# Project: Quantum Galton Box

Summary by: Samradh Bhardwaj

Quantum Fourier Transform was one of the first applications of quantum computers, demonstrating speed up over classical computations. This project uses QFT at its core. Quantum Galton Box is an quantum implementation of classical Galton box, which can simultaneously explore all $2^N$ paths. Generalizing the Quantum Galton Box can be used as universal statistical simulator, solving high dimensional problems ranging from particle transport to complex quantum systems.

## *What's the working logic?*

In an Galton box, each peg has the equal chance of changing the ball path - to either right, or either left. A Galton Box has the structure of pascal's triangle. Following the [1] paper,

- each peg requires 3 working qubits, and 1 control qubit.
- total qubits required for an '*n*' level Galton Board, 2n + 2 (one for ball, and one is control qubit).
- we mark ball qubit, at position 'n+1' for even levels & at position 'n+2' for odd levels, by X gate.

We start by using 1st qubit as control qubit. Then we mark the middlemost qubit as ball, with X gate. When then create a single peg using the logic demonstrated below.

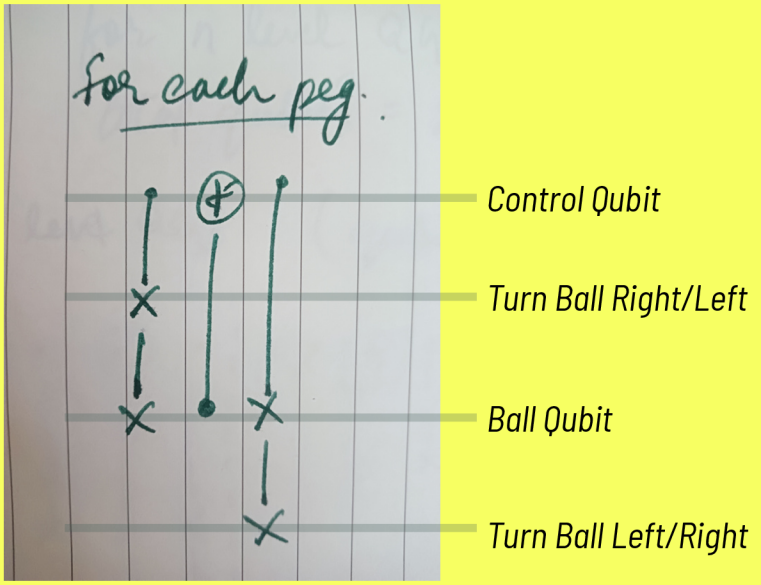


***Fig. 1 Quantum Circuit for individual Peg***

To proceed to next level of qgb, we reset the control qubit, and put it into 50/50 superposition (in case of bias, we use Rx gate). We then add the 1st peg of second layer using the same logic. This time, the qubit above the previous ball qubit becomes the ball qubit. So, if in previous level, if your ball qubit was at 'b' position, in current level, your ball qubit moves to 'b-1' position.

After the first level, we require multiple pegs. We maintain the control superposition by applying cnot gate: ball qubit as control qubit, and control qubit (1st qubit) as target qubit. This helps us reset the control qubit for subsequent peg. We then add another peg, this time using qubit n+1 as ball qubit. See (Fig 2.)

Using this logic, we build the function [*n_qgb(n_level)*], in file *'submission.ipynb', to generalize the Quantum Galton Board to n_level circuits.*

We now have a Universal Statistical Simulator. We can solve many problems using this one template, by modifying its initial states, biases, and bias gate application positions.

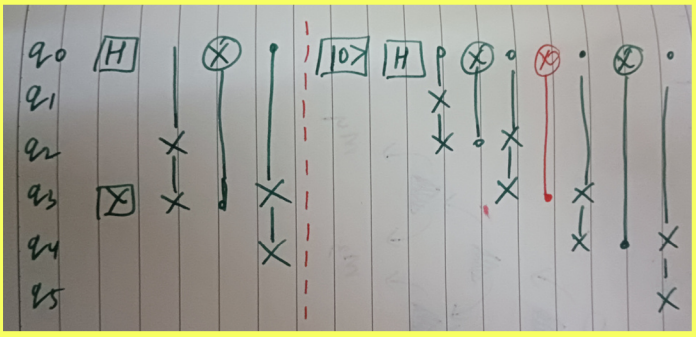


***Fig. 2 Quantum Circuit for 2nd Peg***

# *What have I specifically worked on?*

My main focus in this project has been on bridging the gap between measurement on real noisy hardware, and expected values when simulated on noiseless simulator. I have combined multiple post processing steps to remove the errors, and to increase the fidelity between expected, and measured distributions.

For correcting the measurement errors, I have taken a project specific approach here. Since our measured bitstrings will only contain one '1', We can eliminate noisy measured bitstrings with more than one '1'. We then can apply measurement error mitigation using "MThree" qiskit module. We then group the previously removed bitstrings under the groups of expectation values, by calculating the distance between bitstring and candidate expected bitstrings, by using bit position based distance method.

The Approach that I defined above may sound strange, but it works in action. Through this, I have been able to achieve fidelities such as 0.90, and 0.98.

# *Implementation Structure*

*FILE: submission.ipynb*

1. n_level quantum galton board
2. circuit for gaussian distribution
3. circuit for exponential distribution
4. circuit for hadamard quantum walk
5. running the same circuits first on noiseless simulator, and then on noisy real hardware simulator
6. at last we check the fidelities between measured distribution, and expected distribution

# *Reading and Related Resources*

[1] *https://arxiv.org/pdf/2202.01735*
[2] *https://github.com/Qiskit/qiskit-addon-mthree*
[3] *https://github.com/Qiskit/textbook/blob/main/notebooks/quantum-hardware/measurement-error-mitigation.ipynb*

*Thank you*