Quantum Algorithms & Qiskit

Alexander J. Heilman

August 19, 2021

Intentions

Introduce you to some basic quantum algorithms

Intentions

Introduce you to some basic quantum algorithms

- Quantum Fourier Transform

Intentions

Introduce you to some basic quantum algorithms

- Quantum Fourier Transform
- Phase Estimation

Intentions

Introduce you to some basic quantum algorithms

- Quantum Fourier Transform
- Phase Estimation
- Amplitude Amplification

Intentions

Introduce you to some basic quantum algorithms

- Quantum Fourier Transform
- Phase Estimation
- Amplitude Amplification

Introduce you to some tools available through Qiskit

- Visualize qubit evolution

Intentions

Introduce you to some basic quantum algorithms

- Quantum Fourier Transform
- Phase Estimation
- Amplitude Amplification

Introduce you to some tools available through Qiskit

- Visualize qubit evolution
- Simulate quantum circuits

Intentions

Introduce you to some basic quantum algorithms

- Quantum Fourier Transform
- Phase Estimation
- Amplitude Amplification

Introduce you to some tools available through Qiskit

- Visualize qubit evolution
- Simulate quantum circuits
- Experiment with real quantum computers

Intentions

Introduce you to some basic quantum algorithms

- Quantum Fourier Transform
- Phase Estimation
- Amplitude Amplification

Introduce you to some tools available through Qiskit

- Visualize qubit evolution
- Simulate quantum circuits
- Experiment with real quantum computers

First, the QFT

The Quantum Fourier Transform

The quantum Fourier transform (QFT) is a quantum implementation of the discrete Fourier transform.

The Quantum Fourier Transform

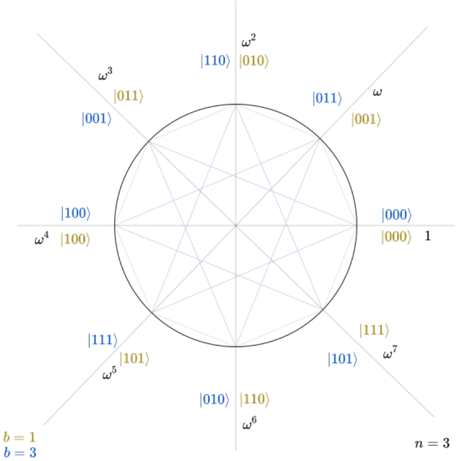
The quantum Fourier transform (QFT) is a quantum implementation of the discrete Fourier transform.

This means if you give the QFT some basis state $|b\rangle$, the QFT will give back states with phases traveling around the unit circle at a rate of ω^b.

The Quantum Fourier Transform

The quantum Fourier transform (QFT) is a quantum implementation of the discrete Fourier transform.

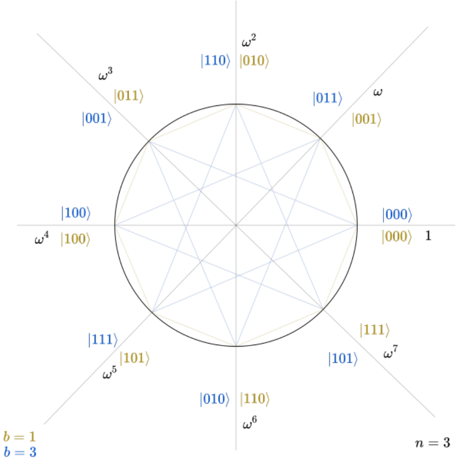
This means if you give the QFT some basis state $|b\rangle$, the QFT will give back states with phases traveling around the unit circle at a rate of ω^b.



The Quantum Fourier Transform

The quantum Fourier transform (QFT) is a quantum implementation of the discrete Fourier transform.

This means if you give the QFT some basis state $|b\rangle$, the QFT will give back states with phases traveling around the unit circle at a rate of ω^b.



But how?

QFT: The gory details

The explict action of the n-qubit QFT on some given basis vector is

$$|j_1, \ldots, j_n\rangle \longrightarrow$$

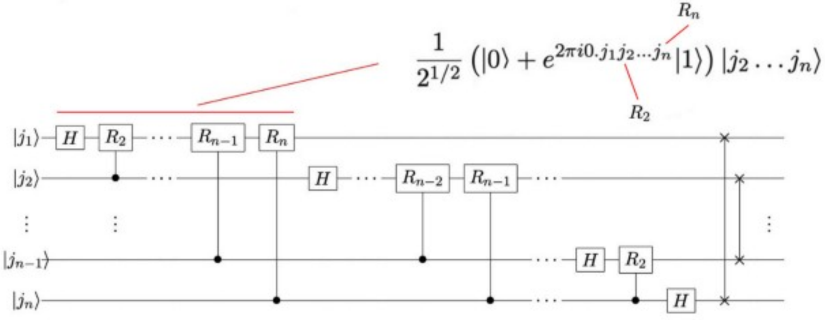
$$\frac{\left(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\right) \otimes \left(|0\rangle + e^{2\pi i 0.j_{n-1}j_n}|1\rangle\right) \otimes \ldots \otimes \left(|0\rangle + e^{2\pi i 0.j_1 \cdots j_n}|1\rangle\right)}{2^{n/2}}.$$

QFT: The gory details

The explict action of the n-qubit QFT on some given basis vector is

$$|j_1, \ldots, j_n\rangle \longrightarrow$$

$$\frac{\left(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\right) \otimes \left(|0\rangle + e^{2\pi i 0.j_{n-1}j_n}|1\rangle\right) \otimes \ldots \otimes \left(|0\rangle + e^{2\pi i 0.j_1 \cdots j_n}|1\rangle\right)}{2^{n/2}}.$$



It's easier to see in matrix form

QFT: The gory details II

For $n = 3$ the QFT is

$$\frac{1}{2\sqrt{2}}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7 \\ 1 & \omega^2 & \omega^4 & \omega^6 & 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega & \omega^4 & \omega^7 & \omega^2 & \omega^5 \\ 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 \\ 1 & \omega^5 & \omega^2 & \omega^7 & \omega^4 & \omega & \omega^6 & \omega^3 \\ 1 & \omega^6 & \omega^4 & \omega^2 & 1 & \omega^6 & \omega^4 & \omega^2 \\ 1 & \omega^7 & \omega^6 & \omega^5 & \omega^4 & \omega^3 & \omega^2 & \omega \end{bmatrix}.$$

QFT: The gory details II

For $n = 3$ the QFT is

$$\frac{1}{2\sqrt{2}}\begin{bmatrix}1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7\\ 1 & \omega^2 & \omega^4 & \omega^6 & 1 & \omega^2 & \omega^4 & \omega^6\\ 1 & \omega^3 & \omega^6 & \omega & \omega^4 & \omega^7 & \omega^2 & \omega^5\\ 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4\\ 1 & \omega^5 & \omega^2 & \omega^7 & \omega^4 & \omega & \omega^6 & \omega^3\\ 1 & \omega^6 & \omega^4 & \omega^2 & 1 & \omega^6 & \omega^4 & \omega^2\\ 1 & \omega^7 & \omega^6 & \omega^5 & \omega^4 & \omega^3 & \omega^2 & \omega\end{bmatrix}.$$

Not so interesting yet, but it's usefulness will soon be found in phase estimation

Phase Estimation

Phase Estimation takes some given unitary operator and an associated eigenvector and returns the corresponding eigenvalue.

$$\text{If } U|\psi\rangle = \lambda|\psi\rangle \quad \text{then} \quad U, |\psi\rangle \xrightarrow{PE} \lambda$$

.

Phase Estimation

Phase Estimation takes some given unitary operator and an associated eigenvector and returns the corresponding eigenvalue.

$$\text{If } U|\psi\rangle = \lambda|\psi\rangle \quad \text{then} \quad U, |\psi\rangle \xrightarrow{PE} \lambda$$

.

Really it returns $|k\rangle$, where the eigenvalue is $\lambda = e^{i\theta}$ with $\theta = k\frac{2\pi}{2^n}$.

Phase Estimation

Phase Estimation takes some given unitary operator and an associated eigenvector and returns the corresponding eigenvalue.

$$\text{If } U|\psi\rangle = \lambda|\psi\rangle \quad \text{then} \quad U, |\psi\rangle \xrightarrow{PE} \lambda$$

.

Really it returns $|k\rangle$, where the eigenvalue is $\lambda = e^{i\theta}$ with $\theta = k\frac{2\pi}{2^n}$.
(Or closest n-bit approx. with at least 40% probability)

Phase Estimation

Phase Estimation takes some given unitary operator and an associated eigenvector and returns the corresponding eigenvalue.

$$\text{If } U|\psi\rangle = \lambda|\psi\rangle \quad \text{then} \quad U, |\psi\rangle \xrightarrow{PE} \lambda$$

.

Really it returns $|k\rangle$, where the eigenvalue is $\lambda = e^{i\theta}$ with $\theta = k\frac{2\pi}{2^n}$.
(Or closest n-bit approx. with at least 40% probability)

So how do we do this?

Phase Estimation

Phase Estimation takes some given unitary operator and an associated eigenvector and returns the corresponding eigenvalue.

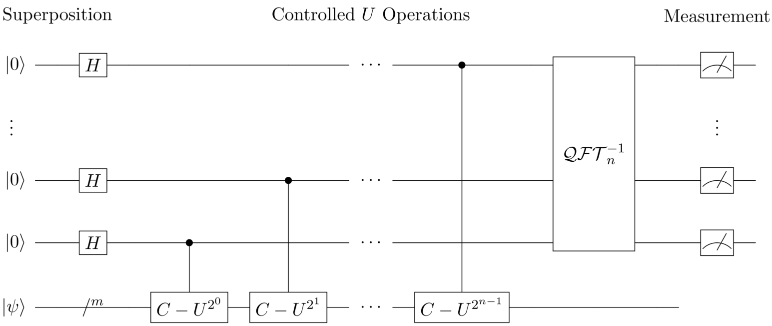
$$\text{If } U|\psi\rangle = \lambda|\psi\rangle \quad \text{then} \quad U, |\psi\rangle \xrightarrow{PE} \lambda$$

.

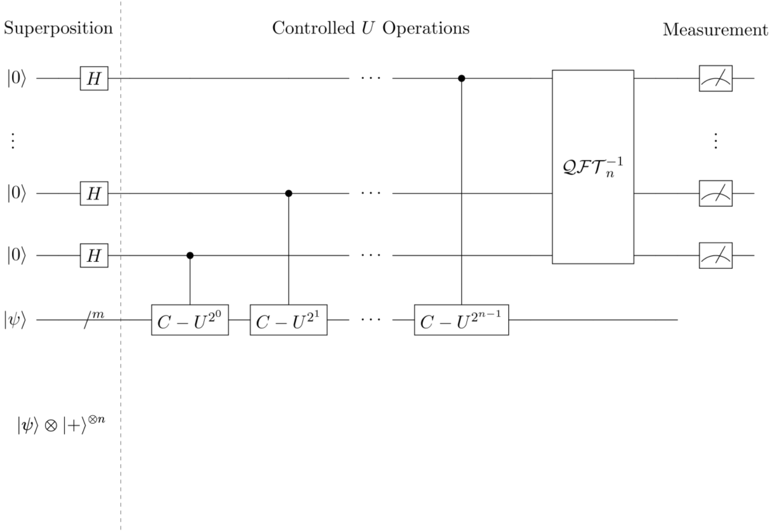
Really it returns $|k\rangle$, where the eigenvalue is $\lambda = e^{i\theta}$ with $\theta = k\frac{2\pi}{2^n}$.
(Or closest n-bit approx. with at least 40% probability)

So how do we do this? Easy!

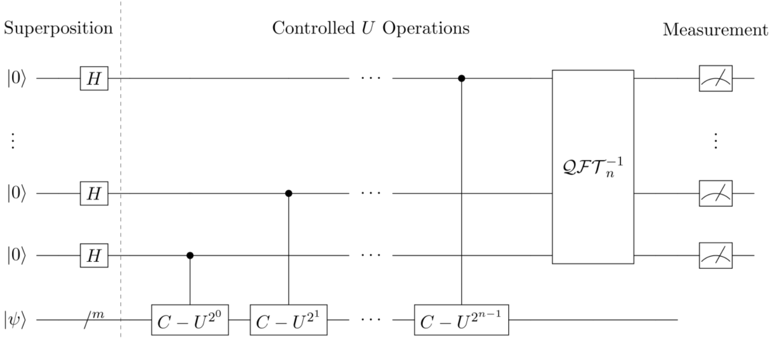
Phase Estimation: Not so bad!



Phase Estimation: Not so bad!

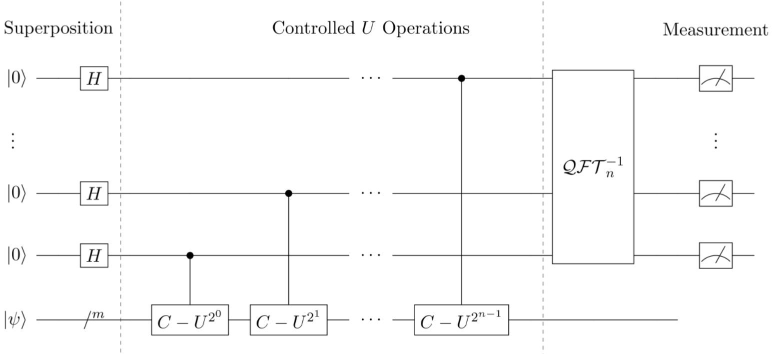


Phase Estimation: Not so bad!



$$|\psi\rangle \otimes |+\rangle^{\otimes n} \to |\psi\rangle \otimes (|0\rangle + e^{i\alpha}|1\rangle) \otimes (|0\rangle + e^{2i\alpha}|1\rangle) \otimes \ldots \otimes (|0\rangle + e^{2^{n-1}i\alpha}|1\rangle)$$

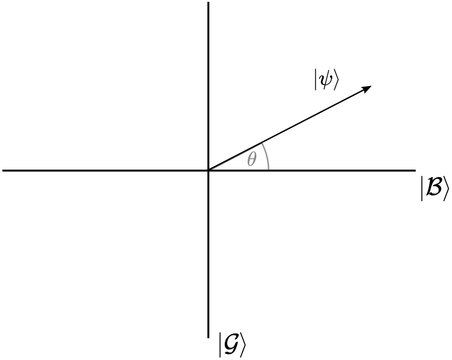
Phase Estimation: Not so bad!



$|\psi\rangle \otimes |+\rangle^{\otimes n} \rightarrow$ $|\psi\rangle\otimes$ $(|0\rangle + e^{i\alpha}|1\rangle) \otimes (|0\rangle + e^{2i\alpha}|1\rangle)\otimes \ldots \otimes(|0\rangle + e^{2^{n-1}i\alpha}|1\rangle)$ $\rightarrow$ $|\psi\rangle \otimes |k\rangle$ with $\alpha = k\frac{2\pi}{2^n}$ and k an integer $mod\ 2^n$

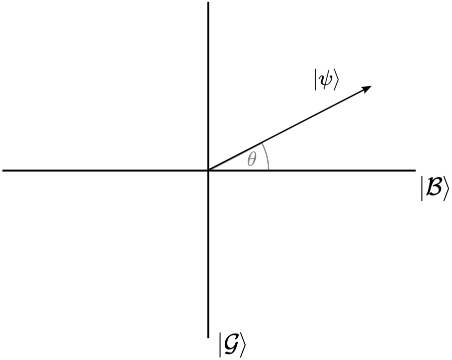
Amplitude Amplification

Amplitude estimation takes some statevector partitioned into a good and a bad subspace



Amplitude Amplification

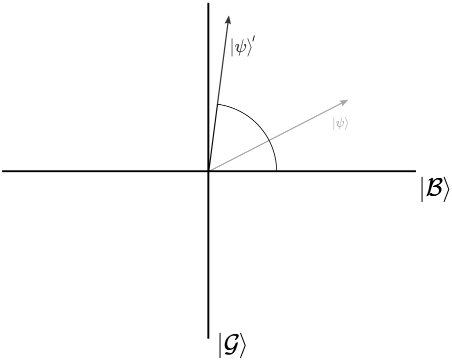
Amplitude estimation takes some statevector partitioned into a good and a bad subspace



$$|\psi\rangle = sin(\theta)|\mathcal{G}\rangle + cos(\theta)|\mathcal{B}\rangle.$$

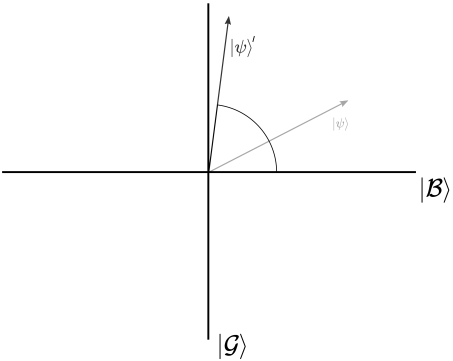
Amplitude Amplification

And returns a statevector nudged towards the good subspace.



Amplitude Amplification

And returns a statevector nudged towards the good subspace.



This is accomplished with consecutive applications of a special operator

Amplitude Amplification: Working principles

Remember $|\psi\rangle = sin(\theta)|\mathcal{G}\rangle + cos(\theta)|\mathcal{B}\rangle$.

Amplitude Amplification: Working principles

Remember $|\psi\rangle = sin(\theta)|\mathcal{G}\rangle + cos(\theta)|\mathcal{B}\rangle$.

We define operators:

Amplitude Amplification: Working principles

Remember $|\psi\rangle = sin(\theta)|\mathcal{G}\rangle + cos(\theta)|\mathcal{B}\rangle$.

We define operators:

1. $\mathcal{S}_{\mathcal{G}} = \mathbb{I} - 2|\mathcal{G}\rangle\langle\mathcal{G}|$

Amplitude Amplification: Working principles

Remember $|\psi\rangle = sin(\theta)|\mathcal{G}\rangle + cos(\theta)|\mathcal{B}\rangle$.

We define operators:

1. $\mathcal{S}_{\mathcal{G}} = \mathbb{I} - 2|\mathcal{G}\rangle\langle\mathcal{G}|$
2. $\mathcal{S}_{\psi} = \mathbb{I} - 2|\psi\rangle\langle\psi|$

Amplitude Amplification: Working principles

Remember $|\psi\rangle = sin(\theta)|\mathcal{G}\rangle + cos(\theta)|\mathcal{B}\rangle$.

We define operators:

1. $\mathcal{S}_{\mathcal{G}} = \mathbb{I} - 2|\mathcal{G}\rangle\langle\mathcal{G}|$
2. $\mathcal{S}_{\psi} = \mathbb{I} - 2|\psi\rangle\langle\psi|$
3. $\mathcal{Q} = -\mathcal{S}_{\psi}\mathcal{S}_{\mathcal{G}}$

Amplitude Amplification: Working principles

Remember $|\psi\rangle = sin(\theta)|\mathcal{G}\rangle + cos(\theta)|\mathcal{B}\rangle$.

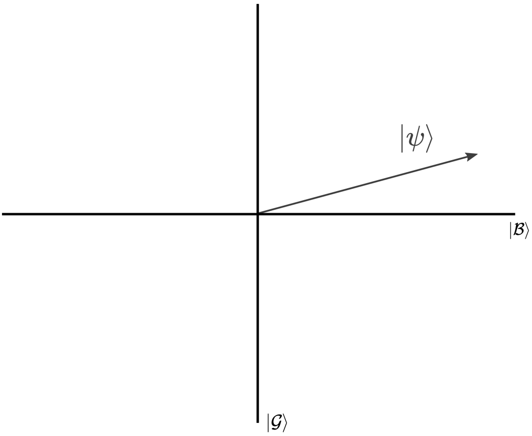
We define operators:

1. $\mathcal{S}_{\mathcal{G}} = \mathbb{I} - 2|\mathcal{G}\rangle\langle\mathcal{G}|$
2. $\mathcal{S}_{\psi} = \mathbb{I} - 2|\psi\rangle\langle\psi|$
3. $\mathcal{Q} = -\mathcal{S}_{\psi}\mathcal{S}_{\mathcal{G}}$

Let's see what they do!

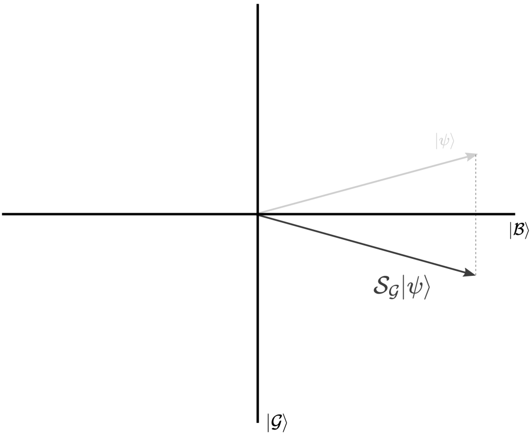
Amplitude Amplification: Visualization I

First apply the operator $\mathcal{S}_{\mathcal{G}}$,



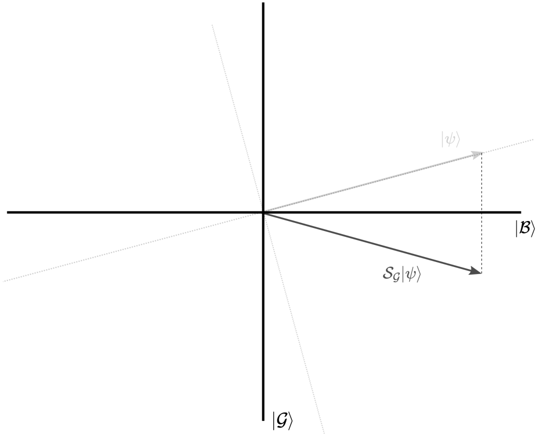
Amplitude Amplification: Visualization I

First apply the operator $\mathcal{S}_{\mathcal{G}}$,



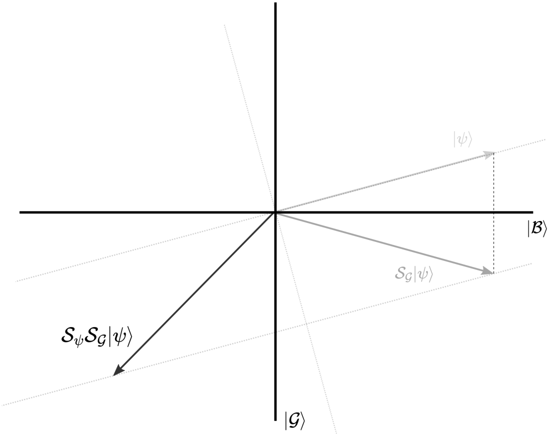
Amplitude Amplification: Visualization II

Then apply the operator $\mathcal{S}_\psi$,



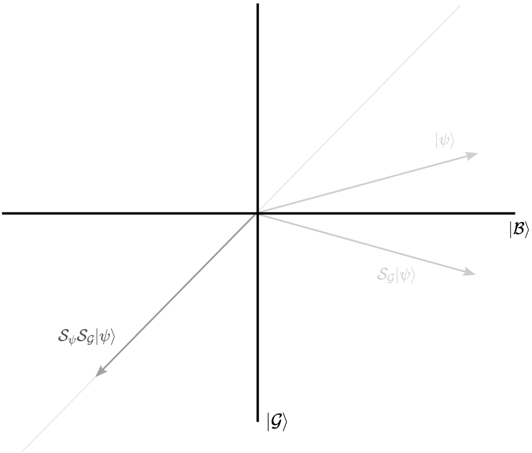
Amplitude Amplification: Visualization II

Then apply the operator $\mathcal{S}_\psi$,



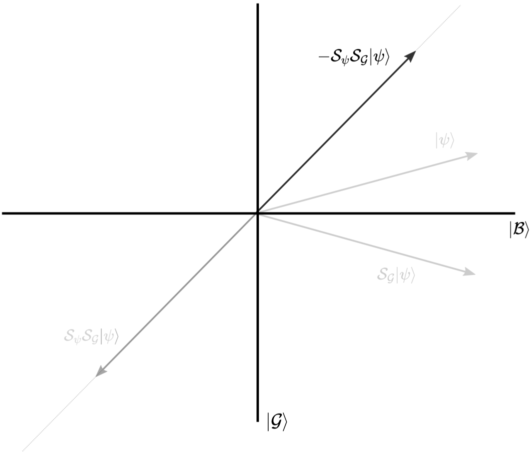
Amplitude Amplification: Visualization III

Now, negate the resulting statevector



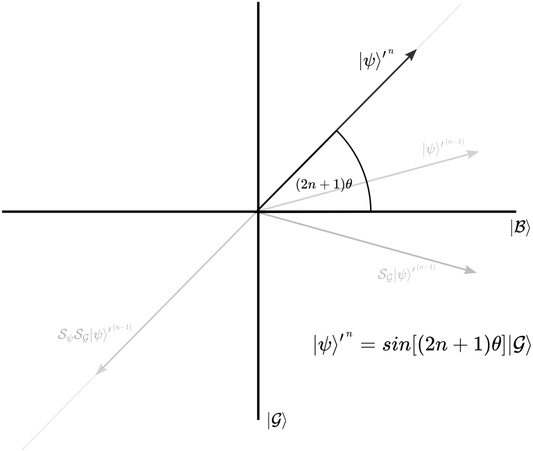
Amplitude Amplification: Visualization III

Now, negate the resulting statevector



Amplitude Amplification: Visualization IV

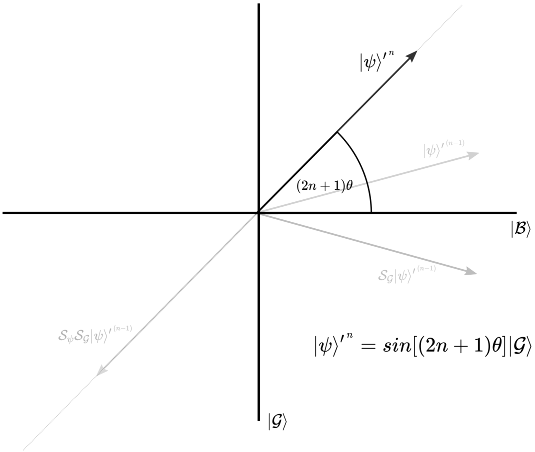
And voilá!



$$|\psi\rangle'^{n} = sin[(2n+1)\theta]|\mathcal{G}\rangle + cos[(2n+1)\theta]|\mathcal{B}\rangle$$

Amplitude Amplification: Visualization IV

And voilá!

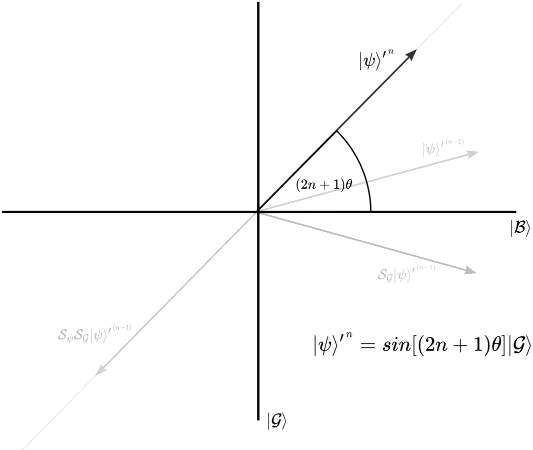


$$|\psi\rangle'^{n} = sin[(2n+1)\theta]|\mathcal{G}\rangle + cos[(2n+1)\theta]|\mathcal{B}\rangle$$

Though, we'd need to know *n* as well

Amplitude Amplification: Visualization IV

And voilá!

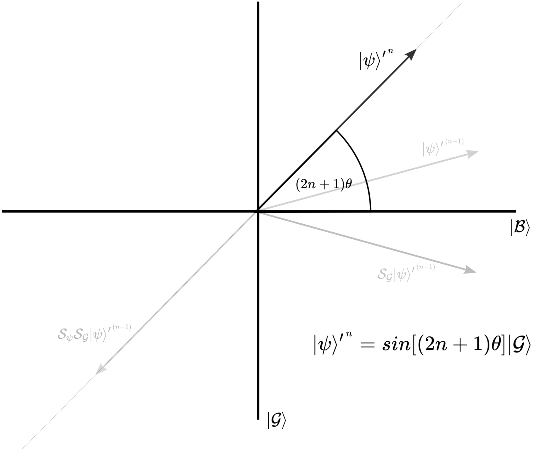


$$|\psi\rangle'^{n} = sin[(2n+1)\theta]|\mathcal{G}\rangle + cos[(2n+1)\theta]|\mathcal{B}\rangle$$

Though, we'd need to know *n* as well. Ideally $\lfloor \frac{\pi}{4\theta} \rfloor$

Amplitude Amplification: Visualization IV

And voilá!



$$|\psi\rangle'^{n} = sin[(2n+1)\theta]|\mathcal{G}\rangle + cos[(2n+1)\theta]|\mathcal{B}\rangle$$

Though, we'd need to know *n* as well. Ideally $\lfloor \frac{\pi}{4\theta} \rfloor$

Now, let's do some coding!

Qiskit

Qiskit is a quantum software devlopment kit with a python front-end partially developed by IBM.

Qiskit

Qiskit is a quantum software devlopment kit with a python front-end partially developed by IBM.

Features

1. Simulate + visualize quantum circuits you create yourself

Qiskit

Qiskit is a quantum software devlopment kit with a python front-end partially developed by IBM.

Features

1. Simulate + visualize quantum circuits you create yourself
2. Compatible with IBM's current quantum computers

Qiskit

Qiskit is a quantum software devlopment kit with a python front-end partially developed by IBM.

Features

1. Simulate + visualize quantum circuits you create yourself
2. Compatible with IBM's current quantum computers
3. Vibrant and active online community

Running the QFT

Let's run the QFT on some states in Qiskit.

Running the QFT

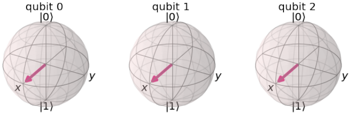
Let's run the QFT on some states in Qiskit. We'll use a statevector simulation.

Running the QFT

Let's run the QFT on some states in Qiskit. We'll use a statevector simulation. The local evolution of the qubits is depicted in the bloch spheres below:

Running the QFT

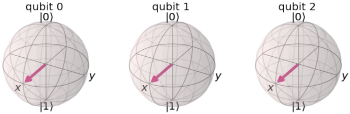
Let's run the QFT on some states in Qiskit. We'll use a statevector simulation. The local evolution of the qubits is depicted in the bloch spheres below:

Input: $|000\rangle \rightarrow$ 

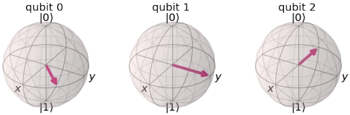
Running the QFT

Let's run the QFT on some states in Qiskit. We'll use a statevector simulation. The local evolution of the qubits is depicted in the bloch spheres below:

Input: $|000\rangle \rightarrow$



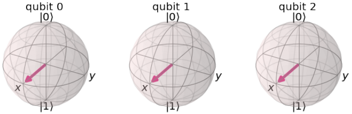
Input: $|001\rangle \rightarrow$



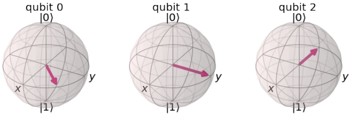
Running the QFT

Let's run the QFT on some states in Qiskit. We'll use a statevector simulation. The local evolution of the qubits is depicted in the bloch spheres below:

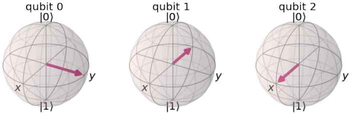
Input: $|000\rangle \rightarrow$



Input: $|001\rangle \rightarrow$



Input: $|010\rangle \rightarrow$



Where's the code?

For the QFT transforms just shown, it's rather simple:

Where's the code?

For the QFT transforms just shown, it's rather simple:

```
from qiskit.circuit.library import QFT
qft = QFT(3)
qft3_000 = execute(qft, backend).result()
plot_bloch_multivector(qft3_000.get_statevector())
```

Where's the code?

For the QFT transforms just shown, it's rather simple:

```
from qiskit.circuit.library import QFT
qft = QFT(3)
qft3_000 = execute(qft, backend).result()
plot_bloch_multivector(qft3_000.get_statevector())
```

Moving forward, I'll just link the code online.

Finding Phases

Let's run the 3-qubit phase estimation algorithim on some gates.

Finding Phases

Let's run the 3-qubit phase estimation algorithim on some gates.

We'll use controlled phase gates

Finding Phases

Let's run the 3-qubit phase estimation algorithim on some gates.

We'll use controlled phase gates, with matrices of the form

$$CP(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\alpha} \end{bmatrix}.$$

Finding Phases

Let's run the 3-qubit phase estimation algorithim on some gates.

We'll use controlled phase gates, with matrices of the form

$$CP(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\alpha} \end{bmatrix}.$$

We already know that $|1\rangle$ is an eigenvector of basic phase gates.

Finding Phases: CP($\pi/4$)

This time, we'll use the qasm simulator

Finding Phases: CP($\pi/4$)

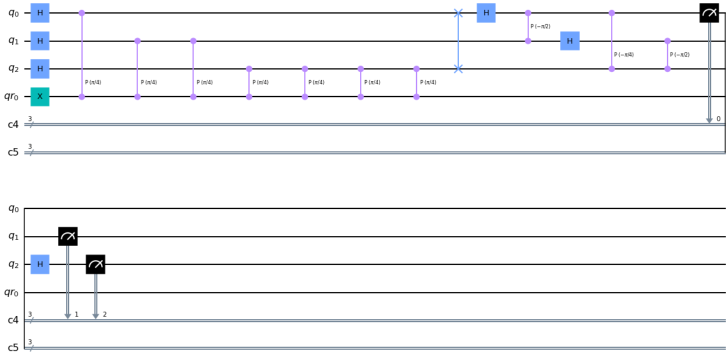
This time, we'll use the qasm simulator. This lets us get counts of simulated measurements at the end

Finding Phases: CP($\pi/4$)

This time, we'll use the qasm simulator. This lets us get counts of simulated measurements at the end. So, let's see the circuit:

Finding Phases: CP($\pi/4$)

This time, we'll use the qasm simulator. This lets us get counts of simulated measurements at the end. So, let's see the circuit:

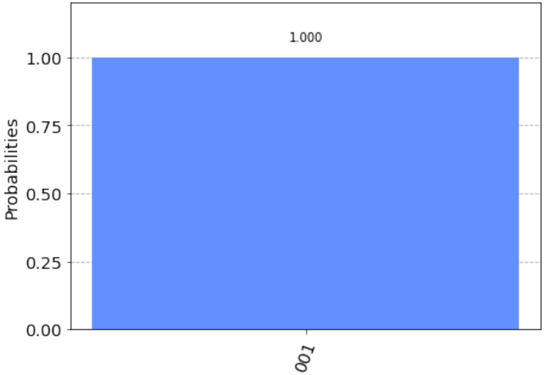


Finding Phases: CP($\pi/4$)

Running the simulation gives a simulated measurement set:

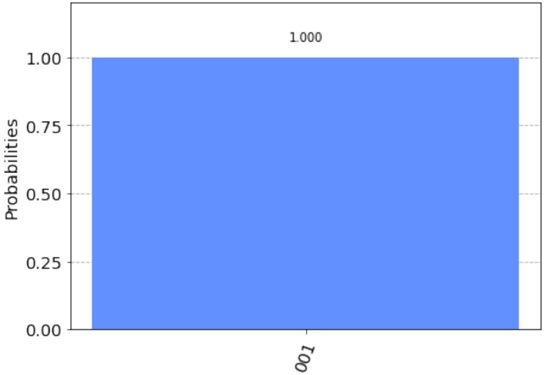
Finding Phases: CP($\pi/4$)

Running the simulation gives a simulated measurement set:



Finding Phases: CP($\pi/4$)

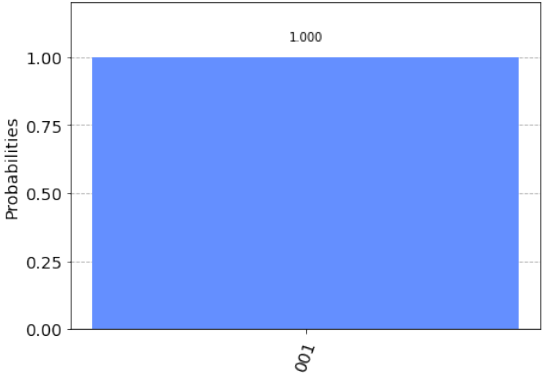
Running the simulation gives a simulated measurement set:



Is this what we would expect?

Finding Phases: CP($\pi/4$)

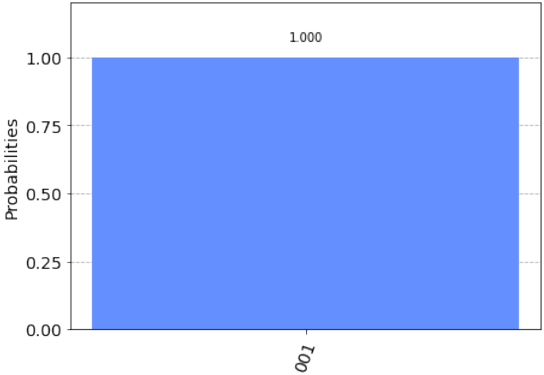
Running the simulation gives a simulated measurement set:



Is this what we would expect? Yes!

Finding Phases: CP($\pi/4$)

Running the simulation gives a simulated measurement set:

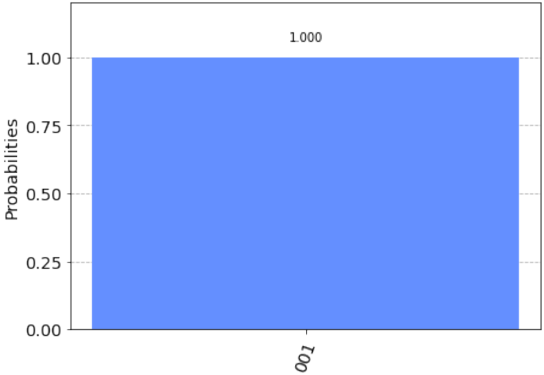


Is this what we would expect? Yes!

As the phase can be encoded perfectly in 3 qubits, we should expect the output vector to be $|k\rangle = |\alpha \frac{2^{n-1}}{\pi}\rangle$

Finding Phases: CP($\pi/4$)

Running the simulation gives a simulated measurement set:



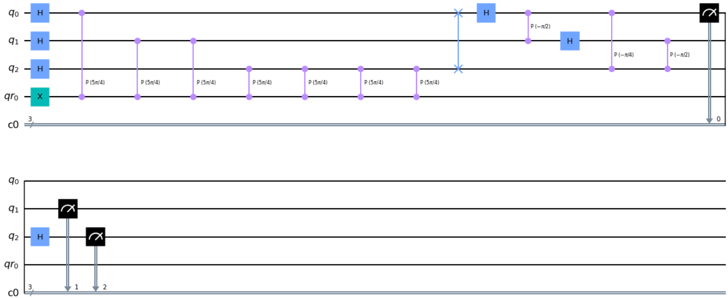
Is this what we would expect? Yes!
As the phase can be encoded perfectly in 3 qubits, we should expect the output vector to be $|k\rangle = |\alpha \frac{2^{n-1}}{\pi}\rangle$ or $|001\rangle$ in binary

Finding Phases: CP($5\pi/4$)

Let's try it again with another ideal phase:

Finding Phases: CP($5\pi/4$)

Let's try it again with another ideal phase:

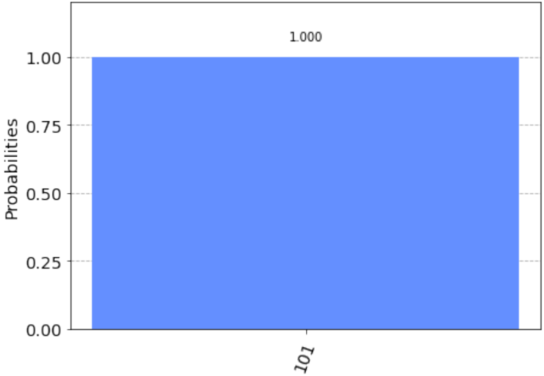


Finding Phases: CP($5\pi/4$)

Now, we should get $|5\rangle = |101\rangle$:

Finding Phases: CP($5\pi/4$)

Now, we should get $|5\rangle = |101\rangle$:



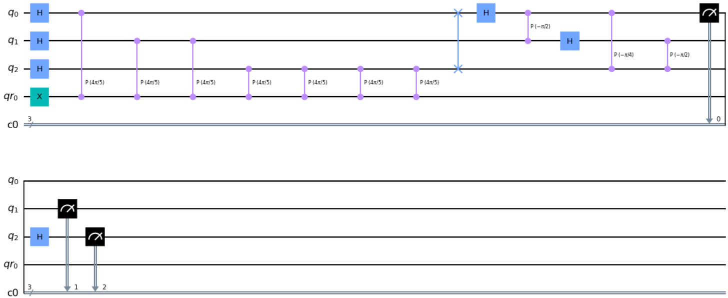
Yup!

Finding Phases: CP($4\pi/5$)

Let's try something that's not an ideal phase:

Finding Phases: CP($4\pi/5$)

Let's try something that's not an ideal phase:

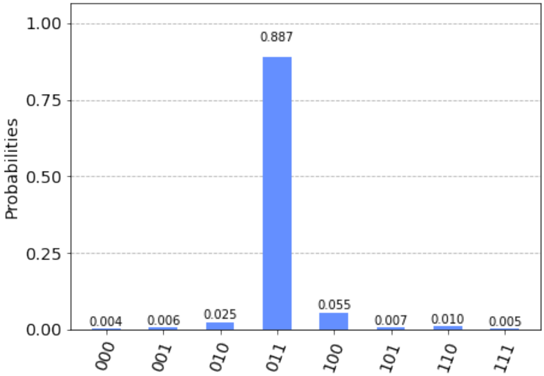


Finding Phases: CP($4\pi/5$)

Let's see what we get when we run the simulation several times:

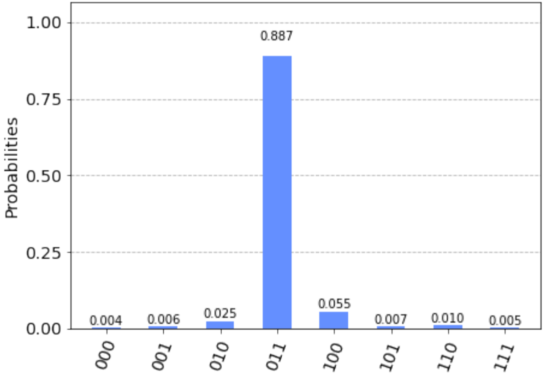
Finding Phases: CP($4\pi/5$)

Let's see what we get when we run the simulation several times:



Finding Phases: CP($4\pi/5$)

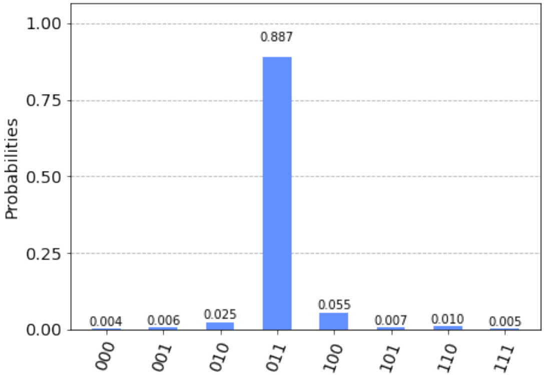
Let's see what we get when we run the simulation several times:



This seems to be a good estimate given $\frac{4\pi}{5}\frac{4}{\pi} = 3.2 \approx 3$.

Finding Phases: $CP(4\pi/5)$

Let's see what we get when we run the simulation several times:

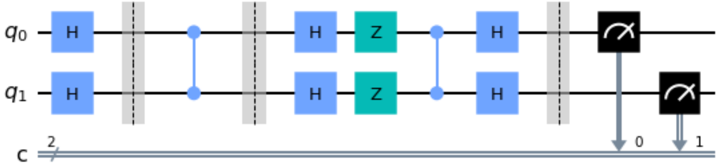


This seems to be a good estimate given $\frac{4\pi}{5}\frac{4}{\pi} = 3.2 \approx 3$.

Let's do some amplification!

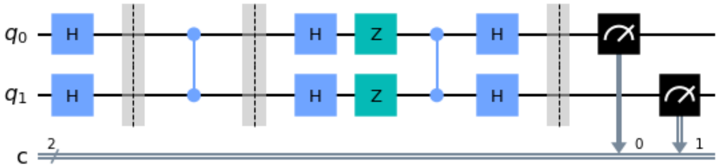
Amplitude Amplification in Practice

Below is a simple 2-qubit circuit for amplitude amplification that searches for the $|11\rangle$ state:



Amplitude Amplification in Practice

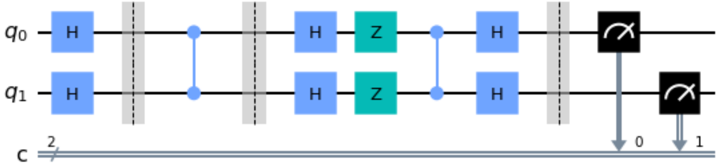
Below is a simple 2-qubit circuit for amplitude amplification that searches for the $|11\rangle$ state:



The first section initializes the state into $|+\rangle^{\otimes 2}$

Amplitude Amplification in Practice

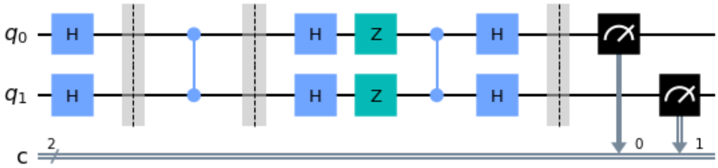
Below is a simple 2-qubit circuit for amplitude amplification that searches for the $|11\rangle$ state:



The first section initializes the state into $|+\rangle^{\otimes 2}$.
The second portion flips the sign of only $|11\rangle$

Amplitude Amplification in Practice

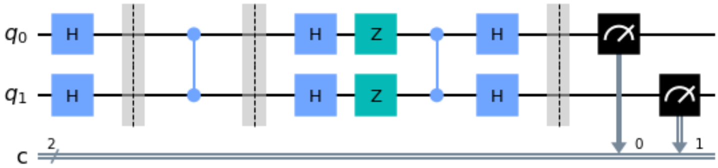
Below is a simple 2-qubit circuit for amplitude amplification that searches for the $|11\rangle$ state:



The first section initializes the state into $|+\rangle^{\otimes 2}$.
The second portion flips the sign of only $|11\rangle$.
The third section flips the state about the original vector.

Amplitude Amplification in Practice

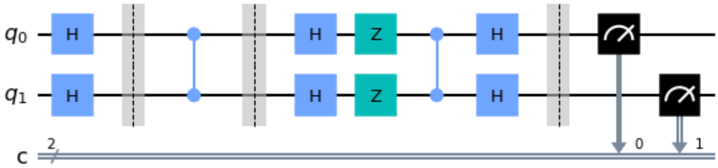
Below is a simple 2-qubit circuit for amplitude amplification that searches for the $|11\rangle$ state:



The first section initializes the state into $|+\rangle^{\otimes 2}$.
The second portion flips the sign of only $|11\rangle$.
The third section flips the state about the original vector.
We only need to run the operations once as $n = 1$ here

Amplitude Amplification in Practice

Below is a simple 2-qubit circuit for amplitude amplification that searches for the $|11\rangle$ state:

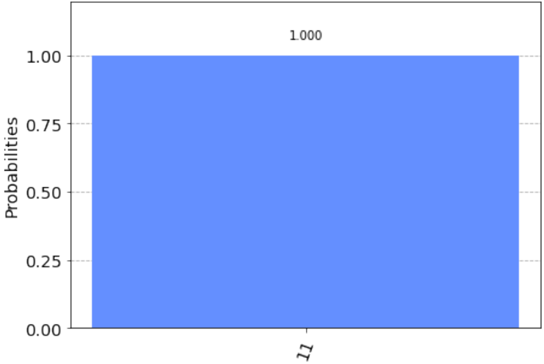


The first section initializes the state into $|+\rangle^{\otimes 2}$.
The second portion flips the sign of only $|11\rangle$.
The third section flips the state about the original vector.
We only need to run the operations once as $n = 1$ here.

Let's run it!

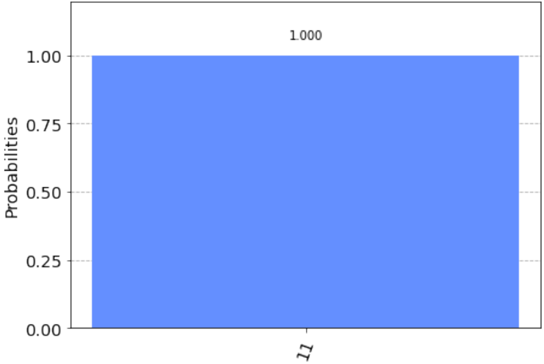
Amplitude Amplification in Practice: Simulation

Running the previous circuit on the simulation we used for phase estimation:



Amplitude Amplification in Practice: Simulation

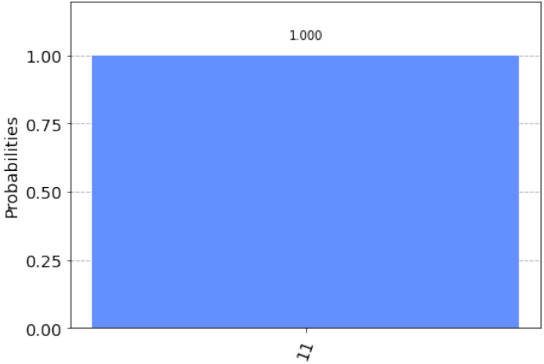
Running the previous circuit on the simulation we used for phase estimation:



Perfect results!

Amplitude Amplification in Practice: Simulation

Running the previous circuit on the simulation we used for phase estimation:



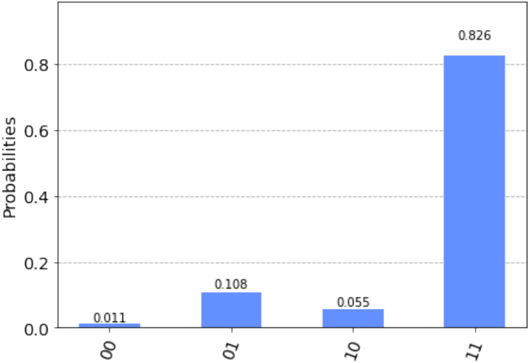
Perfect results! Let's run it for real!

Amplitude Amplification in Practice: Calling IBMQ

Now, running our circuit on IBMQ-Lima, a real 5-qubit quantum computer:

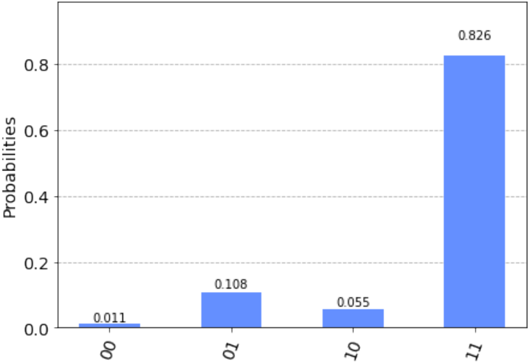
Amplitude Amplification in Practice: Calling IBMQ

Now, running our circuit on IBMQ-Lima, a real 5-qubit quantum computer:



Amplitude Amplification in Practice: Calling IBMQ

Now, running our circuit on IBMQ-Lima, a real 5-qubit quantum computer:



Not too bad!

Wrapping Up!

Basic Quantum Algorithms

Wrapping Up!

Basic Quantum Algorithms

- Quantum Fourier Transform

Wrapping Up!

Basic Quantum Algorithms

- Quantum Fourier Transform
- Phase Estimation

Wrapping Up!

Basic Quantum Algorithms

- Quantum Fourier Transform
- Phase Estimation
- Amplitude Amplification

Wrapping Up!

Basic Quantum Algorithms

- Quantum Fourier Transform
- Phase Estimation
- Amplitude Amplification

Qiskit

Wrapping Up!

Basic Quantum Algorithms

- Quantum Fourier Transform
- Phase Estimation
- Amplitude Amplification

Qiskit

- Visualize qubit evolution

Wrapping Up!

Basic Quantum Algorithms

- Quantum Fourier Transform
- Phase Estimation
- Amplitude Amplification

Qiskit

- Visualize qubit evolution
- Simulate experiments on circuits

Wrapping Up!

Basic Quantum Algorithms

- Quantum Fourier Transform
- Phase Estimation
- Amplitude Amplification

Qiskit

- Visualize qubit evolution
- Simulate experiments on circuits
- Run circuits on real computers!

Moving Forward

Where to go from here?

Moving Forward

Where to go from here?

- Build to more complex applications

Moving Forward

Where to go from here?

- Build to more complex applications
- Extend to quantum machine learning

Moving Forward

Where to go from here?

- Build to more complex applications
- Extend to quantum machine learning
- Run some interesting experiments

That's it!

Notebooks for the circuits run here can be found at
`https://alexheilman.com`

References I

[1] *Qiskit: An Open-source Framework for Quantum Computing*. 2019. DOI: 10.5281/zenodo.2562110.

[2] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. 10th. USA: Cambridge University Press, 2011. ISBN: 1107002176.

[3] *n-qubit QFT diagram source*. https://jonathan-hui.medium.com/qc-quantum-fourier-transform-45436f90a43.

[4] *Phase estimation diagram source*. https://en.wikipedia.org/wiki/Quantum_phase_estimation_algorithm#/media/File:PhaseCircuit-crop.svg.

[5] *Qiskit Textbook*. https://qiskit.org/textbook/preface.html.

References II

[6] Gilles Brassard, Peter Hyer, Michele Mosca, et al. “Quantum amplitude amplification and estimation”. In: *Quantum Computation and Information* (2002), 53Ž01374. ISSN: 0271-4132. DOI: 10.1090/conm/305/05215. URL: http://dx.doi.org/10.1090/conm/305/05215.

Thanks