Quantum Evolution with Measurement and Reset

By: Team JQC

Abstract

The iterative evolution of quantum systems under the influence of measurement and subsequent reset operations presents a fascinating challenge in quantum control. Traditionally, such processes necessitate sequential measurement and re-preparation, hindering their direct implementation on coherent quantum hardware**.** We propose two solutions that effectively synthesize the conditional evolution of a two-qubit system undergoing repeated measurement-reset cycles, *without* performing actual intermediate measurements. One solution has O(1) qubits complexity and the other has O(n) qubit complexity. Our circuit prepares the final state on the primary qubits, conditioned on the entire history of the simulated feedback loop. We draw an analogy to high-frequency trading (HFT) loops, where rapid decisions based on market measurements feed back into subsequent trading strategies, highlighting the system's ability to model complex feedback-driven dynamics in a purely unitary fashion. This work paves the way for efficient simulation of open quantum systems and classical feedback mechanisms on future quantum computers.

1. Introduction and Design Objective

In HFQT simulation, we have a U unitary acting on three qubits. One acts as a signal {s0} and the other two as internal strategy or market sentiment, let them be {q1, q2}. The |q1q2> state tells us whether to buy or sel, based on that calibration is done. The state so achieved can be used for next discussion cycle. This loop is continuous.

Market signal -> Strategy Reset -> Trade execution -> New Market State -> Market Signal -> …

Traditionally, simulating such a process on a quantum computer would be a painful exercise in "measure, reset, repeat" – a bit like repeatedly hitting refresh on your trading screen and hoping for the best. This breaks coherence and makes large-scale simulation impractical. Our goal, therefore, is to create a quantum circuit that can look at the *entire history* of these hypothetical measurements and resets to 0 and magically produce the final state as if all those pesky intermediate steps actually happened, but without ever *really* measuring.

So given a list of signal states {y0, … ,yn}, our objective is to design a quantum circuit that prepares the final state of two qubit system (q1, q2) that has undergone sequence of U operations and hypothetical state measurement of signal to yi with signal reset to 0. By this we recreate the same state based on feedbacks without doing repetitive measurements.

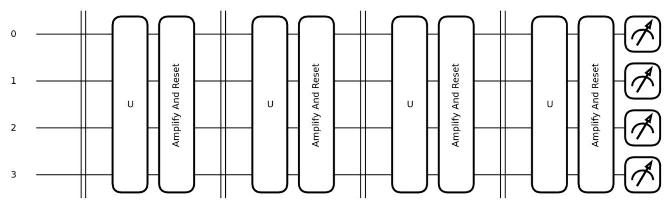
2. Proposed Solution and Circuit Design

We have two proposed circuit designs. Section A will follow the circuit design having O(1) qubit complexity and the other will be in Section B having O(n) qubit complexity. Since, each additional qubit in a quantum system introduces more opportunities for errors due to noise and decoherence, hence we have done more in-depth analysis of the first approach as that is more qubit efficient and comparatively less error prone while running in QPU.

2.1 SectionA: Solution with O(1) qubit complexity

2.1.1 Circuit Construction and Logic

In this proposed solution we use 3 qubits { s0, q1, q2 } with one auxiliary qubit, using which we amplify the probability of yi signal on s0.



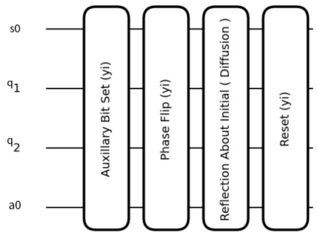


Figure: 1 Figure: 2

In figure 1, We have the circuit where we are applying U and then Amplify and Reset logic. We apply a series of n such operations. In each loop, we give input yi to Amplify and Reset logic, using which we manipulate the amplitude of the s0 qubit, mimicking the measure.

The amplify and reset logic (ARB) is expanded in figure 2, where we first set the auxiliary bit based on the yi. Next two blocks are similar to what we use in Grovers, where we are trying to amplify the state that we want the signal bit to have.

During the course of the gates, the two qubits, q1, q2 are not dis-entangeled and left untouched from all the operations, except in Reflection About Initial stage where we do U^{-1} and U operation back to get the state back again, hence not disturbing{ q1, q2 } entanglement.. After this stage we get the yi with probability 1 at s0, hence effectively mimicking measurement. In reset stage we set back the bit to 0 if yi was 1 and reset a0 to 0 by applying H.
Probability of measuring yi at ith loop after U and Amply Reset logic is approximately equal to 1, hence the state entangled to s0 at yi is also amplified. Subsequent series of n such will result in a consistent { q1, q2 } value, as the probability is 1 at each ith loop. Since we are using state flip , Grovers to effectively increase the P(yi)≈1 without breaking the entanglement of { q1, q2 }, the result remains consistent.

The circuit creation is controlled by classical analysis. Based on yi value we add X gate to control operation.
Figure3.1 is for yi=1 and Figure 3.2 is for yi=0 . Section 1 is implementing U. When yi=0, we use X on the q3, in section 2 and then we phase flip in section 3. Reflection phase i.e section 4 is independent of yi value and then we finally reset the value in section 5. We have used q3 as ancilla bit, which does the magic for us. |q3> is entangled with |s0> at section 2. When we apply reflection at section4 we basically disentangle |s0q3> and expand the vector space, and on U operation, we collapse to the state that we want. The relative state we create at section 2 is the secret recipe.

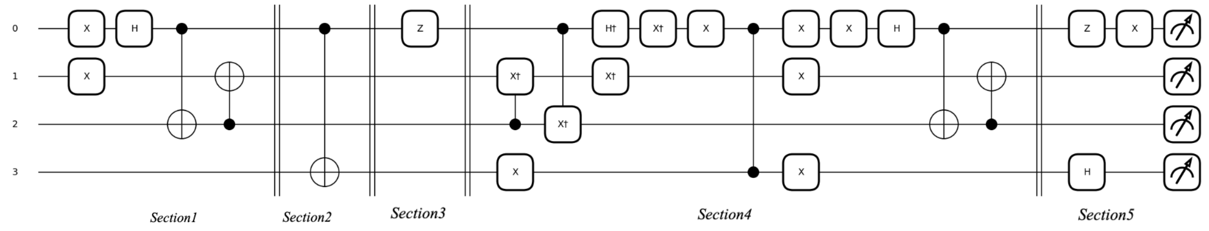


Figure 3.1

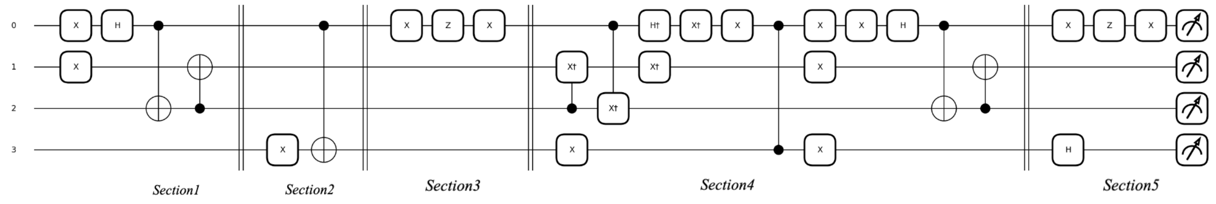


Figure 3.2

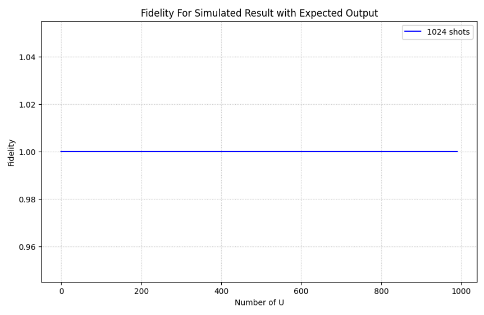


Figure 4.1

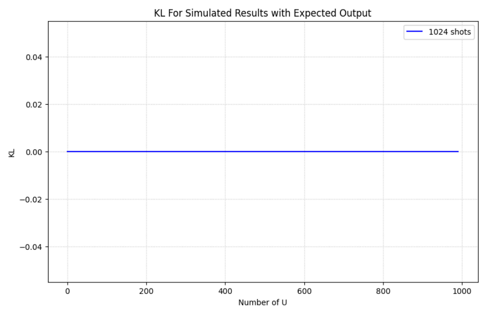


Figure 4.2

We ran the designed circuit for 1024 shots using local simulation and AWS SV1. We got the fidelity of 1 with the expected output. Figure 4.1 shows the fidelity we got and Figure 4.2 shows KL value. Figure 5.1 and Figure 5.2 has snapshot of the expected statevector with what we got after simulating the circuit.

This proves our circuit is able to get the same output after n U operations without doing any actual mid measurements. Figure4.1 shows result for n=200, while Figure4.2 shows result for n=1000

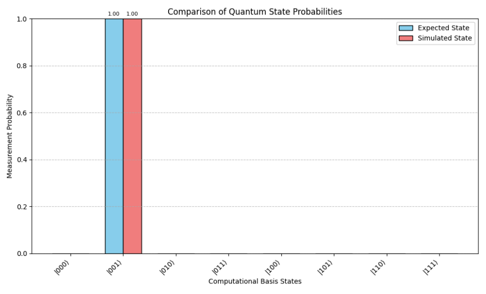


Figure 5.2

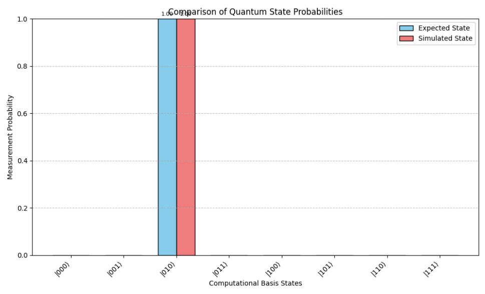


Figure 5.2

2.1.2 Performance

The fidelity of our circuit when we simulate it is 1 with the expected output. The KL is 0. We tested the same with n=1500 runs for 3 qubits and got the same results. *Figure4* and *Figure5* shows the result for local simulation.
We timed our **simulation with AWS SV1**, and we were able to complete the simulation for n=2000 in 1.75 seconds. More over, we were able to simulate the same locally in 4min (2vCPU, 12GB). *Figure6* shows the graph for $n \epsilon \{1 ... 2000\}$

We also analysed the **Gate Count and Circuit Depth** of our circuit and based lined it with U^n Gate Count requirements. *Figure7.1* and *Figure7.2* shows us there is linear increase in the gate count with n.

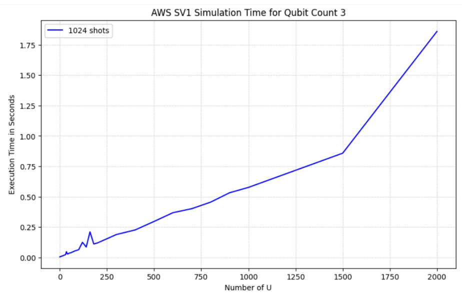


Figure 9.2

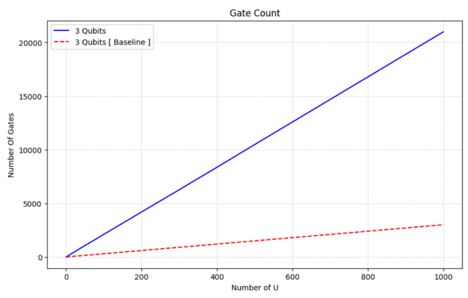


Figure 7.1

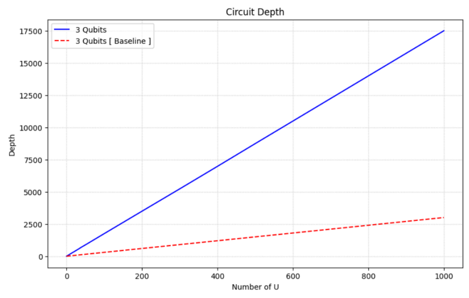


Figure 7.2

On running on AWS IONQ Aria, we found the fidelity for running for 10 qubits and n=100, was 0.99 with just 1024 shots. When we tried running for different n counts again (after a week of the first run), but we faced issue with AWS IONQ Aria1, which has been down for a few weeks. Due to this, we were not able to run our circuit against multiple n, but this gives a promising result when we tried initially for n=100. IONQ has build in error mitigation, which makes it also one of a good candidate for running our circuit against.

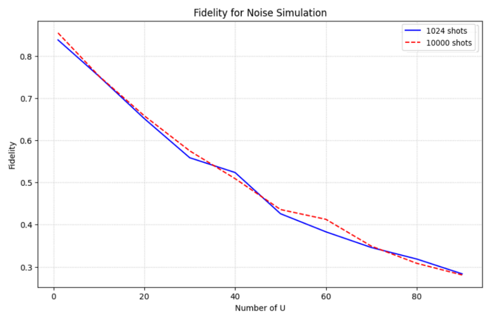


Figure 8.1

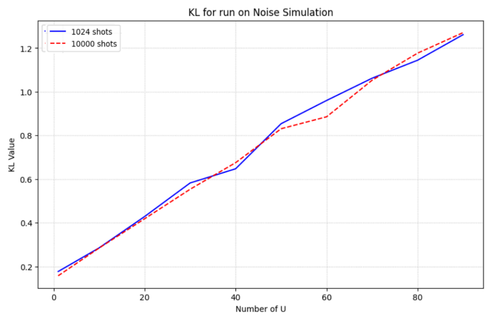


Figure 8.2

We next analysed our circuit with noisy simulation. We added noise on each U iteration. We got max fidelity at n=1, with continuous decrease with increasing n. *Figure8.1* shows us the Fidelity with noisy simulation. KL is min at n=1, but increases with increasing n. *Figure8.2* shows KL for noisy simulation.

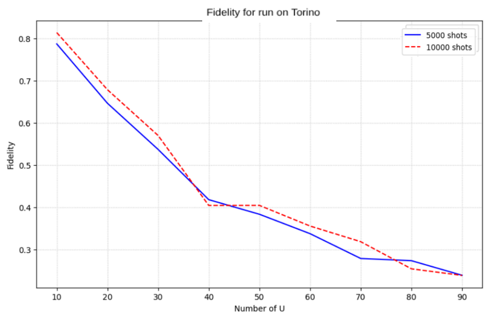


Figure 9.1

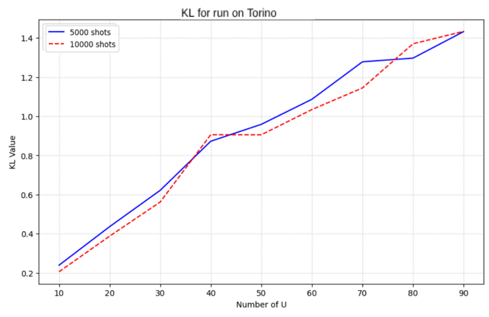


Figure 9.2

We also ran our circuit on IBM Torino[4], which is one of the IBM Quantum Heron1 Processors, and we got similar result as noisy simulation. *Figure9.1* shows fidelity and *Figure 9.2* shows KL. We also did time analysis and it took 4s to run for n=100. Heron2 processor was not available to us because of the account upgrade issue.

2.1.3 **Financial Feedback System Analogy**

HFQT

In HFQT simulation, we have a U unitary acting on three qubits. One acts as a signal {s0} and the other two as internal strategy or market sentiment, let them be {q1, q2}. The |q1q2> state tells us whether to buy or sel, based on that calibration is done. The state so achieved can be used for next discussion cycle. This loop is continuous.

Market signal -> Strategy Reset -> Trade execution -> New Market State -> Market Signal -> …

The |s0> can be the signal qubit (0 for bearish, 1 for bullish) and the |q1q2> two "strategy" or "market sentiment" qubits. The unitary U will essentially evolve the system based on the current market signal and the internal strategy, and then *encode* the hypothetical measurement and reset of the signal qubit. If values in |q1q2> can be encoded as $|00\rangle$ means "hold", $|01\rangle$ means "buy", $|10\rangle$ means "sell".
So using the sequence of feedback whether bearish or bullish, we apply the n U operation and at the end we get the trading decision we should take based on the frequently changing market sentiment.

Adaptive Portfolio Rebalancing

In case of managing a portfolio of investments. We need to continuously observe the market and evolve our strategy. Using our circuit, the {y0..yn}are like snapshots of the market or performance of a specific asset or critical market indicator. The U operation, based on the feedback will apply transformation on the portfolio like **Tweaking asset allocation, Refining trading strategies, Updating risk assessments.** Once we learned from the measurement, we reset the qubit. This could mirror neutralizing the immediate impact of the last decision or measurement so a fresh start for the next market cycle. After n operations of U, we finally get an adaptive portfolio, which reflects the long-term outcome of recorded sequential decisions.

2.1.4 **Error Analysis and Mitigation**

The initial design (section B) we tried, had a higher qubit count. Due to higher qubit count the circuit was more error prone on QPUs. We then changed the design to make the qubit Q(1), making it less expensive and less error prone. To mitigate the error further, we tried minimising the use of 2 qubit gates. The number of two qubit gates are 2+nk, where k is the number of 2 qubit gates used in U.

Our circuit is able to give probability of 1 for each n is because of one ancilla bit we are using, entangling it and then using Grover's on the |s0q1q2>. We faced issue with using the ancilla qubit and maintaining it's value to |0> at the end of each loop. The nature of this is controlled by relative phase, which is attained when we amplify. If the phase is different, then on H application we get |1> instead of |0>. This is specific to the nature of U and can easily be mathematically modulated for class of U. We have successfully executed the same with a class of U. Based on the nature of U, we change the X operation on the ancilla, which helps in amplifying the state that we require, maintaining the phase and state of ancilla. The circuit we designed is successfully working with fidelity of 1 for $n \in \{1... 2000\}$, when done with noiseless simulation. We have successfully also modeled for different class of Us, for which we have attached the notebook. For each class we have provided at least 5 different U tests.

Third is the execution on the QPU, even though we are getting fidelity of 1 on a noiseless simulator. By increasing the shots we were able to increase the fidelity to some degree. Using a QPU, like AWS IONQ Aria-1 gives us much better fidelity. We tried with 10 qubit U and n=100, and we got 0.9 fidelity in 1024 shots.

2.1.5 **Generalization and Future Use Cases**

Our implementation can easily be generalized to m qubits. We tested our implementation for 5 and 10 qubits with AWS simulator and we were able to simulate for n=1500 with t=1s. Figure11 shows the time analysis for 3,5,10 qubits. Moreover, it took us 3mins to simulate 10 qubits and n=2000 with 4GB RAM, 2cores instance. In terms of simulation time, we can clearly say our approach is scalable. Figure12.1 and Figure12.2 shows circuit depth and gate count analysis.

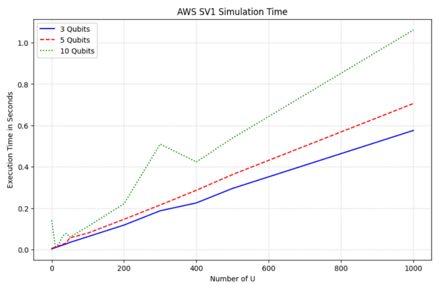


Figure 11

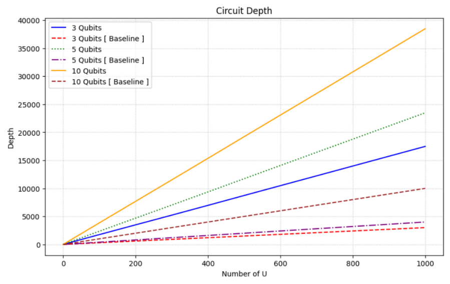


Figure 12.1

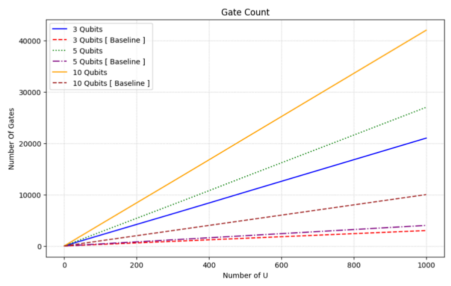


Figure 12.2

Due to O(1) qubit complexity, and classical circuit analysis to detect the nature of U, we can easily generalize and scale this for several implementations. We also simulated 10 qubit U on **AWS IONQ Aria-1**[5], and for n=100, we got fidelity of 0.99

The circuit can be extended to quantum sensors that are hyper-sensitive to subtle changes in the environment. These sensors leverage quantum phenomena like superposition, entanglement, and coherence to achieve unprecedented precision and sensitivity. In the extended multi qubit circuit, each qubit could represent a **quantum sensor element** (e.g., an individual atomic clock, a quantum magnetometer).

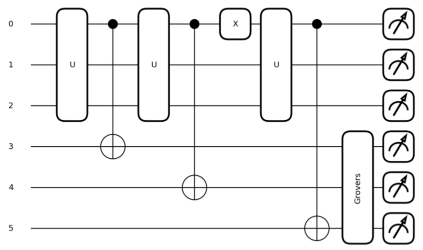
The U could represent the **interaction of the quantum sensors with their environment** (the market, a physical location, a communication channel) and the subsequent internal processing of that interaction. Each application of U is a **"sensing cycle"** where the quantum sensor network gathers new, ultra-precise data. This could be real-time market data infused with quantum-enhanced timestamps, or incredibly subtle environmental cues picked up by the sensors. The output becomes a **reinforcement signal for decision making.** When yi=1, then Anomaly else normal state. The virtual resetting is equivalent to adaptive sensing and real-time decision loops. The reset of the first qubit means the sensor is ready to provide the *next* reinforcement signal. This creates a continuous, real-time monitoring loop where the quantum sensor is always providing up-to-the-minute, ultra-sensitive feedback, enabling agile, adaptive responses in dynamic financial environments.

This can effectively be used to create an **Early Warning System.** The quantum sensors can act as an unparalleled early warning system, capable of detecting the most subtle pre-cursors to market crashes, cyberattacks, or fraudulent activities that would be missed by classical sensors. The reinforcement aspect comes from the fact that the *detection itself* triggers an adaptive response in the financial system.

Another application can be in **Adaptive Decision Policies:** The value we get from such a sequence of {yi} is not just a final number, but a dynamic, evolving **adaptive decision policy**. It reflects how our financial entity has *learned to manage risk in real-time* by constantly sensing its environment with quantum precision and adjusting its actions based on positive or negative reinforcement signals.

2.1 SectionB: Solution with O(n) qubit complexity

2.1.1 Circuit Design and Analysis



We use n auxiliaries and we store the result of each of the yi in a_i . After n loops are over we apply the Grovers to the n auxiliaries to get state { y0, …, yn }. The grover's phase flip and diffusion increases the amplitude of { y0, …, yn }, increasing the probability of getting { y0, …, yn } on auxiliary, hence increasing the probability of getting { q1, q2 } desired end state. Code for this approach can be found in github.

The approach has more qubit complexity but is suitable for generic approach. The complexity is almost same as the first, but simulation is heaver, which can be compounded by the fact when we tried to simulate for n > 30, it took mins. On doing the same on IBM, it took 30s as compared to 3s for first solution. So we can clearly see, the first is 10x better in terms of speed efficiency. Also, this is more prone to errors when we run on QPUs.

3. **References**

1. https://pennylane.ai/qml/demos/tutorial_grovers_algorithm
2. https://learning.quantum.ibm.com/tutorial/grovers-algorithm
3. IBM Quantum. https://quantum.ibm.com/, 2021
4. IBM Quantum. https://quantum.cloud.ibm.com/, 2025
5. AWS IONQ Aria, https://ionq.com/quantum-systems/aria