Implementing quantum Galton boards

1. Purpose and scope

This summary explains how to design and implement quantum Galton boards (QGBs) that emulate a Galton box and its Monte Carlo sampling behavior. A QGB produces samples distributed as $Bin(n, p)$ with n rows and left-right bias p, and supports per-row biases for shaped distributions. Circuits are modular and hardware-aware, and their output matches binomial predictions, approaching a Gaussian for large n.

2. Core idea

A QGB routes a single logical ball through n rows of pegs. Each peg performs a coin toss on a reusable coin qubit and conditionally moves a one-hot token along a channel register. After n pegs, exactly one of the $n + 1$ channel wires is 1 (one-hot), encoding the final column index $k \in 0, \ldots, n$. Measuring the channel register yields one Monte Carlo sample.

2.1. Classical reference

For a classical n-row Galton box with per-row success probability p,

$$P(K = k) = \binom{n}{k} p^k (1 - p)^{n-k},$$

so for $p = 0.5$ one has $P(K = k) = 2^{-n}\binom{n}{k}$ and the histogram approaches a normal curve as n grows.

3. Circuit primitives

3.1. Qubits and registers

- Coin qubit: $q[0]$, reused every layer.
- Data rails: $q[1] \ldots q[2L + 1]$ for a total of $2(L + 1)$ qubits including the coin. Odd indices $q[1], q[3], \ldots, q[2L + 1]$ are the bin qubits that are finally measured. Even indices are auxiliary routing rails.
- Mid-circuit reset: the coin is reset after each layer except the last.

3.2. Required operations

- Coin toss per layer: apply H to $q[0]$.
- Routing sweep per layer $l = 1 \dots L$: define start $= (L+1) - l$ and end $= (L+1) + (l-1)$. For each integer k from start to end do, in order:
  1. $CSWAP(q[0];\ q[k],\ q[k+1])$.
  2. $CX(q[k+1] \rightarrow q[0])$, except skip this CX only in the special case $L = 1$ and $k =$ end.
- Coin recycle: if $l < L$, reset $q[0]$ to 0.

4. A single layer l

Assume exactly one of $q[1] \dots q[2L+1]$ holds 1 at layer entry (at $l = 1$ it is initialized at $q[L+1]$).

1. Apply H on $q[0]$.
2. For $k =$ start, …, end, apply $CSWAP(q[0];\ q[k],\ q[k+1])$ then $CX(q[k+1] \rightarrow q[0])$ (with the single skip noted above).
3. If $l < L$, reset $q[0]$.

5. Building an n-row QGB

5.1. Initialization

Set $q[0] = 0$. Prepare the walker at the center by setting $q[L+1] = 1$ and all other $q[i] = 0$.

5.2. Row loop

For $l = 1, \dots, L$ execute the layer steps exactly as listed above. Insert a barrier after each layer if desired for readability.

After the loop, measure only the bin qubits $q[1], q[3], \dots, q[2L+1]$ into $L+1$ classical bits. Decode the one-hot position to a bin index $k \in 0, \dots, L$. If the backend returns little-endian bitstrings, interpret the rightmost measured bit as $k = 0$.