Transactions of the Nigerian Association of Mathematical Physics
Volume 4, **(***July, 2017***), pp37 – 40**

Mathematical Formulation of An Octal – Binary Discrete Fourier Transform Algorithm.

Atonuje A. O[†] and Tsetimi J.[††]

Department of Mathematics Delta State University, Abraka, Nigeria.

Abstract

The Cooley-Tukey and Sande – Tukey fast Fourier transform algorithms are efficient techniques which reduce the prohibitive running time as well as the number of computational operations of discrete Fourier transform (DFT) of functions. In this paper, a matrix argument is applied to extend the mathematical formulation of the Cooley–Tukey (C-T) FFT algorithm. This extension is relatively more efficient and amenable to the construction of signal flow graphs and reduces the number of computational operations from N^2 ***to*** $O(N\log_2 N)$***dependent operations, where*** N ***is the number of sampled points over which the DFT is computed.***

Keywords: Cooley - Tukey, fast Fourier transform algorithm, matrix argument, discrete Fourier transform, Octal – Binary FFT, computational operations.

1.0 Introduction

In mathematics, physics and computer science, difficult problems are expressed in continuous Fourier series which is a periodic system associated with trigonometric and other transcendental functions. If the continuous Fourier series is generalized for infinite region, it is called continuous Fourier transform (CFT). Most times, the CFT can be sampled at regular intervals of points and as such represented by discrete Fourier transform (DFT). In practice, one frequently works with data that is given as a set of discrete quantities and then, the finite discrete Fourier transform may be useful because it approximates the continuous Fourier transform.

In the past few decades, DFT had captured the attention of many mathematicians, numerical analysts and computer scientists and in many applications, large digitized data - sets are becoming available. However, such data - sets cannot be easily computed as a result of the running time of DFT as well as the large number of computational operations both through the direct method or the use of the computer machine. Efforts in the literature aimed at tackling this problem can be found in [1-5].

Consider the polynomial function of the form;

$$f(x)=a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0 \qquad (1)$$

where $a_n \neq 0$ is the leading coefficient with $Deg\ f = n$. Suppose that *Equation (1)* is sampled (that is, its values are recorded) at N number of evenly spaced points which are Δ_x units apart. The continuous function in *Equation 1* now takes the form;

$$f(x_0),\ f(x_0+\Delta_x),\ f(x_0+2\Delta_x),\ \ldots\ f(x_0+n\Delta_x) \qquad (2)$$

where x assumes discrete values 0, 1, 2, . . . N-1, that is, $f(x+\Delta_x)$ as in *Figure 1*

Correspondence Author: Atonuje A.O., Email: austino412@yahoo.com, Tel: +23408035085758, +2348068561884 (TJ)

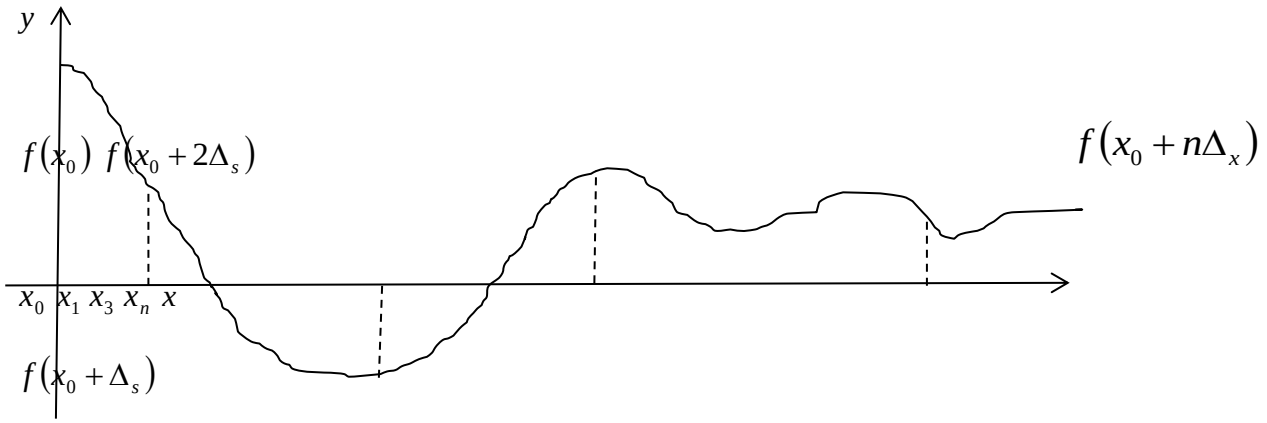

Figure 1: Discrete Fourier transform of a polynomial

The discrete Fourier transform of the function *f* in *Equation 1* is written as

$$F(u)=\sum_{x=0}^{N-1} f(x)e^{-i2\pi ux/N} \qquad (3)$$

Where the functions $f(x)_s$ are complex, $u=0,1,2,...N-1$. A straight forward calculation using *Equation (3)* would require N^2 operations, where "operation" means as it will throughout this article, a complex multiplication followed by a complex addition.

Example 1

Consider the infinite continuous polynomial function $f(x)=x+1$ sampled at four points whose x coordinate values are 0, 1, 2, 3. The required discrete Fourier transform is the four-point type, that is N = 4.

To compute the DFT of $f(x)$ at the given points, one first calculates the value of the function as follows:

$$\left.\begin{aligned} f(0)&=0+1=1\\ f(1)&=1+1=2\\ f(2)&=2+1=3\\ f(3)&=3+1=4 \end{aligned}\right\} \qquad (5)$$

Applying the formula of DFT,

$F(u)=\sum_{x=0}^{N-1} f(x)e^{-i2\pi ux/N}$ for $u=0, 1, 2, 3.$

$$F(0)=\sum_{x=0}^{N-1} f(x)\exp(-i2\pi(0)(0))=\sum_{x=0}^{N-1} f(0)(1)=f(0)+f(1)+f(2)+f(3)=1+2+3+4=10$$

$$F(1)=\sum_{X=0}^{3} f(x)\exp(-i2\pi(1)x/N)$$

$$=1\exp(-i2\pi(1)0/4)+2\exp(-i2\pi(1)1/4)+3\exp(-i2\pi(1)2/4)+4\exp(-i2\pi(1)3/4)$$

$$=1-2i-3+4=-2+2i$$

$$F(2)=\sum_{X=0}^{3} f(x)\exp(-i2\pi(2)x/4)$$

$$=1\exp(-i2\pi(2)0/4)+2\exp(-i2\pi(2)1/4)+3\exp(-i2\pi(2)2/4)+4\exp(-i2\pi(2)3/4)$$

$$=1-2+3-4=-2$$

$$F(3)=\sum_{X=0}^{3} f(x)\exp(-i2\pi(3)x/4)$$

$$=1\exp(-i2\pi(3)0/4)+2\exp(-i2\pi(3)1/4)+3\exp(-i2\pi(3)2/4)+4\exp(-i2\pi(3)3/4)$$

$$=1\exp(0)+2\exp\left(-i\frac{3}{2}\pi\right)+3\exp(-i3\pi)+4\exp\left(-i\frac{9\pi}{2}\right)$$

$$=1+2i-3-4i=-2-2i.$$

We can see that the number of complex multiplications for each *F* is $N^2 = 4^2 = 16$, that is , four complex multiplications for each F. The total number of complex additions in $N(N-1) = 4(4-1) = 4 \times 3 = 12.$

Suppose that the number of sampled points N is large, a total of N^2 multiplication operations and *N(N-1)* complex additions are quite enormous and could be time consuming and discouraging. The answer to the reduction of number of complex operations was suggested and made popular in [6], although a similar algorithm was earlier developed in [7]. This time saving and computational operation reduction method is called 'The fast Fourier transform (FFT) algorithm'.

In [8], the Cooley – Tukey (C – T) and Sande – Tukey (S – T) FFT algorithms discussed, assumed that the number of sampled points *N* over which the discrete Fourier transform is computed satisfies the relationship $N = 2^Z$ *where* $z \in Z$ is a positive integer value. The constraint appears to be too restrictive.

In the present article, a FFT algorithm which removes the assumption that $N = 2^Z$ is presented and an octal – binary FFT algorithm is developed. This is made possible by extending the ideas in [6] and [8] to verify that significant time savings can be obtained if $N^2 \neq 2^z$ but rather that $N = k_1.k_2$, which is less restrictive.

2. **Preliminary Notational Developments and theGeneralized Cooley – Tukey and Sande - Tukey FFT Algorithms for $N = 2^z$:**

In this section, the FFT algorithms in which the number *N* of sampled points for the continuous function $f(x)$ satisfies the relationship $N = 2^z$, where z is an integer value are presented. Detailed theoretical development of the C – T and S – T base *2* or binary bits algorithms can be found in [8]

Recall that the discrete Fourier transform algorithm any function *f* is usually given by;

$$F(u) = \sum_{x=0}^{N-1} f(x) e^{-i2\pi ux/N} \quad u = 0, 1, 2, ., N-1. \tag{6}$$

where $N = 2^Z$ by the C – T and S – T FFT algorithm. The decimated indices *u* and *x* in binary bit is written as

$$\left.\begin{aligned} u &= 2^{z-1}u_{z-1} + 2^{z-2}u_{z-2} + \ldots + u_0 \\ x &= 2^{z-1}x_{z-1} + 2^{z-2}x_{z-2} + \ldots + x_0 \end{aligned}\right\} \tag{7}$$

By substituting $W = e^{-i2\pi/N}$ into *Equation (6)* one gets

$$F(u) = \sum_{x=0}^{N-1} f(x) W^{ux} \quad u = 0, 1, 2, ., N-1. \tag{8}$$

Applying *Equation (7)* to *Equation 6* we have;

$$F(u_{z-1}, u_{z-2}, \ldots; u_0) = \sum_{x_0=0}^{1}\sum_{x_1=0}^{1}\cdots\sum_{u_{z-1}=0}^{1} f(x_{z-1,} x_{z-2}, \ldots x_0) W^P \tag{9}$$

where $P = \left(2^{z-1}u_{z-1} + 2^{z-2}u_{z-2} + \ldots + u_0\right)\left(2^{z-1}x_{z-1} + 2^{z-2}x_{z-2} + \ldots + x_0\right)$

By computing each of the summations separately, then tagging the intermediate results, the set of C – T FFT algorithm's recursive formulation becomes;

$$\left.\begin{aligned} f_1(u_0, x_{z-2}, \ldots x_0) &= \sum_{x_{z-2}=0}^{1} f_0(x_{z-1}, x_{z-2}, \ldots x_0) W^{2^{z-1}(u_0 x_{z-1})} \\ f_2(u_0, u_1 x_{z-3}, \ldots x_0) &= \sum_{x_{z-2}=0}^{1} f_0(u_0, x_{z-2}, x_{z-3}, \ldots x_0) W^{(2u_1+u_0)2^{z-2}x_{z-2}} \\ \vdots \qquad & \vdots \qquad \vdots \qquad \vdots \\ f_y(u_0, u_1 \ldots u_{z-1}) &= \sum_{x_0}^{1} f_{z-1}(u_0, u_1, \ldots x_0) W^{\left(2^{z-1}u_{z-1} + 2^{z-2}u_{z-2} + \ldots + u_0\right)x_0} \\ F(u_{z-1}, u_{z-2}, \ldots u_0) &= f_z(u_0, u_1, \ldots . u_{z-2}, u_{z-1}) \end{aligned}\right\} \tag{10}$$

Similarly, by defining the separate intermediate computational steps and by first separating the components of u instead of the components of *x*, one gets the Sande – Tukey FFT algorithm as

$$\left.\begin{aligned} f_1(u_0, x_0) &= \sum_{x_1=0}^{1} f_0(x_1, x_0) W^{2u_0 x_1} W^{u_0 x_0} \\ f_1(u_0, u_1) &= \sum_{x_0=0}^{1} f_1(u_0, x_0) W^{2u_1 x_0} \\ F(u_1, u_0) &= f_2(u_0, u_1) \end{aligned}\right\} \tag{11}$$

3. Mathematical Formulation of the Octal – Binary C – T FFT Algorithm:

Consider the case where the number *N* of sampled points fails to satisfy the relationship $N = 2^z$ for some integer value z, which is considered to be too restrictive, but rather *N* satisfies the relationship $N = k_1.k_2$, where $k_1 = 8$ and $k_2 = 2$. This is a case which represents the base *8* and base *2* or the simplest form of the octal – binary FFT algorithm for time savings and complex computational operation reduction.

Octal – binary FFT algorithm, as it is used throughout this paper, implies that as many arrays as possible are computed with a base *8* and then a base *2* array. To this end, substitute $k_1 = 8$ and $k_2 = 2$ into the decimated binary equivalent to obtain

$$\left.\begin{aligned} u &= 8u_1 + u_0, \;\; u_0 = 0, 1, 2, 3, 4, 5, 6, 7; \;\; u_1 = 0,1 \\ x &= 2x_1 + x_0, \;\; x_0 = 0, 1; \;\; x_1 = 0, 1, 2, 3, 4, 5, 6, 7; \end{aligned}\right\} \qquad (12)$$

Recall that the DFT of the continuous function sampled at *N* number of points is

$$F(u) = \sum_{x=0}^{N-1} f(x) e^{-i2\pi ux/N}$$

This can be re-written as

$$F(u) = \sum_{x=0}^{N-1} f(x) W^{ux}, \; u = 0, 1, 2, 3, 4, 5, 6, 7; \quad W = e^{-2\pi i/N}$$

Applying *Equation (12)* into *Equation (8)*, we have;

$$F(u_1, u_0) = \sum_{x_0=0}^{1} \left[\sum_{x_1=0}^{1} f_0(x_1, x_0) W^{2ux_1} \right] W^{ux_0} \qquad (13)$$

We now expands the quantity W^{2ux_1} using *Equation (12)* to get;

$$\begin{aligned} W^{2ux_1} &= W^{2(8u_1 + u_0)x_1} \\ &= \left[W^{16}\right]^{u_1 x_1} W^{2u_0 x_1} \\ &= W^{2u_0 x_1} \end{aligned} \qquad (14)$$

By the resulting expression in *Equation (14)* the inner sum of *Equation (13)* will now be expressed as

$$f_1(u_0, x_0) = \sum_{x_1=0}^{7} f_0(x_1, x_0) W^{2u_0 x_1} \qquad (15)$$

More so, the outer loop of *Equation (13)* can be expressed as

$$f_2(u_{0,} u_1) = \sum_{x_0=0}^{1} f_1(u_0, x_0)\, W^{(8u_1 + u_0)x_0} \qquad (16)$$

The unscrambling which result from the property of FFT is achieved according to the relation

$$F(u_1, u_0) = f_2(u_0, u_1) \qquad (17)$$

Equations (15), (16) and (17) represent the octal – binary FFT algorithm.

4. Conclusion

From the theoretical development of the octal – binary FFT algorithm we can see that *Equation (15)* is a base *8* iteration on the data array. Moreover, *Equation (16)* is a base 2 iteration on the data array. The octal – binary FFT algorithm is more efficient than the binary C – T and S - T algorithms as there is obvious reduction in the computation effort.

References:

[1] Elliot, D. F. and Rao, k. R. (1982). Fast Fourier Transform Algorithm Analysis and Applications. Academic Press. New York.

[2] Champeney, D. C. (1973). Fourier Transform and Their Physical Applications. Academic Press. New York

[3] Hamming, R. W. (1977). Digital Filters. Prentice-Hall Inc. Englewood Cliff New York.

[4] Papoulis, A. (1962). The Fourier Integral and its Applications. McGraw-Hill. New York.

[5] Atonuje, A. O, and Njoseh, I. N. (2004); On The Cooley – Tukey fast Fourier Transform Algorithm for Arbitrary factors. Jour. Nig. Assoc. of Mathematical Physics (JNAMP). Vol. 8, 121 – 124.

[6] Cooley, J. W. and Tukey, J. W. (1965); An algorithm for Machine calculation of Complex Fourier Series. Jour. Math. Comp. Vol. 19, 279 – 301.

[7] Danielson, G. C. and Lanczos, C (1942); Some improvements in practical Fourier Analysis and their Application to X – Ray scattering from liquids. J. Franklin Institute Vol. 233, 365 – 380 and 435 – 451.

[8] Atonuje, A. O. (2013); Discrete Fourier Transform: The Cooley – Tukey and Sande –Tukey Fast Fourier Transform Algorithm. Nigerian Jour. Math. Appl. Vol. 22, 70 – 79.