JOURNAL OF COMPUTATIONAL PHYSICS 75, 190–198 (1988)

A New Set of Minimum-Add Small-n Rotated DFT Modules

CLIVE TEMPERTON

Division de recherche en prévision numérique,
Service de l'Environnement Atmosphérique,
2121 Trans-Canada Highway,
Dorval, Québec, Canada H9P 1J3

Received October 24, 1986; revised April 1, 1987

A self-sorting in-place prime factor FFT algorithm introduced in an earlier paper was based on a set of modules for performing small discrete Fourier transforms (DFTs) with an optional "rotation" of the results. The modules were designed to reduce the total number of additions in the FFT algorithm. In specializing this algorithm to the case of real or conjugate-symmetric input data, it was found necessary to redesign these modules to impose a particular structure. In some cases, the new modules proved to require fewer operations than the old. We describe the design procedure and compare the new operation counts with those for Winograd's DFT modules. The algorithms for the new modules are given in detail; these will be useful in coding FFT algorithms of either the "prime factor" or conventional variety.

1. INTRODUCTION

In a previous paper [7], the author described a self-sorting, in-place complex FFT (fast Fourier transform) algorithm based on the decomposition of N (the length of the transform) into mutually prime factors. This algorithm was developed from those of Burrus and Eschenbacher [1] and Rothweiler [2], but departed from theirs in two important respects: the indexing scheme was quite different (and considerably simpler), and the algorithm was designed to reduce the number of *additions* rather than the number of multiplications. The implementation of this algorithm on the Cray-1 was described in [8], where it was shown to be up to 32% faster than the conventional FFT.

At the heart of this algorithm is a set of "rotated" small-n DFT (discrete Fourier transform) modules, each of which performs a transform

$$x_j = \sum_{k=0}^{n-1} z_k \omega_n^{jkr} \qquad (0 \leqslant j \leqslant n-1), \tag{1}$$

where

$$\omega_n = \exp(2\pi i/n)$$

0021-9991/88 $3.00

and r is any integer mutually prime to n. The set of permissible values of n was $\{2, 3, 4, 5, 7, 8, 9, 16\}$, as in most published "prime factor" FFT algorithms. These DFT modules differ from those of Winograd [10] in explicitly allowing for the possibility of rotation ($r \neq 1$) and in minimizing the number of additions (at the expense of extra multiplications, which are, however, "free" in the context of implementation on machines such as the Cray-1 [6]).

Most applications of the FFT in computational fluid dynamics actually require transforms between the real data in physical space and the half-complex (conjugate-symmetric) data in transform space. The self-sorting, in-place prime factor complex FFT described in [7] has now been specialized to the case of real/half-complex transforms; an account of this new algorithm is given in [9]. In the course of this work, it was found necessary to redesign the set of rotated DFT modules of [7] in order to give them a particular structure. Specifically, Eq. (1) may be rewritten in matrix/vector form as

$$\mathbf{x} = W_n^{[r]}\mathbf{z},$$

where $W_n^{[r]}$ is the matrix with element (j, k) given by ω_n^{jkr} (rows and columns of matrices are indexed here from 0 to $n-1$). The required structure corresponds to a decomposition of the form

$$W_n^{[r]} = X_n V_n^{[r]}, \tag{2}$$

where the elements of $V_n^{[r]}$ are all pure real numbers, and X_n is defined as follows: if $\mathbf{x} = X_n\mathbf{y}$, then

$$\begin{aligned}
x_0 &= y_0; & & \\
x_{n/2} &= y_{n/2} & & (\text{if } n \text{ is even}); \\
x_j &= y_j + iy_{n-j} & & (1 \leqslant j < n/2) \\
x_{n-j} &= y_j - iy_{n-j} & & (1 \leqslant j < n/2).
\end{aligned}$$

Since $V_n^{[r]}$ is a real matrix, any multiplications by imaginary quantities during the rotated DFT algorithm are delayed until the final stage, corresponding to the multiplication by the matrix X_n.

Now, the rotated DFT modules described in [7] for $n = 2, 3, 4, 5$, and 7 were already in the required form, but those for $n = 8, 9, 16$ were not. A systematic procedure (outlined in Section 2) was developed for constructing modules with the required structure, and the full set of modules is given in Section 3. As noted later, the new modules were improvements on those given in [7], requiring the same number of additions but fewer logical operations ($n = 8$, 16) or fewer multiplications ($n = 7, 9$).

While the specific structure of Eq. (2) was necessary for specializing the prime factor FFT algorithm to the case of real/half-complex transforms, the new set of modules will be useful not only for coding prime factor FFT algorithms, but also (with $r=1$, i.e., no rotation) for coding conventional mixed-radix FFT algorithms [4].

2. Redesign of DFT Modules

Any complex transform defined by Eq. (1) can be computed by separately transforming the real and imaginary parts of the input data and then combining the results. We will show here that this procedure is the key to obtaining an algorithm with the required structure.

Suppose that in Eq. (1) we split the input data into its real and imaginary components: thus $z_k = a_k + ib_k$, where a_k and b_k are both real. Now if we calculate separately

$$g_j = \sum_{k=0}^{n-1} a_k \omega_n^{jkr}, \qquad h_j = \sum_{k=0}^{n-1} b_k \omega_n^{jkr}, \qquad 0 \leqslant j \leqslant n-1, \tag{3}$$

then the sequences $\{g_j\}$, $\{h_j\}$ will each be conjugate-symmetric; for example,

$$\begin{aligned} g_{n-j} &= \sum_{k=0}^{n-1} a_k \omega_n^{(n-j)kr} = \sum_{k=0}^{n-1} a_k (\omega_n^{jkr})^* \\ &= \left(\sum_{k=0}^{n-1} a_k \omega_n^{jkr} \right)^* = g_j^*, \end{aligned}$$

since $\omega_n^n = 1$, $\omega^{-jkr} = (\omega^{jkr})^*$, and the a_k's are real. Let us define n *real* quantities p_j as follows: for $0 \leqslant j \leqslant n/2$, p_j is the real part of g_j; while for $1 \leqslant j < n/2$, p_{n-j} is the imaginary part of g_j. Thus

$$p_0 = g_0; \tag{4}$$

$$p_{n/2} = g_{n/2} \qquad \text{if } n \text{ is even}; \tag{5}$$

$$p_j = \tfrac{1}{2}(g_j + g_j^*), \qquad 1 \leqslant j < n/2; \tag{6}$$

$$p_{n-j} = -\tfrac{1}{2} i (g_j - g_j^*), \qquad 1 \leqslant j < n/2. \tag{7}$$

Similarly, define n real quantities q_j in terms of the h_j's. Finally, set

$$x_0 = p_0 + iq_0; \tag{8}$$

$$x_{n/2} = p_{n/2} + iq_{n/2} \qquad \text{if } n \text{ is even}; \tag{9}$$

$$x_j = (p_j + iq_j) + i(p_{n-j} + iq_{n-j}) \qquad \text{for } 1 \leqslant j < n/2, \tag{10}$$

$$x_{n-j} = (p_j + iq_j) - i(p_{n-j} + iq_{n-j}) \qquad \text{for } 1 \leqslant j < n/2. \tag{11}$$

It is easily verified from (4)–(7) and (8)–(11) that

$$x_j = g_j + ih_j \tag{12}$$

for $0 \leqslant j \leqslant n-1$, and substituting (3) into (12) we see that we have computed the transform defined in Eq. (1).

Since the transforms defined by Eq. (3) are real to half-complex, and the results of each transform have been interpreted as n real numbers, they can be written as

$$\mathbf{p} = V_n^{[r]}\mathbf{a}, \qquad \mathbf{q} = V_n^{[r]}\mathbf{b},$$

where the elements of $V_n^{[r]}$ are all real. Thus

$$\mathbf{p} + i\mathbf{q} = V_n^{[r]}(\mathbf{a} + i\mathbf{b}) = V_n^{[r]}\mathbf{z}. \tag{13}$$

Finally, Eqs. (8)–(11) correspond to

$$\mathbf{x} = X_n(\mathbf{p} + i\mathbf{q}). \tag{14}$$

Combining (13) and (14) shows that we have constructed a decomposition of the form (2).

However, we still need a representation of the matrix $V_n^{[r]}$ in efficient algorithmic form. This may be obtained as follows. Algorithms for multiplication by $W_n^{[r]}$ were defined in [7], but did not necessarily have the required structure. If we apply such an algorithm to *real* input data we find that about half the computations are redundant, as in [5]. Pruning the redundant operations and arranging the results in the required order yields an algorithm for multiplication by $V_n^{[r]}$.

The rotated DFT modules constructed using this procedure for $n = 8, 9, 16$ turned out to be improvements on those given in [7]. For $n = 8, 16$ it became evident that the number of multiplications by ± 1 (treated as logical operations on sign bits) could be reduced, while the rearrangement for $n = 9$ revealed that four of the original 40 multiplications could be saved by combining them with other multiplications. In each case the number of additions remained the same as in [7].

Since the time that the new modules were used in the real/half-complex transform package described in [9], further reductions have been noted in the numbers of multiplications required. In the $n = 7$ module, a simple way of saving 4 of the original 36 real multiplications occurred to the author while this paper was being written. Very recently, Suzuki *et al.* [3] have presented a new radix-3 algorithm which, for $n = 9$, requires 32 real multiplications. Generalizing their algorithm to include rotations and then subjecting it to the design procedure described above showed that 4 of these multiplications could be saved.

The final total of operation counts for each rotated DFT module is shown in Table I, and compared with the corresponding operation counts for the Winograd [10] DFT modules. The logical operations in the modules for $n = 4, 8, 16$ are only required for rotations and would not be needed if the modules were used as part of a "conventional" mixed-radix transform algorithm. For

TABLE I

Real Operation Counts for Small-n DFT Modules

	Minimum-add			Winograd	
n	Adds	Mults	Logical	Adds	Mults
2	4	0	0	4	0
3	12	4	—	12	4
4	16	0	2	16	0
5	32	12	—	34	10
7	60	32	—	72	16
8	52	4	4	52	4
9	80	28	—	88	20
16	144	24	6	148	20

$n = 2, 3, 4, 8$ the addition and multiplication counts for the "minimum-add" modules are the same as for the Winograd modules; for $n = 5, 7, 9, 16$ some additions are saved at the cost of extra multiplications. It is interesting that the minimum-add and Winograd (minimum-multiply) modules now require the same *total* number of arithmetic operations, except for $n = 7$.

A different set of algorithms may be obtained by noting that since the matrix $W_n^{[r]}$ is symmetric, Eq. (2) can be transposed into the form

$$W_n^{[r]} = U_n^{[r]} X_n^{\mathrm{T}}, \tag{15}$$

where $U_n^{[r]}$ is just the transpose of $V_n^{[r]}$, and again has all its elements real. This alternative form was also found useful in developing a real/half-complex version of the self-sorting, in-place prime factor FFT algorithm [9]. The procedure for constructing an algorithm for multiplication by $U_n^{[r]}$ involves applying the original algorithm of [7] to conjugate-symmetric input data and pruning the redundant operations. In all cases it was found that the operation counts for the algorithm defined by (15) were the same as those for the algorithm defined by (2).

3. The New Set of Modules

Since the construction of the rotated minimum-add DFT modules is a nontrivial task, this section is devoted to a complete specification of the set of algorithms having the structure defined by Eq. (2) and incorporating the latest improvements.

Each of the modules in the set computes

$$x_j = \sum_{k=0}^{n-1} z_k \omega_n^{jkr} \qquad (0 \leqslant j \leqslant n-1),$$

where $\omega_n = \exp(2\pi i/n)$, x_j and z_k are complex, and r is an integer mutually prime to n. The algorithms are defined in terms of complex numbers; but all multiplications are by real constants, except for multiplications by i in the final stage.

For purposes of comparison, the modules are given below in a style similar to that usually adopted for the Winograd modules [10]. To verify that they are mathematically correct, Fortran versions of all the algorithms have been used and tested in an extended version of the self-sorting in-place prime factor complex FFT routine given in the Appendix of [7].

(a) $n=2$ $(r=1)$: $x_0 = z_0 + z_1$; $x_1 = z_0 - z_1$.

(b) $n=3$ $(r=1, 2)$: $\theta = 2\pi/3$; $c_1 = \sin(r\theta)$.

$$t_1 = z_1 + z_2; \quad y_0 = z_0 + t_1; \quad y_1 = z_0 - \tfrac{1}{2}t_1; \quad y_2 = c_1(z_1 - z_2);$$

$$x_0 = y_0; \quad x_1 = y_1 + iy_2; \quad x_2 = y_1 - iy_2.$$

(c) $n=4$ $(r=1, 3)$: $\theta = \pi/2$; $c_1 = \sin(r\theta) = \pm 1$.

$$t_1 = z_0 + z_2; \quad t_2 = z_1 + z_3; \quad y_1 = z_0 - z_2;$$

$$y_3 = c_1(z_1 - z_3); \quad y_0 = t_1 + t_2; \quad y_2 = t_1 - t_2;$$

$$x_0 = y_0; \quad x_1 = y_1 + iy_3; \quad x_2 = y_2; \quad x_3 = y_1 - iy_3.$$

(d) $n=5$ $(r=1, 2, 3, 4)$:

$$\theta = 2\pi/5; \quad c_1 = \tfrac{1}{2}[\cos(r\theta) - \cos(2r\theta)] = \pm\sqrt{5}/4;$$

$$c_2 = \sin(r\theta); \quad c_3 = \sin(2r\theta)$$

$$t_1 = z_1 + z_4; \quad t_2 = z_2 + z_3; \quad t_3 = z_1 - z_4; \quad t_4 = z_2 - z_3;$$

$$t_5 = t_1 + t_2; \quad t_6 = c_1(t_1 - t_2); \quad t_7 = z_0 - \tfrac{1}{4}t_5;$$

$$y_0 = z_0 + t_5; \quad y_1 = t_7 + t_6; \quad y_2 = t_7 - t_6;$$

$$y_3 = c_3 t_3 - c_2 t_4; \quad y_4 = c_2 t_3 + c_3 t_4;$$

$$x_0 = y_0; \quad x_1 = y_1 + iy_4; \quad x_2 = y_2 + iy_3; \quad x_3 = y_2 - iy_3; \quad x_4 = y_1 - iy_4.$$

(e) $n=7$ $(r=1, 2, 3, 4, 5, 6)$:

$$\theta = 2\pi/7; \quad c_1 = \cos(r\theta); \quad c_2 = \cos(2r\theta); \quad c_3 = \cos(3r\theta);$$

$$c_4 = \sin(r\theta); \quad c_5 = \sin(2r\theta); \quad c_6 = \sin(3r\theta);$$

$$t_1 = z_1 + z_6; \quad t_2 = z_2 + z_5; \quad t_3 = z_3 + z_4;$$

$$t_4 = z_1 - z_6; \quad t_5 = z_2 - z_5; \quad t_6 = z_3 - z_4;$$

$$t_7 = z_0 - \tfrac{1}{2}t_3; \quad t_8 = t_1 - t_3; \quad t_9 = t_2 - t_3;$$

$$y_0 = z_0 + t_1 + t_2 + t_3; \quad y_1 = t_7 + c_1 t_8 + c_2 t_9;$$

$y_2 = t_7 + c_2 t_8 + c_3 t_9$; $\quad y_3 = t_7 + c_3 t_8 + c_1 t_9$;

$y_4 = c_6 t_4 - c_4 t_5 + c_5 t_6$; $\quad y_5 = c_5 t_4 - c_6 t_5 - c_4 t_6$;

$y_6 = c_4 t_4 + c_5 t_5 + c_6 t_6$;

$x_0 = y_0$; $\quad x_1 = y_1 + iy_6$; $\quad x_2 = y_2 + iy_5$; $\quad x_3 = y_3 + iy_4$;

$x_4 = y_3 - iy_4$; $\quad x_5 = y_2 - iy_5$; $\quad x_6 = y_1 - iy_6$.

(f) $n = 8$ $(r = 1, 3, 5, 7)$: $\theta = \pi/4$; $\quad c_1 = \sin(2r\theta) = \pm 1$;

$c_2 = \cos(r\theta) = \pm 1/\sqrt{2}$; $\quad c_3 = c_1 c_2$;

$t_1 = z_0 + z_4$; $\quad t_2 = z_0 - z_4$; $\quad t_3 = z_1 + z_5$; $\quad t_4 = z_1 - z_5$;

$t_5 = z_2 + z_6$; $\quad t_6 = c_1(z_2 - z_6)$; $\quad t_7 = z_3 + z_7$; $\quad t_8 = z_3 - z_7$;

$t_9 = t_1 + t_5$; $\quad t_{10} = t_3 + t_7$; $\quad t_{11} = c_2(t_4 - t_8)$; $\quad t_{12} = c_3(t_4 + t_8)$;

$y_0 = t_9 + t_{10}$; $\quad y_1 = t_2 + t_{11}$; $\quad y_2 = t_1 - t_5$; $\quad y_3 = t_2 - t_{11}$;

$y_4 = t_9 - t_{10}$; $\quad y_5 = t_{12} - t_6$; $\quad y_6 = c_1(t_3 - t_7)$; $\quad y_7 = t_{12} + t_6$;

$x_0 = y_0$; $\quad x_1 = y_1 + iy_7$; $\quad x_2 = y_2 + iy_6$; $\quad x_3 = y_3 + iy_5$;

$x_4 = y_4$; $\quad x_5 = y_3 - iy_5$; $\quad x_6 = y_2 - iy_6$; $\quad x_7 = y_1 - iy_7$.

(g) $n = 9$ $(r = 1, 2, 4, 5, 7, 8)$: $\theta = 2\pi/9$; $c_1 = \sin(3r\theta)$;

$c_2 = \cos(r\theta)$; $\quad c_3 = \sin(r\theta)$; $\quad c_4 = \cos(2r\theta)$; $\quad c_5 = \sin(2r\theta)$;

$c_6 = c_1 c_2$; $\quad c_7 = c_1 c_3$; $\quad c_8 = c_1 c_4$; $\quad c_9 = c_1 c_5$;

$t_1 = z_3 + z_6$; $\quad t_2 = z_0 - \frac{1}{2} t_1$; $\quad t_3 = c_1(z_3 - z_6)$; $\quad t_4 = z_0 + t_1$;

$t_5 = z_4 + z_7$; $\quad t_6 = z_1 - \frac{1}{2} t_5$; $\quad t_7 = z_4 - z_7$; $\quad t_8 = z_1 + t_5$;

$t_9 = z_2 + z_5$; $\quad t_{10} = z_8 - \frac{1}{2} t_9$; $\quad t_{11} = z_2 - z_5$; $\quad t_{12} = z_8 + t_9$;

$t_{13} = t_8 + t_{12}$; $\quad t_{14} = t_6 + t_{10}$; $\quad t_{15} = t_6 - t_{10}$; $\quad t_{16} = t_7 + t_{11}$;

$t_{17} = t_7 - t_{11}$; $\quad t_{18} = c_2 t_{14} - c_7 t_{17}$; $\quad t_{19} = c_4 t_{14} + c_9 t_{17}$;

$t_{20} = c_3 t_{15} + c_6 t_{16}$; $\quad t_{21} = c_5 t_{15} - c_8 t_{16}$; $\quad t_{22} = t_{18} + t_{19}$;

$t_{23} = t_{20} - t_{21}$;

$y_0 = t_4 + t_{13}$; $\quad y_1 = t_2 + t_{18}$; $\quad y_2 = t_2 + t_{19}$; $\quad y_3 = t_4 - \frac{1}{2} t_{13}$;

$y_4 = t_2 - t_{22}$; $\quad y_5 = t_5 - t_{23}$; $\quad y_6 = c_1(t_8 - t_{12})$;

$y_7 = t_{21} - t_3$; $\quad y_8 = t_3 + t_{20}$;

$x_0 = y_0$; $\quad x_1 = y_1 + iy_8$; $\quad x_2 = y_2 + iy_7$; $\quad x_3 = y_3 + iy_6$; $\quad x_4 = y_4 + iy_5$;

$x_5 = y_4 - iy_5$; $\quad x_6 = y_3 - iy_6$; $\quad x_7 = y_2 - iy_7$; $\quad x_8 = y_1 - iy_8$.

(h) $n=16$ $(r=1,3,5,7,9,11,13,15)$: $\theta=\pi/8$: $c_1=\sin(4r\theta)=\pm 1$

$c_2=\cos(r\theta)$; $c_3=\sin(r\theta)$; $c_4=\cos(2r\theta)$;

$c_5=c_1c_4$; $c_6=c_1c_3$; $c_7=c_1c_2$;

$t_1=z_0+z_8$; $t_2=z_4+z_{12}$; $t_3=z_0-z_8$; $t_4=c_1(z_4-z_{12})$;

$t_5=t_1+t_2$; $t_6=t_1-t_2$; $t_7=z_1+z_9$; $t_8=z_5+z_{13}$;

$t_9=z_1-z_9$; $t_{10}=z_5-z_{13}$; $t_{11}=t_7+t_8$; $t_{12}=t_7-t_8$;

$t_{13}=z_2+z_{10}$; $t_{14}=z_6+z_{14}$; $t_{15}=z_2-z_{10}$; $t_{16}=z_6-z_{14}$;

$t_{17}=t_{13}+t_{14}$; $t_{18}=c_4(t_{15}-t_{16})$; $t_{19}=c_5(t_{15}+t_{16})$;

$t_{20}=c_1(t_{13}-t_{14})$; $t_{21}=z_3+z_{11}$; $t_{22}=z_7+z_{15}$; $t_{23}=z_3-z_{11}$;

$t_{24}=z_7-z_{15}$; $t_{25}=t_{21}+t_{22}$; $t_{26}=t_{21}-t_{22}$; $t_{27}=t_9+t_{24}$;

$t_{28}=t_{10}+t_{23}$; $t_{29}=t_9-t_{24}$; $t_{30}=t_{10}-t_{23}$; $t_{31}=t_5+t_{17}$;

$t_{32}=t_{11}+t_{25}$; $t_{33}=t_3+t_{18}$; $t_{34}=c_2t_{29}-c_6t_{30}$;

$t_{35}=t_3-t_{18}$; $t_{36}=c_7t_{27}-c_3t_{28}$; $t_{37}=t_4+t_{19}$;

$t_{38}=c_3t_{27}+c_7t_{28}$; $t_{39}=t_4-t_{19}$; $t_{40}=c_6t_{29}+c_2t_{30}$;

$t_{41}=c_4(t_{12}-t_{26})$; $t_{42}=c_5(t_{12}+t_{26})$;

$y_0=t_{31}+t_{32}$; $y_1=t_{33}+t_{34}$; $y_2=t_6+t_{41}$; $y_3=t_{35}+t_{40}$;

$y_4=t_5-t_{17}$; $y_5=t_{35}-t_{40}$; $y_6=t_6-t_{41}$; $y_7=t_{33}-t_{34}$;

$y_8=t_{31}-t_{32}$; $y_9=t_{38}-t_{37}$; $y_{10}=t_{42}-t_{20}$; $y_{11}=t_{36}+t_{39}$;

$y_{12}=c_1(t_{11}-t_{25})$; $y_{13}=t_{36}-t_{39}$; $y_{14}=t_{42}+t_{20}$;

$y_{15}=t_{38}+t_{37}$;

$x_0=y_0$; $x_8=y_8$; $x_i=y_i+iy_{16-i}$, $1\leqslant i\leqslant 7$;

$x_{16-i}=y_i-iy_{16-i}$, $1\leqslant i\leqslant 7$.

Acknowledgments

The author thanks Evhen Yakimiw for reviewing a first draft of this paper, and Maryse Ferland for typing the manuscript. He is also grateful to Dr. Peter Mayes of NAG, for pointing out a couple of mistakes in the DFT modules as given in a draft version of the paper.

References

1. C. S. Burrus and P. W. Eschenbacher, *IEEE Trans. Acoust. Speech Signal Process.* **29**, 806 (1981).
2. J. H. Rothweiler, *IEEE Trans. Acoust. Speech Signal Process.* **30**, 105 (1982).
3. Y. Suzuki, T. Sone, and K. Kido, *IEEE Trans. Acoust. Speech Signal Process.* **34**, 380 (1986).
4. C. Temperton, *J. Comput. Phys.* **52**, 1 (1983).
5. C. Temperton, *J. Comput Phys.* **52**, 340 (1983).
6. C. Temperton, *J. Comput. Phys.* **52**, 198 (1983).
7. C. Temperton, *J. Comput. Phys.* **58**, 283 (1985).
8. C. Temperton, Implementation of a prime factor FFT algorithm on Cray-1, *Parallel Comput.*, in press.
9. C. Temperton, A self-sorting in-place prime factor real/half-complex FFT algorithm, *J. Comput. Phys.*, in press.
10. S. Winograd, *Math. Comp.* **32**, 175 (1978).