N-Point Discrete Fourier Transform in $O(N \log_2 N)$ Time Complexity for Any $N > 0$

Saulo Queiroz

Abstract

We present a novel recursive algorithm for computing the N-point Discrete Fourier Transform (DFT) with $O(N \log_2 N)$ time complexity, applicable for any integer $N > 0$. Unlike conventional Fast Fourier Transform (FFT) algorithms that often require zero-padding to the nearest power of two, our method operates directly on arbitrary-length input sequences, eliminating the need for zero-padding. At each recusive stage, the algorithm relies on a novel decomposition structure that performs $2(N-1)$ complex additions and N complex multiplications for typical cases, yielding a total complexity of $T(N) = 2T(N/2) + 10N - 4 = 10N \log_2 N - 4N + K$ real floating point operations (where K is a constant). Although the resulting constants are higher than in classic FFT algorithms – like radix-2 FFT, that performs $5N \log_2 N$ real floating point operations – our algorithm does not need zero-padding, which end up by increasing the true complexity of existing FFTs due to the enlarged signal length.

Index Terms

Discrete Fourier Transform, Lossless Signal Compression, Computational Complexity.

I. Introduction

The Discrete Fourier Transform (DFT) is a fundamental tool in signal processing, communications, and applied mathematics. For an N-point input signal x_n ($n = 0, 1, \ldots, N-1$) the DFT is defined by:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-j2\pi kn/N}, \quad k = 0, 1, \ldots, N-1. \tag{1}$$

The naive evaluation requires $O(N^2)$ complex multiplications and additions for any positive integer N. The landmark $O(N \log_2 N)$ time complexity was popularized by the so-called radix-2 Cooley–Tukey algorithm[1], which also coined the term "Fast Fourier Transform" (FFT). Since then, several other $O(N \log_2 N)$ algorithms—such as the Prime-Factor algorithm, Rader's algorithm, Bluestein's algorithm, and mixed-radix algorithms—have been proposed in the literature. Despite these advances, whether the N-point DFT can be computed in less than $O(N \log_2 N)$ time complexity remains a fascinating open question [2].

A common characteristic of current FFT algorithms is the requirement that N follow specific patterns, such as being a prime length, a power of two, or factorizable into co-prime factors. In practice, this often means that signals must be zero-padded to meet the requirements of the chosen DFT algorithm, resulting in increased computational complexity. Moreover, in scenarios where the growth of spectrum is of interest (like wireless telecommunications), the bandwidth can grow exponentially because of the power of two requisite of some FFTs, like radix-2 [3].

In some cases, the input length restriction is relaxed for the user but is still imposed internally due to the algorithm's structure. This is the case for Bluestein's algorithm, which internally enlarges the signal to perform a convolution. To the best of the authors' knowledge, there is no FFT algorithm that works efficiently without relying on zero-padding, index reordering, or convolutions.

In this work, we propose the first $O(N \log_2 N)$ DFT algorithm free of zero-padding procedures or factorization schemes. Unlike the other FFT algorithms, our approach does not rely on decomposition strategy introduces by the radix-2 FFT. Instead, it employs a procedure that losslessly compresses the N-point signal into two smaller DFTs of $N/2$ points whose DFTs are equivalent to full N-point DFT [4] [5]. The compression requires $O(N)$ complex operations, resulting in an overall of $O(N \log_2 N)$ time complexity. Although the constants involved in the resulting complexity are higher compared to other FFT algorithms, the proposed solution works for any positive integer N, avoiding the complexity penalties caused by zero-padding to fit specific numerical patterns.

We believe our algorithm opens a new avenue of research in implementation, optimization, circuit design, and applications.

Saulo Queiroz (sauloqueiroz@utfpr.edu.br) is with the Academic Department of Informatics of the Federal University of Technology Paraná (UTFPR), Ponta Grossa, PR, Brazil.

Work developed in the context of the UTFPR SISPEQ projects numbers 00356 and 03279.

[1]Please refer to the Wikipedia entry "Cooley–Tukey FFT algorithm" for the history of this algorithm before [1]

II. Algorithm Description

The algorithm proceeds recursively on the input signal $x = \{x_n\}_{n=0}^{N-1}$. Assuming firstly even N, the algorithm works as follows:

1) **Base Case**: If $N <= 3$, compute the DFT
2) **Compression**: Compute the $C = N/2$ point compressed signal $\hat{x}_n$ for $n = 0, \ldots, C-1$ at the expense of $N-1$ complex additions using [4]. Cost= $O(N)$ complex sums.
3) **Time-domain Modulation**: Modulate the input signal to shift the spectrum by one frequency bin. Cost= $O(N)$ multiplications.
4) **Second Compression**: Apply the same compression on the modulated signal, obtaining another C-point sequence $\hat{s}_n$ for $n = 0, \ldots, C-1$
5) **Recursive Calls**: Apply the algorithm recursively on the two compressed signals, getting the FFTs $E = \mathcal{F}(\hat{x})$ and $O = \mathcal{F}(\hat{s})$.
6) **Assignment of recursive calls**: Do $Y[2k] \leftarrow E[k]$ and $Y[2k+1] \leftarrow O[k]$ for $k = 0, 1, \ldots, N/2 - 1$.

For odd N, one can still use the same FFT algorithm outlined above but for the firsts $N-1$ samples, i.e., ignoring the x_{N-1} time sample. To compute the full N-point DFT, just multiply each obtained $(N-1)$-point DFT with the time sample ignored before multiplied by its respective complex exponential. Finally, the $N-1$-th DFT bin can be computed aside at a cost of $O(N)$. Thus, for odd N, the asymptotic dominant term is still $O(N \log_2 N)$.

III. Recurrence Relation

Let $T(N)$ denote the total number of complex operations (additions and multiplications) required by the algorithm.

Each recursive step involves:

- $2(N-1)$ complex additions for the two compressions.
- N complex multiplications for modulation.
- Two recursive calls of size $N/2$.

Thus, the recurrence relation is:

$$T(N) = 2T(N/2) + 2(N-1) + N = 2T(N/2) + 3N - 2 \tag{2}$$

A. Solving the Recurrence

By applying the Master Theorem or recursively expanding:

$$\begin{aligned} T(N) &= 2T(N/2) + O(N) && (3) \\ &= 2^k T(N/2^k) + kO(N) && (4) \\ & && (5) \end{aligned}$$

At $k = \log_2 N$, we reach $T(1) = O(1)$. Therefore, the total complexity is:

$$T(N) = O(N \log_2 N) \tag{6}$$

matching the optimal complexity of the traditional FFT.

IV. Discussion

The proposed algorithm offers an alternative to radix-based FFTs, with the key advantage of working uniformly for all integer $N > 0$, without the need for specific factorization schemes. The trade-off is a slightly higher constant factor due to additional modulation and compression steps.

Potential applications include digital signal processing systems where N is arbitrary, or in hardware implementations that benefit from uniform recursive structure.

V. Conclusion

We introduced a recursive algorithm for computing the N-point Discrete Fourier Transform with $O(N \log_2 N)$ time complexity for any integer $N > 0$. Future work includes empirical performance evaluation, numerical stability analysis, and optimization of the compression and modulation operations.

References

[1] J. Cooley and J. Tukey, “An algorithm for the machine calculation of complex fourier series,” *Mathematics of Computation*, vol. 19, no. 90, pp. 297–301, 1965.

[2] S. V. Lokam, *Complexity Lower Bounds Using Linear Algebra*. Hanover, MA, USA: Now Publishers Inc., 2009.

[3] S. Queiroz, J. P. Vilela, and E. Monteiro, “Is FFT Fast Enough for Beyond 5G Communications? A Throughput-Complexity Analysis for OFDM Signals,” *IEEE Access*, vol. 10, pp. 104 436–104 448, 2022.

[4] S. Queiroz, J. P. Vilela, B. K. K. Ng, C.-T. Lam, and E. Monteiro, “Fast computation of the discrete fourier transform rectangular index coefficients,” 2025. [Online]. Available: https://arxiv.org/abs/2504.12551

[5] S. Queiroz, J. ao P. Vilela, and E. Monteiro, “Fast computation of the discrete fourier transform square index coefficients,” *scheduled for publication in IEEE Signal Processing Magazine (Tips & Tricks)*, 2025.