DFT and FFT Algorithms

Exploring the Core of Signal Processing

Presentation by Mohammad Mohammadi (402208592)
School of Mathematics - Sharif University of Technology
Course instructed by Dr. Alireza Zarei

Abstract
This report delves into the world of Fourier Transforms, with a particular focus on the Discrete Fourier Transform (DFT) and its faster counterpart, the Fast Fourier Transform (FFT). These mathematical techniques play a fundamental role in signal analysis, providing invaluable insights across various domains, from telecommunications to medical imaging.

Table of Contents

1. Introduction

Fourier Transforms: Bridging the Time-Frequency Gap

Fourier Transforms are a cornerstone of signal processing, offering a unique perspective into the frequency components of a signal. By decomposing signals into constituent frequencies, they reveal crucial information for a wide range of applications.

The Spectrum of Signal Analysis

Understanding Fourier Transforms is akin to unlocking a spectrum—an array of frequencies that compose a signal. This report navigates through the intricacies of Discrete Fourier Transform (DFT) and Fast Fourier Transform (FFT), uncovering their profound impact on technology and science.

2. DFT vs. FFT: Unraveling the Efficiency

The DFT Challenge

DFT, the foundational technique, computes the frequency components of a signal but suffers from O(N*2) complexity, making it computationally demanding for larger datasets.

Enter FFT: A Computational Revolution

FFT steps in as a computational hero, introducing a transformative algorithm that dramatically reduces the complexity to O(NlogN). This efficiency empowers real-time data analysis and enables the processing of extensive datasets.

FFT Algorithm: Divide and Conquer

The Cooley-Tukey algorithm, a key player in FFT, employs a "divide and conquer" strategy to break down the DFT calculation into smaller, manageable parts. This ingenious approach revolutionized signal analysis.

3. Mathematical Deep Dive - DFT

The DFT Formula

The heart of DFT lies in its formula:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N}$$

This equation computes the amplitude and phase of each frequency component in the signal.

Signal Decomposition

DFT decomposes a signal into a spectral landscape, with each frequency component represented by a complex number, offering both amplitude and phase information.

Phase and Amplitude Unveiled

In DFT's output, the magnitude of each complex number represents the amplitude, while its angle, or phase, unveils the phase shift of the corresponding frequency component in the original signal.

4. Mathematical Deep Dive - FFT

FFT Algorithm Details

FFT algorithms, exemplified by the Cooley-Tukey algorithm, efficiently compute DFT by breaking it down into smaller, more manageable parts. This recursive approach optimizes computational efficiency.

Butterfly Operation

At the heart of FFT lies the "butterfly" operation, a specific computation step that combines pairs of complex numbers in a manner that contributes significantly to FFT's computational efficiency.

Sample Decomposition

FFT strategically divides the original dataset into subsets, processes them individually, and then cleverly combines the results to produce the final frequency domain representation—a feat of mathematical ingenuity.

5. Applications and Impact

Telecommunications: The Signal Highway
FFT is the engine behind signal modulation and demodulation, driving data transmission in telecommunications systems, enabling us to communicate seamlessly across the globe.

Audio and Acoustic Engineering: Harmonic Precision
In the realm of audio processing, FFT reigns supreme. It enables noise reduction, equalization, and detailed analysis of audio signals, enriching our listening experiences.

Medical Imaging: Healing Through Insights
Medical imaging, including MRI and CT scans, relies on FFT to transform raw data into detailed images, aiding in diagnosis and treatment planning.

6. Case Study: FFT in Seismic Data Analysis

Seismic Data: Unpredictable Tremors

We explore a case study where FFT plays a pivotal role in analyzing seismic data to predict earthquakes, a real-world application that showcases the algorithm's effectiveness.

Data Analysis Unveiled

The raw seismic waveforms, representing ground movements over time, are processed using FFT to transform them into the frequency domain. This transformation unveils patterns and frequencies critical for understanding seismic activities.

Impact and Outcomes

FFT enables faster and more accurate earthquake predictions by efficiently processing large datasets in real-time. This application is a testament to the algorithm's significance in real-world scenarios.

7. The Future of Fourier Transforms

Emerging Trends
The world of Fourier transforms is evolving. Recent advancements, including novel algorithms and techniques, enhance their capabilities for various applications.

Integration with AI and ML
Integration with artificial intelligence (AI) and machine learning (ML) is on the horizon. Fourier transforms play a crucial role in feature extraction and data analysis, enabling advanced applications in data science and automation.

Potential Future Applications
The possibilities are limitless. From cutting-edge technologies to scientific discoveries, Fourier transforms are poised to continue shaping the future in fields ranging from healthcare to telecommunications.

8. DFT Implementation

DFT in python

```
def DFT(x):
    """
    Function to calculate the
    discrete Fourier Transform
    of a 1D real-valued signal x
    """
    N = len(x)
    n = np.arange(N)
    k = n.reshape((N, 1))
    e = np.exp(-2j * np.pi * k * n / N)
        X = np.dot(e, x)
    return X
```

9. FFT Implementation

Naive FFT in python

```
def naiveFFT(x):
    ''' The naive implementation for comparison '''
    N = x.size
    X = np.ones(N)*(0+0j)

    for k in range(N):
        A = np.ones(N)*(0+0j)
        for n in range(N):
            A[n] = x[n]*np.exp(-np.complex(0, 2*np.pi*k*n/N))
        X[k] = sum(A)

    return X
```

Radix-2 FFT in python

```
def FFT(x):
    ''' Recursive radix-2 FFT'''
    x = np.array(x, dtype=float)
    N = int(x.size)
    # Use the naive version when the size is small enough
    if N <= 8:
        return naiveFFT(x)
    else:
        # Calculate first half of the W veco
        k = np.arange(N//2)
        W = np.exp(-2j*np.pi*k/N)
        evens = FFT(x[::2])
        odds = FFT(x[1::2])
        return np.concatenate([evens + (W * odds), evens - (W * odds)])

    return 0
```

10. Conclusion

Unlocking the Frequency Spectrum

In conclusion, this report has journeyed through the intricate world of Fourier Transforms, unraveling their significance in signal analysis. From the fundamental DFT to the computational prowess of FFT, these algorithms have left an indelible mark on technology and science.

11. Acknowledgments

The author extends gratitude to all those who have contributed to the understanding and development of Fourier Transforms, as well as to the audience for their attention and participation in this exploration.