3-qubit chain "punched" by measurements - A simple physical application of Qiskit

QXQ Introduction to quantum computing course - Final project

Abstract

In this work I use the Qiskit BasicAer statevector simulator to simulate a system of 3 qubits, with one of the qubits being subject to periodic measurements, in an ideal (noise-free) quantum computer. I introduce the concept of quantum trajectories and use it alongside Qiskit to show that periodic measurements stabilize the energy of the system in time.

1. Introduction

As an application of the concepts developed in the course, I decided to implement a simulation in Qiskit to investigate how a chain of 3 qubits is affected when one of its qubits is submitted to periodic measurements.

I thought that the investigation of how exactly the energy of qubit chains vary due to measurements may contribute to the general problem of determining new techniques for the manipulation of energy exchanges within quantum processors, while serving as a new and exciting way for me to test and deepen my abilities with Qiskit.

In §2 I shall present a brief outline of the main concept I used to make my investigation, known as *quantum trajectories*, and define the notation of the physical and mathematical quantities I am going to consider. In §3 I move into the algorithm that I wrote with Qiskit to implement the process of §2. Finally, in §4 I present some simple results that I have obtained so far, and in §5, I digress about some conclusions and future perspectives.

Throughout my presentation, I shall assume that the reader is familiar with the concepts presented along our course.

2. Outline and methods

Being more specific about what was mentioned above, the idea is to repeat the following instructions several times:

1. *The chain evolves a time T.*
2. *One of the qubits is measured*[1].
3. *The expectation value of the energy of the chain E is computed.*

It is useful to call a single implementation of these three instructions a *step* and a sequence of steps a *trajectory*.

[1] Formally, we measure Z, which is the measurement process inherent to Qiskit.

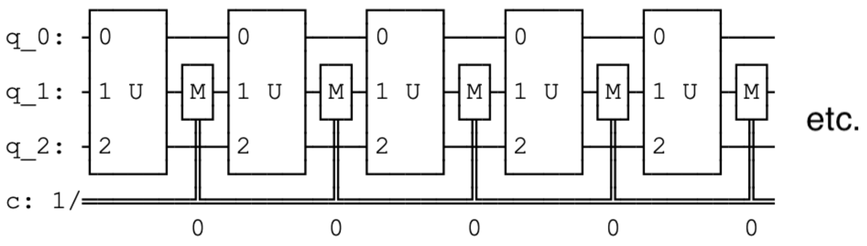

Figure 1. *A quantum circuit of 3 qubits submitted to periodic time-evolutions followed by measurement "punches" on qubit 1.*

This idea is similar to what happens in the so-called Ising model for ferromagnetic materials [2]. In this model, however, a spin-flip is induced on a chain of spins as the net effect of an external magnetic field, whereas, here, the intention is to "punch" a qubit of the chain with a quantum measurement.

A generic unitary time-evolution rotates each qubit along its own Bloch's sphere and it is well known from group theory that the set of all possible single-qubit rotations can be described in terms of the Pauli gates, I, X, Y and Z. Hence, to the end of implementing the time-evolution of instruction 1 and of calculating the mean energy of the system in instruction 3, we take the Hamiltonian operators

$$H_{open} = -J_1 I \otimes X \otimes X - J_2 Z \otimes Z \otimes I$$

$$H_{closed} = -J_1 I \otimes X \otimes X - J_2 Z \otimes Z \otimes I - J_3 X \otimes I \otimes X$$

such that $U = exp\{-iHT\}$ and $E = <\psi|H|\psi>$, where $|\psi>$ is the state of the system at a given instant of time. The $J_i's,\ i = 1, 2, 3,$ are coupling constants. These are simple choices for H, but many others could be explored as well. The first one represents an *open* qubit chain, which is a chain where qubits 1 and 3 do not interact with one another, and the second a *closed* qubit chain, which is a chain where qubits 1 and 3 do interact with one another.

Noting that the outcome of each measurement is inherently probabilistic, we expect the system to behave very differently along different trajectories. E.g. in a trajectory of, say, 4 steps, the set of outcomes of the measurements on the chosen qubit may be $\{|0>, |1>, |1>, |0>\}$, $\{|1>, |0>, |1> |1>\}$, $\{|1> |0>, |0>, |0>\}$, etc. We shall then, in this case, obtain a different set of energy expectation values for *each* possible trajectory the system may undergo (see Fig.(?)).

However, what I want to investigate is actually the *general* behavior of the system. This means that we may simply run a sufficiently large number of trajectories, calculate E for each step of each trajectory, and at the end calculate the mean value of E for each step. For example, we may run 1000 trajectories and discover that we obtain at step 3 of each the energies $E_1^3, E_2^3, \ldots, E_{1000}^3$ with probability $p_1^3, p_2^3, \ldots, p_{1000}^3$. Thus, at the end we simply calculate

$$E_{mean} = (p_1^3 E_1^3 + p_2^3 E_2^3 + \ldots + p_{1000}^3 E_{1000}^3)/(p_1^3 + p_2^3 + \ldots + p_{1000}^3)$$

to obtain a satisfactory result for the *general* behavior of the system at step 3. The intention is to apply this procedure to all steps in order to set a picture of the general behavior of the system as a whole.

The mean values are needed in order to account for both quantum (relative to the probabilistic nature of measurements outcomes) and classical (relative to the fact that we cannot know exactly which trajectory the system will undergo) uncertainties that, in general, arise along the process.

3. Structure of the algorithm

In order to implement my project I had to learn more about how to use simulators and backends in Qiskit. These features are essential if we wish to simulate or run actual experiments. Reference [1] was very useful for me.

Through my journey I learned that a *backend* in Qiskit is simply an actual device (quantum computer) or a Python library called a *simulator* that, as the name suggests, can simulate an actual device. We know that a quantum circuit is a set of instructions (gates) applied to a set of qubits and classical bits producing some final output. In order to send these instructions to a backend, Qiskit needs to *transpile* the gates of the circuit to a set of gates that can be read by the backend. After the transpilation, we may run our circuit even accounting for actual effects, such as noise and hardware imperfections, and then obtain information, such as the circuit's state vector. This state vector, for instance, may be used to compute expectation values of physical quantities, as we did.

I chose, however, a simulator without noise as the backend of my project. This means that I am supposing that my instructions are running in a noise-free (ideal) quantum computer. Qiskit has many simulators, but for my purposes I thought that it would be enough to use the most basic one, the *BasicAer*. For the sake of briefness, I included my code, with comments, in the Appendix (§7).

4. Results

Let us fix the simple scenario of a chain of 3 qubits only (labeled, of course, by 0, 1 and 2). Using the inherent configuration of Qiskit, the outcomes of the measurements are, of course, 0 or 1. This was more appropriate for my personal hardware, since a large number of trajectories must be performed, and for each of them the system must be submitted to a considerable number of steps, making the algorithm somewhat heavy. Nevertheless, with more time, I do intend to investigate larger chains in the future (see §5).

For both the open and the closed chains, Fig.(2) indicates that the energy stabilizes at some point. This means that *the measurement processes can bring the system to a situation of constant energy*. In fact, I tested many scenarios, varying the values of both the coupling constants and the evolution time (§2), and found that the general behavior of the system is invariably the same. In Fig.(3) we find a sample of such tests. It must be pointed out, however, that, although Fig.(2) indicates that the stabilization is at zero, we cannot conclude immediately that measurements "drained" energy from the system, because it can be shown that zero is not the lowest possible energy of the Hamiltonians[2] in §2. This is further reinforced by Fig.(3).

We can also observe that, in this specific system that I am treating, when we let the system evolve more time between measurements, the stabilization effect occurs in fewer steps, as indicated for a closed chain in Fig.(3). It would be interesting to investigate how general this behavior is for measurements in an arbitrary system.

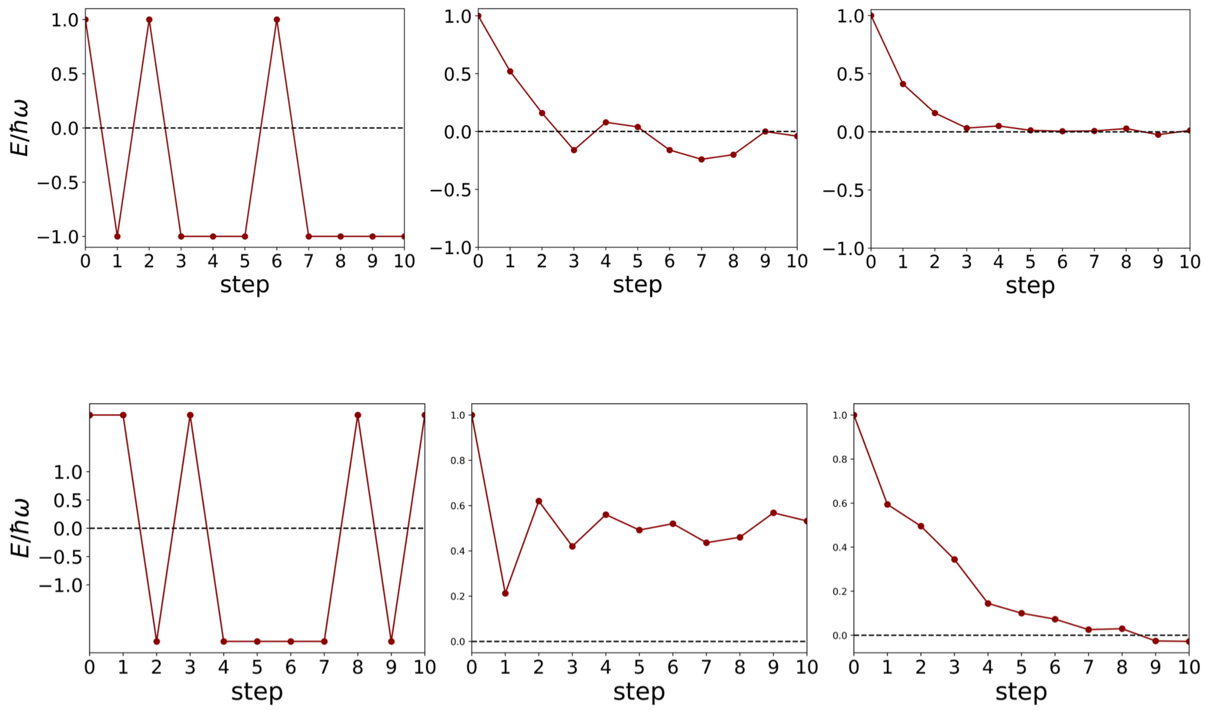

Figure 2. First row: energy of a closed 3 qubit chain for 1, 50 and 5000 trajectories, respectively. Second row: energy of an open 3 qubit chain for 1, 500 and 5000 trajectories, respectively. In both figures we set $J_i = -2.0$ and $T = \pi/5$. It can be noted that both systems stabilize at zero energy, the closed chain being, however, a little more resistant to stabilization.

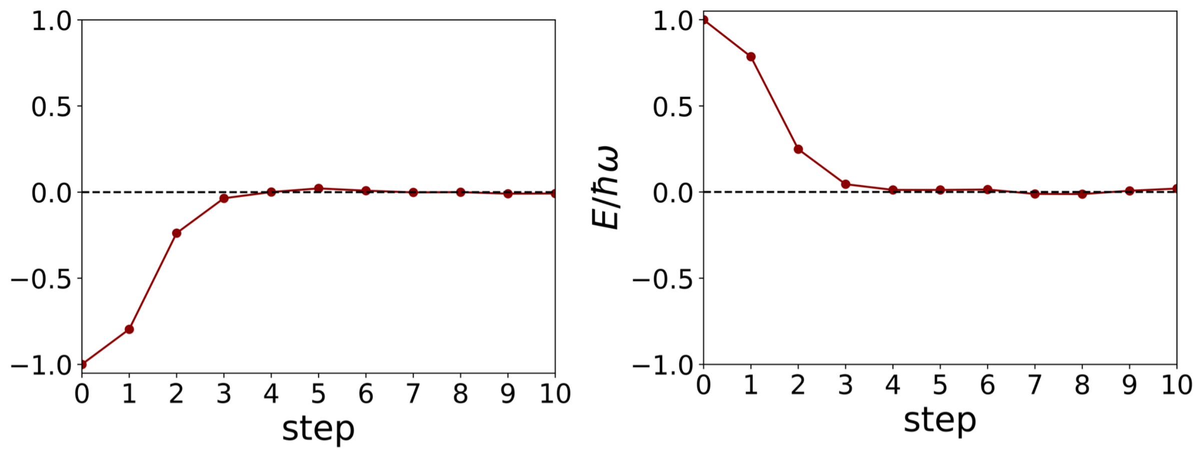

Figure 3. Left: Energy of a closed 3 qubit chain for all $J_i = -2.0$. Right: Energy of a closed 3 qubit chain for $J_1 = -2.0$, $J_2 = J_3 = +2.0$. In each plot we considered 5000 trajectories and set

[2] In a more sophisticated language, it is a simple matter to construct the matrix representation of the Hamiltonians and find their eigenvalues.

$T = \pi/3$. In comparison with Fig.(2), it is evident that If we raise the evolution time T, the effect of stabilization occurs in fewer steps.

5. Conclusion and perspectives

In this work I showed a simple application of the Qiskit BasicAer statevector simulator to a simple physical system, reinforcing that Qiskit can be used alongside many physical concepts, such as quantum trajectories, to obtain results of experiments that can be performed in an actual hardware.

More specifically, we showed that periodic measurements induce an energy stabilization in a 3-qubit chain. In the future, I intend to continue using Qiskit to analyze many other different sorts of interaction Hamiltonians and chains of N > 3 qubits. Also, I wish to investigate if the phenomenon of increasing the evolution time (Fig.(3)) holds for measurements on an arbitrary system. To these ends, I am looking forward to knowing more about how to run simulations and develop research on actual devices with IBM.

6. References

[1] *Weaver, James. "Qiskit Pocket Guide: Quantum Development with Qiskit.", O'REILLY MEDIA, 2023.* https://www.amazon.com/Qiskit-Pocket-Guide-Quantum-Development/dp/1098112474

[2] *"Ising Model - Wikipedia." Ising Model - Wikipedia, 21 Apr. 2013,* en.wikipedia.org/wiki/Ising_model.

7. Appendix

Here is the code I constructed to obtain my results. Let me try to denote a general overview: In the first part I define a function to use *BasicAer* to obtain the state vector of a quantum circuit after a measurement process. Then I define the operators and gates I am going to use to implement the main part of the algorithm, which follows next. Generally speaking, in the main part, I simply use this function and these gates and operations to consider the process proposed in §3. Finally, in the last part, which I did not include here for the sake of redundancy, I use Python's well known matplotlib library to make the plots shown in §3.

```python
import numpy as np
```

```python
from qiskit import BasicAer, transpile

# Setting the statevector simulator as backend
backend = BasicAer.get_backend("statevector_simulator")

# Creating a function to obtain the statevector of a quantum circuit
def Simulate_StateVector(q_circuit):
    tqc = transpile(q_circuit, backend) # Transpiling circuit to be read by the backend
    job = backend.run(tqc) # Telling backend to read the circuit
    result = job.result() # Getting outputs from backend
    statevector = result.get_statevector(tqc, 4) # Selecting the statevector output
    return statevector # Returning circuit statevector
```

```python
from qiskit.opflow import I, X, Y, Z

# Constructing hamiltonian H and time-evolution operator U

# Coupling constants for H
J_1 = -2.0
J_2 = 2.0
J_3 = 2.0

# Evolution time between measurements
evolution_time = np.pi/3

# Operators
H = - (J_1/2) * (I ^ X ^ X) + (J_2/2) * (Z ^ Z ^ I) - (J_3/2)*(X ^ I ^ X) # Hamiltonian
U = (evolution_time*H).exp_i() # Time-evolution
```

```python
from qiskit import QuantumCircuit
from qiskit.quantum_info import Statevector

# Number of qubits in the circuit
num_qubits = 3

# Number of trajectories and steps
num_trajectories = 1
num_steps = 10

# Array to store mean energies over all trajectories
Final_energies = np.zeros(num_steps + 1)

# Qubit to be measured
measured_qubit = 1

for trajectory in range(num_trajectories):

    if trajectory%100 == 0: print(trajectory)

    # Creating a circuit of N qubits and cbits
    qc = QuantumCircuit(num_qubits, num_qubits)

    # Array to store energies
    Energies = list([])

    # Appending initial energy
    psi = Simulate_StateVector(qc) # initial state array of the system
    psi = Statevector(psi) # Converting state array to qiskit.quantum_info Statevector class
    expectation_H = psi.expectation_value(H) # Expectation value of H
    Energies.append(np.real(expectation_H)) # Appending E to array 'Energies'

    # Evolutions and measurements
    for step in range(num_steps):
        qc.unitary(U, range(num_qubits), label = 'U') # Evolving the system
        qc.measure(measured_qubit, measured_qubit) # Measuring qubit 1
        psi = Simulate_StateVector(qc) # Getting the state array of the system after measurement
        psi = Statevector(psi) # Converting state array to qiskit.quantum_info Statevector class
        expectation_H = psi.expectation_value(H) # Expectation value of H
        Energies.append(np.real(expectation_H)) # Appending E to array 'Energies'

    for step in range(num_steps + 1):
        Final_energies[step] += Energies[step] # Summing energy values of the trajectory to 'Final_energies' array

# Calculating mean energies after all trajectories
Final_energies /= num_trajectories
```

8. Acknowledgements

I would like to thank IBM for sponsoring my participation in this course and the fellows from QubitXQubit for the amazing job they did along these two instructive and pleasant semesters.

I would like to thank my university supervisor for suggesting the main idea of this project, and for all the directions and professional support.

Finally, I would like to thank my family for giving me all the love, support, and a meaningful life.