MULTI-QUBIT SYSTEM

Tensor Product;

Single qubit states live in a Hilbert Space, which is a 2-dimensional vector space spanned by basis vectors $|0\rangle$ and $|1\rangle$

Hilbert Uzayı; kuantum mekaniğinde kullanılan soyut, matematiksel yapı. Sonsuz boyutlu olabilir, ancak tek qubit için 2 boyutlu olan bir vektör uzayı.

Bu uzay → kuantum durumlarının matematiksel olarak temsil edildiği yer.

Baz vektörleri; Bu iki boyutlu uzayın baz vektörler $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ kuantum mekaniğinde bir qubitin iki olası durumunu ifade eder.

Hilbert Uzayındaki Durumlar; Bir kuantum durumu, bu baz vektörlerinin doğrusal kombinasyonu olarak ifade edilir. $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$

Bu genlikler **Born Kuralına Göre** kuantum durumunun ölçüm olasılıklarını verir ve şu koşulu sağlar;

} kuantum durumunun genliklerini temsil eden karmaşık sayılar.

$$|\alpha|^2 + |\beta|^2 = 1$$

Bloch Sphere; Kubitin durumu, kürenin yüzeyinde bir nokta olarak gösterilir ve bu nokta, iki baz vektörün ($|0\rangle$ ve $|1\rangle$) bir kombinasyonudur.

When we put more than one qubit together → How these vector spaces are composed.

Hilbert Spaces are combined using → **Tensor Product**

$$\begin{pmatrix} a \\ b \end{pmatrix} \otimes \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} a\begin{pmatrix} c \\ d \end{pmatrix} \\ b\begin{pmatrix} c \\ d \end{pmatrix} \end{pmatrix} = \begin{pmatrix} ac \\ ad \\ bc \\ bd \end{pmatrix}$$

Sadece Quantum State'ler için geçerli değil. Aynı zamanda bu durumlar üzerinde işlem yapan Unitary İşlemler için de geçerli

Unitary İşlemler; kuantum sistemlerinde durumların evrimini temsil eder ve bu işlemler daima tersinir (invertable) ve norm koruyucu (norm-preserving) olmalı

Tensor Product; İki kuantum sistemini birleştirerek

$UU^{\dagger} = I$

matrisin hermisyen eşleniği
(Hermitian conjugate)

daha büyük bir kuantum sis. kurmak için kullanılır.

$|\psi\rangle \otimes |\phi\rangle$

Qubit Ordering Convention;

$|10100\rangle \rightarrow$

top of the circuit

$|1\rangle$ ———
$|0\rangle$ ———
$|1\rangle$ ———
$|0\rangle$ ———
$|0\rangle$ ———

Sen library'den library'e değişir baştanddan check et.

```
dev = qml.device('default.qubit', wires=3)

@qml.qnode(dev)
def circuit(feature_vector):
    qml.BasisEmbedding(features=feature_vector, wires=range(3))
    return qml.state()

X = [1,1,1]
```

The resulting circuit is:

```
>>> print(qml.draw(circuit, expansion_strategy="device")(X))
0: ──X─┤  State
1: ──X─┤  State
2: ──X─┤  State
```

Multi Qubit Bases;

$|10\rangle \Rightarrow 2_2$

$|111\rangle = 7_2$

$2^2 \; 2^1 \; 2^0$

$4+2+1 = 7.$

$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \end{pmatrix} = |00\rangle$

Bütün vectörlerde tek bir 1 var.

Bu vektörlerin bir temel oluşturduğunu açıkça gösterir.

$$\begin{matrix} 0 & 0 \\ 1 & 1 \end{matrix} \qquad \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} |10\rangle$$

Bu temel vektörlerin doğrusal kombinasyonlarıyla herhangi bir 4X1 komplex değerli vektör oluşturulabilir, yani bu vektörler, dört boyutlu uzayda bir temel oluşturur ve herhangibir vektör bu baz vektörlerin doğrusal kombinasyonu olarak yazılabilir.

$$\begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix} \qquad \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

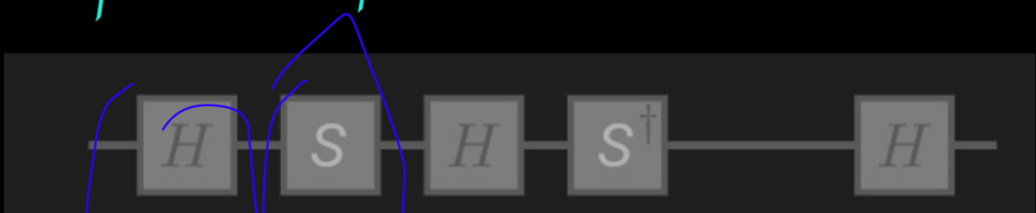
$$\begin{matrix} 0 & 1 \\ 1 & 0 \end{matrix} \qquad \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

Everytime when a new qubit added we multiple w/ 2 so n qubit systems has 2^n basis vectors w/ 2^n elements. → To operate on such basis vectors, we'll need **unitary matrices size of $2^n \times 2^n$** → This exponantional dependence means that simulating quantum com. will become computationally intractable very quickly.

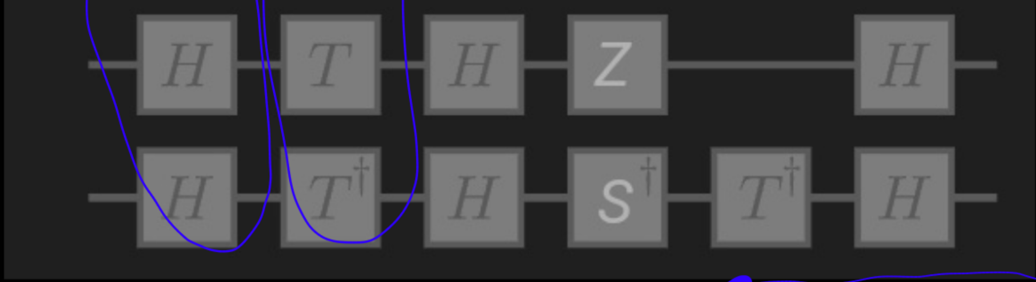
Exponansiyel Bağımlılık; Bir sistemin büyüklüğü arttıkça gereken hesaplama kaynaklarında (zaman, hafıza, işlemci gücü) eksponansiyel olarak ↑ yani qubit sayısı arttıkça, klasik bilgisayarları bu kuantum sistemlerini simüle etmesi giderek daha zor ve zaman alıcı hale gelir.

$$|+\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Seperatable Operations;



→ Bu devrenin (circuit) Her bir wire'ını



(katmanı) single qubit operation gibi düşünebilirsin.

For example, the first step in the circuit is to apply a Hadamard to each qubit. This corresponds to applying $H \otimes H \otimes H$. The second layer applies $S \otimes T \otimes T^\dagger$, and so on. While circuits are read from left to right, the matrices are applied to the states from right to left, i.e., the result of applying the first two layers of gates to an input state $|\psi\rangle$ would be $(S \otimes T \otimes T^\dagger)(H \otimes H \otimes H)|\psi\rangle$.

Exercise I.11.4. Suppose we have two tensor product operations, $A \otimes B$, and $C \otimes D$. A cool fact is that if we multiply these two together, the multiplication works on each side of the tensor product independently, i.e., $(A \otimes B) \cdot (C \otimes D) = (AC) \otimes (BD)$. If you like, you can take a crack at proving this mathematically. However, there is a more intuitive way of understanding this: using a circuit diagram and your knowledge of how single- and two-qubit operations work, reason why this identity must hold.

▼ *Hint.*

Start by drawing the quantum circuit that applies $A \otimes B$ followed by $C \otimes D$, then look at the operations acting on the individual qubits.

Use PennyLane to create the state $|+1\rangle = |+\rangle \otimes |1\rangle$. Then, return two measurements:

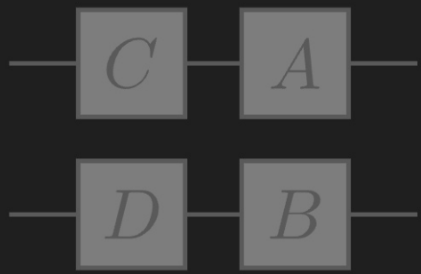
- the expectation value of Y on the first qubit
- the expectation value of Z on the second qubit

In PennyLane, you can return measurements of multiple observables as a tuple, as long as they don't share wires.

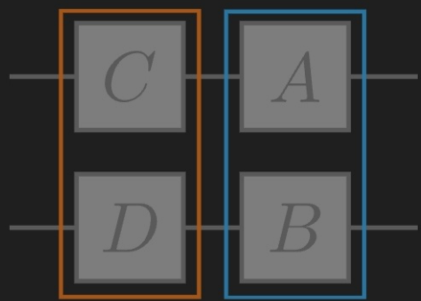
```
import pennylane as qml
from pennylane import numpy as np

# İki kübitli bir cihaz oluşturur
dev = qml.device('default.qubit', wires=2)

@qml.qnode(dev)
def two_qubit_circuit():
    # |+>|1> durumunu hazırlar
    qml.Hadamard(wires=0)
    qml.PauliX(wires=1)

    # İki beklenen değeri döndür: ilk kübit için Y, ikinci kübit için Z
    k = qml.expval(qml.PauliY(wires=0))
    m = qml.expval(qml.PauliZ(wires=1))
    return [k, m]

# Devreyi çalıştır ve sonuçları yazdır
print(two_qubit_circuit())
```

→ Beklenen değerler k ve m değişkenine atanır ve liste olarak döndürülür

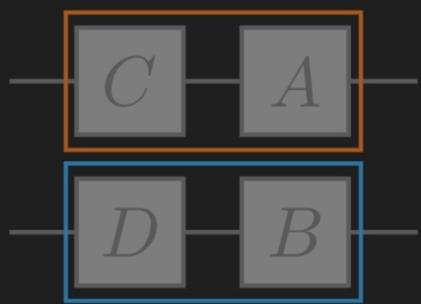
Let's start by drawing what this would look like as a quantum circuit:



We can think of it as $(A \otimes B) \cdot (C \otimes D)$ like so:



However, since these are separable operations, the qubits are acted on independently of each other, so we could just as well conceptually group the operations together like so:



Each qubit has two unitary operations acting on it; applying C then A to the first qubit is equivalent to applying the combined operation AC, and similarly for the second qubit. Thus, we can rewrite this circuit as

```
print(two_qubit_circuit())
```

User output

```
[tensor(0., requires_grad=True), tensor(-1., requires_grad=True)]
```

Bu çıktıda iki tensör nesnesi var. Her biri beklenen değeri (expectation value) temsil ediyor.

1) İlk tensör

↳ İlk kübit (wire-0) üzerinde Pauli operatörünün beklenen değerini temsil eder. 0.0

"requires_grand = True" → Bu tensörün geri yayılım (backpropagation) için gradyan hesaplamaları desteklediğini belirtir. → Quantum devrelerini optimize ederken ve makine öğrenmesi uygulamalarında önemli.

2. **İkinci Tensor (tensor(-1., requires_grad=True))**
   - Bu, ikinci kübit (wire 1) üzerinde Pauli-Z operatörünün beklenen değerini temsil eder.
   - Beklenen değerin (-1.0) olduğunu gösteriyor.
   - Yine, `requires_grad=True` ifadesi, bu tensorun gradyan hesaplamaları için uygun olduğunu belirtir.

which we can now see to be the tensor product $AC \otimes BD$. □

Pauli-Y operatörü bu süperpozisyon durumunda sıfır beklenen değerine sahiptir.

Pauli-Z operatörünün beklenen değeri (-1.0) olur çünkü Pauli-Z operatörü $|1\rangle$ durumunda (-1) beklenen değerine sahiptir.

Her iki tensorun da `requires_grad=True` olması, bu tensorların gradyan hesaplamaları için uygun olduğunu ve potansiyel olarak optimizasyon süreçlerinde kullanılabileceğini belirtir.

Geri Yayılım (Backpropagation)

Geri yayılım, bir yapay sinir ağında hatayı minimize etmek için kullanılan bir algoritmadır. Bu algoritma, ağın çıktısı ile hedef değerler arasındaki hatayı hesaplar ve bu hatayı azaltmak için ağın ağırlıklarını geri yayarak günceller. Bu süreç, gradyan inişi (gradient descent) algoritmasının bir parçasıdır.

Gradyan Hesaplamaları

Gradyan, bir fonksiyonun değişim oranını gösteren vektördür. Gradyan hesaplamaları, fonksiyonun minimum veya maksimum noktalarını bulmak için kullanılır. Geri yayılımda, gradyanlar hata fonksiyonunun türevlerini temsil eder ve bu türevler, ağın ağırlıklarını güncellemek için kullanılır.

Kuantum Hesaplamada Geri Yayılım ve Gradyan Hesaplamaları

Kuantsal devrelerde, özellikle parametrik kuantum devrelerinde, parametreleri optimize etmek için geri yayılım ve gradyan hesaplamaları kullanılabilir. Bu optimizasyon, kuantum devrelerinin belirli bir hedef fonksiyonu minimize edecek şekilde ayarlanmasını sağlar.

`requires_grad=True`

Bu ifade, bir tensorun gradyan hesaplamaları için uygun olduğunu belirtir. Pennylane ve benzeri kuantum hesaplama kütüphanelerinde, bu ifade ile bir değişkenin veya tensorun optimizasyon süreçlerinde kullanılabileceğini belirtiriz.

Expectation Value of two qubit Observable

Write a PennyLane circuit that creates the state $|1-\rangle = |1\rangle \otimes |-\rangle$. Then, measure the expectation value of the *two-qubit observable* $Z \otimes X$. In PennyLane, you can combine observables using the `@` symbol to represent the tensor product, e.g., `qml.PauliZ(0) @ qml.PauliZ(1)`.

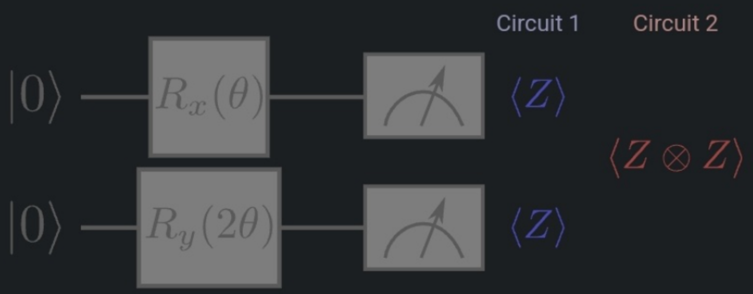
```python
import pennylane as qml

dev = qml.device("default.qubit", wires=2)

@qml.qnode(dev)
def create_one_minus():
    ##################
    # YOUR CODE HERE #
    ##################

    # PREPARE |1>|->
    qml.PauliX(wires=0)
    qml.PauliX(wires=1)
    qml.Hadamard(wires=1)
    # RETURN A SINGLE EXPECTATION VALUE Z \otimes X

    return qml.expval(qml.PauliZ(0) @ qml.PauliX(1))
```

→ Bunu kullanmassın çünkü durumun beklenen değeri

```python
print(create_one_minus())
```

→ [illegible]
hesaplıyorsun ya hani 0 yüzden? Bu devredeki ölçüm sonuçlarını döndürür. Bu, tek bir ölçüm yapmaktan daha güvenilir ve istatistiksel olarak anlamlı sonuçlar elde etmemizi sağlar.

Implement the following circuit twice. For one version, measure the observables Z on the first qubit (i.e., $Z \otimes I$), and Z on the second qubit ($I \otimes Z$). For the other version, measure the observable $Z \otimes Z$. How do think the results of the first circuit will relate to those of the second? Plot the results as a function of θ to test your hypothesis.



Tip. In PennyLane, you don't need to specify the identity portion of observables. For example, $I \otimes Z$ is simply `qml.PauliZ(1)` rather than `qml.Identity(0) @ qml.PauliZ(1)`.

▼ *Hint.*

Consider how the observables themselves are related. If you have $Z \otimes I$ and $I \otimes Z$, can you produce $Z \otimes Z$?

→ Her iki qubit için Pauli-Z operatörünün beklenen değerlerini döndürür.

→ Bu fonksiyon circuit_1 devresinden elde edilen ZI ve IZ sonuçlarını alır ve bu sonuçları çarparak ZZ sonuçlarına eşdeğer olan sonuçları üretir.

All Tied Up

Mathematics of Entangled State;

Ayrılabilir Çoklu Qubit Operasyonları; Başlangıçta, çoklu kubit sistemlerinde tartışılan operasyonlar ayrılabilirdir. Yani her qubit, diğer qubitlerle etkileşime girmeyen tek qubit kapıları tarafından işlenir. Bu her qubitin durumunun diğerlerinden bağımsız olarak evrildiği en basit çoklu qubit işlemidir.

Dolanıklık

İki veya daha fazla kubitin quantum durumları birbirleriyle bağımlı hale gelir → Bir qubitin durumu mesafeye bakılmaksızın anında diğer qubitin durumunu etkilemesidir.
Bu yerel olmayan korelasyon, klasik sistemlerin kopyalayamayacağı temel bir özelliktir.

Kuantum Bilgi Teleportasyonudur; Burada dolanıklık, kübitler arasında kuantum bilgisini fiziksel olarak taşımadan transfer etmek için kullanılır.

Non-local corelation; Kuantum mekaniğinde, birbirleriyle dolanık olan parçacıklar arasında meydana gelen ve bu parçacıklar arasındaki mesafeye bakılmaksızın anında gerçekleşen bağlantıyı ifade eder.

Yerel Korelasyon vs Yerel Olmayan Korelasyon

- **Yerel Korelasyon**: Bu tür korelasyon, klasik sistemlerde gözlemlenir. İki parçacık arasındaki ilişki, doğrudan fiziksel temasa veya yakın mesafeye dayanır. Örneğin, bir çift zar attığınızda, bir zarın sonucu diğerinden bağımsızdır ve herhangi bir korelasyon varsa, bu korelasyon her iki zarın atılış şekli ve yüzey özellikleri gibi yerel etkileşimlerden kaynaklanır.
- **Yerel Olmayan Korelasyon (Non-local Correlation)**: Bu, kuantum mekaniğinde gözlemlenen ve parçacıkların birbirinden ne kadar uzakta olduğuna bakılmaksızın geçerli olan korelasyondur. İki dolanık parçacık, birbirinden çok uzak mesafelerde bile olsa, birinin durumu anında diğerinin durumunu etkileyebilir. Bu, Einstein'ın "spooky action at a distance" (mesafede ürkütücü etki) olarak adlandırdığı olgudur. Bu tür korelasyonlar, klasik fiziğin öngörüleriyle uyumsuzdur ve sadece kuantum dolanıklık ile açıklanabilir.

1. **Bell Durumları**: İki kübitin dolanık olduğu ve bu kübitlerin birinde yapılan bir ölçümün diğer kübitin durumunu belirlediği özel kuantum durumlarıdır.
2. **Kuantum Teleportasyonu**: Bir parçacığın kuantum durumunun başka bir parçacığa aktarılması sürecidir, burada dolanıklık kullanılarak bilgi mesafe tanımaksızın anında iletilir.

Bu yerel olmayan korelasyon, kuantum bilgisayarlarının ve kuantum bilgi işlem teknolojilerinin temelini oluşturur ve klasik sistemlerin kopyalayamayacağı bir avantaj sağlar.