Qiskit Lab Manual

QAMP Fall 21 - Checkpoint 2

Manjula Gandhi S (Mentor)

Associate Professor, Coimbatore Institute of Technology, India

Guncha Malik (Mentee)

CISSP

Security Compliance Architect, IBM India

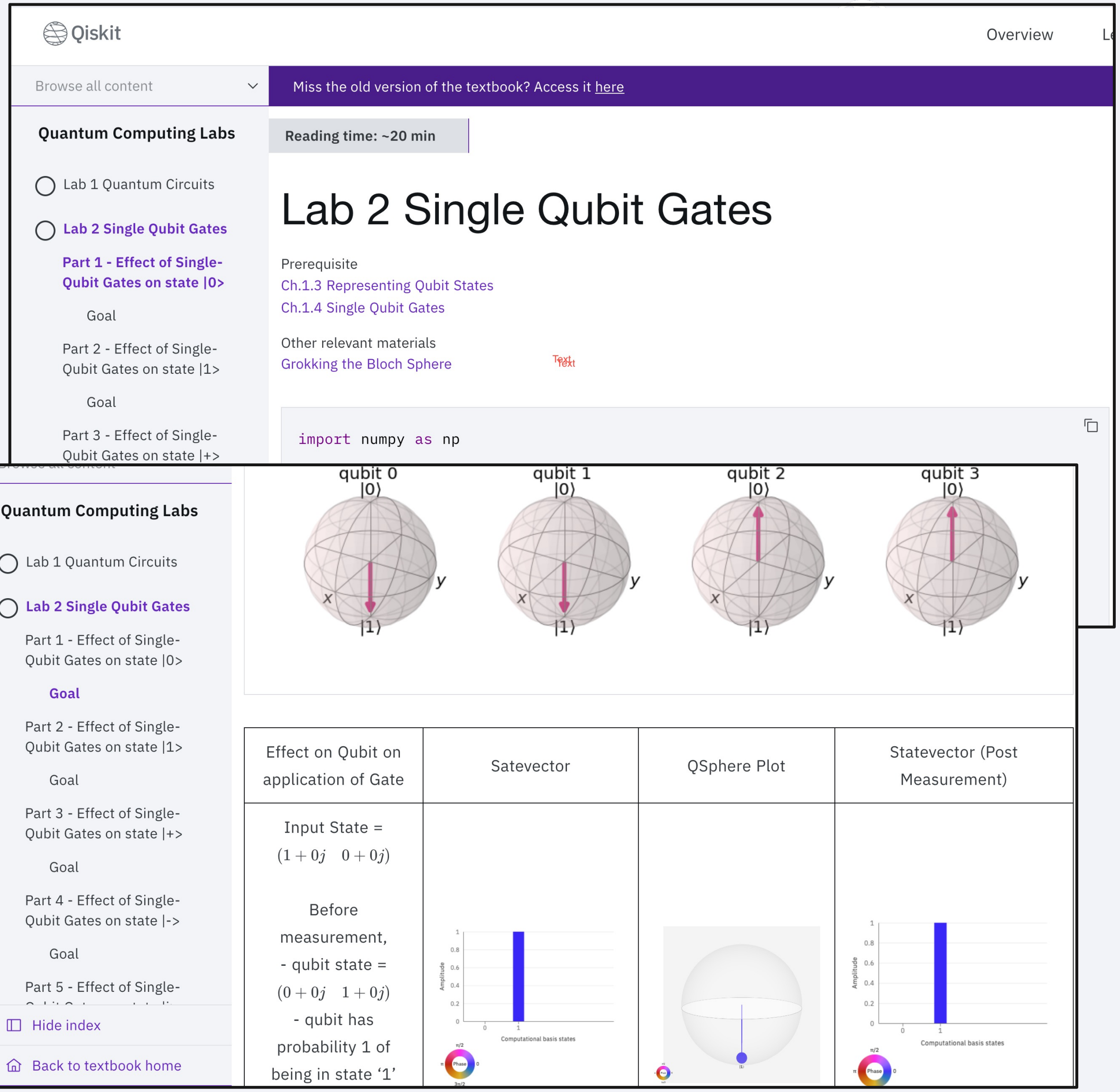
Qiskit

Lab 1 Completed

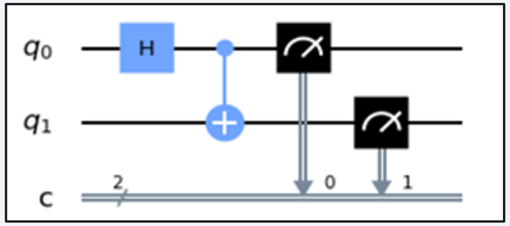
Lab on Single Qubit Gates has now been published on the Qiskit (Quantum Computing Labs) web page and is available for the users to consume.

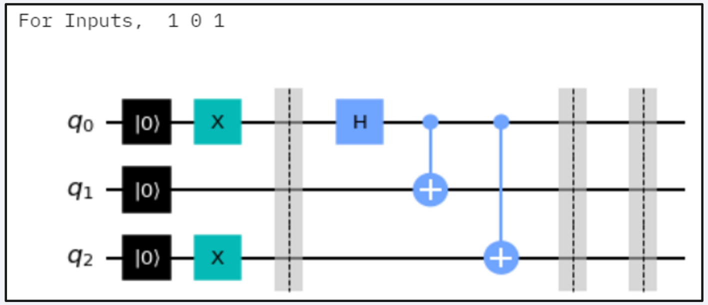
The lab enables the learners to understand the effects of single qubit gates like Pauli gates and Phase gate (S) on the eigenvectors of X, Y and Z.

The lab also provides a visual depiction of the circuit along with the plot on blochsphere, Qsphere and the input and output/state probability as statevector.



Lab 2 and 3 Progress





We have started working on Labs 2 and 3 that will focus on the Bell Circuits and GHZ Circuit.

These lab manuals are aimed at students/ beginners starting on Qiskit to smoothly transition from fundamentals to advanced concepts on the Qiskit (Quantum Computing Labs).

These lab manuals are targeted to be completed by the Checkpoint 3 and published on the Quantum Computing Labs webpage.

Q-SPHERE REPRESENTATION FOR ALL THE STATES IN A 3 QUBIT GHZ CIRCUIT

```
In [1]: import numpy as np
        from qiskit import QuantumCircuit, QuantumRegister, ClassicalRegister
        from qiskit import IBMQ, Aer, transpile, assemble
        from qiskit.visualization import plot_histogram, plot_bloch_multivector, plot_state_qsphere

In [2]: import warnings
        warnings.filterwarnings('ignore')

In [3]: sim = Aer.get_backend('aer_simulator')

In [6]: def ghzCircuit(inp1,inp2,inp3):

            qc = QuantumCircuit(3)
            qc.reset(range(3))

            if inp1 == '1':
                qc.x(0)
            if inp2 == '1':
                qc.x(1)
            if inp3 == '1':
                qc.x(2)

            qc.barrier()

            qc.h(0)
            qc.cx(0,1)
            qc.cx(0,2)

            qc.barrier()

            qc.save_statevector()
            qobj = assemble(qc)
            state = sim.run(qobj).result().get_statevector()
```