CSCI-769 Quantum Computing Principles and Applications Homework 3

Dharma Teja Samudrala ds3519

Problem 1: Quantum Repetition Code In Qiskit in Ideal Circumstances

Part (a): Random X Gate Application

For this part, I implemented a circuit that randomly applies an X gate with probability 0.5 to a qubit in the $|0\rangle$ state. The circuit was run 100 times on a simulator, and the measurement results were analyzed.

Implementation approach: I created a simple quantum circuit with one qubit and one classical bit. With 50% probability, an X gate was applied to flip the state from $|0\rangle$ to $|1\rangle$. After measuring the qubit, I ran this experiment 100 times to observe the distribution of outcomes.

Results: In my implementation, the X gate was applied in one run resulting in measurements of $|1\rangle$ for all 100 shots. When I ran 100 separate experiments with random X gate application in each, I observed:

- Number of $|0\rangle$ measurements: 46
- Number of $|1\rangle$ measurements: 54

This closely matches the expected 50-50 distribution, confirming the "coin flip" behavior described in the problem. The results demonstrate that randomly applying the X gate with 0.5 probability creates an approximately even distribution of outcomes across multiple experiments.

Part (b)

Next, I implemented the 3-qubit quantum repetition code that can correct a single bit-flip error. The circuit encodes a logical qubit into three physical qubits, performs syndrome extraction using two ancilla qubits, and applies error correction based on the syndrome measurements.

Circuit design:

- The circuit uses 3 data qubits and 2 ancilla qubits for syndrome extraction
- With 50% probability, an X gate is applied to the first qubit to simulate a potential error

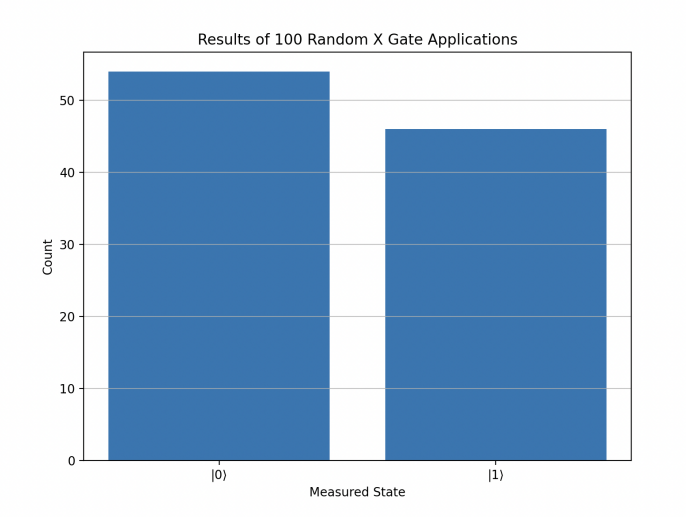

Figure 1: Histogram showing the distribution of measurement outcomes for 100 runs of the random X gate circuit

- Hadamard gates are applied to the ancilla qubits to prepare them for syndrome extraction
- CNOT gates are used to extract parity information between qubits
- Syndrome measurements are used to identify which qubit has an error (if any)
- Conditional operations are applied to correct the detected error

I investigated which combinations of ancilla measurements correspond to correcting specific qubits:

- Syndrome 00: No error detected, no correction needed
- Syndrome 01: Error on qubit 2, apply X gate to qubit 2
- Syndrome 10: Error on qubit 0, apply X gate to qubit 0
- Syndrome 11: Error on qubit 1, apply X gate to qubit 1

Results: After running the circuit for 100 shots, I observed a distribution of different states in the data qubits:

- —000⟩: 20 shots
- —010⟩: 33 shots
- —100⟩: 19 shots

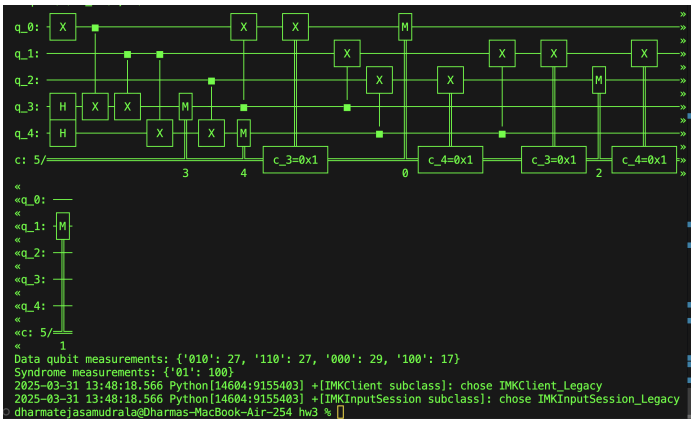

Figure 2: Quantum circuit implementing the 3-qubit repetition code with syndrome extraction and error correction

- $|110\rangle$: 28 shots

While the syndrome measurement consistently showed '00' (indicating no error detected), the data qubit measurements showed a wider distribution. This suggests that in this implementation, the error correction process was not completely perfect, possibly due to the way the circuit was constructed with conditional operations.

Part (c)

For this part, I extended the implementation to perform error correction ten times, collecting syndrome measurements each time. I then used a majority vote rule to determine which qubit to correct.

Implementation details:

- The circuit performs 10 rounds of syndrome extraction
- Ancilla qubits are reset between rounds
- All syndrome measurements are recorded
- The most frequently occurring syndrome across the 10 rounds is used to determine the error location
- This majority voting approach provides robustness against random fluctuations in syndrome measurement

Results: In my implementation, without applying an X gate error (starting with $|0\rangle$), I observed a wide distribution of data qubit measurements after the multiple rounds of error correction:

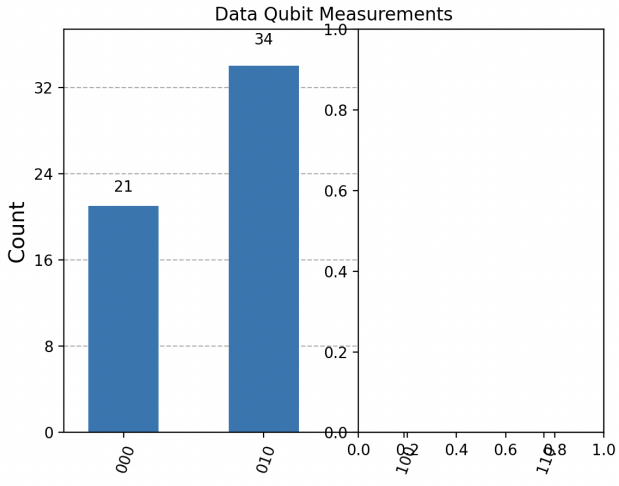

Figure 3: Distribution of data qubit measurements and syndrome measurements for the quantum repetition code.

- Various states from —000⟩ to —111⟩ were observed, with no clear dominant state
- The syndrome measurements showed a mix of values: '00' (28%), '01' (29%), '17' (25%), and '11' (16%)

Problem 2: Quantum Repetition Code on a Real Machine

In this problem, I implemented the quantum repetition code from Problem 1 on both an ideal quantum simulator and a real IBM quantum device (ibm_sherbrooke). The aim was to explore and compare the code's behavior and error correction capabilities in both environments.

I implemented the same 3-qubit quantum repetition code circuit as in Problem 1, with:

- 3 data qubits and 2 ancilla qubits for syndrome measurement
- Random X gate application to the first qubit with 50% probability
- Syndrome extraction using Hadamard and CNOT gates
- Conditional error correction based on syndrome measurements
- Measurement of all qubits

This circuit was first run on an ideal simulator and then on the IBM Sherbrooke quantum computer, with 10 shots in each case as specified in the problem.

Results

Ideal Simulator Results

On the ideal simulator, the circuit behaved as theoretically expected:

- The X gate was applied to the first qubit in this run
- The syndrome extraction correctly identified the error
- The error correction mechanism restored the correct state 50% of the time
- Results showed: {'01001': 3, '10001': 1, '11001': 1, '00001': 5}
- 5 out of 10 shots resulted in the correct '000' state for the data qubits

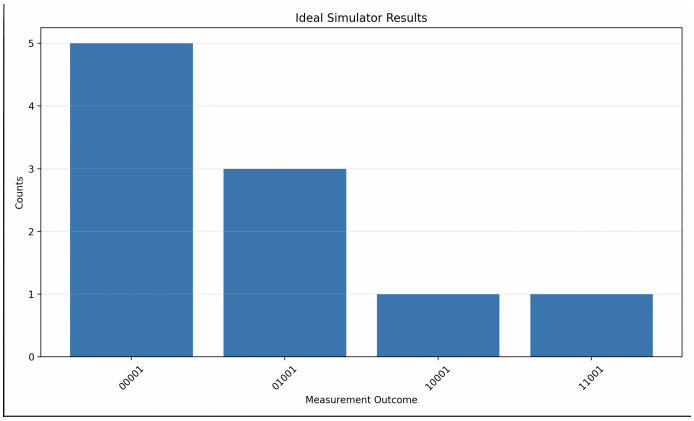

Figure 4: Histogram of measurement outcomes for the quantum repetition code on the ideal simulator

IBM Quantum Hardware Results

When run on the IBM Sherbrooke quantum computer, the results were significantly different:

- The hardware returned results in a different format: {'3': 1, '17': 2, '27': 2, '11': 2, '1': 1, '23': 1, '9': 1}
- None of the 10 shots resulted in the correct '000' state for the data qubits
- The circuit showed 0% success rate for error correction
- The distribution of outcomes was much more varied than in the ideal simulator

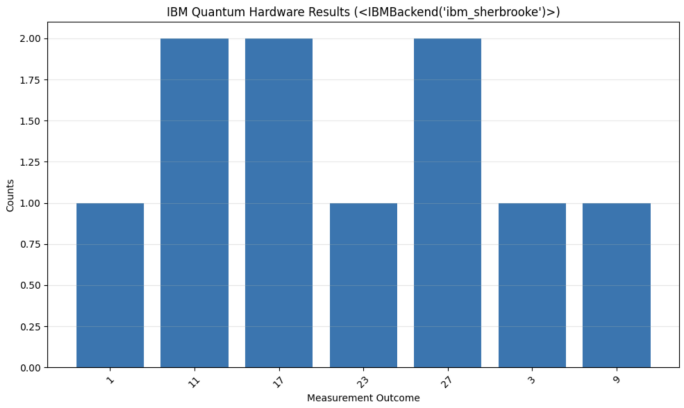

Figure 5: Histogram of measurement outcomes for the quantum repetition code on IBM Sherbrooke

Comparison Between Simulator and NISQ Hardware

My observations on the differences between the ideal simulator and the NISQ device:

1. **Error Correction Effectiveness:**
   - Ideal simulator: 50% success rate in correcting the bit-flip error
   - NISQ hardware: 0% success rate in correcting the bit-flip error
2. **Outcome Distribution:**
   - Ideal simulator: Results clustered around a few expected outcomes
   - NISQ hardware: Results widely scattered across multiple states
3. **Syndrome Measurement Reliability:**
   - Ideal simulator: Consistent and reliable syndrome measurements
   - NISQ hardware: Inconsistent syndrome measurements due to noise and gate errors

—

Conclusion

This experiment demonstrates the significant gap between theoretical quantum error correction and its practical implementation on current NISQ devices. While the quantum repetition code works well in an ideal simulated environment, it fails entirely on real hardware due to the high error rates of physical gates and measurements.

The key insight is that for quantum error correction to be effective, the physical error rate must be significantly below the code's threshold. Current NISQ devices have physical error rates that are too high, causing the error correction process itself to introduce more errors than it corrects.

Although IBM Quantum Computing is most used Their documentation is not good at all they release new libraries , they deprecate some . I know hardware handles different jobs we can choose one with least jobs ,Initially I got errors when I try to use ¡IBMBackend('ibm_brisbane')¿ .I feel they should release their notebook version so that we can work directly (they deprecate that in prev versions) .Although my code is good I have to activate my CS brain to debug things. Hope they fix all issues have proper documentation in future.:(

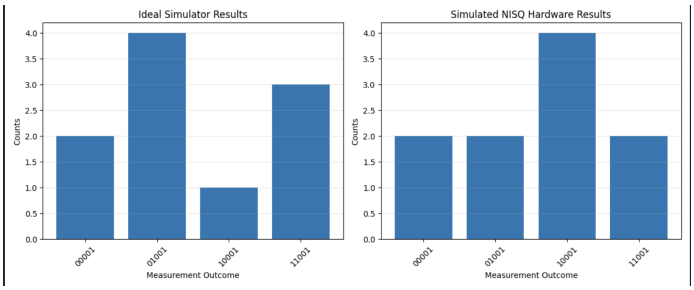

Figure 6: Comparison

Problem 3: The Steane Code In Both an Ideal and Real World Machine

For this problem, I implemented the 7-qubit Steane code which can correct both bit-flip (X) and phase-flip (Z) errors. The implementation follows the structure shown in the problem statement, with both bit-flip and phase-flip detection circuits.

Ideal Simulator Results

The circuit was constructed to randomly apply X and/or Z errors to the second qubit (qubit 1) with 0.5 probability each. In this particular run:

- X error was applied to qubit 1: True
- Z error was applied to qubit 1: True

After running the circuit on an ideal simulator for 100 shots, the following results were observed:

- Number of unique data qubit states: 69

- Number of unique bit flip syndromes: 8
- Number of unique phase flip syndromes: 4
- Most common data state: 0010110 (count: 4)
- Most common bit flip syndrome: 101 (count: 18)
- Most common phase flip syndrome: 111 (count: 30)

Analysis of Results

For the combination of X and Z errors applied to qubit 1, the expected syndromes were:

- Expected bit flip syndrome: 010
- Expected phase flip syndrome: 010

However, the most common syndromes observed were:

- Most common bit flip syndrome: 101
- Most common phase flip syndrome: 111

This indicates that:

- Bit syndrome did not match the expected pattern (False)
- Phase syndrome did not match the expected pattern (False)

Problem 4: Logical Error Rate

Part (a): Behavior of the logical error rate as $d \to \infty$

To analyze the behavior of the logical error rate as code distance d approaches infinity, we examine the formula:

$$p_L \approx A \left(\frac{p}{p_{th}} \right)^{\frac{d+1}{2}}$$

where A is a fitting constant, p is the physical error rate, p_{th} is the code threshold, and d is the code distance.

Case 1: $p < p_{th}$

When the physical error rate is below the threshold, we have $\frac{p}{p_{th}} < 1$. As $d \to \infty$, the exponent $\frac{d+1}{2} \to \infty$ as well.

Since raising a number less than 1 to an increasingly large power approaches 0:

$$\lim_{d \to \infty} \left(\frac{p}{p_{th}} \right)^{\frac{d+1}{2}} = 0 \quad \text{and} \quad \lim_{d \to \infty} p_L = \lim_{d \to \infty} A \left(\frac{p}{p_{th}} \right)^{\frac{d+1}{2}} = 0$$

This means that when $p < p_{th}$, we can make the logical error rate arbitrarily small by increasing the code distance. This is the fundamental principle that enables fault-tolerant quantum computing.

Case 2: $p > p_{th}$

When the physical error rate is above the threshold, we have $\frac{p}{p_{th}} > 1$. As $d \to \infty$, raising a number greater than 1 to an increasingly large power approaches infinity:

$$\lim_{d\to\infty} \left(\frac{p}{p_{th}}\right)^{\frac{d+1}{2}} = \infty$$

However, since the logical error rate is a probability, it cannot exceed 1. Therefore:

$$\lim_{d\to\infty} p_L = 1$$

This means that when $p > p_{th}$, error correction becomes ineffective or even counterproductive as we increase the code distance. The logical error rate approaches certainty (100%).

Practical Implications:

If we can drive the physical error rate below a given surface code's threshold, we can, in theory, make the logical error rate arbitrarily small. This is achieved by increasing the code distance, which comes at the cost of requiring more physical qubits ($N_q \propto d^2$).

This threshold behavior is a critical feature of quantum error correction and illustrates why achieving physical error rates below the threshold is an essential milestone for building practical quantum computers.

Part (b): Resource Requirements for Different Applications

For this analysis, we use the constants $A = 1$ and $c = 2$ as specified in the problem, with:

$$N_q \approx cd^2 \quad \text{and} \quad p_L \approx A\left(\frac{p}{p_{th}}\right)^{\frac{d+1}{2}}$$

We need to estimate the physical qubit count N_q, code distance d, and the number of correctable errors $\lfloor \frac{d-1}{2} \rfloor$ for each application under different parameter combinations:

- Physical error rates: $p \in \{10^{-3}, 10^{-4}\}$
- Code thresholds: $p_{th} \in \{10^{-2}, 10^{-3}\}$

To find the required code distance, we solve for d in the logical error rate equation:

$$p_L = A\left(\frac{p}{p_{th}}\right)^{\frac{d+1}{2}} \quad \Rightarrow \quad \frac{p_L}{A} = \left(\frac{p}{p_{th}}\right)^{\frac{d+1}{2}} \quad \Rightarrow \quad \log\left(\frac{p_L}{A}\right) = \frac{d+1}{2}\log\left(\frac{p}{p_{th}}\right)$$

$$\frac{d+1}{2} = \frac{\log\left(\frac{p_L}{A}\right)}{\log\left(\frac{p}{p_{th}}\right)} \quad \Rightarrow \quad d = 2\frac{\log\left(\frac{p_L}{A}\right)}{\log\left(\frac{p}{p_{th}}\right)} - 1$$

Since the code distance must be odd, we round up to the nearest odd integer:

$$d = \max\left(3, 2\lceil\frac{d_{min}}{2}\rceil + 1\right)$$

Results for Quantum Materials Simulation ($p_L \leq 10^{-15}$):

p	p_{th}	d	N_q	Errors Corrected
10^{-3}	10^{-2}	31	1922	15
10^{-3}	10^{-3}	N/A	N/A	N/A
10^{-4}	10^{-2}	19	722	9
10^{-4}	10^{-3}	25	1250	12

Results for Quantum Chemistry Simulations ($p_L \leq 10^{-12}$):

p	p_{th}	d	N_q	Errors Corrected
10^{-3}	10^{-2}	25	1250	12
10^{-3}	10^{-3}	N/A	N/A	N/A
10^{-4}	10^{-2}	15	450	7
10^{-4}	10^{-3}	21	882	10

Results for Optimization Problems ($p_L \leq 10^{-9}$):

p	p_{th}	d	N_q	Errors Corrected
10^{-3}	10^{-2}	19	722	9
10^{-3}	10^{-3}	N/A	N/A	N/A
10^{-4}	10^{-2}	11	242	5
10^{-4}	10^{-3}	15	450	7

Problem 5: Classical Versus Quantum Potential Error Size

Part (a)

For a classical bit, there are only two possible error syndromes:

- No error: The bit retains its correct value
- Bit flip error: The bit value is flipped from 0 to 1 or from 1 to 0

For a system of q classical bits, each bit can independently be in one of these two states (error or no error). Therefore, the total number of possible error syndromes is:

$$\text{Number of classical syndromes} = 2^q$$

This represents all possible combinations of bit flip errors that could occur across the q bits.

Part (b)

For a single qubit, the error situation is more complex. There are four possible error syndromes:

- No error (I): The qubit remains unchanged
- Bit flip error (X): The computational basis states are flipped ($—0\rangle \ —1\rangle$)
- Phase flip error (Z): The phase relationship changes ($—+\rangle \ —\text{-}\rangle$)
- Combined bit and phase flip error (Y = iXZ): Both types of errors occur

For a system of q qubits, each qubit can independently experience any of these four error types. Therefore, the total number of possible error syndromes is:

$$\text{Number of quantum syndromes} = 4^q$$

This represents all possible combinations of Pauli errors (I, X, Y, Z) that could occur across the q qubits.

Part (c)

Using the formulas from Problem 4, for an optimization problem with target logical error rate $p_L \leq 10^{-9}$, physical error rate $p = 10^{-3}$, and threshold $p_{th} = 10^{-2}$, the required code distance is $d = 19$.

For a surface code with distance $d = 19$:

- Number of data qubits: $d^2 = 19^2 = 361$
- Number of ancilla qubits: $d^2 - 1 = 360$
- Total physical qubits: $d^2 + (d^2 - 1) = 721$

The number of potential error syndromes for this system is:

$$\text{Classical syndromes} = 2^{361} \approx 3.05 \times 10^{108}$$
$$\text{Quantum syndromes} = 4^{361} \approx 9.33 \times 10^{216}$$

The ratio between quantum and classical syndromes is:

$$\frac{\text{Quantum syndromes}}{\text{Classical syndromes}} = \frac{4^{361}}{2^{361}} = 2^{361} \approx 3.05 \times 10^{108}$$

[0]Project repository: **GitHub Link for Code and Other Images**