FFT Algorithms Documentation

Meriem Si

1 Cooley-Tukey FFT Algorithm (Radix-2)

1.1 Usage

The Cooley-Tukey FFT Algorithm is predominantly used for efficiently computing the Discrete Fourier Transform (DFT) of sequences where the length is a power of two. It's widely applied in digital signal processing, audio and image compression, and solving numerical problems involving Fourier transforms.

1.2 Theory

The Radix-2 algorithm is a divide-and-conquer approach that recursively breaks down a DFT of size N into smaller DFTs. This method exploits the periodicity and symmetry properties of the DFT.

1.3 Importance

- Complexity Reduction: It reduces the computational complexity from $O(N^2)$ to $O(N \log N)$, making it significantly faster, especially for large N.
- Efficiency: The algorithm's recursive nature allows for substantial savings in computation, particularly for sequences with lengths that are powers of two.

1.4 Equations and Usage

- The DFT is defined as: $X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-\frac{i2\pi}{N}kn}$
- In Radix-2, the DFT is divided into two smaller DFTs: one for even-indexed and one for odd-indexed elements.
- The final DFT is obtained by combining these smaller DFTs.

2 Bluestein's FFT Algorithm (Chirp Z-Transform)

2.1 Usage

Bluestein's FFT Algorithm is used for computing the DFT of sequences of any length, not limited to powers of two. It is particularly useful in applications where the sequence length is prime or does not have an efficient FFT implementation.

2.2 Theory

Bluestein's algorithm, or the Chirp Z-Transform, transforms a DFT into a convolution problem, which can then be efficiently solved using an FFT of a convenient size.

2.3 Importance

- Flexibility: It can compute the DFT for any sequence length, providing greater flexibility than algorithms like Radix-2.
- General Applicability: This method broadens the applicability of FFT to more diverse real-world problems.

2.4 Equations and Usage

- The algorithm uses a "chirp" signal: $w[n] = e^{-\frac{i\pi}{N}n^2}$
- The DFT is computed by convolving the input sequence with the chirp signal, performing an FFT, and then multiplying by the inverse chirp signal.
- Chirp Z-Transform can be expressed as a series of multiplications and an FFT, allowing efficient computation of the DFT.

3 Importance of Optimization and Parallelism

3.1 Optimization

- Performance: Optimizing FFT algorithms is crucial for performance, especially in applications requiring real-time processing.
- Resource Efficiency: Efficient algorithms reduce computational resource requirements, making them suitable for systems with limited processing power.

3.2 Parallelism as a Solution

- Speedup: Parallelism allows FFT computations to be divided and processed simultaneously, significantly speeding up the overall computation.
- Scalability: Parallel algorithms can be scaled across multiple processors or cores, making them ideal for large-scale and high-performance computing environments.
- Real-World Applicability: Many real-world applications, such as image and signal processing, benefit from the speed and efficiency gains offered by parallel FFT computations.