Today's outline - February 22, 2022

Today's outline - February 22, 2022

- A word about hands-on component

Today's outline - February 22, 2022

- A word about hands-on component
- Distributed computation

Today's outline - February 22, 2022

- A word about hands-on component
- Distributed computation
- Quantum Fourier transform

Today's outline - February 22, 2022

- A word about hands-on component
- Distributed computation
- Quantum Fourier transform
- Quantum Fourier transform circuits

Today's outline - February 22, 2022

- A word about hands-on component
- Distributed computation
- Quantum Fourier transform
- Quantum Fourier transform circuits

Reading Assignment: Chapter 8.1-8.2

Today's outline - February 22, 2022

- A word about hands-on component
- Distributed computation
- Quantum Fourier transform
- Quantum Fourier transform circuits

Reading Assignment: Chapter 8.1-8.2

Homework Assignment #05:
Chapter 7:1,3,4
due Thursday, March 03, 2022

Today's outline - February 22, 2022

- A word about hands-on component
- Distributed computation
- Quantum Fourier transform
- Quantum Fourier transform circuits

Reading Assignment: Chapter 8.1-8.2

Homework Assignment #05:
Chapter 7:1,3,4
due Thursday, March 03, 2022

Exam #1 Tuesday, March 01, 2022
Covers Chapters 2-5

Today's outline - February 22, 2022

- A word about hands-on component
- Distributed computation
- Quantum Fourier transform
- Quantum Fourier transform circuits

Reading Assignment: Chapter 8.1-8.2

Homework Assignment #05:
Chapter 7:1,3,4
due Thursday, March 03, 2022

Exam #1 Tuesday, March 01, 2022
Covers Chapters 2-5

Quantum circuit simulator https://algassert.com/quirk

Distributed computation

Alice and Bob are each provided with an $N = 2^n$ bit number, u and v respectively

Distributed computation

Alice and Bob are each provided with an $N = 2^n$ bit number, u and v respectively

Alice must compute an n-bit number a and Bob must compute an n-bit number b such that

Distributed computation

Alice and Bob are each provided with an $N = 2^n$ bit number, u and v respectively

Alice must compute an n-bit number a and Bob must compute an n-bit number b such that

$$d_H(u, v) = 0 \quad \longrightarrow \quad a = b$$

Distributed computation

Alice and Bob are each provided with an $N = 2^n$ bit number, u and v respectively

Alice must compute an n-bit number a and Bob must compute an n-bit number b such that

$$\begin{aligned} d_H(u, v) = 0 \quad &\longrightarrow \quad a = b \\ d_H(u, v) = N/2 \quad &\longrightarrow \quad a \neq b \end{aligned}$$

Distributed computation

Alice and Bob are each provided with an $N = 2^n$ bit number, u and v respectively

Alice must compute an n-bit number a and Bob must compute an n-bit number b such that

$$\begin{aligned} d_H(u, v) = 0 &\longrightarrow a = b \\ d_H(u, v) = N/2 &\longrightarrow a \neq b \\ \text{else} &\longrightarrow \text{no condition on } a \text{ and } b \end{aligned}$$

Distributed computation

Alice and Bob are each provided with an $N = 2^n$ bit number, u and v respectively

Alice must compute an n-bit number a and Bob must compute an n-bit number b such that

$$\begin{array}{ccc} d_H(u,v) = 0 & \longrightarrow & a = b \\ d_H(u,v) = N/2 & \longrightarrow & a \neq b \\ \text{else} & \longrightarrow & \text{no condition on } a \text{ and } b \end{array}$$

This is a challenging problem because u and v are exponentially larger than a and b

Distributed computation

Alice and Bob are each provided with an $N = 2^n$ bit number, u and v respectively

Alice must compute an n-bit number a and Bob must compute an n-bit number b such that

$$\begin{array}{ccc} d_H(u,v) = 0 & \longrightarrow & a = b \\ d_H(u,v) = N/2 & \longrightarrow & a \neq b \\ \text{else} & \longrightarrow & \text{no condition on } a \text{ and } b \end{array}$$

This is a challenging problem because u and v are exponentially larger than a and b

A classical solution requires a communication of at least $N/2$ bits but with enough entangled pairs, no additional communication is needed in a quantum solution

Distributed computation

Alice and Bob are each provided with an $N = 2^n$ bit number, u and v respectively

Alice must compute an n-bit number a and Bob must compute an n-bit number b such that

$$\begin{array}{ccl} d_H(u,v) = 0 & \longrightarrow & a = b \\ d_H(u,v) = N/2 & \longrightarrow & a \neq b \\ \text{else} & \longrightarrow & \text{no condition on } a \text{ and } b \end{array}$$

This is a challenging problem because u and v are exponentially larger than a and b

A classical solution requires a communication of at least $N/2$ bits but with enough entangled pairs, no additional communication is needed in a quantum solution

Start with n entangled pairs of particles, (a_i, b_i) in states $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ with

Distributed computation

Alice and Bob are each provided with an $N = 2^n$ bit number, u and v respectively

Alice must compute an n-bit number a and Bob must compute an n-bit number b such that

$$\begin{array}{ccc} d_H(u,v) = 0 & \longrightarrow & a = b \\ d_H(u,v) = N/2 & \longrightarrow & a \neq b \\ \text{else} & \longrightarrow & \text{no condition on } a \text{ and } b \end{array}$$

This is a challenging problem because u and v are exponentially larger than a and b

A classical solution requires a communication of at least $N/2$ bits but with enough entangled pairs, no additional communication is needed in a quantum solution

Start with n entangled pairs of particles, (a_i, b_i) in states $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ with

$$a_0, a_1, \ldots, a_{n-1}, b_0, b_1, \ldots, b_{n-1} \longrightarrow |\psi\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i, i\rangle$$

Distributed computation

Alice uses the phase change subroutine with
$f(i) = u_i$

Distributed computation

Alice uses the phase change subroutine with $f(i) = u_i$

$$\sum_{i=0}^{N-1} |i\rangle \longrightarrow \sum_{i=0}^{N-1} (-1)^{u_i} |i\rangle$$

Distributed computation

Alice uses the phase change subroutine with $f(i) = u_i$

$$\sum_{i=0}^{N-1} |i\rangle \longrightarrow \sum_{i=0}^{N-1} (-1)^{u_i} |i\rangle$$

Bob uses the phase change subroutine with $f(i) = v_i$

Distributed computation

Alice uses the phase change subroutine with $f(i) = u_i$

$$\sum_{i=0}^{N-1} |i\rangle \longrightarrow \sum_{i=0}^{N-1} (-1)^{u_i} |i\rangle$$

Bob uses the phase change subroutine with $f(i) = v_i$

$$\sum_{i=0}^{N-1} |i\rangle \longrightarrow \sum_{i=0}^{N-1} (-1)^{v_i} |i\rangle$$

Distributed computation

Alice uses the phase change subroutine with $f(i) = u_i$

$$\sum_{i=0}^{N-1} |i\rangle \longrightarrow \sum_{i=0}^{N-1} (-1)^{u_i} |i\rangle$$

Bob uses the phase change subroutine with $f(i) = v_i$

$$\sum_{i=0}^{N-1} |i\rangle \longrightarrow \sum_{i=0}^{N-1} (-1)^{v_i} |i\rangle$$

They each apply the Walsh transformation to get a common global state

Distributed computation

Alice uses the phase change subroutine with $f(i) = u_i$

$$\sum_{i=0}^{N-1} |i\rangle \longrightarrow \sum_{i=0}^{N-1} (-1)^{u_i} |i\rangle$$

Bob uses the phase change subroutine with $f(i) = v_i$

$$\sum_{i=0}^{N-1} |i\rangle \longrightarrow \sum_{i=0}^{N-1} (-1)^{v_i} |i\rangle$$

They each apply the Walsh transformation to get a common global state

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} (-1)^{u_i \oplus v_i} \left(W|i\rangle \otimes W|i\rangle \right)$$

Distributed computation

Alice uses the phase change subroutine with $f(i) = u_i$

$$\sum_{i=0}^{N-1} |i\rangle \longrightarrow \sum_{i=0}^{N-1} (-1)^{u_i} |i\rangle$$

Bob uses the phase change subroutine with $f(i) = v_i$

$$\sum_{i=0}^{N-1} |i\rangle \longrightarrow \sum_{i=0}^{N-1} (-1)^{v_i} |i\rangle$$

They each apply the Walsh transformation to get a common global state

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} (-1)^{u_i \oplus v_i} \left(W|i\rangle \otimes W|i\rangle \right) = \frac{1}{N\sqrt{N}} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} (-1)^{u_i \oplus v_i} (-1)^{i \cdot j} (-1)^{i \cdot k} |jk\rangle$$

Distributed computation

Alice uses the phase change subroutine with $f(i) = u_i$

$$\sum_{i=0}^{N-1} |i\rangle \longrightarrow \sum_{i=0}^{N-1} (-1)^{u_i} |i\rangle$$

Bob uses the phase change subroutine with $f(i) = v_i$

$$\sum_{i=0}^{N-1} |i\rangle \longrightarrow \sum_{i=0}^{N-1} (-1)^{v_i} |i\rangle$$

They each apply the Walsh transformation to get a common global state

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} (-1)^{u_i \oplus v_i} \left(W|i\rangle \otimes W|i\rangle \right) = \frac{1}{N\sqrt{N}} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} (-1)^{u_i \oplus v_i} (-1)^{i \cdot j} (-1)^{i \cdot k} |jk\rangle$$

The probability that the measurement results in $a = x = b$ is the modulus squared of $\langle x, x|\psi\rangle$

Distributed computation

Alice uses the phase change subroutine with $f(i) = u_i$

$$\sum_{i=0}^{N-1} |i\rangle \longrightarrow \sum_{i=0}^{N-1} (-1)^{u_i} |i\rangle$$

Bob uses the phase change subroutine with $f(i) = v_i$

$$\sum_{i=0}^{N-1} |i\rangle \longrightarrow \sum_{i=0}^{N-1} (-1)^{v_i} |i\rangle$$

They each apply the Walsh transformation to get a common global state

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} (-1)^{u_i \oplus v_i} \left(W|i\rangle \otimes W|i\rangle \right) = \frac{1}{N\sqrt{N}} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} (-1)^{u_i \oplus v_i} (-1)^{i \cdot j} (-1)^{i \cdot k} |jk\rangle$$

The probability that the measurement results in $a = x = b$ is the modulus squared of $\langle x, x|\psi\rangle$

$$\langle x, x|\psi\rangle = \frac{1}{N\sqrt{N}} \sum_{i=0}^{N-1} (-1)^{u_i \oplus v_i} (-1)^{i \cdot x} (-1)^{i \cdot x}$$

Distributed computation

Alice uses the phase change subroutine with $f(i) = u_i$

$$\sum_{i=0}^{N-1} |i\rangle \longrightarrow \sum_{i=0}^{N-1} (-1)^{u_i} |i\rangle$$

Bob uses the phase change subroutine with $f(i) = v_i$

$$\sum_{i=0}^{N-1} |i\rangle \longrightarrow \sum_{i=0}^{N-1} (-1)^{v_i} |i\rangle$$

They each apply the Walsh transformation to get a common global state

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} (-1)^{u_i \oplus v_i} \left(W|i\rangle \otimes W|i\rangle \right) = \frac{1}{N\sqrt{N}} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} (-1)^{u_i \oplus v_i} (-1)^{i \cdot j} (-1)^{i \cdot k} |jk\rangle$$

The probability that the measurement results in $a = x = b$ is the modulus squared of $\langle x, x|\psi\rangle$

$$\langle x, x|\psi\rangle = \frac{1}{N\sqrt{N}} \sum_{i=0}^{N-1} (-1)^{u_i \oplus v_i} (-1)^{i \cdot x} (-1)^{i \cdot x} = \frac{1}{N\sqrt{N}} \sum_{i=0}^{N-1} (-1)^{u_i \oplus v_i}$$

Distributed computation

The probability that Alice and Bob measure the same n bit value, x, is given by

Distributed computation

The probability that Alice and Bob measure the same n bit value, x, is given by

$$P_{xx} = |\langle x, x|\psi\rangle|^2$$

Distributed computation

The probability that Alice and Bob measure the same n bit value, x, is given by

$$P_{xx} = |\langle x, x|\psi\rangle|^2 = \left| \frac{1}{N\sqrt{N}} \sum_{i=0}^{N-1} (-1)^{u_i \oplus v_i} \right|^2$$

Distributed computation

The probability that Alice and Bob measure the same n bit value, x, is given by

$$P_{xx} = |\langle x, x|\psi\rangle|^2 = \left| \frac{1}{N\sqrt{N}} \sum_{i=0}^{N-1} (-1)^{u_i \oplus v_i} \right|^2$$

If $u = v$, then $(-1)^{u_i \oplus v_i} = 1$

Distributed computation

The probability that Alice and Bob measure the same n bit value, x, is given by

$$P_{xx} = |\langle x, x|\psi\rangle|^2 = \left| \frac{1}{N\sqrt{N}} \sum_{i=0}^{N-1} (-1)^{u_i \oplus v_i} \right|^2$$

If $u = v$, then $(-1)^{u_i \oplus v_i} = 1$

$$\langle x, x|\psi\rangle = \frac{1}{\sqrt{N}}$$

Distributed computation

The probability that Alice and Bob measure the same n bit value, x, is given by

$$P_{xx} = |\langle x, x|\psi\rangle|^2 = \left| \frac{1}{N\sqrt{N}} \sum_{i=0}^{N-1} (-1)^{u_i \oplus v_i} \right|^2$$

If $u = v$, then $(-1)^{u_i \oplus v_i} = 1$ so when summed over the N possible values of x, $P_{xx} = 1$ and Alice and Bob will measure $a = b$ with probability 1

$$\langle x, x|\psi\rangle = \frac{1}{\sqrt{N}}$$

Distributed computation

The probability that Alice and Bob measure the same n bit value, x, is given by

$$P_{xx} = |\langle x, x|\psi\rangle|^2 = \left| \frac{1}{N\sqrt{N}} \sum_{i=0}^{N-1} (-1)^{u_i \oplus v_i} \right|^2$$

If $u = v$, then $(-1)^{u_i \oplus v_i} = 1$ so when summed over the N possible values of x, $P_{xx} = 1$ and Alice and Bob will measure $a = b$ with probability 1

$$\langle x, x|\psi\rangle = \frac{1}{\sqrt{N}}$$

For $d_H(u, v) = N/2$ there will be exactly the same number of 1 and -1 values in the sum

Distributed computation

The probability that Alice and Bob measure the same n bit value, x, is given by

$$P_{xx} = |\langle x, x|\psi\rangle|^2 = \left| \frac{1}{N\sqrt{N}} \sum_{i=0}^{N-1} (-1)^{u_i \oplus v_i} \right|^2$$

If $u = v$, then $(-1)^{u_i \oplus v_i} = 1$ so when summed over the N possible values of x, $P_{xx} = 1$ and Alice and Bob will measure $a = b$ with probability 1

$$\langle x, x|\psi\rangle = \frac{1}{\sqrt{N}}$$

For $d_H(u, v) = N/2$ there will be exactly the same number of 1 and -1 values in the sum so $P_{xx} = 0$ and Alice and Bob will measure $a = b$ with probability 0

$$\langle x, x|\psi\rangle = 0$$

Discrete Fourier transform

The quantum Fourier transform is an important building block for many quantum algorithms

Discrete Fourier transform

The quantum Fourier transform is an important building block for many quantum algorithms

In order to develop the efficient implementation of the quantum Fourier transform, it is useful to start with the classical discrete and fast Fourier transforms

Discrete Fourier transform

The quantum Fourier transform is an important building block for many quantum algorithms

In order to develop the efficient implementation of the quantum Fourier transform, it is useful to start with the classical discrete and fast Fourier transforms

The discrete Fourier transform (DFT) is a linear transformation which takes a discrete column vector $a(k)$ to a column vector of Fourier coefficients, $A(x)$, where $0 \leq k, x \leq N-1$

Discrete Fourier transform

The quantum Fourier transform is an important building block for many quantum algorithms

In order to develop the efficient implementation of the quantum Fourier transform, it is useful to start with the classical discrete and fast Fourier transforms

The discrete Fourier transform (DFT) is a linear transformation which takes a discrete column vector $a(k)$ to a column vector of Fourier coefficients, $A(x)$, where $0 \leq k, x \leq N-1$

$$A(x) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a(k) e^{2\pi i k x/N}$$

Discrete Fourier transform

The quantum Fourier transform is an important building block for many quantum algorithms

In order to develop the efficient implementation of the quantum Fourier transform, it is useful to start with the classical discrete and fast Fourier transforms

The discrete Fourier transform (DFT) is a linear transformation which takes a discrete column vector $a(k)$ to a column vector of Fourier coefficients, $A(x)$, where $0 \leq k, x \leq N-1$

$$A(x) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a(k) e^{2\pi i k x / N}$$

The DFT operator is an $N \times N$ matrix with elements

Discrete Fourier transform

The quantum Fourier transform is an important building block for many quantum algorithms

In order to develop the efficient implementation of the quantum Fourier transform, it is useful to start with the classical discrete and fast Fourier transforms

The discrete Fourier transform (DFT) is a linear transformation which takes a discrete column vector $a(k)$ to a column vector of Fourier coefficients, $A(x)$, where $0 \leq k, x \leq N-1$

$$A(x) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a(k) e^{2\pi ikx/N}$$

The DFT operator is an $N \times N$ matrix with elements

$$F_{xk} = \tfrac{1}{\sqrt{N}} e^{2\pi ikx/N}$$

Discrete Fourier transform

The quantum Fourier transform is an important building block for many quantum algorithms

In order to develop the efficient implementation of the quantum Fourier transform, it is useful to start with the classical discrete and fast Fourier transforms

The discrete Fourier transform (DFT) is a linear transformation which takes a discrete column vector $a(k)$ to a column vector of Fourier coefficients, $A(x)$, where $0 \leq k, x \leq N-1$

$$A(x) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a(k) e^{2\pi ikx/N}$$

The DFT operator is an $N \times N$ matrix with elements

$$F_{xk} = \tfrac{1}{\sqrt{N}} e^{2\pi ikx/N}$$

Assume $a(k) = e^{-2\pi iuk/N}$ is a function of frequency $u < N$ which evenly divides N

Discrete Fourier transform

The quantum Fourier transform is an important building block for many quantum algorithms

In order to develop the efficient implementation of the quantum Fourier transform, it is useful to start with the classical discrete and fast Fourier transforms

The discrete Fourier transform (DFT) is a linear transformation which takes a discrete column vector $a(k)$ to a column vector of Fourier coefficients, $A(x)$, where $0 \leq k, x \leq N-1$

$$A(x) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a(k) e^{2\pi ikx/N}$$

The DFT operator is an $N \times N$ matrix with elements

$$F_{xk} = \tfrac{1}{\sqrt{N}} e^{2\pi ikx/N}$$

Assume $a(k) = e^{-2\pi iuk/N}$ is a function of frequency $u < N$ which evenly divides N

Computing the Fourier coefficients,

Discrete Fourier transform

The quantum Fourier transform is an important building block for many quantum algorithms

In order to develop the efficient implementation of the quantum Fourier transform, it is useful to start with the classical discrete and fast Fourier transforms

The discrete Fourier transform (DFT) is a linear transformation which takes a discrete column vector $a(k)$ to a column vector of Fourier coefficients, $A(x)$, where $0 \leq k, x \leq N-1$

$$A(x) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a(k) e^{2\pi ikx/N}$$

The DFT operator is an $N \times N$ matrix with elements

$$F_{xk} = \tfrac{1}{\sqrt{N}} e^{2\pi ikx/N}$$

Assume $a(k) = e^{-2\pi iuk/N}$ is a function of frequency $u < N$ which evenly divides N

Computing the Fourier coefficients,

$$A(x) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a(k) e^{2\pi ikx/N}$$

Discrete Fourier transform

The quantum Fourier transform is an important building block for many quantum algorithms

In order to develop the efficient implementation of the quantum Fourier transform, it is useful to start with the classical discrete and fast Fourier transforms

The discrete Fourier transform (DFT) is a linear transformation which takes a discrete column vector $a(k)$ to a column vector of Fourier coefficients, $A(x)$, where $0 \leq k, x \leq N-1$

$$A(x) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a(k) e^{2\pi ikx/N}$$

The DFT operator is an $N \times N$ matrix with elements

$$F_{xk} = \frac{1}{\sqrt{N}} e^{2\pi ikx/N}$$

Assume $a(k) = e^{-2\pi iuk/N}$ is a function of frequency $u < N$ which evenly divides N

Computing the Fourier coefficients,

$$A(x) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a(k) e^{2\pi ikx/N} = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{-2\pi iuk/N} e^{2\pi ikx/N}$$

Discrete Fourier transform

The quantum Fourier transform is an important building block for many quantum algorithms

In order to develop the efficient implementation of the quantum Fourier transform, it is useful to start with the classical discrete and fast Fourier transforms

The discrete Fourier transform (DFT) is a linear transformation which takes a discrete column vector $a(k)$ to a column vector of Fourier coefficients, $A(x)$, where $0 \leq k, x \leq N-1$

$$A(x) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a(k) e^{2\pi ikx/N}$$

The DFT operator is an $N \times N$ matrix with elements

$$F_{xk} = \frac{1}{\sqrt{N}} e^{2\pi ikx/N}$$

Assume $a(k) = e^{-2\pi iuk/N}$ is a function of frequency $u < N$ which evenly divides N

Computing the Fourier coefficients,

$$A(x) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a(k) e^{2\pi ikx/N} = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{-2\pi iuk/N} e^{2\pi ikx/N} = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi ik(x-u)/N}$$

Discrete Fourier transform

The quantum Fourier transform is an important building block for many quantum algorithms

In order to develop the efficient implementation of the quantum Fourier transform, it is useful to start with the classical discrete and fast Fourier transforms

The discrete Fourier transform (DFT) is a linear transformation which takes a discrete column vector $a(k)$ to a column vector of Fourier coefficients, $A(x)$, where $0 \leq k, x \leq N-1$

$$A(x) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a(k) e^{2\pi ikx/N}$$

The DFT operator is an $N \times N$ matrix with elements

$$F_{xk} = \frac{1}{\sqrt{N}} e^{2\pi ikx/N}$$

Assume $a(k) = e^{-2\pi iuk/N}$ is a function of frequency $u < N$ which evenly divides N

Computing the Fourier coefficients,

$$A(x) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a(k) e^{2\pi ikx/N} = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{-2\pi iuk/N} e^{2\pi ikx/N} = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi ik(x-u)/N}$$

All are zero except for when $x - u = 0 \mod N$ so the only term which survives is $A(u)$

DFT example

Start with the definition of the DFT

DFT example

Start with the definition of the DFT

$$A_x = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k e^{2\pi i k x / N}$$

DFT example

Start with the definition of the DFT and suppose that $a = \{1, 2, 3, 4\}$, $n = 2$, and $N = 4$

$$A_x = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k e^{2\pi ikx/N}$$

DFT example

$$A_x = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k e^{2\pi ikx/N}$$

Start with the definition of the DFT and suppose that $a = \{1, 2, 3, 4\}$, $n = 2$, and $N = 4$

$$A_0 = \tfrac{1}{2}\left(a_0 + a_1 + a_2 + a_3\right)$$

DFT example

Start with the definition of the DFT and suppose that $a = \{1, 2, 3, 4\}$, $n = 2$, and $N = 4$

$$A_x = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k e^{2\pi ikx/N}$$

$$A_0 = \frac{1}{2}\left(a_0 + a_1 + a_2 + a_3\right) = \frac{1}{2}(1 + 2 + 3 + 4) = 5$$

DFT example

$$A_x = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k e^{2\pi ikx/N}$$

Start with the definition of the DFT and suppose that $a = \{1, 2, 3, 4\}$, $n = 2$, and $N = 4$

$$A_0 = \tfrac{1}{2}\left(a_0 + a_1 + a_2 + a_3\right) = \tfrac{1}{2}(1 + 2 + 3 + 4) = 5$$

$$A_1 = \tfrac{1}{2}\left(a_0 + a_1 e^{i\pi/2} + a_2 e^{i\pi} + a_3 e^{i3\pi/2}\right)$$

DFT example

Start with the definition of the DFT and suppose that $a = \{1, 2, 3, 4\}$, $n = 2$, and $N = 4$

$$A_x = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k e^{2\pi ikx/N}$$

$$A_0 = \tfrac{1}{2}\left(a_0 + a_1 + a_2 + a_3\right) = \tfrac{1}{2}(1 + 2 + 3 + 4) = 5$$

$$A_1 = \tfrac{1}{2}\left(a_0 + a_1 e^{i\pi/2} + a_2 e^{i\pi} + a_3 e^{i3\pi/2}\right) = \tfrac{1}{2}(1 + 2i - 3 - 4i) = \tfrac{1}{2}(-2 - 2i) = -(1 + i)$$

DFT example

Start with the definition of the DFT and suppose that $a = \{1, 2, 3, 4\}$, $n = 2$, and $N = 4$

$$A_x = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k e^{2\pi i k x/N}$$

$$A_0 = \tfrac{1}{2}\left(a_0 + a_1 + a_2 + a_3\right) = \tfrac{1}{2}(1 + 2 + 3 + 4) = 5$$

$$A_1 = \tfrac{1}{2}\left(a_0 + a_1 e^{i\pi/2} + a_2 e^{i\pi} + a_3 e^{i3\pi/2}\right) = \tfrac{1}{2}(1 + 2i - 3 - 4i) = \tfrac{1}{2}(-2 - 2i) = -(1 + i)$$

$$A_2 = \tfrac{1}{2}\left(a_0 + a_1 e^{i\pi} + a_2 e^{i2\pi} + a_3 e^{i3\pi}\right)$$

DFT example

Start with the definition of the DFT and suppose that $a = \{1, 2, 3, 4\}$, $n = 2$, and $N = 4$

$$A_x = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k e^{2\pi ikx/N}$$

$$A_0 = \tfrac{1}{2}\left(a_0 + a_1 + a_2 + a_3\right) = \tfrac{1}{2}(1 + 2 + 3 + 4) = 5$$

$$A_1 = \tfrac{1}{2}\left(a_0 + a_1 e^{i\pi/2} + a_2 e^{i\pi} + a_3 e^{i3\pi/2}\right) = \tfrac{1}{2}(1 + 2i - 3 - 4i) = \tfrac{1}{2}(-2 - 2i) = -(1 + i)$$

$$A_2 = \tfrac{1}{2}\left(a_0 + a_1 e^{i\pi} + a_2 e^{i2\pi} + a_3 e^{i3\pi}\right) = (1 - 2 + 3 - 4) = -1$$

DFT example

Start with the definition of the DFT and suppose that $a = \{1, 2, 3, 4\}$, $n = 2$, and $N = 4$

$$A_x = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k e^{2\pi i k x/N}$$

$$A_0 = \tfrac{1}{2}\left(a_0 + a_1 + a_2 + a_3\right) = \tfrac{1}{2}(1 + 2 + 3 + 4) = 5$$

$$A_1 = \tfrac{1}{2}\left(a_0 + a_1 e^{i\pi/2} + a_2 e^{i\pi} + a_3 e^{i3\pi/2}\right) = \tfrac{1}{2}(1 + 2i - 3 - 4i) = \tfrac{1}{2}(-2 - 2i) = -(1 + i)$$

$$A_2 = \tfrac{1}{2}\left(a_0 + a_1 e^{i\pi} + a_2 e^{i2\pi} + a_3 e^{i3\pi}\right) = (1 - 2 + 3 - 4) = -1$$

$$A_3 = \tfrac{1}{2}\left(a_0 + a_1 e^{i3\pi/2} + a_2 e^{i3\pi} + a_3 e^{i9\pi/2}\right) = \tfrac{1}{2}(1 - 2i - 3 + 4i) = -(1 - i)$$

DFT example

Start with the definition of the DFT and suppose that $a = \{1, 2, 3, 4\}$, $n = 2$, and $N = 4$

$$A_x = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k e^{2\pi ikx/N}$$

$$A_0 = \tfrac{1}{2}\left(a_0 + a_1 + a_2 + a_3\right) = \tfrac{1}{2}(1 + 2 + 3 + 4) = 5$$

$$A_1 = \tfrac{1}{2}\left(a_0 + a_1 e^{i\pi/2} + a_2 e^{i\pi} + a_3 e^{i3\pi/2}\right) = \tfrac{1}{2}(1 + 2i - 3 - 4i) = \tfrac{1}{2}(-2 - 2i) = -(1 + i)$$

$$A_2 = \tfrac{1}{2}\left(a_0 + a_1 e^{i\pi} + a_2 e^{i2\pi} + a_3 e^{i3\pi}\right) = (1 - 2 + 3 - 4) = -1$$

$$A_3 = \tfrac{1}{2}\left(a_0 + a_1 e^{i3\pi/2} + a_2 e^{i3\pi} + a_3 e^{i9\pi/2}\right) = \tfrac{1}{2}(1 - 2i - 3 + 4i) = -(1 - i)$$

But if $u = 2$ and $a_k = e^{-2\pi iuk/N}$, we have

DFT example

Start with the definition of the DFT and suppose that $a = \{1, 2, 3, 4\}$, $n = 2$, and $N = 4$

$$A_x = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k e^{2\pi ikx/N}$$

$$A_0 = \tfrac{1}{2}\left(a_0 + a_1 + a_2 + a_3\right) = \tfrac{1}{2}(1+2+3+4) = 5$$

$$A_1 = \tfrac{1}{2}\left(a_0 + a_1 e^{i\pi/2} + a_2 e^{i\pi} + a_3 e^{i3\pi/2}\right) = \tfrac{1}{2}(1+2i-3-4i) = \tfrac{1}{2}(-2-2i) = -(1+i)$$

$$A_2 = \tfrac{1}{2}\left(a_0 + a_1 e^{i\pi} + a_2 e^{i2\pi} + a_3 e^{i3\pi}\right) = (1-2+3-4) = -1$$

$$A_3 = \tfrac{1}{2}\left(a_0 + a_1 e^{i3\pi/2} + a_2 e^{i3\pi} + a_3 e^{i9\pi/2}\right) = \tfrac{1}{2}(1-2i-3+4i) = -(1-i)$$

But if $u = 2$ and $a_k = e^{-2\pi iuk/N}$, we have

$$A_0 = \tfrac{1}{2}\left(1 + e^{-i\pi} + e^{-i2\pi} + e^{-i3\pi}\right)$$

DFT example

Start with the definition of the DFT and suppose that $a = \{1, 2, 3, 4\}$, $n = 2$, and $N = 4$

$$A_x = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k e^{2\pi i k x/N}$$

$$A_0 = \tfrac{1}{2}\left(a_0 + a_1 + a_2 + a_3\right) = \tfrac{1}{2}(1+2+3+4) = 5$$

$$A_1 = \tfrac{1}{2}\left(a_0 + a_1 e^{i\pi/2} + a_2 e^{i\pi} + a_3 e^{i3\pi/2}\right) = \tfrac{1}{2}(1+2i-3-4i) = \tfrac{1}{2}(-2-2i) = -(1+i)$$

$$A_2 = \tfrac{1}{2}\left(a_0 + a_1 e^{i\pi} + a_2 e^{i2\pi} + a_3 e^{i3\pi}\right) = (1-2+3-4) = -1$$

$$A_3 = \tfrac{1}{2}\left(a_0 + a_1 e^{i3\pi/2} + a_2 e^{i3\pi} + a_3 e^{i9\pi/2}\right) = \tfrac{1}{2}(1-2i-3+4i) = -(1-i)$$

But if $u = 2$ and $a_k = e^{-2\pi i u k/N}$, we have

$$A_0 = \tfrac{1}{2}\left(1 + e^{-i\pi} + e^{-i2\pi} + e^{-i3\pi}\right) = \tfrac{1}{2}(1-1+1-1) = 0$$

DFT example

Start with the definition of the DFT and suppose that $a = \{1, 2, 3, 4\}$, $n = 2$, and $N = 4$

$$A_x = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k e^{2\pi ikx/N}$$

$$A_0 = \tfrac{1}{2}\left(a_0 + a_1 + a_2 + a_3\right) = \tfrac{1}{2}(1 + 2 + 3 + 4) = 5$$

$$A_1 = \tfrac{1}{2}\left(a_0 + a_1 e^{i\pi/2} + a_2 e^{i\pi} + a_3 e^{i3\pi/2}\right) = \tfrac{1}{2}(1 + 2i - 3 - 4i) = \tfrac{1}{2}(-2 - 2i) = -(1 + i)$$

$$A_2 = \tfrac{1}{2}\left(a_0 + a_1 e^{i\pi} + a_2 e^{i2\pi} + a_3 e^{i3\pi}\right) = (1 - 2 + 3 - 4) = -1$$

$$A_3 = \tfrac{1}{2}\left(a_0 + a_1 e^{i3\pi/2} + a_2 e^{i3\pi} + a_3 e^{i9\pi/2}\right) = \tfrac{1}{2}(1 - 2i - 3 + 4i) = -(1 - i)$$

But if $u = 2$ and $a_k = e^{-2\pi iuk/N}$, we have

$$A_0 = \tfrac{1}{2}\left(1 + e^{-i\pi} + e^{-i2\pi} + e^{-i3\pi}\right) = \tfrac{1}{2}(1 - 1 + 1 - 1) = 0$$

$$A_1 = \tfrac{1}{2}\left(1 + e^{-i\pi} e^{i\pi/2} + e^{-i2\pi} e^{i\pi} + e^{-i3\pi} e^{i3\pi/2}\right)$$

DFT example

Start with the definition of the DFT and suppose that $a = \{1, 2, 3, 4\}$, $n = 2$, and $N = 4$

$$A_x = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k e^{2\pi i k x/N}$$

$$A_0 = \tfrac{1}{2}\left(a_0 + a_1 + a_2 + a_3\right) = \tfrac{1}{2}(1 + 2 + 3 + 4) = 5$$

$$A_1 = \tfrac{1}{2}\left(a_0 + a_1 e^{i\pi/2} + a_2 e^{i\pi} + a_3 e^{i3\pi/2}\right) = \tfrac{1}{2}(1 + 2i - 3 - 4i) = \tfrac{1}{2}(-2 - 2i) = -(1 + i)$$

$$A_2 = \tfrac{1}{2}\left(a_0 + a_1 e^{i\pi} + a_2 e^{i2\pi} + a_3 e^{i3\pi}\right) = (1 - 2 + 3 - 4) = -1$$

$$A_3 = \tfrac{1}{2}\left(a_0 + a_1 e^{i3\pi/2} + a_2 e^{i3\pi} + a_3 e^{i9\pi/2}\right) = \tfrac{1}{2}(1 - 2i - 3 + 4i) = -(1 - i)$$

But if $u = 2$ and $a_k = e^{-2\pi i u k/N}$, we have

$$A_0 = \tfrac{1}{2}\left(1 + e^{-i\pi} + e^{-i2\pi} + e^{-i3\pi}\right) = \tfrac{1}{2}(1 - 1 + 1 - 1) = 0$$

$$A_1 = \tfrac{1}{2}\left(1 + e^{-i\pi} e^{i\pi/2} + e^{-i2\pi} e^{i\pi} + e^{-i3\pi} e^{i3\pi/2}\right) = \tfrac{1}{2}(1 - i - 1 + i) = 0$$

DFT example

Start with the definition of the DFT and suppose that $a = \{1, 2, 3, 4\}$, $n = 2$, and $N = 4$

$$A_x = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k e^{2\pi ikx/N}$$

$$A_0 = \tfrac{1}{2}\left(a_0 + a_1 + a_2 + a_3\right) = \tfrac{1}{2}(1+2+3+4) = 5$$

$$A_1 = \tfrac{1}{2}\left(a_0 + a_1 e^{i\pi/2} + a_2 e^{i\pi} + a_3 e^{i3\pi/2}\right) = \tfrac{1}{2}(1+2i-3-4i) = \tfrac{1}{2}(-2-2i) = -(1+i)$$

$$A_2 = \tfrac{1}{2}\left(a_0 + a_1 e^{i\pi} + a_2 e^{i2\pi} + a_3 e^{i3\pi}\right) = (1-2+3-4) = -1$$

$$A_3 = \tfrac{1}{2}\left(a_0 + a_1 e^{i3\pi/2} + a_2 e^{i3\pi} + a_3 e^{i9\pi/2}\right) = \tfrac{1}{2}(1-2i-3+4i) = -(1-i)$$

But if $u = 2$ and $a_k = e^{-2\pi iuk/N}$, we have

$$A_0 = \tfrac{1}{2}\left(1 + e^{-i\pi} + e^{-i2\pi} + e^{-i3\pi}\right) = \tfrac{1}{2}(1-1+1-1) = 0$$

$$A_1 = \tfrac{1}{2}\left(1 + e^{-i\pi} e^{i\pi/2} + e^{-i2\pi} e^{i\pi} + e^{-i3\pi} e^{i3\pi/2}\right) = \tfrac{1}{2}(1-i-1+i) = 0$$

DFT example

Start with the definition of the DFT and suppose that $a = \{1, 2, 3, 4\}$, $n = 2$, and $N = 4$

$$A_x = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k e^{2\pi i k x/N}$$

$$A_0 = \tfrac{1}{2}\left(a_0 + a_1 + a_2 + a_3\right) = \tfrac{1}{2}(1+2+3+4) = 5$$

$$A_1 = \tfrac{1}{2}\left(a_0 + a_1 e^{i\pi/2} + a_2 e^{i\pi} + a_3 e^{i3\pi/2}\right) = \tfrac{1}{2}(1+2i-3-4i) = \tfrac{1}{2}(-2-2i) = -(1+i)$$

$$A_2 = \tfrac{1}{2}\left(a_0 + a_1 e^{i\pi} + a_2 e^{i2\pi} + a_3 e^{i3\pi}\right) = (1-2+3-4) = -1$$

$$A_3 = \tfrac{1}{2}\left(a_0 + a_1 e^{i3\pi/2} + a_2 e^{i3\pi} + a_3 e^{i9\pi/2}\right) = \tfrac{1}{2}(1-2i-3+4i) = -(1-i)$$

But if $u = 2$ and $a_k = e^{-2\pi i u k/N}$, we have

$$A_0 = \tfrac{1}{2}\left(1 + e^{-i\pi} + e^{-i2\pi} + e^{-i3\pi}\right) = \tfrac{1}{2}(1-1+1-1) = 0$$

$$A_1 = \tfrac{1}{2}\left(1 + e^{-i\pi}e^{i\pi/2} + e^{-i2\pi}e^{i\pi} + e^{-i3\pi}e^{i3\pi/2}\right) = \tfrac{1}{2}(1-i-1+i) = 0$$

$$A_2 = \tfrac{1}{2}\left(1 + e^{-i\pi}e^{i\pi} + e^{-i2\pi}e^{i2\pi} + e^{-i3\pi}e^{i3\pi}\right)$$

DFT example

Start with the definition of the DFT and suppose that $a = \{1, 2, 3, 4\}$, $n = 2$, and $N = 4$

$$A_x = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k e^{2\pi ikx/N}$$

$$A_0 = \tfrac{1}{2}\left(a_0 + a_1 + a_2 + a_3\right) = \tfrac{1}{2}(1+2+3+4) = 5$$

$$A_1 = \tfrac{1}{2}\left(a_0 + a_1 e^{i\pi/2} + a_2 e^{i\pi} + a_3 e^{i3\pi/2}\right) = \tfrac{1}{2}(1+2i-3-4i) = \tfrac{1}{2}(-2-2i) = -(1+i)$$

$$A_2 = \tfrac{1}{2}\left(a_0 + a_1 e^{i\pi} + a_2 e^{i2\pi} + a_3 e^{i3\pi}\right) = (1-2+3-4) = -1$$

$$A_3 = \tfrac{1}{2}\left(a_0 + a_1 e^{i3\pi/2} + a_2 e^{i3\pi} + a_3 e^{i9\pi/2}\right) = \tfrac{1}{2}(1-2i-3+4i) = -(1-i)$$

But if $u = 2$ and $a_k = e^{-2\pi iuk/N}$, we have

$$A_0 = \tfrac{1}{2}\left(1 + e^{-i\pi} + e^{-i2\pi} + e^{-i3\pi}\right) = \tfrac{1}{2}(1-1+1-1) = 0$$

$$A_1 = \tfrac{1}{2}\left(1 + e^{-i\pi}e^{i\pi/2} + e^{-i2\pi}e^{i\pi} + e^{-i3\pi}e^{i3\pi/2}\right) = \tfrac{1}{2}(1-i-1+i) = 0$$

$$A_2 = \tfrac{1}{2}\left(1 + e^{-i\pi}e^{i\pi} + e^{-i2\pi}e^{i2\pi} + e^{-i3\pi}e^{i3\pi}\right) = \tfrac{1}{2}(1+1+1+1+1) = 2$$

DFT example

Start with the definition of the DFT and suppose that $a = \{1, 2, 3, 4\}$, $n = 2$, and $N = 4$

$$A_x = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k e^{2\pi i k x/N}$$

$$A_0 = \tfrac{1}{2}\left(a_0 + a_1 + a_2 + a_3\right) = \tfrac{1}{2}(1+2+3+4) = 5$$

$$A_1 = \tfrac{1}{2}\left(a_0 + a_1 e^{i\pi/2} + a_2 e^{i\pi} + a_3 e^{i3\pi/2}\right) = \tfrac{1}{2}(1+2i-3-4i) = \tfrac{1}{2}(-2-2i) = -(1+i)$$

$$A_2 = \tfrac{1}{2}\left(a_0 + a_1 e^{i\pi} + a_2 e^{i2\pi} + a_3 e^{i3\pi}\right) = (1-2+3-4) = -1$$

$$A_3 = \tfrac{1}{2}\left(a_0 + a_1 e^{i3\pi/2} + a_2 e^{i3\pi} + a_3 e^{i9\pi/2}\right) = \tfrac{1}{2}(1-2i-3+4i) = -(1-i)$$

But if $u = 2$ and $a_k = e^{-2\pi i u k/N}$, we have

$$A_0 = \tfrac{1}{2}\left(1 + e^{-i\pi} + e^{-i2\pi} + e^{-i3\pi}\right) = \tfrac{1}{2}(1-1+1-1) = 0$$

$$A_1 = \tfrac{1}{2}\left(1 + e^{-i\pi}e^{i\pi/2} + e^{-i2\pi}e^{i\pi} + e^{-i3\pi}e^{i3\pi/2}\right) = \tfrac{1}{2}(1-i-1+i) = 0$$

$$A_2 = \tfrac{1}{2}\left(1 + e^{-i\pi}e^{i\pi} + e^{-i2\pi}e^{i2\pi} + e^{-i3\pi}e^{i3\pi}\right) = \tfrac{1}{2}(1+1+1+1+1) = 2$$

$$A_3 = \tfrac{1}{2}\left(1 + e^{-i\pi}e^{i3\pi/2} + e^{-i2\pi}e^{i3\pi} + e^{-i3\pi}e^{i9\pi/2}\right)$$

DFT example

Start with the definition of the DFT and suppose that $a = \{1, 2, 3, 4\}$, $n = 2$, and $N = 4$

$$A_x = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k e^{2\pi i k x/N}$$

$$A_0 = \tfrac{1}{2}\left(a_0 + a_1 + a_2 + a_3\right) = \tfrac{1}{2}(1+2+3+4) = 5$$

$$A_1 = \tfrac{1}{2}\left(a_0 + a_1 e^{i\pi/2} + a_2 e^{i\pi} + a_3 e^{i3\pi/2}\right) = \tfrac{1}{2}(1+2i-3-4i) = \tfrac{1}{2}(-2-2i) = -(1+i)$$

$$A_2 = \tfrac{1}{2}\left(a_0 + a_1 e^{i\pi} + a_2 e^{i2\pi} + a_3 e^{i3\pi}\right) = (1-2+3-4) = -1$$

$$A_3 = \tfrac{1}{2}\left(a_0 + a_1 e^{i3\pi/2} + a_2 e^{i3\pi} + a_3 e^{i9\pi/2}\right) = \tfrac{1}{2}(1-2i-3+4i) = -(1-i)$$

But if $u = 2$ and $a_k = e^{-2\pi i u k/N}$, we have

$$A_0 = \tfrac{1}{2}\left(1 + e^{-i\pi} + e^{-i2\pi} + e^{-i3\pi}\right) = \tfrac{1}{2}(1-1+1-1) = 0$$

$$A_1 = \tfrac{1}{2}\left(1 + e^{-i\pi}e^{i\pi/2} + e^{-i2\pi}e^{i\pi} + e^{-i3\pi}e^{i3\pi/2}\right) = \tfrac{1}{2}(1-i-1+i) = 0$$

$$A_2 = \tfrac{1}{2}\left(1 + e^{-i\pi}e^{i\pi} + e^{-i2\pi}e^{i2\pi} + e^{-i3\pi}e^{i3\pi}\right) = \tfrac{1}{2}(1+1+1+1+1) = 2$$

$$A_3 = \tfrac{1}{2}\left(1 + e^{-i\pi}e^{i3\pi/2} + e^{-i2\pi}e^{i3\pi} + e^{-i3\pi}e^{i9\pi/2}\right) = \tfrac{1}{2}(1-i-1+i) = 0$$

Fast Fourier transform

The fast Fourier transform (FFT) is an efficient implementation for $N = 2^n$, called $F^{(n)}$

Fast Fourier transform

The fast Fourier transform (FFT) is an efficient implementation for $N = 2^n$, called $F^{(n)}$

The implementation involves recursive decomposition of $F^{(n)}$ in terms of Fourier transforms of lower powers of 2

Fast Fourier transform

The fast Fourier transform (FFT) is an efficient implementation for $N = 2^n$, called $F^{(n)}$

The implementation involves recursive decomposition of $F^{(n)}$ in terms of Fourier transforms of lower powers of 2

If $\omega_{(n)} = e^{2\pi i/N}$ is the Nth root of unity, the elements of the Fourier transform matrix are

Fast Fourier transform

The fast Fourier transform (FFT) is an efficient implementation for $N = 2^n$, called $F^{(n)}$

The implementation involves recursive decomposition of $F^{(n)}$ in terms of Fourier transforms of lower powers of 2

If $\omega_{(n)} = e^{2\pi i/N}$ is the Nth root of unity, the elements of the Fourier transform matrix are

$$F^{(n)}_{xy} = \omega^{xy}_{(n)}$$

Fast Fourier transform

The fast Fourier transform (FFT) is an efficient implementation for $N = 2^n$, called $F^{(n)}$

The implementation involves recursive decomposition of $F^{(n)}$ in terms of Fourier transforms of lower powers of 2

If $\omega_{(n)} = e^{2\pi i/N}$ is the Nth root of unity, the elements of the Fourier transform matrix are

$$F^{(n)}_{xy} = \omega^{xy}_{(n)}$$

where $x, y \in \{0, \ldots, N-1\}$

Fast Fourier transform

The fast Fourier transform (FFT) is an efficient implementation for $N = 2^n$, called $F^{(n)}$

The implementation involves recursive decomposition of $F^{(n)}$ in terms of Fourier transforms of lower powers of 2

If $\omega_{(n)} = e^{2\pi i/N}$ is the Nth root of unity, the elements of the Fourier transform matrix are

$$F^{(n)}_{xy} = \omega^{xy}_{(n)}$$

where $x, y \in \{0, \ldots, N-1\}$

Let $F^{(k)}$, $I^{(k)}$, $R^{(k)}$ be the 2^k-dimensional Fourier transform matrix, identity matrix, and a permutation matrix defined by

Fast Fourier transform

The fast Fourier transform (FFT) is an efficient implementation for $N = 2^n$, called $F^{(n)}$

The implementation involves recursive decomposition of $F^{(n)}$ in terms of Fourier transforms of lower powers of 2

If $\omega_{(n)} = e^{2\pi i/N}$ is the Nth root of unity, the elements of the Fourier transform matrix are

$$F^{(n)}_{xy} = \omega^{xy}_{(n)}$$

where $x, y \in \{0, \ldots, N-1\}$

Let $F^{(k)}$, $I^{(k)}$, $R^{(k)}$ be the 2^k-dimensional Fourier transform matrix, identity matrix, and a permutation matrix defined by

$$R^{(k)}_{xy} = \begin{cases} 1 & \text{for } 2x = y \\ 1 & \text{for } 2x - 2^k + 1 = y \\ 0 & \text{otherwise} \end{cases}$$

Fast Fourier transform

The fast Fourier transform (FFT) is an efficient implementation for $N = 2^n$, called $F^{(n)}$

The implementation involves recursive decomposition of $F^{(n)}$ in terms of Fourier transforms of lower powers of 2

If $\omega_{(n)} = e^{2\pi i/N}$ is the Nth root of unity, the elements of the Fourier transform matrix are

$$F^{(n)}_{xy} = \omega^{xy}_{(n)}$$

where $x, y \in \{0, \dots, N-1\}$

Let $F^{(k)}$, $I^{(k)}$, $R^{(k)}$ be the 2^k-dimensional Fourier transform matrix, identity matrix, and a permutation matrix defined by

$$R^{(k)}_{xy} = \begin{cases} 1 & \text{for } 2x = y \\ 1 & \text{for } 2x - 2^k + 1 = y \\ 0 & \text{otherwise} \end{cases}$$

$$R^{(3)} = \left(\quad \right)$$

Fast Fourier transform

The fast Fourier transform (FFT) is an efficient implementation for $N = 2^n$, called $F^{(n)}$

The implementation involves recursive decomposition of $F^{(n)}$ in terms of Fourier transforms of lower powers of 2

If $\omega_{(n)} = e^{2\pi i/N}$ is the Nth root of unity, the elements of the Fourier transform matrix are

$$F^{(n)}_{xy} = \omega^{xy}_{(n)}$$

where $x, y \in \{0, \ldots, N-1\}$

Let $F^{(k)}$, $I^{(k)}$, $R^{(k)}$ be the 2^k-dimensional Fourier transform matrix, identity matrix, and a permutation matrix defined by

$$R^{(k)}_{xy} = \begin{cases} 1 & \text{for } 2x = y \\ 1 & \text{for } 2x - 2^k + 1 = y \\ 0 & \text{otherwise} \end{cases}$$

$$i = 0; \quad y = 2x = 0$$

$$R^{(3)} = \begin{pmatrix} 1 & & & & & & & \\ & & & & & & & \\ & & & & & & & \\ & & & & & & & \\ & & & & & & & \\ & & & & & & & \\ & & & & & & & \\ & & & & & & & \end{pmatrix}$$

Fast Fourier transform

The fast Fourier transform (FFT) is an efficient implementation for $N = 2^n$, called $F^{(n)}$

The implementation involves recursive decomposition of $F^{(n)}$ in terms of Fourier transforms of lower powers of 2

If $\omega_{(n)} = e^{2\pi i/N}$ is the Nth root of unity, the elements of the Fourier transform matrix are

$$F^{(n)}_{xy} = \omega^{xy}_{(n)}$$

where $x, y \in \{0, \ldots, N-1\}$

Let $F^{(k)}$, $I^{(k)}$, $R^{(k)}$ be the 2^k-dimensional Fourier transform matrix, identity matrix, and a permutation matrix defined by

$$R^{(k)}_{xy} = \begin{cases} 1 & \text{for } 2x = y \\ 1 & \text{for } 2x - 2^k + 1 = y \\ 0 & \text{otherwise} \end{cases}$$

$$x = 1; \quad y = 2x = 2$$

$$R^{(3)} = \begin{pmatrix} 1 & & & & & & & \\ & & 1 & & & & & \\ & & & & & & & \\ & & & & & & & \\ & & & & & & & \\ & & & & & & & \\ & & & & & & & \\ & & & & & & & \end{pmatrix}$$

Fast Fourier transform

The fast Fourier transform (FFT) is an efficient implementation for $N = 2^n$, called $F^{(n)}$

The implementation involves recursive decomposition of $F^{(n)}$ in terms of Fourier transforms of lower powers of 2

If $\omega_{(n)} = e^{2\pi i/N}$ is the Nth root of unity, the elements of the Fourier transform matrix are

$$F^{(n)}_{xy} = \omega^{xy}_{(n)}$$

where $x, y \in \{0, \ldots, N-1\}$

Let $F^{(k)}$, $I^{(k)}$, $R^{(k)}$ be the 2^k-dimensional Fourier transform matrix, identity matrix, and a permutation matrix defined by

$$R^{(k)}_{xy} = \begin{cases} 1 & \text{for } 2x = y \\ 1 & \text{for } 2x - 2^k + 1 = y \\ 0 & \text{otherwise} \end{cases}$$

$$x = 2; \quad y = 2x = 4$$

$$R^{(3)} = \begin{pmatrix} 1 & & & & & & & \\ & & 1 & & & & & \\ & & & & 1 & & & \\ & & & & & & & \\ & & & & & & & \\ & & & & & & & \\ & & & & & & & \\ & & & & & & & \end{pmatrix}$$

Fast Fourier transform

The fast Fourier transform (FFT) is an efficient implementation for $N = 2^n$, called $F^{(n)}$

The implementation involves recursive decomposition of $F^{(n)}$ in terms of Fourier transforms of lower powers of 2

If $\omega_{(n)} = e^{2\pi i/N}$ is the Nth root of unity, the elements of the Fourier transform matrix are

$$F^{(n)}_{xy} = \omega^{xy}_{(n)}$$

where $x, y \in \{0, \ldots, N-1\}$

Let $F^{(k)}$, $I^{(k)}$, $R^{(k)}$ be the 2^k-dimensional Fourier transform matrix, identity matrix, and a permutation matrix defined by

$$R^{(k)}_{xy} = \begin{cases} 1 & \text{for } 2x = y \\ 1 & \text{for } 2x - 2^k + 1 = y \\ 0 & \text{otherwise} \end{cases}$$

$x = 3; \quad y = 2x = 6$

$$R^{(3)} = \begin{pmatrix} 1 & & & & & & & \\ & & 1 & & & & & \\ & & & & 1 & & & \\ & & & & & & 1 & \\ & & & & & & & \\ & & & & & & & \\ & & & & & & & \\ & & & & & & & \end{pmatrix}$$

Fast Fourier transform

The fast Fourier transform (FFT) is an efficient implementation for $N = 2^n$, called $F^{(n)}$

The implementation involves recursive decomposition of $F^{(n)}$ in terms of Fourier transforms of lower powers of 2

If $\omega_{(n)} = e^{2\pi i/N}$ is the Nth root of unity, the elements of the Fourier transform matrix are

$$F^{(n)}_{xy} = \omega^{xy}_{(n)}$$

where $x, y \in \{0, \ldots, N-1\}$

Let $F^{(k)}$, $I^{(k)}$, $R^{(k)}$ be the 2^k-dimensional Fourier transform matrix, identity matrix, and a permutation matrix defined by

$$R^{(k)}_{xy} = \begin{cases} 1 & \text{for } 2x = y \\ 1 & \text{for } 2x - 2^k + 1 = y \\ 0 & \text{otherwise} \end{cases}$$

$$x = 4; \quad y = 2x - 8 + 1 = 1$$

$$R^{(3)} = \begin{pmatrix} 1 & & & & & & & \\ & & 1 & & & & & \\ & & & & 1 & & & \\ & & & & & & 1 & \\ & 1 & & & & & & \\ & & & & & & & \\ & & & & & & & \\ & & & & & & & \end{pmatrix}$$

Fast Fourier transform

The fast Fourier transform (FFT) is an efficient implementation for $N = 2^n$, called $F^{(n)}$

The implementation involves recursive decomposition of $F^{(n)}$ in terms of Fourier transforms of lower powers of 2

If $\omega_{(n)} = e^{2\pi i/N}$ is the Nth root of unity, the elements of the Fourier transform matrix are

$$F^{(n)}_{xy} = \omega^{xy}_{(n)}$$

where $x, y \in \{0, \ldots, N-1\}$

Let $F^{(k)}$, $I^{(k)}$, $R^{(k)}$ be the 2^k-dimensional Fourier transform matrix, identity matrix, and a permutation matrix defined by

$$R^{(k)}_{xy} = \begin{cases} 1 & \text{for } 2x = y \\ 1 & \text{for } 2x - 2^k + 1 = y \\ 0 & \text{otherwise} \end{cases}$$

$$x = 5; \quad y = 2x - 8 + 1 = 3$$

$$R^{(3)} = \begin{pmatrix} 1 & & & & & & & \\ & & 1 & & & & & \\ & & & & 1 & & & \\ & & & & & & 1 & \\ & 1 & & & & & & \\ & & & 1 & & & & \\ & & & & & & & \\ & & & & & & & \end{pmatrix}$$

Fast Fourier transform

The fast Fourier transform (FFT) is an efficient implementation for $N = 2^n$, called $F^{(n)}$

The implementation involves recursive decomposition of $F^{(n)}$ in terms of Fourier transforms of lower powers of 2

If $\omega_{(n)} = e^{2\pi i/N}$ is the Nth root of unity, the elements of the Fourier transform matrix are

$$F^{(n)}_{xy} = \omega^{xy}_{(n)}$$

where $x, y \in \{0, \dots, N-1\}$

Let $F^{(k)}$, $I^{(k)}$, $R^{(k)}$ be the 2^k-dimensional Fourier transform matrix, identity matrix, and a permutation matrix defined by

$$R^{(k)}_{xy} = \begin{cases} 1 & \text{for } 2x = y \\ 1 & \text{for } 2x - 2^k + 1 = y \\ 0 & \text{otherwise} \end{cases}$$

$$x = 6; \quad y = 2x - 8 + 1 = 5$$

$$R^{(3)} = \begin{pmatrix} 1 & & & & & & & \\ & & 1 & & & & & \\ & & & & 1 & & & \\ & & & & & & 1 & \\ & 1 & & & & & & \\ & & & 1 & & & & \\ & & & & & 1 & & \\ & & & & & & & \end{pmatrix}$$

Fast Fourier transform

The fast Fourier transform (FFT) is an efficient implementation for $N = 2^n$, called $F^{(n)}$

The implementation involves recursive decomposition of $F^{(n)}$ in terms of Fourier transforms of lower powers of 2

If $\omega_{(n)} = e^{2\pi i/N}$ is the Nth root of unity, the elements of the Fourier transform matrix are

$$F^{(n)}_{xy} = \omega^{xy}_{(n)}$$

where $x, y \in \{0, \ldots, N-1\}$

Let $F^{(k)}$, $I^{(k)}$, $R^{(k)}$ be the 2^k-dimensional Fourier transform matrix, identity matrix, and a permutation matrix defined by

$$R^{(k)}_{xy} = \begin{cases} 1 & \text{for } 2x = y \\ 1 & \text{for } 2x - 2^k + 1 = y \\ 0 & \text{otherwise} \end{cases}$$

$$x = 7; \quad y = 2x - 8 + 1 = 7$$

$$R^{(3)} = \begin{pmatrix} 1 & & & & & & & \\ & & 1 & & & & & \\ & & & & 1 & & & \\ & & & & & & 1 & \\ & 1 & & & & & & \\ & & & 1 & & & & \\ & & & & & 1 & & \\ & & & & & & & 1 \end{pmatrix}$$

Fast Fourier transform

The fast Fourier transform (FFT) is an efficient implementation for $N = 2^n$, called $F^{(n)}$

The implementation involves recursive decomposition of $F^{(n)}$ in terms of Fourier transforms of lower powers of 2

If $\omega_{(n)} = e^{2\pi i/N}$ is the Nth root of unity, the elements of the Fourier transform matrix are

$$F^{(n)}_{xy} = \omega^{xy}_{(n)}$$

where $x, y \in \{0, \ldots, N-1\}$

Let $F^{(k)}$, $I^{(k)}$, $R^{(k)}$ be the 2^k-dimensional Fourier transform matrix, identity matrix, and a permutation matrix defined by

$$R^{(k)}_{xy} = \begin{cases} 1 & \text{for } 2x = y \\ 1 & \text{for } 2x - 2^k + 1 = y \\ 0 & \text{otherwise} \end{cases}$$

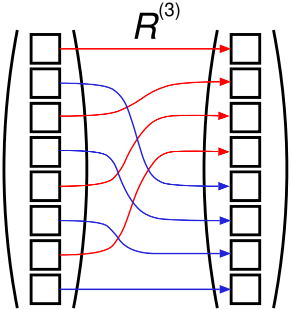
$$R^{(3)} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Fast Fourier transform

The $R^{(k)}$ matrix performs a shuffle transform on a column vector

Fast Fourier transform

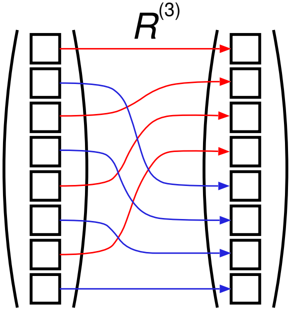
The $R^{(k)}$ matrix performs a shuffle transform on a column vector



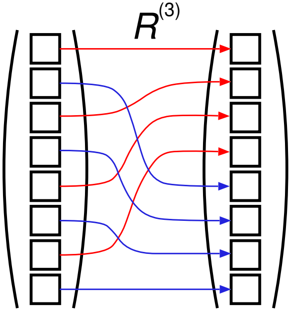
Fast Fourier transform

The $R^{(k)}$ matrix performs a shuffle transform on a column vector

Using $F^{(k)}$, $I^{(k)}$, $R^{(k)}$ plus $D^{(k)}$, a diagonal matrix with entries $\omega^0_{(k+1)}, \ldots, \omega^{2^k-1}_{(k+1)}$ it is possible to solve for $F^{(k)}$ recursively



Fast Fourier transform



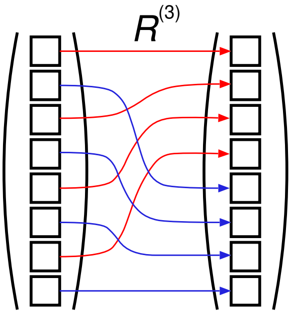
The $R^{(k)}$ matrix performs a shuffle transform on a column vector

Using $F^{(k)}$, $I^{(k)}$, $R^{(k)}$ plus $D^{(k)}$, a diagonal matrix with entries $\omega^{0}_{(k+1)}, \ldots, \omega^{2^k-1}_{(k+1)}$ it is possible to solve for $F^{(k)}$ recursively

$$F^{(k)} = \frac{1}{\sqrt{2}} \begin{pmatrix} I^{(k-1)} & D^{(k-1)} \\ I^{(k-1)} & -D^{(k-1)} \end{pmatrix} \begin{pmatrix} F^{(k-1)} & 0 \\ 0 & F^{(k-1)} \end{pmatrix} R^{(k)}$$

Fast Fourier transform

The $R^{(k)}$ matrix performs a shuffle transform on a column vector

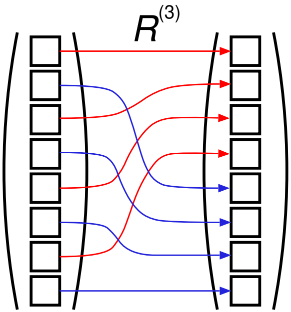


Using $F^{(k)}$, $I^{(k)}$, $R^{(k)}$ plus $D^{(k)}$, a diagonal matrix with entries $\omega^{0}_{(k+1)}, \ldots, \omega^{2^k-1}_{(k+1)}$ it is possible to solve for $F^{(k)}$ recursively

$$F^{(k)} = \frac{1}{\sqrt{2}} \begin{pmatrix} I^{(k-1)} & D^{(k-1)} \\ I^{(k-1)} & -D^{(k-1)} \end{pmatrix} \begin{pmatrix} F^{(k-1)} & 0 \\ 0 & F^{(k-1)} \end{pmatrix} R^{(k)}$$

The $R^{(k)}$ operator serves to reorder the data vector into odd and even elements and multiplication by block diagonal matrices is very efficient

Fast Fourier transform



The $R^{(k)}$ matrix performs a shuffle transform on a column vector

Using $F^{(k)}$, $I^{(k)}$, $R^{(k)}$ plus $D^{(k)}$, a diagonal matrix with entries $\omega^0_{(k+1)}, \dots, \omega^{2^k-1}_{(k+1)}$ it is possible to solve for $F^{(k)}$ recursively

$$F^{(k)} = \frac{1}{\sqrt{2}} \begin{pmatrix} I^{(k-1)} & D^{(k-1)} \\ I^{(k-1)} & -D^{(k-1)} \end{pmatrix} \begin{pmatrix} F^{(k-1)} & 0 \\ 0 & F^{(k-1)} \end{pmatrix} R^{(k)}$$

The $R^{(k)}$ operator serves to reorder the data vector into odd and even elements and multiplication by block diagonal matrices is very efficient

Computing $F^{(k)}$ becomes computing 2 $F^{(k-1)}$, then 4 $F^{(k-2)}$, and so on until 2^{k-1} $F^{(1)}$ matrices and $O(nN)$ computation

Quantum Fourier transform

The quantum Fourier transform (QFT) like the FFT assumes $N = 2^n$ and that the amplitudes $a_x \equiv a(x)$ of the superposition state $|\psi\rangle$ are the function to be transformed

Quantum Fourier transform

The quantum Fourier transform (QFT) like the FFT assumes $N = 2^n$ and that the amplitudes $a_x \equiv a(x)$ of the superposition state $|\psi\rangle$ are the function to be transformed

$$|\psi\rangle = \sum_{x=0}^{N-1} a(x)|x\rangle \longrightarrow \sum_{x=0}^{N-1} A(x)|x\rangle$$

Quantum Fourier transform

The quantum Fourier transform (QFT) like the FFT assumes $N = 2^n$ and that the amplitudes $a_x \equiv a(x)$ of the superposition state $|\psi\rangle$ are the function to be transformed

$$|\psi\rangle = \sum_{x=0}^{N-1} a(x)|x\rangle \longrightarrow \sum_{x=0}^{N-1} A(x)|x\rangle$$

The QFT does not require an output register as the output quantum state contains the Fourier transform in its complex amplitudes

Quantum Fourier transform

The quantum Fourier transform (QFT) like the FFT assumes $N = 2^n$ and that the amplitudes $a_x \equiv a(x)$ of the superposition state $|\psi\rangle$ are the function to be transformed

$$|\psi\rangle = \sum_{x=0}^{N-1} a(x)|x\rangle \longrightarrow \sum_{x=0}^{N-1} A(x)|x\rangle$$

The QFT does not require an output register as the output quantum state contains the Fourier transform in its complex amplitudes

If the initial state is such that the amplitudes are a periodic function with period $r = 2^m$, the resultant Ax would be zero unless $x = j\frac{N}{r}$ with $j = 0, 1, \ldots, \frac{N}{r} - 1$

Quantum Fourier transform

The quantum Fourier transform (QFT) like the FFT assumes $N = 2^n$ and that the amplitudes $a_x \equiv a(x)$ of the superposition state $|\psi\rangle$ are the function to be transformed

$$|\psi\rangle = \sum_{x=0}^{N-1} a(x)|x\rangle \longrightarrow \sum_{x=0}^{N-1} A(x)|x\rangle$$

The QFT does not require an output register as the output quantum state contains the Fourier transform in its complex amplitudes

If the initial state is such that the amplitudes are a periodic function with period $r = 2^m$, the resultant Ax would be zero unless $x = j\frac{N}{r}$ with $j = 0, 1, \dots, \frac{N}{r} - 1$

When $r \neq 2^m$, the QFT will produce an approximate solution with higher probability coefficients for states with integers near multiples of $\frac{N}{r}$

Quantum Fourier transform

The quantum Fourier transform (QFT) like the FFT assumes $N = 2^n$ and that the amplitudes $a_x \equiv a(x)$ of the superposition state $|\psi\rangle$ are the function to be transformed

$$|\psi\rangle = \sum_{x=0}^{N-1} a(x)|x\rangle \longrightarrow \sum_{x=0}^{N-1} A(x)|x\rangle$$

The QFT does not require an output register as the output quantum state contains the Fourier transform in its complex amplitudes

If the initial state is such that the amplitudes are a periodic function with period $r = 2^m$, the resultant Ax would be zero unless $x = j\frac{N}{r}$ with $j = 0, 1, \ldots, \frac{N}{r} - 1$

When $r \neq 2^m$, the QFT will produce an approximate solution with higher probability coefficients for states with integers near multiples of $\frac{N}{r}$

As the base for the QFT, $N = 2^n$ is increased, the approximation improves

Quantum Fourier transform

The quantum Fourier transform (QFT) like the FFT assumes $N = 2^n$ and that the amplitudes $a_x \equiv a(x)$ of the superposition state $|\psi\rangle$ are the function to be transformed

$$|\psi\rangle = \sum_{x=0}^{N-1} a(x)|x\rangle \longrightarrow \sum_{x=0}^{N-1} A(x)|x\rangle$$

The QFT does not require an output register as the output quantum state contains the Fourier transform in its complex amplitudes

If the initial state is such that the amplitudes are a periodic function with period $r = 2^m$, the resultant Ax would be zero unless $x = j\frac{N}{r}$ with $j = 0, 1, \ldots, \frac{N}{r} - 1$

When $r \neq 2^m$, the QFT will produce an approximate solution with higher probability coefficients for states with integers near multiples of $\frac{N}{r}$

As the base for the QFT, $N = 2^n$ is increased, the approximation improves

The QFT is exponentially faster, $[O(n^2)]$, than the discrete $[O(N^2)]$, and the fast $[O(N \log N)]$ transforms

Quantum Fourier transform

For $N = 2^n$, the quantum Fourier transform acting on $|k\rangle$ is defined as

Quantum Fourier transform

For $N = 2^n$, the quantum Fourier transform acting on $|k\rangle$ is defined as

$$U_F^{(n)}|k\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} e^{2\pi ikx/N} |x\rangle$$

Quantum Fourier transform

For $N = 2^n$, the quantum Fourier transform acting on $|k\rangle$ is defined as

$$U_F^{(n)}|k\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} e^{2\pi ikx/N}|x\rangle$$

For $N = 1$ the quantum Fourier transform is identical to the Hadamard transform

Quantum Fourier transform

For $N = 2^n$, the quantum Fourier transform acting on $|k\rangle$ is defined as

$$U_F^{(n)}|k\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} e^{2\pi i k x/N}|x\rangle$$

For $N = 1$ the quantum Fourier transform is identical to the Hadamard transform

$$U_F^{(1)}|0\rangle = \frac{1}{\sqrt{2}} \sum_{x=0}^{1} e^{0}|x\rangle$$

Quantum Fourier transform

For $N = 2^n$, the quantum Fourier transform acting on $|k\rangle$ is defined as

$$U_F^{(n)}|k\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} e^{2\pi ikx/N}|x\rangle$$

For $N = 1$ the quantum Fourier transform is identical to the Hadamard transform

$$U_F^{(1)}|0\rangle = \frac{1}{\sqrt{2}} \sum_{x=0}^{1} e^{0}|x\rangle = \tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle),$$

Quantum Fourier transform

For $N = 2^n$, the quantum Fourier transform acting on $|k\rangle$ is defined as

$$U_F^{(n)}|k\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} e^{2\pi i k x/N}|x\rangle$$

For $N = 1$ the quantum Fourier transform is identical to the Hadamard transform

$$U_F^{(1)}|0\rangle = \frac{1}{\sqrt{2}} \sum_{x=0}^{1} e^{0}|x\rangle = \tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \qquad U_F^{(1)}|1\rangle = \frac{1}{\sqrt{2}} \sum_{x=0}^{1} e^{\pi i x}|x\rangle$$

Quantum Fourier transform

For $N = 2^n$, the quantum Fourier transform acting on $|k\rangle$ is defined as

$$U_F^{(n)}|k\rangle = \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1} e^{2\pi i k x/N}|x\rangle$$

For $N = 1$ the quantum Fourier transform is identical to the Hadamard transform

$$U_F^{(1)}|0\rangle = \frac{1}{\sqrt{2}}\sum_{x=0}^{1} e^{0}|x\rangle = \tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \qquad U_F^{(1)}|1\rangle = \frac{1}{\sqrt{2}}\sum_{x=0}^{1} e^{\pi i x}|x\rangle = \tfrac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

Quantum Fourier transform

For $N = 2^n$, the quantum Fourier transform acting on $|k\rangle$ is defined as

$$U_F^{(n)}|k\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} e^{2\pi ikx/N}|x\rangle$$

For $N = 1$ the quantum Fourier transform is identical to the Hadamard transform

$$U_F^{(1)}|0\rangle = \frac{1}{\sqrt{2}} \sum_{x=0}^{1} e^{0}|x\rangle = \tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \qquad U_F^{(1)}|1\rangle = \frac{1}{\sqrt{2}} \sum_{x=0}^{1} e^{\pi ix}|x\rangle = \tfrac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

The recursive decomposition of the fast Fourier transform is used to compute the $N = 2^n$ transform

Quantum Fourier transform

For $N = 2^n$, the quantum Fourier transform acting on $|k\rangle$ is defined as

$$U_F^{(n)}|k\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} e^{2\pi i k x/N}|x\rangle$$

For $N = 1$ the quantum Fourier transform is identical to the Hadamard transform

$$U_F^{(1)}|0\rangle = \frac{1}{\sqrt{2}} \sum_{x=0}^{1} e^{0}|x\rangle = \tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \qquad U_F^{(1)}|1\rangle = \frac{1}{\sqrt{2}} \sum_{x=0}^{1} e^{\pi i x}|x\rangle = \tfrac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

The recursive decomposition of the fast Fourier transform is used to compute the $N = 2^n$ transform

$$U_F^{(k+1)} = \frac{1}{\sqrt{2}} \begin{pmatrix} I^{(k)} & D^{(k)} \\ I^{(k)} & -D^{(k)} \end{pmatrix} \begin{pmatrix} U_F^{(k)} & 0 \\ 0 & U_F^{(k)} \end{pmatrix} R^{(k+1)}$$

Quantum Fourier transform

For $N = 2^n$, the quantum Fourier transform acting on $|k\rangle$ is defined as

$$U_F^{(n)}|k\rangle = \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1} e^{2\pi ikx/N}|x\rangle$$

For $N = 1$ the quantum Fourier transform is identical to the Hadamard transform

$$U_F^{(1)}|0\rangle = \frac{1}{\sqrt{2}}\sum_{x=0}^{1} e^{0}|x\rangle = \tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \qquad U_F^{(1)}|1\rangle = \frac{1}{\sqrt{2}}\sum_{x=0}^{1} e^{\pi ix}|x\rangle = \tfrac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

The recursive decomposition of the fast Fourier transform is used to compute the $N = 2^n$ transform

$$U_F^{(k+1)} = \frac{1}{\sqrt{2}}\begin{pmatrix} I^{(k)} & D^{(k)} \\ I^{(k)} & -D^{(k)} \end{pmatrix}\begin{pmatrix} U_F^{(k)} & 0 \\ 0 & U_F^{(k)} \end{pmatrix} R^{(k+1)}$$

All matrices are unitary and can be implemented efficiently on a quantum computer

QFT implementation

$$U_F^{(k+1)} = \frac{1}{\sqrt{2}} \begin{pmatrix} I^{(k)} & D^{(k)} \\ I^{(k)} & -D^{(k)} \end{pmatrix} \begin{pmatrix} U_F^{(k)} & 0 \\ 0 & U_F^{(k)} \end{pmatrix} R^{(k+1)}$$

QFT implementation

$$U_F^{(k+1)} = \frac{1}{\sqrt{2}} \begin{pmatrix} I^{(k)} & D^{(k)} \\ I^{(k)} & -D^{(k)} \end{pmatrix} \begin{pmatrix} U_F^{(k)} & 0 \\ 0 & U_F^{(k)} \end{pmatrix} R^{(k+1)}$$

The implementation starts with the rotation $R^{(k+1)}$

QFT implementation

$$U_F^{(k+1)} = \frac{1}{\sqrt{2}} \begin{pmatrix} I^{(k)} & D^{(k)} \\ I^{(k)} & -D^{(k)} \end{pmatrix} \begin{pmatrix} U_F^{(k)} & 0 \\ 0 & U_F^{(k)} \end{pmatrix} R^{(k+1)}$$

The implementation starts with the rotation $R^{(k+1)}$

$$R^{(k+1)} = \sum_{i=0}^{2^k-1} |i\rangle\langle 2i| + |i+2^k\rangle\langle 2i+1|$$

QFT implementation

$$U_F^{(k+1)} = \frac{1}{\sqrt{2}} \begin{pmatrix} I^{(k)} & D^{(k)} \\ I^{(k)} & -D^{(k)} \end{pmatrix} \begin{pmatrix} U_F^{(k)} & 0 \\ 0 & U_F^{(k)} \end{pmatrix} R^{(k+1)}$$

The implementation starts with the rotation $R^{(k+1)}$

$$R^{(k+1)} = \sum_{i=0}^{2^k-1} |i\rangle\langle 2i| + |i+2^k\rangle\langle 2i+1|$$

This can be accomplished by a permutation of the $k+1$ qubits,

QFT implementation

$$U_F^{(k+1)} = \frac{1}{\sqrt{2}} \begin{pmatrix} I^{(k)} & D^{(k)} \\ I^{(k)} & -D^{(k)} \end{pmatrix} \begin{pmatrix} U_F^{(k)} & 0 \\ 0 & U_F^{(k)} \end{pmatrix} R^{(k+1)}$$

The implementation starts with the rotation $R^{(k+1)}$

$$000 \rightarrow 000$$

$$R^{(k+1)} = \sum_{i=0}^{2^k-1} |i\rangle\langle 2i| + |i+2^k\rangle\langle 2i+1|$$

This can be accomplished by a permutation of the $k+1$ qubits,

QFT implementation

$$U_F^{(k+1)} = \frac{1}{\sqrt{2}} \begin{pmatrix} I^{(k)} & D^{(k)} \\ I^{(k)} & -D^{(k)} \end{pmatrix} \begin{pmatrix} U_F^{(k)} & 0 \\ 0 & U_F^{(k)} \end{pmatrix} R^{(k+1)}$$

The implementation starts with the rotation $R^{(k+1)}$

$$R^{(k+1)} = \sum_{i=0}^{2^k-1} |i\rangle\langle 2i| + |i+2^k\rangle\langle 2i+1|$$

This can be accomplished by a permutation of the $k+1$ qubits,

$$\begin{aligned} 000 &\rightarrow 000 \\ 001 &\rightarrow 010 \end{aligned}$$

QFT implementation

$$U_F^{(k+1)} = \frac{1}{\sqrt{2}} \begin{pmatrix} I^{(k)} & D^{(k)} \\ I^{(k)} & -D^{(k)} \end{pmatrix} \begin{pmatrix} U_F^{(k)} & 0 \\ 0 & U_F^{(k)} \end{pmatrix} R^{(k+1)}$$

The implementation starts with the rotation $R^{(k+1)}$

$$R^{(k+1)} = \sum_{i=0}^{2^k-1} |i\rangle\langle 2i| + |i+2^k\rangle\langle 2i+1|$$

This can be accomplished by a permutation of the $k+1$ qubits,

$$\begin{aligned} 000 &\rightarrow 000 \\ 001 &\rightarrow 010 \\ 010 &\rightarrow 100 \end{aligned}$$

QFT implementation

$$U_F^{(k+1)} = \frac{1}{\sqrt{2}} \begin{pmatrix} I^{(k)} & D^{(k)} \\ I^{(k)} & -D^{(k)} \end{pmatrix} \begin{pmatrix} U_F^{(k)} & 0 \\ 0 & U_F^{(k)} \end{pmatrix} R^{(k+1)}$$

The implementation starts with the rotation $R^{(k+1)}$

$$R^{(k+1)} = \sum_{i=0}^{2^k-1} |i\rangle\langle 2i| + |i+2^k\rangle\langle 2i+1|$$

This can be accomplished by a permutation of the $k+1$ qubits,

$$\begin{aligned} 000 &\rightarrow 000 \\ 001 &\rightarrow 010 \\ 010 &\rightarrow 100 \\ 011 &\rightarrow 110 \end{aligned}$$

QFT implementation

$$U_F^{(k+1)} = \frac{1}{\sqrt{2}} \begin{pmatrix} I^{(k)} & D^{(k)} \\ I^{(k)} & -D^{(k)} \end{pmatrix} \begin{pmatrix} U_F^{(k)} & 0 \\ 0 & U_F^{(k)} \end{pmatrix} R^{(k+1)}$$

The implementation starts with the rotation $R^{(k+1)}$

$$R^{(k+1)} = \sum_{i=0}^{2^k-1} |i\rangle\langle 2i| + |i+2^k\rangle\langle 2i+1|$$

This can be accomplished by a permutation of the $k+1$ qubits,

$$\begin{aligned} 000 &\rightarrow 000 \\ 001 &\rightarrow 010 \\ 010 &\rightarrow 100 \\ 011 &\rightarrow 110 \\ 100 &\rightarrow 001 \\ 101 &\rightarrow 011 \end{aligned}$$

QFT implementation

$$U_F^{(k+1)} = \frac{1}{\sqrt{2}} \begin{pmatrix} I^{(k)} & D^{(k)} \\ I^{(k)} & -D^{(k)} \end{pmatrix} \begin{pmatrix} U_F^{(k)} & 0 \\ 0 & U_F^{(k)} \end{pmatrix} R^{(k+1)}$$

The implementation starts with the rotation $R^{(k+1)}$

$$R^{(k+1)} = \sum_{i=0}^{2^k-1} |i\rangle\langle 2i| + |i+2^k\rangle\langle 2i+1|$$

This can be accomplished by a permutation of the $k+1$ qubits,

$$\begin{aligned} 000 &\rightarrow 000 \\ 001 &\rightarrow 010 \\ 010 &\rightarrow 100 \\ 011 &\rightarrow 110 \\ 100 &\rightarrow 001 \\ 101 &\rightarrow 011 \\ 110 &\rightarrow 101 \end{aligned}$$

QFT implementation

$$U_F^{(k+1)} = \frac{1}{\sqrt{2}} \begin{pmatrix} I^{(k)} & D^{(k)} \\ I^{(k)} & -D^{(k)} \end{pmatrix} \begin{pmatrix} U_F^{(k)} & 0 \\ 0 & U_F^{(k)} \end{pmatrix} R^{(k+1)}$$

The implementation starts with the rotation $R^{(k+1)}$

$$R^{(k+1)} = \sum_{i=0}^{2^k-1} |i\rangle\langle 2i| + |i+2^k\rangle\langle 2i+1|$$

This can be accomplished by a permutation of the $k+1$ qubits,

$$\begin{aligned} 000 &\rightarrow 000 \\ 001 &\rightarrow 010 \\ 010 &\rightarrow 100 \\ 011 &\rightarrow 110 \\ 100 &\rightarrow 001 \\ 101 &\rightarrow 011 \\ 110 &\rightarrow 101 \\ 111 &\rightarrow 111 \end{aligned}$$

QFT implementation

$$U_F^{(k+1)} = \frac{1}{\sqrt{2}} \begin{pmatrix} I^{(k)} & D^{(k)} \\ I^{(k)} & -D^{(k)} \end{pmatrix} \begin{pmatrix} U_F^{(k)} & 0 \\ 0 & U_F^{(k)} \end{pmatrix} R^{(k+1)}$$

The implementation starts with the rotation $R^{(k+1)}$

$$R^{(k+1)} = \sum_{i=0}^{2^k-1} |i\rangle\langle 2i| + |i+2^k\rangle\langle 2i+1|$$

This can be accomplished by a permutation of the $k+1$ qubits, resulting in just the kind of shuffling needed

000	$\rightarrow$	000
001	$\rightarrow$	010
010	$\rightarrow$	100
011	$\rightarrow$	110
100	$\rightarrow$	001
101	$\rightarrow$	011
110	$\rightarrow$	101
111	$\rightarrow$	111

QFT implementation

$$U_F^{(k+1)} = \frac{1}{\sqrt{2}} \begin{pmatrix} I^{(k)} & D^{(k)} \\ I^{(k)} & -D^{(k)} \end{pmatrix} \begin{pmatrix} U_F^{(k)} & 0 \\ 0 & U_F^{(k)} \end{pmatrix} R^{(k+1)}$$

The implementation starts with the rotation $R^{(k+1)}$

$$R^{(k+1)} = \sum_{i=0}^{2^k-1} |i\rangle\langle 2i| + |i+2^k\rangle\langle 2i+1|$$

This can be accomplished by a permutation of the $k+1$ qubits, resulting in just the kind of shuffling needed

Only $k-1$ swap operations are needed to perform this permutation

$$\begin{array}{ccc} 000 & \rightarrow & 000 \\ 001 & \rightarrow & 010 \\ 010 & \rightarrow & 100 \\ 011 & \rightarrow & 110 \\ 100 & \rightarrow & 001 \\ 101 & \rightarrow & 011 \\ 110 & \rightarrow & 101 \\ 111 & \rightarrow & 111 \end{array}$$

QFT implementation

$$U_F^{(k+1)} = \frac{1}{\sqrt{2}} \begin{pmatrix} I^{(k)} & D^{(k)} \\ I^{(k)} & -D^{(k)} \end{pmatrix} \begin{pmatrix} U_F^{(k)} & 0 \\ 0 & U_F^{(k)} \end{pmatrix} R^{(k+1)}$$

The implementation starts with the rotation $R^{(k+1)}$

$$R^{(k+1)} = \sum_{i=0}^{2^k-1} |i\rangle\langle 2i| + |i+2^k\rangle\langle 2i+1|$$

This can be accomplished by a permutation of the $k+1$ qubits, resulting in just the kind of shuffling needed

Only $k-1$ swap operations are needed to perform this permutation

Next is the QFT transformation matrix U_F^{k+1},

000	$\rightarrow$	000
001	$\rightarrow$	010
010	$\rightarrow$	100
011	$\rightarrow$	110
100	$\rightarrow$	001
101	$\rightarrow$	011
110	$\rightarrow$	101
111	$\rightarrow$	111

QFT implementation

$$U_F^{(k+1)} = \frac{1}{\sqrt{2}} \begin{pmatrix} I^{(k)} & D^{(k)} \\ I^{(k)} & -D^{(k)} \end{pmatrix} \begin{pmatrix} U_F^{(k)} & 0 \\ 0 & U_F^{(k)} \end{pmatrix} R^{(k+1)}$$

The implementation starts with the rotation $R^{(k+1)}$

$$R^{(k+1)} = \sum_{i=0}^{2^k-1} |i\rangle\langle 2i| + |i+2^k\rangle\langle 2i+1|$$

This can be accomplished by a permutation of the $k+1$ qubits, resulting in just the kind of shuffling needed

Only $k-1$ swap operations are needed to perform this permutation

Next is the QFT transformation matrix U_F^{k+1},

$$\begin{aligned} 000 &\rightarrow 000 \\ 001 &\rightarrow 010 \\ 010 &\rightarrow 100 \\ 011 &\rightarrow 110 \\ 100 &\rightarrow 001 \\ 101 &\rightarrow 011 \\ 110 &\rightarrow 101 \\ 111 &\rightarrow 111 \end{aligned}$$

$$\begin{pmatrix} U_F^{(k)} & 0 \\ 0 & U_F^{(k)} \end{pmatrix} = I \otimes U_F^{(k)}$$

QFT implementation

$$U_F^{(k+1)} = \frac{1}{\sqrt{2}} \begin{pmatrix} I^{(k)} & D^{(k)} \\ I^{(k)} & -D^{(k)} \end{pmatrix} \begin{pmatrix} U_F^{(k)} & 0 \\ 0 & U_F^{(k)} \end{pmatrix} R^{(k+1)}$$

The implementation starts with the rotation $R^{(k+1)}$

$$R^{(k+1)} = \sum_{i=0}^{2^k-1} |i\rangle\langle 2i| + |i+2^k\rangle\langle 2i+1|$$

This can be accomplished by a permutation of the $k+1$ qubits, resulting in just the kind of shuffling needed

Only $k-1$ swap operations are needed to perform this permutation

$$\begin{array}{lcl} 000 & \rightarrow & 000 \\ 001 & \rightarrow & 010 \\ 010 & \rightarrow & 100 \\ 011 & \rightarrow & 110 \\ 100 & \rightarrow & 001 \\ 101 & \rightarrow & 011 \\ 110 & \rightarrow & 101 \\ 111 & \rightarrow & 111 \end{array}$$

Next is the QFT transformation matrix U_F^{k+1}, which is implemented by recursively applying the QFT to qubits 0 to k

$$\begin{pmatrix} U_F^{(k)} & 0 \\ 0 & U_F^{(k)} \end{pmatrix} = I \otimes U_F^{(k)}$$

QFT implementation

$$U_F^{(k+1)} = \frac{1}{\sqrt{2}} \begin{pmatrix} I^{(k)} & D^{(k)} \\ I^{(k)} & -D^{(k)} \end{pmatrix} \begin{pmatrix} U_F^{(k)} & 0 \\ 0 & U_F^{(k)} \end{pmatrix} R^{(k+1)}$$

QFT implementation

$$U_F^{(k+1)} = \frac{1}{\sqrt{2}} \begin{pmatrix} I^{(k)} & D^{(k)} \\ I^{(k)} & -D^{(k)} \end{pmatrix} \begin{pmatrix} U_F^{(k)} & 0 \\ 0 & U_F^{(k)} \end{pmatrix} R^{(k+1)}$$

The diagonal matrix of phase shifts, $D^{(k)}$ is decomposed recursively,

QFT implementation

$$U_F^{(k+1)} = \frac{1}{\sqrt{2}} \begin{pmatrix} I^{(k)} & D^{(k)} \\ I^{(k)} & -D^{(k)} \end{pmatrix} \begin{pmatrix} U_F^{(k)} & 0 \\ 0 & U_F^{(k)} \end{pmatrix} R^{(k+1)}$$

The diagonal matrix of phase shifts, $D^{(k)}$ is decomposed recursively, where $\omega_{(k+1)} = e^{2\pi i/2^{k+1}}$

$$D^{(k)} = D^{(k-1)} \otimes \begin{pmatrix} 1 & 0 \\ 0 & \omega_{(k+1)} \end{pmatrix}$$

QFT implementation

$$U_F^{(k+1)} = \frac{1}{\sqrt{2}} \begin{pmatrix} I^{(k)} & D^{(k)} \\ I^{(k)} & -D^{(k)} \end{pmatrix} \begin{pmatrix} U_F^{(k)} & 0 \\ 0 & U_F^{(k)} \end{pmatrix} R^{(k+1)}$$

The diagonal matrix of phase shifts, $D^{(k)}$ is decomposed recursively, where $\omega_{(k+1)} = e^{2\pi i/2^{k+1}}$

$$D^{(k)} = D^{(k-1)} \otimes \begin{pmatrix} 1 & 0 \\ 0 & \omega_{(k+1)} \end{pmatrix}$$

This recursive decomposition applies a phase rotation of ω_{j+1} to the j^{th} qubit for $1 \leq j \leq k$ and can be implemented using k single-qubit gates

QFT implementation

$$U_F^{(k+1)} = \frac{1}{\sqrt{2}} \begin{pmatrix} I^{(k)} & D^{(k)} \\ I^{(k)} & -D^{(k)} \end{pmatrix} \begin{pmatrix} U_F^{(k)} & 0 \\ 0 & U_F^{(k)} \end{pmatrix} R^{(k+1)}$$

The diagonal matrix of phase shifts, $D^{(k)}$ is decomposed recursively, where $\omega_{(k+1)} = e^{2\pi i/2^{k+1}}$

$$D^{(k)} = D^{(k-1)} \otimes \begin{pmatrix} 1 & 0 \\ 0 & \omega_{(k+1)} \end{pmatrix}$$

This recursive decomposition applies a phase rotation of ω_{j+1} to the j^{th} qubit for $1 \le j \le k$ and can be implemented using k single-qubit gates

Thus only k gates are necessary for the implementation of the controlled $D^{(k)}$

QFT implementation

$$U_F^{(k+1)} = \frac{1}{\sqrt{2}} \begin{pmatrix} I^{(k)} & D^{(k)} \\ I^{(k)} & -D^{(k)} \end{pmatrix} \begin{pmatrix} U_F^{(k)} & 0 \\ 0 & U_F^{(k)} \end{pmatrix} R^{(k+1)}$$

The diagonal matrix of phase shifts, $D^{(k)}$ is decomposed recursively, where $\omega_{(k+1)} = e^{2\pi i/2^{k+1}}$

$$D^{(k)} = D^{(k-1)} \otimes \begin{pmatrix} 1 & 0 \\ 0 & \omega_{(k+1)} \end{pmatrix}$$

This recursive decomposition applies a phase rotation of ω_{j+1} to the j^{th} qubit for $1 \leq j \leq k$ and can be implemented using k single-qubit gates

Thus only k gates are necessary for the implementation of the controlled $D^{(k)}$

$$\frac{1}{\sqrt{2}} \begin{pmatrix} I^{(k)} & D^{(k)} \\ I^{(k)} & -D^{(k)} \end{pmatrix} = \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) \langle 0| \otimes I^{(k)} + \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) \langle 1| \otimes D^{(k)}$$

QFT implementation

$$U_F^{(k+1)} = \frac{1}{\sqrt{2}} \begin{pmatrix} I^{(k)} & D^{(k)} \\ I^{(k)} & -D^{(k)} \end{pmatrix} \begin{pmatrix} U_F^{(k)} & 0 \\ 0 & U_F^{(k)} \end{pmatrix} R^{(k+1)}$$

The diagonal matrix of phase shifts, $D^{(k)}$ is decomposed recursively, where $\omega_{(k+1)} = e^{2\pi i/2^{k+1}}$

$$D^{(k)} = D^{(k-1)} \otimes \begin{pmatrix} 1 & 0 \\ 0 & \omega_{(k+1)} \end{pmatrix}$$

This recursive decomposition applies a phase rotation of ω_{j+1} to the j^{th} qubit for $1 \leq j \leq k$ and can be implemented using k single-qubit gates

Thus only k gates are necessary for the implementation of the controlled $D^{(k)}$

$$\frac{1}{\sqrt{2}} \begin{pmatrix} I^{(k)} & D^{(k)} \\ I^{(k)} & -D^{(k)} \end{pmatrix} = \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) \langle 0| \otimes I^{(k)} + \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) \langle 1| \otimes D^{(k)}$$

$$= (H|0\rangle\langle 0|) \otimes I^{(k)} + (H|1\rangle\langle 1|) \otimes D^{(k)}$$

QFT implementation

$$U_F^{(k+1)} = \frac{1}{\sqrt{2}} \begin{pmatrix} I^{(k)} & D^{(k)} \\ I^{(k)} & -D^{(k)} \end{pmatrix} \begin{pmatrix} U_F^{(k)} & 0 \\ 0 & U_F^{(k)} \end{pmatrix} R^{(k+1)}$$

The diagonal matrix of phase shifts, $D^{(k)}$ is decomposed recursively, where $\omega_{(k+1)} = e^{2\pi i/2^{k+1}}$

$$D^{(k)} = D^{(k-1)} \otimes \begin{pmatrix} 1 & 0 \\ 0 & \omega_{(k+1)} \end{pmatrix}$$

This recursive decomposition applies a phase rotation of ω_{j+1} to the j^{th} qubit for $1 \leq j \leq k$ and can be implemented using k single-qubit gates

Thus only k gates are necessary for the implementation of the controlled $D^{(k)}$

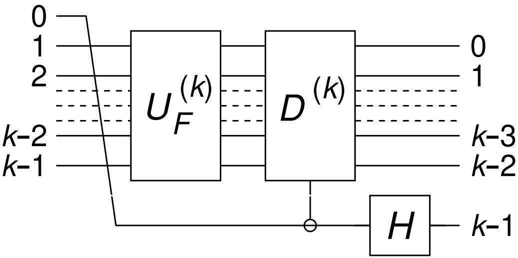
$$\begin{aligned} \frac{1}{\sqrt{2}} \begin{pmatrix} I^{(k)} & D^{(k)} \\ I^{(k)} & -D^{(k)} \end{pmatrix} &= \tfrac{1}{\sqrt{2}} (|0\rangle + |1\rangle) \langle 0| \otimes I^{(k)} + \tfrac{1}{\sqrt{2}} (|0\rangle - |1\rangle) \langle 1| \otimes D^{(k)} \\ &= (H|0\rangle\langle 0|) \otimes I^{(k)} + (H|1\rangle\langle 1|) \otimes D^{(k)} \\ &= \left(H \otimes I^{(k)}\right) \left(|0\rangle\langle 0| \otimes I^{(k)} + |1\rangle\langle 1| \otimes D^{(k)}\right) \end{aligned}$$

QFT implementation

One possible recursive circuit for QFT is

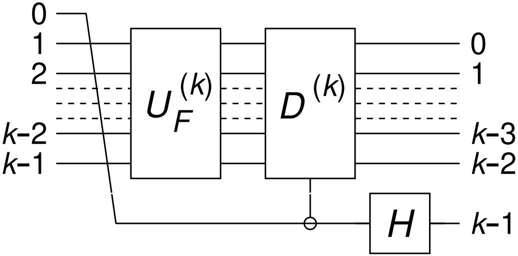
QFT implementation

One possible recursive circuit for QFT is



QFT implementation

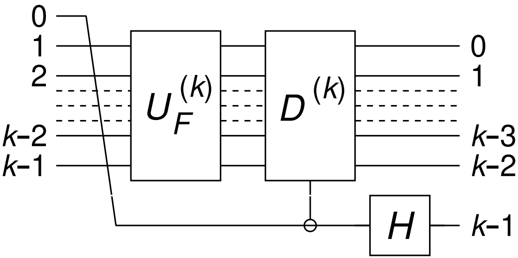
One possible recursive circuit for QFT is



The recursive circuit for $U_F^{(k+1)}$ can be implemented as

QFT implementation

One possible recursive circuit for QFT is



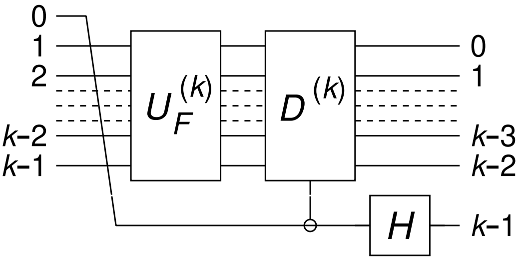
The recursive circuit for $U_F^{(k+1)}$ can be implemented as

define $QFT|x[1]\rangle = H|x\rangle$

$QFT|x[n]\rangle =$

QFT implementation

One possible recursive circuit for QFT is



The recursive circuit for $U_F^{(k+1)}$ can be implemented as

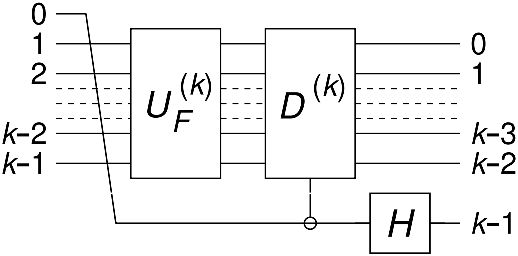
define $QFT|x[1]\rangle = H|x\rangle$

$QFT|x[n]\rangle =$

1. $Swap|x_0\rangle|x_1 \cdots x_{n-1}\rangle$

QFT implementation

One possible recursive circuit for QFT is



The recursive circuit for $U_F^{(k+1)}$ can be implemented as

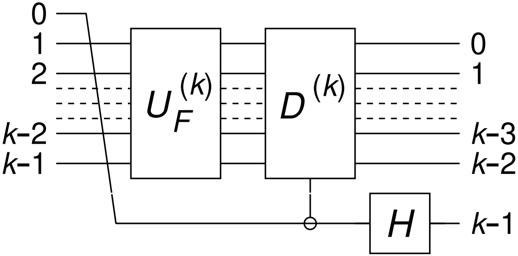
define $QFT|x[1]\rangle = H|x\rangle$

$QFT|x[n]\rangle =$

1. $Swap|x_0\rangle|x_1 \cdots x_{n-1}\rangle$
2. $QFT|x_0 \cdots x_{n-2}\rangle$

QFT implementation

One possible recursive circuit for QFT is



The recursive circuit for $U_F^{(k+1)}$ can be implemented as

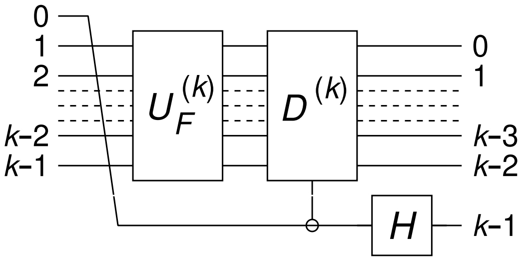
$$\textbf{define } QFT|x[1]\rangle = H|x\rangle$$

$$QFT|x[n]\rangle =$$

1. $Swap|x_0\rangle|x_1 \cdots x_{n-1}\rangle$
2. $QFT|x_0 \cdots x_{n-2}\rangle$
3. $|x_{n-1}\rangle$ **control** $D^{(n-1)}|x_0 \cdots x_{n-2}\rangle$

QFT implementation

One possible recursive circuit for QFT is



The recursive circuit for $U_F^{(k+1)}$ can be implemented as

define $QFT|x[1]\rangle = H|x\rangle$

$QFT|x[n]\rangle =$

1. $Swap|x_0\rangle|x_1 \cdots x_{n-1}\rangle$
2. $QFT|x_0 \cdots x_{n-2}\rangle$
3. $|x_{n-1}\rangle$ **control** $D^{(n-1)}|x_0 \cdots x_{n-2}\rangle$
4. $H|x_{n-1}\rangle$

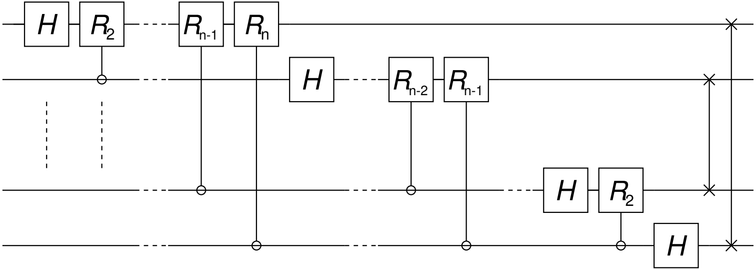
$D^{(k)}$ and $R^{(k)}$ can be implemented with $O(k)$ gates and the k^{th} step in the recursion adds $O(K)$ gates to the implementation of $U_F^{(n)}$, overall $U_F^{(n)}$ takes $O(n^2)$ gates to implement which exponentially faster than the $O(n2^n)$ for a classical FFT

QFT example

A more straightforward circuit provides insight to the recursive one just described

QFT example

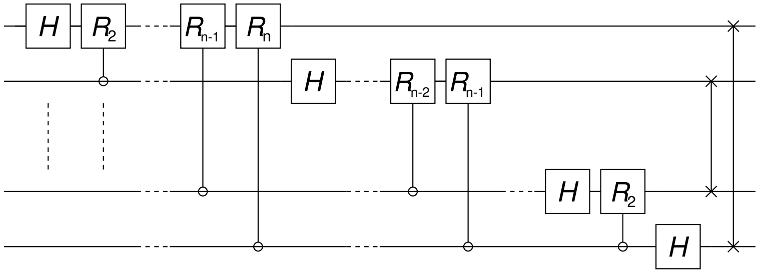
A more straightforward circuit provides insight to the recursive one just described



$$R_n = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^n} \end{pmatrix}$$

QFT example

A more straightforward circuit provides insight to the recursive one just described

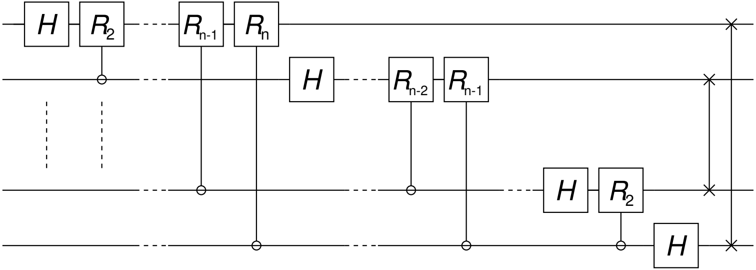


$$R_n = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^n} \end{pmatrix}$$

Starting with the high order qubit at the top, the Hadamard transform is followed by controlled rotations from each of the other $N-1$ qubits

QFT example

A more straightforward circuit provides insight to the recursive one just described



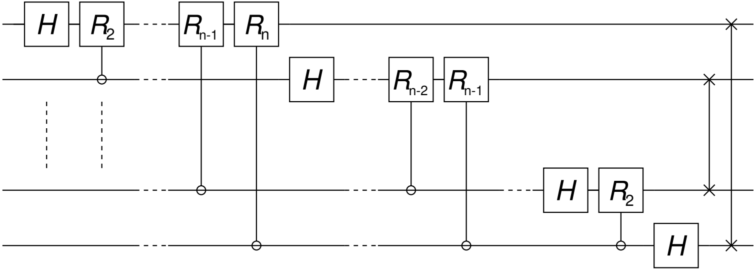
$$R_n = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^n} \end{pmatrix}$$

Starting with the high order qubit at the top, the Hadamard transform is followed by controlled rotations from each of the other $N-1$ qubits

The next qubit is transformed the same way using the $N-2$ lower order qubits, and so on until the last qubit which only has a Hadamard gate applied

QFT example

A more straightforward circuit provides insight to the recursive one just described



$$R_n = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^n} \end{pmatrix}$$

Starting with the high order qubit at the top, the Hadamard transform is followed by controlled rotations from each of the other $N-1$ qubits

The next qubit is transformed the same way using the $N-2$ lower order qubits, and so on until the last qubit which only has a Hadamard gate applied

At the end, all the qubits need to be swapped to recover the proper order