J. Nig. Ass. Math. Phys.
Vol. 6 (2002)

ON THE DISCRETE FOURIER TRANSFORM: THE COLLEY – TUKEY FAST FOURIER TRANSFORM ALGORITHM

ATONUJE A. O. & OKONTA, P.N.
Department Of Mathematics,
Faculty Of Science
Delta State University
Abraka.

ABSTRACT

In this paper we look at the Cooley – Tukey Fast Fourier Transform (C-T FFT) algorithm and its application to the Fourier Transform of discretely sampled Data. We apply the C-T FFT algorithm to the computation of a discrete Fourier transform and show that the number of complex multiplications followed by the number of additions in the computations of the discrete Fourier transform can be reduced from N^2 and $N(N-1)$ to $N_y/2$ and Ny respectively. Here y is a power of 2 to which N is expressed and N is the number of sampled points over which the discrete Fourier transform is computed.

1. INTRODUCTION

One is always faced with the question of the number of computations involved in computing the discrete Fourier transform for equally spaced N points. For many years until mid 1960's, the standard answer was this:

Consider the discrete Fourier transform for N points usually written for any continuous function f(x) as

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

for u = 0, 1, 2, ..., N – 1

Define W as the complex number

$$W = e^{-2\pi i/N} \qquad (1.0)$$

The Fourier transform becomes

$$F(u) = \sum_{x=0}^{N-1} f(x)W^{ux} \qquad (1.1)$$

In other words, the vector of f(x) is multiplied by a matrix whose (u, x)th element is the constant W to the power U x X. The matrix multiplication produces a vector result whose components are F(u)'s.

The matrix multiplication evidently requires N^2 complex multiplications plus a smaller number of addition operation to generate the

required power of W. So the discrete Fourier transform appears to be an $O(N^2)$ process. These appearance is called aliasing. This discrete Fourier transform can be computed in O(N log N) operations with an algorithm called the Fast Fourier transform derived by John W Tukey and James W Cooley.

Various authors have presented different types of Fast Fourier Transforms. These include Bloomfield [3], Champeney [6], Hockeny [11], Singleton [5], et cetera.

In this paper, we review the work of Cooly and Tukey to show that by algebraically decomposing the discrete Fourier transform, the number of operations can be reduced from N^2 dependence to $N \log_2 N$ subject to N being expressed as a power of 2. The difference between N^2 and $N \log_2 N$ is immense.

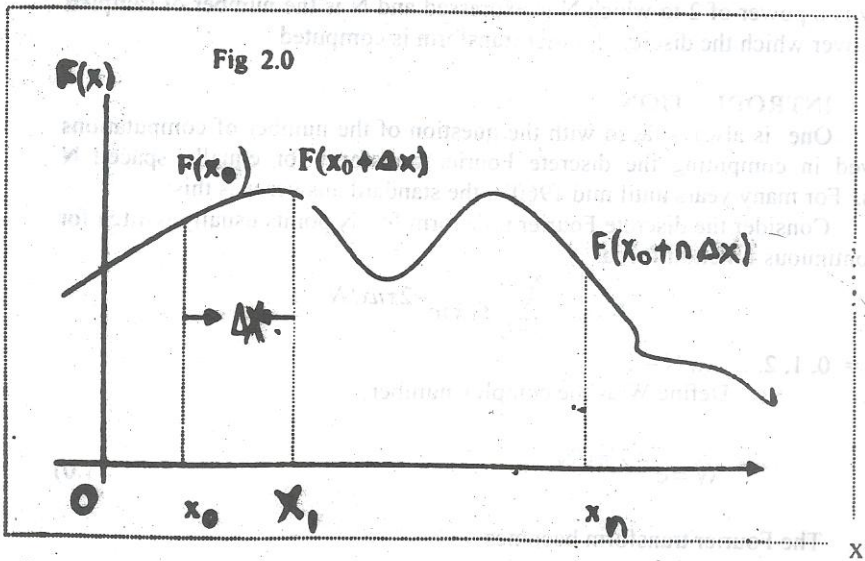
3. DISCRETE FOURIER TRANSFORM OPERATIONS

The figure below shows a continuous function f(x) sampled at points (that is its values are recorded) at evenly spaced points which are Δx units apart. The continuous function now takes the form;

$$f(x_o), f(x_o + \Delta x), f(x_o + 2\Delta x), \ldots\ldots, f(x_o + n\Delta x)$$

where x assumes discrete values 0, 1, 2, ..., N – 1, ie f(x + x Δ x).

Fig 2.0



From the figure 2.0, where the calculation of the Fourier transform is repe for each value of x, we define the direct finite Fourier transform of this discrete function by

$$F(u) = \sum_{x=0}^{N-1} f(x) e^{-2\pi i u x / N} \qquad (2.1)$$

For N values of u corresponding to the diserete values u = 0,1,2..... N – 1 (2.1) is evaluated as follows:

$$\left\{\begin{array}{l} F(0) = f(0)\exp(-i2\pi(0)\times 0/N) + f(1)\exp(-i2\pi(0)\times 1/N) + \ldots\ldots\ldots \\ F(1) = f(0)\exp(-i2\pi(1)\times 0/N) + f(1)\exp(-i2\pi(1)\times 1/N) + \ldots\ldots\ldots \\ \\ F(N-1) = f(0)\exp(-i2\pi(N-1)\times 0/N) + f(1)\exp(-i2\pi(N-1)\times 1/N) + \ldots. \end{array}\right. \quad (2.2)$$

Evaluation of F(0) requires a total of (2N-1) operations since we have to perform N multiplications (that is f(x) x exp .....) and N – 1 additions. Similarly, the evaluation of F(1) requires (2N – 1) operations and so on until the evaluation of F(u) over all N values of u is completed. This requires a total number of N^2 operations. (Note that the evaluation of exp (-i2πux/N) for values of U and x can be computed once and stored in a table for subsequent operations and does not constitute a direct part of the implementation).

3 THEORETICAL DEVELOPMENT OF THE GENERAL BASE 2 COOLEY – TUKEY FFT ALGORITHM, FOR THE CASE N = 2^V

When $N = 2^y$; (y an integer value), we consider the discrete Fourier transform

$$X(n) = \sum_{k=0}^{N-1} x_o(k) e^{-i2\pi nk/N} \quad (3.1)$$

where $x_o(k)$ are complex.

Set $W = e^{-i2\pi/N}$ and rewrite (3.1) as $X(n) = \sum_{k=0}^{N-1} x_o W^{nk}$ (3.2)

N = 0, 1, 2, ..., N – 1.

It is desirable to write or represent the integers n and k in binary number form as

$$\left.\begin{array}{l} n = 2^{y-1}k_{y-1} + 2^{y-2}n_{y-2} + \ldots\ldots + n_o \\ k = 2^{y-1}k_{y-1} + 2^{y-2}k_{y-2} + \ldots\ldots + k_o \end{array}\right\} \ldots\ldots\ldots (3.3)$$

Using the representation in (3.3) we can write (3.2) as

$$X(n_{y-1}, n_{y-2}, \ldots n_o) = \sum_{k_o=0}^{1}\sum_{k_1=0}^{1}\ldots\sum_{k_{y-1}=0}^{1} x_o(k_{y-1}, k_{y-2}, \ldots k_o) W^P \quad (3.4)$$

where $P = (2^{y-1}n_{y-1} + 2^{y-2}n_{y-2} + \ldots + n_o)(2^{y-1}k_{y-1} + \ldots\ldots + k_o)$ (3.5)

Since $W^{a+b} = W^a.W^b$, we rewrite W^p as

$$W^p = W^{(2^{y-1}n_{y-1}+2^{y-1}n_{y-2}+\ldots+n_o)(2^{y-1}k_{y-1})} x W^{(2^{y-1}n_{y-1}+2^{y-2}n_{y-2}+\ldots+n_o)\ldots\ldots}$$
$$\ldots\ldots\ldots\ldots\ldots x W^{(2^{y-1}n_{y-1}+2^{y-2}n_{y-2}+\ldots\ldots+n_o)k} \quad (3.6)$$

We now consider the first term in (3.6)

$$W^{(2^{n-1}n_{y-1}+2^{y-2}n_{y-2}+\ldots\ldots+n_o)(2^{y-1}k_{y-1})}$$
$$=[W^{2^y(2^{y-2}n_{y-1}k_{y-1}}][W^{2^y(2^{y-3}n_{y-2}k_{y-1}}]x\ldots\ldots x$$
$$[W^{2^y(n.k_{y-1})}]W^{2^y(n_ok_{y-1})} = W^{2^y(n_ok_{y-1})} \quad (3.7)$$

Since $$W^{2^y} = W^N = e^{-i2\pi/N} = 1 \quad (3.8)$$

Similarly, the second term of equation (3.6) yields

$$W^{(2^{y-1}n_{y-1}+2^{y-2}n_{y-2}+\ldots\ldots+n_6)(2^{y-2}k_{y-2})}$$
$$=[W^{2^y(2^{y-3}n_{y-1}k_{y-2}}][W^{2^y(2^{y-4}n_{y-2}k_{y-2}}]$$
$$x \ldots\ldots x [W^{2^y(n_1k_{y-1})}]W^{2^{y-1}(n_ok_{y-1})}$$
$$= W^{2^y(n_ok_{y-1})} \quad (3.7)$$

Since $$W^{2^y} = W^N = e^{-i2\pi/N} = 1 \quad (3.8)$$

Similarly, the second term of equation (3.6) yields

$$W^{(2^{y-1}n_{y-1}+2^{y-2}n_{y-2}+\ldots+n_o)(2^{y-2}k_{y-2})}$$

$$= [W^{2^y(2^{y-3}n_{y-1}k_{y-2}}][W^{2^y(2^{y-4}n_{y-2}k_{y-2}}]$$

$$\text{x} \ldots\ldots [W^{2^{y-1}(n_1k_{y-2})}]W^{2^{y-2}(n_ok_{y-2})}$$

$$= W^{(2n_1+n_o)2^{y-2}k_{y-2}} \qquad (3.9)$$

Note that as we progress through the terms of equation (3.6), we add another factor which does not cancel by the condition that W = 1.

This process continues until we reach the last term in which there is no cancellation.

Using these relationships, equation (3.2) can be rewritten as;

$$X(n_{y-1},n_{y-2},\ldots n_o)= \sum_{k_o=0}\sum_{k_1=0}\ldots\ldots\sum_{k_{y-1}=0} x_o(k_{y-1},k_{y-2},\ldots k_o)$$

$$\ldots\ldots\text{x } W^{(2^{y-1}n_{y-1}+2^{y-2}n_{y-2}+\ldots\ldots+n_o)k_o} \qquad (3.10)$$

Performing each of the summations separately and labeling the intermediate results, we obtain

$$x_1(n_o,k_{y-2},\ldots k_o)= \sum_{k_{y-2}=o}^{1} x_o(k_{y-1},k_{y-2},\ldots\ldots k_o)W^{2^{y-1}(n_ok_{y-1})}$$

$$x_2(n_o,n_1,k_{y-3},\ldots k_o)= \sum_{k_{y-1}=o}^{1} x_o(n_ok_{y-2},\ldots k_o)W^{(2n_1+n_o)2^{y-2}k_{y-2}}$$

$$x_y(n_o,n_1,\ldots n_{y-1})= \sum_{k_o=o}^{1} x_{y-1}(n_o,n_1,\ldots k_o)W^{(2^{y-1}n_{y-1}+2^{y-2}n_{y-2}+\ldots+n_o)k_o}$$

$$X(n_{y-1},n_{y-2},\ldots\ldots,n_o)=x_y(n_o,n_1,\ldots\ldots,n_{y-2},n_{y-1}) \qquad (3.11)$$

This set of recursive equations (3.11) represents the original Cooley-Tukey formulation of FFT algorithm for the general $N = 2^y$.

4. APPLICATION: THEORETICAL DEVELOPMENT OF THE BASE TWO COOLEY – TUKEY FFT ALGORITHM FOR THE CASE N = 4

A necessary evil of most theoretical developments is the introduction of new and unfamiliar notations. In the case of the Cooley – Tukey FFT algorithm, the simplicity achieved as a result of the notation change is worth the effort.

Consider the discrete Fourier transform

$$X(n)=\sum_{k=0}^{N-1} x_o(k)e^{-i2\pi nk/N}$$

Where $x_o(k)$ are complex. We now rewrite the above discrete Fourier transform as

$$X(n)=\sum_{k=0}^{N-1} x_o(k)W^{nk}, n=0,1,2,....,N-1 \qquad (4.1)$$

where we set $W=e^{-i2\pi/N}$

It is desirable to represent the integer n and k as binary numbers, that is; if we assume N = 4, then y = 2 from $N = 4 = 2^y = 2^2$.
Then we can represent k and n as 2-bit binary numbers.

K = 0, 1, 2, 3 or (k_1, k_o) = 00, 01, 10, 11
N = 0, 1, 2, 3 or (n_1, n_o) = 00, 01, 10, 11

A compact method of writing k and n is

$$K = 2k_1 + k_o,\ n = 2n_1 + n_o \qquad (4.2)$$

Where k_o, k_1, n_o, n_1 can take on the values 0, 1.

Equation (4.2) is simply the method of writing a binary number as its base 10 equivalent.

We can now write (4.1) and (4.2) for the case N = 4 as

$$X(n_1,n_o)=\sum_{k_o=o}^{1}\sum_{k_i=o}^{1} x_o(k_1,k_o)W^{(2n_1+n_o)(2k_1+k_o)} \qquad (4.3)$$

Note that the single summation in (4.1) must not be replaced by y summation in order to enumerate all the bits of the binary representation of k.

We can now attempt to factorize the power of W. We now consider the W^F term. Since $W^{a+b} = W^a W^b$, then

$$W^{(2n_1+n_o)(2k_1+k_o)} = W^{(2n_1+n_o)2k_1} W^{(2n_1+n_o)k_o}$$

$$= [W^{4n_1k_1}] W^{2n_ok_1} W^{(2n_1+n_o)k_o}$$

$$= W^{2n_ok_1} W^{(2n_1+n_o)k_o} \quad (4.4)$$

Note that the term in the brackets is equal to unity since

$$W^{4n_1k_1} = [W^4]^{n_1k_1}$$

$$= [e^{-i2\pi 4/4}]^{n_1k_1}$$

$$= [1]^{n_1k_1} = 1 \quad (4.5)$$

This follows from Euler's form. Thus (4.3) can be written in

$$X(n_1,n_o) = \sum_{k_o=0}^{1} [\sum_{k_1=o}^{1} x_o(k_1,k_o) W^{2n_ok_1}] W^{(2n_1+} \quad (4.6)$$

Equation (4.6) represents the foundation of the FFT algorithm.

Let us consider equation (4.6) individually. First rewrite the summation in the brackets as; $x_1(n_1,k_o) = \sum_{k_1=o}^{1} x_o(k_1,k_o) W^{2n_ok_1}$ (4

EQUATION (4.7) CAN BE WRITTEN AS

$X_1(0,0) = x_o(0,0) + 0 + x_o(1,0)W^0 + 0$
$X_1(0,1) = 0 + x_o(0,1) + 0 + x_o(1,1)W^0$
$X_1(1,0) = x_o(0,0) + 0 + x_o(1,0)W^2 + 0$
$X_1(1,1) = 0 + x_o(0,1) + 0 + x_o(1,1)W^2$

This can simply be rewritten by omitting the zero as;

$$\left.\begin{aligned} X_1(0,0) &= x_o(0,0) + x_o(1,0)W^0 \\ X_1(0,1) &= x_o(0,1) + x_o(1,1)W^0 \\ X_1(1,0) &= x_o(0,0) + x_o(1,0)W^2 \\ X_1(1,1) &= x_o(0,1) + X_o(1,1)w^2 \end{aligned}\right\} \quad (4.8)$$

If we rewrite (4.8) in matrix notation, we get;

$$\begin{bmatrix} X_1(0,0) \\ X_1(0,1) \\ X_1(1,0) \\ X_1(1,1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^0 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{bmatrix} \begin{bmatrix} x_0(0,0) \\ x_0(0,1) \\ x_0(1,0) \\ x_0(1,1) \end{bmatrix} \quad (4.9)$$

Similarly, if we write the outer summation of (4.6) as;

$$x_2(n_o,n_1) = \sum_{k_o=0}^{1} x_1(n_o,k_o) W^{(2n_1+n_o)k_o} \quad (4.10)$$

Enumerating the equation represented in (4.10), we have

$X_2(0,0) = x_1(0,0) + x_1(0,1)W^0 + 0 + 0$
$X_2(0,1) = x_1(0,0) + x_1(0,1)W^2 + 0 + 0$
$X_2(1,0) = 0 + 0 + x_1(1,0) + x_1(1,1)W^1$
$X_2(1,1) = 0 + 0 + x_1(1,0) + x_1(1,1)W^3$

We can now write the above in the form

$X_2(0,0) = x_1(0,0) + x_1(0,1)W^0$
$X_2(0,1) = x_1(0,0) + x_1(0,1)W^2$
$X_2(1,0) = x_1(1,0) + x_1(11,)W^1$
$X_2(1,1) = x_1(1,0) + x_1(1,1)W^3$

By enumerating the result in matrix form, we obtain;

$$\begin{bmatrix} X_2(0,0) \\ X_2(0,1) \\ X_2(1,0) \\ X_2(1,1) \end{bmatrix} = \begin{bmatrix} 1 & W^0 & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^1 \\ 0 & 0 & 1 & W^3 \end{bmatrix} \begin{bmatrix} x_1(0,0) \\ x,() \\ x_1(1,0) \\ x_1(1,1) \end{bmatrix} \quad (4.11)$$

From equation (4.6) and (4.10), we have by equating these two;

$$X(n_1,n_o) = x_2(n_o,n_1) \quad (4.12)$$

Thus the Outer Summation of equation (4.6) determines the second of the factored matrices.

From equation (4.12), we note that the final result $x_2(n_o,n_1)$ as obtained from the outer sum are in bit reversed order with respect to the desired values $Xn_1,n_o)$. This is simply the scrambling which results from the past Fourier Transform algorithm.

If we combine equation (4.7), (4.10) and (4.12), we have;

$$\left.\begin{aligned} x_1(n_o,k_o) &= \sum_{k_1=0}^{1} x_o(k_1,k_o) W^{2n_o k_1} \\ x_2(n_o,n_1) &= \sum_{k_1=0}^{1} x_1(n_o,k_o) W^{(2n_1+n_o)k_o} \\ X(n_1,n_o) &= x_2(n_o,n_1) \end{aligned}\right\} \quad (4.13)$$

Then the Set (4.13) represents the original Cooley – Tukey formulation of the FFT algorithm for N = 4. We term these equations recursive in that the second computation is done in terms of the first.

CONCLUSION.

The Fast Fourier Transform algorithm presented here brings about the matrix factorization process which introduces zeros into the factored matrices and as a result reduces the required number of complex multiplies to Ny/2 and complex additions to NY in the computation of the Fourier transform. The direct method of computation requires complex multiplications and N(N – 1) complex additions.

REFERENCES.

[1] BAKER, C.T.H. (1977), Numerical Treatment of Integral Equations (Oxford, Clarendon Press).

[2] BARRETH, A.N. And MACKAY, A.L. (1987), Spatial Structure and the Micro Computer: Selected Mathematical Techniques. (London, Macmillan Education Ltd).

[3] BLOOMFIELD, P. (1976), Fourier Analysis of Time Series: An Introduction (New York, John Wiley and Sons).

[4] BRACEWILL, R. (1948), The Fourier Transform And Its Application. (New York, Van Nostrand Reinhold).

[5] BRIGHAN, E.O. (1986), The Fast Fourier Transform And Its Application. (New Jersey, Prentice Hall Inc. Englewood Cliff).

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

[7] CONTE, S.D. And CARL, D. (1980); Elementary Numerical Analysis; An Algorithm Approach. (New Your, St. Louis, McGraw – Hill Book Company) 3rd Ed.

[8] DAHLQUIST, G. And BJORCK, A. (1974), Numerical Methods. (New Jersey, Prentice Hall, Englewood Cliff).

[9] ELLIOT, D.F. And RAO, K.R. (1982), Fast Fourier Transform Algorithm, Analyses, Application. (New York, Academic Press).

[10] HAMMING, R.W. (1977): Digital Filters. (New Jersey, Prentice Hall Inc. Englewood Cliffs).

[11] HOCKNEY, R.W. (1971): Methods In Computational Physics, Vol. 9. (New York, Academic Press).

[12] PAPOULIS, A. (1962): The Fourier Integral And Its Applications. (New York, McGraw Hill).

[13] RAYMON, E.A.C. And NORBER, W. (1967): Fourier Transform In The Complex Domain. (Providence, Rhode Island, American Mathematical Society).