FFT

Fast Fourier Transform (FFT) Derivation

Discrete Fourier Transform (DFT)

The Discrete Fourier Transform (DFT) is defined as:

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-j\frac{2\pi}{N}kn}$$

where:

- $X(k)$ is the frequency domain signal, the k-th frequency component
- $x(n)$ is the time domain signal, the n-th time sample
- N is the total number of samples in the signal
- j is the imaginary unit

Fast Fourier Transform (FFT)

The Fast Fourier Transform (FFT) is an efficient algorithm for computing the DFT.

Divide-and-Conquer Approach

Assume we have a signal of length N. The DFT can be broken down into smaller DFTs using the divide-and-conquer approach. If N is a power of 2 ($N = 2^m$), we can split the DFT into two smaller DFTs of length $N/2$.

1. Separate the even and odd indexed samples:

$$x(n) = x(2r) + x(2r+1)$$

2. Compute the DFT of the even-indexed samples ($X_e(k)$) and odd-indexed samples ($X_o(k)$):

$$X_e(k) = \sum_{r=0}^{N/2-1} x(2r) \cdot e^{-j\frac{2\pi}{N}kr}$$

$$X_o(k) = \sum_{r=0}^{N/2-1} x(2r+1) \cdot e^{-j\frac{2\pi}{N}kr}$$

3. Combine the results to get the final DFT:

$$X(k) = X_e(k) + e^{-j\frac{2\pi}{N}k} \cdot X_o(k)$$

$$X(k+N/2) = X_e(k) - e^{-j\frac{2\pi}{N}k} \cdot X_o(k)$$

The exponential term $e^{-j\frac{2\pi}{N}k}$ is known as the "twiddle factor."

Full FFT Algorithm

The full FFT algorithm can be implemented using a recursive approach. The base case is when $N = 1$, where the DFT is simply the value of the single sample. For larger N, we recursively apply the above steps until we reach the base case.