Mathematical Problems of Computer Science 32, 39–44, 2009.

A Modified Algorithm of Fast Fourier Transform

Rafayel V. Barseghyan

Institue for Informatics and Automation Problems of NAS of RA
barseghyan@gmail.com

Abstract

In this paper we present a new, efficient modification of split-radix algorithm for computing a power of four fast Fourier transforms.

1. Introduction

People, when their imagination is captured by an idea, can become energized and deeply involved, producing remarkable results and reaching new heights that lead to the creation of landmarks in the progress of man. The fast Fourier Transform (FFT) was such an idea. Cooley and Tukey published their historic paper on the computation of the Fourier transform in 1965. Overnight, in universities and laboratories around the world, scientists and engineers began developing computer programs and electronic circuits to implement the FFT. The FFT is a brilliant technique for computing the discrete Fourier (DFT) transform quickly. By recognizing that the Fourier transform of a sequence can be derived from the Fourier transforms of two half length sequences more economically than if the whole sequence is transformed directly and by carrying this concept through to its logical conclusion of evaluating only the direct transform of sequences of two terms, Cooley and Tukey showed that the FFT required only $O(N \log N)$ operations while the direct form took $O(N^2)$ operations.

All known FFT algorithms compute the discrete Fourier transform of size N in $O(N \log N)$ operations, so any improvement in them appears to rely on reducing the exact number or cost of these operations rather than their asymptotic functional form [1], [2]. For many years, the time to perform an FFT was dominated by real-number arithmetic, and so considerable effort was devoted towards proving and achieving lower bounds on the exact count of arithmetic operations (real additions and multiplications),called "flops" (floating-point operations), required for a DFT of a given size [3]. Although the performance of FFTs on recent computer hardware is determined by many factors besides pure arithmetic counts, there still remains an intriguing unsolved mathematical question: what is the smallest number of flops required to compute a DFT of a given size N?

Let $x = (x_0, x_1, \ldots, x_{N-1})$ be an original column-vector with complex components (N is the power of four). DFT of this vector is determined by the following formula

$$X[n] = \frac{1}{N} \sum_{k=0}^{N-1} x[k] W_N^{nk}, \qquad n = 0, 1, \ldots, N-1, \tag{1}$$

where $W_N = \exp(-j\frac{2\pi}{N}) = \cos\frac{2\pi}{N} - j\sin\frac{2\pi}{N}, \quad j = \sqrt{-1}$.

From (1) we can write (without coefficients $1/N$)

$$\begin{aligned} X[n] &= \sum_{k=0}^{N/2-1} x[k]W_N^{nk} + \sum_{k=0}^{N/2-1} x[k+N/2]W_N^{n(k+N/2)} \\ &= \sum_{k=0}^{N/2-1} (x[k] + (-1)^n x[k+N/2])W_N^{nk}, \quad n = \overline{0, N-1}. \end{aligned}$$

Hence we find

$$\begin{aligned} X[2n] &= \sum_{k=0}^{N/2-1} (x[k] + x[k+N/2])W_N^{nk}, \\ X[2n+1] &= \sum_{k=0}^{N/2-1} (x[k] - x[k+N/2])W_N^{k}W_N^{nk}, \quad n = \overline{0, N/2-1} \end{aligned} \tag{2}$$

Formula (2) shows that the coefficients of N - point DFT can be computed using the two $N/2$–point DFTs. It can be easily shown by the process of decomposition that the DFT matrix can be represented as the product of sparse matrices

$$F_N = P_n B_n A_{n-1} B_{n-1} \cdots A_2 B_2 A_1 B_1, \tag{3}$$

where

$$A_r = I_{2^{r-1}} \otimes I_{2^{n-r}} \oplus W^0_{2^{n-r+1}} \oplus W^1_{2^{n-r+1}} \oplus \cdots \oplus W^{2^{n-r}-1}_{2^{n-r+1}}, \quad r = \overline{1, n-1},$$

$$B_i = I_{2^{i-1}} \otimes H_2 \otimes I_{2^{n-i}}, \quad i = \overline{1, n}, \quad H_2 = \begin{pmatrix} + & + \\ + & - \end{pmatrix},$$

and $\otimes$ - sign of multiplication of Kronecker, $\oplus$ - sign of a direct sum of matrices, and P_N - bit reversal matrix.

Using the decomposition (3) one can calculate the count of necessary operations (real additions and multiplications, $N > 4$) for the implementation of fast Fourier transform [4]

$$R_N^+ = 3N\log_2 N - 2N + 2, \qquad R_N^\times = 2N\log_2 N - 4N + 4, \quad (R_4^+ = 16,\ R_4^\times = 0) \tag{4}$$

2. Modified FFT

Since we assumed that the length of N is power of four, the second sum from (2) can be represented in the following form

$$\begin{aligned} X[2n+1] &= \sum_{k=0}^{N/4-1} (x[k] - x[k+N/2])W_N^k W_{N/2}^{nk} \\ &+ \sum_{k=0}^{N/4-1} (x[k+N/4] - x[k+N/4+N/2])W_N^{k+N/4} W_{N/2}^{n(k+N/4)} \end{aligned} \tag{5}$$

But since $W_N^{N/4} = -j, \quad W_{N/2}^{nN/4} = (-1)^n$, (5) takes the following form

$$\begin{aligned} X[2n+1] &= \sum_{k=0}^{N/4-1} (x[k] - x[k+N/2])W_N^k W_{N/2}^{nk} \\ &-j(-1)^n \sum_{k=0}^{N/4-1} (x[k+N/4] - x[k+N/4+N/2])W_N^k W_{N/2}^{nk}. \end{aligned}$$

Hence we find

$$X[4n+1] = \sum_{k=0}^{N/4-1} [(x[k] - x[k+N/2]) - j(x[k+N/4] - x[k+N/4+N/2])]\, W_N^k W_{N/4}^{nk},$$

$$X[4n+3] = \sum_{k=0}^{N/4-1} [(x[k] - x[k+N/2]) + j(x[k+N/4] - x[k+N/4+N/2])]\, W_N^{3k} W_{N/4}^{nk},$$

where $n = \overline{0, N/4-1}$. Thus, the N-point DFT can be computed according to the following formulas

$$\begin{aligned} X[2n] &= \sum_{k=0}^{N/2-1} (x[k] + x[k+N/2]) W_{N/2}^{nk}, \qquad n = \overline{0, N/2-1}, \\ X[4n+1] &= \sum_{k=0}^{N/4-1} [(x[k] - x[k+N/2]) - j(x[k+N/4] - x[k+N/4+N/2])]\, W_N^k W_{N/4}^{nk}, \\ X[4n+3] &= \sum_{k=0}^{N/4-1} [(x[k] - x[k+N/2]) + jx([k+N/4] - x[k+N/4+N/2])]\, W_N^{3k} W_{N/4}^{nk}, \end{aligned} \tag{6}$$

in last two equations $n = \overline{0, N/4-1}$.

(6) implies that the computation N -point DFT is reduced to the calculation of one $\frac{N}{2^{2i-1}}$ -point and two $\frac{N}{4^i}$-point DFT, but first it is required to define three new sequences

$$v_1: \quad \{x[k] + x[k+N/2]\}_{k=0}^{N/2-1},$$

$$v_2: \quad \left\{([x[k] - x[k+N/2]) - jx([k+N/4] - x[k+N/4+N/2])]\, W_N^k\right\}_{k=0}^{N/4-1},$$

$$v_3: \quad \left\{([x[k] - x[k+N/2]) + jx([k+N/4] - x[k+N/4+N/2])]\, W_N^{3k}\right\}_{k=0}^{N/4-1}.$$

To determine the members of the v_1 sequence N - real additions must be performed. And to define the elements of the v_2 and v_3 sequences $(2N-2)$ real additions, $(N-6)$ real multiplications and $(N-2)$ real additions, $(N-6)$ multiplications are required.

Now we define the count of operations of N-point DFT according to the formula (6) using the FFT estimates by formula (4).

$$\begin{aligned} 1:&\quad C^+_{\frac{N}{2}} = \tfrac{3}{2}N\log_2 N - \tfrac{3}{2}N + 2, &\quad C^\times_{\frac{N}{2}} &= N\log_2 N - 3N + 4, \\ 2:&\quad C^+_{\frac{N}{4}} = \tfrac{3}{4}N\log_2 N, & C^\times_{\frac{N}{4}} &= \tfrac{1}{2}N\log_2 N - N - 2, \\ 3:&\quad C^+_{\frac{N}{4}} = \tfrac{3}{4}N\log_2 N - N, & C^\times_{\frac{N}{4}} &= \tfrac{1}{2}N\log_2 N - N - 2. \end{aligned}$$

Thus, the complexity of real-number operations for N-point DFT by the formula (6) is determined from the following formulas

$$C^+_N = 3N\log_2 N - \tfrac{5N}{2} + 2, \qquad C^\times_N = 2N\log_2 N - 5N.$$

Algorithm for the input complex-data vector of length $N = 4^k$ is illustrated in the end of this paper (Figure 1).

Continuing the process of decomposition of the form (6) in i-step iteration we will get triple of vectors of length $\frac{N}{2^{2i-1}}$, $\frac{N}{4^i}$ and $\frac{N}{4^i}$.For defining the components of vectors in the i-step we need to perform $(i = \overline{1, k-1})$

$$A_1^i = \tfrac{N}{2^{2i-2}}, \quad A_2^i = \tfrac{N}{2^{2i-3}} - 2, \quad A_3^i = \tfrac{N}{2^{2i-2}} - 2 \quad \text{additions}$$
$$M_1^i = 0, \quad M_2^i = \tfrac{N}{2^{2i-2}} - 6, \quad M_3^i = \tfrac{N}{2^{2i-2}} - 6 \quad \text{multiplications.} \tag{7}$$

Note that i = k-1 requires only 8 - and 4-point FFT, which according to the formula (4) require the $R_8^+ = 58$, $R_8^\times = 20$, $R_4^+ = 16$, $R_4^\times = 0$ operations and the number of transactions forming respective vectors are determined from (7) and are listed below

$$\begin{aligned} &A_1^{k-1} = 16, \quad A_2^{k-1} = 30, \quad A_3^{k-1} = 14, \\ &M_1^{k-1} = 0, \quad M_2^{k-1} = 10, \quad M_3^{k-1} = 10. \end{aligned} \tag{8}$$

Thus, the count of operations of additions and multiplications of the new FFT algorithm is determined by formulas of type

$$C_N^+ = \sum_{i=1}^{k-2} 2^{i-1}(A_1^i + A_2^i + A_3^i + R^+(2i-1)) + 2^{k-2}((R_8^+ + 2R_4^+ + A_1^{k-1} + A_2^{k-1} + A_3^{k-1})),$$
$$C_N^\times = \sum_{i=1}^{k-2} 2^{i-1}(M_2^i + M_3^i + R^\times(2i-1)) + 2^{k-2}((R_8^\times + 2R_4^\times + M_2^{k-1} + M_3^{k-1})),$$

where $R^{op}(t)$-the estimate of operations in $N/2^t$-point FFT. Given (8) finally find

$$\begin{aligned} C_N^+ &= \sum_{i=1}^{k-2} 2^{i-1}(A_1^i + A_2^i + A_3^i + R^+(2i-1)) + 2^{k-2} \cdot 150, \\ C_N^\times &= \sum_{i=1}^{k-2} 2^{i-1}(M_2^i + M_3^i + R^\times(2i-1)) + 2^{k-2} \cdot 40. \end{aligned} \tag{9}$$

Find

$$\begin{aligned} &A_1^i + A_2^i + A_3^i = \tfrac{16N}{2^{2i}} - 4, \qquad M_2^i + M_3^i = \tfrac{8N}{2^{2i}} - 12, \\ &R^+(2i-1) = 6N/2^{2i}\lg_2 N - 12i/2^{2i}N + 2N/2^{2i} + 2, \\ &R^\times(2i-1) = 4N/2^{2i}\lg_2 N - 8i/2^{2i}N + 4N/2^{2i} + 4. \end{aligned} \tag{10}$$

Substituting (10) into (9) we finally find

$$C_N^+ = 3(1 - \tfrac{1}{2^{k-2}})N\log_2 N + 9N(1 - \tfrac{1}{2^{k-2}}) - 6NS(k) + 37 \cdot 2^k + 2,$$
$$C_N^\times = 2(1 - \tfrac{1}{2^{k-2}})N\log_2 N + 6N(1 - \tfrac{1}{2^{k-2}}) - 4NS(k) + 8 \cdot 2^k + 8,$$

where S(3) = 1 / 2, S (4) = 1, S(5) = 11 / 8, S(6) = 13 / 8.

The scheme of the algorithm for the 64-point Fourier transforms (records over the edge (m, n) imply: m-the count of additions, n - the count of multiplications) is illustrated below

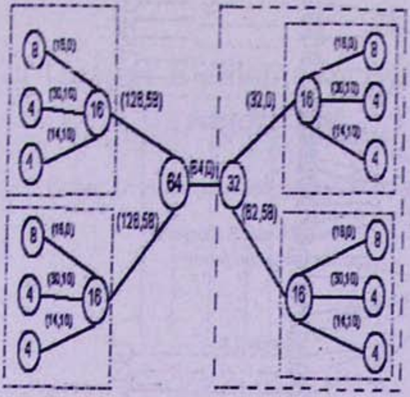

Figure 2: Algorithm for a 64-point DFT.

The calculation of the count of operations of FFT for 64-point complex input vector.

$$C_4^+ = 16,\ C_8^+ = 52,\ C_{16}^+ = 144,\ C_{32}^+ = 382,\ C_{64}^+ = 922,$$
$$C_4^\times = 0,\quad C_8^\times = 4,\quad C_{16}^\times = 24,\quad C_{32}^\times = 82,\quad C_{64}^\times = 246.$$

In practice, we have to deal with the source vectors with real components. In this case, we get better estimates for the count of operations.

Comparison of operation counts for standard complex-data split-radix algorithm and our new algorithm are described in the following table.

N	FFT for $N = 2^k$		A modified FFT for $N = 4^k$		Acceleration	
	Add.	Mul.	Add.	Mul.	Add.	Mul.
16	177	68	144	24	1.23	2.83
32	449	196	382	82	1.18	2.39
64	1089	516	922	246	1.18	2.1

Add. - count of real additions, Mul.- count of real multiplications.

References

[1] D. E. Knuth, Fundamental Algorithms, 3rd ed., ser. "The Art of Computer Programming". Addison-Wesley, vol. 1, 1997.

[2] P. Duhamel and M. Vetterli, "Fast Fourier transforms: a tutorial review and a state of the art", Signal Processing - vol. 19, pp. 259–299, 1990.

[3] M. Frigo and S. G. Johnson, "A modified split-radix FFT with fewer arithmetic operations", IEEE TRANS. SIGNAL PROCESSING - vol. 55, pp. 111-119, 2207.

[4] H.Sarukhanyan, S.Agaian, "Conventional, Integer to Integer and Quantized Fast Fourier Transforms", CSIT - p.p. 204-207, 2007.

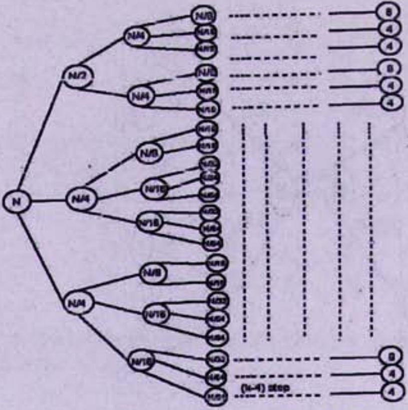

Figure 1:The modified algorithm FFT for $N = 4^k$

ՖԱՁ ալգորիթմի մի մոդիֆիկացիայի մասին

Ռ. Բարսեղյան

Ամփոփում

Աշխատանքում մշակված է $N = 4^k$- չափանի վկտորի Ֆուրյեի արագ ձևափոխության նոր, առավել արդյունավետ ալգորիթմ: