東京大学
量子ソフトウェア寄付講座

第5回 量子ソフトウェアハンズオン

2024年 3月 11日

本日の演習の概要

- 量子回路における誤り訂正符号について、Qiskitのシミュレータを用いて実装を行います。
- また、演習の後半では、誤り低減(Error Mitigation)の技術についても簡単に紹介します。
- 演習を通じて、以下の目標を達成することを目指します。

本日の目標

- **誤り訂正**

1. 量子回路の誤り訂正符号の実装方法を学ぶ。
2. 誤り訂正符号が誤り訂正に成功する場合/失敗する場合を確認する。
3. 符号化された状態での論理ビット操作を試してみる。
4. 実際の誤り訂正において、求められる性質について考えてみる。

- **誤り低減**

1. 誤り低減の手法の一つである、外挿法（ZNE：Zero Noise Extrapolation)の実装方法を学ぶ。

Error Mitigationの訳語には、誤り低減/緩和/軽減/抑制/補償など、様々な語が使われており、統一されていない。
今回の講義では、「誤り低減」という語を用いることにする。

演習の内容 – 誤り訂正

- 前述の目標を達成するため、本日は3つの演習に取り組んでいただきます
- 分からない点などがあれば、積極的に周囲の人と相談しながら取り組んでください
  - 講師側でも適宜サポートに回ります

#	演習テーマ	概要
1	誤り訂正符号の実装	• 1論理量子ビットの誤り訂正を行う方法について、実装を行います。 • 誤り訂正が成功する場合/失敗する場合を実際に確認します。
2	論理ビット操作	• 符号化された論理量子ビットの状態で、各種操作を実施する方法を体験します。 • 今回の演習では、2論理量子ビットを用いてベル状態を作成する回路を題材にします。
3	表面符号のデモ	• 誤り訂正の実現方法の一つとして期待されている表面符号について、簡単なデモを実施します。

演習環境の準備

- 今回の演習はGoogle Colaboratoryを使って、演習・解説を行います
- 各自のPCのローカル環境上で行っていただいても構いませんが、ご自身の環境に依存するエラー対応については、自己責任でお願いいたします

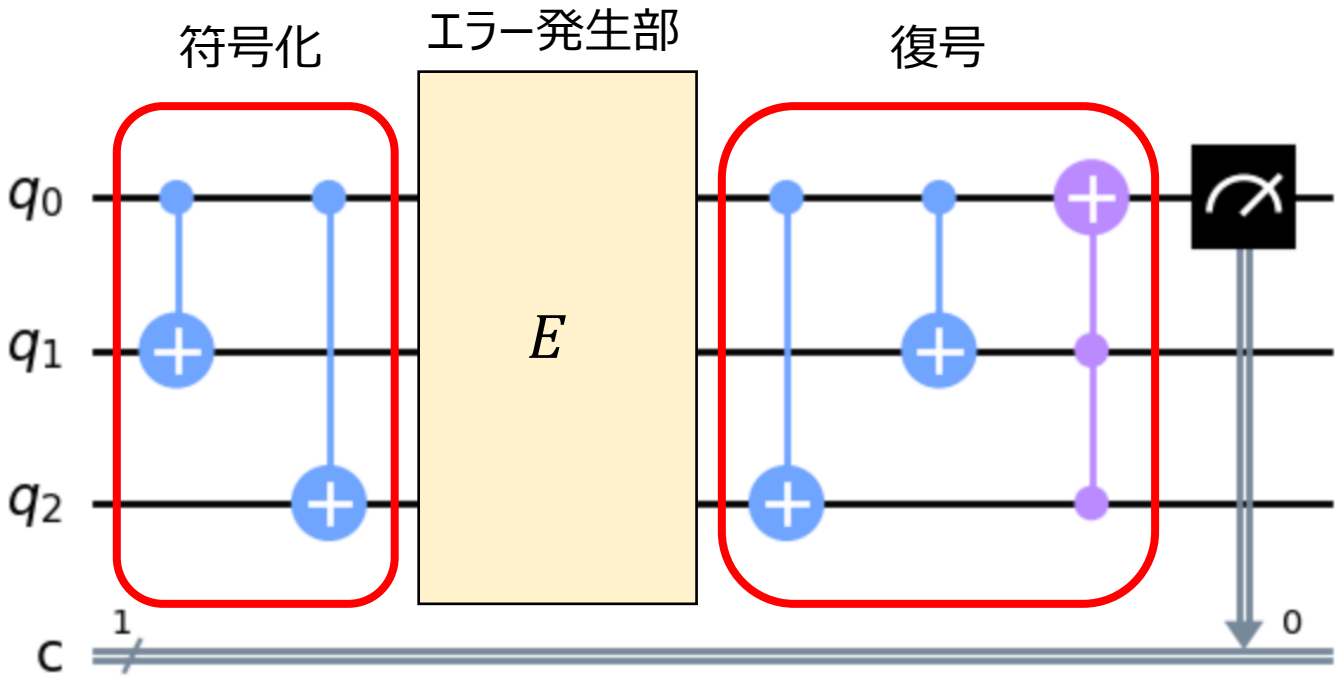
今回の講義のgithubページに7つのノートブックを掲載してあります。

ご自身のドライブにコピーしていただき、Google Colab環境で「01_誤り訂正符号の実装.ipynb」を開いてください。

今回、利用するライブラリの一部にバージョン指定をかけています。
これは、最新のバージョンでは、課題の一部で量子回路の出力が想定と異なる事態が確認できたためです。
詳細は、該当の課題の時点で説明しますが、ローカルで行う方はご注意ください。

演習1．誤り訂正符号の実装

- 最初に、1量子ビットのビット反転エラーを修正できる回路を作成します。
- 古典コンピュータにおけるシンプルな多数決による修正方法と基本的には同じです。



演習1．誤り訂正符号の実装

- 1量子ビットのビット反転エラーを修正できる回路では、2量子ビット以上がビット反転してしまうと、エラーを修正できません。
- しかしながら、2量子ビット以上でエラーが発生する確率が、1量子ビットでのエラー発生率より低ければ、エラーが発生する確率を下げる役割を果たせます。

α：1量子ビットあたりのエラー発生確率

エラーが発生する確率

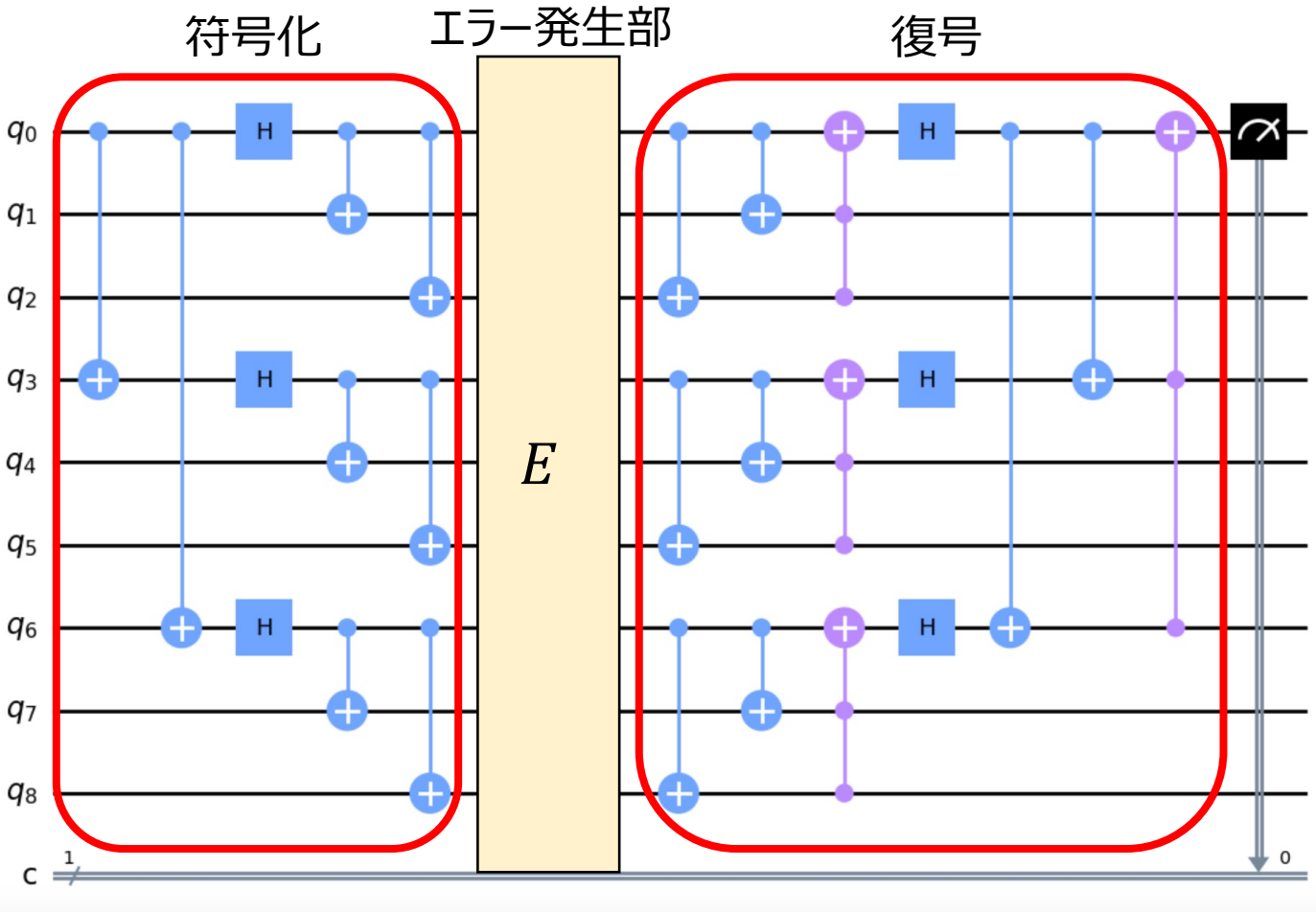
$$1-\underbrace{(1-\alpha)^3+3\alpha(1-\alpha)^2}$$

エラーが訂正できる確率
（エラー無しの確率）＋（1量子ビットのみエラーの確率）

$\alpha = 0.5$のとき、ちょうど0.5になり、
$\alpha < 0.5$では、αより低い値を取る
→ エラー発生確率が下がる

演習1. 誤り訂正符号の実装

- 1量子ビットのビット反転エラーと位相反転エラーに対応できるShorの符号について、実際に実装してみます。
- Shorの符号では9個の物理量子ビットを利用して、1個の論理量子ビットを作成します。



ディスカッション

- Shorの符号を実際の誤り訂正に活用しようとした際に、課題となりそうな点、気をつけなければいけない点について、演習の結果を踏まえて考えてみてください。
- 周辺の方と演習の結果の解釈なども含めて、ディスカッションをしてみましょう。

ディスカッションの進め方

1. 演習の結果について共有する
2. 演習の結果や内容について意見を交換する

演習1．誤り訂正符号の実装

- Shorの符号では、9量子ビットのうち、2量子ビット以上でエラーが発生すると、エラーを訂正できません。
- このため、9量子ビット中2量子ビット以上でエラーが発生する確率が、単一のビットのエラー率よりも高い場合、全体で誤り確率が上昇してしまいます。
- おおよその閾値は3%程度です。

α：1量子ビットあたりのエラー発生確率

エラーが発生する確率

$$1-\underbrace{\{(1-\alpha)^9+9(1-\alpha)^8\alpha\}}_{\text{エラーが訂正できる確率}}$$

$\alpha = 0.05$のとき、0.071
$\alpha = 0.03$のとき、0.028
$\alpha = 0.01$のとき、0.003

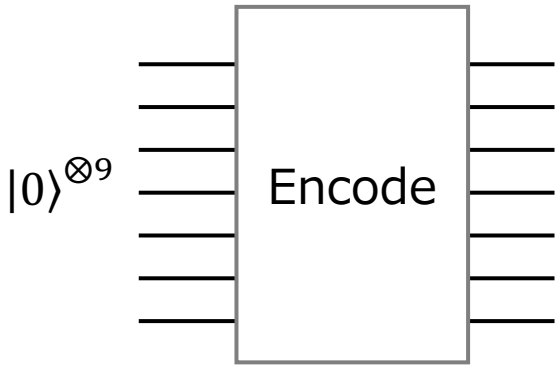
このため、符号化に多くの量子ビットを使うアルゴリズムでは、十分に誤り確率が低い必要があります。
誤り訂正を実際に活用していくためには、以下のようなことが重要になります。

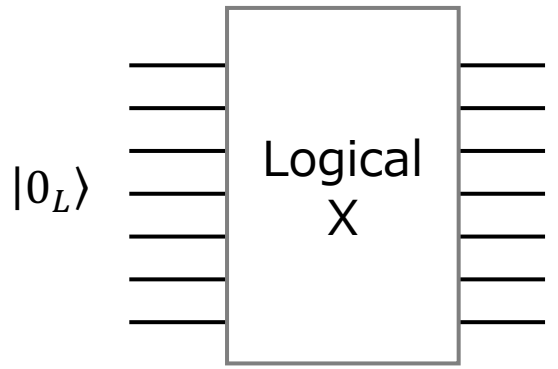
- エラーの発生率そのものを下げる
- 効率のよい符号化をする

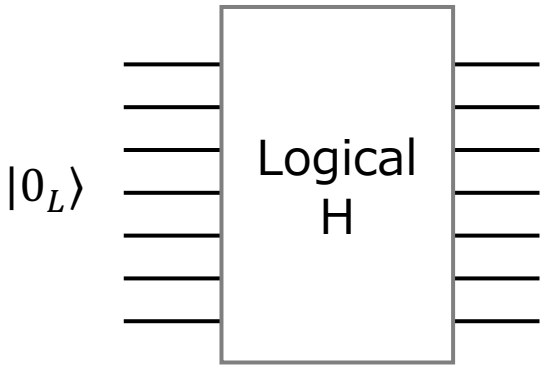
演習2.1. Shorの符号による論理ビット操作

- 現在の量子コンピュータにおいては、各種ゲート操作も大きなノイズの一因です。
- したがって、符号化された論理量子ビットに対して、各種ゲート操作に相当する操作を行い、このエラーを修正することが望ましいです。
- まずは、shorの符号を用いて、実装を確認します。

＜論理ビット操作のイメージ図＞



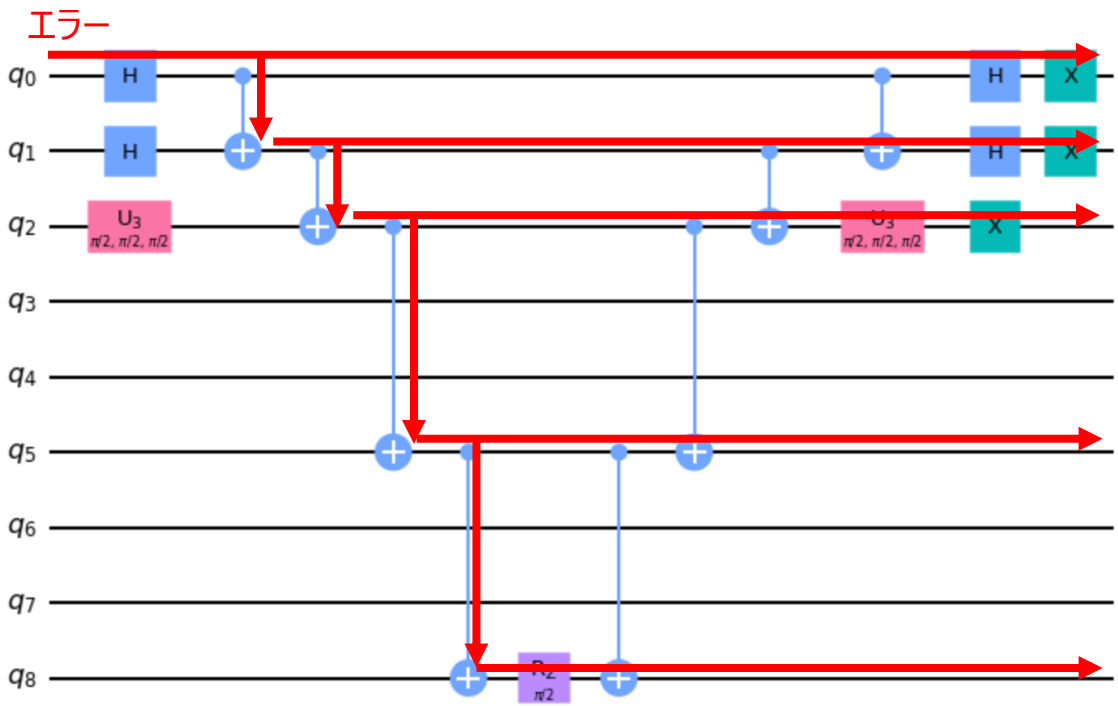




演習2.1. Shorの符号による論理ビット操作

- Shorの符号における論理ゲート操作の実装には課題が存在します。
- 論理Hゲートの実装では内部に2量子ビットゲートが存在するため、エラーが発生していると、他の量子ビットに伝播してしまう恐れがあります。
- このため、せっかく冗長化した意味が薄れてしまいます。

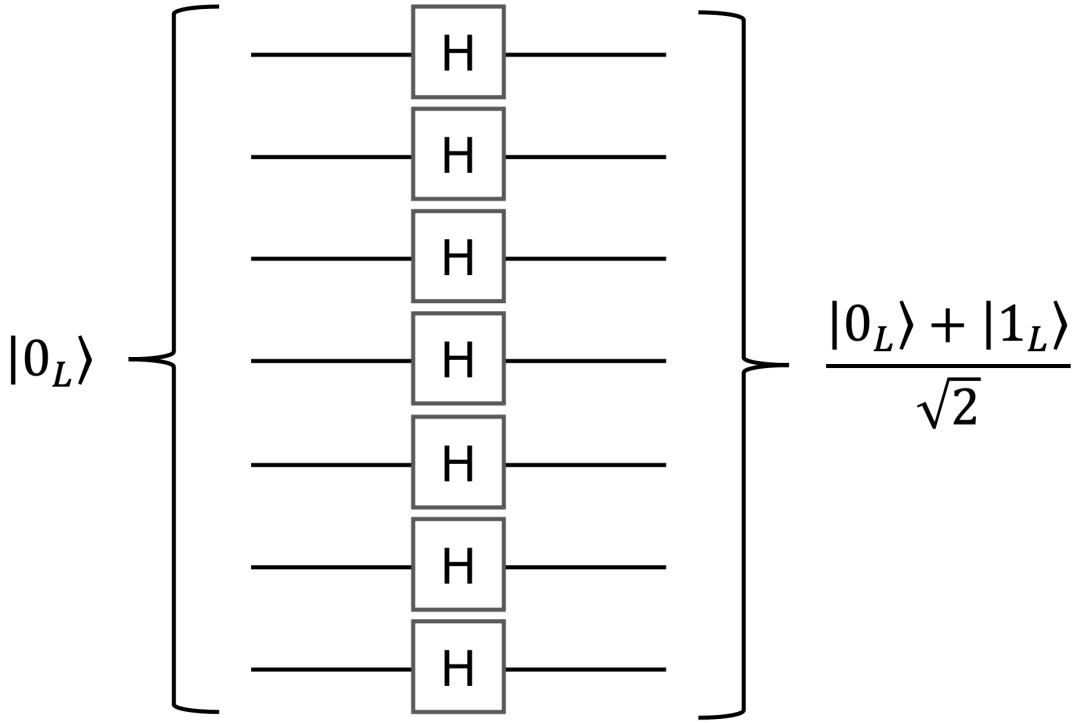
＜論理Hゲートでエラーが伝播していくイメージ＞



演習2.2. Steane符号による論理ビット操作

- ここで、Steane符号という別の誤り訂正符号を導入します。
- この符号では、論理Hゲートや、論理Xゲートを実装する際に、各量子ビットに並列に操作を加えることで、論理ビット操作を実現できます。
  - このような性質をTransversalityと呼びます。

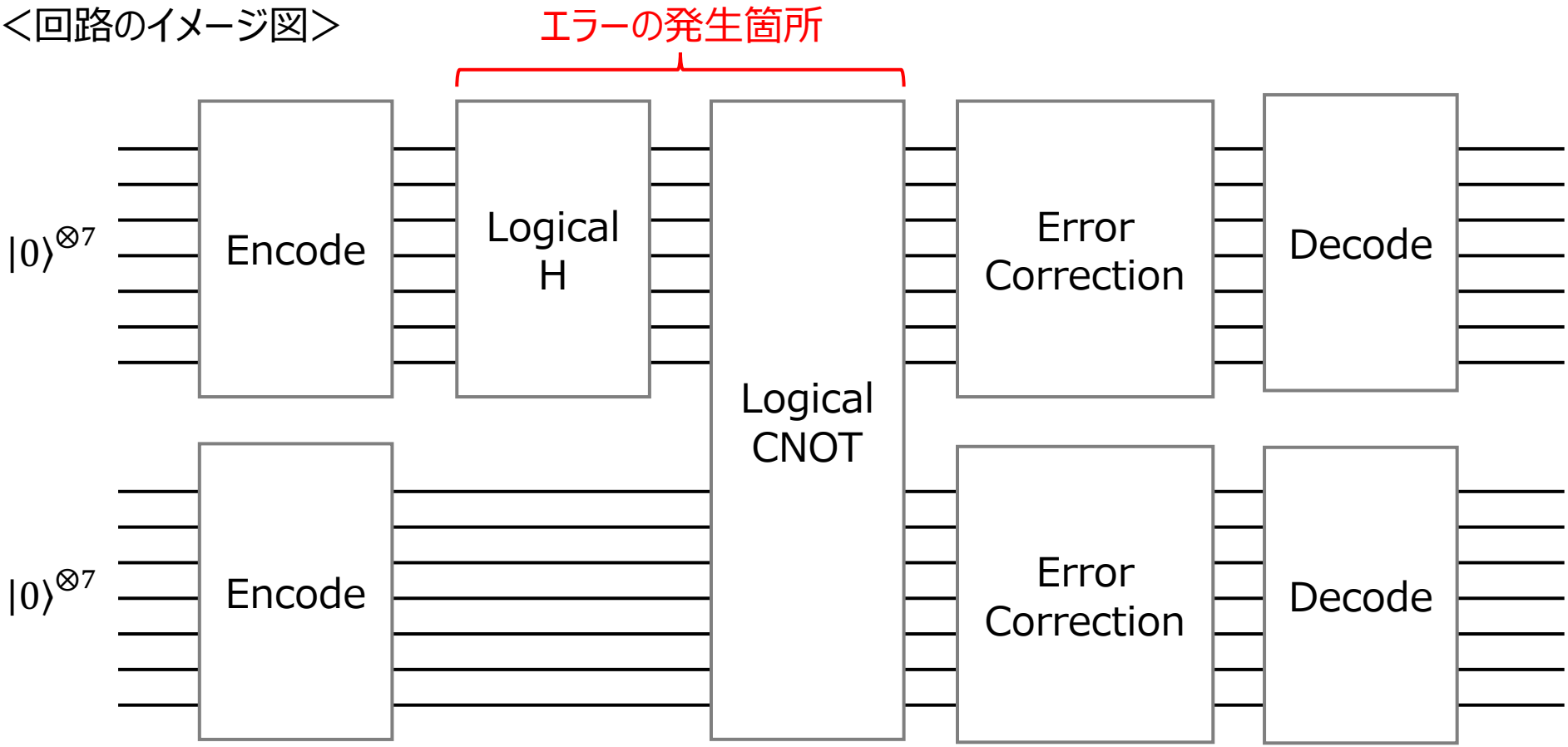
＜Steane符号における論理Hゲート＞



演習2.2. Steane符号による論理ビット操作

- 今回の論理ビット操作の演習では、2論理量子ビットを用いて、ベル状態の作成を実施してみます。
- エラー訂正は、アダマールゲートとCNOTゲート適用後に1回だけ行います。
- 実用上は、長いアルゴリズムの場合は、何度も途中でエラー訂正を行いながら実行します。

＜回路のイメージ図＞



（参考）Steane符号の基底とエラーシンドローム

- 今回用いたSteane符号では、符号化後、以下のような状態になっています。

$$|0_L\rangle = \frac{1}{2}\left(|0000000\rangle + |0011101\rangle + |0101011\rangle + |0110110\rangle + |1000111\rangle + |1011010\rangle + |1101100\rangle + |1110001\rangle\right)$$

$$|1_L\rangle = \frac{1}{2}\left(|1111111\rangle + |1100010\rangle + |1010100\rangle + |1001001\rangle + |0111000\rangle + |0100101\rangle + |0010011\rangle + |0001110\rangle\right)$$

7量子bitのSteane符号は、X,Y,Z,H,CNOTなどの論理ビット操作が、単純に実装できることが特徴です。

エラーシンドロームは以下のとおりです。

$$M_0 = X_0X_4X_5X_6 \qquad M_3 = Z_0Z_4Z_5Z_6$$
$$M_1 = X_1X_3X_5X_6 \qquad M_4 = Z_1Z_3Z_5Z_6$$
$$M_2 = X_2X_3X_4X_6 \qquad M_5 = Z_2Z_3Z_4Z_6$$

ディスカッション

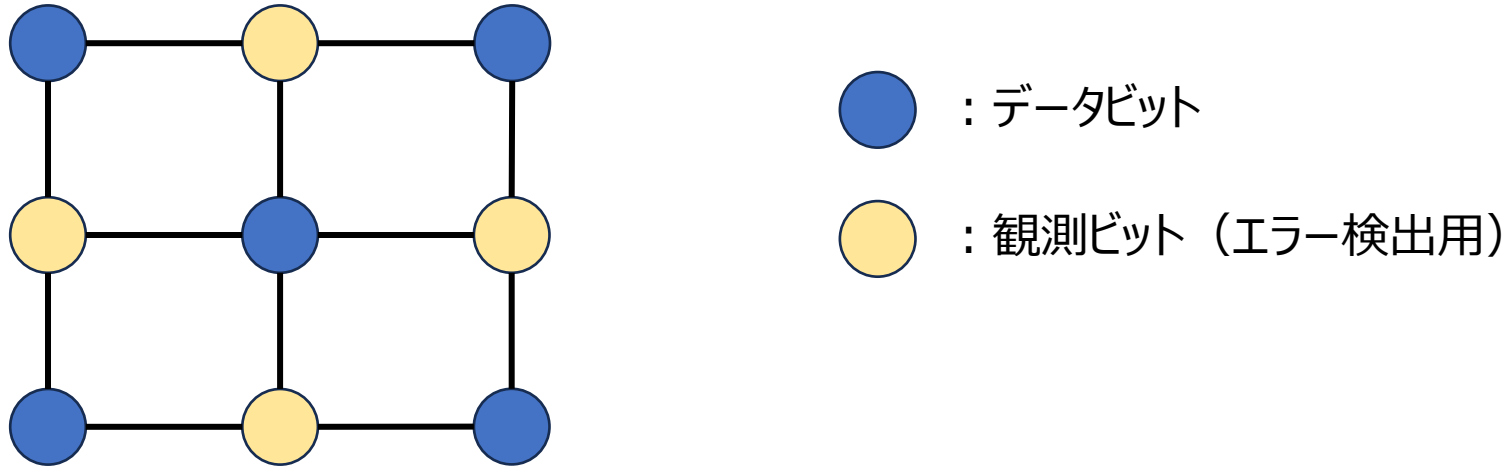
- Shorの符号、Steane符号の実験結果も踏まえて、改めて実際の誤り訂正に活用しようとした際に、課題となりそうな点、気をつけなければいけない点について、ディスカッションしてみてください。
- 実行時間が長いため、周囲の人と分担しながら色々なケースを試してみてください。

ディスカッションの進め方

1. 周囲の人と分担しながら、ノートブックの課題1,課題2を実行してみる
2. 演習の結果について共有する
3. 演習の結果や内容について意見を交換する

演習3. 表面符号のデモ

- ここでは、誤り訂正符号の実装方式の一つとして注目されている表面符号について、簡単なデモを行います。
- 表面符号は、正方格子のような見た目をしており、実機の形状との相性などから注目されています。
- 今回は、1ビットまでの誤りを検出できるようなものを実装してみます。



データビットに1ビット以下のエラーが発生した場合、観測ビットを測定することで、エラーの発生を検知できます。

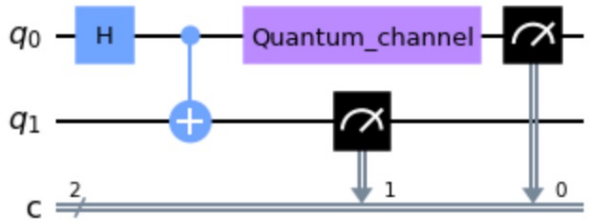
演習の内容 – 誤り低減

- 残りの時間では、誤り低減のアルゴリズムについて実装を試してみます。
- 今回は以下の三つの内容を実施します。

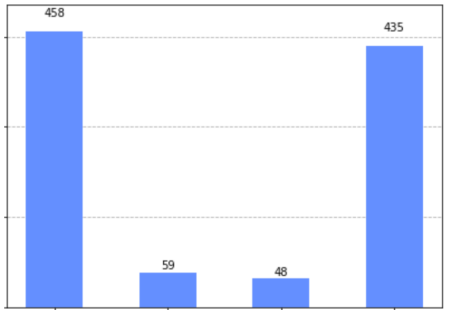
#	演習テーマ	概要
1	外挿法による誤り低減の基礎	• 非常に簡単な回路を用いて、外挿法（ZNE：Zero Noise Extrapolation)の仕組みについて学ぶ。
2	QPEに対する誤り低減	• 具体的なアルゴリズムの一例として、QPE(Quantum Phase Estimation)を題材に、外挿法を利用してみます。
3	実機での誤り低減	• IBM Quantum Platformを活用して、実機で誤り低減を行う方法について紹介します。

演習4. 外挿法による誤り低減の基礎

- NISQデバイスの活用方法の一つとして、誤り低減の活用が考えられています。
- 様々な手法が提案されていますが、基本的な考え方としては、量子回路の出力を古典コンピュータで後処理することで、誤りのない量子回路の出力を近似的に計算するものです。
- 誤り訂正と違い、利用する量子ビット数が増えないことが特徴です。

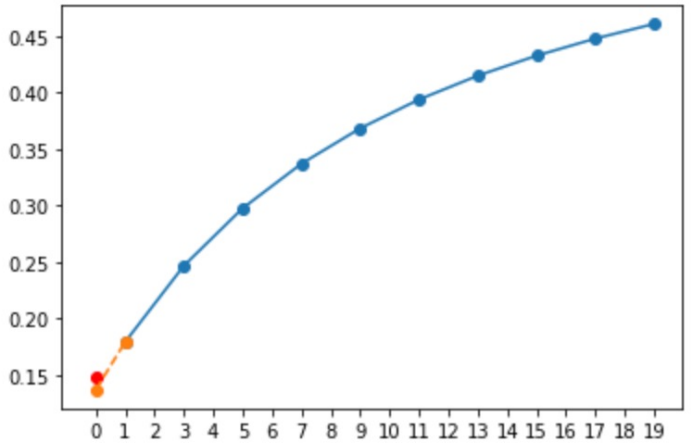


ノイズを含む量子回路



ノイズを含む観測結果

古典コンピュータによる
後処理



回路の出力の推定

演習4. IBM Quantum Platformを用いた実機での誤り低減

- IBM Quantum Platformでは、IBMの提供する量子コンピュータへのアクセスが可能。
- 無料でも毎月10分の利用枠が与えられるため、手軽に試すことができる。
  - 今回の教材のコードを利用すると、1分程度の利用枠を消費するため、実行する場合は気を付けること。