



Science

## ON THE COMPARATIVE ANALYSIS OF THE FAST BLUESTEIN AND FAST COOLEY-TUKEY NUMERICAL ALGORITHMS FOR DIGITAL AND ANALOG SIGNALS PROCESSING

Amannah, Constance Izuchukwu<sup>1</sup>, Bakpo, Francis Sunday<sup>2</sup>

<sup>1,2</sup> Department of Computer Science, University of Nigeria, Nsukka, Enugu State



### ABSTRACT

*This study was designed to compare the computing efficiency of FC-TNADSP and the FBNADSP to ascertain a faster numerical algorithm necessary for the processing of digital signals. The faster numerical algorithm established in this study is abbreviated with RCFC-TNADSP (Compared Resultant-TNADSP). The methodology adopted in this work was comparative analysis development design. The major technologies used in this work are the FC-TNADSP and FBNADSP which were hitherto simulated on the c++ programming technologies. The c++ served as a signal processing language simulator (SPLS). The execution times of the fast Cooley-Tukey and the fast Bluestein algorithms were 1.70 seconds and 1.74 seconds respectively. On comparing the speeds of the fast Cooley-Tukey and the fast Bluestein algorithms we observed that the fast Cooley-Tukey algorithm has 0.04 seconds speed improvement over the fast Bluestein algorithm. In line with this outcome, we concluded that the fast Cooley-Tukey algorithm (FC-TNADSP) is faster than the fast Bluestein algorithm (FBNADSP). In the same vein the fast Cooley-Tukey algorithm (FC-TNADSP algodsp-2) is therefore the fastest DSP algorithm. This is however faster than the spectrum of FFT algorithms of  $O(n \log n)$  computing speed, a speed considered to be the fastest hitherto. The result of this study shows we can have faster numerical algorithms other than the traditional spectrum of FFT algorithms of  $O(n \log n)$  computing speed. The algorithms were tested on input block of width 1000 units, and above, and can be implemented on input size of 100 000, and 1000 000 without the challenge of storage overflow. The input samples tested in this work was the discretized pulse wave form with undulating shape out of which the binary equivalents were extracted. Other forms of signals may also be tested in this fast algorithm provided they are interpreted in the digital wave type. In order to optimized the advantage of the developed algorithms, the frequency index,  $K$  should be as defined in this study, that is  $K = \pi e^\theta$  iff  $0 < \theta \leq 8$*

### Keywords:

Algorithm, FFT, Cooley-Tukey, Bluestein, Comparasion, Analisis.

**Cite This Article:** Amannah, Constance Izuchukwu, and Bakpo, Francis Sunday, “ON THE COMPARATIVE ANALYSIS OF THE FAST BLUESTEIN AND FAST COOLEY-TUKEY NUMERICAL ALGORITHMS FOR DIGITAL AND ANALOG SIGNALS PROCESSING” International Journal of Research – Granthaalayah, Vol. 3, No. 11(2015): 133-146.

## 1. INTRODUCTION

Signals play an important role in our daily life. A signal is the variable parameter that contains information and by which information is transmitted in an electronic system or circuit Dictionary of Science. The majority of the signals found in science are analog in nature. In analog signals, both dependent variable and independent variables are continuous. Such signals may be processed directly by analog systems (i.e., analog filters) for the purpose of changing their characteristics or extracting some desired information.

Digital Signal Processing (DSP) is an area of Science and Engineering which has developed very rapidly over the past few decades. As a matter of fact, the techniques and applications of Digital Signal Processing (DSP) are as old as Newton and Gauss and also as new as today's digital computers and Integrated Circuits (ICs). This rapid development of Digital Signal Processing (DSP) has been a result of the significant advances in digital computer technology and IC fabrication techniques. Signal processing is a method of extracting information from the signal which in turn, depends upon the type of signal and the nature of information it carries. Thus, signal processing is concerned with representing signals in mathematical terms and extracting the information by carrying out algorithmic operations on the signal. Mathematically, a signal can be represented in terms of basic functions in the domain of the original independent variable or it can be represented in terms of basic functions in a transformed domain. Similarly, the information contained in the signal can also be extracted either in the original domain or in the transformed domain.

### ***STATEMENT OF THE PROBLEM***

The speed and scope of transmitting from analog to digital remain issues that need scientific resolution. In view of the foregoing an efficient computing algorithmic platform is a requirement for effective processing of digital signals. This research is therefore designed to compare the computing speeds of the Fast Cooley-Tukey (FC-TNADSP) and Fast Bluestein numerical algorithms (FBNADSP).

### ***AIM AND OBJECTIVES OF THE STUDY***

The aim of the study is compare the computing efficiency of the (FC-TNADSP) and the (FBNADSP). In order to attain this aim, the following objectives were considered;

- a) To investigate the FC-TNADSP
- b) To investigate the FBNADSP
- c) To compare the computing speeds of the FC-TNADSP and the FBNADSP
- d) To establish the faster algorithm between FC-TNADSP and FBNADSP

### ***SCOPE OF THE STUDY***

There are spectrums of FFT numerical algorithms. This work did not compare these FFT algorithms but rather the FC-TNADSP and the FBNADSP. This study is restricted to the software approach of DSP-algorithm implementation on a minicomputer or a personal computer. A detailed discussion of hardware, firmware, and DSP chip implementation is beyond the scope of this study.

## ***SIGNIFICANCE OF THE STUDY***

When implemented, the faster recommended in this study will be of significant assistance to the broadcasting industry in support of their bid to transit to an inclusive digital processing. Furthermore, the Digital Signal Processing algorithms of this research will provide computing framework for simulating signals associated with speech, image, communication, and so on.

## **2. RELATED LITERATURE**

### ***2.1.THE INTERPOLATION IN DIGITAL SIGNAL PROCESSING AND NUMERICAL ANALYSIS***

[2] Investigated the Interpolation in Digital Signal Processing and Numerical Analysis. He reviewed some of the methods being used in interpolation such as Lagrange, Hermite, Shannon, and Piecewise polynomials interpolators, but there are lots of other methods that he did not mention. [2] Concluded that it is impossible to give a general comparison of them and say which one is the best. It depends on the case we are dealing with he added to choose a method he further added; “I don’t think there will be any guarantee that the chosen method be the best possible one. Some people have compared some of these methods in their papers”. For instance, [3] says “unlike small-kernel convolution methods, which have poor accuracy anywhere near the Nyquist limit, the accuracy of the FFT method is maintained to high frequencies, very close to Nyquist”. From [2], it is obvious that [2] did not effectively determine nor recommend an efficient algorithm for DSP and numerical analysis. Furthermore, the work did not also delve into detailed numerical principles compatible enough with DSP operations. [2] Dealt passively on FFT. [2] Did not also reconcile the place of DSP in relation to numerical algorithms. In view of the foregoing, we can therefore see the vacuum created by [2]; the non-determination of the place of efficient numerical algorithms for processing digital signals. Our study is poised to fill this gap. Hence our study is consequential, timely, and pivotal.

### ***2.2.THE APPLICATION OF COMPLEX DIGITAL PROCESSING IN COMMUNICATIONS***

[1] Examined the application of complex digital processing in communications. They observed that the concept of DSP is coloured with the principles of mathematics. They opined that complex DSP has no obvious connection with our everyday experience. Their reason is occasioned by the fact that many DSP problems are explained mainly by means of real numbers mathematics. Their study agreed that some DSP problems are based on mathematics, such as Fast Fourier Transform (FFT), z-transform, representation of periodical signals and linear systems, and so on. Their research shows that the imaginary part of complex transformations is usually ignored or regarded as zero due to the inability to provide a readily comprehensible physical explanation. The result of their investigation shows that two DSPs exit namely complex DSP and real DSP. Their position explains that the principle of complex DSP is Complex math and that of real DSP is real math. Their approach is not universally applicable and can only be used with problems and applications which conform to the requirements of complex math techniques. Making a complex number entirely mathematically equivalent to a substantial physical problem is the real essence of complex DSP.

It is clear to see that the world has gone digital and almost being completely controlled by numbers. The concept of theoretical DSP which presupposes that DSP has no obvious connection with our everyday experience has become a mere tale. DSP can now be applied to virtually every aspect of our human endeavour. What were considered as complex DSPs occasioned by complex math can now be processed using simulateable numerical algorithmic processes such as the difference equation and the Fast Fourier Transform algorithms. In line with this the need of investigating the feasibility of fast numerical algorithms for DSP applications is both necessary and timely. This study is therefore needful and timely considering the fact that some countries are yet to integrate into the digital communication even as Nigeria is preparing to do so in the year to come.

### ***2.3.EFFICIENT DIGITAL FILTERS***

In [4] investigation of Efficient Digital Filters, filters can be sharpened to meet the need of predefined DSP operations. His work shows that filters (FIR) can be sharpened either by reducing passband ripple or increasing stopband attenuation. His work proved that an existing system can still be used to perform the function of an improved new filter by simply sharpening the existing filter. This way, an old filter becomes new not by creation but by improvement. Simply stated, filter sharpening is a technique for creating a new filter from an old one [5, 6, 7]. From this line knowledge it is obvious that DSP activities can be analyzed with filters and the filters can be sharpened in different ways. [4]'s work centered on software filter. He did not however introduce math-based into his work. This cannot be exhaustively concluded considering the fact that DSP is mainly the analysis and manipulation of mathematical principles in relation to signals. In such case the math platform becomes the filter which could be interlink with software filters. To enhance the effectiveness of DSP operation numerical algorithms could best be considered especially if efficiency is the focus. This is an area that is compactable with contemporary communications systems. A research work addressing this gap is vital and timely. My research is aimed at investigating efficient numerical algorithms for convenient DSP operations. Its DSP filters will be math-based and could be sharpened to address the contemporary issues of DSP activity in signal driven society.

### ***2.4.ALGORITHMS AND TOOLS FOR AUTOMATIC GENERATION OF DSP HARDWARE STRUCTURES***

[8]'s research concentrated on hardware-based DSP. Its results are basically hardware structures. Some kinds of algorithms, namely the scheduling algorithms were employed but obviously not complex numerical algorithms. Furthermore, the basis for testing these algorithms is equally not specified in his work. And finally, the mode of analysis of the operations of DSP is not also delineated. In view of these articulated gaps in [8]'s work, an extended research work capable of addressing some of the concerns will be in order. This is why a work on efficient algorithms for DSP is required. My research is therefore aimed at addressing the need of efficient numerical algorithms for DSP.

### ***2.5.SPL: A LANGUAGE AND COMPILER FOR DSP ALGORITHMS***

The formulas are represented in a language that the authors called SPL, an acronym for Signal Processing Language. The SPL work was conducted by [9]. This was about the design and

implementation of a compiler that translates formulas representing signal processing transform into efficient C or FORTRAN programs. Their results showed that SPIRAL, which can be used to implement many classes of algorithms, produces program that perform as well as had systems like FFTW. SPL is a descendant of the TPL (Tensor Product Language), [10] that was developed for the automatic generation of FFT algorithms. [11, 12] differ in their findings. They hold that signal processing is not necessarily better suited for an approach like the one discussed above. Their approach is similar to that used by FFTW. The works of [13, 14, 15] concentrated on Fast Fourier Transform (FFT) and FFTW. The SPL is a tool used to implement DSP in C or FORTRAN programs. However, the work stresses the method of translation and compilation of signals.

### **2.6.SPL: A LANGUAGE AND COMPILER FOR DSP ALGORITHMS**

The formula are represented in a language that the authors called SPL, an acronym for Signal Processing Language. The SPL work was conducted by [16]. This was about the design and implementation of a compiler that translates formulas representing signal processing transform into efficient C or FORTRAN programs. Their results showed that SPIRAL, which can be used to implement many classes of algorithms, produces program that perform as well as have systems like FFTW. SPL is a descendant of the TPL (Tensor Product Language), [17] that was developed for the automatic generation of FFT algorithms. [18, 19] differ in their findings. They hold that signal processing is not necessarily better suited for an approach like the one discussed above. Their approach is similar to that used by FFTW. The works of [19, 20, and 21] concentrated on Fast Fourier Transform (FFT) and FFTW. The SPL is a tool used to implement DSP in C or FORTRAN programs. However, the work stresses the method of translation and compilation of signals.

## **3. MATERIAL AND METHOD**

The method adapted in this work is the comparative analysis methodology. The materials are the FBNADSP and the FC-TNADSP. They are the products of the Fast Fourier Transforms (FFTs) which are surely the most widely used signal processing algorithms. It is the basic building block for a large percentage of algorithms in current usage. An FFT is a way to compute the same result more quickly.

### **3.1.AN OVERVIEW OF THE EXISTING SYSTEM**

The existing system is consist of the fast Bluestein and Cooley-Tukey algorithms. Each of the two algorithms investigated is traced to and transformed from the Discrete Fourier Transform (DFT). The DFT is of two phases; the forward and inverse phases.

#### **Discrete Fourier Transform (DFT)**

##### **DFT Formula**

Let  $x_0, \dots, x_{N-1}$  be complex numbers. The DFT is defined by the formula:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi k \frac{n}{N}}, \quad k=0 \dots N-1 \quad \dots (1)$$

Evaluating this definition directly requires  $O(N^2)$  operations.



**Inverse Discrete Fourier Transform (IDFT)**

The inverse transform of DFT:

$$\text{DFT: } F[n] = \sum_{k=0}^{N-1} f[k] e^{-j \frac{2\pi}{N} nk}$$

The required inverse DFT becomes

$$F[k] = \frac{1}{N} \sum_{n=0}^{N-1} F[n] e^{+j \frac{2\pi}{N} nk} \dots (2)$$

From the inverse transform formula, the contribution to  $f[k]$  of  $F[n]$  and  $F[N-n]$  is

$$fn[k] = \frac{1}{N} \left[ F[n] e^{j \frac{2\pi}{N} nk} + F[N-n] e^{j \frac{2\pi}{N} (N-n)k} \right]$$

For all  $f[k]$  real,  $F[N-n] = \sum_{k=0}^{N-1} f[k] e^{-j \frac{2\pi}{N} (N-n)k}$

But  $e^{-j \frac{2\pi}{N} (N-n)k} = \underbrace{e^{-j 2\pi k}}_{1 \text{ for all } k} e^{+j \frac{2\pi}{N} nk} = e^{+j \frac{2\pi}{N} nk}$

i.e.  $F[N-n] = F^*(n)$  (i.e. the complex conjugate)

Comparing equations (1) and (2) we infer that the inverse DFT is the same as the DFT, but with the opposite sign in the exponent and a  $\frac{1}{N}$  factor. Any fast Fourier Transform (FFT) algorithm can easily be adapted from it.

**3.2. THE FAST BLUESTEIN NUMERICAL ALGORITHM FOR DSP (FBNADSP)**

**a. The Bluestein’s FFT Algorithm**

Bluestein’s FFT algorithm also known as chirp Z-transform algorithm can be used to compute prime length DFTs in  $O(N \log N)$  operations. However, it is not restricted to prime lengths, and it can compute other kinds of transform. The Bluestein’s FFT algorithm can be expressed from DFT as:

$$X(k) = \sum_{n=0}^{N-1} x(n) W^{-kn}, \quad k = 0, 1, 2, \dots, N-1$$

Multiplying and dividing by  $W^{(n^2 + k^2)/2}$  yields

$$\begin{aligned} X(k) &= \sum_{n=0}^{N-1} x(n) W^{-kn} W^{1/2 (n^2 + k^2)} W^{-1/2 (n^2 + k^2)} \\ &= W^{-1/2 k^2} \sum_{n=0}^{N-1} [x(n) W^{-1/2 n^2}] W^{1/2 (k-n)^2} \\ &= W^{-1/2 k^2} (x_q * W_q) \dots \end{aligned} \tag{3}$$

Eqn (3) is called the Bluestein’s FFT algorithm

Where ‘\*’ denotes convolution and  $x_q$  and  $w_q$  are defined by

$$x_q(n) \triangleq x(n) W^{1/2 n^2}, \quad n = 0, 1, 2, \dots, N-1$$

$$w_q(n) \triangleq W^{1/2 n^2}, \quad n = -N+1, -N+2, \dots, -1, 0, 1, 2, \dots, N-1$$

$$W \triangleq \exp(j2\pi / N)$$

The Bluestein's FFT algorithm provides  $N \log N$  complexity for any positive integer DFT – length  $N$  whatsoever, even when  $N$  is prime. The sequence  $x_q$  above consists of the original data sequence  $x(n)$  multiplied by a signal  $\exp(j\pi^2/N)$  what can be interpreted as a sampled complex sinusoid with instantaneous radian  $2\pi n/N$  frequency. Such signals are called chirp signals. For this reason, Bluestein algorithm is also the chirp-z transform algorithm.

### b. The Cooley-Tukey Numerical Algorithm

The Cooley-Tukey algorithm was named by [14]. Since Cooley-Tukey FFT algorithm is a combination of DFT of Even-indexed of  $X_n$  with DFT of odd-indexed part of  $X_m$ , we can expect a slightly different result.

$$X_K = \sum_{m=0}^{N/2-1} X_{2m} e^{-\frac{2\pi}{N}(2m)k} + \sum_{m=0}^{N/2-1} x_{2m+1} e^{-\frac{2\pi}{N}(2m+1)k} \quad \dots (4)$$

### 3.3. THE PROPOSED SYSTEM

The proposed system is abbreviated as CFBFC-TNADSP, which means comparison on Fast Bluestein and Fast Cooley-Tukey Numerical algorithms for Digital Signal Processing. The CFBFC-TNADSP is basically designed to compare the computing efficiency of the Fast Bluestein and Fast Cooley-Tukey Algorithms.

#### The Cooley-Tukey FFT Algorithm

The Cooley-Tukey is the most common FFT algorithm. This is a divide and conquer algorithm that recursively breaks down a DFT of any composite size  $N = N_1 N_2$  into many smaller DFTs of sizes  $N_1$  and  $N_2$ , along with  $O(N)$  multiplications by complex roots of unity traditionally called twiddle factor.

Radix-2 decimation-in-time (DIT) FFT is the simplest and most common form of the Cooley-Tukey algorithm. Radix-2 DIT divides a DFT of size  $N$  into two interleaved DFTs (hence the name "radix-2") of size  $N/2$  with each recursive state. The DFT has a forward and inverse form which is defined as:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-j2\pi kn/N} \quad (\text{Forward DFT}) \quad \dots (5)$$

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{j2\pi kn/N} \quad (\text{Inverse DFT}) \quad \dots (6)$$

Eq (10) can simply be re-expressed as:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi}{N}nk} \quad \dots (7)$$

Where  $K$  is an integer ranging from 0 to  $N-1$ . Radix-2 DIT first computes the DFTs of the even indexed inputs ( $x_{2m} = x_0, x_2, \dots, x_{N-2}$ ) and of the odd-indexed inputs ( $x_{2m+1} = x_1, x_3, \dots, x_{N-1}$ ) and then combines those two results to produce the DFT of the whole sequence. This idea can then be performed recursively to reduce the overall runtime to  $O(N \log N)$ .

The above Radix-2 DIT algorithm shows that the DFT can be rearranged as a function of  $x_n$  with two parts; a sum over the even-numbered indices  $n = 2m$  and a sum over the odd-numbered indices  $n = 2m + 1$ .

The sum over the even-numbered indices combined with the sum over the odd-numbered indices yields:

$$X_k = \sum_{m=0}^{N/2-1} x_{2m} e^{-\frac{2\pi i}{N} (2m)k} + \sum_{m=0}^{N/2-1} x_{2m+1} e^{-\frac{2\pi i}{N} (2m+1)k} \dots (8)$$

A common multiplier in Eq (12) is  $e^{-\frac{2\pi i}{N} k}$

If the DFT of the Even-indexed inputs  $x_{2m}$  is denoted by  $E_k$  and the DFT of the odd-indexed inputs  $x_{2m+1}$  is denoted by  $O_k$ , and  $E_k + O_k$  results to:

$$X_k = \underbrace{\sum_{m=0}^{N/2-1} x_{2m} e^{-\frac{2\pi i}{N/2} mk}}_{\text{DFT of Even-indexed of } xm} + e^{-\frac{2\pi i}{N} k} \underbrace{\sum_{m=0}^{N/2-1} x_{2m+1} e^{-\frac{2\pi i}{N/2} mk}}_{\text{DFT of Odd-indexed part of } xm} \dots (9)$$

$$= E_k + e^{-\frac{2\pi i}{N} k} O_k$$

$$X_K = \sum_{m=0}^{N/2-1} X_{2m} e^{-\frac{2\pi i}{N} (2m)k} + \sum_{m=0}^{N/2-1} x_{2m+1} e^{-\frac{2\pi i}{N} (2m+1)k} \dots (10)$$

Substituting the unity factor ( $e^{2\pi i} = 1$ ) in eqn (10) we have;

$$X_K = \sum_{m=0}^{N/2-1} X_{2m} \left( -\frac{1}{N} 2mk \right) + \sum_{m=0}^{N/2-1} x_{2m+1} \left( \frac{-1}{N} (2m+1)K \right) \dots (11)$$

On simplifying we have:

$$X_K = \sum_{m=0}^{N/2-1} -\frac{1}{N} K [X_{2m} 2m + X_{2m+1} 2m+1] \dots (12)$$

$$X_K = N^{-1} K \sum_{m=0}^{N/2-1} (X_{2m}) 2m + (X_{2m+1}) 2m+1 \quad \text{(FC-TNADSP)} \dots (13)$$

Because of the periodicity of the DFT,

$$E_k + \frac{N}{2} = E_k \text{ and} \dots (14a)$$

$$O_k + \frac{N}{2} = O_k \dots (14b)$$

The outcomes of the DFT periodicity, Eq (14a) and (14b) when substituted into Eq( 9) yields:

$$X_K = \begin{cases} E_k + e^{-\frac{2\pi i}{N} k} O_k & \text{for } 0 \leq k < N/2 \\ E_k - N/2 + e^{-\frac{2\pi i}{N} k} O_k - N/2 & \text{for } N/2 \leq k < N \end{cases} \dots (14c)$$

Since the twiddle factor  $e^{-2\pi i k / N}$  obeys the relation:

$$e^{-2\pi i (k + N/2)} = e^{-\frac{2\pi i}{N} k - \pi i}$$



$$\begin{aligned}
 &= e^{-\pi i} e^{-\frac{2\pi i}{N}k} \\
 &= -e^{-\frac{2\pi i}{N}k} \quad \dots (15)
 \end{aligned}$$

This situation cuts the number of twiddle factor calculation to half, i.e.  $= 0 \leq k < \frac{N}{2}$ , this condition results to:

$$X_k = E_k + e^{-\frac{2\pi i}{N}k} O_k \quad \dots (16)$$

$$X_k + \frac{N}{2} = E_k - e^{-\frac{2\pi i}{N}k} O_k + e^{-\frac{2\pi i}{N}k} O_k \quad \dots (17)$$

Eqn (13) is called the **FC-TNADSP**. It is a resultant equation from the forward and inverse DFTs. This equation has four products, one addition, and one exponentiation. Eqn (11) in its original source (eqn 8) had eleven products one addition and two divisions. But upon re-indexing and subsequent simplifications, the parameter operators were reduced to four products, one addition, and one exponentiation. This drastic reduction in the number of operators accounts for the desired speed of the new algorithm.

Eqs (16) and (17), express the DFT of length N recursively in terms of two DFTs of size N/2, is the core of the radix-2 DIT Fast Fourier Transform. The algorithm gains its speed by re-using the results of intermediate computations to compute multiple DFT outputs. The final outputs are obtained by a+/- combination of  $E_k$  and  $O_k \text{Exp}(-2\pi i k/N)$ , which is simply a size -2 DFT (sometime called a butterfly).

**The Bluestein FFT Algorithm**

$$X_{(K)} = W^{-1/2} K^2 (xq * Wq)_K \quad \dots (18)$$

$$Wq(n) \triangleq W^{-1/2} n^2, \quad n = -N + 1, -N + 2, \dots -1, 0, 1, 2, \dots N - 1 \quad (18a)$$

$$W \triangleq \text{esp}(j2\pi / N) \quad \dots (18b)$$

$$Xq(n) \triangleq x(n)N^{-1/2} n^2, \quad n = 0, 1, 2, \dots N - 1 \quad \dots (18c)$$

Eqn (18) can be re-indexed by substituting for  $X_q$  and  $W_q$

$$X(K) = W^{-1/2} K^2 [(x(n)N^{-1/2} n^2) * W^{1/2} n^2]_k \quad \dots (19)$$

Collecting like term we have;

$$\begin{aligned}
 X(K) &= W^{-1/2} * W^{1/2} K^2 [(x(n)N^{-1/2} n^2) * W^{1/2} n^2]_k \\
 &= W^{-1/2} * W^{1/2} (K^2 * K)(n^2 * n^2) [x(n)N^{-1/2}] \\
 &= W^{-1/2} +^{1/2} (K^{2+1})(n^{2+2}) [x(n)N^{-1/2}] \\
 &= W^0 (K^3)(n^4) [x(n)N^{-1/2}] \\
 &= K^3 n^4 [x(n)N^{-1/2}] \quad \text{(FBNADSP)} \quad \dots (20)
 \end{aligned}$$

$$K = -N + 1, -N + 2 \dots -1, 0, 1, 2, \dots N - 1$$

$$n = 0, 1, 2 \dots N - 1$$

x(n) is defined as;

$$x(n) = kn = K^2 / 2 + n^2 / 2 - [(k - n)^2 / 2]$$

Eqn (20) is called the **FBNADSP**. It is the product of re-indexing and simplification. We can further decompose eqn. (20) by referencing the relationship between the DFT and the Bluestein FFT algorithm. The twiddle factor of unity  $i2\pi/N$  exists in DFT in eqn (17) but as **W** in the Bluestein FFT algorithm in eqn (18).

**Since W is defined as:**

$W \triangleq \text{esp } (j2\pi / N)$ , it therefore holds that the twiddle factor also exist in W. Thus, we can express the relationship as:

$$W = i2\pi / N = j2\pi / N$$

Substituting  $W = \text{esp } (j2\pi/N)$  into eqn. (20) We have;

$$X(k) = \text{esp } (j2\pi / N) K^3 n^4 [x(n) N^{-1/2}] \dots (21)$$

Since the twiddle factor is unity (equal to 1), a substitution of the unity factor into eqn (21) yields;

$$X(k) = \frac{1}{N} k^3 n^4 [x(n) N^{-1/2}] \dots (22)$$

$$= N^{-1} k^3 n^4 [x(n) N^{-1/2}]$$

$$= N^{-1} * n^{-1/2} k^3 n^4 [x(n)] \dots (23)$$

$$= N^{-1-1/2} k^3 n^4 [x(n)]$$

$$= N^{-3/2} k^3 n^4 [x(n)] \dots (24)$$

Eqn (24) is the resulting algorithm with three exponentiations and four products. This is a minimized Bluestein FFT algorithm which has six exponentiations and eight products. Such reduction in the number of operators can account for speed of the generated algorithm even as their implementation reveals. Table 2 below s shows the performance evaluation process of the system.

**Table 1:** Performance Evaluation Metric for FBC-TNADSP

S/N	Performance metrics	Normal range	Abnormal range	Value of compared algorithm	Rating of FC-TNADSP
1.	Fault Density (Fd)	0.0001 – 1.49	1.5 – 5.9	0.0019980	*HP
2.	Defect density (DD)	0.01 – 0.49	0.5 – 3.9	0.02	*HP
3.	Modular test coverage	40 -100	10 – 39%	48	*HP
4.	Requirement traceability (TM)	41 – 100	10 – 40	90%	*HP
5.	Software maturity index (SMI)	0.5 – 1.0	0.1 – 3.9	0.5263	*HP
6.	Cyclomatic complexity (C)	0.1 – 4.49	4.5 - ∞	4	*HP

Adapted from [12]\*HP  High Performance

## 4. DISCUSSION OF RESULTS

### 4.1.COMPARISON OF THE FAST COOLEY-TUKEY AND FAST BLUESTEIN ALGORITHMS

The two algorithms were evaluated to have different speed parameters. Table 2 below shows the variation in computing speed of each of the two algorithms.

**Table 2:** Output Comparison of theFast Cooley-Tukey Numerical Algorithm (FC- TNADSP) and Fast Bluestein Numerical Algorithm (FBNADSP)

Warehouse Input	NUMERICAL ALGORITHMS			Comparison
	FAST ALGORITHMS			
Number of sampled DSP inputs (N)	ALGORITHMS	Time (Sec)	Difference between FBNADSP and FC-TNADSP(in Seconds)	% Improvement
N=1000	FBNADSP	1.74	0.04	49.42%
	FC-TNADSP	1.70		
N=1000 000	FBNADSP	1.74	0.04	
	FC-TNADSP	1.70		

### 4.2.RESULTS

There were two fast algorithmic models that the study comparatively analyzed namely the fast Cooley–Tukey and the fast Bluestein algorithms. The execution times of the fast Cooley-Tukey and the fast Bluestein algorithms were 1.70 seconds and 1.74 seconds respectively as shown in Table 2 above. On comparing the speeds of the fast Cooley–Tukey and the fast Bluestein algorithms we observed that the fast Cooley-Tukey algorithm has 0.04 seconds speed improvement over the fast Bluestein algorithm. In line with this outcome, we conclude that the fast Cooley-Tukey algorithm (FC-TNADSP) is faster than the fast Bluestein algorithm (FBNADSP). In the same vein the fast Cooley-Tukey algorithm (FC-TNADSP algodsp-2) is therefore the fastest DSP algorithm. It is faster than the FFT algorithm species (the Bluestein FFT algorithm, the Cooley-Tukey FFT algorithm, the PFA FFT algorithm, and the Brunn’s FFT algorithm). This result therefore shows that a version of algorithm with computing speed that is faster than the  $O(n \log n)$  computing speed of the fast Fourier algorithms exist.

## 5. SUMMARY AND CONCLUSION

### 5.1.SUMMARY OF ACHIEVEMENTS

This study was aimed at comparing the computing speeds of the fast Cooley-Tukey algorithm (FC-TNADSP) and the fast Bluestein algorithm (FBNADSP). The study anchored its comparison on the well-established research results of fast Cooley-Tukey algorithm (FC-TNADSP) and that of the fast Bluestein algorithm (FBNADSP).

The fast Cooley-Tukey algorithm (FC-TNADSP) and the fast Bluestein algorithm (FBNADSP) have computing speeds of 1.70 seconds and 1.74 seconds respectively. This is however faster than the spectrum of FFT algorithms of  $O(n \log n)$  computing speed, a speed considered to be the fastest hitherto.

The result of this study shows we can have faster numerical algorithms other than the traditional spectrum of FFT algorithms of  $O(n \log n)$  computing speed. In favour of this result, we conclude the study therefore concludes that the fastest numerical algorithm for Digital signal Processing with 1.76 seconds faster than the traditional FFT algorithms and 0.04 faster than the Fast Bluestein algorithm which is on its own faster than the routine FFT algorithms by 1.74 seconds.

### 5.2.CONCLUSION

In-depth analysis in this study revealed that the FFT are predicated on the discrete Fourier transform (DFT) of  $O(N^2)$  computing speed while the Cooley-Tukey algorithm (FC-TNADSP) and the fast Bluestein algorithm (FBNADSP) resulted from the spectrum of the FFT algorithms. In the same vein, the trust of this study was the Cooley-Tukey algorithm (FC-TNADSP) and the fast Bluestein algorithm (FBNADSP)..

### 5.3.RECOMMENDATIONS

The result of this study is certainly going to enhance the computing speed of digital signals. The confirmed fast algorithm is basically recommended for the processing of digital signals and not analog signals. However, it can also be used for analog signals only when they are converted to digital signals. The algorithms were tested on input block width of 1000, and above, and can be implemented on input size of 100 000, and 1000 000 000 without the challenge of storage overflow. In order to optimize the advantage of the developed algorithms, the frequency index,  $K$  should be as defined in this study, that is  $K = \pi e^{\theta}$  iff  $0 < \theta \leq 8$

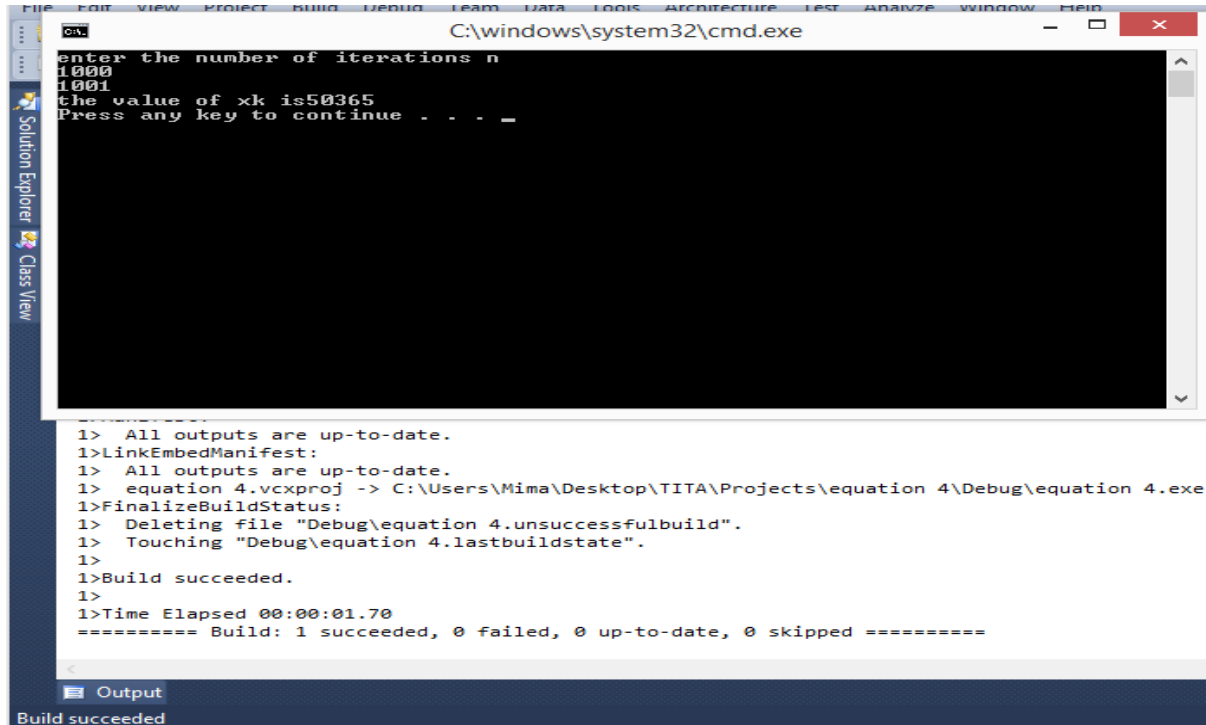
## 6. REFERENCES

- [1] Vladimir, P. and Zlatka, N., Georgi, I., Miglen, O., (2011). *Complex Digital Signal Processing in Telecommunications: Applications of Digital Signal Processing*, Dr. Christian Cuadrado-Laborde (Ed.), 307-406.
- [2] Saeed, B. (2003). *Interpolation in Digital Signal Processing and Numerical Analysis*. New York: Springer-Verlag.

- [3] Fraser, D. (1989). *Interpolation by the FFT Revisited An Experimental Investigation*, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, (37)5, pp. 665-675.
- [4] Matthew, P.D. (2000). *Efficient Digital Filters*, *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-
- [5] Sanjit, K.M. (2001). *Digital Signal Processing: A Computer Approach*, McGraw-Hill, New York
- [6] Hamming, R. and Kaiser, J. (1977). *Sharpening the Response of a Symmetric No recursive Filter by Multiple Use of the Same Filter*, " *IEEE Trans. Acoustics, Speech, Signal Proc*, ASSP, (25) 5, pp. 415 – 422 .
- [7] Hamming, R. (1998). *Digital Filters*, 3rd ed. Dover, Mineola, New York, pp. 140 – 145 .
- [8] Pedro, F. Z. T. (2009). *Algorithms and tools for automatic generations of DSP hardware structures*, Edition, Boston: McGraw Hill.pp.18-23, 317-320.
- [9] Jianxin, X., Johnson, J.mRebert, Padua, D. (2001). *SPL: A Language and Compiler for DSP algorithms*.
- [10] Auslander, L., Johnson, J., Johnson, R.W (1996). *Automatic Implementation of FFT Algorithms*, Technical Report DU – MCS – 96 – 01, Department of Mathematics and Computer Sciences, Drexel University, Philadelphia, PA, Presented at the DARPA ACMP PI Meeting.
- [11] Frigo, M. and Johnson, S.G. (1998). *FFTW: An Adaptive Software architecture for the FFT*. In *ICASSP*, Vol.3, pp 1381 – 1384
- [12] Frigo, M. (1999). *A fast Fourier Transform Compiler*. In *PLDI*.
- [13] Blesser, B. &Kates, J.M. (1978). *Digital processing in audio signals*. In A.V. Oppenheim, editor. *Applications of Digital Signal Processing*, Prentice Hall, Englewood Cliffs NJ.
- [14] Cooley James W., and John W. Tukey (1965). *An algorithm for the machine calculation of complex fourier series*. *Mathematics computer*. Vol. 19, 1965
- [15] Bedekar, P. P., Blade, S. R., & Kale, V. S., (2009). *Optimum time coordination of overcurrent relays in distribution system using Big-M ( Penalty) method*, " *WSEAS Trans. On Power systems*. 4(11), pp 341-350.
- [16] Al-Hasawi, W. &Gilany, M. (2009). *Proposed techniques for identifying open and short circuit sections in distribution networks*: *WSEAS Trans. on Power Systems*,4(12) Pp.372-381.
- [17] *An Introduction to the Mathematics of Digital Signal Processing: Part I: Algebra, Trigonometry, and the Most Beautiful Formula in Mathematics* Author(s): F. R. Moore Source: *Computer Music Journal*, Vol. 2, No. 1, (Jul., 1978), pp. 38-47 Published by: The MIT Press Stable
- [18] <http://www.jstor.org/stable/3680137> Accessed: 30/07/2012 23:37 your
- [19] Good, I.J. "The interaction algorithm and practical fourier analysis". *Journal of the royal statistical society, series B* 20 (2): 361 – 372 JSTOR 2983896, 1958. (<https://www.jstor.or/state/2983896>). Addendum, *ibid.* 22(2), 375 (1960)
- [20] Thomas, L.H. (1963). *Using Computers to solve problems in Physics*. *Applications of Digital computers* .Bostom:Ginn
- [21] Pavan Kumar K.M., Priya Jain, Ravi Kiran S, Rohith N., Ramamani K. *FFT Algorithm: A Survey*. *The International Journal of Engineering and Sciene (IJES)* Volume 2 Issue 4 pages 22-26, 2013 ISSN(e): 2319-1813 ISSN():. *FFT Algorithm: A Survey*. *The International Journal of Engineering and Sciene (IJES)* Volume 2 Issue 4 pages 22-26, 2013 ISSN(e): 2319-1813 ISSN()::319-1805

## APPENDIX A: SAMPLE OUTPUTS OF THE FC-TNADSP AND THE FBNADSP

### FASTNUMALGO\_2: RE-INDEXEDCOOLEY-TUKEY FAST FOURIER TRANSFORM ALGORITHM

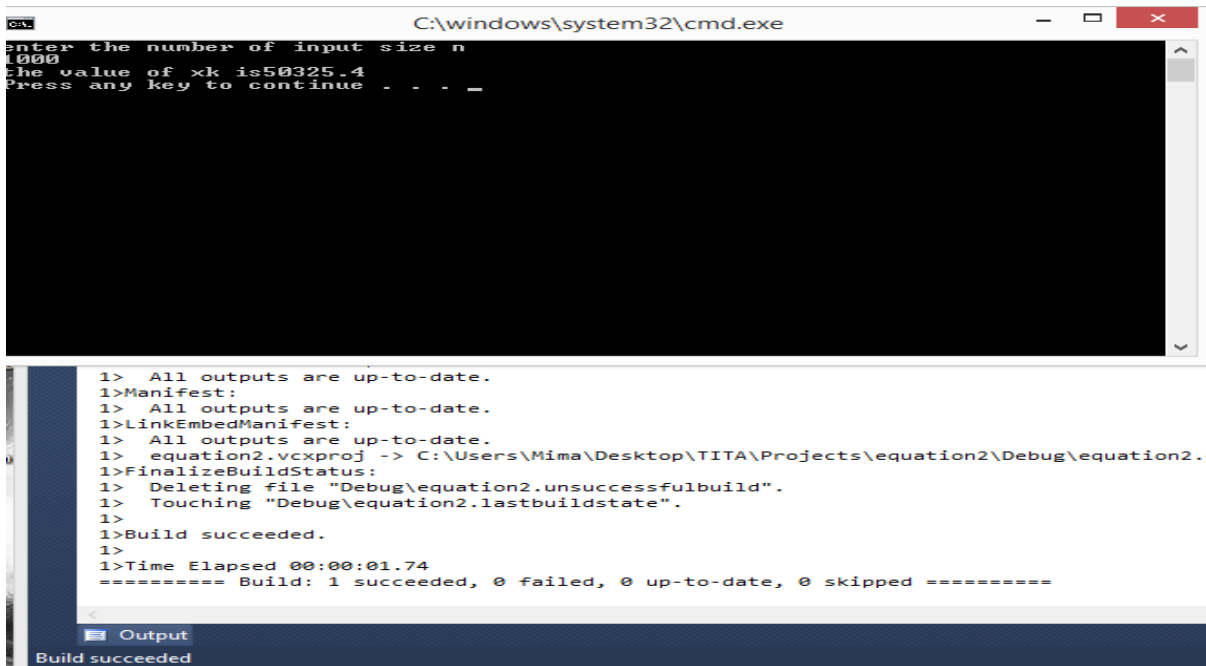


```
C:\windows\system32\cmd.exe
enter the number of iterations n
1000
1001
the value of xk is50365
Press any key to continue . . . _

1> All outputs are up-to-date.
1>linkEmbedManifest:
1> All outputs are up-to-date.
1> equation 4.vcxproj -> C:\Users\Mima\Desktop\TITA\Projects\equation 4\Debug\equation 4.exe
1>FinalizeBuildStatus:
1> Deleting file "Debug\equation 4.unsuccessfulbuild".
1> Touching "Debug\equation 4.lastbuildstate".
1>
1>Build succeeded.
1>
1>Time Elapsed 00:00:01.70
===== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped =====

Output
Build succeeded
```

### FASTNUMALGO\_1 RE-INDEXEDBLUESTEINS'S FAST FOURIER TRANSFORM ALGORITHM



```
C:\windows\system32\cmd.exe
enter the number of input size n
1000
the value of xk is50325.4
Press any key to continue . . . _

1> All outputs are up-to-date.
1>Manifest:
1> All outputs are up-to-date.
1>LinkEmbedManifest:
1> All outputs are up-to-date.
1> equation2.vcxproj -> C:\Users\Mima\Desktop\TITA\Projects\equation2\Debug\equation2..
1>FinalizeBuildStatus:
1> Deleting file "Debug\equation2.unsuccessfulbuild".
1> Touching "Debug\equation2.lastbuildstate".
1>
1>Build succeeded.
1>
1>Time Elapsed 00:00:01.74
===== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped =====

Output
Build succeeded
```