards practical large-scale quantum computation. *Phys. Rev. A*, 86:032324, Sep 2012.

38. Hendrik Weimer, Markus Müller, Igor Lesanovsky, Peter Zoller, and Hans Peter Büchler. A rydberg quantum simulator. *Nature Physics*, 6(5):382–388, 2010.

39. K. J. Satzinger, Y.-J Liu, A. Smith, C. Knapp, M. Newman, C. Jones, Z. Chen, C. Quintana, X. Mi, A. Dunsworth, and et al. Realizing topologically ordered states on a quantum processor. *Science*, 374(6572):1237–1241, 2021.

40. Han-Ning Dai, Bing Yang, Andreas Reingruber, Hui Sun, Xiao-Fan Xu, Yu-Ao Chen, Zhen-Sheng Yuan, and Jian-Wei Pan. Four-body ring-exchange interactions and anyonic statistics within a minimal toric-code hamiltonian. *Nature Physics*, 13(12):1195–1200, 2017.

41. Xing-Can Yao, Tian-Xiong Wang, Hao-Ze Chen, Wei-Bo Gao, Austin G. Fowler, Robert Raussendorf, Zeng-Bing Chen, Nai-Le Liu, Chao-Yang Lu, You-Jin Deng, and et al. Experimental demonstration of topological error correction. *Nature*, 482(7386):489–494, 2012.

42. E. Knill. Quantum computing with realistically noisy devices. *Nature*, 434, 2005.

43. Simon J Devitt, William J Munro, and Kae Nemoto. Quantum error correction for beginners. *Reports on Progress in Physics*, 76(7):076001, Jun 2013.

44. Norbert M. Linke, Mauricio Gutierrez, Kevin A. Landsman, Caroline Figgatt, Shantanu Debnath, Kenneth R. Brown, and Christopher Monroe. Fault-tolerant quantum error detection. *Science Advances*, 3(10), 2017.

45. J. F. Marques, B. M. Varbanov, M. S. Moreira, H. Ali, N. Muthusubramanian, C. Zachariadis, F. Battistel, M. Beekman, N. Haider, W. Vlothuizen, and et al. Logical-qubit operations in an error-detecting surface code. *Nature Physics*, 18(1):80–86, 2021.

46. IMB Quantum, https://quantum-computing.ibm.com/lab/docs/iql/manage/systems/midcircuit-measurement/, 2021.

47. Luc Devroye, Abbas Mehrabian, and Tommy Reddad. The total variation distance between high-dimensional gaussians with the same mean, 2018.

48. Konstantinos Georgopoulos, Clive Emary, and Paolo Zuliani. Modeling and simulating the noisy behavior of near-term quantum computers. *Phys. Rev. A*, 104:062432, Dec 2021.

49. William H. Gustafson. A note on matrix inversion. *Linear Algebra and its Applications*, 57:71–73, 1984.

50. Tzon-Tzer Lu and Sheng-Hua Shiou. Inverses of 2×2 block matrices. *Computers Mathematics with Applications*, 43(1):119–129, 2002.

51. Farrokh Vatan and Colin Williams. Optimal quantum circuits for general two-qubit gates. *Phys. Rev. A*, 69:032315, Mar 2004.

52. J. F. Marques, B. M. Varbanov, M. S. Moreira, H. Ali, N. Muthusubramanian, C. Zachariadis, F. Battistel, M. Beekman, N. Haider, W. Vlothuizen, A. Bruno, B. M. Terhal, and L. DiCarlo. Logical-qubit operations in an error-detecting surface code, 2021.

53. Hongxiang Chen, Michael Vasmer, Nikolas Breuckmann, and Edward Grant. Machine learning logical gates for quantum error correction, 12 2019.

54. Fernando Pastawski and Beni Yoshida. Fault-tolerant logical gates in quantum error-correcting codes. *Phys. Rev. A*, 91:012305, Jan 2015.

55. L. Hu, Y. Ma, W. Cai, X. Mu, Y. Xu, W. Wang, Y. Wu, H. Wang, Y. P. Song, C.-L. Zou, S. M. Girvin, L.-M. Duan, and L. Sun. Quantum error correction and universal gate set operation on a binomial bosonic logical qubit. *Nature Physics*, 15(5):503–508, May 2019.

56. R. W. Hamming. Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2):147–160, 1950.

57. Diogo Cruz, Romain Fournier, Fabien Gremion, Alix Jeannerot, Kenichi Komagata, Tara Tosic, Jarla Thiesbrummel, Chun Lam Chan, Nicolas Macris, Marc-André Dupertuis, and et al. Efficient quantum algorithms for ghz and w states, and implementation on the ibm quantum computer. *Advanced Quantum Technologies*, 2(5-6):1900015, 2019.

58. Jonathan J. Burnett, Andreas Bengtsson, Marco Scigliuzzo, David Niepce, Marina Kudra, Per Delsing, and Jonas Bylander. Decoherence benchmarking of superconducting qubits. *npj Quantum Information*, 5(1):54, Jun 2019.

59. Rahaf Youssef. Measuring and simulating t1 and t2 for qubits. 8 2020.

60. D Gottesman. The heisenberg representation of quantum computers. 6 1998.

61. Scott Aaronson and Daniel Gottesman. Improved simulation of stabilizer circuits. *Phys. Rev. A*, 70:052328, Nov 2004.

62. IMB Quantum, https://quantum-computing.ibm.com/, 2021.

63. Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. 2014.

64. Seth Lloyd. Quantum approximate optimization is computationally universal. 2018.

65. Edward Farhi and Aram W Harrow. Quantum supremacy through the quantum approximate optimization algorithm. 2016.

66. Zhihui Wang, Stuart Hadfield, Zhang Jiang, and Eleanor G. Rieffel. Quantum approximate optimization algorithm for maxcut: A fermionic view. *Phys. Rev. A*, 97:022304, Feb 2018.

67. Guido Pagano, Aniruddha Bapat, Patrick Becker, Katherine S. Collins, Arinjoy De, Paul W. Hess, Harvey B. Kaplan, Antonis Kyprianidis, Wen Lin Tan, Christopher Baldwin, Lucas T. Brady, Abhinav Deshpande, Fangli Liu, Stephen Jordan, Alexey V. Gorshkov, and Christopher Monroe. Quantum approximate optimization of the long-range ising model with a trapped-ion quantum simulator. *Proceedings of the National Academy of Sciences*, 117(41):25396–25401, 2020.

68. Michael Streif and Martin Leib. Training the quantum approximate optimization algorithm without access to a quantum processing unit. *Quantum Science and Technology*, 5(3):034008, may 2020.

69. Wolfgang Lechner. Quantum approximate optimization with parallelizable gates. *IEEE Transactions on Quantum Engineering*, 1, 02 2018.

70. Edward Farhi, Jeffrey Goldstone, Sam Gutmann, Joshua Lapan, Andrew Lundgren, and Daniel Preda. A quantum adiabatic evolution algorithm applied to random instances of an np-complete problem. *Science*, 292(5516):472–475, 2001.

71. Madita Willsch, Dennis Willsch, Fengping Jin, Hans De Raedt, and Kristel Michielsen. Benchmarking the quantum approximate optimization algorithm. *Quantum Information Processing*, 19(7), 2020.

72. Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph Bardin, Rami Barends, Sergio Boixo, Michael Broughton, Bob Buckley, David Buell, Brian Burkett, Nicholas Bushnell, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Sean Demura, Andrew Dunsworth, and Leo Zhou. Quantum approximate optimization of non-planar graph problems on a planar superconducting processor. 04 2020.

73. Michael R. Garey and David S. Johnson. *Computers and intractability: A guide to the theory of NP-completeness*. FbCNIB, 1996.

74. Richard M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, page 85–103, 1972.

75. https://www.ibm.com/blogs/research/2019/03/power-quantum-device/, 2019.

76. Colin P. Williams. Solving np-complete problems with a quantum computer. *Texts in Computer Science*, page 293–318, 2011.

77. Shaohan Hu, Peng Liu, Chun-Fu (Richard) Chen, and Marco Pistoia. Automatically solving np-complete problems on a quantum computer. *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings*, 2018.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
15–09–2022	Dissertation	Mar 2020 — Sept 2022

4. TITLE AND SUBTITLE

Quantum Error Detection Without Using Ancilla Qubits

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Nicolas Guerrero, Captain, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/ENP)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENP-DS-22-S-044

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Lababoratories
Quantum Information Sciences
26 Electronic Parkway
Rome NY, 13441-4514
Email: afrl.ritq.office@us.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

AFRL/RITQ

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Quantum computers are beset by errors from a variety of sources. Although quantum error correction and detection codes have been developed since the 1990s, these codes require mid-circuit measurements in order to operate. In order to avoid these measurements we have developed a new error detection code that only requires state collapses at the end of the circuit, which we call no ancilla error detection (NAED). We investigate some of the mathematics behind NAED such as which codes can detect which errors. We then ran NAED on three separate types of circuits: Greenberger–Horne–Zeilinger circuits, phase dependent circuits, and a quantum approximate optimization algorithm running the max cut problem. In total, we used the IBMQ quantum computers over 325 million times and were able to show that NAED can be used to improve the performance of the quantum computers. Additionally, we present generalized logical encodings and gates as well as proofs of the fidelity of these gates.

15. SUBJECT TERMS

quantum computing (QC), quantum error correction codes (QEC), quantum error detection codes (QED), no ancilla quantum error detection (NAED), quantum logical gates

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. David Weeks, AFIT/ENP
U	U	U	UU	171	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3636 x4561; david.weeks@afit.edu