Korean J. Math. **19** (2011), No. 3, pp. 243–253

A COOLEY-TUKEY MODIFIED ALGORITHM IN FAST FOURIER TRANSFORM

HwaJoon Kim* and Somchai Lekcharoen

Abstract. We would like to propose a Cooley-Tukey modified algorithm in fast Fourier transform(FFT). Of course, this is a kind of Cooley-Tukey twiddle factor algorithm and we focused on the choice of integers. The proposed algorithm is better than existing ones in speeding up the calculation of the FFT.

1. Introduction

The discrete Fourier transform (DFT) is a discretization of Fourier transform defined in a way to make it accessible to computer calculation, and fast Fourier transform (FFT) is not a transform at all, but an efficient algorithm for computing DFT. It has made possible far-reaching applications of Fourier methods in optics, digital filtering, spectral analysis of signals, and many other important areas of science and engineering [1]. We would like to begin discussion of DFT/FFT with definition. Let N is the number of samples, T is period of f and i is unit of imaginary number. Then the DFT/ inverse DFT of f are

$$F(\frac{n}{NT}) = \sum_{k=0}^{N-1} f(kT)e^{-2\pi ikn/N}$$

and

$$f(kT) = \frac{1}{N}\sum_{k=0}^{N-1} F(\frac{n}{NT})e^{2\pi ikn/N}$$

Received January 23, 2010. Revised July 20, 2011. Accepted July 25, 2011.
2000 Mathematics Subject Classification: 65T50, 65K05.
Key words and phrases: Cooley-Tukey algorithm, DFT, FFT, scaling, minimal operation.
*Corresponding author.

respectively, for $n = 0, 1, \cdots, N-1$. Afterward, let us leave off the scaling factor of T because this will not influence the efficiency of computing the numbers in the summation, which constitutes the real problem. So, we are interested in computing sums of the form

$$A(n) = \sum_{k=0}^{N-1} a(k)e^{-2\pi ikn/N}$$

for $n = 0, 1, \cdots, N-1$. Consequently, by using of FFT, we can do the savings in computation time, and this is the idea of FFT. In a word, FFT reduces the order of the number of operations from N^2 to $Nlog_2N$ for a length $N = 2^n$ DFT.

Since 1965 [5], FFT usage has rapidly expanded and personal computers fuel an explosion of additional FFT applications [2-9, 13]. Lately, many applications have been pursued containing nonuniform fast Fourier transform [11], determination of price of option [4], magnetic resonance imaging(MRI) [11] and applications related to resolution [7, 8]. Among these papers, Spilt-Radix algorithm [9] consists of low arithmetric complexity and relatively simple structure is noticeable [12] and closely related to Cooley-Tukey FFT. The most important thing in this kind of researches is to find an algorithm which we get the minimal operation compared with existing ones. Because it is related to practical use such as the resolution, data transmission and image compression.

We would like to propose a Cooley-Tukey modified FFT in order to obtain better result in speeding up the calculation of the FFT.

2. Cooley-Tukey modified FFT

We would like to modify Cooley-Tukey algorithm which is a kind of FFT algorithm, and we shall call to Cooley-Tukey modified FFT(CTMF). Consider the discrete Fourier transform(DEF)

$$X(n) = \sum_{k=0}^{N-1} x_0(k)e^{-2\pi ikn/N}$$

for $n = 0, 1, \cdots, N-1$. If we let $W = e^{-2\pi i/N}$, then $X(n)$ can be denoted by

$$X(n) = \sum_{k=0}^{N-1} x_0(k)W^{kn}$$

for $n = 0, 1, \cdots, N-1$. From now on, without repeatedly mentioned the scope of n again and again, we assume that it takes $0, 1, \cdots, N-1$. First, we would like to deal with the case $N = 4$.

A. CTMF for $N = 4$

Let us consider the DEF

$$X(n) = \sum_{k=0}^{N-1} x_0(k) W^{kn}$$

for $W = e^{-2\pi i/N}$, and denote integers n and k as binary numbers. Since $N = 4$, k and n take values 0,1,2 and 3 only. Implies, the binary form of k and n take 00, 01, 10 and 11 only. In simpler expression, it is denoted by

$$n = 2n_1 + n_0, \;\; k = 2k_1 + k_0$$

where k_0, k_1, n_0 and n_1 can take the values of 0 and 1 only.

Thus, DEF can be represented by the form

$$X(n_1, n_0) = \sum_{k_1=0}^{1} \sum_{k_0=0}^{1} x_0(n_0, k_0) W^{(2n_1+n_0)(2k_1+k_0)}.$$

Next, let us change $W^{(2n_1+n_0)(2k_1+k_0)}$ to compact form, then we get

$$W^{(2n_1+n_0)(2k_1+k_0)} = W^{4n_1k_1} W^{2n_1k_0} W^{2n_0k_1} W^{n_0k_0} = W^{(2n_1+n_0)k_0} \; W^{2n_0k_1}$$

because of

$$W^{4n_1k_1} = (e^{-8\pi i/4})^{n_1k_1} = (e^{-2\pi i})^{n_1k_1} = 1.$$

Hence $X(n_1, n_0)$ can be denoted as

$$X(n_1, n_0) = \sum_{k_1=0}^{1} [\sum_{k_0=0}^{1} x_0(n_0, k_0) W^{(2n_1+n_0)k_0}] W^{2n_0k_1}.$$

Let us put

$$x_1(n_1, n_0) = \sum_{k_0=0}^{1} x_0(n_0, k_0) W^{(2n_1+n_0)k_0},$$

and enumerating the $x_1(n_1, n_0)$, we have

$$x_1(0,0) = x_0(0,0) + x_0(0,1)W^0$$
$$x_1(0,1) = x_0(1,0) + x_0(1,1)W^1$$
$$x_1(1,0) = x_0(0,0) + x_0(0,1)W^2$$

$$x_1(1,1) = x_0(1,0) + x_0(1,1)W^3$$

where, W^0 is not reduced to 1 in order to develop a generalized result. Let us rewrite this in matrix notation. Then we get

$$\begin{bmatrix} x_1(0,0) \\ x_1(0,1) \\ x_1(1,0) \\ x_1(1,1) \end{bmatrix} = \begin{bmatrix} 1 & w^0 & 0 & 0 \\ 0 & 0 & 1 & w^1 \\ 1 & w^2 & 0 & 0 \\ 0 & 0 & 1 & w^3 \end{bmatrix} = \begin{bmatrix} x_0(0,0) \\ x_0(0,1) \\ x_0(1,0) \\ x_0(1,1) \end{bmatrix}.$$

If we check the number of used operation, it is used 4 additions and 2 multiplications only, whereas the direct method requires 4^2 complex multiplications and 4×3 complex additions. Let us change the topic to the outer sum.

Replace $x_1(n_1, n_0)$ with $x_1(k_1, n_0)$, if we write the outer summation of $X(n_1, n_0)$ as

$$x_2(n_0, n_1) = \sum_{k_1=0}^{1} x_1(k_1, n_0) W^{2n_0 k_1}$$

and enumerating this equality, we get

$$\begin{aligned} x_2(0,0) &= x_1(0,0) + x_1(1,0)W^0 \\ x_2(0,1) &= x_1(0,0) + x_1(1,0)W^0 \\ x_2(1,0) &= x_1(0,1) + x_1(1,1)W^2 \\ x_2(1,1) &= x_1(0,1) + x_1(1,1)W^2. \end{aligned}$$

The matrix form is

$$\begin{bmatrix} x_2(0,0) \\ x_2(0,1) \\ x_2(1,0) \\ x_2(1,1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & w^0 & 0 \\ 1 & 0 & w^0 & 0 \\ 0 & 1 & 0 & w^2 \\ 0 & 1 & 0 & w^2 \end{bmatrix} = \begin{bmatrix} x_1(0,0) \\ x_1(0,1) \\ x_1(1,0) \\ x_1(1,1) \end{bmatrix},$$

and the recursive form is as the following;

$$x_1(n_1, n_0) = \sum_{k_0=0}^{1} x_0(n_0, k_0) W^{(2n_1+n_0)k_0}$$

$$x_2(n_0, n_1) = \sum_{k_1=0}^{1} x_1(k_1, n_0) W^{2n_0 k_1} \; for \; n_1 = k_1$$

$$X(n_1, n_0) = x_2(n_0, n_1).$$

In the second step, 1 multiplication and 2 additions are used. Hence CTMF can be denoted by the following recursive form for $N = 4$;

$$x_1(n_1, n_0) = \sum_{k_0=0}^{1} x_0(n_0, k_0) W^{(2n_1+n_0)k_0}$$

$$x_2(n_0, n_1) = \sum_{k_1=0}^{1} x_1(k_1, n_0) W^{2n_0k_1}$$

$$X(n_1, n_0) = x_2(n_0, n_1)$$

where k_0, k_1, n_0 and n_1 are all integer. An alternative form is simply expressed by

$$\sum_{k_1=0}^{1} [\sum_{k_0=0}^{1} x_0(n_0, k_0) W^{(2n_1+n_0)k_0}] W^{2n_0k_1},$$

based on the above equations we checked.

B. The comparison of the number of operations with related algorithms of FFT for $N = 4$.

In general, for $N = 2^r$, the FFT algorithm is then simply a procedure for factoring an $N \times N$ matrix into r matrices such that each of the factored matrices has the special property of minimizing the number of complex multiplications and additions. If we representing the equation of example 1 by 4×4 matrix, we obtain the following result, where W^0 is not reduced to 1 in order to develop a generalized result.

1) Direct evaluation(DE)
   It requires 4^2 complex multiplications and 4×3 complex addition.
2) Cooley-Tukey FFT(CTF)
   Cooley-Tukey FFT brings the complexity down to $Nlog_2N$ operations. Implies, in the first step, it needs 2 complex multiplications and 4 complex additions, and the second step is equally needed. Thus, $X(n_0, n_1)$ totally requires 4 complex multiplications and 8 complex additions.

TABLE 1. The comparison of the number of operations with related algorithms of FFT for $N = 4$.

Type	DE	CTF	STF	CTTF	CTMF
No. multiplications	16	4	6	6	3
No. additions	12	8	8	8	6

3) Sande-Tukey algorithm in FFT(STF)
Sande-Tukey algorithm is a canonic form of the FFT and this is expressed by

$$\sum_{k_0=0}^{1}[\sum_{k_1=0}^{1} x_0(k_1,k_0)W^{2n_0k_1}W^{n_0k_0}]\ W^{2n_1k_0}.$$

In the first step, it needs 4 complex multiplications and 4 complex additions, and in the second step, it is needed 2 and 4, respectively. Thus, 6 complex multiplications and 8 complex additions are totally required.

4) A representative Cooley-Tukey twiddle factor algorithm(CTTF)
The below algorithm is a representative of twiddle form of the FFT and this is expressed by

$$\sum_{k_0=0}^{1}[\sum_{k_1=0}^{1} x_0(k_1,k_0)W^{(2k_1+k_0)n_0}]W^{2n_1k_0}$$

Similar to 3), in the first step it needs 4 complex multiplications and 4 complex additions, and in the second step, it needs 2 and 4, respectively. Thus, 6 and 8, respectively.

5) The proposed Cooley-Tukey modified FFT(CTMF)
Similarly to 2), it needs 2 complex multiplications and 4 complex additions, and in the second step, it needs 1 and 2, respectively. So totally is needs 3 complex multiplications and 6 complex additions.

Normally, since computing time is proportional to the number of multiplications, we would like to restrict to the case in computing of the approximate ratio. For the number of samples is $N = 4$, the approximate ratio of direct calculation to CTMF is given by

$$4^2/3,$$

and this is reducer than 4 of FFT.

If N is a large sampling number, a computational reduction is worthy of close attention. For example, if $N = 1024 = 2^{10}$, then the approximate ratio of direct calculation to FFT is

$$\frac{2^{10} \cdot 2^{10}}{\frac{1}{2} \cdot 2^{10} \cdot 10} = \frac{2 \cdot 2^{10}}{10} = \frac{4 \cdot 2^8}{5} < 2^8 = 256,$$

and in case of CTMF, we can obtain more reduced result.

Now that we would like to extend to the case in which CTMF has some arbitrary factors in FFT. We would like to confine the case $N = r_1 r_2$ because of easy extension to finite dimension, where r_1 and r_2 are integer valued.

THEOREM 1. *(CTMF for $N = r_1 r_2$) Cooley-Tukey modified algorithm in FFT can be denoted by the following recursive form for $N = r_1 r_2$.*

$$x_1(n_1, n_0) = \sum_{k_0=0}^{r_2-1} x_0(n_0, k_0) w^{(n_1 r_1 + n_0) k_0}$$

$$x_2(n_0, n_1) = \sum_{k_1=0}^{r_1-1} x_1(k_1, n_0) w^{r_2 n_0 k_1}$$

$$X(n_1, n_0) = x_2(n_0, n_1).$$

An alternative form is simply expressed by

$$\sum_{k_1=0}^{r_1-1} [\sum_{k_0=0}^{r_2-1} x_0(n_0, k_0) w^{(n_1 r_1 + n_0) k_0}] w^{r_2 n_0 k_1}.$$

Proof. By letting $W = e^{-2\pi i/N}$, then DFT $X(n)$ was denoted by

$$X(n) = \sum_{k=0}^{N-1} x_0(k) W^{kn}$$

for $n = 0, 1, \cdots, N-1$. Here, let us denote the n and k as

$$n = n_1 r_1 + n_0$$

$$k = k_1 r_2 + k_0$$

where n_0 and k_1 take value in $[0,\ r_1-1]$, n_1 and k_0 in $[0,\ r_2-1]$ and all integer. Then we can rewrite DFT

$$X(n) = \sum_{k=0}^{N-1} x_0(k)W^{kn}$$

as

$$X(n_1, n_0) = \sum_{k_1=0}^{r_1-1}\sum_{k_0=0}^{r_2-1} x_0(n_0, k_0)W^{(n_1r_1+n_0)(k_1r_2+k_0)}$$

where $n_1 = k_1$. Since $w^{r_1r_2} = w^N = 1$, we get

$$W^{(n_1r_1+n_0)(k_1r_2+k_0)} = w^{(n_1r_1+n_0)k_0} \cdot w^{r_2n_0k_1}.$$

Implies, $X(n_1, n_0)$ can be expressed by

$$\sum_{k_1=0}^{r_1-1}[\sum_{k_0=0}^{r_2-1} x_0(n_0, k_0)w^{(n_1r_1+n_0)k_0}]w^{r_2n_0k_1}$$

and if we define the intermediate computational steps, then

$$x_1(n_1, n_0) = \sum_{k_0=0}^{r_2-1} x_0(n_0, k_0)w^{(n_1r_1+n_0)k_0}$$

$$x_2(n_0, n_1) = \sum_{k_1=0}^{r_1-1} x_1(k_1, n_0)w^{r_2n_0k_1}$$

$$X(n_1, n_0) = x_2(n_0, n_1).$$

□

Base 4+2 means that we compute as many arrays as possible with the base-4 algorithm and then compute a base-2 array([1]), but in CTMF, it has the opposite order.

C. The validity of theorem 1 with Base $4+2$ *algorithm of CTMF for* $N = 8$

Henceforth, we would like to develop the base $4 + 2$ algorithm of CTMF. To begin with, let us put $n = 4n_1 + n_0$ and $k = 2k_1 + k_0$ where $n_1 = 0, 1$, $n_0 = 0, 1, 2, 3$, $k_1 = 0, 1, 2, 3$ and $k_0 = 0, 1$. In DEF

$$X(n) = \sum_{k=0}^{N-1} x_0(k)W^{kn},$$

we have

$$W^{nk} = W^{(4n_1+n_0)(2k_1+k_0)} = W^{(4n_1+n_0)k_0}\ W^{2n_0k_1}$$

where $W = e^{-2\pi i/N}$. Implies, DEF becomes

$$X(n_1, n_0) = \sum_{k_1=0}^{3}[\sum_{k_0=0}^{1} x_0(n_0, k_0)W^{(4n_1+n_0)k_0}]W^{2n_0k_1},$$

the base $4 + 2$ algorithm of CTMF, and the inner sum of the above equality becomes

$$\sum_{k_0=0}^{1} x_0(n_0, k_0)W^{(4n_1+n_0)k_0}.$$

If we give detailed explanation, we get

$$\begin{aligned}
x_1(0,0) &= x_0(0,0) + x_0(0,1)W^0\\
x_1(0,1) &= x_0(1,0) + x_0(1,1)W^1\\
x_1(0,2) &= x_0(2,0) + x_0(2,1)W^2\\
x_1(0,3) &= x_0(3,0) + x_0(3,1)W^3\\
x_1(1,0) &= x_0(0,0) + x_0(0,1)W^4\\
x_1(1,1) &= x_0(1,0) + x_0(1,1)W^5\\
x_1(1,2) &= x_0(2,0) + x_0(2,1)W^6\\
x_1(1,3) &= x_0(3,0) + x_0(3,1)W^7
\end{aligned}$$

where, W^0 is not reduced to 1 in order to develop a generalized result.

Replace $x_1(n_1, n_0)$ with $x_1(k_1, n_0)$, and if we write the outer summation of $X(n_1, n_0)$ as

$$x_2(n_0, n_1) = \sum_{k_1=0}^{3} x_1(k_1, n_0)W^{2n_0k_1}$$

and enumerating this equality, we get

$$\begin{aligned}
x_2(0,0) &= x_1(0,0) + x_1(1,0)W^0 + x_1(2,0)w^0 + x_1(3,0)w^0 = x_2(0,1)\\
x_2(1,0) &= x_1(0,1) + x_1(1,1)W^2 + x_1(2,1)w^4 + x_1(3,1)w^6 = x_2(1,1)\\
x_2(2,0) &= x_1(0,2) + x_1(1,2)W^4 + x_1(2,2)w^0 + x_1(3,2)w^4 = x_2(2,1)\\
x_2(3,0) &= x_1(0,3) + x_1(1,3)W^6 + x_1(2,3)w^4 + x_1(3,3)w^2 = x_2(3,1).
\end{aligned}$$

On the other hand, in theorem 1, let us put $r_2 = 2$ and $r_1 = 4$. Then, we obtain

$$X(n_1, n_0) = \sum_{k_1=0}^{3} [\sum_{k_0=0}^{1} x_0(n_0, k_0) W^{(4n_1+n_0)k_0}] W^{2n_0k_1},$$

and this is the same result with the above equation.

3. Conclusion

We have showed that CTMF affords better result compared with existing algorithms in speeding up the calculation of the FFT. This means that we can get faster things in the resolution, data transmission and image compression.

References

[1] G. Beylkin, *On the fast Fourier transform of functions with singularities*, Appl. Comput. Harmon. Anal. **2** (1995), 363–381.
[2] E.O. Brigham, *The fast Fourier transform and its applications*, Prentice-Hall International Inc. New Jersey, 1988.
[3] E. Candes, L. Demanet and L. Ying, *Fast computation of Fourier integral operators*, Math. NA, 0610051v1(2006), 1-31.
[4] P. Carr and D.B. Madan, *Option valuation using the fast Fourier transform*, Journal of Computational Finance **2** (1999), no. 4, 1–18.
[5] J.W. Cooley, and J.W. Tukey, *An algorithm for the machine calculation of complex Fourier series*, Math. Comp. **19** (1965), no 90, 297–301.
[6] M.E. Deisher and A.S. Spanias, *Adaptive noise cancellation using fast optimum block algorithms*, IEEE Int. Symposium on Circuits and Systems **1**, 698–701, 1991.
[7] D.L. Jones, *A resolution comparison of several time-frequency representations*, IEEE Trans. Signal Process. **40** (1992), no. 2, 413–420.
[8] A. Oppenheim and D. Johnson, *Computation of spectra with unequal resolution using the fast Fourier transform*, Proc. IEEE **59** (1971), 299–301.
[9] P. Duhamel and H. Hollmann, *Spilt-radix FFT algorithm*, Electronics Letters **20** (1984), no.1, 14–16.
[10] P. Duhamel and M. Vetterli, *Fast Fourier transforms: a tutorial review and a state of the art*, Signal Processing **19** (1990), 259–299.
[11] L. Greengard and JY. Lee, *Accelerating the nonuniform fast Fourier transform*, SIAM Rev. **46** (2004), no. 3, 443–454.
[12] S. Winograd, *On computing the discrete Fourier transform*, Math. Comp. **32** (1978), no. 141, 175–199.

[13] X.G. Xia, *Discrete Chirp-Fourier transform and its application to Chirp rate estimation*, IEEE Trans. Signal Process. **48** (2000), no. 11, 3122–3133.

Department of Computational Mathematics
in Rangsit University, Thailand
E-mail: cellmath@gmail.com

Department of Computational Mathematics
in Rangsit University, Thailand
E-mail: somchai@rangsit.rsu.ac.th