Paired Workshop Tasks

PHYS:4340L Matthew Jarek

---

1. Quick M1 Review Instructor Demo

In [2]:
```
# M1 Refesher: Basic qubit probability calculation
import numpy as np

shots = 1024
outcomes = np.random.choice([0,1], size = shots, p=[.7,.3])
counts = np.bincount(outcomes)

print(f"Theoretical: P(|0>)=70%, P(|1>) =30%")
print(f"Observed: P(|0>)={ counts [0]/ shots :.1%}, P(|1>)={ counts [1]/
print(f"Counts: |0>={ counts [0]}, |1>={ counts [1]}")
```

```
Theoretical: P(|0>)=70%, P(|1>) =30%
Observed: P(|0>)=70.2%, P(|1>)=29.8%
Counts: |0>=719, |1>=305
```

2. Paired Workshop Tasks

Task 1: Arrays for Qubit Probabilities (10 min)

Complete the code below Expected probs ≈ 70%/30%

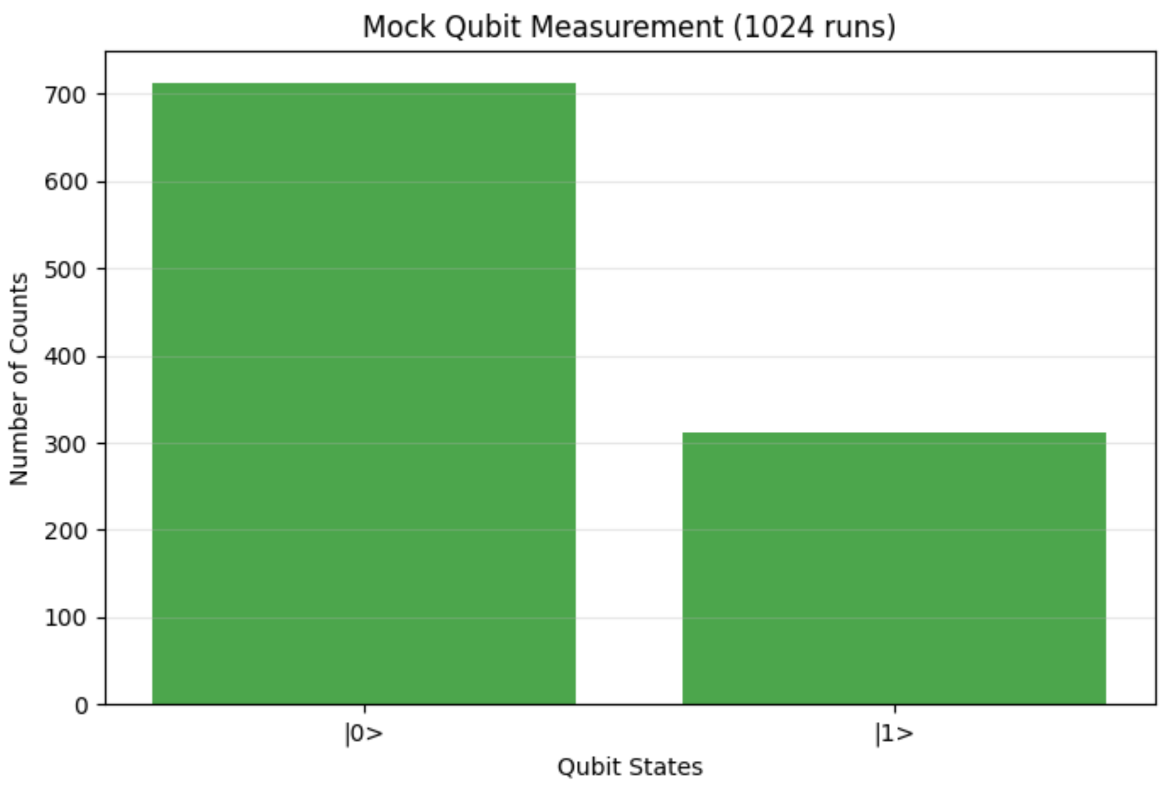
In [27]:
```
import numpy as np

# Task 1: Sumulate biased qubit (70% |0>, 30% |1>)
shots = 1024
outcomes = np.random.choice([0,1], size = shots, p=[.7,.3])
unique, counts = np.unique(outcomes, return_counts=True)

prob_0 = counts[0]/shots
prob_1 = counts[1]/shots

print(f"Probabilities: P(|0>) ={prob_0 : .2%},   P(|1>) ={prob_1 : .2%}")
print(f"Counts        :    |0>  = {counts[0]},         |1>  = {counts[1]}")
```

```
Probabilities: P(|0>) = 69.63%,   P(|1>) = 30.37%
Counts        :    |0>  = 713,         |1>  = 311
```

Task 2: Visualize Quantum Data (15 min)

Create a bar plot like Qisket histograms. Use counts from Task 1.

In [ ]:
```
import matplotlib.pyplot as plt

# Task 2: Bar plot of shot counts
plt.figure(figsize=(8,5))
plt.bar(['|0>','|1>'], counts, alpha=0.7, color = 'g')
plt.xlabel('Qubit States')
plt.ylabel('Number of Counts')
plt.title("Mock Qubit Measurement (1024 runs)")
plt.grid(axis='y', alpha = 0.3)
plt.show()
```

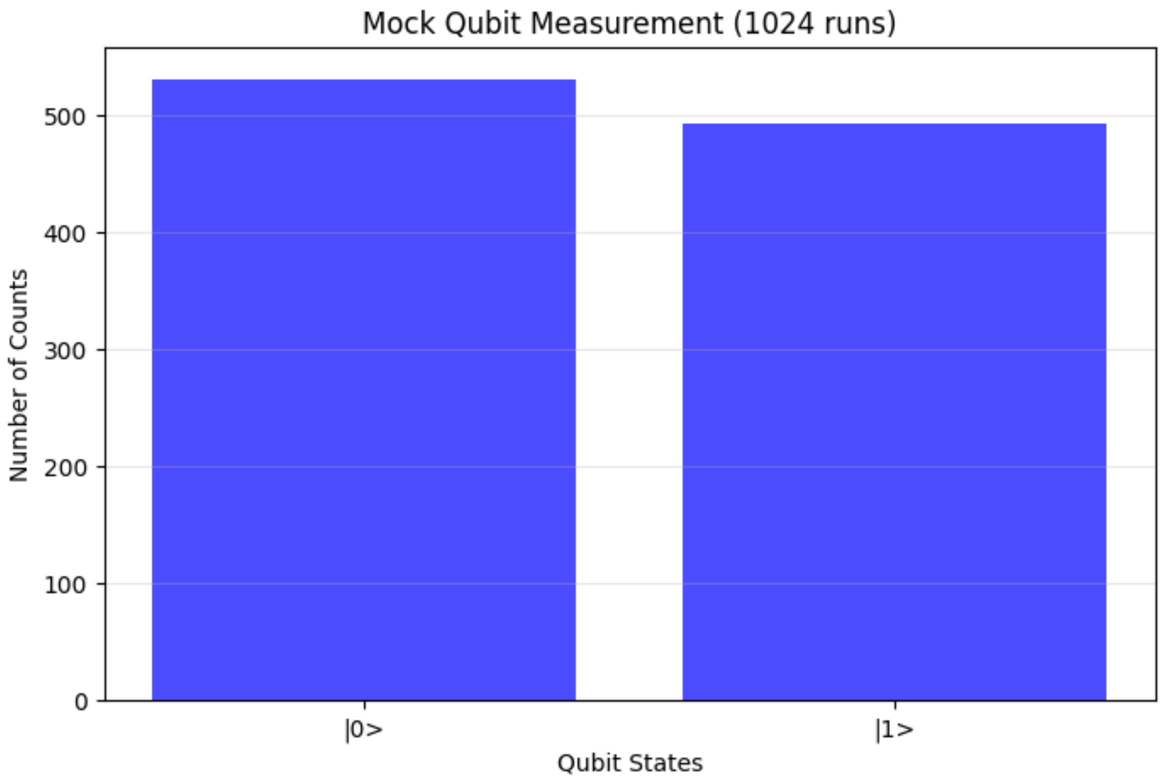


Bonus

Rerun Task 1 with new probabilities p=[.5,.5].

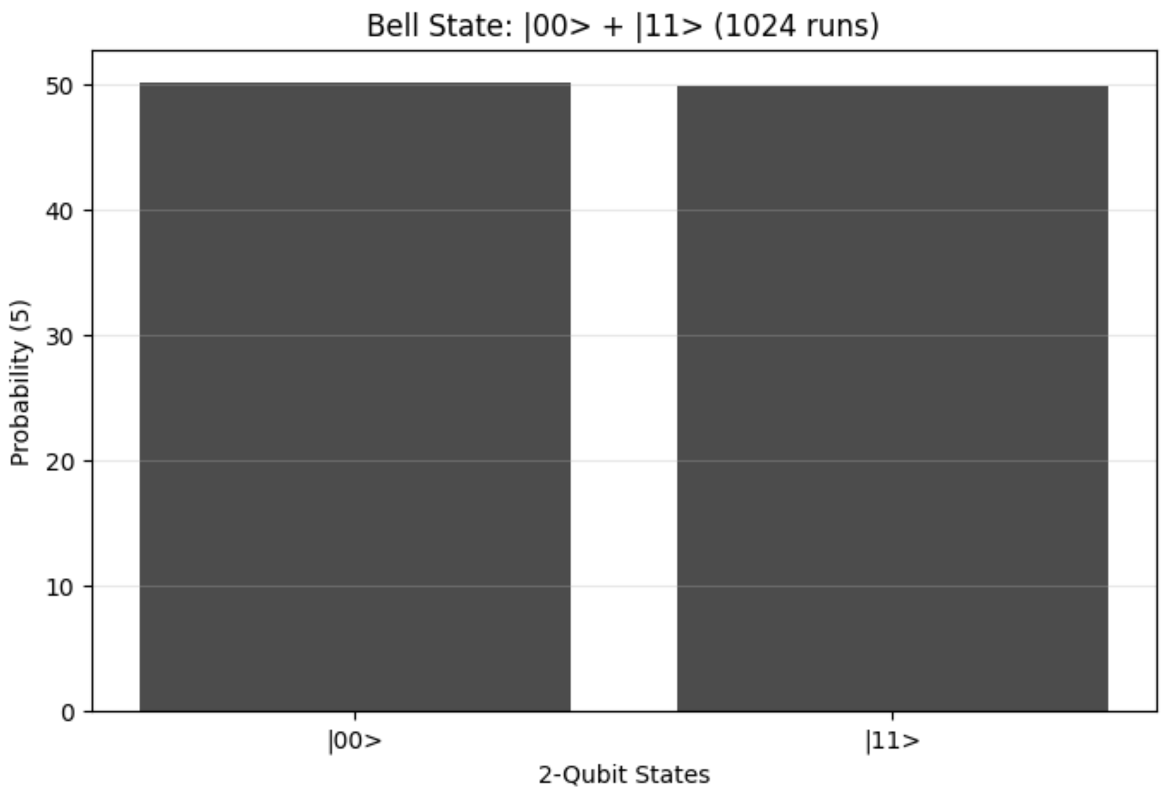
In [34]:
```
import numpy as np
import matplotlib.pyplot as plt

# Chage Sumulate to nonbiased qubit (50% |0>, 50% |1>)
shots = 1024
outcomes = np.random.choice([0,1], size = shots, p=[.5,.5])
unique, counts = np.unique(outcomes, return_counts=True)

prob_0 = counts[0]/shots
prob_1 = counts[1]/shots

print(f"Probabilities: P(|0>) ={prob_0 : .2%},   P(|1>) ={prob_1 : .2%}")
print(f"Counts        :   |0>  = {counts[0]},        |1>  = {counts[1]}")

# Bar plot of shot counts
plt.figure(figsize=(8,5))
plt.bar(['|0>','|1>'], counts, alpha=0.7, color = 'b')
plt.xlabel('Qubit States')
plt.ylabel('Number of Counts')
plt.title("Mock Qubit Measurement (1024 runs)")
plt.grid(axis='y', alpha = 0.3)
plt.show()
```

```
Probabilities: P(|0>) = 51.86%,   P(|1>) = 48.14%
Counts        :   |0>  = 531,        |1>  = 493
```



Task 3: Multi-Qubit Preview (15 min)

```
In [48]: # Task 3: 2-qubit correlated measurements
         # Get the measurements
         runs = 1024
         measurements = np.random.choice(['|00>', '|11>'], size = runs, p=[0.5,0.5
         unique, counts = np.unique(measurements, return_counts = True)

         # Bar plot of shot counts
         plt.figure(figsize=(8,5))
         plt.bar(unique, counts/ runs * 100, alpha=0.7, color = 'black')
         plt.xlabel('2-Qubit States')
         plt.ylabel('Probability (5)')
         plt.title(f"Bell State: |00> + |11> ({runs} runs)")
         plt.grid(axis='y', alpha = 0.3)
         plt.show()

         print("Observed probabilities:")
         for state, count in zip(unique, counts):
             print(f"State {state}: {count:.1%}")
```



```
Observed probabilities:
State |00>: 51400.0%
State |11>: 51000.0%
```

Task 4: Documentation (10 min)

Methods

Simulated measuring biased qibits using `np.random.choice()`

Generated 1024 'measurements'

Used `np.unique()` to seperate the states and count results

Ploted the results using `plt.bar()`

Key Question: Increasing the shots or runs will make the plots more smooth or have the Observed results be more closely matching the Expected value. As the number of runs increases the error of the observed values decreases.

Expected vs Observed

State	Expected	Observed	Notes
|0>	70.0%	70.2%	Slightly above expected
|1>	30.0%	29.8%	Slightly below expected

One-sentence obsevation: The probabilites, while close to the expected outcome will not always result it that exact value, the observed value while different, stays around the expected result and the two results alway equal 1.