Comparison of DFT, FFT, and QFT

Ryan So

12/10/2025

Digital Signal Processing

Contents

DFT

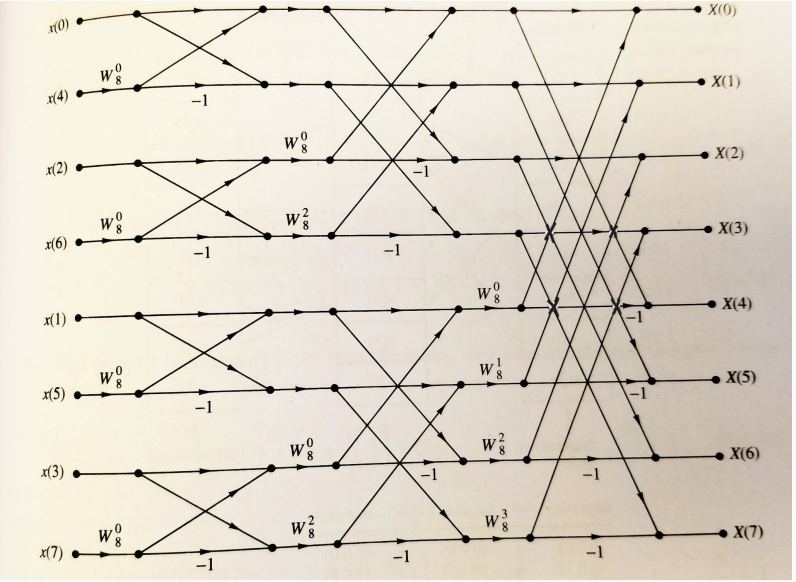
Given a discrete signal, transform the signal into discrete frequencies.

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-jk\frac{2\pi}{N}n}$$

```
dft_result = []
for k in range(self.N):
    sum_val = 0
    for n in range(self.N):
        angle = 2j * np.pi * k * n / self.N
        sum_val += self.signal[n] * np.exp(-angle)
    magnitude = np.abs(sum_val)
    if np.isclose(magnitude, 0):
        magnitude = 0
    dft_result.append(float(magnitude))
return dft_result
```

Radix-2 FFT

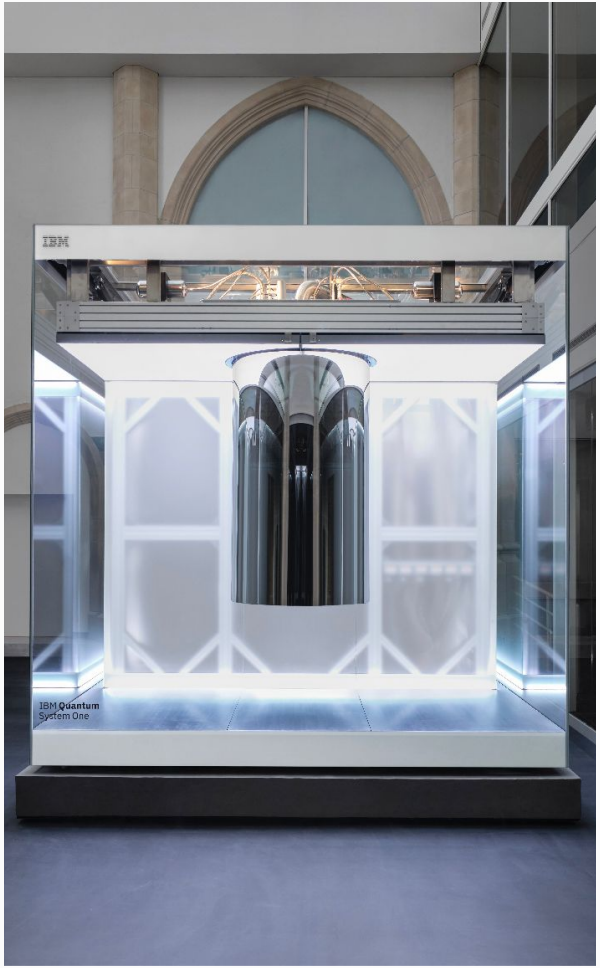
A derivation of the DFT by dividing the samples, performing butterfly operations, and conquering them to represent the input signal as it's frequencies.




```python
def _fft_recursive(self, x: np.ndarray) -> np.ndarray:
    N = x.shape[0]
    if N == 1:
        return x

    # Even and odd parts
    X_even = self._fft_recursive(x[0::2])
    X_odd  = self._fft_recursive(x[1::2])

    # Twiddle factors for this stage (depend on current N, not total length)
    k = np.arange(N // 2)
    twiddle = np.exp(-2j * np.pi * k / N) * X_odd

    # Combine (this returns outputs in standard order — no bit-reversal needed)
    return np.concatenate([X_even + twiddle,
                           X_even - twiddle])

def compute_fft(self) -> list:
    X = self._fft_recursive(self.signal)
    mags = np.abs(X)

    # Optional: zero-out tiny numerical noise
    mags[np.isclose(mags, 0)] = 0.0

    # Return as plain Python list, like your original function
    return mags.tolist()
```

Quantum Computing

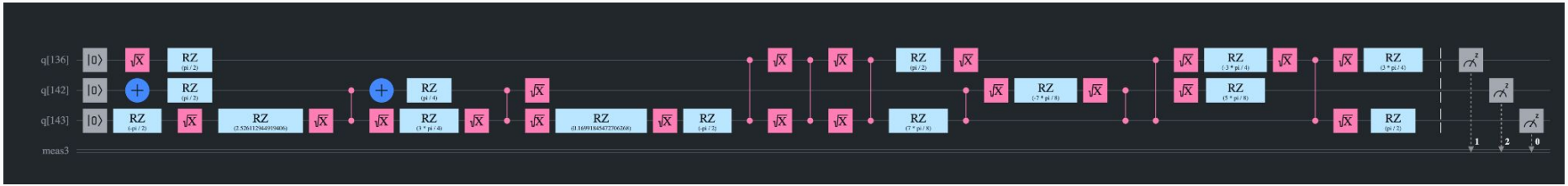
1. RPI's quantum computer works by perform gates on qubits, similar to how we perform and, or, and not gates on bits.
2. All gates are rotations of qubits in the x, y, and z direction.
3. Measurements are taken to collapse the qubits to a 0 or 1. The act of collapsing is a probabilistic measurement which becomes apparent later.
4. Measuring a measured state will return the same result.
5. Depending on the number of "shots", this process is done that many times.



QFT

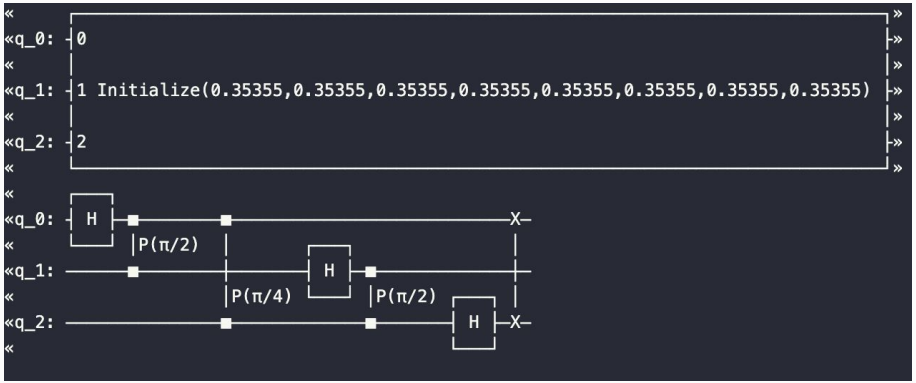
1. Initialize the qubits with given input by normalizing input and store the amplitude.
2. Multiply each amplitude with a phase rotation. (twiddle factor)
3. Bit reverse as this returns the inverse of what we want.

```
for j in range(n):
    k = qubits[j]
    self.qc.h(k)
    # Controlled phase rotations
    for m in range(j + 1, n):
        l = qubits[m]
        angle = pi / (2 ** (m - j))
        self.qc.cp(angle, l, k)

# Bit-reversal via swaps
for i in range(n // 2):
    self.qc.swap(qubits[i], qubits[n - i - 1])
```



Pros and Cons

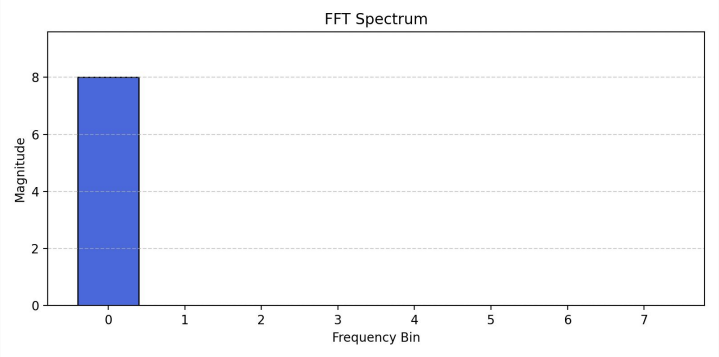
Pros	Cons
• Algorithm is much faster as samples are stored as amplitudes. • Memory efficient.	• Big overhead with amplitude encoding. • Noise and decoherence destroys the structure. • Energy heavy. • Depends on System One's server response



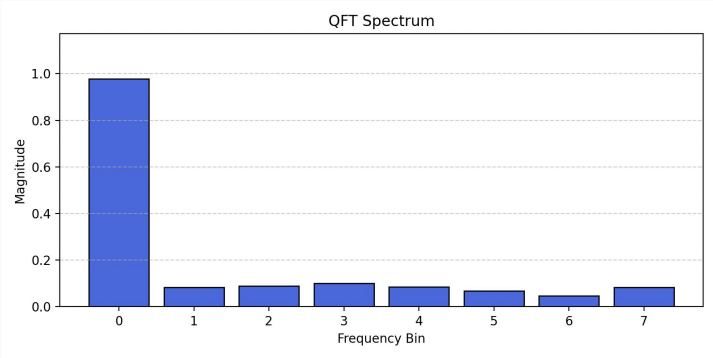
Comparison Between the 3

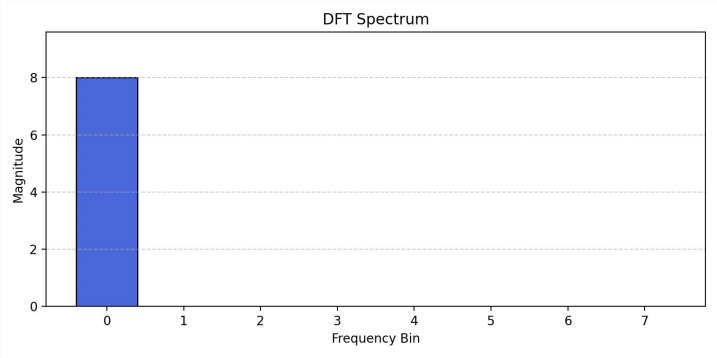
Methods	DFT	Radix-2 FFT	QFT
Time complexity	O(n^2)	nlog(n)	(log(n))^2
Requirements	None	2^n samples	2^n samples
I/O Bound?	No	No	Yes
Simple?	1	2	3

Demo with simple signal

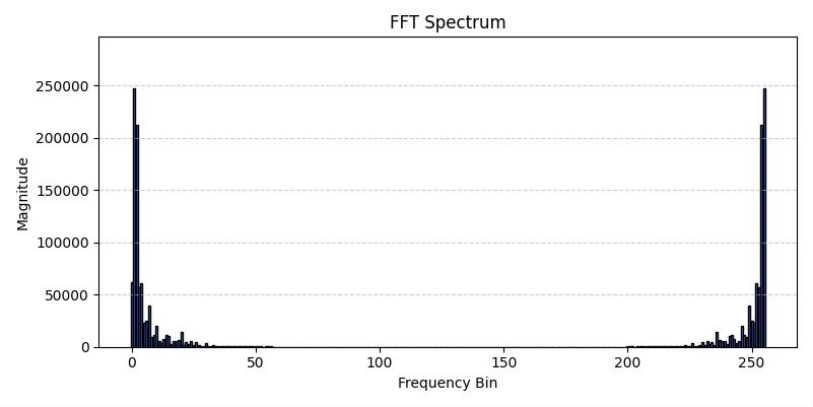


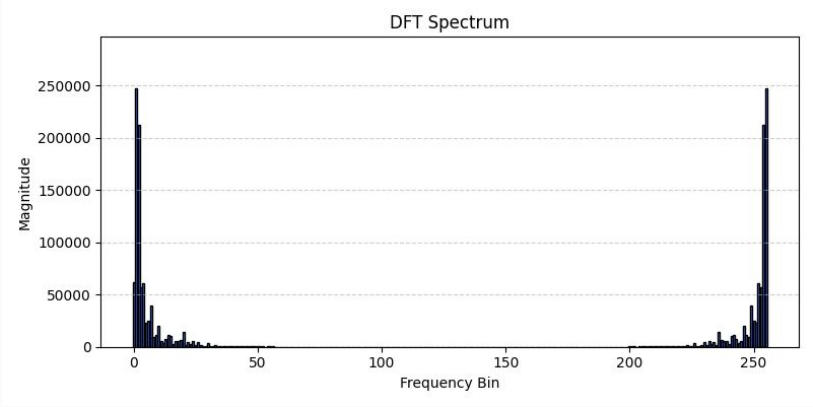
$x[n] = [\underline{1}, 1, 1, 1, 1, 1, 1, 1]$

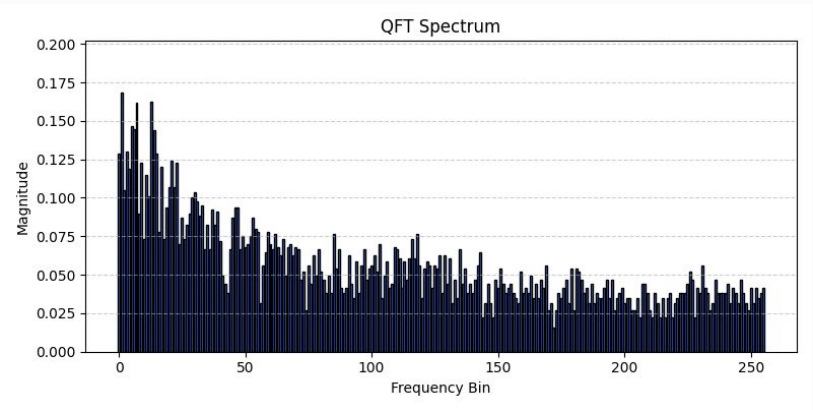




Demo with C major chord





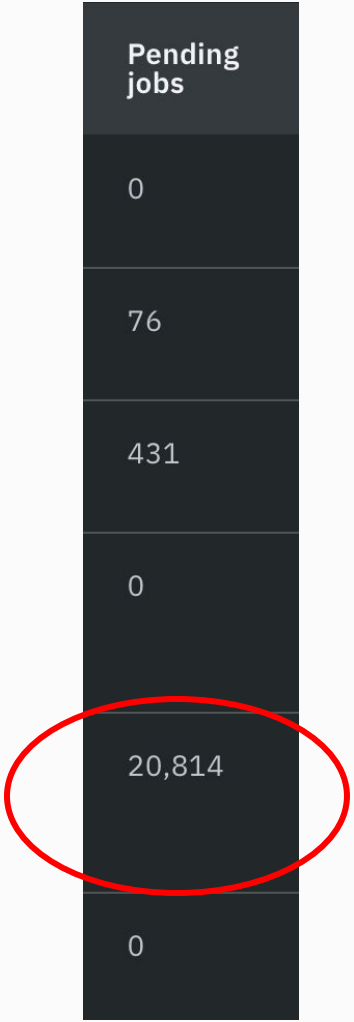


FFT computation time: 0.0007379055023193359 seconds
DFT computation time: 0.03589296340942383 seconds
QFT computation time: 5.973768949508667 seconds

Comparison of DFT, FFT, and QFT

Conclusion

Although QFT and quantum computing in general promises a lot especially when it comes to runtime speedup, it falls short in a lot of areas. From noisy systems to long queue times, quantum computing needs a lot to go right and very little to go wrong. If one day we find a way to circumvent all these hurdles, quantum computing could be a step in the next big jump in technology. For now, it serves as a researcher's best friend.



Sources

Introduction to Qiskit | IBM Quantum Documentation," *IBM Quantum Documentation*, 2017. https://quantum.cloud.ibm.com/docs/en/guides
Oakes, K. (2025, September 15). *Lec6-DFT*. Piazza. https://piazza.com/class_profile/get_resource/mdenb3m6wean7/mfljlrzwpdt44m
Oakes, K. (2025, October 9). *Lec-13-FFT-Radix-2*. Pizza. https://piazza.com/class_profile/get_resource/mdenb3m6wean7/mgjvtpjcvur56g
T. G. Wong, "Quantum Fourier Transform," in *Introduction to Classical and Quantum Computing*, Chapter 7.7, 2023. [Online]. Available: https://cs.uwaterloo.ca/~watrous/QC-notes/. Accessed: Dec. 10e, 2025.

My github for this project:
https://github.com/ryanso128/DSP_Final_Project

Questions?