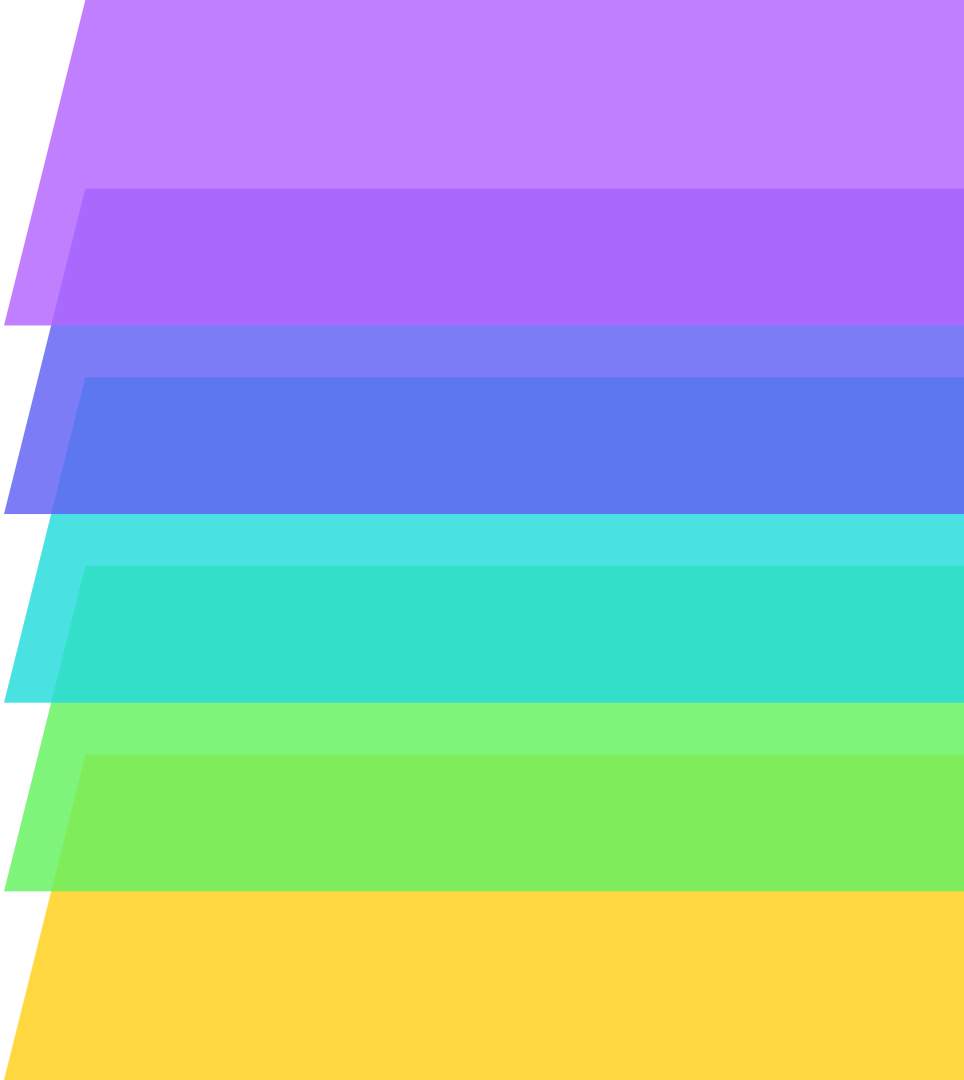
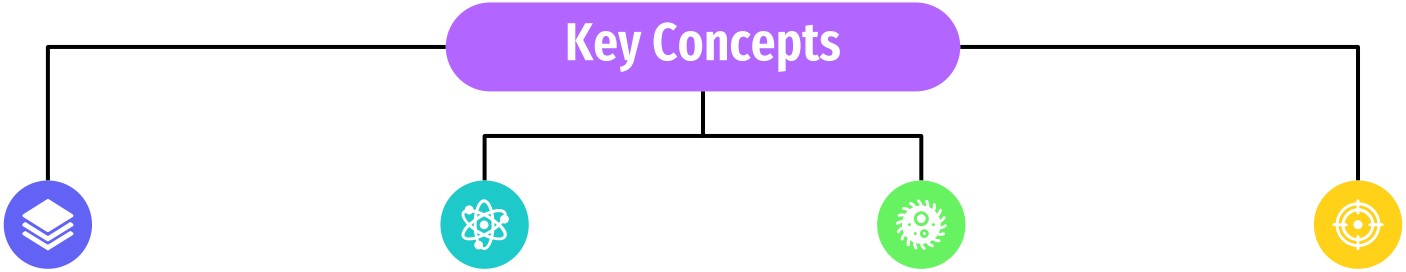
Quantum computation

Density Matrix vs Stabilizer Formalism



Quantum systems

A quantum system can be made using different technologies, but regardless of this they are governed by many universal postulates



Hilbert Space

- A system can be modelled by a complex Hilbert Space
- State described by state vector up to a global phase in the system state space
- Superposition

Measurement

- Basis dependent
- Probabilistic
- Transform system state in one of the preferred state

Multi-qubit

- Systems combine by tensor product
- Exponential growth of space state
- Can still exists in superposition
- Entanglement

Transformations

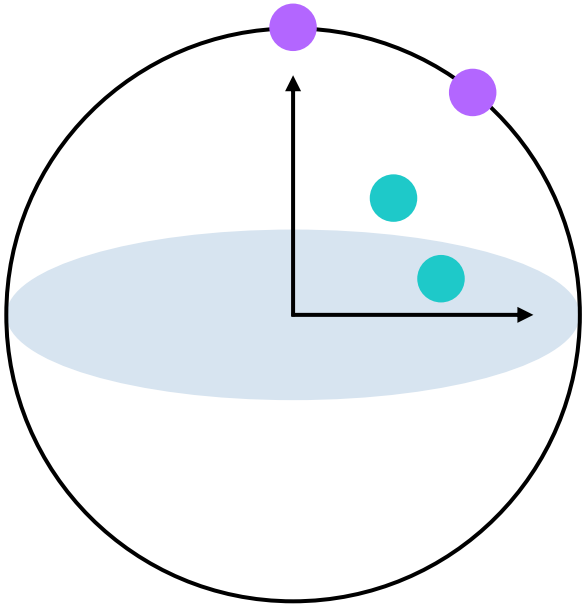
- Isolated quantum system evolves with reversible unitary operators
- Open systems affected by decoherence
- No cloning theorem

Most Common technologies

Optical		Superconductive
Use photons as qubit		Use superconductive circuits manipulated by microwaves
Polarization, …		System phase, Charge, …
Coaxial or fiber technology		Need system at cryogenics temperature
Low Decoherence, Slow gates		High Decoherence, Fast gates

Bloch Sphere

Qubit: geometric definition



Pure states

Represented by unitary vector that always lives on Bloch Sphere's surface

Mixed states

Represent a probabilistic ensemble of pure state due to uncertainty about system's state, where graphically it's a non unitary vector contained inside the Bloch Sphere. Usually caused by decoherence, that 'mixes' pure states or uncertainty about system initialization

Common quantum representation

Pure states

Ket

- Denotes column vector
- Coefficients are the amplitude factor
- Only pure states

$$|\varphi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle$$
$$\alpha_0, \alpha_1 \in \mathbb{C}$$

- α_0, α_0: probability amplitude
- $|\alpha_0|^2$: probability of measuring $|0\rangle$
- $|\alpha_1|^2$: probability of measuring $|1\rangle$
- $|\alpha_0|^2 + |\alpha_1|^2 = 1$

$$\frac{\sqrt{2}}{2}|0\rangle + \frac{\sqrt{2}}{2}|1\rangle$$

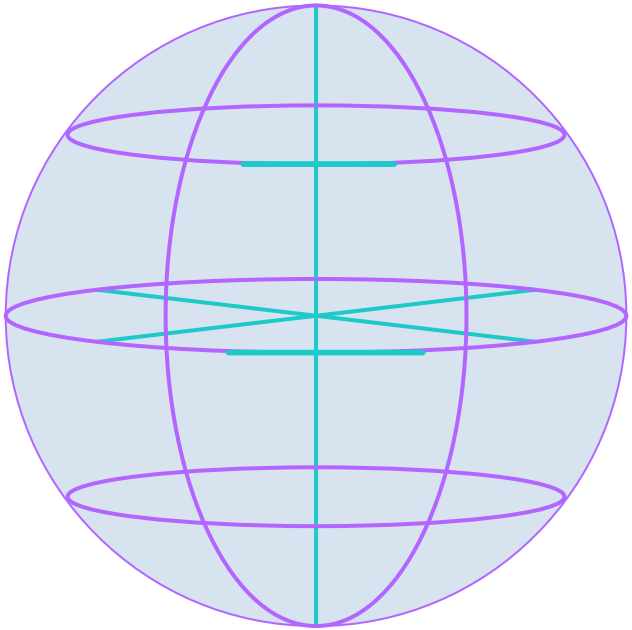
General

Density Matrix

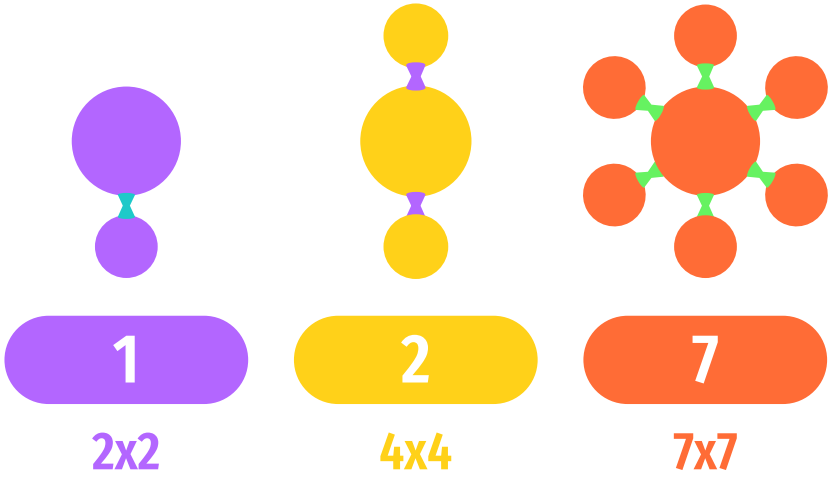
- Matrix representation
- On main diagonal there are classic probabilities of pure states
- On secondary diagonal there are coherence elements
- Universal representation (isolated closed system)

$$\rho = \sum_i p_i |\psi_i\rangle \langle\psi_i|$$

$$\begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$



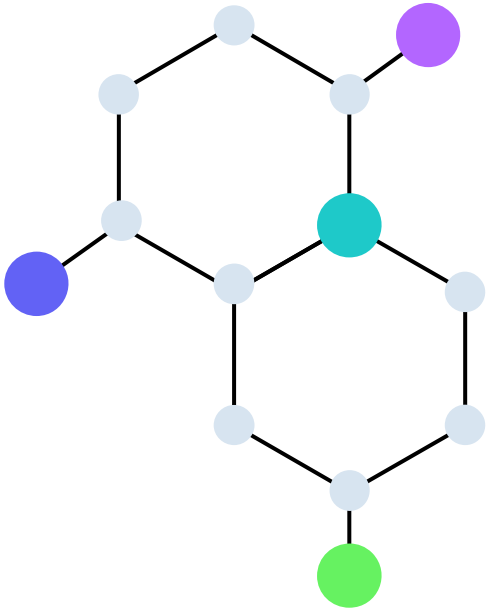
Density matrix exponential growth



As seen previously, quantum system combines by tensor product and the density matrix representation uses matrix of dimension $2^n x 2^n$.

This imply an enormous computational cost for every kind of process that we would simulate, especially on classical computers, that would allow to only simulate systems with just a few qubits.

Stabilizer formalism



Stabilizer

Allows to represent a quantum state (or space state) by its stabilizer.
A stabilizer (S) is an operator's subgroup of the Pauli group on n qubits, that applied on a quantum state (or a space state) doesn't change it

$$G_1 = \{\pm I, \pm iI, \pm X, \pm iX, \pm Z, \pm iZ, \pm Y, \pm iY\}$$

Why

Now we can uniquely represent a stabilizer state by its stabilizer's generator ($\log_2(|n|)$), instead of using an $2^n x 2^n$ matrix, also if only for pure state or uniform mixed states.
Furthermore, we can track the evolution of the system due to Clifford gates, directly in this representation

U
Clifford gate

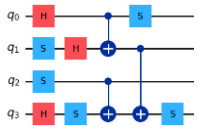
$$\mathcal{S}' = \{USU^\dagger \mid S \in \mathcal{S}\}$$

Stabilizer State

Equivalently, a state is called a stabilizer state of S, if is stabilized by it.
We can also view it as a state that can be generated by a stabilizer (Clifford) circuit (only CNOT, H, S)

$$X_1 X_2 |\Phi\rangle^+ = |\Phi\rangle^+$$

$$\rho_{\mathcal{S}} = |\psi\rangle\langle\psi| = \frac{1}{2^n}\sum_{M\in\mathcal{S}} M.$$

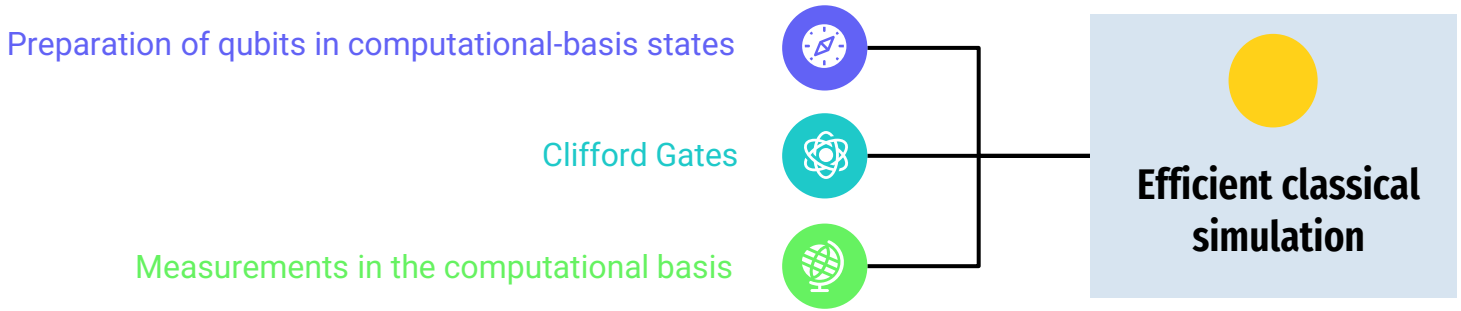


Gain

As we can directly see, we obtain a big gain in terms of required memory.
But now, thanks to many researches we've also reached a big improvement in terms of simulation on classical computers (for Clifford circuits) passing from exponential to polynomial time

Gottesman–Knill theorem

A quantum circuit using only the following elements can be simulated efficiently on a classical computer by a polynomial-time algorithm:



In this case, classical simulation will just track, at each point in time, of a list of generators for the current state's stabilizer group, updating it whenever a Clifford gate is applied.

Gottesman–Knill algorithm

How it works

First of all, starting with a stabilizer state we need to get its stabilizer representation (the generators list).
Now we write them with the **Tableau Representation**, that use 2 $n_x n$ matrix of 0s and 1s, where each row represent a generator.

The X Matrix and The Z Matrix

+(0000 | 1000)		ZIII
+(0000 | 0100)		IZII
+(0000 | 0010)		IIZI
+(0000 | 0001)		IIIZ
↑	↑	
1 if X or Y	1 if Z or Y	
0 otherwise	0 otherwise	

|0000>

{ZIII, IZII, IIZI, IIIZ}

After that, we just need to **apply the rules** to update the state when a gate its applied

Time

Updates corresponding **unitary gates** require **O(n)** for gate.

But for **measurement gates**, a random outcome require **$O(n^2)$** while a deterministic outcome require **$O(n^3)$**

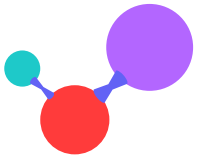
Memory

n generators, each one takes **2n +1 bits**:
2 bits for each of the n Pauli matrices, plus 1 additional bit for the ± sign.

So, **total number of bits** is n(2n+1) = **$O(n^2)$**, while writing out the entire amplitude vector, would have taken ~2^n bits,

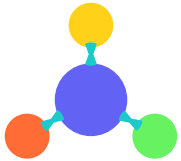
Gottesman–Knill algorithm

Update rules



H

To apply H on i^{th}, swap i^{th} column of X matrix with i^{th} column of Z matrix



S

Bitwise XOR the i^{th} column of the X matrix into the i^{th} column of the Z matrix

CNOT

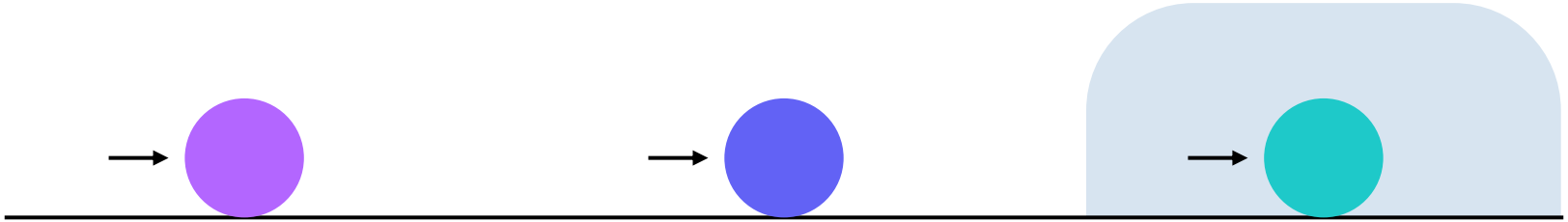
Bitwise XOR the i^{th} [j^{th}] column of the X [Z] matrix into the j^{th} [i^{th}]column of the X [Z] matrix

MEASUREMENTS

To measure the i^{th} qubit, first check if the i^{th} column of the X matrix is all zeros:
- yes, the result is deterministic based on the i^{th} row of the Z matrix
- no, the result is random, then pick a row where the i^{th} entry of the X matrix is 1 and reduce other rows using bitwise XOR (**Gaussian elimination**), and update the Z matrix accordingly.

Finally, if the measurement result is +1, add it to the stabilizer; if -1, flip the sign of the corresponding stabilizer generator

Further improvements



Gottesman-Knill algorithm

- $O(n^2)$ memory
- $O(n)$ unitary gates
- $O(n^2)$ random measurements
- $O(n^3)$ deterministic measurements

CHP

- $2 * O(n^2)=O(n^2)$ memory
- $O(n)$ unitary gates
- $O(n^2)$ all measurements

remove Gaussian elimination

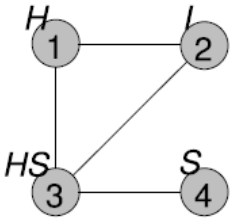
GraphSim

- $O(n\log n)$ memory
- $O(1)$ single qubit gates
- $O(\log n)$ Z base measurements
- $O(\log n^2)$ X, Y basis measurements

Use graph representation
(generic times)

Graph Representation

	1	2	3	4
+	Z	Z	X	I
+	X	X	X	I
−	X	Z	Y	Z
+	I	I	X	Y



Vertex	VOP	adjacency list
1	10	2, 3
2	0	1, 3
3	17	1, 2, 4
4	6	3

Really interesting fact about graph state topology of GHZ here.

Graph

A graph is a mathematical structure of vertices and edges.
In this case, the vertex are the qubits while edges represents entanglements or interactions

Example

On the left are shown three equivalent representation of the state 1/√2 (|0000> + |1111>), respectively stabilizer representation, graph representation and adjacency list of the graph representation (mathematical form of a graph).

In graph representation, on every qubit appears the gates to apply over them and connections indicates the entanglement, while in the adjacency list the operations are labelled with VOP code (1 to 24) while connections are a list in every qubit

Stabilizer – Graph

Every graph state is a stabilizer state, but also every stabilizer state is equivalent to a graph state in the following sense: **any stabilizer state can be transformed to a graph state** by applying a tensor product of local Clifford (LC) operations (vertex operations VOPs).
So, we can say that a stabilizer state is always **Local-Clifford equivalent*** (LC-equivalent) to its respective graph state representation.

We can always choose what representation use by the problem we're facing, where this will be shown that allows to further improve stabilizers circuit simulations

* equivalent ≠ is a graph state

Implementation

Similarly to what seen previously, also here the key to update system's state is following specific rules (not mentioned here)

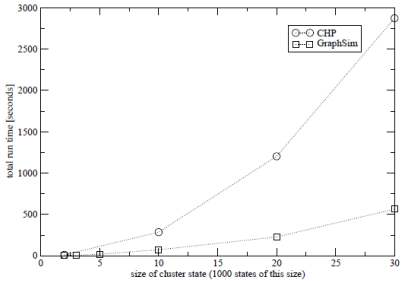
Time

The transformation from stabilizer to graph can be done with a sort of Gaussian elimination $O(n^3)$ from Tableau representation, so we want to avoid it when possible.
This approach allows us to reach:

- Single qubit gates requires just a **lookup in a table** $O(1)$
- Z measurement requires to **remove the edges** of the measured vertex a $O(d)$
- Y, X measurements require **local complementation** $O(d^2)$
- Two qubit gates require **5 local complementation** $O(d^2)$

memory

Tableau matrix requires $O(n^2)$ memory, while a graph representation that uses adjacent list took $O(nd)$ where **d is the average number of neighbours** + a $O(n)$ list of the VOPs (numbered from 1 to 24) that take us to stabilizer state.
Usually the neighbours are O(logn), but in worst case are $O(n)$ and in this case we have no improvements in confront of CHP



Entanglement purification benchmark

Use cases

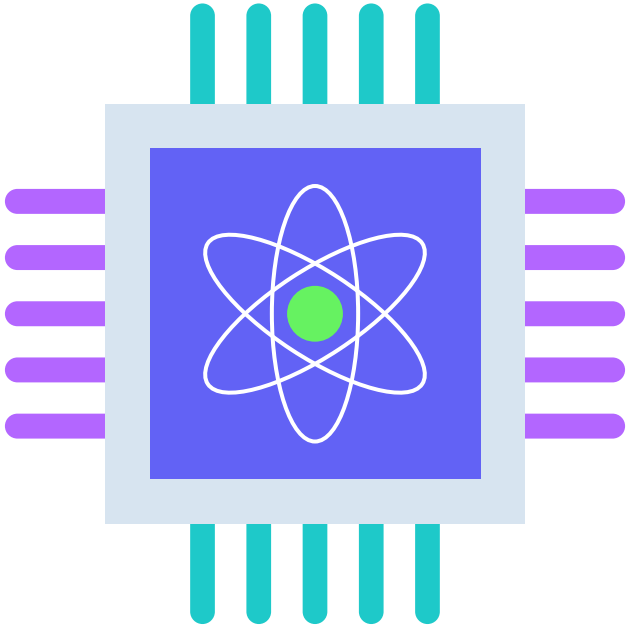
Density matrix vs Stabilizer

Mixed State

Non-Pure states, partial trace of an entangled system…

Open Quantum Systems

Decoherence apply non unitary transformation of a system taking it to a mixed state



Clifford Circuits

Efficient simulation for stabilizer states and Clifford circuits

Quantum Error Correction

It offers a structured approach to designing and analyzing quantum error-correcting codes that simplify calculations and reasoning.
At cost of approximating quantum errors as Pauli Gates
(not too far from common cases)
and encoding / decoding must be done by Clifford circuits

Allows entanglement and state distillation, …

Notebook Epilogue

Representation

3 qubit GHZ state $|\mathrm{GHZ}\rangle = \frac{|000\rangle + |111\rangle}{\sqrt{2}}$

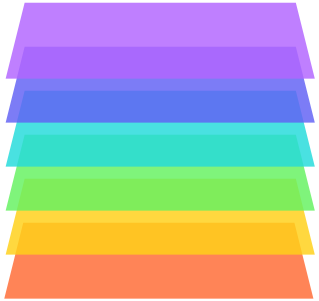
$$\begin{bmatrix} \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \end{bmatrix}$$ vs [+XXX, +IZZ, +ZIZ]

Mixed states

$$\begin{bmatrix} \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{3}{4} \end{bmatrix}$$

```
# Try stabilizer rapresentation
try:
    stabilizer_state = StabilizerState(mixed_state.to_statevector())
except Exception as e:
    print(f"Conversion to StabilizerState failed: {e}")
```

```
Conversion to StabilizerState failed: 'Density matrix is not a pure state'
```



Circuit simulation

60 qubit benchmark

```
error during density matrix simulation
ERROR:  [Experiment 0] a circuit requires more memory than max_memory_mb.
```

```
Result(backend_name='aer_simulator_stabilizer',
Method used: stabilizer
Total time: 0.21551871299743652
```

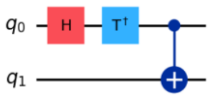
1000 qubit, depth 800 benchmark (~2 min.)

```
Method used: stabilizer
Total time: 116.5292661190033
```

Process tomography

Simulators with method stabilizer currently not available for process tomography

Non-Clifford Circuit simulation

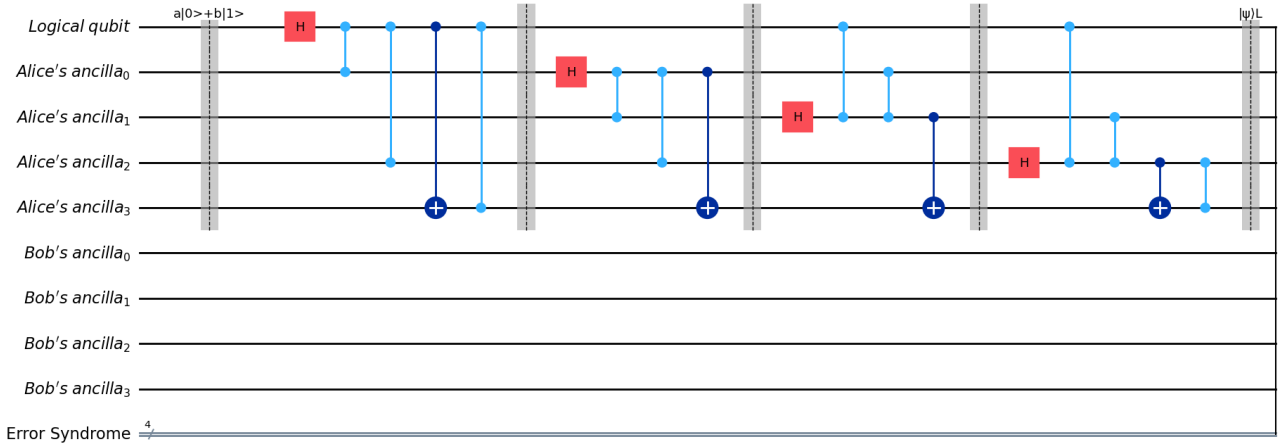


$$\begin{bmatrix} \frac{1}{2} & 0 & 0 & \frac{\sqrt{2}}{4} + \frac{\sqrt{2}i}{4} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{\sqrt{2}}{4} - \frac{\sqrt{2}i}{4} & 0 & 0 & \frac{1}{2} \end{bmatrix}$$

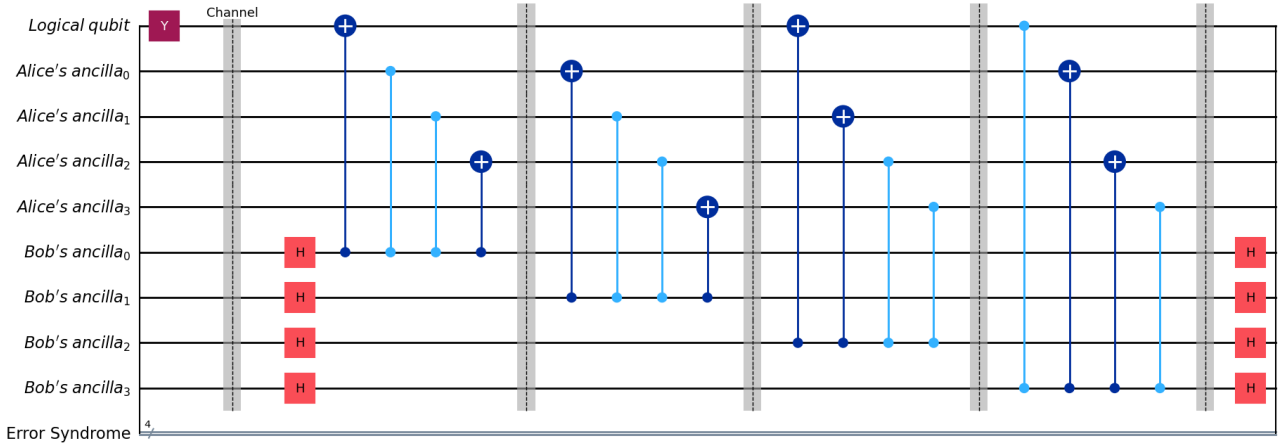
```
try:
    StabilizerState(qc_non_clifford)
except Exception as e:
    print(f"Error using stabilizer formalism: {e}")
```

```
Error using stabilizer formalism: 'Cannot update Clifford with non-Clifford gate tdg'
```

Quantum error correction



5 qubit QECC example




```
{'1011': 100}
Error found: Y1
Total time: 0.0029637813568115234
```

Sources

GitHub Repo

Other

Wikipedia,
StackExchange,
...

15%

40%

Course material

Improved Simulation of Stabilizer Circuits

10%

20%

Scott Aaranson Lecture

15%

Fast simulation of stabilizer circuits using a graph state representation