Project 1
Quantum Walks and Monte Carlo

2025 Womanium & WISER Quantum program

Adele Mehrafrouz

1. Developed Fully Generalized Quantum Galton Board (QGB) Circuits

Technical Detail:

- Layers = number of pegs (depth of board)
- Each peg gate:
    - Hadamard gate → $\theta = \pi/2$ (for fair split)
    - Rx(θ) gate → $\theta \neq \pi/2$ (for bias)
- The ball qubit is routed conditionally at each layer
- Measurement of final register gives "slot" count

- I wrote a **modular function** to generate QGB circuits with arbitrary layer depth and configurable gate bias.

Where in Code:

- quantum_peg_multi.py: contains generate_qgb(layers, theta)
- Uses controlled-SWAP logic and either Hadamard or Rx(θ) gates

2.Gaussian Output from Unbiased QGB

Goal:
Simulate a **fair Galton Box**, where a ball has 50/50 chance at each peg → should produce a **Gaussian-like distribution**at the output.

Implementation:

- Set θ = π/2 (use Hadamard gates)
- Simulate with many shots (2048 shots)
- Process measurement results into slot histogram

Where in Code:

- quantum_peg_multi.py + run_all_multi.ipynb
- `verify_gaussian.py` confirms the Gaussian shape
- Output saved to: results/qgb_gaussian_vs_biased.png

3. Rx-biased QGB → Exponential-like Output

Goal:

By rotating with Rx(θ) where θ ≠ π/2 (e.g., 2π/3), the board becomes **biased**, so more balls go one way → leading to **skewed (exponential-like)** distributions.

Implementation:

- Same QGB logic, but θ is set to a value like 2π/3
- compare results with unbiased case

Where in Code:

- biased_distribution_multi.py
- run_all_multi.ipynb
- Visualization in same plot as unbiased

4. Distance Metrics: TVD & KL Divergence

Problem:

We needed to **quantify** how similar or different two distributions are (e.g., Gaussian vs biased, noisy vs ideal).

Implementation:
I implemented two standard metrics:

- **TVD (Total Variation Distance):** max difference in probabilities
- **KL Divergence**: how much information is lost moving from true to estimated distribution

Where in Code:

- distance_metrics_multi.py contains both functions

```
total_variation_distance(dist1, dist2)
kl_divergence(dist1, dist2)
```

Used in run_all_multi.ipynb and noise_vs_ideal.ipynb

5. Hadamard Quantum Walk Circuit

Goal:

Simulate a **quantum walk**, where quantum interference causes a walker to behave differently than in a classical random walk.

Implementation:

- Built a circuit using **Hadamard gates** and **controlled shifts**
- The position register stores the walker's position
- Unlike a classical random walk, the output is uneven and shows distinct peaks and dips, a result of quantum interference between possible paths.

Where in Code:

- hadamard_walk_qiskit.py
- Called in run_all_multi.ipynb
- Output saved in: results/hadamard_walk_qiskit.png

6. Noise Simulations with Qiskit Aer

Goal:
Test how **hardware noise** affects simulation output.

Implementation:

- Used Qiskit AerSimulator with:
  - Depolarizing error (1-3 qubit gates)
  - **Readout error** (measurement error)
- Compared "ideal" and "noisy" distributions
- Re-ran distance metrics to measure deviation

Where in Code:

- noise_vs_ideal.ipynb
- Uses:

```
from qiskit_aer.noise import NoiseModel, depolarizing_error, ReadoutError
```

- Output plot: results/noisy_comparison.png