The Fourier Transform: From Continuous to Discrete and the FFT Algorithm

Your Name

July 28, 2025

Contents

1 Introduction

The Fourier Transform is one of the most powerful mathematical tools in signal processing, physics, and engineering. It allows us to decompose any signal into its constituent frequency components, revealing the "hidden" periodicities within seemingly complex waveforms. This document provides a comprehensive overview of the Fourier Transform, starting from the continuous case, moving to the discrete version, and finally exploring the Fast Fourier Transform (FFT) algorithm.

2 The Continuous Fourier Transform

2.1 Definition and Intuition

The Continuous Fourier Transform (CFT) of a function $f(t)$ is defined as:

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt \tag{1}$$

where:

- $F(\omega)$ is the frequency domain representation
- $f(t)$ is the time domain signal
- ω is the angular frequency (radians per second)
- i is the imaginary unit

The inverse Fourier Transform recovers the original signal:

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{i\omega t} d\omega \tag{2}$$

2.2 Physical Interpretation

The Fourier Transform essentially asks: "How much of each frequency ω is present in the signal $f(t)$?" The complex exponential $e^{-i\omega t} = \cos(\omega t) - i\sin(\omega t)$ acts as a "probe" that correlates the signal with sinusoidal components at frequency ω.

When we compute $F(\omega)$, we obtain:

- **Magnitude** $|F(\omega)|$: represents the amplitude of frequency component ω
- **Phase** $\arg(F(\omega))$: represents the phase shift of that frequency component

2.3 Key Properties

2.3.1 Linearity

If $\mathcal{F}\{f(t)\} = F(\omega)$ and $\mathcal{F}\{g(t)\} = G(\omega)$, then:

$$\mathcal{F}\{af(t) + bg(t)\} = aF(\omega) + bG(\omega) \tag{3}$$

2.3.2 Time Shifting

$$\mathcal{F}\{f(t - t_0)\} = F(\omega) e^{-i\omega t_0} \tag{4}$$

2.3.3 Frequency Shifting

$$\mathcal{F}\{f(t)e^{i\omega_0 t}\} = F(\omega - \omega_0) \tag{5}$$

2.3.4 Convolution Theorem

$$\mathcal{F}\{f(t) * g(t)\} = F(\omega) \cdot G(\omega) \tag{6}$$

where $*$ denotes convolution.

3 The Discrete Fourier Transform (DFT)

3.1 Motivation and Definition

In practice, we work with discrete, finite-length signals. The Discrete Fourier Transform (DFT) is the discrete analog of the continuous Fourier Transform, designed for sequences of N samples.

For a discrete signal $x[n]$ where $n = 0, 1, 2, \ldots, N-1$, the DFT is defined as:

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-i2\pi kn/N} \tag{7}$$

where $k = 0, 1, 2, \ldots, N-1$ are the frequency bin indices.

The inverse DFT (IDFT) is:

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{i2\pi kn/N} \tag{8}$$

3.2 Frequency Interpretation

The DFT computes the spectrum at N equally spaced frequency points:

$$\omega_k = \frac{2\pi k}{N}, \quad k = 0, 1, \ldots, N-1 \tag{9}$$

If the sampling rate is f_s, then the actual frequencies are:

$$f_k = \frac{kf_s}{N}, \quad k = 0, 1, \ldots, N-1 \tag{10}$$

3.3 Key Differences from CFT

- **Periodicity**: Both $x[n]$ and $X[k]$ are periodic with period N
- **Finite**: We only consider N samples in both domains
- **Sampling**: The frequency domain is also sampled (quantized)

3.4 Matrix Representation

The DFT can be written as a matrix multiplication:

$$\mathbf{X} = \mathbf{W}_N\mathbf{x} \tag{11}$$

where $\mathbf{W}_N$ is the $N \times N$ DFT matrix with elements:

$$[\mathbf{W}_N]_{k,n} = e^{-i2\pi kn/N} \tag{12}$$

Direct computation requires $O(N^2)$ complex multiplications.

4 The Fast Fourier Transform (FFT)

4.1 The Computational Challenge

Computing the DFT directly requires N^2 complex multiplications and additions. For large N (e.g., $N = 1024$ or $N = 4096$), this becomes computationally expensive. The FFT algorithm reduces this complexity from $O(N^2)$ to $O(N \log N)$.

4.2 The Cooley-Tukey Algorithm

The most common FFT algorithm is the Cooley-Tukey algorithm, which uses a divide-and-conquer approach. We'll focus on the radix-2 decimation-in-time version, which requires N to be a power of 2.

4.2.1 Key Insight: Divide and Conquer

The main idea is to recursively break down the DFT of length N into two DFTs of length $N/2$:

$$\begin{aligned} X[k] &= \sum_{n=0}^{N-1} x[n] W_N^{kn} && (13) \\ &= \sum_{n \text{ even}} x[n] W_N^{kn} + \sum_{n \text{ odd}} x[n] W_N^{kn} && (14) \end{aligned}$$

Let $n = 2m$ for even indices and $n = 2m + 1$ for odd indices:

$$\begin{aligned} X[k] &= \sum_{m=0}^{N/2-1} x[2m] W_N^{2km} + \sum_{m=0}^{N/2-1} x[2m+1] W_N^{k(2m+1)} && (15) \\ &= \sum_{m=0}^{N/2-1} x[2m] W_{N/2}^{km} + W_N^k \sum_{m=0}^{N/2-1} x[2m+1] W_{N/2}^{km} && (16) \end{aligned}$$

where we used the identity $W_N^{2km} = W_{N/2}^{km}$.
This gives us:

$$X[k] = E[k] + W_N^k \cdot O[k] \tag{17}$$

where:

- $E[k]$ is the DFT of the even-indexed samples
- $O[k]$ is the DFT of the odd-indexed samples
- $W_N^k = e^{-i2\pi k/N}$ is the "twiddle factor"

4.2.2 Periodicity and Symmetry

Since $E[k]$ and $O[k]$ are periodic with period $N/2$, we have:

$$\begin{aligned} X[k] &= E[k] + W_N^k \cdot O[k], \quad k = 0, 1, \ldots, N/2 - 1 && (18) \\ X[k + N/2] &= E[k] - W_N^k \cdot O[k], \quad k = 0, 1, \ldots, N/2 - 1 && (19) \end{aligned}$$

This symmetry allows us to compute all N output points using only $N/2$ evaluations of each sub-DFT.

Algorithm 1 Cooley-Tukey FFT (Radix-2, Decimation-in-Time)

```
Require: $x[0..N-1]$ where $N = 2^m$
Ensure: $X[0..N-1]$ (DFT of $x$)
 1: if $N = 1$ then
 2:     return $x[0]$
 3: end if
 4: $x_{even} \leftarrow [x[0], x[2], x[4], \ldots, x[N-2]]$
 5: $x_{odd} \leftarrow [x[1], x[3], x[5], \ldots, x[N-1]]$
 6: $E \leftarrow \text{FFT}(x_{even})$ {Recursive call}
 7: $O \leftarrow \text{FFT}(x_{odd})$ {Recursive call}
 8: for $k = 0$ to $N/2 - 1$ do
 9:     $t \leftarrow e^{-i2\pi k/N} \cdot O[k]$ {Twiddle factor multiplication}
10:     $X[k] \leftarrow E[k] + t$
11:     $X[k + N/2] \leftarrow E[k] - t$
12: end for
13: return $X$
```

4.3 FFT Algorithm Pseudocode

4.4 Computational Complexity

The FFT reduces the computational complexity from $O(N^2)$ to $O(N \log N)$:

- **Levels**: $\log_2 N$ levels of recursion
- **Operations per level**: N complex multiplications and additions
- **Total**: $N \log_2 N$ operations

For $N = 1024$:

- Direct DFT: $1024^2 = 1,048,576$ operations
- FFT: $1024 \times 10 = 10,240$ operations
- Speedup: $\sim 100\times$

4.5 Bit-Reversal and In-Place Implementation

The recursive formulation can be converted to an iterative, in-place algorithm using bit-reversal ordering. The input array is first rearranged so that element at position n is moved to position obtained by reversing the binary representation of n.

For example, with $N = 8$:

Decimal	Binary	Bit-Reversed	New Position
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

5 Applications and Practical Considerations

5.1 Common Applications

- **Digital Signal Processing**: Filtering, spectral analysis
- **Image Processing**: Compression (JPEG), enhancement
- **Audio Processing**: Music analysis, noise reduction
- **Communications**: Modulation schemes (OFDM)
- **Scientific Computing**: Solving PDEs, convolution

5.2 Windowing

When applying the DFT to finite-length signals, edge effects can cause spectral leakage. Windowing functions (Hamming, Hanning, Blackman) are applied to mitigate these effects:

$$x_w[n] = x[n] \cdot w[n] \tag{20}$$

where $w[n]$ is the window function.

5.3 Zero-Padding

Zero-padding increases frequency resolution and can improve the appearance of the spectrum:

$$x_{padded}[n] = \begin{cases} x[n] & \text{if } 0 \leq n < N \\ 0 & \text{if } N \leq n < M \end{cases} \tag{21}$$

where $M > N$ is the new length.

6 Conclusion

The Fourier Transform provides a fundamental bridge between time and frequency domains, enabling us to analyze and manipulate signals in ways that would be impossible in a single domain. The progression from the continuous Fourier Transform to the discrete version, and finally to the computationally efficient FFT algorithm, demonstrates how mathematical theory translates into practical computational tools.

The FFT's $O(N \log N)$ complexity has made real-time spectral analysis possible in countless applications, from digital audio processing to medical imaging. Understanding these concepts provides the foundation for advanced signal processing techniques and helps in choosing appropriate tools for specific applications.

7 References

- Cooley, J. W., & Tukey, J. W. (1965). An algorithm for the machine calculation of complex Fourier series. Mathematics of Computation, 19(90), 297-301.
- Oppenheim, A. V., & Schafer, R. W. (2009). Discrete-time signal processing. Pearson.
- Brigham, E. O. (1988). The fast Fourier transform and its applications. Prentice Hall.