

While obtaining measurement outcome probabilities gives us useful information about a qubit's state, we're often interested in other measurable quantities that correspond to something physical, such as its energy. In quantum mechanics, and consequently quantum computing, physical quantities are related to a special type of object called an **observable**. Observables correspond to **Hermitian matrices** whose eigenvalues represent the possible values of the measurement outcome.

Tip. A matrix B is **Hermitian** if $B = B^\dagger$. Hermitian matrices have real eigenvalues, which is a sensible property for measurable physical quantities to have.

As an example, let us consider the Pauli Z operation. You can check that

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

is Hermitian, and has eigenvalues 1 and -1 with corresponding eigenvectors, or eigenstates $|0\rangle$ and $|1\rangle$. Similarly, Pauli X is Hermitian and has eigenvalues 1 and -1 , but the associated eigenstates are $|+\rangle$ and $|-\rangle$. If we were to "measure the observable Pauli Z " (or Pauli X) for a given qubit state, we would obtain one of the two possible eigenvalues. (This is why in the earlier nodes, we computed the eigenvalues of some of these operations!) Furthermore, following the measurement, the qubit's state will be that of the eigenvector of the corresponding measurement outcome.

(b) For each outcome, what state is the qubit in after the measurement? Express this in the computational basis.

▼ **Solution.**

The possible outcomes are the associated eigenvectors,

$$\begin{aligned} |v_1\rangle &= -0.61541221i|0\rangle + 0.78820544|1\rangle, \\ |v_2\rangle &= 0.78820544|0\rangle - 0.61541221i|1\rangle. \end{aligned}$$

(c) What is the expectation value of B computed analytically?

▼ **Solution.**

The analytical expectation value is

$$\langle\psi|B|\psi\rangle = \frac{1}{25}(4 - 3e^{-i\pi/3}) \begin{pmatrix} 1 & -2i \\ 2i & 2 \end{pmatrix} \begin{pmatrix} 4 \\ -3e^{i\pi/3} \end{pmatrix} = -0.302769.$$

(d) Suppose we prepared this state and measured B 1000 times. 54 of the trials yielded the larger of the two possible measurement outcomes, and the remaining tries yielded the smaller one. What is the experimentally-obtained expectation value of B ?

In order to get a clearer picture of the value of an observable for a given qubit state, we need to measure its **expectation value**. This is essentially the weighted average of what we would see over many experiments.

Exercise I.10.1. Suppose we measure an observable B for which there is a 70% chance we obtain eigenvalue 1, and 30% chance we obtain the eigenvalue -1 . What is the expectation value of B ?

► **Solution.**

Expressed mathematically, the expectation value of some observable B for a state $|\psi\rangle$ is given by

$$\langle B \rangle = \langle \psi | B | \psi \rangle. \quad (1)$$

Namely, we first compute the result of the matrix B applied to $|\psi\rangle$ followed by its inner product with $|\psi\rangle$.

Exercise I.10.2. Let

$$|\psi\rangle = \frac{4}{5}|0\rangle - \frac{3}{5}e^{i\pi/3}|1\rangle,$$

and suppose we want to measure the observable

$$B = \begin{pmatrix} 1 & -2i \\ 2i & 2 \end{pmatrix}.$$

(a) What are the possible outcomes of the measurement?

▼ **Tip.**

Tip. It is common to specify a measurement by saying "measure in the X basis", or "measure X ". These statements relate directly back to projective measurements and observables respectively. Measuring in the X basis means to take a projective measurement with respect to the basis composed of the eigenvectors of X (these happen to be $|+\rangle$ and $|-\rangle$). The output of this measurement is either $|+\rangle$, or $|-\rangle$. Measuring X , on the other hand, means to measure the observable X , which has eigenvalues $+1$ and -1 , corresponding to those same eigenstates $|+\rangle$ and $|-\rangle$. The output of this measurement is then either $+1$, or -1 . While the two measurements are ultimately the same, the context and language indicate what the output value should look like.

▼ **Solution.**

To determine the possible measurement outcomes, we must compute the eigenvalues of B . They are $\lambda_1 = 3.56155281$ and $\lambda_2 = -0.56155281$. □

□

▼ Solution.

To compute the experimentally obtained eigenvalue, we take the weighted average:

$$\langle \tilde{B} \rangle = \frac{54 \cdot 3.56155281 + 946 \cdot (-0.56155281)}{1000} = -0.338905.$$

The experimentally-obtained value does not exactly match the analytically-obtained one in this case, but will do so in the limit of an infinite number of trials. □

```
import pennylane as qml
from pennylane import numpy as np
```

```
# Sonuçları saklamak için bir dizi oluştur
shot_results = []
```

```
# Kuantum devresini tanımla
```

```
def circuit():
    qml.RX(np.pi/4, wires=0)
    qml.Hadamard(wires=0)
    qml.PauliZ(wires=0)
    return qml.expval(qml.PauliY(0))
```

```
# Farklı atış sayıları
```

```
shot_values = [100, 1000, 10000, 100000, 1000000]
```

```
# Her bir atış sayısı için devreyi çalıştır ve sonuçları sakla
for shots in shot_values:
```

→ Verilen atış sayısı ile bir cihaz oluşturulur

```
# Cihazı oluştur
```

```
dev = qml.device('default.qubit', wires=1, shots=shots)
```

```
# QNode'u oluştur ve devreyi çalıştır
```

```
qnode = qml.QNode(circuit, dev) → Tamamlanan kuantum devresi ve cihaz kullanılarak QNode oluşturulur.
```

```
# Sonucu sakla
```

```
shot_results.append(qnode()) → QNode çalıştırılır ve elde edilen belinen değer 'shot_results' dizisine eklenir.
```

```
# Sonuçları yazdır
```

```
print(qml.math.unwrap(shot_results))
```

→ Pennylane'inunwrap fonksiyonunun kullanılarak
belinen değerler diziye numpy dizisine dönüştürülür ve yazdırılır.

NDJ, Atış sayısı arttıkça, belinen değer de her kezde hale gelir ve tekrar olurken
belinen değerde yoktur.

İstatistiksel yaklaşımı, yüksük atış sayılarında sonuçlar daha az genitöz (j)

ve dekor kullanır. Bu, gerekli donanımlar, yapılan ölçümlerin istatistiksel doğruluğunu sağlar.

```
def compute_expval_from_samples(samples):
    """Compute the expectation value of an observable given a set of
    sample outputs. You can assume that there are two possible
    outcomes,
    1 and -1.

    Args:
        samples (array[float]): 100000 samples representing the
    results of
        running the above circuit.

    Returns:
        float: the expectation value computed based on samples.
    """
    estimated_expval = 0
```

\rightarrow Basitçe bu belli bir değer D'yi tahmin etmek.

```
#####
# YOUR CODE HERE #
#####
k = np.array(samples)  $\rightarrow$  'samples' vektörü Numpy dizisine dönüştürülür
estimated_expval = np.average(k)  $\rightarrow$  Numpy kütüphanesi, örneklerin ortalaması hesaplanır. Bu belli bir değer tahmini.
# USE THE SAMPLES TO ESTIMATE THE EXPECTATION VALUE
return estimated_expval  $\rightarrow$  Hesaplanan belli bir değer dönür.
```

The variance of Simple measurements

Önek seferi ve belli bir değer; Her seferinde kodu çalıştırıldığınızda farklı bir önek seti elde edersiniz ve bu farklı bir seferde belli bir değer tahmini sağlar.

Zehirli bir atış sayısına (shots) sahip olduğunuzda, belli bir değer ne kadar doğru olacağın tahmin edebilirsiniz?

Varyansın ölçülmesi; Varyansın atış sayısına göre nasıl değiştığıni inceleyelim.

```
import pennylane as qml
from pennylane import numpy as np
import matplotlib.pyplot as plt

def variance_experiment(n_shots):
    """Run an experiment to determine the variance in an expectation
    value computed with a given number of shots.

    Args:
        n_shots (int): The number of shots

    Returns:
        float: The variance in expectation value we obtain running the
    circuit 100 times with n_shots shots each.
    """

```

\rightarrow Zehirli bir atış sayısına (n_shots) bir deneyin 100 kez çalıştırılması sonucunda elde edilen belli bir değerlerin varyansının nasıl değiştiğini inceleyelim.

```
# To obtain a variance, we run the circuit multiple times at each
shot value.
```

n_trials = 100 \rightarrow Deney sayısını belirt.

samples = [] \rightarrow Ölüm sonuçlarını saklamak için boş bir liste

```
# CREATE A DEVICE WITH GIVEN NUMBER OF SHOTS
dev = qml.device('default.qubit', wires=1, shots=n_shots)
```

\rightarrow Belirli sayıda bir atış ile akıllı cihazın düzgünnesini tanımır.

```
# DECORATE THE CIRCUIT BELOW TO CREATE A QNODE
```

\rightarrow qml.Qnode(dev) \rightarrow circuit1 fonsiyonunu bir QNode (Quantum Deresi) olarak tanımlar.

```

def circuit():
    qml.Hadamard(wires=0)
    return qml.expval(qml.PauliZ(wires=0)) → Pauli Z gözlemebilirliğin beklenen değerini döner.

```

RUN THE QNODE N_TRIALS TIMES AND RETURN THE VARIANCE OF THE RESULTS

```

for i in range(n_trials): → Deneyi 100 kez çalıştırır
    sample = circuit() → Deneyi çalıştırır, sonuç alır.
    samples.append(sample) → Sonuç histogram'e ekler.

```

$k = \text{np.var}(\text{np.array(samples)})$ → Numpy kullanılarak örneklerin varyansını hesaplar.
return k → Hesaplanan varyansı döner.

```

def variance_scaling(n_shots): → Atış sayısına bağlı olarak beklenen değerin varyansını programatik olarak temsil eden fonksiyon.
    """Once you have determined how the variance in expectation value
    scales
        with the number of shots, complete this function to
    programmatically
        represent the relationship.

```

Args:

n_shots (int): The number of shots

Returns:

```

        float: The variance in expectation value we expect to see when
    we run
        an experiment with n_shots shots.
    """

```

return $1/n_{\text{shots}}$ → Atış sayısına bağlı olarak beklenen varyansı hesaplar ve döner. Bu, varyansın atış sayısının tersine orantılı olduğunu gösterir.

Various numbers of shots; you can change this

shot_vals = [10, 20, 40, 100, 200, 400, 1000, 2000, 4000]

Used to plot your results

```

results_experiment = [variance_experiment(shots) for shots in
shot_vals] → Her atış sayısının 'variance-experiment' fonksiyonu çalıştırılarak elde edilen sonuçlar
results_scaling = [variance_scaling(shots) for shots in shot_vals] → Her atış sayısının 'variance-scaling' fonksiyonu çalıştırılarak elde edilen sonuçlar.

```

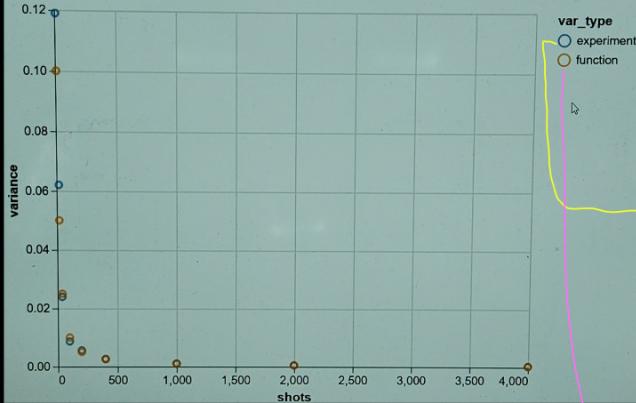
Plotting function

```

def plotter(shot_vals, results_experiment, results_scaling):
    plt.figure(figsize=(10, 6))
    plt.plot(shot_vals, results_experiment, 'o-', label='experiment',
color='blue')
    plt.plot(shot_vals, results_scaling, 'o-', label='function',
color='orange')
    plt.xlabel('shots')
    plt.ylabel('variance')
    plt.xscale('log')
    plt.yscale('log')
    plt.legend() → grafik üzerinde gelen verilen anakutamelere eklenerek tüm kütüphaneler. Bu nedenle bu kutucuk hemen silin.
    plt.grid(True) → Grafik üzerinde bir logaritik (grid) göstergesi sunulmalıdır.
    plt.show()

```

plotter(shot_vals, results_experiment, results_scaling) → Sonuçları karşılaştırın.



Varyans → Beklenen değerin varyansını gösterir. Yani en iyi tahmindeki varyans, bu atış sayısının tersine orantılı olarak azalır.
Deneyel olursa elde edilen, her bir atış sayısının deneyinin 100 kez çalıştırılmıştır sonuçları elde edilen varyanslardır.

Teorik varyans neye lesin temsil etti. Varyansın atış sayısını göre nasıl değiştiğini gösterir.

Varyansın $\frac{1}{n_{\text{shots}}}$ ile ilişkili olduğunu

Varyans / M.J.T.Y

Atış sayısının \uparrow varyansı \downarrow → Daha fazla atış yapıldığında daha fazla sonuç, daha karlı hale gelir ve varyansı.

Deneyel ve teorik varyans (deneyel)

