

In [1]:

```
from qiskit import QuantumCircuit, assemble, Aer
from qiskit.visualization import plot_histogram, plot_bloch_vector
from math import sqrt, pi
import qiskit.quantum_info as qi
import numpy as np
```

Problem 1.1

$$\frac{1}{\sqrt{2}}(|0\rangle - i|1\rangle) \otimes \frac{1}{\sqrt{2}}(\langle 0| + i\langle 1|)$$
$$\frac{1}{2} \begin{bmatrix} 1 \\ -i \end{bmatrix} \otimes \begin{bmatrix} 1 & i \end{bmatrix}$$
$$\frac{1}{2} \begin{bmatrix} 1 & i \\ -i & 1 \end{bmatrix}$$

In [2]:

```
qc = QuantumCircuit(1)
initial_state = [1/sqrt(2), -1j/sqrt(2)]
qc.initialize(initial_state, 0)
qc.draw()
```

Out[2]:

$$q - \begin{array}{c} |\psi\rangle \\ [0.707, -0.707j] \end{array} -$$

In [3]:

```
psi = qi.Statevector.from_instruction(qc)
psi.draw('latex', prefix='|\\psi\\rangle = ')
```

Out[3]:

$$\frac{\sqrt{2}}{2}|0\rangle - \frac{\sqrt{2}i}{2}|1\rangle$$

In [4]:

```
rho = qi.DensityMatrix.from_instruction(qc)
rho.draw('latex', prefix='\\rho = ')
```

Out[4]:

$$\rho = \begin{bmatrix} \frac{1}{2} & \frac{1}{2}i \\ -\frac{1}{2}i & \frac{1}{2} \end{bmatrix}$$

I started by declared an empty circuit with 1 qubit. I then create an initial state for a single qubit and gave it the appropriate value for a column vector. After initializing this state, I transferred its data to the qubit in the circuit I declared. I passed this circuit with the single qubit to the 'DensityMatrix.from_instruction()' method in order to create a density matrix from the original column vector. Finally, I displayed the calculated density matrix using the 'draw()' method.

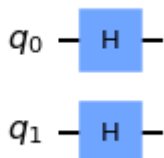
Problem 1.2

$$\begin{aligned}
 & \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) \otimes \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) \\
 & \frac{1}{2} \left(\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \right) \\
 & \otimes \frac{1}{2} \left(\begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \right) \\
 & \frac{1}{2} \left(\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \right) \otimes \frac{1}{2} \left(\begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} \right) \\
 & \frac{1}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}
 \end{aligned}$$

In [5]:

```
qc2 = QuantumCircuit(2)
qc2.h(0)
qc2.h(1)
qc2.draw()
```

Out[5]:



In [6]:

```
psi2 = qi.Statevector.from_instruction(qc2)
psi2.draw('latex', prefix='\\psi = ')
```

Out[6]:

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

In [7]:

```
rho2 = qi.DensityMatrix.from_instruction(qc2)
rho2.draw('latex', prefix='\\rho = ')
```

Out[7]:

$$\rho = \begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{bmatrix}$$

For problem 1.2, I had to declare a circuit with 2 qubits this time. Instead of simply initializing the qubit as I did in 1.1, I performed a Hadamard gate on both qubits to get the desired state. This worked because qubits that are not given a value are automatically initialized to the $|0\rangle$ state. Similarly with problem 1.1, I used the 'DensityMatrix.from_instruction()' method to calculate the density matrix of my state. Finally, I displayed the density matrix using the 'draw()' method.

Problem 2.1

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

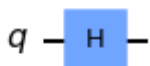
$$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \otimes \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \end{bmatrix}$$

$$\frac{1}{2} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

In [8]:

```
qc3 = QuantumCircuit(1)
qc3.h(0)
qc3.draw()
```

Out[8]:



In [9]:

```
psi3 = qi.Statevector.from_instruction(qc3)
psi3.draw('latex', prefix='\\psi = ')
```

Out[9]:

$$\frac{\sqrt{2}}{2}|0\rangle + \frac{\sqrt{2}}{2}|1\rangle$$

In [10]:

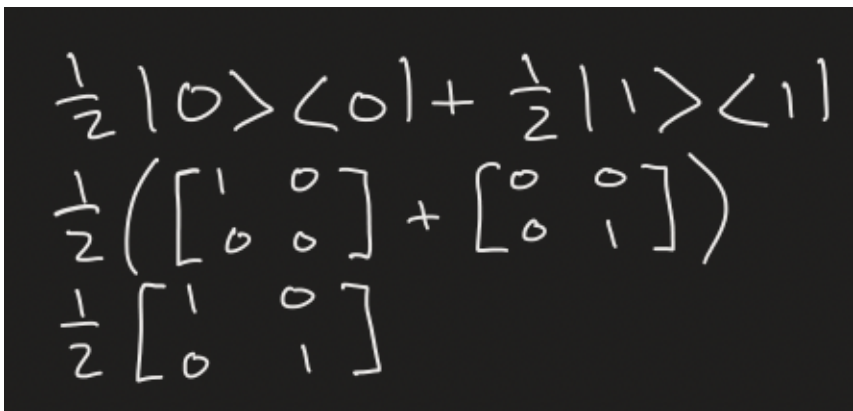
```
rho3 = qi.DensityMatrix.from_instruction(qc3)
rho3.draw('latex', prefix='\\rho = ')
```

Out[10]:

$$\rho = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

Problem 2.1 was similar to 1.2, only a little simpler. I declared a circuit with only 1 qubit, as I did in 1.1. I then performed a Hadamard gate on the only qubit, as I did in 1.2. Again to find the density matrix, I used the 'DensityMatrix.from_instruction()' method. Finally, i used the 'draw()' method once again to display the matrix.

Problem 2.2


$$\begin{aligned} & \frac{1}{2} |0\rangle\langle 0| + \frac{1}{2} |1\rangle\langle 1| \\ & \frac{1}{2} \left(\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \right) \\ & \frac{1}{2} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{aligned}$$

In [11]:

```
rho4 = np.array([[1/2, 0], [0, 1/2]])  
rho4 = qi.DensityMatrix(rho4)  
rho4.draw('latex', prefix='\\rho = ')
```

Out[11]:

$$\rho = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix}$$

Problem 2.2 was a little different. Since I was working with a mixed state as opposed to a pure state, I could not create the density matrix by calculating the outer product. Instead, I had to use the numpy module and directly input the values of the density matrix. After doing this, I passed the array to the 'DensityMatrix()' method and then displayed the matrix using the 'draw()' method.

Problem 3.A

The goal of the T1 experiment is to determine, to some accuracy, the decay rate of an excited qubit toward the ground state. To do this, we take an excited qubit and we create a delay before we measure it. The larger the delay, the more likely the excited qubit has decayed to the ground state. By testing different delays many times, we can observe the likelihood that the qubit has returned to the ground state for each fixed delay period. After gathering the data for each of these delays, we can calculate an estimated decay rate.

We start our experiment with a target T1 taken from the device properties of qubit 0 and three delay intervals of $1e-6$, $3 * \text{the T1}$ we retrieved, and $3e-5$ respectively. For each delay interval, we attempt to measure the qubit. We display a graph showing that as the delay increases, the curve approaches the x-axis.

Next, we attempt to run the same experiment but with two parallel qubits. By doing this, we can see that the result is not always the same even though we set up the experiment the same. When we look at the graphs of these two parallel qubits, we see they are very similar, but not identical.

The importance of running this with multiple delays and parallel qubits is that it helps us get a more precise view of what is really happening. To get more accurate data, we would need to run hundred, if not, thousands of tests and find a curve of best fit between them all.

Problem 3.B

The goal of the Calibrating Qubits experiment is to test out Qiskit Pulse, which allows us to choose a pulse without adhering to the hardware and learn how to calibrate qubits.

Before we begin, we set up the dependencies and check if the backend supports Pulse. We then set the backend to use the default.

In order to know how to configure our pulses so that they affect the qubit in the manner we want, we must first find the frequency of the qubit. We retrieve the qubit frequency given by the backend defaults and widen our net by creating a range within 40 MHz of the default frequency. We specify to test this range 1 MHz at a time during our sweep.

Next, we take this range to create a schedule for our pulse sweep. We define the drive pulse, which is a Gaussian pulse, and truncate it since it is not finite. The drive pulse is used at each frequency, after which we measure the pulse. After defining the drive pulse, we define our measurement pulse, and then our frequencies using the range we set up in the previous step. If we want, we can display a visual graph of our schedule to get a better sense of what we have just put together.

While running this schedule once can give us some insight, it would be more useful to run this multiple times and put all the data together. To do this, we define the number of runs or "shots" as 1024. We also declare that our output should be a single value for each frequency, rather than an array of values for each shot at each frequency. Then we run the schedule.

Unfortunately, running this schedule resulted in an error. The error specified was error #9443. Qiskit is handy enough to include a link to their error codes when such errors appear so I traveled to the error site to see what might have been happening. Error #9443 was

not specified in the error list. My best guess is that the error is something behind the scenes that IBM does not feel comfortable sharing with external users. I was not able to run the code after this, but I did continue reading the tutorial and learned about the expected results.

The next step was to display the results on a graph. The middle of the graph was much higher than the outer edges. Overall, the graph appeared to be a wave function with a peak in the middle around 4.9725 GHz. When creating a line of best fit, we could see that the peak was actually closer to 4.97164 GHz. We now knew that frequency of our qubit so next we would determine the strength of a pi pulse, which alternates the qubit between $|0\rangle$ and $|1\rangle$. To do this, we would test a 50 drive pulses spanning from 0 to .75 to see what oscillations we might get.

We ran this with 1024 shots as we did with the previous schedule and displayed the results in a graph. This time, we saw a wave function with 3 distinct peaks, roughly at .15, .45, and .75. Again, we created a line of best fit but this time we measured the distance between the first peak and the first bottom to find the pi amplitude of 0.15537015203563695. Using this amplitude, we defined our pi pulse as its own variable to be used later.

Next, we would create two separate schedules, one with the pi pulse and one without. Since we know that our ground state is $|0\rangle$, we could get a decent picture of the difference between $|0\rangle$ and $|1\rangle$ by using the pi pulse.

We ran this with 1024 shots and display the results on a scatter plot. We defined the ground state points to be blue and the excited state points to be red. When plotted, we could clearly see two separate collections of points focused around two distinct centers. This was a beautiful visualization of our two separate states. We now had a way to classify if any result is closer to $|0\rangle$ or closer to $|1\rangle$.

For the final part of this experiment, we were to measure the T1, or decay rate, of our qubit. To do this, we applied a drive pulse, a pi pulse, and a measurement pulse, but this time we added a delay before the measurement so that we could get a sense of the decay. We defined a range of delay times to test in order to see how the decay changes over different periods. This time, we ran our schedule with only 256 shots. With the results plotted, we could clearly see a downward curve approaching the x-axis as the delay increased. Finally, we created a line of best fit for this data to better represent the result.

In []: