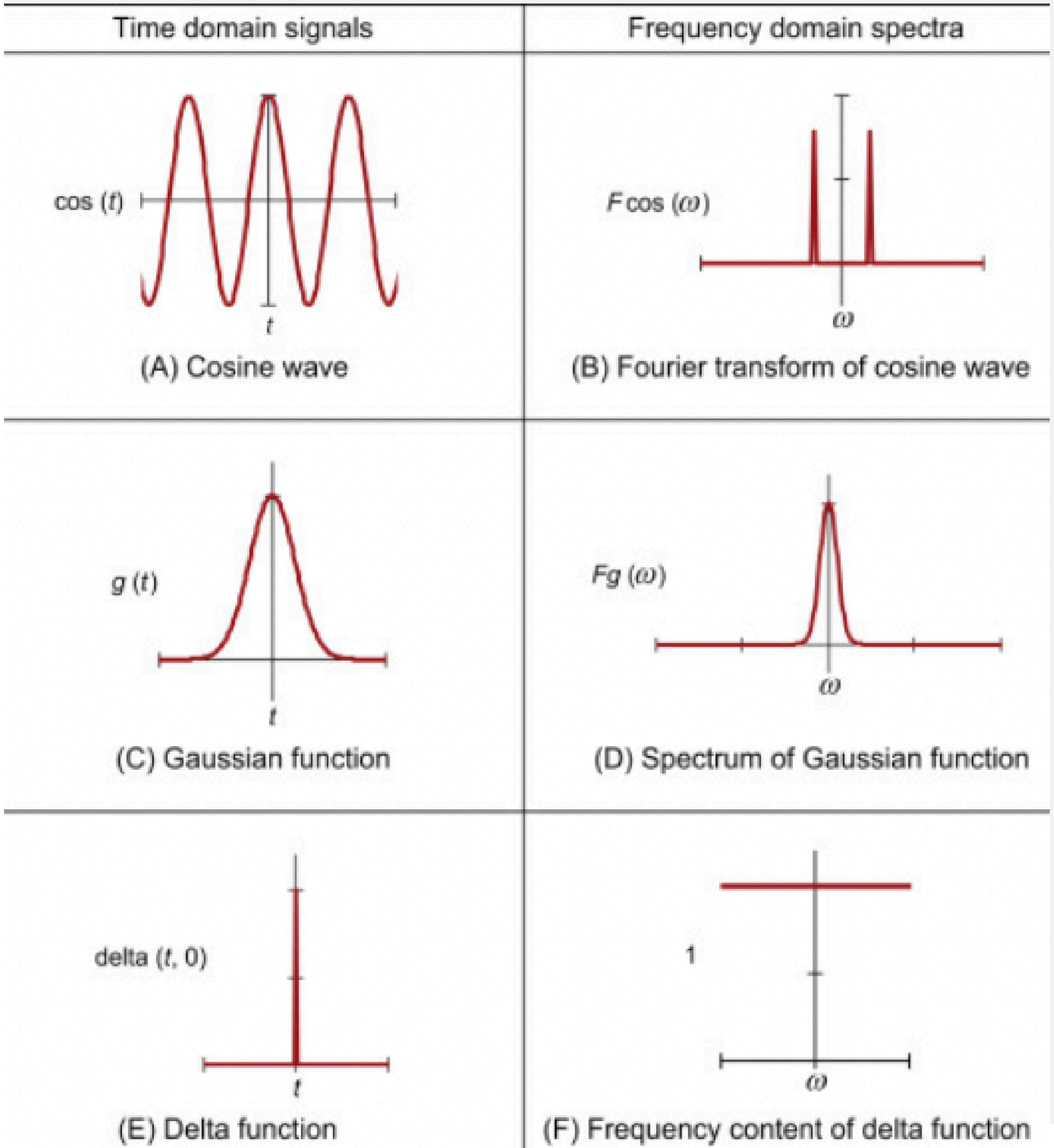
QUANTUM FOURIER TRANSFORM

Tensor Networks Approach

EDARA YASWANTH BALAJI

EP22BTECH11007



FOURIER TRANSFORM

- It is a mathematical model which helps to transform the signals between two different domains

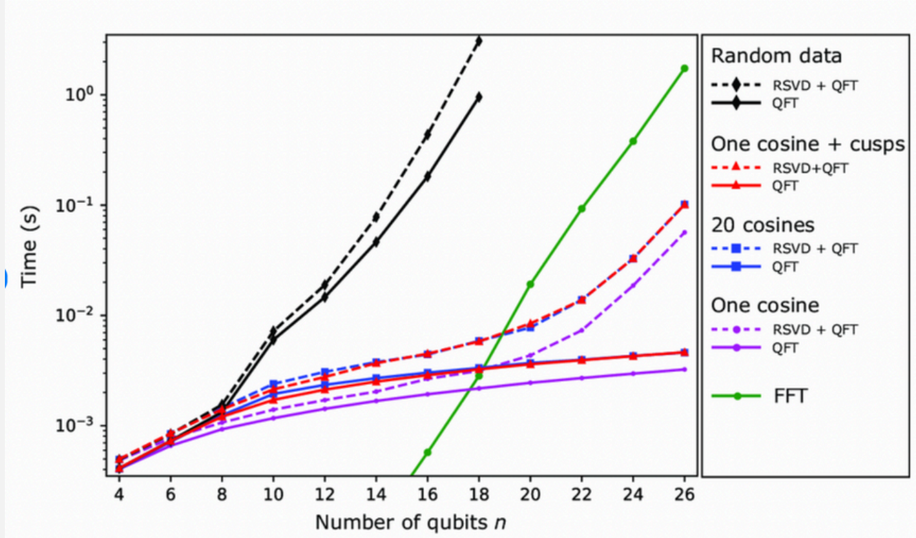
APPLICATIONS

- Signal processing
- Image processing
- Data analysis, ML
- Solving Laplace equations and many simulations
- many more!

PROBLEM

Fast Fourier Transform (FFT) algorithm is used to implement fourier transform.
Is it fast enough??

- For input data which is less than 2^20, it is really fast.
- But for larger data, the time keeps increasing



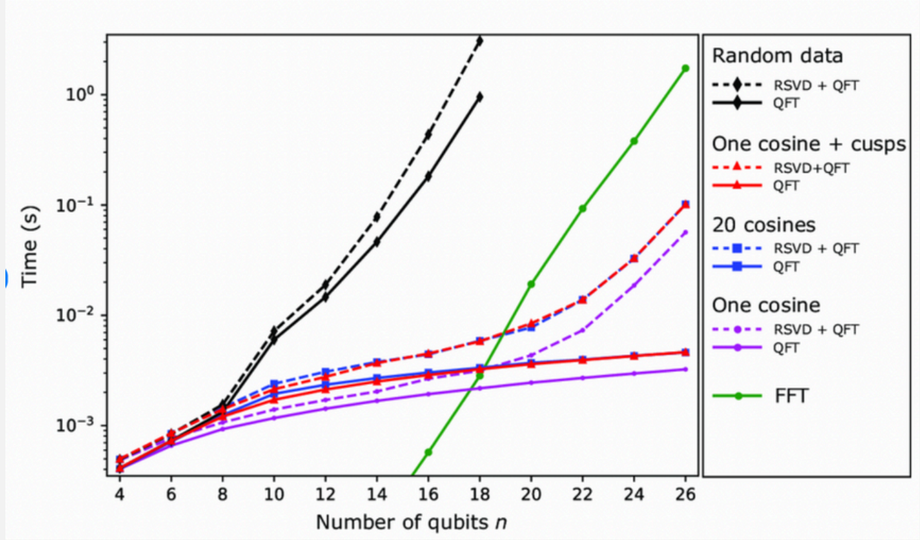
PRX QUANTUM 4, 040318 (2023)

$$y = F_n x \quad \text{with} \quad F_n = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & z & z^2 & \cdots & z^{n-1} \\ 1 & z^2 & z^4 & \cdots & z^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & z^{n-1} & z^{2(n-1)} & \cdots & z^{(n-1)^2} \end{bmatrix} \quad \text{and} \quad z = e^{-2i\pi/n}$$

PROBLEM

So can we find a better algorithm?

- Simulation Quantum Fourier Transform (QFT) on a classical computer efficiently can help us gain time advantage over FFT algorithm



PRX QUANTUM 4, 040318 (2023)

QUANTUM FOURIER TRANSFORM (QFT)

- Quantum Theory
- Tensor Networks
- Implementation
- Comparing with FFT

QUANTUM COMPUTING

- Qubits
  - Quantum Bits
  - Superposition
  - Entanglement

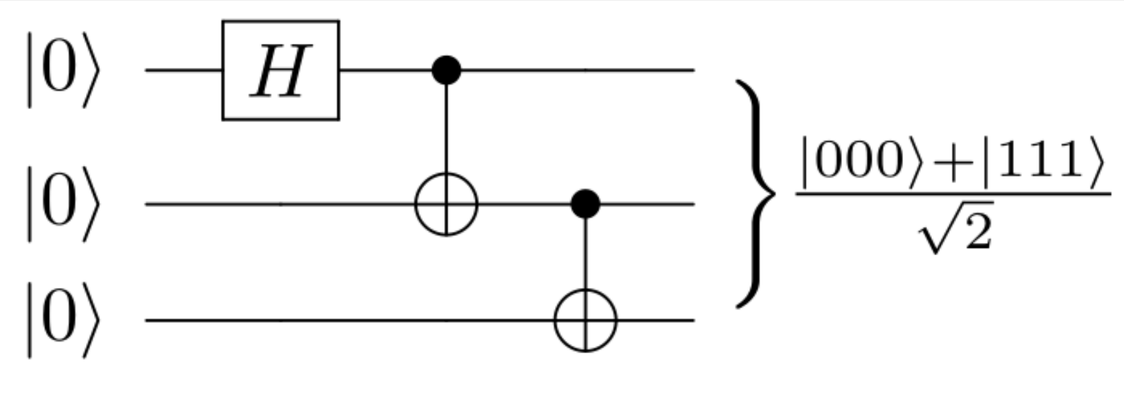
$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

$$\begin{bmatrix}\alpha\\ \beta\end{bmatrix} \text{ -> } \textit{Qubit State}$$

- Quantum Gates
  - Matrices (2^n * 2^n)
  - Single qubit and multi qubit
  - Unitary matrices

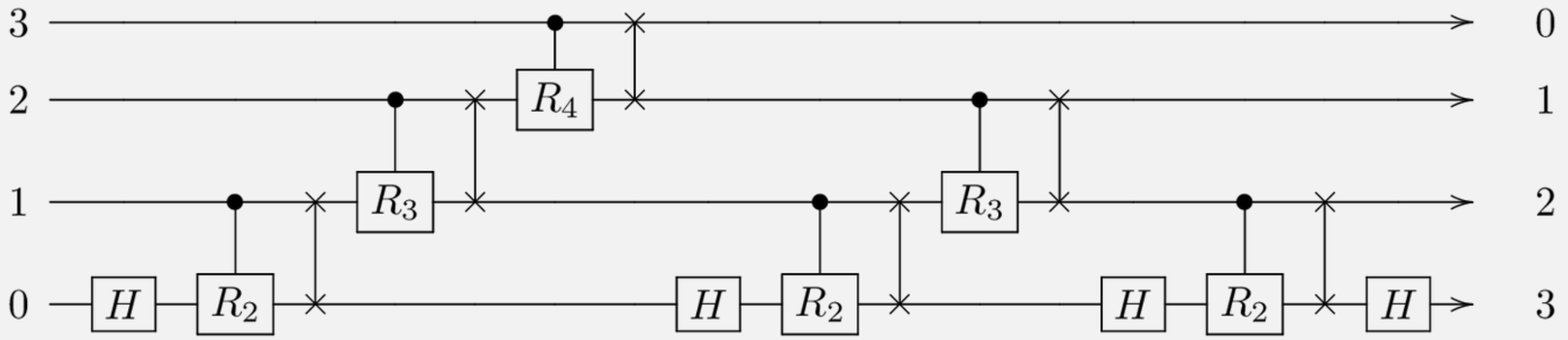
$$\mathsf{U}\,\mathsf{U}^{*} = \mathsf{I}, \quad \det \mathsf{U} = 1.$$

- Measurement
  - We get a result with a probability, not certain



QFT

- Input should be a 2^N array (N qubit quantum state)
- Output Amplitudes, will have their fourier components of the input state



$$\mathrm{QFT}_4 = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix}$$

$$\mathrm{QFT}_N|y\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} e^{2\pi i \frac{x y}{N}} |x\rangle$$

$$y = F_n x \quad \text{with} \quad F_n = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & z & z^2 & \cdots & z^{n-1} \\ 1 & z^2 & z^4 & \cdots & z^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & z^{n-1} & z^{2(n-1)} & \cdots & z^{(n-1)^2} \end{bmatrix}$$

Matrix size
2^n * 2^n

TENSOR NETWORKS

- **Most powerful simulators**
- **Quantum algorithm -> Quantum inspired classical algorithm**

- The unitary matrix for applying quantum gates are huge as “n” increases. So directly applying that won’t be possible
- So we represent it in a different way using the tensor networks approach to make it very efficient

We are trying to reduce the dimension of the tensor by finding the similarities between individual small tensors

Singular value decomposition (SVD)

$$M = \begin{bmatrix} 0.435839 & 0.223707 & 0.10 \\ 0.435839 & 0.223707 & -0.10 \\ 0.223707 & 0.435839 & 0.10 \\ 0.223707 & 0.435839 & -0.10 \end{bmatrix}$$

Can factorize as

$$\begin{bmatrix} 1/2 & -1/2 & 1/2 \\ 1/2 & -1/2 & -1/2 \\ 1/2 & 1/2 & 1/2 \\ 1/2 & 1/2 & -1/2 \end{bmatrix} \begin{bmatrix} 0.933 & 0 & 0 \\ 0 & 0.300 & 0 \\ 0 & 0 & 0.200 \end{bmatrix} \begin{bmatrix} 0.707107 & 0.707107 & 0 \\ -0.707107 & 0.707107 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

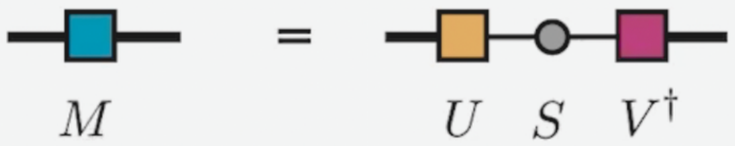
U S V^T

$$\begin{bmatrix} 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \end{bmatrix} \quad \begin{bmatrix} 0.933 \end{bmatrix} \quad \begin{bmatrix} 0.707107 & 0.707107 & 0 \end{bmatrix}$$

$$M_3 = \begin{bmatrix} 0.329773 & 0.329773 & 0 \\ 0.329773 & 0.329773 & 0 \\ 0.329773 & 0.329773 & 0 \\ 0.329773 & 0.329773 & 0 \end{bmatrix}$$

Truncating SVD =

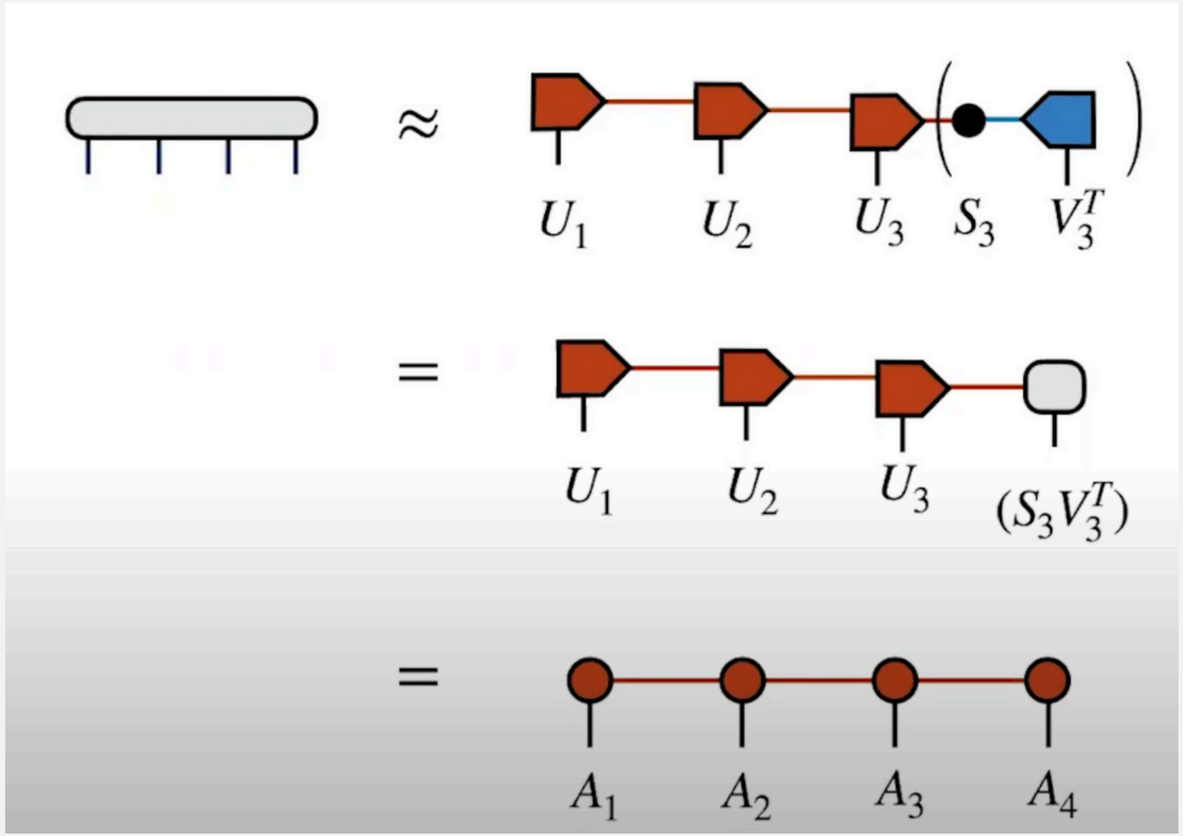
Controlled approximation for M



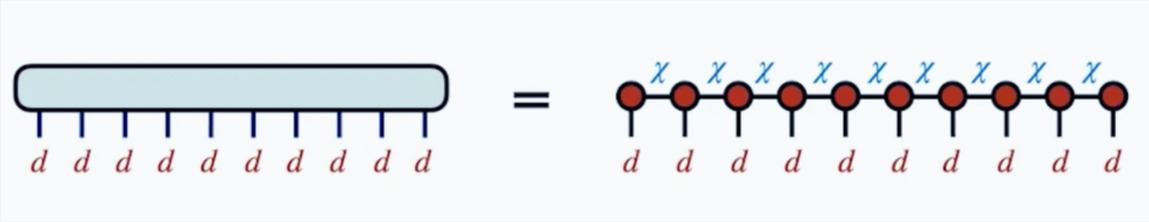
$$||M_3 - M||^2 = 0.13 = (0.3)^2 + (0.2)^2$$

Error

We put a cutoff limit and throw away some values



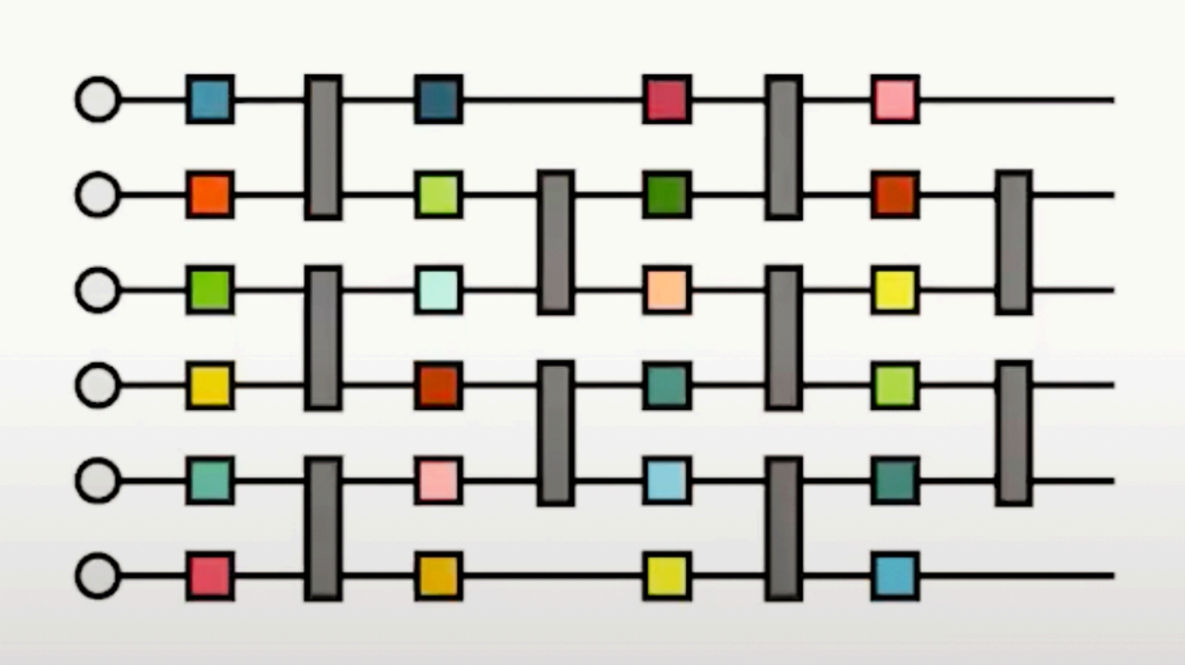
We keep repeating SVD till we get this result



$$d^N \longrightarrow N d \chi^2$$

The new state is called ‘Matrix Product State’ (MPS)

Quantum Circuit



Now we don’t need to store the whole matrix and multiply, we can apply indivudual gates (small matrices) directly to the required qubit.

IMPLEMENTATION

- Tensor network simulation is efficient only in the case of limited entanglement and low number of qubits
- QFT has low entanglement!

- Three major parts to optimize
  - Quantum circuit choice
  - using MPO
  - Ways of converting input to MPS

TAKING THE INPUT

```
N = 8
s = siteinds("Qubit",N)
```

```
function generate_basis_states(n)
    basis_states = []
    for i in 0:(2^n - 1)
        binary_str = bitstring(i)[end-n+1:end]  # Get last n bits of bitstring
        push!(basis_states, collect(binary_str))  # Collect splits string into array of characters
    end
    return basis_states
end

MPS_states = []
basis_states = generate_basis_states(N)
for i in 1:(2^N)
    push!(MPS_states, MPS(s,string.(basis_states[i])))
end
println(basis_states)
```

```
input_array = rand(2^N)

array = input_array / norm(input_array) # Input

ψ = array[1] *MPS(s, string.(basis_states[1]))
for i in 2:2^N

    ψ += array[i] * MPS(s, string.(basis_states[i]))
end

println(maxlinkdim(ψ))
```

Taking Input (Initial)

But this requires a lot of space and time to convert it

TAKING THE INPUT

```
N = 8
s = siteinds("Qubit",N)
```

```
# Taking the input

x = range(0, stop=2π, length=2^(N))
input_array = cos.(x)


array = input_array / norm(input_array) # Input
ITensors.disable_warn_order()

cutoff1 = 1E-18
maxdim1 = 10


T = ITensor(array,s)

start = time()

ψ = MPS(T,s;cutoff=cutoff1,maxdim=maxdim1)

orthogonalize!(ψ, 1) # Orthogonalize Psi
```

- Now we used MPS function directly
- Faster conversion, no additional space needed
- Problems
  - It reverses the order of the input data
  - Array size must be 2^N exactly

QUANTUM CIRCUIT

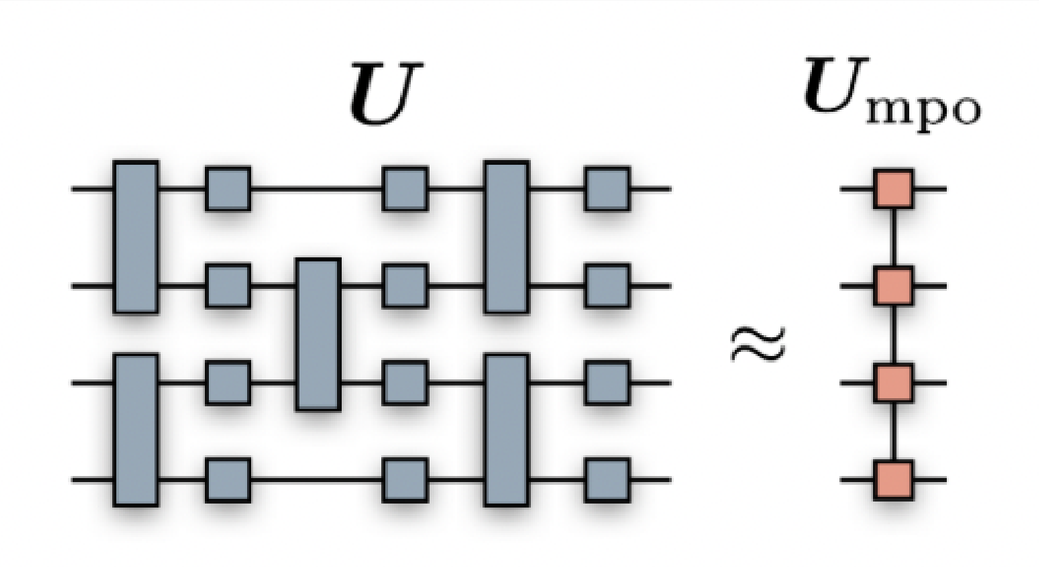
- There are two kinds of quantum circuits for QFT. One involves swapping of qubits and one does not. The one without swapping is observed to be faster after comparing

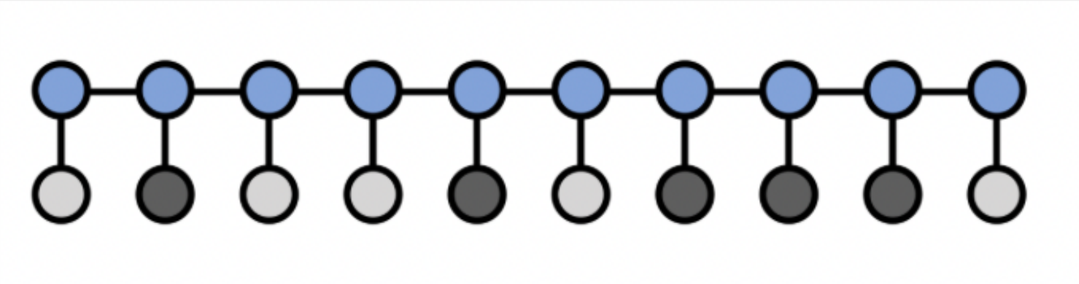
```
function H_gate(x)
    Hadamard = op("H",s[x]) # op  is  a function which converts matrices to ITensors
    push!(gates,Hadamard)
end

function swap_gate(i,j)
    swap = [ 1 0 0 0 ; 0 0 1 0; 0 1 0 0; 0 0 0 1]
    swap_gate = op(swap, s[i] , s[j])
    push!(gates,swap_gate)
end

function phase_gate(x,i,j)
    phase = [ 1 0 0 0 ; 0 1 0 0; 0 0 1 0; 0 0 0 exp(-1* im * 2*π/2^x)]
    phase_gate = op(phase, s[i] , s[j])
    # s[i] and s[j] are the control qubit
    push!(gates,phase_gate)
end
```

```
gates = ITensor[]

r_max = N

for i in 1:N

    H_gate(1)

    for j in 2:r_max
        phase_gate(j,j-1,j)
        swap_gate(j-1,j)
    end

    TEBD_cutoff = 1E-16
    ψ = apply(gates,ψ,cutoff=TEBD_cutoff)
    println("Maximum bond dimension in the MPS : " , maxlinkdim(ψ))
    empty!(gates)

    r_max = r_max - 1
end
```

MATRIX PRODUCT OPERATOR (MPO)

- The full circuit can be stores in a MPO format and then directly multiplied to the MPS after taking the input. This reduces time by a lot as we don’t need to apply gates anymore.





Directly multiply MPO and MPS

MATRIX PRODUCT OPERATOR (MPO)

- We can create MPO and store before itself and open the file to read them when needed

```
# Define the QFT MPO
function qft_mpo(N::Int, sites)
    # Initialize MPO with sites
    W = MPO(sites,"I")

    cutoff=1e-18
    # Apply Hadamard and phase gates to build the QFT MPO


    for i in 1:N
        W = apply(op("H",sites[N+1-i]),W,cutoff=cutoff)

        for j in i+1:N
            θ = π / 2^(j-i)
            P = phase_gate1(N+1-i,N+1-j,θ)
            # Update MPO by combining with phase gate tensor
            W = apply(P,W,cutoff=cutoff)


        end

    end

    return W
end
```

We multiply all the gates to an Identity and then store the resulting MPO in a file. This is ready to be multiplied with any input MPS.

MATRIX PRODUCT OPERATOR (MPO)

- The site indices must match to multiply them.
- So when we initialise the sites, we read the site id’s from MPO files

```
N = 25

f4 = h5open("Basis states Final/MPO_$N.h5","r")
W1 = read(f4,"W",MPO)
close(f4)

# Assuming `siteinds(W1)` gives the indices of the MPO
all_sites = siteinds(W1)

# Consider only physical sites
s = [pair[2] for pair in all_sites]
```

OBSERVATION MADE

- The MPS function reverses the data order
- But multiplying with MPO reverses it back! So finally we get correct order.

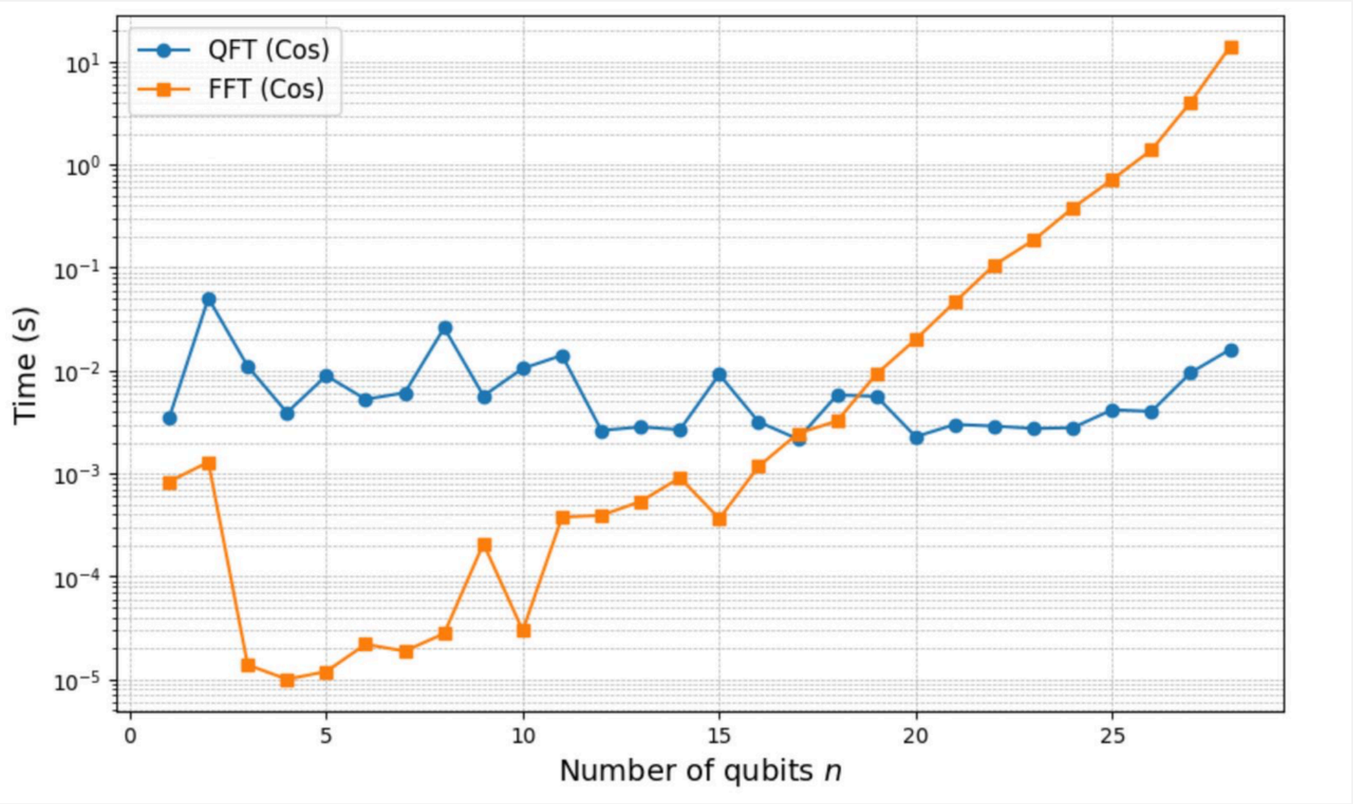
VERIFYING THE OUTPUT

- Dot product with basis states to verify the values

```
inner_products = []

MPS2 = []
for i in 1:2^3
    f2 = h5open("Basis states Final/MPS_$N/MPS_create_$i.h5","r")
    mps1 = read(f2,"M",MPS)
    push!(MPS2,mps1)
    close(f2)
end

for i in 1:(2^3)
    push!(inner_products, inner(MPS2[i], result))
    print(inner(MPS2[i], result)*norm(input_array)*2^(N/2))
    println()
end
```

- The basis states files are already created and stored

COMPARISION WITH FFT

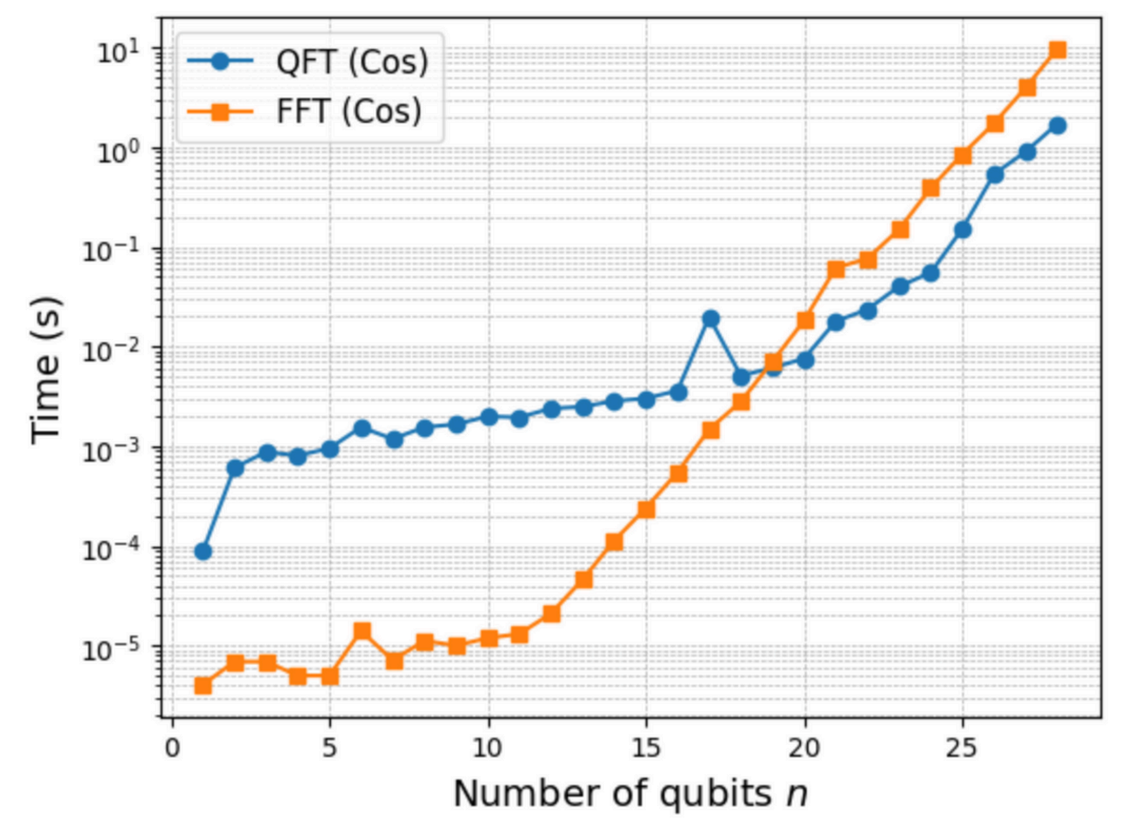


Time to convert the array to MPS is not included in this graph

QFT time = time to multiplying MPO and MPS

AS the number of qubits increase, we can see the advantage

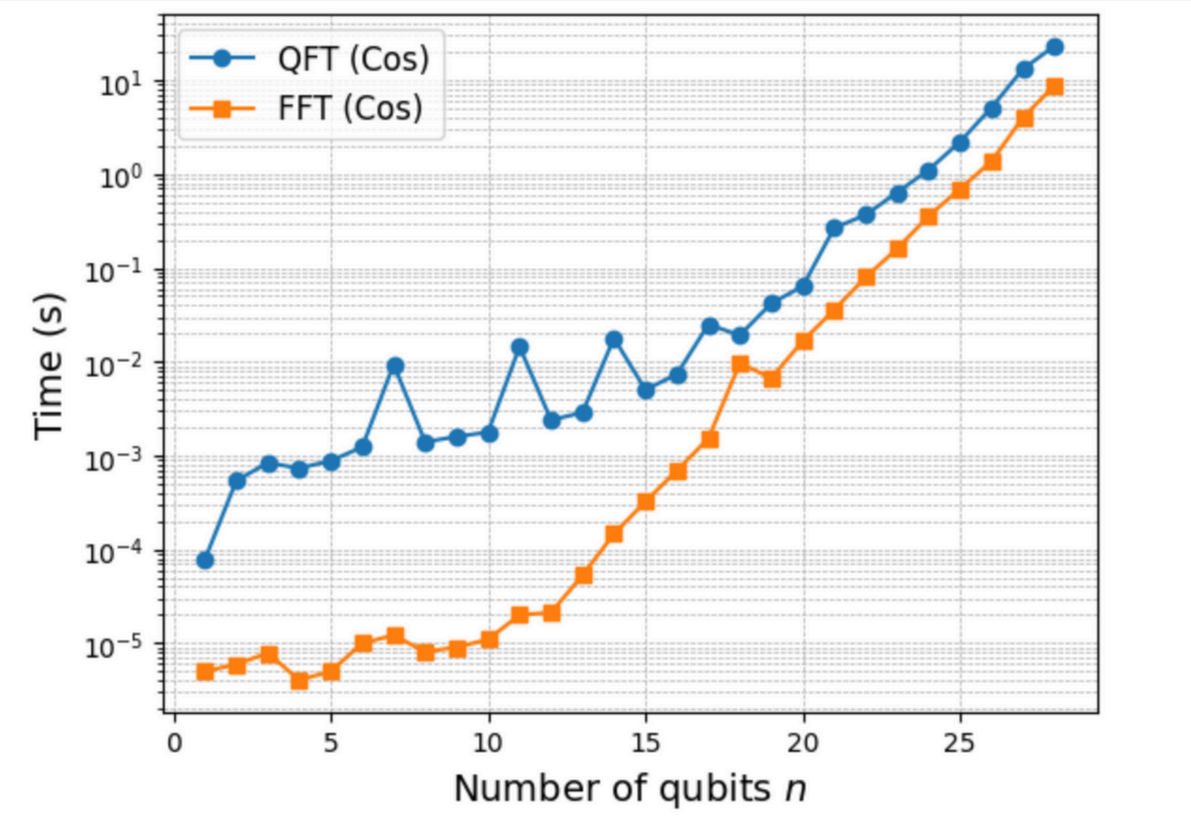
GRAPH



Time to convert the array to MPS is included in this graph

Cutoff Value set : 10^(-12)

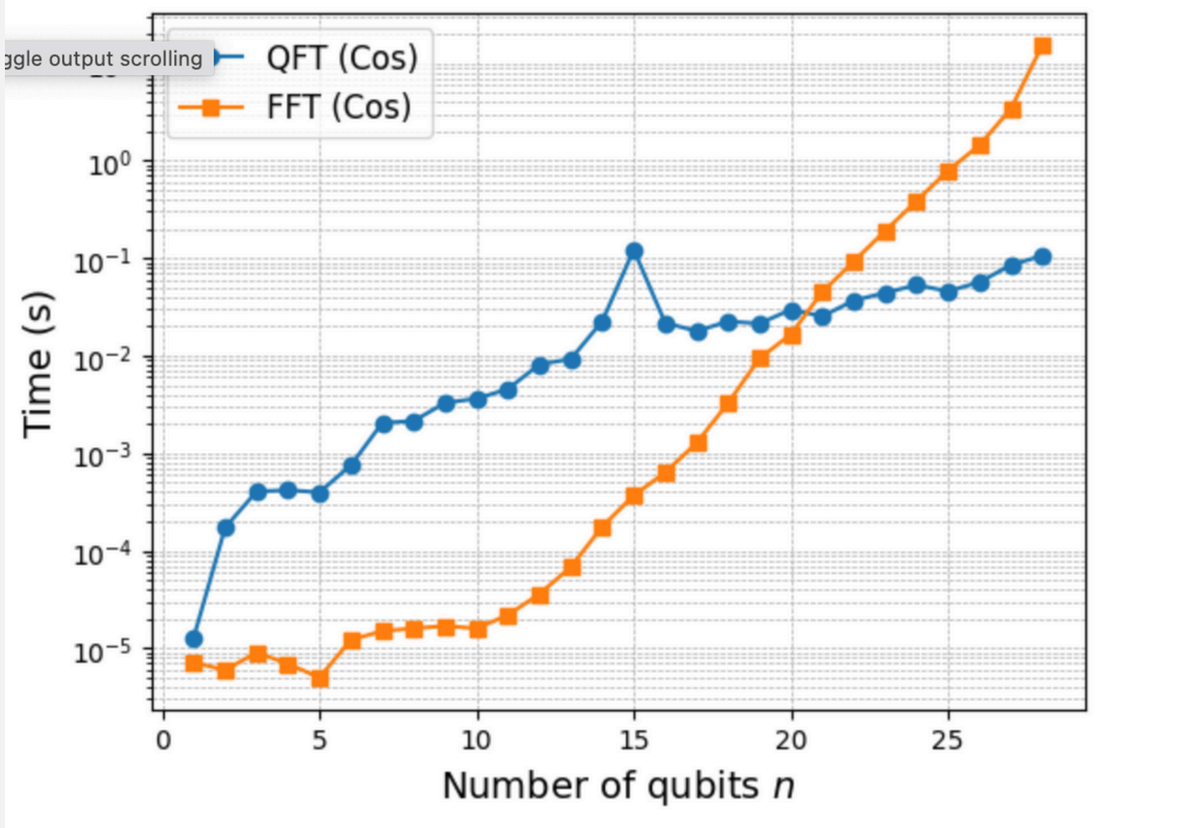
GRAPH



Time to convert the array to MPS is included in this graph

Cutoff Value set : 10^(-18)

COMPARISION WITH FFT



Time to convert to MPS is not included

<u>Random Input</u>

The values in the input array are random and in the range [0,1)

Cutoff = 10^(-18)

Not expected!

RUNTIME

Runtime still depends on the below things

- Type of Data: QFT is faster than FFT only when there is some kind of structure in the data. If the data is random, then FFT is always better than QFT.

- Cutoff value when we are converting it to MPS: If the cutoff is set low, then it takes more time to calculate the MPS. We should set the cutoff keeping in the accuracy and time taken for optimal results

FUTURE WORK

- Coming up with a faster way to convert input data into MPS
- Using this algorithm instead of FFT and gaining significant time advantage in simulations, signal processing etc
- Use this method to simulate other quantum algorithms which can’t be done classically, like Grovers algorithm (quantum-inspired classical)
- Modifying classical algorithms to work with MPS structures instead of arrays or tensors to make efficient calculations using tensor networks

REFERENCES

- PRX QUANTUM 4, 040318 (2023) [Research Paper]

Quantum Fourier Transform Has Small Entanglement
Jielun Chen (陈捷伦) ,1,2,* E.M. Stoudenmire ,3 and Steven R. White 1
Department of Physics and Astronomy, University of California, Irvine, California 92697-4575, USA
Department of Physics, California Institute of Technology, Pasadena, California 91125, USA
Center for Computational Quantum Physics, Flatiron Institute, 162 Fifth Avenue, New York, New York 10010, USA

- https://itensor.github.io/ITensors.jl/dev/MPSandMPO.html

MPO and MPS code documentation here

- https://www.youtube.com/watch?v=fq3_7vBcj3g

Tensor networks explaination images are taken from here

THANK YOU!