**1. transient stability data of the SMIB system**

Our plan is to apply quantum machine learning to power system **transient stability assessment (TSA)** problem, and investigate the expressibility of quantum neural network in TSA issues.

Fig.1 provides the single-machine infinite-bus (SMIB) power system, where a generator is connected to a bus with a constant voltage level (we call it an infinite bus because it has an infinite power capacity to maintain its voltage level). This is one of the most widely-used test systems in power system research. Since the stability region of SMIB (i.e., a 2-dimensional region) can be analytically computed, SMIB always serves as an indispensable benchmark for testing the performance of a TSA method.

Diagram

Description automatically generated

Figure 1. Topology of the SMIB system

We apply fault in the SMIB system (as shown in Fig.1) to trigger its dynamic response. Under different fault clearing times, the system will have different behaviors.

In the provided data, “SMIB\_Traj.mat” provides the SMIB system’s responses under different fault clearing time. If you plot out the Traj. simData.TSAfeature, you will see that in some trajectories, the oscillation can be damped (as shown in Fig.2(a)), which refers to the stable cases. But in some trajectories, the system finally loses stability, which refers to the unstable cases (as shown in Fig.2(b)). Note: “SMIB\_Traj.mat” will not be used for QML training. I am providing those data only to provide you an intuitive idea about what power system TSA problem looks like.

Chart, scatter chart

Description automatically generated Chart

Description automatically generated

Figure 2. Stable and unstable cases in the SMIB system

You will be using “SMIB\_feature.mat” for QML. “SMIB\_feature.X\_train” and “SMIB\_feature.Y\_train” provides the training set (“X” refers to the 2-dimensioanl feature and “Y” provides the classification result, i.e., 0 or 1). If you further plot out the 2D features, you will see the stable and unstable cases are distributed as a balloon-shape shown in Fig.3, which is called the stability region of the SMIB system. **Our target is to classify the stable and unstable cases using QNN, or in other words, try to depict the stability region using QNN.**

Chart

Description automatically generated

Figure 3. Stability region of the SMIB system

**2. noisy simulator set up in qiskit**

Ideally, the quantum simulator you are using is a noise-free quantum simulator, meaning all the quantum gates/quantum operators do not have any error. However, in real quantum computers, all the quantum gates will have certain errors, which is called the noisy-intermediate-scale quantum (NISQ) devices. For example, Fig.4 provides the error information of a typical IBM quantum computer. You can see that each quantum gate has uncertain errors. Reading out a quantum circuit also involves error. Therefore, only studying quantum algorithms in the noise-free environment is too optimistic and does not reflect their practical performance.

Graphical user interface, text, application

Description automatically generated

A picture containing graphical user interface

Description automatically generated

Figure 4. Error rate of a typical IBM quantum computer

There are basically two approaches to study the performance of a quantum algorithm in the NISQ environment.

**(1) Using real quantum computers**

For example, for IBM, you can register an account here: <https://quantum-computing.ibm.com/>. And then you will be able to access some of IBM’s real quantum computers through cloud. However, in recent months, IBM quantum computers are becoming very popular and the waiting list can be relatively long (you may need to wait for several days until you can use it for a single run).

Therefore, if you are interested in using the real hardware, you can try IBM quantum computers. But this is possibly very time consuming because free accounts only have access to two quantum computers and there is always a long queue.

**(2) Using noisy simulators**

An alternative approach is to add noises to the quantum simulators to mimic the performance of real quantum computers. Although it is not easy to accurately replicate the noise of real quantum computers, using noisy simulators will more or less provide us with some conclusions on the noisy performance of quantum algorithms.

After you have completed the training of the QNN, you can test its performance in noisy environments. The setup is provided in the attached python codes. More information can be found here: <https://qiskit.org/documentation/apidoc/aer_noise.html>

We have also provided python scripts about setting up the noise models in the package, which is through the python package pennylane https://pennylane.ai/blog/2021/05/how-to-simulate-noise-with-pennylane/.

**3. Schedule**

The following table provides a suggested schedule for the 6-week program.

|  |  |
| --- | --- |
| Week | Research plan |
| 1 | Read relevant documents about pennylane or qiskit, and get familiar with the provided code package. |
| 2 | Generate power system data through the provided matlab code, and try to run the python code. |
| 3 | Train the quantum neural network using the provided package and try to examine its performance both on the noise-free simulator and noisy simulator using the provided noisy model. |
| 4 | Try to change the structure of the quantum neural network to see how different QNN settings (such as number of qubits, number of layers, or used quantum gates) will impact the performance of its classification capability for power system TSA. |
| 5 |
| 6 | Collect results and wrap up |