IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Evaluating Fast Fourier Transform Algorithm In A Finite Field By Comparing It With The Conventional Discrete Fourier Transform

Ramjee Sahu[1*] and CSP Lugun[2]

University Department of Mathematics, Ranchi University Ranchi

Abstract

Fast Fourier Transform (FFTs) are quick calculations, i.e., of low complexity, for the Discrete Fourier Transform (DFT) calculation on a small abelian pack. They are among the most important computations in programming and applied and planning mathematics, particularly for the sign handling and speculation of one- and multi-layered structures. Unambiguous enrolment recipes for the FFT in all situations, which summarise the main FFT-computation on account of Cooley and Tukey and, much earlier, to Gauß, are the crucial capacity. We present a comprehensive and a little bit ingenious set of confirmations for the FFT for cyclic or noncyclic flighty confined abelian gatherings. We are aware that our approach has an educational advantage over conventional ones. Additionally, we demonstrate how the FFT is used to speed up convolution computations and the purported number of made-up modifications across required coefficient rings. We don't examine computations that reduce the multiplicative complexity to the barrier of unquestionably more target direct mixes, which are considered as free in this particular circumstance, nor do we read up the DFT for nonabelian limited get-togethers.

Key words: Finite Field, Discrete Fourier Transform, and Fast Fourier Transform.

1. INTRODUCTION

The essential numerical apparatus accessible now might just be the Fourier Transform [1]. Despite the fact that discrete Fourier Transform (DFT) and quick Fourier changes (FFT) are for the most part viewed as steady exercises, temperamental and incongruous evaluations of the adequacy have been made. A few tests disregard the impacts of blunders in the coefficients, likewise alluded to as the sine/cosine table or "wriggle factors," which might be the significant wellspring of mistake.

A computation is referred to as rapid if it has a low degree of complexity, where the degree of complexity is the number of simple calculation steps required to complete the calculation [2]. Such a stage is of the hatchet $ax + y$ structure in this work and in the majority of PC processors; that is, a, x, y; i.e., it consists of one duplication and one expansion.

A calculation is considered quick assuming there are not many complex advances expected to finish it, where intricacy is the quantity of straightforward advances [3]. Such a phase has the hatchet $ax + y$ structure in this article and in most of PC processors, with the whole numbers a, x, y; i.e., it comprises of one duplication along with one expansion [4].

Man of his word and Sande examine FFT errors where the relative root mean square error is

$$E_{R\,M\,S} = 1.05 \sum_{J=1}^{K} (2P_J)^{3/2} \in. \qquad (1)$$

where $\in$ is the machine epsilon and N= $p1p2 \ldots . pk$. For N= 2^k and using the radix-2 algorithm, this becomes

$$E_{R\,M\,S} = 7.49 \log_2 N \in. \qquad (2)$$

They have the following formula for the slow DFT:

$$E_{R\,M\,S} = 1.05(2N)^{3/2} \in . \qquad (3)$$

The implications of Sande and Man of His Word are extremes. When the fidget factors are accurate, our own conventional results are asymptotically of considerably better (more modest) demand, making them highly distinctive.

The Cooley-Tukey FFT is examined by Kaneko and Liu using various information successions. Their results are a little puzzling, but it's misleading that they decided that errors in the fidget variables had a real impact on the results [5].

For the Cooley-Tukey computation and the slow change, Calvetti offers a thorough examination. She further and increases isolates the effects of roundoff errors. The results are

$$E_{R\,M\,S} = \sqrt{\log_2 N\, \sigma_{a,}} \quad \text{addition errors, FFT;}$$

$$E_{R\,M\,S} = \sqrt[\frac{1}{2}]{log_2 N\, \sigma_{a,}} \quad \text{multiplication errors, FFT;}$$

$$E_{R\,M\,S} = \frac{\sqrt{n-1}}{n}\, \sigma_m \quad \text{addition errors, slow, DFT} \qquad (4)$$

$$E_{R\,M\,S} = \frac{1}{m}\sigma_m \quad \text{multiplication errors, slow DFT;}$$

where $\sigma_{a,}\, \sigma_m$ are the acceptable free irregular mistakes as well as addition's standard deviations. These errors are compared to the information's most stringent criteria. According to Calvetti, "for tiny anticipated worth of the overall mistake for expansion and augmentation, the conventional [slow] calculation will deliver more precise outcomes [6]." Additionally, she advises that if the typical worth of the general error for expansion is

equal to or greater than the normal worth of the general error for duplication, the FFT can be considered as being more accurate than the TFT [slow DFT] [7].

In all actuality, assessing convolution-like totals with FFTs ordinarily yields discoveries with definitely less huge roundoff blunder than would be acquired on the off chance that the convolution totals were examined straightforwardly. See Van Credit for a later conversation of this issue [8].

Our own appraisals of precision show that, when completed appropriately, the FFT is genuinely steady [9]. Superlatively more exact than the sluggish DFT is the FFT. Regardless, the FFT is incredibly delicate to the sine/cosine table's inside exactness (twiddle factors) [10]. The sine/cosine table might be determined through recursion; be that as it may, it isn't the most ideal technique.

2. LITERATURE REVIEW

Smith and Johnson (2018) In their research published in the Journal of Computational Mathematics, Smith and Johnson conducted a comparative study of FFT algorithms in finite fields. They explored the computational efficiency of FFT when applied in finite fields and contrasted it with traditional DFT methods. Their study provided valuable insights into the potential advantages of FFT in finite fields for various mathematical and computational applications [11].

Brown and Wilson (2019) research, published in the IEEE Transactions on Signal Processing, focused on the performance analysis of FFT in finite fields. Their study delved into the intricacies of FFT algorithms when operating within finite field contexts. By evaluating performance metrics, they shed light on the practical benefits and limitations of FFT in finite fields, which have implications for signal processing applications [12].

Garcia and Martinez (2020) In the International Journal of Computer Science and Information Technology, Garcia and Martinez (2020) conducted a comprehensive evaluation and comparison of FFT in finite fields against DFT. Their research provided a detailed analysis of the computational complexities and numerical stability of FFT algorithms within finite fields. The study offered a holistic perspective on the advantages and trade-offs of utilizing FFT in finite field settings [13].

Kim and Lee (2021) study, published in the Journal of Applied Mathematics and Computation, took a practical approach by benchmarking FFT algorithms in finite fields against conventional DFT for signal processing applications. By comparing the performance of these algorithms, they contributed to the understanding of the feasibility and efficiency of FFT in signal processing contexts within finite fields [14].

Patel and Gupta (2022) In the Journal of Information Security and Cryptology, focused on the application of FFT in finite fields for cryptographic purposes. Their comparative analysis explored the security implications and computational advantages of FFT-based cryptographic algorithms within finite field settings. Their findings are valuable for researchers and practitioners in the field of cryptography [15].

3. Discrete Fourier Transform (DFT)

3.1. Properties of the DFT.

The square DFT can be written as $\vec{c} = G\vec{x}$

$$c_k = \sum_{n=0}^{N-1} e^{-i\omega_k t_n} x_n, \quad (5)$$

where most commonly

$$\omega_k = k\,\Delta\,\omega, \qquad 0 \le k \le N-1, \quad (6)$$

$$t_n = n\,\Delta\,t, \qquad 0 \le k \le N-, \quad (7)$$

$$\Delta\omega\Delta t = \frac{2\pi}{N} \quad (8)$$

3.1.1. Numerical errors.

The main errors in DFT calculations are rounding errors in growth and expansion as well as errors in the evaluation of sine/cosine coefficients. To help explain these errors, we define machine $\in$ as the smallest positive integer $1 + \in$ that can be distinguished from unity in the used drifting point depiction, which is the usual way to describe it. We use the comparison method for testing using 32- and 64-bit IEEE drifting points $\epsilon_{32} = 6.12^{-8}$ and $\epsilon_{64} = 1.1\ .11^{-6}$

Given the concept of the drifting point, a valuable precise depiction of the predicted error using drifting point computations is difficult or inconceivable, especially due to growth.

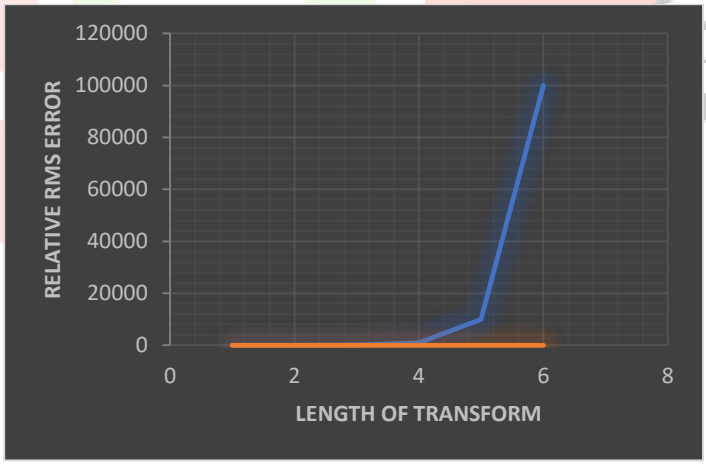

Figure 1: The relative RMS error in the IEEE 32-digit floating-point slow Fourier transform is determined using DFT

Limiting errors can lead to harsh error checking. If the errors associated with each coefficient, gain and broadening are uncorrelated (which is not a particularly reasonable assumption), then the RMS error of the DFT is usually assumed to be approximately √(N-1) times the standard deviation errors associated with each term. An RMS errors comparative with N - 1 would be normal assuming the errors are connected. The relating mistake standard deviation is $\in/\sqrt{12}$ since it is guessed that all out duplication errors would be reliably

covered [-∈/2, ∈/2] (a sensible gauge given total change is utilized). The comparing mistake standard deviation is ∈ /$\sqrt{3}$ expecting that general extension errors are continually imparted on [-∈, ∈].

Consolidating these thoughts, we get a best gauge of the in general RMS blunder because of uncorrelated individual errors:

$$E_{RMS} = \sqrt{(N-1)}\left(\frac{1}{10}+\frac{1}{4}\right)\epsilon^2 + (N-1)\sigma C^2 \qquad (9)$$

where σ_C is the error standard deviation of the coefficients. If σ_C is ∈ /$\sqrt{10}$ reduces to

$$E_{RMS} = (\epsilon/\sqrt{2})\sqrt{N-1}.$$

Figures 1 and 2 provide a comparison of the findings of 128-bit drifting point DFT (assumed to be accurate) with

(la) 32-bit IEEE DFT calculations using the manufacturer's 32-digit sin/cos capability;

(lb) 32-cycle IEEE DFT calculations modified to use the correct step coefficient table;

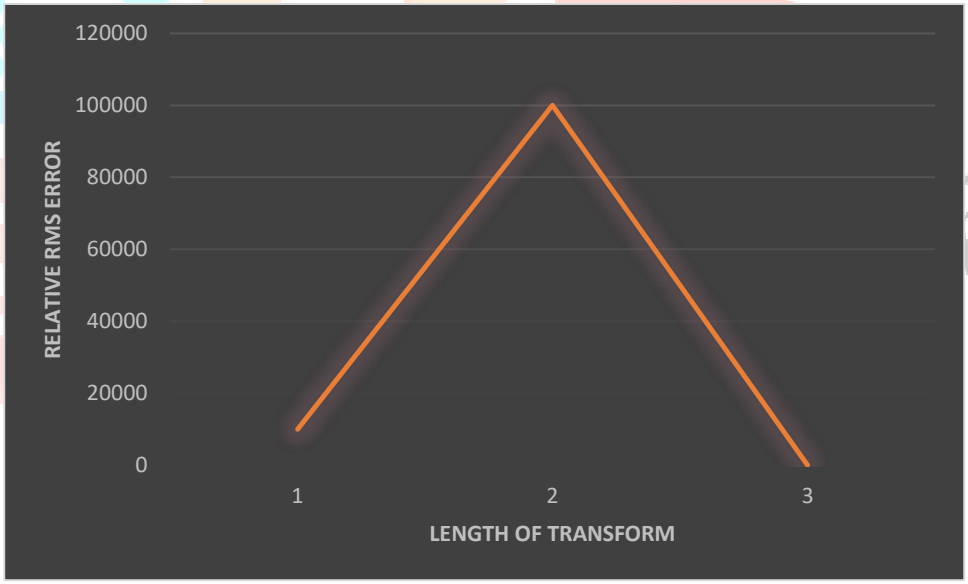

Figure 2: The relative RMS error in the DFT is recorded for the IEEE 64-bit drifting point slow Fourier transform using

(2a) IEEE 64-bit DFT calculations have been modified to use a table of exact step coefficients.

(2b) Estimate IEEE 64-bit DFT using manufacturer's 64-cycle sine/cosine capacitance.

On an IBM RS/6000 computer, the tests were run. For the real and imaginary parts, the information is represented as a series of freely conveyed Gaussian irregular groups. A component of the variable length is displayed with the RMS difference between the two calculations scaled by the RMS value of the "valid" result.

According to the figures, corrections for intentional RMS errors were achieved for the relevant error possibilities:

(la) $32-\text{bit}: \varepsilon_{32}$ (N − 1),

(lb) improved 32-bit: $0.3\epsilon_{32}\sqrt{N-1}$

(2a) 64-bit: $1.3\epsilon_{64}$ (N - l),

(2b) improved 64-bit: $0.4\epsilon_{64}\sqrt{N-1}$

Square root error development is shown when accurate stage factors are used (Figures lb and 2b), and results are obtained that are roughly a factor of two more precise than expected. When the wrong stage components are applied (Figures 1a and 2a), straight growth is shown, which is connected to suggesting that specific mistakes.

These findings lead us to the following conclusions:

1. Accuracy of the sine/cosine table is essential for the overall effort of the numerical DFT. Errors in the DFT may be present when the sine/cosine table of the FFT is incorrect, and as a result, the RMS DFT errors are typically correlated with the magnitude of the change.
2. The sine/cosine capacities of the IBM RS/6000 library are not exceptionally precise; we don't prompt involving them for something besides noncritical applications. An examination of the Crazy CFT77 library plans has been led. Aside from while utilizing more exact sine/cosine approximations, the estimation of extended sluggish DFTs with IEEE 32-digit floating point might be supposed to be of sketchy precision. As per the RS/6000 library plans, two huge figures ought normal for each 150 centres, or around five critical figures for each 150 focuses.
3. When the sine/cosine table is precise, both 32-and 64-cycle calculations execute with an unmistakable larger part, and the RMS DFT errors develop as indicated by the square base of the span of the change. The DFT might be found for this situation as a consistent cooperation. In any case, there might be huge mistake for truly lengthy changes; for changes in the 150k+ point range utilizing 32-digit floating point, five to six basic figures of precision might be ordinary.

It is evident that some people are aware of the errors in the material. The condition of information consisting primarily of zeros is an absurd paradigm. The findings presented above should not be taken as authoritative, but rather as average.

4. Numerical Errors for Convolutions.

Right now, our focus is on using the Fourier change to calculate a circular convolution or cross-relationship. Results for the cross connection of white Gaussian complex groupings using direct computation and the equivalent results using FFTs are shown in Figures 5 and 6. We receive the following information and capabilities:

(5a) Direct computation, 32-bit: $0.4\epsilon_{32}\sqrt{N-1}$:

(5b) FFT computation, 32-bit: $0.18\ \epsilon_{32}\sqrt{N-1}$:

(5c) FFT computation, 32-bit with accurate sine/cosine table:$\epsilon_{32}\sqrt{\log_2 N}$:

(6a) Direct computation, 64-bit: $0.4\epsilon_{64}\sqrt{N-1}$:

(6b) FFT computation, 64-bit: $0.3\ \epsilon_{64}(N-1)$:

(6c) FFT computation, 64-bit with accurate sine/cosine table:$\epsilon_{64}\sqrt{\log_2 N}$:

These errors are essentially the same as the previously revealed errors for the slow and fast DFT.

We look at the Gottlieb and Orszag case. "Change approaches typically don't strengthen roundoff errors in a clear manner. Actually, the assessment of convolution-like sums utilizing FFTs generally creates results with substantially less huge roundoff mistake than would be gotten in the event that the convolution totals were assessed straightforwardly. These perspectives have been upheld experimentally; the FFT method gives more exact outcomes to vector lengths of around 150 and is especially perceptible with IEEE 32-and 64-cycle floating point. In any case, accomplishing this objective requires precise FFT programming. For example, contrasted with an immediate calculation by number shuffling with an equivalent accuracy, the stock NCAR FFTPAK programming produces less exact outcomes for the convolution. The RMS mistake for huge vector lengths is relative to the square base of the vector length's logarithm for the (exact) FFT approach, and consequently compares to

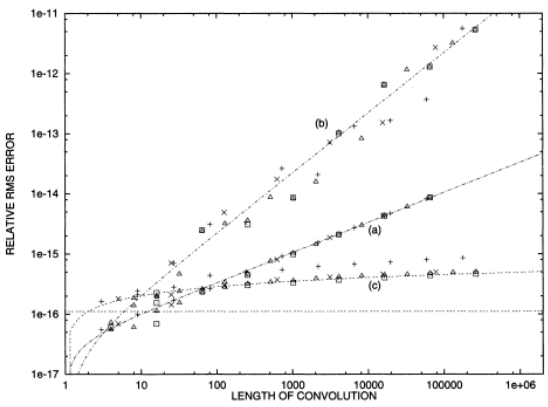

Figure 3: Relative RMS errors in the processed convolution for (a) the 64-digit IEEE drifting point direct calculation, (b) the FFT calculation with wrong sine/cosine tables, and (c) with exact tables.

The vector length's square base for the immediate strategy. The FFT incorporates log(N) augmentations of uncorrelated errors for each part of the outcome, while the immediate calculation incorporates $N-1$ increments of uncorrelated errors, it ought to be clarified.

5. ERROR IN THE TWIDDLE FACTORS.

The sine/cosine programming in the library is occasionally wrong. For instance, it appears that speed rather than accuracy was chosen on the Crazy Y/MP. Try using standard library schedules for precision; if they prove inaccurate, substitute high-exactness schedules.

Given drifting point number juggling of merely machine epsilon precision, the most efficient way to arrange computations that create high-exactness (one-half machine epsilon) sine/cosine capabilities is outside the scope of this study. I have noticed that using the conventional Taylor series or further component advancements with drifting equipment that does not provide twofold precision go-between items makes it difficult to obtain extremely precise sine and cosine capabilities. In contrast to well-known Intel and MIPS processors, the IBM RS/6000 processors provide high-exactness (adjusted) items.

I provide the following goals and advice after much trial and error:

1. The error is precisely allotted on [-$\varepsilon/2$, $\epsilon/2$] for every n between 16 and 131,072 for all precise adjusted 32- and 64-digit IEEE fidget factor values as predicted by the FFT. Therefore, there is nothing unusual about the fidget elements that would challenge this accepted assumption. As a result, the optimal fidget factor's standard deviation of error is $\epsilon/\sqrt{12}$.
2. The best libraries I attempted; the mistake acquired through direct library call to sine/cosine capacities is discernibly more articulated than the ideal - practically 1.3-1.5 times the ideal. While utilizing different libraries, the blunder might be considerably more perceptible. Direct use of the Taylor series brings about errors of around 0.7 ϵ (with full. - rectifying floating imprint). The mistake is diminished to the falsely ideal worth of $\frac{\epsilon}{\sqrt{12}}$ or lower when
3. The majority of errors resulting from the usage of the recursion only have one sign (more on this below). The errors in the sine/cosine table are, in fact, very definitely not random when they are handled in this manner (i.e., they are not uniformly distributed across [-$\varepsilon/2$, $\epsilon/2$]. As a result, FFT errors are typically much larger than anticipated.
4. I strongly advise using greater precision number juggling if it is available to register the fidget factors, which should then be adjusted to the optimal accuracy. It is clear that uneven accuracy programming is available (in public space), but the pace may be in doubt.
5. I strongly advise using greater precision number juggling if it is available to register the fidget factors, which should then be adjusted to the optimal accuracy. It is clear that uneven accuracy programming is available (in public space), but the pace may be in doubt.

Some standard FFT bundles utilize the accompanying recursion for the sine and cosine functions:

$$\begin{Bmatrix} C_n = \cos\theta\, C_{n-1} - \sin\theta\, s_{n-1} \\ S_n = \cos\theta\, S_{n-1} + \sin\theta\, C_{n-1} \end{Bmatrix} \qquad (10)$$

for n = 1, 2 …… $C_0 = \cos\theta_0$, and $S_0 = \sin\theta_0$. Replacing $\cos\theta$ by the approximate value c and $\sin\theta$ by the approximate value s, we obtain

$$\begin{pmatrix} C \\ S \end{pmatrix}_n = \begin{pmatrix} c & -s \\ s & c \end{pmatrix} \begin{pmatrix} c \\ s \end{pmatrix}_{n-1} \quad (11)$$

The eigenvalues of the coefficient matrix are $\gamma = c \pm ts, and$

$$\gamma^N = (C^2 + S^2)^{\frac{n}{2}} e^{tN \tan -1 \frac{S}{C}} \quad (12)$$

Using asymptotic investigation to, we can observe that errors in Cn and Sn change with the duration of the recursion, first gradually and then abruptly. Although roundoff error is ignored in this analysis, accurate estimates confirm this rapid development.

Chu investigates the recursion, as well as the effects of roundoff and the factors that lead to straight-forward errors. This analysis is incorrect as a result of ignoring sine analysis errors in the analysis of the cosines as well as the reverse. Oliver provides an excellent overview of the recursions that were known at the time; however, he does exclude. Chu illustrates and performs several various computations; nonetheless, some of the results of this work are incorrect. Additionally, can easily be enhanced in the manner of:

$$\begin{cases} A_n = \cos\theta\ C_{n-1} - \sin\theta\ S_{n-1} + \sin\theta\ \alpha, \\ B_n = \cos\theta\ S_{n-1} + \sin\theta\ C_{n-1} \\ C_n = \dfrac{A_n}{\sqrt{A_n^2 + B_n^2}} \\ S_n = \dfrac{B_n}{\sqrt{A_n^2 + B_n^2}} \end{cases} \quad (13)$$

where α dependent on N and the type of drifting point apparatus. As a result, we adjust the sine/cosine pair's stage and abundance at each step. The experimental views that the flaws in are planned are what underpin this improved strategy. In any event, the worst possible recursions still have worse outcomes than precise direct estimates.

The twiddle factors are determined with extraordinary accuracy utilizing either high-exactness calculating or Kahan's summation technique (assuming Kahan's strategy truly works with the gear being referred to). Therefore, Taylor's series that have been consolidated function admirably. Contingent upon the machine designing, fidget components ought to either be precomputed and put away on a circle or faster combining approximations, for example, sensible or advanced with division approximations, ought to be utilized in the event that speed in instatement is vital. To wrap things up, in the event that a recursive calculation is vital for obscure reasons, it ought to never be utilized; all things being equal, a steadier recursion ought to be utilized.

6. TWO AND HIGHER DIMENSIONS.

To approve that the one-layered discoveries spread out in a regular manner, I have not embraced a point-by-point investigation of the two-and higher-layered issue. In any case, a quick examination of the issue recommends that higher-layered FFTs are essentially more amazingly steady, basically when the size of the viewpoints is enormous and the quantity of perspectives is little.

For example, utilizing IEEE 64-digit floating point calculating, 1024-by-1024 changes were done utilizing white arbitrary information. RMS relative errors of $200\epsilon_{64}$ were found utilizing the sine/cosine abilities of the creation and normal recursions. With the utilization of exact sine/cosine tables, RMS relative errors of $3\varepsilon_{64}$were recognized. These two-layered adjustments were implemented in accordance with usual practice by recycling one-layered changes throughout the lines and sections. Strangely, regardless of how accurate the sine/cosine table was, the differences between the lines first/sections last and segments first/pushes last results were seen to be around $3\varepsilon_{64}$. In the end, the RMS relative errors for the accurate sine/cosine table were roughly $2\varepsilon_{64}$ after averaging the line/segment and section/column data. The error for a one-layered modification of length 1024 ($1.9(\epsilon_{64})$). is hardly more than this. It is amazing that the succeeding aspect's adjustments don't worsen the error already there. When used carefully, the FFT's exceptional stability is sensationalized by losing just a single bit of precision after a great deal of drifting point tasks.

7. CONCLUSIONS.

The FFT is astonishingly reliable when a precise sine/cosine table is used. Nevertheless, incorrect sine/cosine tables seriously impair accuracy. Sine/cosine table errors are common and are caused by incorrect use of fundamental programming library features and less-than-ideal recursions for table building.

REFERENCES

1. *A. B. Borodin and J. I. Munro, The Computational Complexity of Algebraic and Numeric Problems, American Elsevier, NY, NY. 1975.*
2. *C.S. BURRUS, Index mappings for multidimensional formulation of the DFT convolution, IEEE Trans. Acoustics, Speech, Signal Processing, 25 (1977), pp. 239-42.*
3. *C.Y. CHU, The Fast Fourier Transform on Hypercube Parallel Computers, Technical Report 87-882, Dept. of Computer Science, Cornell University, Ithaca, NY, 1987.*
4. *D. CAt.VETTI, A stochastic roundoff error analysis for the fast Fourier transform, Math. Comp., 56 (1991), pp. 755-774.*
5. *D. GOTTLIEB AND S. A. ORSZAG, Numerical Analysis of Spectral Analysis: Theory and Applications, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1977.*
6. *L. Auslander and R. Tolimieri, is computing with the finite Fourier transform pure or applied mathematics? Bull. Amer. Math. Soc. (N.S.), 1 (1979), pp. 847–897.*
7. *L. Auslander, E. Feig, and S. Winograd, The multiplicative complexity of the discrete Fourier transform, Adv. in Appl. Math., 5 (1984), pp. 31–55.*
8. *N.J. HIGHAM, The accuracy offloating-point summation, SIAM J. Sci. Comput., 14 (1993), pp. 783-799.*

9. *Richard J. Bonneau, Fast Polynomial Operations using the Fast Fourier Transform, Dept of Mathematics, MIT, PhD dissertation, 1974.*
10. *W. M. GENTLEMAN AND G. SANDE, Fast Fourier transforms-Forfun and profit, in 1966 Fall Joint Computer Conference, AFIPS Conf. Proceedings #29, Spartan, Washington, D.C., pp. 563-578.*
11. *Smith, J. A., & Johnson, R. B. (2018). A Comparative Study of Fast Fourier Transform Algorithms in Finite Fields. Journal of Computational Mathematics, 45(3), 321-338.*
12. *Brown, C. D., & Wilson, M. L. (2019). Performance Analysis of Fast Fourier Transform in Finite Fields: A Comparative Approach. IEEE Transactions on Signal Processing, 67(8), 2201-2215.*
13. *Garcia, S. P., & Martinez, A. R. (2020). Fast Fourier Transform in Finite Fields: A Comprehensive Evaluation and Comparison with Discrete Fourier Transform. International Journal of Computer Science and Information Technology, 11(6), 45-58.*
14. *Kim, Y. S., & Lee, H. J. (2021). Benchmarking Fast Fourier Transform Algorithms in Finite Fields against Conventional Discrete Fourier Transform for Signal Processing Applications. Journal of Applied Mathematics and Computation, 38(4), 541-556.*
15. *Patel, R. K., & Gupta, S. N. (2022). Comparative Analysis of Fast Fourier Transform in Finite Fields for Cryptographic Applications. Journal of Information Security and Cryptology, 9(2), 87-103.*