CPEN 400Q Lecture 06
Superdense coding and quantum teleportation; Measurement II (expectation values)

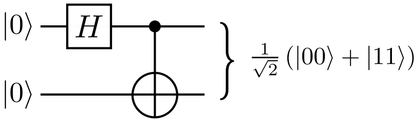
Wednesday 22 January 2025

Announcements

- Assignment 1 due Sun 26 1 at 23:59 (will be adjusted based on what we cover today)
- Midterm in class on Wed 29 Jan (see PrairieLearn for details)
- Quiz 3 on Monday

Last time

We defined entangled states and entangling gates:



Entangled states *cannot be expressed* as a tensor products of all constituent single-qubit states.

An **entangling gate** sends some non-entangled (separable state) to an entangled state.

Last time

We saw more examples of two-qubit gates:

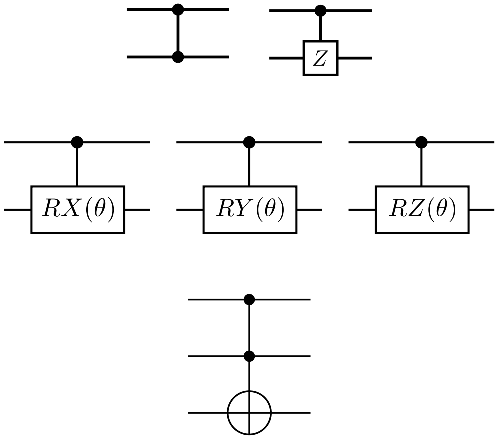


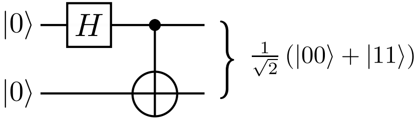
Image credit: Codebook node I.13, I.14

Last time

We saw how to make arbitrary controlled operations (or functions) in PennyLane using `qml.ctrl`.

```
@qml.qnode(dev)
def my_circuit():
    qml.CNOT(wires=[2, 3])
    qml.ctrl(qml.S, control=1)(wires=0)
    qml.Toffoli(wires=[0, 1, 2])
    return qml.sample()
```

In preparation for some algorithms, we created an orthonormal basis of entangled states called the Bell basis:



$$
\begin{aligned}
|\Psi_{00}\rangle &= \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right) \\
|\Psi_{01}\rangle &= \frac{1}{\sqrt{2}}\left(|01\rangle + |10\rangle\right) \\
|\Psi_{10}\rangle &= \frac{1}{\sqrt{2}}\left(|00\rangle - |11\rangle\right) \\
|\Psi_{11}\rangle &= \frac{1}{\sqrt{2}}\left(|01\rangle - |10\rangle\right)
\end{aligned}
$$

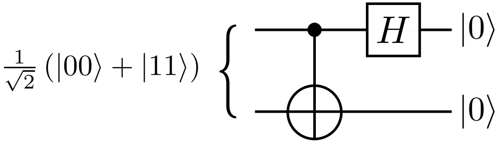
Learning outcomes

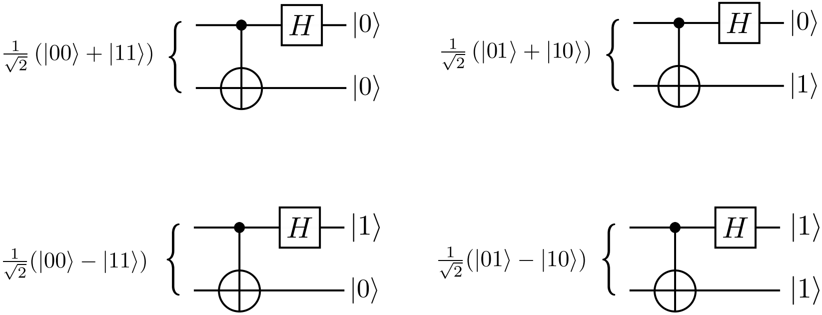
- leverage entanglement to implement superdense coding
- prove that arbitrary quantum states cannot be cloned
- teleport a qubit
- define observables and expectation values
- compute expectation values of an observable after running a circuit

The Bell basis

$$
\begin{aligned}
|\Psi_{00}\rangle &= \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right) \\
|\Psi_{01}\rangle &= \frac{1}{\sqrt{2}}\left(|01\rangle + |10\rangle\right) \\
|\Psi_{10}\rangle &= \frac{1}{\sqrt{2}}\left(|00\rangle - |11\rangle\right) \\
|\Psi_{11}\rangle &= \frac{1}{\sqrt{2}}\left(|01\rangle - |10\rangle\right)
\end{aligned}
$$

We can measure in this basis by applying the adjoint of the circuit:





Two famous quantum algorithms, **superdense coding** and **teleportation** work by performing a measurement in the Bell basis.

Suppose Alice wants to send Bob two classical bits of information, say ‘1’ and ‘0’.

Q1: How many classical bits must she send to Bob to do this?

Q2: How many *qubits* must she send to Bob to do this?

Superdense coding

Alice and Bob start the protocol with this shared entangled state:

$$|\Phi\rangle_{AB} = \frac{1}{\sqrt{2}} \left(|00\rangle + |11\rangle\right)$$

Next, depending on her bits, Alice performs one of the following operations on her qubit:

$$\begin{aligned} 00 &\rightarrow I \\ 01 &\rightarrow X \\ 10 &\rightarrow Z \\ 11 &\rightarrow ZX \end{aligned}$$

What happened to the entangled state?

$$|\Phi\rangle_{AB} = \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right)$$

It will transform to:

$$\begin{aligned} 00 &\rightarrow_I \\ 01 &\rightarrow_X \\ 10 &\rightarrow_Z \\ 11 &\rightarrow_{ZX} \end{aligned}$$

Superdense coding

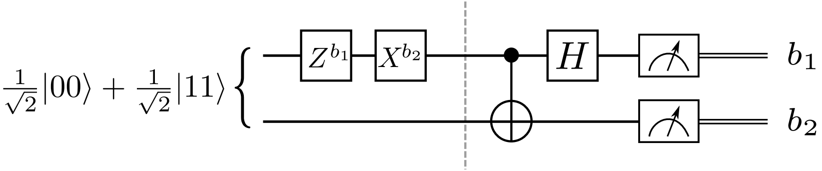
Now, Bob can either

- perform a measurement directly in the Bell basis
- perform a basis transformation from the Bell basis back to the computational basis

to determine his state with certainty, and thus, the bits Alice sent.

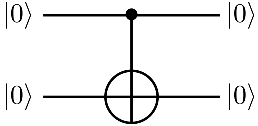
$$
\begin{aligned}
(H \otimes I)\mathrm{CNOT} \cdot \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) &= |00\rangle \\
(H \otimes I)\mathrm{CNOT} \cdot \frac{1}{\sqrt{2}}(|10\rangle + |01\rangle) &= |01\rangle \\
(H \otimes I)\mathrm{CNOT} \cdot \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) &= |10\rangle \\
(H \otimes I)\mathrm{CNOT} \cdot \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle) &= |11\rangle
\end{aligned}
$$

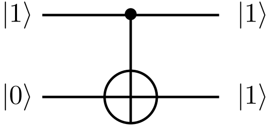
Let's go implement it!



Suppose you found a really cool quantum state, and you want to send a copy to a friend. Can you?

Idea: CNOT sends $|00\rangle$ to $|00\rangle$, and $|10\rangle$ to $|11\rangle$, thus copying the first qubit's state to the second.





Everything is linear, so will this work in general?

Very easy to find a state for which this fails:

(Not) copying quantum states

The no-cloning theorem

It is impossible to create a copying circuit that works for arbitrary quantum states.

In other words, there is no circuit that sends

$$|\psi\rangle \otimes |s\rangle \to |\psi\rangle \otimes |\psi\rangle$$

for any arbitrary $|\psi\rangle$.

Suppose we want to clone a state $|\psi\rangle$. We want a unitary operation that sends

where $|s\rangle$ is some arbitrary state.

Suppose we find one. If our cloning machine is to be universal, we must also be able to clone some other state, $|\varphi\rangle$.

Proof of the no-cloning theorem

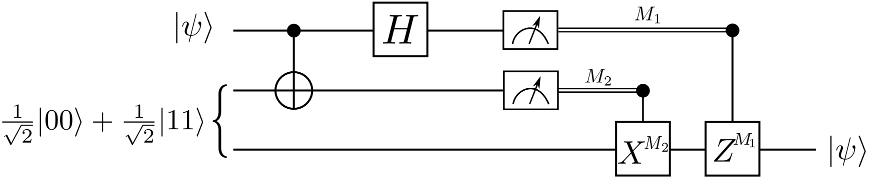
We purportedly have:

$$\begin{aligned} U(|\psi\rangle \otimes |s\rangle) &= |\psi\rangle \otimes |\psi\rangle \\ U(|\varphi\rangle \otimes |s\rangle) &= |\varphi\rangle \otimes |\varphi\rangle \end{aligned}$$

Take the inner product of the LHS of both equations:

Now take the inner product of the RHS of both equations:

We cannot clone arbitrary qubit states, but we *can* teleport them!



Homework: work through this circuit and determine the state after each gate (it is worth doing this once!).

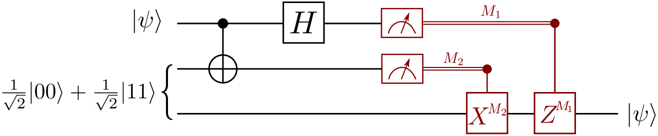
Before measurements, the combined state of the system is

What do you notice about this state?

Alice measures in the computational basis and sends her results to Bob, who can adjust his state as needed.

$$
\begin{aligned}
00 &: I(\alpha|0\rangle + \beta|1\rangle) = (\alpha|0\rangle + \beta|1\rangle) \\
01 &: X(\alpha|1\rangle + \beta|0\rangle) = (\alpha|0\rangle + \beta|1\rangle) \\
10 &: Z(\alpha|0\rangle - \beta|1\rangle) = (\alpha|0\rangle + \beta|1\rangle) \\
11 &: ZX(\alpha|1\rangle - \beta|0\rangle) = (\alpha|0\rangle + \beta|1\rangle)
\end{aligned}
$$

Let's implement it!



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 of an observable are its eigenvalues, and eigenvalues of Hermitian operators are **real**.

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

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

Expectation values: analytical

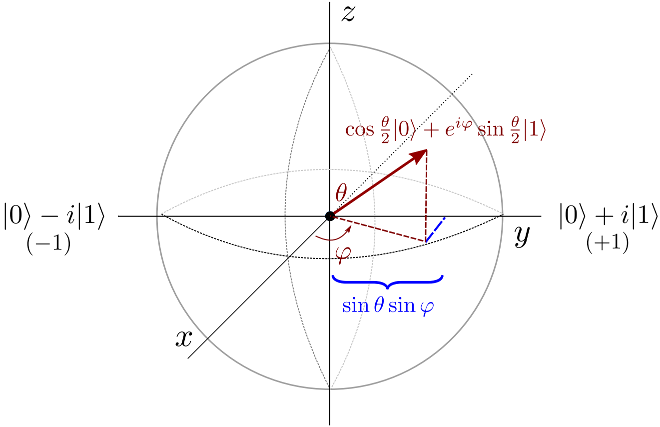
Exercise: consider the quantum state

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

Compute the expectation value of Y:

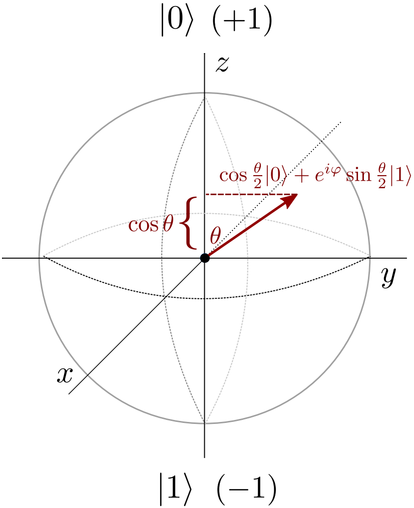
Expectation values and the Bloch sphere

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



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))
```

Multi-qubit expectation values

Example: operator $Z \otimes Z$.

Eigenvalues are computational basis states:

$$
\begin{aligned}
(Z \otimes Z)|00\rangle &= |00\rangle \\
(Z \otimes Z)|01\rangle &= -|01\rangle \\
(Z \otimes Z)|10\rangle &= -|10\rangle \\
(Z \otimes Z)|11\rangle &= |11\rangle
\end{aligned}
$$

To compute an expectation value from data:

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

Multi-qubit expectation values

Example: operator $X \otimes I$.

Eigenvalues of X are the $|+\rangle$ and $|-\rangle$ states:

$$
\begin{aligned}
(X \otimes I)|+0\rangle &= |+0\rangle \\
(X \otimes I)|+1\rangle &= |+1\rangle \\
(X \otimes I)|-0\rangle &= -|-0\rangle \\
(X \otimes I)|-1\rangle &= -|-1\rangle
\end{aligned}
$$

Fun fact: All Pauli operators have an equal number of $+1$ *and* -1 *eigenvalues!*

Multi-qubit expectation values

How to compute expectation value of X from data, when we can only measure in the computational basis?

Basis rotation: apply H to first qubit

$$
\begin{aligned}
(H \otimes I)(X \otimes I)|+0\rangle &= |00\rangle \\
(H \otimes I)(X \otimes I)|+1\rangle &= |01\rangle \\
(H \otimes I)(X \otimes I)|-0\rangle &= -|10\rangle \\
(H \otimes I)(X \otimes I)|-1\rangle &= -|11\rangle
\end{aligned}
$$

When we measure and obtain $|10\rangle$ or $|11\rangle$, we know those correspond to the -1 eigenstates of $X \otimes I$.

Hands-on: multi-qubit expectation values

Multi-qubit expectation values can be created using the `@` symbol:

```
@qml.qnode(dev)
def circuit(x):
    qml.Hadamard(wires=0)
    qml.CRX(x, wires=[0, 1])
    return qml.expval(qml.PauliZ(0) @ qml.PauliZ(1))
```

Can return *multiple* expectation values if no shared qubits.

```
@qml.qnode(dev)
def circuit(x):
    qml.Hadamard(wires=0)
    qml.CRX(x, wires=[0, 1])
    return qml.expval(qml.PauliZ(0)), qml.expval(qml.PauliZ(
                                                  1))
```

Next time

Content:

- Measurement part 3: generalized measurements and state discrimination

Action items:

1. Assignment 1 due Sunday 23:59
2. Quiz 3 on Monday
3. Study for midterm

Recommended reading:

- Codebook nodes IQC, SQ, MQ; Nielsen & Chuang 1.2, 1.3.1-1.3.7, 2.2.3-2.2.5, 2.3, 4.3