Quantum Simulator Noise Modeling using Qiskit and Real Device Calibration Data

Nasir Ali

October 12, 2023

1 Introduction

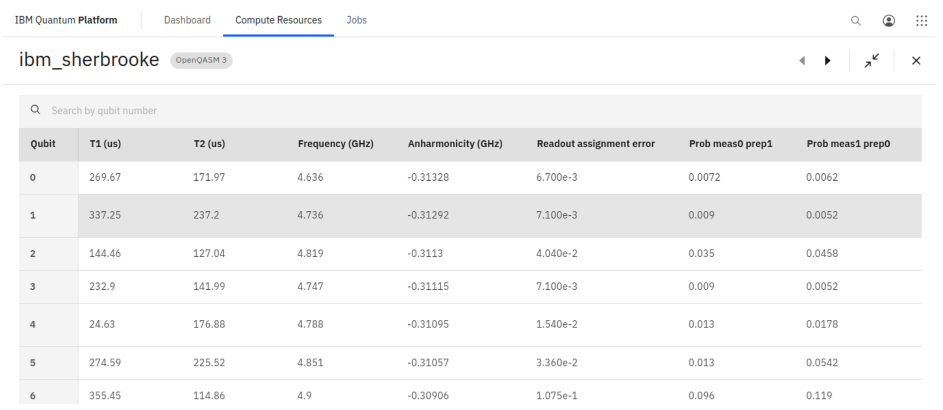
In the pursuit of realizing practical quantum computing, understanding the noise in quantum devices and how to mitigate them is of paramount importance. By integrating real device calibration data into quantum simulations, we can emulate the behavior of real quantum devices more accurately. This report discusses a method to incorporate calibration data from IBM's quantum devices into a quantum noise model using Qiskit.

2 Understanding the Calibration Data

The key parameters in the data are:

- **Qubit**: The identifier of the qubit.
- **T1 (us)**: The energy relaxation time. It signifies how long a qubit can maintain its state before it decays. Higher values generally represent better performance.
- **T2 (us)**: The dephasing time. It provides an idea of how long a qubit can maintain its phase coherence. Like T1, higher values are generally preferable.
- **Frequency (GHz)**: The operating frequency of the qubit.

- **Anharmonicity (GHz)**: Indicates the energy difference between the |1> and |2> states of a qubit relative to the |0> and |1> states.
- **Readout assignment error**: This signifies the error in reading the state of the qubit. Lower values represent better accuracy in reading the qubit's state.
- **Prob meas0 prep1**: The probability that the qubit, when prepared in the |1> state, will be measured as being in the |0> state.
- **Prob meas1 prep0**: The probability that the qubit, when prepared in the |0> state, will be measured as being in the |1> state.



Qubit	T1 (us)	T2 (us)	Frequency (GHz)	Anharmonicity (GHz)	Readout assignment error	Prob meas0 prep1	Prob meas1 prep0
0	269.67	171.97	4.636	-0.31328	6.700e-3	0.0072	0.0062
1	337.25	237.2	4.736	-0.31292	7.100e-3	0.009	0.0052
2	144.46	127.04	4.819	-0.3113	4.040e-2	0.035	0.0458
3	232.9	141.99	4.747	-0.31115	7.100e-3	0.009	0.0052
4	24.63	176.88	4.788	-0.31095	1.540e-2	0.013	0.0178
5	274.59	225.52	4.851	-0.31057	3.360e-2	0.013	0.0542
6	355.45	114.86	4.9	-0.30906	1.075e-1	0.096	0.119

Figure 1: Calibration Data ibm_sherbrooke

3 How to Use this Data for Building a Noise Model

- **T1 and T2 Relaxation Times:** These can be used to model amplitude damping and phase damping channels. The inverse of these values provides the rate at which these errors occur.
- **Readout Assignment Error:** Helps in modeling readout errors. If you know the qubit is in state |0>, but there's a 0.0072 chance you'll read it as |1>, that's a readout error you'll want to include in your model.

- **Anharmonicity:** While not directly an error source, understanding the anharmonicity can be essential in certain multi-qubit operations where the qubit might accidentally transition to the |2> state.
- **Prob meas0 prep1 and Prob meas1 prep0:** These probabilities give you direct insights into the state preparation and measurement (SPAM) errors. When building your noise model, these probabilities can be used to model the chance of incorrect state preparation or measurement.

4 Setting Up the Noise Model

4.1 Steps

- **Define Quantum Noise Channels:** Based on T1, T2, and readout errors, define the appropriate noise channels for each qubit.
- **Apply Noise to Quantum Operations:** For every quantum operation (like gates or measurements), apply the corresponding noise channel.
- **Simulate with Noise Model:** Run your quantum algorithms using the noisy simulator.
- **Validation:** Compare the results of your noisy simulator with the real IBM Quantum device to validate the accuracy of your noise model.

The code provided offers a comprehensive methodology to construct a noise model suitable for quantum simulations using Qiskit's Aer module.
`https://docs.quantum-computing.ibm.com/test/building_noise_models`

4.2 Import Necessary Modules

Start by importing the required libraries and functions essential for building and simulating the noise model.

```
import numpy as np
from qiskit import QuantumCircuit, transpile
from qiskit.quantum_info import Kraus, SuperOp
from qiskit_aer import AerSimulator
from qiskit.tools.visualization import plot_histogram

# Import from Qiskit Aer noise module
from qiskit_aer.noise import (NoiseModel, QuantumError, ReadoutError,
pauli_error, depolarizing_error, thermal_relaxation_error)
```

4.3 Error Definitions

Bit-flip, phase-flip, and combined bit-and-phase-flip errors are defined. These are typical errors in quantum computations.

```
# Construct a 1-qubit bit-flip and phase-flip errors
p_error = 0.05
bit_flip = pauli_error([('X', p_error), ('I', 1 - p_error)])
phase_flip = pauli_error([('Z', p_error), ('I', 1 - p_error)])
print(bit_flip)
print(phase_flip)
```

4.4 Readout Errors

Defines the probability of misreading a qubit's state during measurement.

```
# Measurement miss-assignement probabilities
p0given1 = 0.1
p1given0 = 0.05

ReadoutError([[1 - p1given0, p1given0], [p0given1, 1 - p0given1]])
```

4.5 Bit-flip Noise Model

Constructs a noise model with bit-flip errors added to all qubit gates.

```
# Example error probabilities
p_reset = 0.03
p_meas = 0.1
p_gate1 = 0.05

# QuantumError objects
```

```
error_reset = pauli_error([('X', p_reset), ('I', 1 - p_reset)])
error_meas = pauli_error([('X',p_meas), ('I', 1 - p_meas)])
error_gate1 = pauli_error([('X',p_gate1), ('I', 1 - p_gate1)])
error_gate2 = error_gate1.tensor(error_gate1)

# Add errors to noise model
noise_bit_flip = NoiseModel()
noise_bit_flip.add_all_qubit_quantum_error(error_reset, "reset")
noise_bit_flip.add_all_qubit_quantum_error(error_meas, "measure")
noise_bit_flip.add_all_qubit_quantum_error(error_gate1,["u1","u2","u3"]
noise_bit_flip.add_all_qubit_quantum_error(error_gate2, ["cx"])

print(noise_bit_flip)
```

4.6 Thermal Relaxation Errors

Defines T1 (energy relaxation) and T2 (dephasing) times to introduce realistic noise based on the provided calibration data.

```
# T1 and T2 values for qubits 0-3
T1s = np.random.normal(50e3, 10e3, 4)
T2s = np.random.normal(70e3, 10e3, 4)
# Truncate random T2s <= T1s
T2s = np.array([min(T2s[j], 2 * T1s[j]) for j in range(4)])

# Instruction times (in nanoseconds)
time_u1 = 0   # virtual gate
time_u2 = 50  # (single X90 pulse)
time_u3 = 100 # (two X90 pulses)
time_cx = 300
time_reset = 1000  # 1 microsecond
time_measure = 1000 # 1 microsecond

# QuantumError objects
errors_reset = [thermal_relaxation_error(t1, t2, time_reset)
for t1, t2 in zip(T1s, T2s)]
errors_measure = [thermal_relaxation_error(t1, t2, time_measure)
for t1, t2 in zip(T1s, T2s)]
```

```
errors_u1  = [thermal_relaxation_error(t1, t2, time_u1)
for t1, t2 in zip(T1s, T2s)]
errors_u2  = [thermal_relaxation_error(t1, t2, time_u2)
for t1, t2 in zip(T1s, T2s)]
errors_u3  = [thermal_relaxation_error(t1, t2, time_u3)
for t1, t2 in zip(T1s, T2s)]
errors_cx = [[thermal_relaxation_error(t1a, t2a, time_cx).expand(
thermal_relaxation_error(t1b, t2b, time_cx))
for t1a, t2a in zip(T1s, T2s)]
for t1b, t2b in zip(T1s, T2s)]

# Add errors to noise model
noise_thermal = NoiseModel()
for j in range(4):
noise_thermal.add_quantum_error(errors_reset[j], "reset", [j])
noise_thermal.add_quantum_error(errors_measure[j], "measure", [j])
noise_thermal.add_quantum_error(errors_u1[j], "u1", [j])
noise_thermal.add_quantum_error(errors_u2[j], "u2", [j])
noise_thermal.add_quantum_error(errors_u3[j], "u3", [j])
for k in range(4):
noise_thermal.add_quantum_error(errors_cx[j][k], "cx", [j, k])

print(noise_thermal)
```

5 Using Calibration Data from the Real Quantum Device

The calibration data provided contains critical parameters, T1 and T2 times, for each qubit. This data helps introduce a realistic noise model: `https://ibm.co/3XQgStY`

5.1 Extracting and Integrating Data

1. Extract the T1 and T2 values for each qubit from the calibration data.
2. Replace the randomly generated T1s and T2s in the code with these extracted values.

5.2 Automating Data Integration

Given that IBM provides 126 rows of data every hour, you can fetch and update this data programmatically, ensuring the most recent and accurate values are always used.

6 Conclusion

This report has outlined a detailed method to construct a quantum noise model using Qiskit and integrate real calibration data for enhanced realism. By using actual calibration data, the simulations can offer insights closer to real quantum computations.