Homework 2

Answer the following questions in this Jupyter notebook.

For non-coding questions, put your answers in a new markdown cell or edit the cell where the question is written.

For coding questions:

- Complete all sections marked `# YOUR CODE GOES HERE`.
- Do not modify cells marked with `# DO NOT MODIFY` at the top.

In [ ]:
```
# DO NOT MODIFY - This cell performs all the imports and backend setup you need for this assingment

# General
import numpy as np

# Qiskit imports
from qiskit import QuantumCircuit
from qiskit.quantum_info import Statevector
from qiskit.transpiler.preset_passmanagers import generate_preset_pass_manager
from qiskit.visualization import plot_histogram

# Qiskit Runtime imports
from qiskit_ibm_runtime import QiskitRuntimeService
from qiskit_ibm_runtime import SamplerV2 as Sampler

# Plotting routines
import matplotlib.pyplot as plt

# Setup hardware backend and transpiler
service = QiskitRuntimeService()
backend = service.backend("ibm_rensselaer", instance="rpi-rensselaer/general/general")
pm = generate_preset_pass_manager(target=backend.target, optimization_level=3)

print(f"Using {backend.name} backend")
```

```
Using ibm_rensselaer backend
```

Problem 1 - Drill Questions

1. A certain signal causes a qubit's statevector to rotate around the Y-axis of the Bloch sphere. Assuming the qubit is initialized in the $\ket{0}$ state, what pulse duration (in wave periods) would result in the qubit moving to the $\ket{1}$ state? What would the equivalent quantum gate be?

2. Using `numpy`, show that each of the Pauli gates ($\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$) are unitary. (Hint: Recall that for any unitary matrix $\mathbf{U}$, $\mathbf{U}\mathbf{U}^\dagger = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix. Also recall that the adjoint of a matrix is equal to its conjugate transpose. You may need to consult the `numpy` documentation for a reminder on how to implement these operations)

3. Go to your IBM Quantum dashboard and select ”Compute Resources” from the menu bar along the top of the page. Click on the ibm rensselaer backend to view the current cal- ibration information for the machine. What are the native basis gates supported by the machine?

In [ ]:
```
# YOUR CODE GOES HERE - Put the numpy code for problem 2 in this cell.

# Fill in the elements of the arrays below to create the Pauli matrices.
# Recall that in Python, complex numbers are written a + bj (ex. 1.0 + 1.0j)
X = np.array([[0, 1],
              [1, 0]])

Y = np.array([[0.0+0.0j,0.0-1.0j],
              [0.0+1.0j,0.0+0.0j]])

Z = np.array([[1, 0],
              [0, -1]])
```

In [ ]:
```
# YOUR CODE GOES HERE -- Complete the function
def is_unitary(M):
    """
    Returns True if a matrix M is unitary, False otherwise.
    """
    # FILL THIS IN
    M_dag = np.array([[np.conjugate(M[0][0]), np.conjugate(M[1][0])],
                      [np.conjugate(M[0][1]), np.conjugate(M[1][1])]])

    I = np.eye(np.diag(M).size)

    # Test equality of MM_dag with the identity matrix
    return np.allclose(M @ M_dag, I)
```

In [ ]:
```
# Test your answer - all these examples should print "True"
print(is_unitary(X))
print(is_unitary(Y))
print(is_unitary(Z))
```

```
True
True
True
```

Problem 2 - Programming Exercise

In this exercise, we will do a bit more exploration with the Bell states we ran in the first homework. Recall that Bell states are the simplest instance of an entangled quantum state, and possess correlations that cannot be expressed by looking at the two qubits seperately. There are 4 Bell states, whose statevectors are given as:

- $|\Phi^+\rangle = \frac{|00\rangle+|11\rangle}{\sqrt{2}}$
- $|\Phi^-\rangle = \frac{|00\rangle-|11\rangle}{\sqrt{2}}$
- $|\Psi^+\rangle = \frac{|10\rangle+|01\rangle}{\sqrt{2}}$
- $|\Psi^-\rangle = \frac{|10\rangle-|01\rangle}{\sqrt{2}}$

Additionally, we will compare our results with an uncorrelated (separable) state given by:

- $|\psi\rangle = \mathbf{H}^{\otimes 2}|00\rangle = \frac{|00\rangle+|01\rangle+|10\rangle+|11\rangle}{2}$

Your task will be to complete the code below so that you create all 4 Bell states and the separable state, run them on the IBM backend, and compare the results.

- Complete all sections marked `# YOUR CODE GOES HERE`.
- Do not modify cells marked with `# DO NOT MODIFY` at the top.

1 - Create Bell state circuits

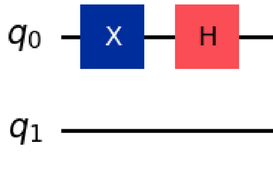
For each case below, use Qiskit to create quantum circuits that recreate the Bell state shown. The first Bell state is shown as an example. Your job will be to fill in the missing parts of the remaining cases.

Case 1: $|\Phi^+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$

In [ ]:
```
# DO NOT MODIFY - Create a quantum circuit and fill in the gates.
phiplus = QuantumCircuit(2)

phiplus.h(0)      # Apply a Hadamrd gate to qubit 0 to put it in superposition.
phiplus.cx(0, 1); # Apply a controlled-X (CNOT) gate from qubit 0 onto qubit 1 to entangle them
```

In [ ]:
```
# DO NOT MODIFY - Plot the circuit
phiplus.draw(output="mpl", idle_wires=False, style="iqp")
```

Out[ ]:



In [ ]:
```
# DO NOT MODIFY - Display the statevector
Statevector(phiplus).draw(output='latex')
```

Out[ ]:

$$\frac{\sqrt{2}}{2}|00\rangle + \frac{\sqrt{2}}{2}|11\rangle$$

Case 2: $|\Phi^-\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}}$

In [ ]:
```
# Create a quantum circuit
phiminus = QuantumCircuit(2)

# YOUR CODE GOES HERE - Fill in the gates in the quantum circuit
phiminus.x(0)
phiminus.h(0)
phiminus.cx(0, 1)
```

Out[ ]:
```
<qiskit.circuit.instructionset.InstructionSet at 0x12baaad70>
```

In [ ]:
```
# DO NOT MODIFY - Plot the circuit
phiminus.draw(output="mpl", idle_wires=False, style="iqp")
```

Out[ ]:

In [ ]:
```
# DO NOT MODIFY - Display the statevector
Statevector(phiminus).draw(output='latex')
```

Out[ ]:

$$\frac{\sqrt{2}}{2}|00\rangle - \frac{\sqrt{2}}{2}|11\rangle$$

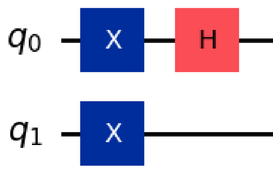
Case 3: $\left|\Psi^{+}\right\rangle = \frac{|10\rangle + |01\rangle}{\sqrt{2}}$

In [ ]:
```
# YOUR CODE GOES HERE - Create a quantum circuit for psiplus and fill in the gates.
psiplus = QuantumCircuit(2)

psiplus.h(0)
psiplus.x(1)
psiplus.cx(0, 1)
```

Out[ ]:
```
<qiskit.circuit.instructionset.InstructionSet at 0x12b96ed10>
```

In [ ]:
```
# Your CODE GOES HERE - Plot the circuit
psiplus.draw("mpl")
```

Out[ ]:



In [ ]:
```
# YOUR CODE GOES HERE - Display the Statevector
Statevector(psiplus).draw("latex")
```

Out[ ]:

$$\frac{\sqrt{2}}{2}|01\rangle + \frac{\sqrt{2}}{2}|10\rangle$$

Case 4: $\left|\Psi^{-}\right\rangle = \frac{|10\rangle - |01\rangle}{\sqrt{2}}$

In [ ]:
```
# YOUR CODE GOES HERE - Create a quantum circuit for psiminus and fill in the gates.
psiminus = QuantumCircuit(2)
psiminus.x(0)
psiminus.h(0)
psiminus.x(1)
psiminus.cx(0, 1)
# psiminus.cz(0, 1)
```

Out[ ]:
```
<qiskit.circuit.instructionset.InstructionSet at 0x12bab05b0>
```

In [ ]:
```
# YOUR CODE GOES HERE - Plot the circuit
psiminus.draw("mpl")
```

Out[ ]:

In [ ]:
```
# YOUR CODE GOES HERE - Display the statevector
Statevector(psiminus).draw("latex")
```

Out[ ]:

$$-\frac{\sqrt{2}}{2}|01\rangle + \frac{\sqrt{2}}{2}|10\rangle$$

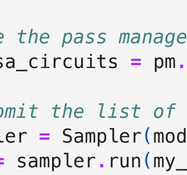
Case 5: Separable State $\mathbf{H}^{\otimes 2}|00\rangle = \frac{|00\rangle+|01\rangle+|10\rangle+|11\rangle}{2}$

In [ ]:
```
# YOUR CODE GOES HERE - Create a quantum circuit for h2 and fill in the gates.
h2 = QuantumCircuit(2)

h2.h(0)
h2.h(1)
```

Out[ ]:
```
<qiskit.circuit.instructionset.InstructionSet at 0x12baaac20>
```

In [ ]:
```
# YOUR CODE GOES HERE - Plot the circuit
h2.draw("mpl")
```

Out[ ]:



In [ ]:
```
# YOUR CODE GOES HERE - Display the statevector
Statevector(h2).draw("latex")
```

Out[ ]:

$$\frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle$$

In [ ]:
```
# Append measurement operations so we can get a result when we run the circuit
phiplus.measure_active()
phiminus.measure_active()

# YOUR CODE GOES HERE - Add measure_active() operations to the rest of the quantum circuits you made.
psiplus.measure_active()
psiminus.measure_active()
h2.measure_active()
```

In [ ]:
```
# YOUR CODE GOES HERE - Put all 5 circuits into a list
my_circuits = [phiplus, phiminus, psiplus, psiminus, h2]
```

2 - Running the circuits and plotting results

Run the circuits and generate plots of the Bell state measurements. You do not need to modify any code in this section, only run it and make sure you obtain 5 plots at the end.

In [ ]:
```
# DO NOT MODIFY - Transpile and run the circuits.

# Use the pass manager to transpile all the circuits together
my_isa_circuits = pm.run(my_circuits)

# Submit the list of circuits we created to a Sampler
sampler = Sampler(mode=backend)
job = sampler.run(my_isa_circuits)
```

In [ ]:
```
# DO NOT MODIFY - Use the job ID to retrieve your job data later
print(f">>> Job ID: {job.job_id()}")
```

```
>>> Job ID: cvzepfsm8yhg0087dxv0
```

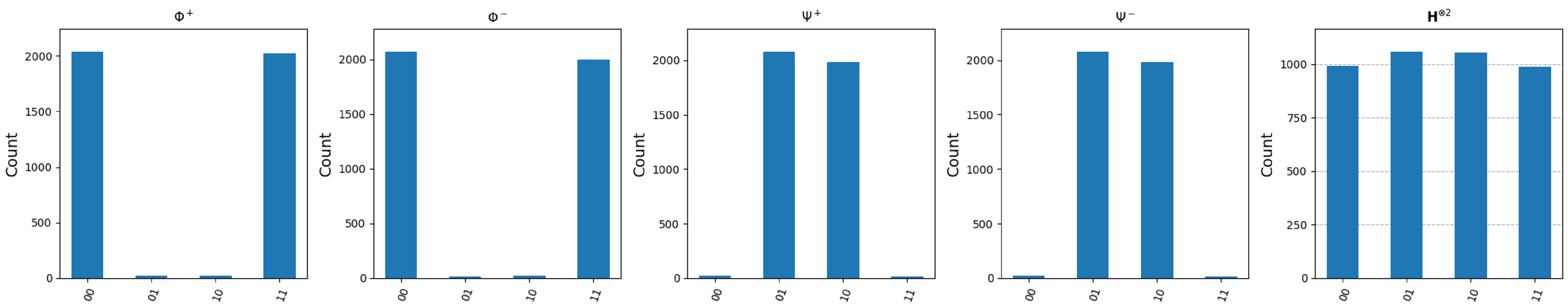
In [ ]:
```
# DO NOT MODIFY - Run this cell to see the status of your job
status = job.status()
print(status)
```

```
DONE
```

In [ ]:
```
# DO NOT MODIFY - Once the job is done, we retrieve the results for
# post-processing.
# Note! If you run this cell before the job is done, it will
# block the execution of any others in the notebook.
results = job.result()
```

In [ ]:
```
# DO NOT MODIFY - We should have 5 histograms from Qiskit's Sampler results.
# Each histogram is a dictionary with outcomes as keys and counts as values.
histograms = []
for result in results:
    counts = result.data.measure.get_counts()
    histograms.append(counts)
```

In [ ]:
```
# DO NOT MODIFY - Plot the results

# Create a figure and a set of subplots
fig, axes = plt.subplots(1, 5, figsize=(20, 4))

# State names for titles
state_names = [r"$\Phi^+$",
               r"$\Phi^-$",
               r"$\Psi^+$",
               r"$\Psi^-$",
               r"$\mathbf{H}^{\otimes 2}$"]

# Loop over the histograms and corresponding axes
for i, (hist, ax) in enumerate(zip(histograms, axes)):
    # Use Qiskit's plot_histogram function
    plot_histogram(hist, ax=ax, bar_labels=False)
    ax.set_title(state_names[i])

# Adjust layout to prevent overlap
plt.tight_layout()

# Display the plots
plt.show()
```



3 - Discussion Questions

Answer each of the following in a few sentences in this markdown cell or create a new one below.

1. Did you have difficulties creating the other three Bell states? What did you have to do to get the statevectors to match?
2. Can you distinguish between the different Bell states based on these measurements alone?
3. How can you tell if noise affected your measurements?
4. What differences would you expect to see between noisy and perfect measurements?
5. What follow-up questions or experiemnts would you propose? (No right or wrong answers. It's ok to bring up anything you found confusing, missing, or things left you courious for more information.)

Problem 3 - Free Response: Qubit Modalities

In *Understanding Quantum Technologies*, navigate to the "Quantum Computing Hardware" chapter and read pages 311-315. Then, select one of the qubit technologies in chapter (use the table of contents to guide you) and read that section. Answer the following questions in 1-2 paragraphs. (Note: You may need to follow the references or do additional searches, as information may have changed.):

1. For the qubit technology you selected, explain qualitatively how the DiVincenzo criteria are satisifed (two levels, initialization, gates, readout, and coherence time)?
2. What peripheral equipment and operating conditions are needed for the qubit method you selected? Discuss the operating temperature and pressure,
3. What is the biggest quantum computer to date that has been implemented using this qubit modality? What labs or companies appear to be leading its development?
4. You are now the founder of a quantum computing startup that specializes in making the qubit modality you selected. How would you persuade prospective users and investors that this qubit type has the best long-term prospects for scaling and commercialization?

Quantum Photonics and the DiVincenzo criteria

The qubit technology I chose is photonics, which uses light to create qubits.

1. The DiVincenzo Criteria focuses on 5 main aspects; scalability, ability to initialize qubits to a fiducial state, long coherence times, a universal set of quantum gates, and the ability to measure qubit states.

Scalability

Quantum photonics has the ability to be scaled up. In a recent paper named "Scalable integrated single-photon source", scalable single photon source qubits are discussed. The way scalability and coherence are meausred is with a boson sampler, and the conclusion of the paper is that it's possible to create a scalable single photon system.

Ability to initialize qubits to a fiducial state

Photonic qubits can be initialized to a fiducial state. A paper named "Comparison Between Different Qubit Designs for Quantum Computing " includes information regarding the initialization procedure for several qubit technologies, and for photonic qubits, it involves controlling the polariziation of the photon source.

Long coherence times

A paper published in Nature named "Decoherence-protected memory for a single-photon qubit" demonstrated that memory could be kept in photonics for upwards of 100ms, which is far more than the microsecond coherence times of superconducting qubits.

Universal set of quantum gates

A universal set of quantum gates could be applied to a photonic system. A paper called "Integrated photonic quantum gates for polarization qubits" demonstrates the uses of a CNOT gate on a photonic system via manipulation of the polarization of the photon source.

Ability to measure qubit states

In the same paper, a set of observables can be measured in a photonic system by receiving the wavepacket of the photons and plotting it out, much like a superconducting qubit system.

Peripheral Equipment and Operating Conditions

As outlined in "Integrated photonic quantum gates for polarization qubits", the operating equipments of a photonic system would include a photon emitter, a way to split the beam, a device for performing quantum gates, and measurement devices. As for operating conditions, photonic systems can operate at room temperature and atmospheric conditions.

Biggest Photonic Quantum Computer to Date

Currently, the bigggest photonic quantum computer is a 20 qubit system made by QuiX Quantum, a company based in Europe. Other companies within the field include Xanadu Quantum Technologies, Orca Computing, and PsiQuantum.

The VC Proposal

Photonics qubits is by far, the best qubit modality for building scalable quantum computing systems. To list all the reasons is to be verbose, but to name a few, its relatively low operating costs, great scalability, long coherence times, and relation to quantum communication all contribute to it being the premier qubit modality in the future.

The equipment necessary for creating a photonics quantum system is quite low. All that's required is a source of polarized photon beam, lensing mirrors, measuring equipment, and some sort of physical representation for quantum gates. There is no need for complicated dilution refrigerators like superconducting qubits or expensive Paul traps like trapped ion qubits. Photonic qubits also offer great scalability. Though the biggest quantum system is built with superconducting qubits (IBM's Condor, sporting 1,121 qubits), photonic systems scale up without issue using photonic chips. Since quantum photonics is manipulated with polarization and lensing, there is no fear that photonic systems would lose coherence as it scales up, unlike superconducting qubits. Photonic qubits also have much longer coherence times when compared to other qubit modalities. Superconducting qubits for example, have coherence times along microseconds, whereas photonics have conherence times in the 100s of milliseconds. This property of long coherent times also lend to photonic systems being incredibly error resistant, which reduces the need for intense error correction/suppression regimes like superconducting qubits or trapped ion systems. Finally, photonic qubits are used in quantum communication as well, and research into photonic qubits would lead to parallel advancement with quantum communication, essentially killing two birds with one stone.

With all that being said, it's clear the investing into photonic research is not only funding a superior qubit modality, but also allowing for a diverse set of quantum hardware to be developed.