TECHNICAL MEMORANDUM

68

A note on prime factor FFT algorithms

C. Temperton

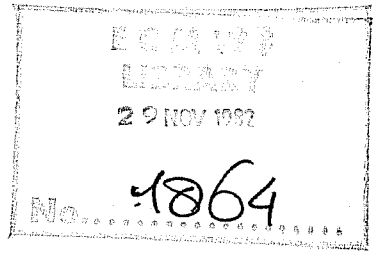
Research Department

November 1982

European Centre for Medium-Range Weather Forecasts
Europäisches Zentrum für mittelfristige Wettervorhersage
Centre européen pour les prévisions météorologiques à moyen

ABSTRACT

The implementation of prime factor FFT algorithms is considered in the case of computers such as the Cray-1 and Cyber 205, where multiplications can be performed in parallel with additions. It is shown that only very modest gains over the conventional FFT can be achieved using Good's algorithm. Winograd's technique is likely to be no faster than the conventional FFT, particularly on the Cray-1 where it becomes memory-bound.



1. INTRODUCTION

The publication of a short paper by Cooley and Tukey [2] in 1965 led to a revolution in the computation of discrete Fourier transforms (DFT's). They showed that if N can be decomposed into suitable small factors, then a complex DFT of length N can be computed in O(N log N) operations, compared with the $O(N^2)$ operations required in a direct implementation. In practice the computation of DFT's was speeded up by one or two orders of magnitude. Many different variants of the algorithm were subsequently developed (see [11] for a unified treatment), but all with essentially the same operation count.

A second revolution - at least from a theoretical point of view - resulted from the work of Winograd [12]. He showed that if the factors of N were mutually prime, then the number of multiplications required could be greatly reduced, typically by a factor of 4, while the number of additions remained roughly the same. Winograd's technique was a development of the algorithm due to Good [4] for mutually prime factors, which in fact pre-dated the 'conventional' FFT of Cooley and Tukey.

Kolba and Parks [5] re-evaluated Good's algorithm, incorporating Winograd's 'small-n' transforms, and pointed out that on computers where the time taken for addition is a significant fraction of that taken for multiplication, their modification of Good's algorithm would be faster than Winograd's technique.

In this paper we extend the analysis of Kolba and Parks to the case of large-scale scientific computers such as the Cray-1 and Cyber 205. The crucial point here is not that they are vector machines, but that floating point additions and multiplications can be performed simultaneously. Consequently, in the context of the FFT algorithm, multiplications are 'free', and the time taken depends essentially only on the number of additions. We show that Good's prime factor algorithm can be further improved in this case by using conventional 'small-n' transforms in place of Winograd's. It also emerges, however, that prime factor algorithms offer only very modest improvements over the conventional forms of the algorithm. On the Cray-1 in particular, Winograd's algorithm is likely to be slower than the conventional approach.

2. THREE FFT ALGORITHMS

In order to demonstrate the principles behind the various FFT algorithms discussed here, it will suffice to consider the case $N = pq$.
Let $\omega = \exp(2i\pi/N)$, and let W_N be the DFT matrix of order N, with element (j,k) given by ω^{jk} where the rows and columns of W_N are indexed from 0 to N-1.

The conventional FFT algorithm is based on the factorization [11]:

$$W_{pq} = (W_q \times I_p)\ P^p_q\ D^p_q\ (W_p \times I_q) \tag{1}$$

where W_p, W_q are the DFT matrices of order p, q ; I_p, I_q are the corresponding identity matrices; P^p_q is a permutation matrix, implemented in practice by the indexing scheme given in [11]; D^p_q is a diagonal matrix of complex "rotation factors", and x denotes the Kronecker product.

Good [4] demonstrated that if p and q are mutually prime, then Eq. (1) can be rewritten as

$$\begin{aligned} W_{pq} &= P\ (I_p \times W_q)(W_p \times I_q)Q \\ &= P\ (W_p \times W_q)Q \end{aligned} \tag{2}$$

where P and Q are permutation matrices. Hence by suitable input and output permutations, the one-dimensional transform is mapped into a two-dimensional transform. In terms of operation counts the important point is that the diagonal matrix of rotation factors has been eliminated. Gold and Rader [3, Chapter 6] give a rather clear account of how this works.

Winograd's algorithm [12] depends on the observation that each of the 'small-n' transform matrices W_p, W_q can be written in the form

$$W_p = B_p M_p A_p \tag{3}$$

and similarly for W_q. Here A_p and B_p are matrices whose non-zero entries are all ± 1, i.e. multiplication by these matrices requires only additions and subtractions. M_p is a diagonal matrix whose entries are all pure real or pure imaginary numbers. Note that the order of M_p may be greater than

that of W_p, so that A_p and B_p are rectangular; but in the algorithms given by Winograd, the order of M_p is at most (p+2). Winograd's paper [12] concentrates on finding factorizations of the form (3) which minimize the order of M_p, and the way in which these are combined for composite N is perhaps more clearly explained in papers by other authors [5,7].

Substituting (3) in Eq. (2), and rearranging the factors using the algebra of Kronecker products, we have

$$W_{pq} = P\ (B_p \times B_q)\ (M_p \times M_q)\ (A_p \times A_q)Q \qquad (4)$$

The important points here are that the 'input' and 'output' stages $(A_p \times A_q)$ and $(B_p \times B_q)$ still consist entirely of additions and subtractions, while the matrix $(M_p \times M_q)$ is still diagonal with pure real or pure imaginary elements, and its order is not much greater than pq. This 'nesting' technique is the key to the significant reduction in the number of multiplications achieved by Winograd's algorithm.

In extending these factorizations to more than two factors, it must be remembered that Eqs. (2) and (4) are only valid if all the factors of N are mutually prime.

3. OPERATION COUNTS

Kolba and Parks [5] considered the implementation of Good's prime factor algorithm, based on the use of Winograd's small-n transforms. In comparison with Winograd's procedure, this requires more multiplications but fewer additions. They pointed out that if the time taken for addition is a significant fraction of that taken for multiplication, then their procedure would be faster.

On the Cray-1 and Cyber 205, this argument applies with greater force. (In the context of this discussion we can ignore the question of vectorization, for example assuming that many independent transforms are to be performed in parallel). On the Cray-1, independent additions can be performed in parallel, or triadic operations such as

$\underset{\sim}{a} = \underset{\sim}{b} * (\underset{\sim}{c} + \underset{\sim}{d})$ and $\underset{\sim}{a} = \underset{\sim}{b} + (\underset{\sim}{c} * \underset{\sim}{d})$ can be performed in 'chained' mode whereby the addition and multiplication are in effect carried out simultaneously. On the Cyber 205, there is a more restrictive 'linked' mode analogous to chaining for triadic operations in which one of the operands is a scalar. All variants of the FFT algorithm require considerably more additions than multiplications, and it can be shown that on these machines only the number of additions is relevant, since all multiplications can be chained or linked with additions (or done in parallel on the Cray-1) and are therefore implemented free of charge [9,10]. (On the Cyber 205 there is a slight overhead since the vector start-up time for a linked add-multiply is longer than that for an addition on its own, but for sufficiently long vectors this can be ignored).

Following Kolba and Parks, it can be seen that Good's prime factor algorithm, based on Winograd's small-n transforms, should be faster than Winograd's nested technique on the Cray-1 and Cyber 205 since it requires fewer additions. However, we can improve on this result still further by noting that Winograd's small-n transforms themselves achieve the minimum number of multiplications at the expense of extra additions for some values of n, when compared with "conventional" small-n transforms. Table I compares the number of real additions and multiplications required in each case.

The conventional small-n transforms for n = 2,3,4 are well known and can easily be derived by inspection. That for n=5 is due to Rader, as quoted by Singleton [8] whose algorithm for arbitrary odd prime factors is used here for n=7. The algorithms for n=8 and n=16 can be built up from radix-2 or radix-4 algorithms, taking advantage of occasions when the rotation factor angles are multiples of $\pi/4$ [1]. The algorithm for n=9 is based simply on Eq. (1) with p=q=3.

Kolba and Parks dismiss multiplications by 1/2 and 1/4 as 'shifts', but in Table I they are counted as full multiplications.

No claim is made here that these conventional algorithms actually achieve any theoretical minimum number of additions; it is simply to be noted that in several cases they require fewer additions than Winograd's corresponding small-n transforms.

We now compare, in Table II, the total number of real additions and multiplications required for a DFT of length N implemented in four different ways. The first column is for conventional transforms based on Eq. (1); in order to correspond as nearly as possible to mixed-radix FFT packages which actually exist [11] only factors $2 \leqslant n \leqslant 7$ (including n=6) have been used. The provision of coding for n=6 permits some advantage to be taken of mutually prime factors, without going to the complexity of the full prime factor algorithms. In some cases the operation counts could be improved by adding n=8,9,16, but these factors are not normally included in conventional routines. The operation counts here are derived using the formulae given in [11]; other authors [5,7,12,13] have also compared the operation counts for conventional transforms versus Winograd's technique, but almost invariably they have over-estimated the number of additions required in the conventional case.

The second column is for Good's algorithm using the conventional small-n transforms of Table I as advocated here, while the third column is for the same algorithm using Winograd's small-n transforms as suggested by Kolba and Parks. The fourth column is for Winograd's nested transform; these operation counts are taken from Zohar [13].

Table II shows the impressive reduction in the number of multiplications required by Winograd's approach in comparison with the conventional algorithm, but the important point here is that this is often achieved at the expense of the number of additions. Kolba and Parks reduce the number of additions required in comparison with Winograd, but the minimum number of additions is achieved in the second column with Good's algorithm based on conventional small-n transforms.

Notice, however, that the reduction in the number of additions achieved by Good's algorithm compared with the conventional approach is very modest, typically only 10%. Also, some conventional operation counts are given in Table II for values of N such that the prime factor algorithms are not practicable. These have been included to demonstrate that even if we have some flexibility in the choice of N, there is not much to be gained from the prime factor algorithms when the cost depends only on the number of additions.

Some authors [5,6] have suggested 'split nesting' techniques in which, for example, the case N=pqrs (with all factors mutually prime) could be decomposed using Good's technique into a two-dimensional (pq) x (rs) transform, and the resulting one-dimensional transforms of length pq and rs could be decomposed using Winograd's technique. This would require fewer additions than the full nested algorithm (at the expense of extra multiplications), but it can be seen that using Good's technique in full requires even fewer additions.

4. MEMORY CONSIDERATIONS ON CRAY-1

While the argument of the preceding section is valid as it stands on the Cyber 205, an important factor has been neglected in the case of the Cray-1. Before any arithmetic can be done on this machine, the operands must first be loaded into vector registers, and the eventual results must be stored back in memory. (Temporary results may be held in vector registers). These transfers to and from memory can proceed in parallel with the arithmetic, but they must be taken into account when assessing the time taken for a given computation. In the context of the present discussion, if there are more memory transfers than additions then the time taken will depend only on the number of memory transfers. For example, if we compute $\underset{\sim}{y} = W_2 \underset{\sim}{x}$ we load 4 real components of $\underset{\sim}{x}$ into the registers, perform 4 real additions and store 4 real results back into memory. During the time taken for the total of 8 memory references, we could have performed 8 additions rather than only 4; the computation is said to be memory-bound. However, as shown in [9] n=2 is the only example of such a small-n transform which is memory-bound; in all other cases the time taken depends only on the number of additions, and all memory references can be overlapped with the arithmetic. The number of memory references (4n) and real additions for each n is summarized in Table III. For the larger values of n, additional memory references will in fact be required for temporary results, since only 8 vector registers are provided; but in these cases the number of additions so far exceeds 4n that these extra memory references can be accommodated without penalty.

In Good's prime factor algorithm for composite N, one pass through the data is made for each factor n, and apart from the factor n=2 the time dependence on the number of additions still holds true. (In the case of the conventional algorithm, the number of memory references remains the same

while the number of additions increases, so the time still depends only on the number of additions.)

In the case of Winograd's algorithm, there are two passes for each factor of N, one during the 'input' stage and another during the 'output' stage. For each n, the partition of additions between input and output stages is given by Silverman [7] and reproduced in Table III. Also shown is the number of memory references, given by $2(n+\mu_n)$ where μ_n is the order of the diagonal matrix in the decomposition given by Eq. (3). In most cases this exceeds the number of additions, and in the remaining cases the margin is probably always too small to accommodate the extra memory references needed for temporary results. Thus Winograd's algorithm implemented on Cray-1 will apparently be memory-bound throughout, and the time taken will be significantly longer than suggested by the number of additions. This strengthens the conclusions of the previous section, and suggests that Winograd's algorithm will be slower than a conventional transform for the same value of N on Cray-1.

5. CONCLUSIONS

It has been shown that prime factor FFT algorithms offer little improvement over conventional FFT algorithms on computers such as the Cray-1 and Cyber 205 where the multiplications can be performed in parallel with the additions. A very modest gain may be obtained by using Good's algorithm with conventional small-n transforms. Winograd's technique, despite its impressive reduction in the number of multiplications, is likely to be slower than the conventional algorithm, particularly on the Cray-1 where memory transfers will dominate the computation.

TABLE I

Number of real operations for small-n transforms

	conventional		Winograd	
n	adds	mults	adds	mults
2	4	0	4	0
3	12	4	12	4
4	16	0	16	0
5	32	12	34	10
7	60	36	72	16
8	52	4	52	4
9	80	40	88	20
16	144	24	148	20

TABLE II

Number of real additions/multiplications for DFT's of length N

N	conventional	Good	Kolba & Parks	Winograd
105	2272/1492	1992/932	2214/590	2418/322
108	2018/1012	-	-	-
112	2162/1188	1968/744	2188/396	2332/308
120	2302/1116	2028/508	2076/460	2076/276
126	2684/1672	2452/1208	2780/568	3068/392
128	2242/900	-	-	-
240	5322/2708	4656/1256	4812/1100	5016/632
252	5954/2500	5408/2416	6064/1136	6640/784
256	5122/2050	-	-	-
315	8492/5728	7516/3776	8462/2050	10406/1186
320	7202/3396	-	-	-

TABLE III

Number of memory references and real additions on Cray-1

n	Prime factor algorithm		Winograd input stage		Winograd output stage	
	memory	adds	memory	adds	memory	adds
2	8	4	8	4	8	0
3	12	12	12	6	12	6
4	16	16	16	12	16	4
5	20	32	22	16	22	18
7	28	60	32	34	32	38
8	32	52	32	28	32	24
9	36	80	40	40	40	48
16	64	144	68	80	68	68

REFERENCES

1. G.D. BERGLAND, An FFT algorithm using base 8 iterations, Math. Comp. 22 (1968), 275-279.

2. J.W. COOLEY AND J.W. TUKEY, An algorithm for the machine computation of complex Fourier series, Math. Comp. 19 (1965), 297-301.

3. B. GOLD AND C.M. RADER, "Digital Processing of Signals", McGraw-Hill, New York, 1969.

4. I.J. GOOD, The interaction algorithm and practical Fourier analysis, J. Royal Statist. Soc. Ser. B 20 (1958), 361-372.

5. D.P. KOLBA AND T.W. PARKS, A prime factor FFT algorithm using high-speed convolution, IEEE Trans. Acoustics, Speech and Signal Processing 25 (1977), 281-294.

6. H.J. NUSSBAUMER AND P. QUANDALLE, Fast computation of discrete Fourier Transforms using polynomial transforms, IEEE Trans. Acoustics, Speech and Signal Processing 27 (1979), 169-181.

7. H.F. SILVERMAN, An introduction to programming the Winograd Fourier transform algorithm (WFTA), IEEE Trans. Acoustics, Speech and Signal Processing 25 (1977), 152-165.

8. R.C. SINGLETON, An algorithm for computing the mixed-radix Fast Fourier transform, IEEE Trans. Audio and Electroacoustics 17 (1969), 93-103.

9. C. TEMPERTON, Fast Fourier Transforms and Poisson-solvers on Cray-1, in "Supercomputers", Infotech State of the Art Report, Infotech International Ltd., Maidenhead, U.K., 1979.

10. C. TEMPERTON, Fast Real Fourier Transforms on the Cyber 205, Met.O.11 Technical Note No.155, U.K. Meteorological Office, 1982.

11. C. TEMPERTON, Self-sorting mixed-radix Fast Fourier Transforms, submitted to J. Computational Phys.

12. S. WINOGRAD, On computing the discrete Fourier transform, Math. Comp. 32 (1978), 175-199.

13. S. ZOHAR, A prescription of Winograd's Discrete Fourier Transform Algorithm, IEEE Trans. Acoustics, Speech and Signal Processing 27 (1979), 409-421.

LIST OF SYMBOLS

$\underset{\sim}{a}$, $\underset{\sim}{b}$, $\underset{\sim}{c}$, $\underset{\sim}{d}$, $\underset{\sim}{x}$, $\underset{\sim}{y}$	vectors (bold type, lower case Latin letters)
P^{p}_{q}, D^{p}_{q}	upper case Latin letters with superscript 'p', subscript 'q'
ω, π,	lower case Greek omega, pi
μ_{n}	lower case Greek mu, subscript 'n'
0	numeral zero throughout
1	numeral one throughout