CPEN 400Q Lecture 03
Measurement

Monday 16 January 2023

Announcements

- Quiz 1 today
- Assignment 0 due tonight
- Assignment 1 and literacy assignment 1 coming this week

We learned about the three Pauli rotations

	Math	Matrix	Code	Special cases
RZ	$e^{-i\frac{\theta}{2}Z}$	$\begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix}$	`qml.RZ`	$Z(\pi), S(\pi/2), T(\pi/4)$
RY	$e^{-i\frac{\theta}{2}Y}$	$\begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$	`qml.RY`	$Y(\pi)$
RX	$e^{-i\frac{\theta}{2}X}$	$\begin{pmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$	`qml.RX`	$X(\pi), SX(\pi/2)$

Last time

We saw how qubits can be represented in 3D space on the Bloch sphere, and how unitary operations rotate the Bloch vector.

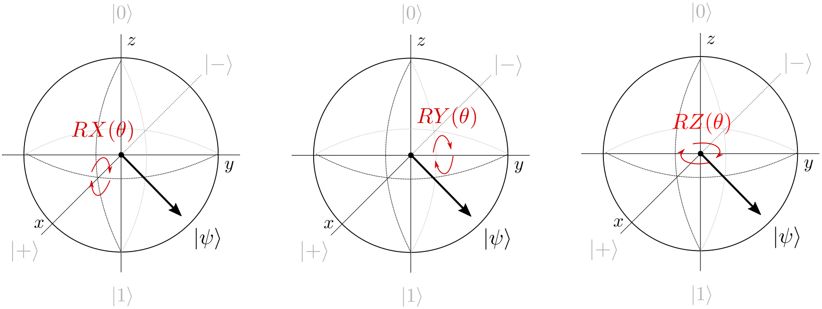


Image credit: Codebook node I.6

Last time

We learned how to implement quantum circuits in PennyLane.

```
import pennylane as qml

dev = qml.device('default.qubit', wires=1, shots=100)

@qml.qnode(dev)
def my_circuit():
    qml.Hadamard(wires=0)
    qml.PauliZ(wires=0)
    qml.PauliX(wires=0)
    return qml.sample()

result = my_circuit()
```

$$|\psi\rangle = e^{i\theta}\alpha|0\rangle + e^{i\theta}\beta|1\rangle \qquad e^{i\theta}\alpha \cdot (e^{-i\theta}\alpha^*) = \alpha\alpha^*$$

We distinguished between two types of phase in a quantum state.

Global phase:

$$|\psi\rangle = e^{i\theta}(\alpha|0\rangle + \beta|1\rangle)$$

Relative phase:

$$|\psi\rangle = \alpha|0\rangle + e^{i\theta}\beta|1\rangle$$

We tried to do the following exercise: Design a quantum circuit to prepare the state

$$|\psi\rangle = \underbrace{\frac{\sqrt{3}}{2}}|0\rangle - \underbrace{\frac{1}{2}}e^{i\frac{5}{4}}|1\rangle$$

$$RY(\theta)|0\rangle = \cos\frac{\theta}{2}|0\rangle + \sin\frac{\theta}{2}|1\rangle$$

$$\theta = -\frac{\pi}{3}$$

$$RY\left(-\frac{\pi}{3}\right)|0\rangle = \frac{\sqrt{3}}{2}|0\rangle - \frac{1}{2}|1\rangle$$

next, do something to change the phase

(see demo: PhaseShift(5/4) or RZ(5/4))

These are the same gate, *up to a global phase*

$$RZ(\theta) = \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix}, \quad RZ'(\theta) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix}$$

$$RZ(\theta)(\alpha|0\rangle + \beta|1\rangle) = e^{-i\theta/2}\alpha|0\rangle + e^{i\theta/2}\beta|1\rangle$$

$$= \underbrace{e^{-i\theta/2}}\,(\alpha|0\rangle + e^{i\theta}\beta|1\rangle)$$

$$\sim \alpha|0\rangle + e^{i\theta}\beta|1\rangle = RZ'(\theta)|\psi\rangle$$

In PennyLane, you can find the latter explicitly as

```
qml.PhaseShift(theta, wires=0)
```

Learning outcomes

- Define a universal gate set
- Compute the inner product between two quantum states
- Perform a projective measurement
- Measure a qubit in different bases
- Measure single-qubit expectation values

General rotations

What about H?

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$$

This does not have the form of RX, RY, or RZ.

But, we can use a combination of these to make an H (actually, just need two of the three).

The $n \times n$ unitary matrices are a mathematical group under matrix multiplication, $U(n)$:

1. Closure: for U, V unitary, UV is also unitary
2. Associativity: $(UV)W = U(VW)$
3. Identity: $\mathbb{1}$
4. Inverses: $U^{-1} = U^\dagger$

Deep dive: unitary operations

The $n \times n$ unitary matrices are a mathematical group under matrix multiplication, $U(n)$:

1. Closure: for U, V unitary, UV is also unitary
2. Associativity: $(UV)W = U(VW)$
3. Identity: $\mathbb{1}$
4. Inverses: $U^{-1} = U^\dagger$

Any unitary matrix can be written in terms of a finite set of real-valued parameters:

$$U(\phi, \theta, \omega) = e^{i\alpha} \begin{pmatrix} e^{-i(\phi+\omega)/2} \cos(\theta/2) & -e^{i(\phi-\omega)/2} \sin(\theta/2) \\ e^{-i(\phi-\omega)/2} \sin(\theta/2) & e^{i(\phi+\omega)/2} \cos(\theta/2) \end{pmatrix}$$

Universal gate sets: Pauli rotations

With just RZ and RY (or RZ/RX, RY/RX), we can implement *any single-qubit unitary operation*[1]:

$$U = e^{i\alpha} RZ(\omega) RY(\theta) RZ(\phi)$$

$\{RZ, RY\}$ is **universal** for single-qubit quantum computing.

Hands-on...

For more fun: do text exercises in Codebook node I.3 and I.7.

[1] Note that the α technically doesn't matter.

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$$

With just H and T, we can approximate any single-qubit rotation up to arbitrary accuracy. For example, we can implement $RZ(0.1)$ up to accuracy 10^{-10}:

$$X = HZH = H(T)^4 H$$

```
→ gridsynth 0.1 -d 10
HTHTHTHTHTSHTHTHTHTSHTSHTHTHTSHTSHTHTSHTHTSHTHTHTSHTSHTHTSHTSHTSHTS
HTHTHTHTHTHTHTHTHTHTHTSHTSHTSHTSHTSHTSHTSHTHTSHTSHTSHTSHTHTHTSHTSHTHT
SHTSHTSHTHTHTHTSHTHTHTSHTSHTHTHTHTHTSHTHTHTSHTSHTSHTSHTSHTHTSHTHTHT
HTHTHTHTHTSHTHTHTSHTHTSHTSHTHTHTSHTSHTHTSHTSHTHXWWW
```

$\sim e^{i\cdots}$

This was generated using the `newsynth` Haskell package:
`https://www.mathstat.dal.ca/~selinger/newsynth/`

Universal gate sets: H and T

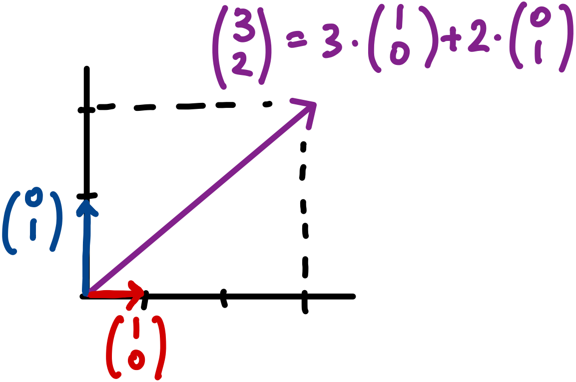
Or to accuracy 10^{-100}:

```
→ gridsynth 0.1 -d 100
SHTSHTHTSHTSHTHTSHTHTSHTSHTHTSHTHTSHTSHTSHTHTSHTHTSHTSHTSHTHTHTSHTHTSHTHTHTHTS
HTSHTHTSHTSHTSHTSHTSHTHTHTHTSHTHTHTHTHTSHTHTSHTSHTHTHTSHTHTSHTHTHTSHTHTSHTSHTS
HTHTSHTHTHTSHTSHTSHTSHTSHTSHTHTHTSHTSHTSHTHTHTSHTHTHTHTHTSHTHTSHTHTSHTHTSHTSHT
SHTSHTHTSHTHTSHTSHTHTHTSHTHTHTSHTSHTHTHTHTHTHTHTHTHTHTHTHTHTHTSHTSHTSHTSHTHTHT
SHTSHTSHTHTSHTHTHTSHTHTSHTSHTSHTHTHTHTHTHTHTSHTSHTSHTSHTHTHTHTSHTSHTSHTSHTSHTH
THTHTSHTSHTSHTSHTSHTHTHTHTHTSHTHTHTSHTHTSHTHTHTSHTHTHTSHTHTSHTHTSHTSHTSHTSHTHT
HTHTSHTSHTSHTHTSHTHTSHTSHTHTHTHTHTHTHTSHTHTHTHTHTSHTHTHTHTSHTSHTSHTSHTSHTSHTSH
TSHTSHTSHTSHTSHTSHTSHTHTSHTHTSHTSHTSHTHTSHTSHTSHTSHTHTHTSHTHTHTHTHTHTHTHTHTHTS
HTHTHTHTHTSHTSHTHTHTSHTHTSHTSHTSHTSHTHTHTSHTSHTSHTHTHTSHTHTHTHTHTSHTSHTSHTSHTS
HTHTSHTSHTHTHTHTHTHTHTHTSHTSHTHTSHTSHTHTHTSHTSHTHTSHTSHTSHTSHTSHTSHTHTHTSHTHTS
HTSHTHTHTHTHTSHTSHTSHTSHTSHTSHTSHTSHTHTSHTSHTHTHTSHTSHTHTHTSHTSHTSHTHTSHTHTSHT
HTHTHTSHTHTSHTHTSHTSHTHTSHTHTSHTHTHTSHTSHTHTSHTHTHTHTHTHTHTSHTHTSHTHTHTSHTHTSH
TSHTHTSHTHTHTSHTHTSHTSHTHTSHTSHTHTSHTHTSHTSHTSHTSHTHTSHTHTSHTHTSHTSHTSHTSHTHTH
THTHTSHTHTSHTHTSHTHTSHTHTSHTSHTHTHTHTSHTSHTHTHTHTSHTHTSHTSHTHTHTSHTHTSHTSHTSHT
SHTHTSHTSHTSHTSHTHTHTSHTHTHTHTSHTHTHTSHTHTSHTHTHTSHTSHTSHTSHTSHTSHTSHTHTHTHTSH
TSHTSHTSHTHTSHTSHTSHTHTHTSHTSHTHTSHTHTSHTSHTSHTHTSHTHTHTHTSHTSHTHTHTHTHTSHTHTH
TSHTHTSHTSHTSHTHTSHTHTSHTHTHTHTSHTHTSHTHTSHTHTSHTHTSHTHTHTHTSHTSHTSHTHTHTSHTHT
SHTHTHTSHTSHTSHTSHTHTSHTSHTHTHTHTSHTSHTHTSHTSHTSHTSHTHTHTSHTHTHTSHTSHTHTHTSHTH
TSHTHTHTSHTSHTHTHTSHTHTSHTHTHTHTHTSHTSHTSHTHTSHTSHTSHTSHTSHTSHTSHTSHTHTHTHTSHT
SHTHTHTSHTSHTSHTSHTSHTSHTSHTSHTSHTSHTHTSHTHTSHTHTHTSHTHTSHTSHTSHTHTHTSHTSHTHTH
THTSHTHTSHTHTHTSHTSHTHTSHTHTSHTSHTSHTHTHTHTSHTSHTHTHTHTSHTHTSHTSHTSHTHTHTHTHTH
THTSHTSHTHTHTHTHTSHTSHTHTHTSHTSHTSHTHTHTSHTHTHTSHTHTSHTSHTHTHTSHTSHTHTSHTSHTSH
THTHTSHTHTHTHTHTSHTHTHTSHTSHTHTHTSHTHTHTSHTSHTSHTHTSHTSHTHTHTSHTSHTHTSHTSHTHTH
TSHTHTSHTSHTHTSHTHTSHTSHTHTSHTHTSHTHTHTSHTSHTSHTSHTSHTSHTSHTHTSHTHTSHTSHTHTHTS
HTHTSHTHTSHTHTHTSHTSHTSHTHTHTSHTHTSHTHTSHTHTSHTSHTSHTSHTSHTHTHTSHTSHTSHTSHTHTS
HTHTHTHTHTHTHTHTSHTHTHTHTSHTHTHTHTHTSHTSHTSHTHTHTSHTHTSHTHTSHTSHTSHTHTSHTHTSHT
HTHTHTSHTSHTSHTHTSHTSHTHTSHTSHTSHTSHTHTHTSHTSHTSHTHTHTSHTHTSHTSHTHTSHTSHTHTSHT
SHTSHTSHTHTHTSHTHTSHTHTSHTSHTHTSHTHTHTHTSHTSHTSHTSHTSHTSHTSHTHTSHTSHTHTHTSHTSH
THTHTHTHTHTHTHTSHTSHTSHTSHTHTSHTSHTSHTHTHTSHTSHTSHTHTSHTHTSHTSHTHTHTHTHTSHTSHTSH
THTHTSHTHTSHTSHTSHTHTHTSHTSHTSHTHTHTHTHTHTHTHTSHTHTSHTSHTHTHTHTSHTSHTSHTSHTHTS
HTHTHTHTHTHTHTSHTSHTHTSHTHTHTSHTSHTSHTHTHTHTSHTHTHTHTHTHTSHTHTHTHTHTSHTHTSHTSH
TSHTSHTHTSHTHTSHTSHTSHTSHTHTSHTHTSHTHTSHTHTHTHTSHTSHTSHTSHTHTSHTHTSHTSHTSHTSHT
HTHTHTHTHTSHTSHTSHTHTHS
```

...we'll talk more about this in a few weeks when we discuss *quantum compilation*.

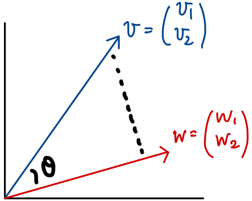
We can now create every single single-qubit quantum state: how do we *compare* them?

Recall what things look like in a classical vector space.



Inner products

We can define an **inner product** between two vectors that tells us how much overlap they have.



$$\vec{v} \cdot \vec{w} = \langle \vec{v}, \vec{w} \rangle = \vec{v}^T \vec{w} = \begin{pmatrix} v_1 & v_2 \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$$

$$= v_1 w_1 + v_2 w_2$$

$$= \sum_{i=1}^{2} v_i w_i$$

$$= |\vec{v}| \cdot |\vec{w}| \cos\theta$$

Take just one of these representations:

$$\langle \vec{v}, \vec{w} \rangle = \vec{v}^T \vec{w}$$

The Hilbert space has complex valued vectors. The inner product looks *similar*, but slightly different. Let

$$|v\rangle = v_1|0\rangle + v_2|1\rangle \quad |w\rangle = w_1|0\rangle + w_2|1\rangle \quad v_i, w_i \in \mathbb{C}$$

The inner product is defined as

$$\langle |v\rangle, |w\rangle \rangle = \left(|v\rangle^T\right)^* |w\rangle = \left(|v\rangle^\dagger\right) |w\rangle$$

Inner products

This notation is cumbersome, so let's complete our knowledge of Dirac notation by introducing the **bra**:

$$\langle v| = \left(|v\rangle^\dagger\right) = \begin{pmatrix} v_1^* & v_2^* \end{pmatrix}$$

The inner product is defined as

$$\langle |v\rangle, |w\rangle \rangle = (|v\rangle)^\dagger |w\rangle = \langle v||w\rangle = \langle v|w\rangle$$

Written another way,

$$\langle v|w\rangle = v_1^* w_1 + v_2^* w_2$$

Pro tip: `np.vdot(v, w)`

Exercise: compute the inner product of the state

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

$$|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

with itself.

$$\langle\psi| = (\alpha^* \quad \beta^*)$$

$$\langle\psi|\psi\rangle = \alpha^*\alpha + \beta^*\beta$$

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

$$= 1$$

Exercise: compute the inner product between all possible combinations of $|0\rangle$ and $|1\rangle$.

$\langle 0\|0\rangle$	1
$\langle 0\|1\rangle$	0
$\langle 1\|0\rangle$	0
$\langle 1\|1\rangle$	1

$$\langle 0|1\rangle = \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = 0$$

Orthonormal bases

For a single qubit, a pair of states that are **normalized** and **orthogonal** constitute an **orthonormal basis** for the Hilbert space.

Exercise: do the states

$$|p\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{i}{\sqrt{2}}|1\rangle, \quad |m\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{i}{\sqrt{2}}|1\rangle$$

form an orthonormal basis?

$$\langle p|p\rangle = \frac{1}{\sqrt{2}} \cdot \frac{1}{\sqrt{2}} + \frac{-i}{\sqrt{2}} \cdot \frac{i}{\sqrt{2}} = 1$$

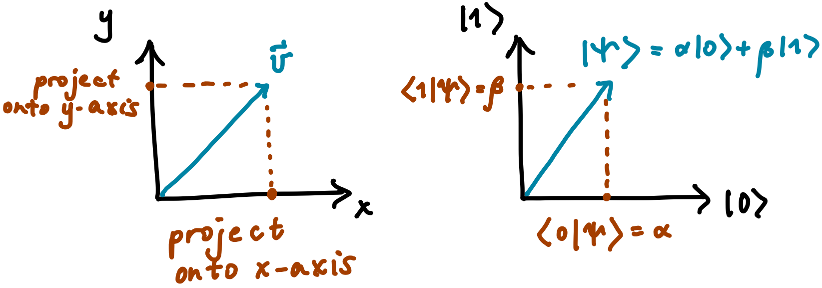
$$\langle m|m\rangle = 1$$

$$\langle m|p\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{i}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 1/\sqrt{2} \\ i/\sqrt{2} \end{pmatrix} = \frac{1}{2} + \left(-\frac{1}{2}\right) = 0$$

$$\langle p|m\rangle = (\langle m|p\rangle)^*$$

Projective measurements

Measurement is performed with respect to a basis; we perform **projections** to determine the overlap with a given basis state.



(Image for expository purposes only!)

Image credit: Xanadu Quantum Codebook I.9

Projective measurements

When we measure state $|\varphi\rangle$ with respect to basis $\{|\psi_i\rangle\}$, the probability of obtaining outcome i is

$$\Pr(\text{outcome } i) = |\langle\psi_i|\varphi\rangle|^2$$

If we observe outcome i, following the measurement the system will be left in state $|\psi_i\rangle$.

Measurement in computational basis

$$\{|0\rangle, |1\rangle\}$$

Let $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$.

Then if we measure $|\psi\rangle$ is the computational basis,

$$\Pr(0) = |\langle 0|\psi\rangle|^2 = |\alpha|^2$$

$$\langle 0|\psi\rangle = \langle 0|\left(\alpha|0\rangle + \beta|1\rangle\right) = \alpha\langle 0|0\rangle + \beta\langle 0|1\rangle$$

$$= \alpha$$

$$\Pr(1) = |\beta|^2 = |\langle 1|\psi\rangle|^2$$

Measurement in computational basis

So far we've seen 3 ways of extracting information out of a QNode:

1. `qml.state()`
2. `qml.probs(wires=x)`
3. `qml.sample()`

These return results of measurements taken with respect to the computational basis; and most hardware only allows for computational basis measurements.

How can we measure with respect to *different bases* with that restriction? (and what does that mean?)

Measurement in computational basis

Exercise: what are the measurement outcome probabilities if we measure

$$|p\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{i}{\sqrt{2}}|1\rangle, \quad |m\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{i}{\sqrt{2}}|1\rangle$$

in the computational basis?

Basis rotations

Projective measurements can be performed with respect to any orthonormal basis. For example, $\{|+\rangle, |-\rangle\}$:

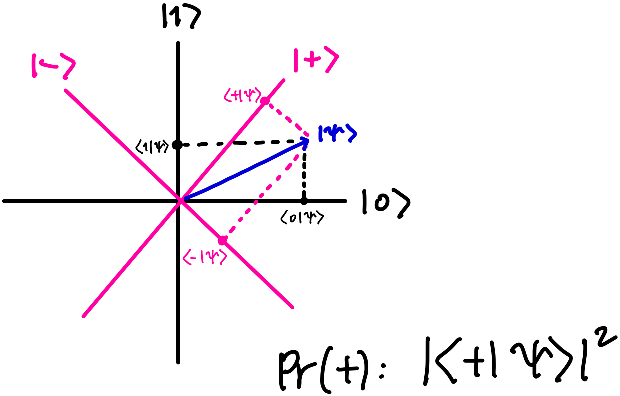


Image credit: Codebook node I.9

We stopped here on Monday.

Use a basis rotation to “trick” the quantum computer.

Suppose we want to measure in the “Y” basis:

$$|p\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + i|1\rangle\right), \quad |m\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle - i|1\rangle\right).$$

Unitary operations preserve length *and* angles between normalized quantum state vectors.

There exists some unitary transformation that will convert between this basis and the computational basis.

Exercise: determine a quantum circuit that sends

$$
\begin{aligned}
|0\rangle \quad &\rightarrow \quad |p\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + i|1\rangle\right) \\
|1\rangle \quad &\rightarrow \quad |m\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle - i|1\rangle\right)
\end{aligned}
$$

At the end of our circuit, we can then apply the reverse (adjoint) of this transformation rotate *back* to the computational basis.

$$|p\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + i|1\rangle\right) \quad \rightarrow \quad |0\rangle$$
$$|m\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle - i|1\rangle\right) \quad \rightarrow \quad |1\rangle$$

That way, if we measure and observe $|0\rangle$, we know that this was previously $|p\rangle$ in the Y basis (and similarly for $|m\rangle$).

Adjoints

In PennyLane, we can compute adjoints of operations *and* entire quantum functions using `qml.adjoint`:

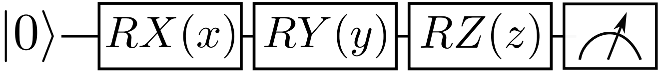
```
def some_function(x):
    qml.RZ(Z, wires=0)

def apply_adjoint(x):
    qml.adjoint(qml.S)(wires=0)
    qml.adjoint(some_function)(x)
```

`qml.adjoint` is a special type of function called a **transform**. We will cover transforms in more detail later in the course.

Basis rotations: hands-on

Let's run the following circuit, and measure in the Y basis



Hands-on time...

Generally, we are interested in measuring real, physical quantities. In physics, these are called observables.

Observables are represented mathematically by Hermitian matrices. An operator (matrix) H is Hermitian if

$$H = H^\dagger$$

Why Hermitian? The possible measurement outcomes are given by the eigenvalues of the operator, and eigenvalues of Hermitian operators are **real**.

Example:

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

Z is Hermitian:

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

Its eigensystem is

$$\lambda_1 = +1, \qquad |\psi_1\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$\lambda_2 = -1, \qquad |\psi_2\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Example:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

X is Hermitian and its (normalized) eigensystem is

$$\lambda_1 = +1, \qquad |\psi_1\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$\lambda_2 = -1, \qquad |\psi_2\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

Observables

Example:

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

Y is Hermitian and its (normalized) eigensystem is

$$\lambda_1 = +1, \qquad |\psi_1\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ i \end{pmatrix}$$

$$\lambda_2 = -1, \qquad |\psi_2\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -i \end{pmatrix}$$

When we measure X, Y, or Z on a state, for each shot we will get one of the eigenstates (/eigenvalues). If we take multiple shots, what do we expect to see *on average*?

Analytically, the **expectation value** of measuring the observable M given the state $|\psi\rangle$ is

$$\langle M \rangle = \langle \psi | M | \psi \rangle.$$

Expectation values: analytical

Example: consider the quantum state

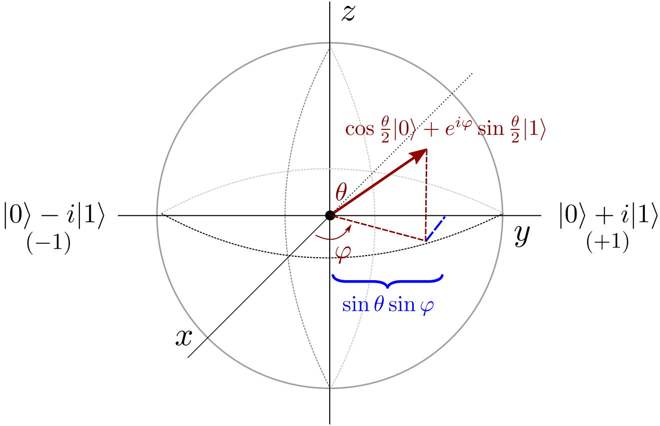
$$|\psi\rangle = \frac{1}{2}|0\rangle - i\frac{\sqrt{3}}{2}|1\rangle.$$

Let's compute the expectation value of Y:

$$\begin{aligned}
|\psi\rangle &= \left(\frac{1}{2}\langle 0| + i\frac{\sqrt{3}}{2}\langle 1|\right) Y \left(\frac{1}{2}|0\rangle - i\frac{\sqrt{3}}{2}|1\rangle\right) \\
&= \left(\frac{1}{2}\langle 0| + i\frac{\sqrt{3}}{2}\langle 1|\right)\left(\frac{i}{2}|1\rangle - \frac{\sqrt{3}}{2}|0\rangle\right) \\
&= \frac{i}{4}\langle 0|1\rangle - \frac{\sqrt{3}}{4}\langle 1|1\rangle - \frac{\sqrt{3}}{4}\langle 0|0\rangle - i\frac{3}{4}\langle 1|0\rangle \\
&= -\frac{\sqrt{3}}{2}
\end{aligned}$$

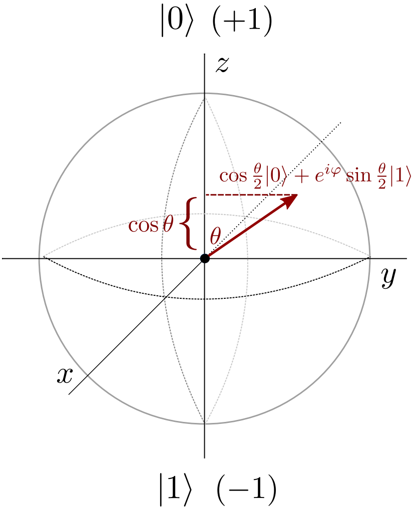
Expectation values and the Bloch sphere

The Bloch sphere offers us some more insight into what a projective measurement is.



Exercise: derive the expression in blue by computing $\langle\psi|\, Y |\psi\rangle$.

Expectation values and the Bloch sphere



$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle$$

$$Z|\psi\rangle = \cos\frac{\theta}{2}|0\rangle - e^{i\varphi}\sin\frac{\theta}{2}|1\rangle$$

$$\langle\psi|\,Z|\psi\rangle = \cos^2\frac{\theta}{2} - \sin^2\frac{\theta}{2}$$

$$= \cos\theta$$

Expectation values: from measurement data

Let's compute the expectation value of Z for the following circuit using 10 samples:

```
dev = qml.device('default.qubit', wires=1, shots=10)

@qml.qnode(dev)
def circuit():
    qml.RX(2*np.pi/3, wires=0)
    return qml.sample()
```

Results might look something like this:

$$[1,\ 1,\ 1,\ 0,\ 1,\ 1,\ 1,\ 0,\ 1,\ 1]$$

Expectation values: from measurement data

The expectation value pertains to the measured eigenvalue; recall Z eigenstates are

$$\lambda_1 = +1, \qquad |\psi_1\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$\lambda_2 = -1, \qquad |\psi_2\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

So when we observe $|0\rangle$, this is eigenvalue $+1$ (and if $|1\rangle$, -1). Our samples shift from

```
[1, 1, 1, 0, 1, 1, 1, 0, 1, 1]
```

to

```
[-1, -1, -1, 1, -1, -1, -1, 1, -1, -1]
```

Expectation values: from measurement data

The expectation value is the weighted average of this, where the weights are the eigenvalues:

$$\langle Z \rangle = \frac{1 \cdot n_1 + (-1) \cdot n_{-1}}{N}$$

where

- n_1 is the number of $+1$ eigenvalues
- n_{-1} is the number of -1 eigenvalues
- N is the total number of shots

For our example, $\langle Z \rangle = -0.6$.

Let's do this in PennyLane instead:

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

@qml.qnode(dev)
def measure_z():
    qml.RX(2*np.pi/3, wires=0)
    return qml.expval(qml.PauliZ(0))
```

Recap

- Define a universal gate set
- Compute the inner product between two quantum states
- Perform a projective measurement
- Measure a qubit in different bases
- Measure single-qubit expectation values

Next time

Content:

- Mathematical representation of multi-qubit systems
- Multi-qubit gates
- Entanglement

Action items:

1. Finish assignment 0
2. Keep an eye out for A1 and literacy assignment

Recommended reading:

- From today: Codebook nodes I.9-I.10
- For next time: Codebook nodes I.11-I.14
- Nielsen & Chuang 4.3