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

Last time

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.

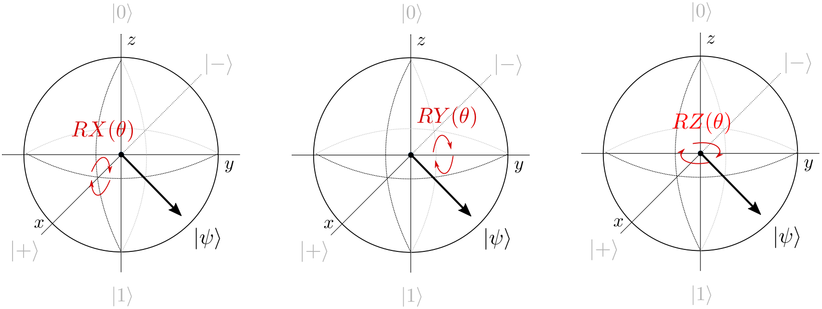


Image credit: Codebook node I.6

Last time

We learned how to implement quantum circuits in PennyLane.

```python
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()
```

Last time

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

Global phase:

Relative phase:

Last time

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

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

Last time

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

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

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).

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: $U^{-1} = U^\dagger$

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.

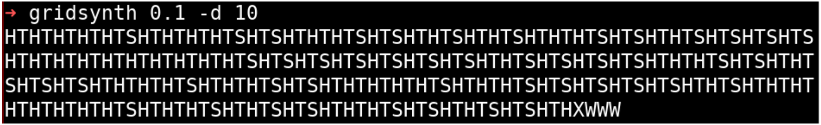
Hands-on...

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

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

Universal gate sets: H and T

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}:

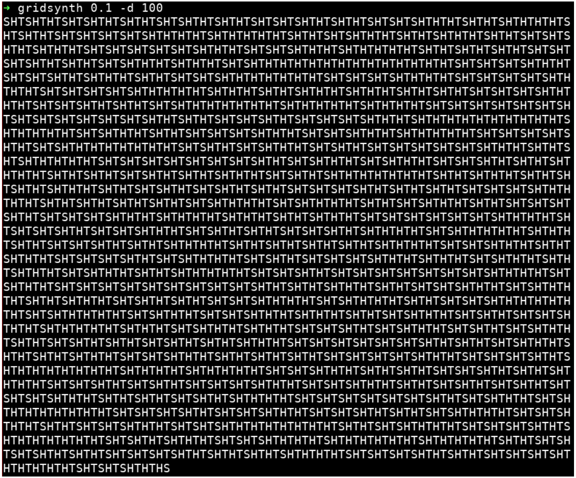


```
→ gridsynth 0.1 -d 10
HTHTHTHTHTSHTHTHTHTSHTSHTHTHTSHTSHTHTSHTHTSHTHTHTSHTSHTHTSHTSHTSHTS
HTHTHTHTHTHTHTHTHTHTHTSHTSHTSHTSHTSHTSHTHTSHTSHTSHTSHTHTHTSHTSHTHT
SHTSHTSHTHTHTHTSHTHTHTSHTSHTHTHTHTHTSHTHTHTSHTSHTSHTSHTSHTHTSHTHTHT
HTHTHTHTHTSHTHTHTSHTHTSHTSHTHTHTSHTSHTHTSHTSHTHXWWW
```

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}:



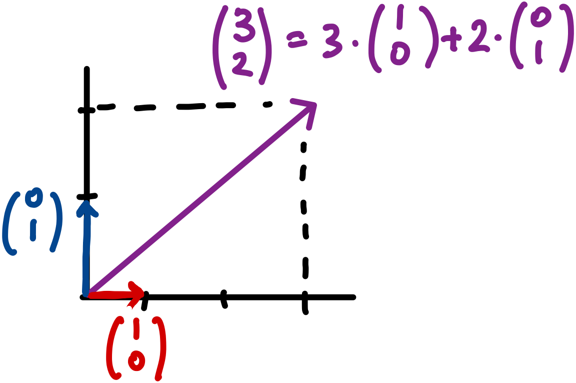
```
→ gridsynth 0.1 -d 100
SHTSHTHTSHTSHTHTSHTHTSHTSHTHTSHTHTSHTSHTSHTHTSHTHTSHTSHTSHTHTHTSHTHTSHTHTHTHTS
HTSHTHTSHTSHTSHTSHTSHTHTHTHTSHTHTHTHTHTSHTHTSHTSHTHTHTSHTHTSHTHTHTSHTHTSHTSHTS
HTHTSHTHTHTSHTSHTSHTSHTSHTSHTHTHTSHTSHTSHTHTHTSHTHTHTHTHTSHTHTSHTHTSHTHTSHTSHT
SHTSHTHTSHTHTSHTSHTHTHTSHTHTHTSHTSHTHTHTHTHTHTHTHTHTHTHTHTHTHTSHTSHTSHTSHTHTHT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

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

Inner products

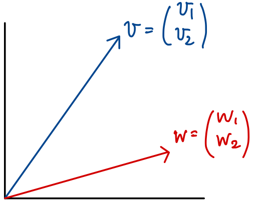
We can now create every single single-qubit quantum state: how do we *compare* them?

Recall what things look like in a classical vector space.


$\begin{pmatrix}3\\2\end{pmatrix} = 3\cdot\begin{pmatrix}1\\0\end{pmatrix} + 2\cdot\begin{pmatrix}0\\1\end{pmatrix}$
$\begin{pmatrix}0\\1\end{pmatrix}$
$\begin{pmatrix}1\\0\end{pmatrix}$

Inner products

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



$v = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$
$w = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$

Inner products

Take just one of these representations:

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

The inner product is defined as

Inner products

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

The inner product is defined as

Written another way,

Pro tip:

Inner products

Exercise: compute the inner product of the state

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

with itself.

Inner products

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

$\langle 0\vert 0\rangle$	
$\langle 0\vert 1\rangle$	
$\langle 1\vert 0\rangle$	
$\langle 1\vert 1\rangle$	

Orthonormal bases

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

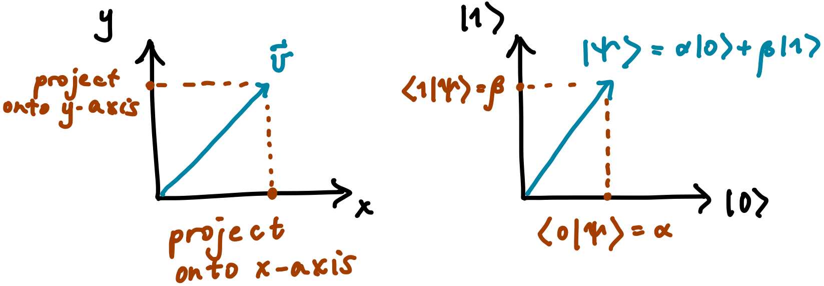
Exercise: do the states

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

form an orthonormal basis?

Projective measurements

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


y
$\vec{v}$
project onto y-axis
x
project onto x-axis
$|1\rangle$
$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$
$\langle 1|\psi\rangle = \beta$
$|0\rangle$
$\langle 0|\psi\rangle = \alpha$

(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

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

Measurement in computational basis

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

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

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, $\{|+\rangle, |-\rangle\}$:

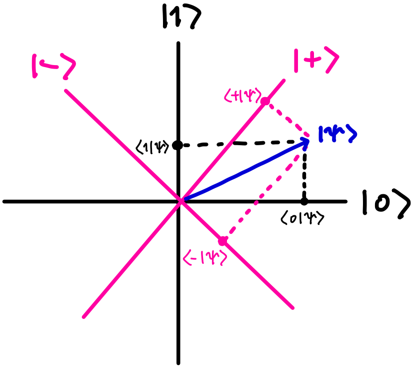


Image credit: Codebook node I.9

Basis rotations

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.

Basis rotations

Exercise: determine a quantum circuit that sends

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

Basis rotations

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

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`:

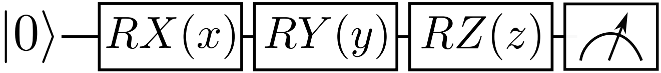
```python
def some_function(x):
    qml.RZ(Z, wires=0)

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...

Observables

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**.

Observables

Example:

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

Z is Hermitian:

Its eigensystem is

Observables

Example:

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

X is Hermitian and its (normalized) eigensystem is

Observables

Example:

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

Y is Hermitian and its (normalized) eigensystem is

Expectation values

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

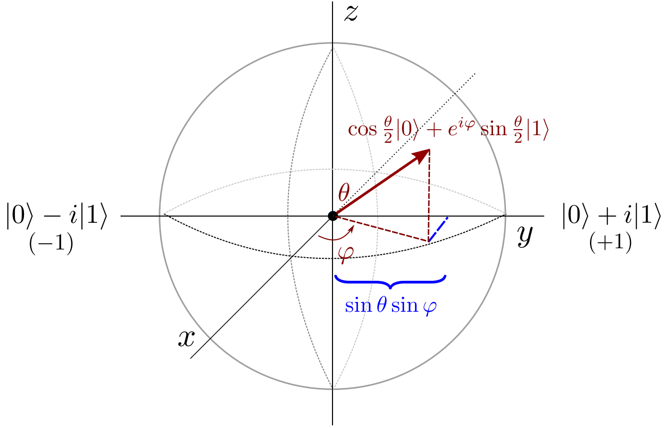
Example: consider the quantum state

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

Let's compute the expectation value of Y:

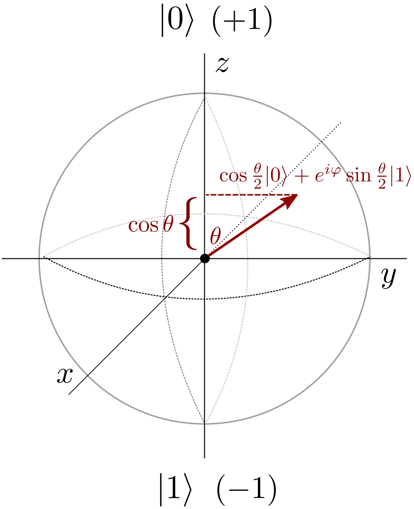
Expectation values and the Bloch sphere

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


$|0\rangle \;\; (+1)$
z
$\cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle$
$\cos\theta$
θ
y
x
$|1\rangle \;\; (-1)$

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:

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,

Expectation values

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