Quantum Fourier Transform

samar.aseeri

July 2024

Quantum Fourier Transform for N Qubits

The Quantum Fourier Transform (QFT) for N qubits is defined as: $QFT_N = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{\frac{2\pi i}{N} jk} |k\rangle$ where $|k\rangle$ represents the basis states.

Quantum State Transformation

Assuming $N = 2^n$ where n is an integer, the transformation can be expressed as: $|j_1 \dots j_n\rangle \rightarrow$
$\frac{1}{2^{n/2}} \left((|0\rangle + e^{2\pi i 0.j_n}|1\rangle)(|0\rangle + e^{2\pi i 0.j_{n-1}j_n}|1\rangle) \dots (|0\rangle + e^{2\pi i 0.j_1 j_2 \dots j_n}|1\rangle) \right)$

wherein $0.j_1 j_2 \dots j_n = \frac{j_1}{2^1} + \frac{j_2}{2^2} + \dots + \frac{j_n}{2^n}$

Circuits for the QFT

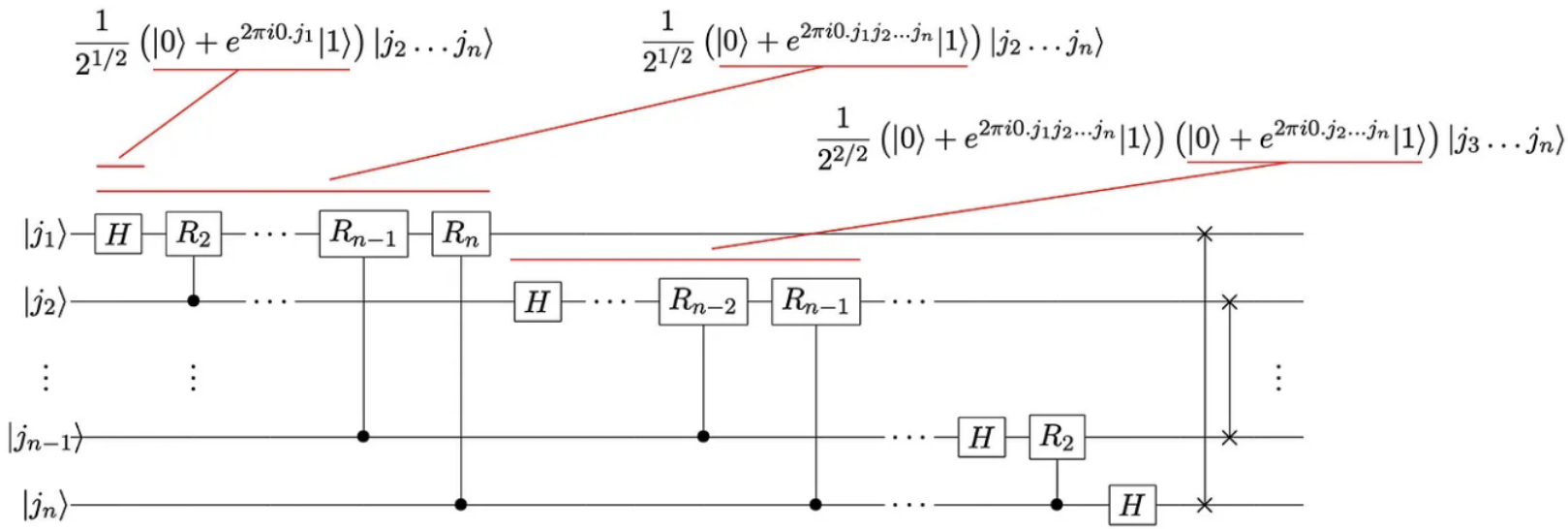
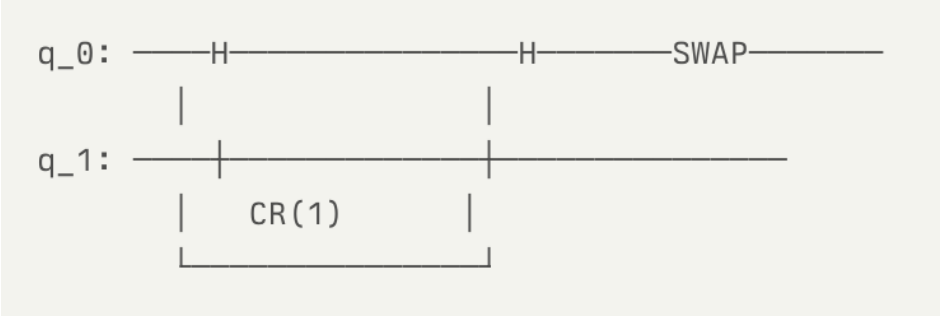


Figure: QFT can be implemented with a series of the controlled-R gates

Example: Two Qubits



For two qubits, the states can be expressed as:

$$|\psi_1\rangle = |q_0\rangle \left(|0\rangle + e^{\frac{2\pi i q_1}{2}}|1\rangle\right)$$

$$|\psi_2\rangle = |q_0\rangle \left(|0\rangle + e^{\frac{2\pi i q_0}{4}} e^{\frac{2\pi i q_1}{2}}|1\rangle\right)$$

$$|\psi_3\rangle = \left(|0\rangle + e^{\frac{2\pi i q_0}{2}}|1\rangle\right) \left(|0\rangle + e^{\frac{2\pi i q_0}{4}} e^{\frac{2\pi i q_1}{2}}|1\rangle\right)$$

Example: Follow up

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, R = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{2\pi i}{2^k}} \end{bmatrix}$$

- Apply Hadamard gate (H) to $|q_0\rangle$.
- Apply controlled phase rotation (CR) between $|q_0\rangle$ and $|q_1\rangle$.
- Apply Hadamard gate (H) to $|q_1\rangle$.
- Swap the qubits.

Inverse Quantum Fourier Transform

The inverse QFT can be applied to the state $|4\rangle$ to complete the circuit: $QFT^{-1}|4\rangle \rightarrow |4\rangle$ When measured, this results in the value 4, confirming that $QFT(4) = |4\rangle$.

Quantum Fourier Transform Calculation

```
from qiskit import QuantumCircuit, Aer, transpile, assemble
from qiskit.visualization import plot_histogram
import numpy as np

def qft(circuit, n):
  for j in range(n):
    circuit.h(j)
    for k in range(j+1, n):
      circuit.cp(np.pi/2**(k-j), k, j)
  circuit.barrier()

n = 3
qc = QuantumCircuit(n)
qft(qc, n)
qc.measure_all()

simulator = Aer.get_backend('qasm_simulator')
compiled_circuit = transpile(qc, simulator)
qobj = assemble(compiled_circuit)
result = simulator.run(qobj).result()

counts = result.get_counts()
print(counts)
plot_histogram(counts)
```

Figure: This code demonstrates how to implement the QFT algorithm on a quantum circuit using Qiskit.

Quantum Fourier Transform Calculation - Output

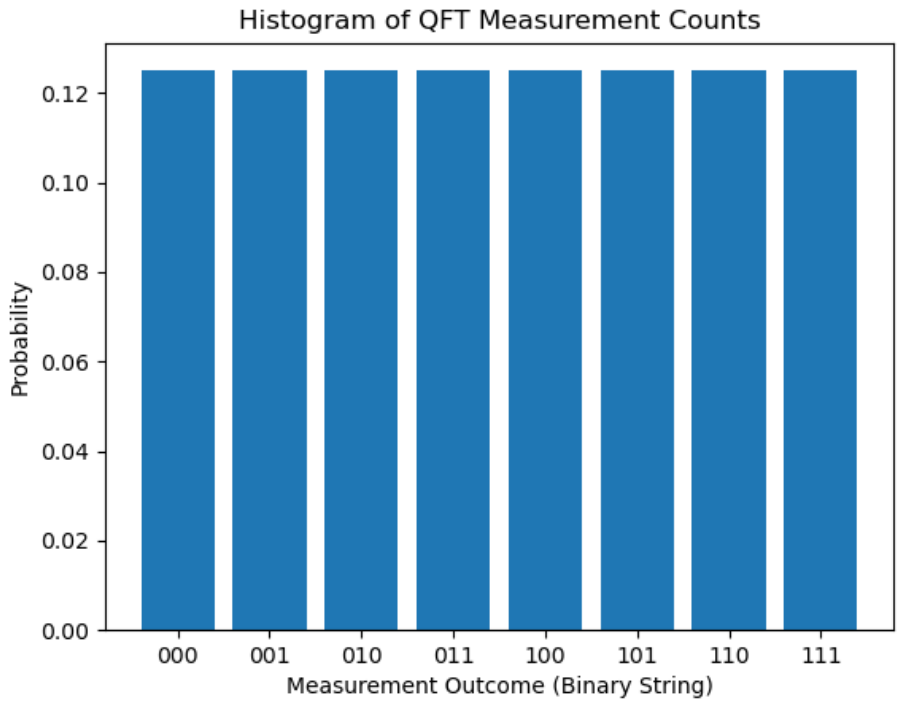


Figure: Code simulates a 3-qubit quantum circuit with QFT. Output histogram shows equal probability for all measurement outcomes due to QFT's property of creating uniform superposition. This is a fundamental step in many quantum algorithms.

Analogues to Classical Fourier Transform

- **Input:** Similar to the classical Fourier Transform, the quantum Fourier transform takes an input signal composed of quantum states.
- **Output:** The quantum Fourier transform produces a quantum superposition of frequency components, providing information about the amplitudes and phases of the input states.
- **Conservation of Energy:** The quantum Fourier transform conserves energy, just like its classical counterpart.

Accuracy-Time Tradeoffs

- The accuracy of the quantum Fourier transform depends on the number of qubits used and the desired precision. As the number of qubits increases, the accuracy improves.
- **Processing time:** The complexity of the quantum Fourier transform depends on the input size (N). Generally, the larger the input size, the longer it takes to perform the transform.

Comparison with Classical Fourier Transform

- The quantum Fourier transform is not a plug-in replacement for the classical Fourier Transform. It utilizes quantum superposition and entanglement to perform computations that are not possible with classical methods. Quantum computing can potentially offer exponential speedup for certain problems compared to classical computing.

Open-Source Software

- There are several open-source software libraries available for quantum computing, such as Qiskit, Cirq, and Forest. These libraries provide tools and algorithms, including implementations of the quantum Fourier transform.

Unique Aspects of Quantum Fourier Transform

- **Sensitivity to qubit errors:** The quantum Fourier transform, like other quantum algorithms, is susceptible to errors caused by noise and decoherence. Error correction techniques and error mitigation strategies can help address these issues.
- **Implementation methods:** Different approaches can be used to implement the quantum Fourier transform, such as quantum circuits or quantum algorithms based on the quantum Fourier transform.
- **Largest size transformed so far:** No specific information available.

Leading researchers in the field

Umesh Vazirani (1945, Indian-American) is a professor at the University of Berkeley. He is one of the founders of quantum computing, with his paper co-authored in 1993 with his student Ethan Bernstein, Quantum Complexity Theory. He is also the creator of the Quantum Fourier Transform (QFT) algorithm, which was used less than a year later by Peter Shor to create his famous integer factoring algorithm that served as a spur to funding research in quantum computing in the USA. The QFT is a founding algorithm used in many other quantum algorithms.

Conclusion

The Quantum Fourier Transform is a crucial component in quantum computing, enabling efficient algorithms for problems such as factoring and discrete logarithms. Understanding its mathematical foundation is essential for leveraging quantum algorithms effectively.

References

- McMahon, D. (2008). Quantum Computing Explained. Wiley-IEEE Computer Society Press. ISBN: 978-0-470-09699-4. Available online: https://www.wiley.com/en-ie/Quantum+Computing+Explained-p-9780470096994
- Colibritd Quantum. (n.d.). Getting to Know Quantum Fourier Transform. Retrieved from https://medium.com/colibritd-quantum/getting-to-know-quantum-fourier-transform-ae60b23e58f4
- Coppersmith, D. (1994). An Approximate Fourier Transform Useful in Quantum Factoring. IBM Research Report.
- Aseeri, S. A. (2025). A Hybrid Quantum–Classical Spectral Solver for Nonlinear Differential Equations. *Algorithms*, 18(11), 678.

References

- Aseeri, S. (2025). Distributed Memory Fast Fourier Transforms. *IEEE*.
- Leu, B., Aseeri, S., & Muite, B. K. (2021). A comparison of parallel profiling tools for programs utilizing the FFT. *International Conference on High Performance Computing in Asia-Pacific Region*.
- Muite, B. K., & Aseeri, S. (2019). Benchmarking solvers for the one dimensional cubic nonlinear klein gordon equation on a single core. *International Symposium on Benchmarking, Measuring and Optimization*, 172-184.
- Aseeri, S., Muite, B. K., & Takahashi, D. (2019). Data for figures in "Reproducibility in Benchmarking Parallel Fast Fourier Transform based Applications".