TECHNICAL MEMORANDUM

71

Fast mixed radix real Fourier transforms

C. Temperton

Research Department

January 1983

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

Abstract

It is shown that the self-sorting variants of the mixed-radix FFT algorithm may be specialized to the case of real or conjugate-symmetric input data. In comparison with conventional procedures, savings of around 20% are achieved in terms of operation counts. A multiple real/half-complex transform package on the Cray-1, based on the algorithms described here, achieves a 30% saving in CPU time compared with a package using conventional algorithms. A similar package has also been implemented on the Cyber 205.

1. INTRODUCTION

In a previous paper [8], the Fast Fourier Transform (FFT) algorithm was derived in terms of a matrix factorization. Particular emphasis was laid on self-sorting variants of the algorithm, which eliminate the need for an explicit data permutation before or after the transform. The algorithms in [8] related to transforms of complex data, though it was mentioned that often the data to be transformed are real or conjugate-symmetric; certainly this is true of all the meteorological applications described. Thus we need to compute:

$$x_j = \sum_{k=o}^{N-1} c_k \exp(2ijk\pi/N), \quad o \leq j \leq N-1 \tag{1}$$

or its inverse,

$$c_k = \frac{1}{N} \sum_{j=o}^{N-1} x_j \exp(-2ijk\pi/N), \quad o \leq k \leq N-1 \tag{2}$$

where the data x_j are real, and the Fourier coefficients c_k satisfy the relationship $c_{N-k} = c_k^*$

Real/half-complex transforms are usually implemented by adding a pre- or post-processing step to a standard complex FFT; two such procedures were given by Cooley, Lewis and Welch [3]. For example, to implement Eq.(2) for x_j real we can form the complex sequence

$$z_j = x_{2j} + ix_{2j+1}, \quad o \leq j \leq N/2-1,$$

perform a complex FFT of length N/2, and recover the coefficients c_k through a post-processing step which requires (2.5N-6) real additions and (N-4) real multiplications. Alternatively, to transform two independent sets of real data x_j and y_j, we can form the complex sequence

$$z_j = x_j + iy_j, \quad o \leq j \leq N-1,$$

perform a complex FFT of length N, and obtain the corresponding Fourier

coefficients through another post-processing step. The operation count per real transform is almost identical to that for the first procedure.

These techniques require either that N is an even number, or that the number of transforms to be performed is even. Usually this is only a minor nuisance, but the point to be made here is that in circumstances where speed is essential, faster alternatives to these procedures exist.

Bergland [1,2], building on an idea credited to Edson, presented a more efficient algorithm. If a real data sequence is used as input to a complex FFT algorithm, then about half the computation is redundant. By pruning out these redundant operations, a faster procedure than those presented in [3] is obtained. Bergland's algorithms were specializations of the Cooley-Tukey [4] and Gentleman-Sande [5] variants of the FFT, and thus required data permutations before or after the transforms. Also, he only gave details for the case $N=2^p$, though the possibility of extending the procedure to the more general case was mentioned.

In this paper we show how to specialize the self-sorting mixed-radix algorithms of [8] to the real/half-complex case. The reduction in the operation count compared with the procedures of [3] is shown to be typically 20%, and savings of around 30% are obtained in the execution time for a real/half-complex FFT package on the Cray-1, as used for the meteorological applications at ECMWF described in [8].

2. PRINCIPLES OF THE ALGORITHM

In [8], Eq.(1) was rewritten as $\underset{\sim}{x} = W_N \underset{\sim}{c}$; the discrete Fourier transform (DFT) matrix W_N is defined by $[W_N]\ (j,k) = \omega^{jk}$ where $\omega = \exp(2i\pi/N)$ and the rows and columns of W_N are indexed from 0 to N-1. The FFT algorithms were derived from the matrix factorization:

$$W_{pq} = (W_q \times I_p) P_q^p D_q^p (W_p \times I_q) \tag{3}$$

where W_p, W_q are the DFT matrices of order p,q; I_p, I_q are the corresponding identity matrices; and P_q^p, D_q^p are permutation and diagonal matrices defined in [8].

In this paper we shall also need the corresponding result for Eq.(2), which can be written (dropping the scaling factor) as $\underset{\sim}{c} = \bar{W}_N \underset{\sim}{x}$ where $\bar{W}_N$ is the complex conjugate of W_N. From (3) we have immediately

$$\bar{W}_{pq} = (\bar{W}_q \times I_p) P_q^p \bar{D}_q^p (\bar{W}_p \times I_q) \tag{4}$$

(Here $\bar{D}_q^p$ is the complex conjugate of D_q^p).

As in [8], Eq.(4) can be extended to the multiple-factor case in several different ways. For the self-sorting decimation-in-time variant, let $N = n_1 n_2 \cdots n_k$, $\ell_1 = 1$, $\ell_{i+1} = n_i \ell_i$ for $1 \leq i \leq k$, and $m_i = N/\ell_{i+1}$. Then the general form is given by

$$\bar{W}_N = T_k T_{k-1} \cdots T_2 T_1 \tag{5}$$

with

$$T_i = (\bar{W}_{n_i} \times I_{N/n_i}) (P_{n_i}^{\ell_i} \bar{D}_{n_i}^{\ell_i} \times I_{m_i}) \tag{6}$$

In connection with self-sorting decimation in time, it was noted in [8] that for example in the case N = pqr, the first stage consists of DFT's of length p on qr interleaved samples of the data; in the second stage these are combined into DFT's of length pq on r samples of the data, and in the final stage these are combined into a single DFT of length pqr. The same is true of the algorithm given by (5) and (6). To be specific, let

$$\underset{\sim}{z}^{(o)} = \underset{\sim}{z},\ \underset{\sim}{z}^{(i)} = T_i\ \underset{\sim}{z}^{(i-1)} \quad \text{for } 1 \leq i \leq k.$$

Then

$$\underset{\sim}{z}^{(i)} = (\bar{W}_{\ell_{i+1}} \times I_{m_i})\ \underset{\sim}{z}\ . \tag{7}$$

Proof: Eq.(7) is true for i=0, since $\ell_i=1$, $m_o=N$. Suppose Eq.(7) holds for i-1. Then

$$\underset{\sim}{z}^{(i)} = T_i\ \underset{\sim}{z}^{(i-1)} = T_i\ (\bar{W}_{\ell_i} \times I_{m_{i-1}})\underset{\sim}{z}$$

by inductive hypothesis.

Using the definition of T_i, this can be written as

$$(\bar{W}_{n_i} \times I_{N/n_i})(P^{\ell_i}_{n_i}\ \bar{D}^{\ell_i}_{n_i} \times I_{m_i})(\bar{W}_{\ell_i} \times I_{m_{i-1}})\underset{\sim}{z}.$$

Since $\ell_i\ m_i = N/n_i$ and $m_{i-1} = m_i n_i$ this becomes

$$(\bar{W}_{n_i} \times I_{\ell_i m_i})(P^{\ell_i}_{n_i}\ \bar{D}^{\ell_i}_{n_i} \times I_{m_i})(\bar{W}_{\ell_i} \times I_{m_i n_i})\underset{\sim}{z}$$

which can be rearranged to give

$$(\{(\bar{W}_{n_i} \times I_{\ell_i})(P^{\ell_i}_{n_i}\ \bar{D}^{\ell_i}_{n_i})(\bar{W}_{\ell_i} \times I_{n_i})\} \times I_{m_i})\underset{\sim}{z}$$

which is just

$$(\bar{W}_{\ell_{i+1}} \times I_{m_i})\underset{\sim}{z}$$

as required, using Eq.(4) with $p=\ell_i$, $q=n_i$ and $\ell_{i+i}=n_i\ell_i$.

Since $\ell_{k+1} = N$ and $m_k = 1$, Eq.(7) implies that $\underset{\sim}{z}^{(k)} = \bar{W}_N\underset{\sim}{z}$, so the above argument constitutes a formal inductive proof that the whole algorithm works.

The significance of Eq.(7) in the present context is as follows. Suppose $\underset{\sim}{z}^{(o)}$ is a vector of <u>real</u> numbers. Then $\underset{\sim}{z}^{(i)}$ consists of m_i interleaved inverse DFT's of length ℓ_{i+1}; each of these is a transform of real data, and is therefore conjugate-symmetric. It follows that $\underset{\sim}{z}^{(i)}$ contains only N independent real numbers for $o \leq i \leq k$. The specialization of the complex FFT algorithm to the real/half-complex case depends on the fact that only N real numbers need be specified at each stage; the "missing" numbers are either zero imaginary parts or the conjugates of complex numbers already specified.

3. ROUTINES FOR THE REAL/HALF-COMPLEX CASE

The specialization of the algorithms given in [8] to the real/half-complex case is most easily described in terms of Fortran routines. We first give a routine for self-sorting decimation in time applied to the inverse transform of complex data.

Suppose that the factors of N have been stored in an array IFAX(1) to IFAX(NFAX), and that a complex array of trigonometric function values has been defined by

$$\text{TRIGS}(K+1) = \exp(2iK\pi/N) \ , \ o \leq K \leq N-1$$

The data to be transformed is in an array A, and a work array C is provided. Each array acts alternately as input and output for successive stages of the algorithm. The FFT routine is then given by:

```
C    DECIMATION IN TIME
     COMPLEX A(N), C(N), TRIGS(N)
     INTEGER IFAX(NFAX)
     LA=N
     DO 1Ø I = 1,NFAX
     IFAC=IFAX(I)
     LA=LA/IFAC
     CALL PASS(A,C,TRIGS,IFAC,LA,N)
C    [now reverse rôles of A and C]
 1Ø  CONTINUE
     STOP
     END
```

The subroutine PASS takes the following form:

```
C    DECIMATION IN TIME FOR INVERSE TRANSFORM
     SUBROUTINE PASS(A,C,TRIGS,IFAC,LA,N)
     COMPLEX A(N), C(N), TRIGS(N)
     M=N/IFAC
C    Define IFAC base addresses in A
     IA=Ø, IB=LA, IC=2*LA, ID=3*LA,...
C    Define IFAC base addresses in  C
     JA=Ø, JB=M, JC=2*M, JD=3*M,...
C    ..............................
     I=1
     J=1
     JUMP=(IFAC-1)*LA
     DO 2Ø K=Ø,M-LA,LA
     DO 1Ø L=1,LA
     C~(J)=W̄(IFAC)*(Ω̄(K)*A~(I))
     I=I+1
     J=J+1
 1Ø  CONTINUE
     I=I+JUMP
```

```
   2Ø CONTINUE
      RETURN
      END
```

In the inner loop, $\bar{W}$(IFAC) is the inverse DFT matrix of order IFAC, $\bar{\Omega}$(K) is the complex conjugate of the diagonal matrix Ω(K) given by

$$\Omega(K)=\text{diag}(\text{TRIGS}(1),\ \text{TRIGS}(K+1),\ \text{TRIGS}(2*K+1),\ldots),$$

$\underset{\sim}{A}(I)$ and $\underset{\sim}{C}(J)$ are vectors of length IFAC defined by

$$\underset{\sim}{A}(I) = \begin{bmatrix} A(IA+I) \\ A(IB+I) \\ A(IC+I) \\ \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix}, \underset{\sim}{C}(J) = \begin{bmatrix} C(JA+J) \\ C(JB+J) \\ C(JC+J) \\ \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix}$$

Suppose now that we apply the routine given above to data which is initially real (Fourier analysis). As shown in the previous section, the vector $\underset{\sim}{z}^{(i)}$ of Eq.(7) consists of m_i interleaved complex conjugate sequences, each of length ℓ_{i+1}. The first m_{i-1} elements of $\underset{\sim}{z}^{(i-1)}$ are therefore real. If ℓ_i is odd, these are followed by $m_{i-1}\ (\ell_i-1)/2$ complex elements, and the remaining $m_{i-1}(\ell_i-1)/2$ elements are their complex conjugates. If ℓ_i is even, these complex conjugate sequences are separated by a further m_{i-1} real elements.

The ith call of subroutine PASS computes $\underset{\sim}{z}^{(i)}$ from $\underset{\sim}{z}^{(i-1)}$, and IFAC*LA corresponds to m_{i-1}. During the first pass through the outer loop (K=0), the elements of $\underset{\sim}{A}(I)$ are taken from the first IFAC*LA $=m_{i-1}$ elements of $\underset{\sim}{z}^{(i-1)}$. The vectors $\underset{\sim}{A}(I)$ are therefore real; $\bar{\Omega}(0)$ is the identity matrix, and the results are conjugate-symmetric, e.g. for IFAC=4 $\underset{\sim}{C}(J)=(c_0,c_1,c_2,c_1^*)$ with c_0,c_2 real; for IFAC=5 $\underset{\sim}{C}(J)=(c_0,c_1,c_2,c_2^*,c_1^*)$ with c_0 real. Special coding is required for this case.

Each subsequent pass through the outer loop accesses the next m_{i-1} entries of $\underset{\sim}{z}^{(i-1)}$. For $0 < K < M/2$, we have full complex transforms as in the case of complex input data.

For K=M/2 (which will be invoked only if ℓ_i is even), the input vectors are again real, but this time $\bar{\Omega}(K)$ is not the identity matrix; in fact $\bar{\Omega}(M/2) = \text{diag}(1, \alpha, \alpha^2, \ldots)$ where $\alpha=\exp(-i\pi/IFAC)$. The results have a "shifted" conjugate symmetry, e.g. for IFAC=4 $\underset{\sim}{C}(J)=(c_0, c_1, c_1^*, c_0^*)$; for IFAC=5 $\underset{\sim}{C}(J)=(c_0, c_1, c_2, c_1^*, c_0^*)$ with c_2 real. Further special coding is required for this case.

For K > M/2 we again have full complex transforms, but the results are just the complex conjugates of the results already obtained for K'=M-K.

In specializing the routine to the case of real input data we adopt the following strategy, which can be implemented by suitable modifications to the indexing. First, zero imaginary parts will not be stored. Second, only one member of each complex conjugate pair is stored; if C(J)=C(I)* and J > I but C(J) is computed first, then that result is conjugated and stored at the location for C(I). This enables us to terminate the outer loop of the subroutine PASS at K=M/2. As a consequence, only the first half of the TRIGS array is ever used.

At each stage of the algorithm, some of the results will be real while others are complex, and the storage pattern remains to be specified. An orderly and convenient arrangement is to separate the real and imaginary parts of a complex number in the vector $\underset{\sim}{z}^{(i)}$ of Eq.(7) by m_i locations.

The corresponding algorithm to compute $\underset{\sim}{x} = W_N \underset{\sim}{c}$ with $\underset{\sim}{c}$ conjugate-symmetric and $\underset{\sim}{x}$ real (Fourier synthesis), can be obtained by "inverting" each operation of the algorithm given above. This is exactly equivalent to an analogous specialization of the decimation-in-frequency form of the complex FFT algorithm applied to Eq.(1).

In both cases the vectors $\underset{\sim}{x}$ and $\underset{\sim}{c}$ are naturally ordered, in contrast to the algorithms of Bergland [1,2]; all necessary permutations are accomplished internally by the indexing scheme of the subroutine PASS.

4. SMALL-n TRANSFORMS AND OPERATION COUNTS

In [8], algorithms were given for the "small-n" transforms $\underset{\sim}{x}=W_n\underset{\sim}{z}$ for $2\leq n\leq 6$. For example, $\underset{\sim}{x}=W_4\underset{\sim}{z}$ is given by

$$t_1=z_0+z_2;\quad t_2=z_1+z_3;\quad t_3=z_0-z_2;\quad t_4=z_1-z_3;$$
$$x_0=t_1+t_2;\quad x_1=t_3+it_4;\quad x_2=t_1-t_2;\quad x_3=t_3-it_4. \qquad (8)$$

For x, z complex this requires 16 real additions.

In the Fourier analysis routine described above, corresponding algorithms are required for transforms of the form $\underset{\sim}{x} = \bar{W}_n\ \underset{\sim}{z}$. These are easily derived by noting that $\bar{W}_n\ \underset{\sim}{z} = W_n\underset{\sim}{z}'$ where $\underset{\sim}{z}'$ is derived from $\underset{\sim}{z}$ by reversing the order of the components z_1 to z_{n-1}. For example, $\underset{\sim}{x} = \bar{W}_4\ \underset{\sim}{z}$ is given by:

$$t_1=z_0+z_2;\quad t_2=z_1+z_3;\quad t_3=z_0-z_2;\quad t_4=z_3-z_1;$$
$$x_0=t_1+t_2;\quad x_1=t_3+it_4;\quad x_2=t_1-t_2;\quad x_3=t_3-it_4 \qquad (9)$$

The operation count is the same; in fact only one statement in (8) has to be changed to obtain (9).

Consider the case K=0 in the Fourier analysis routine; here we have to apply (9) to real input data. The temporary results t_1,t_2,t_3,t_4 are all real; $x_3=x_1^*$ and will not be stored, while x_1 itself requires only an "apparent" addition. In this case the algorithm $\underset{\sim}{x} = \bar{W}_4\underset{\sim}{z}$ requires only 6 real additions. Corresponding operation counts for K=0 and various values of n are given in Table I.

For the case K=0 in the corresponding Fourier synthesis routine, we have to apply (8) to complex conjugate input data, i.e. z_0 and z_2 are real, while $z_3=z_1^*$. If we write

$z_0=y_0$; $z_1=y_1+iy_3$; $z_2 = y_2$; $z_3=y_1-iy_3$

then the algorithm (8), expressed in terms of real numbers, becomes:

$t_1=y_0+y_2$; $t_2=2y_1$; $t_3=y_0-y_2$; $t_4=2y_3$;

$x_0=t_1+t_2$; $x_1=t_3-t_4$; $x_2=t_1-t_2$; $x_3=t_3+t_4$.

Besides 6 real additions, it appears that two doublings are required. Such doublings can be omitted throughout the whole algorithm for $\underset{\sim}{x} = W_N\underset{\sim}{c}$ if the complex elements of $\underset{\sim}{c}$ are doubled before entry. Alternatively the real elements (c_o and $c_{N/2}$ if N is even, c_o only if N is odd) can be halved before entry; the algorithm with the doublings omitted then computes $\underset{\sim}{x}$ = 1/2 $W_N\underset{\sim}{c}$. With this trick included, the operation counts for K=0 during Fourier synthesis become the same as those during Fourier analysis.

Specializing the other small-n transforms given in [8] to the case K=0 for real Fourier analysis or synthesis is equally straightforward, and is left to the interested reader.

As mentioned in the previous section, special transform algorithms are also required for the case K=M/2, to compute $\underset{\sim}{x}=\Omega W_n \underset{\sim}{z}$ or $\underset{\sim}{z} = \bar{W}_n\bar{\Omega}\underset{\sim}{x}$ where $\underset{\sim}{x}$ is real, z is "shifted conjugate-symmetric", and Ω = diag(1, α, α^2,...) with $\alpha=\exp(i\pi/n)$. These algorithms are given in the Appendix, and operation counts for them are included in Table I.

We now consider total operation counts for a real/half-complex transform $\underset{\sim}{x}=W_N \underset{\sim}{z}$ where N is composite. For the complex case, formulae for the operation counts were derived in [8], and it was noted that the counts are independent of the order in which the factors are used. In the real/half-complex case, there is a slight dependence on the order of the factors. The operation counts can be derived by resorting to counting the number of trips through the loops of subroutine PASS (for K=0, 0 < K < M/2 and K=M/2, for each factor in turn) and using the results of Table I.

In Table II we present the operation counts for a selection of values of N, both for a conventional real transform (using the algorithms of [3] together with a complex transform of length N/2) and for the special real transform described here. In the conventional case, allowed factors of N/2 are $2 \leq n \leq 6$ as in [8]. In the special case, a single factor n=8 is also allowed, and the counts are for Fourier synthesis by decimation in frequency, using the factors in ascending order, and ignoring any multiplications used for re-scaling (Fourier analysis using the factors in descending order would give the same operation counts). The single factor n=8 is allowed since with the factors used in this way, only the simple K=0 loop is invoked for n=8. N=256 is factorized as 4^4 rather than $2.4^2.8$ since the latter gives a slightly higher operation count.

Table II shows that use of the special real transform gives a reduction of about 20% in the operation counts, the saving being slightly greater for additions than for multiplications.

Bergland [1] quotes slightly lower operation counts for a radix-2 algorithm than would be obtained using the counting procedure given here; this is because he treats K=M/4 as a special case (the rotation angle in Ω is then $\pi/4$). Also, he exploits the fact that no operations are required in the case K=M/2 to reduce the number of passes for $N=2^P$ from p to (p-1). This trick only works for the radix-2 algorithm.

5. IMPLEMENTATION ON CRAY-1 AND CYBER 205

The first real/half-complex FFT package developed for use on the Cray-1 at ECMWF was based on the procedures of [3] together with a complex FFT algorithm as described in [8]. This was later superseded by a package using the special real/half-complex algorithm described in this paper. As in [8], vectorization was achieved simply by performing multiple transforms in parallel. Both packages were written in Cray Assembly Language (CAL).

Table III shows the times per transform and megaflop rates for various values of N, in the case of 64 transforms being performed simultaneously. The results include the time and operations required for rescaling. The new package achieves a 30% saving in CPU time over the old; as discussed in Section 4 there is a 20% reduction in the operation count, while the remaining saving is a result of programming improvements. As a further measure of efficiency, in the new package the floating-point addition unit is busy for up to 88% of the time.

A similar package has now been implemented on the Cyber 205 at the UK Meteorological Office; some details are given in [7]. As discussed in [8], the straightforward multiple approach to vectorization adopted on the Cray-1 is inadequate on the Cyber 205, which requires much longer vectors to reach near-maximum efficiency. The solution outlined in [8] for the complex case, based on interleaving the transforms and using Eq.(7) for decimation in frequency or an analogous result [6] for decimation in time, carries over directly to the real/half-complex case. Running on the two-pipe Cyber 205 in 32-bit mode, the real FFT package reaches speeds of almost 300 megaflops if many transforms can be computed simultaneously [7].

The prime factor algorithms described in [9] can also be specialized to real/half-complex transforms, but as in the complex case the potential gain appears to be very modest on machines such as the Cray-1 and Cyber 205 where multiplications can be performed in parallel with additions.

APPENDIX: SMALL-n TRANSFORMS FOR K=M/2

Here we set out the algorithms to compute $\underset{\sim}{x} = \Omega W_n \underset{\sim}{z}$ (Fourier synthesis) or $\underset{\sim}{z} = \bar{W}_n \bar{\Omega} \underset{\sim}{x}$ (Fourier analysis) where $\underset{\sim}{x}$ is real, $\underset{\sim}{z}$ is "shifted conjugate symmetric", and $\Omega = \text{diag}(1, \alpha, \alpha^2, \ldots)$ with $\alpha = \exp(i\pi/n)$. The algorithms are expressed in terms of real arithmetic, and $\underset{\sim}{z}$ is written in terms of real vectors as $\underset{\sim}{z} = \underset{\sim}{a} + i\underset{\sim}{b}$.

(a) Fourier synthesis

n=2: $x_o = a_o$; $x_1 = -b_o$

n=3: $t_1 = \frac{1}{2} a_o - a_1$;

$x_o = a_o + a_1$; $x_1 = t_1 - \sin 60 * b_o$; $x_2 = -t_1 - \sin 60 * b_o$

n=4: $t_1 = \sin 45 * (b_o + b_1)$; $t_2 = \sin 45 * (a_o - a_1)$;

$x_o = a_o + a_1$; $x_1 = t_2 - t_1$; $x_2 = b_1 - b_o$; $x_3 = -(t_2 + t_1)$

n=5: $t_1 = a_o + a_1$; $t_1 = \frac{1}{4} t_1 - a_2$; $t_3 = (\sqrt{5}/4) * (a_o - a_1)$;

$t_4 = \sin 36 * b_o + \sin 72 * b_1$; $t_5 = \sin 72 * b_o - \sin 36 * b_1$;

$t_6 = t_3 + t_2$; $t_7 = t_3 - t_2$;

$x_o = t_1 + a_2$; $x_1 = t_6 - t_4$; $x_2 = t_7 - t_5$; $x_3 = -t_7 - t_5$; $x_4 = -t_6 - t_4$

n=6: $t_1 = a_o + a_2$; $t_2 = b_o + b_2$; $t_3 = \sin 60 * (a_o - a_2)$;

$t_4 = \sin 60 * (b_o - b_2)$; $t_5 = \frac{1}{2} t_1 - a_1$; $t_6 = \frac{1}{2} t_2 + b_1$;

$x_o = a_1 + t_1$; $x_1 = t_3 - t_6$; $x_2 = t_5 - t_4$;

$x_3 = b_1 - t_2$; $x_4 = -(t_4 + t_5)$; $x_5 = -(t_3 + t_6)$

(b) Fourier analysis

n=2: $a_o = x_o$; $b_o = -x_1$

n=3: $t_1 = x_1 - x_2$;

$a = x_o + 1/2\ t_1$; $a_1 = x_o - t_1$; $b_o = -\sin 60^\circ (x_1 + x_2)$

n=4: $t_1 = \sin 45^\circ (x_1 - x_3)$; $t_2 = \sin 45^\circ (x_1 + x_3)$;

$a_o = x_o + t_1$; $a_1 = x_o - t_1$; $b_o = -x_2 - t_2$; $b_1 = x_2 - t_2$

n=5: $t_1 = x_1 - x_4$; $t_2 = x_1 + x_4$; $t_3 = x_2 - x_3$; $t_4 = x_2 + x_3$;

$t_5 = t_1 - t_3$; $t_6 = x_o + 1/4\ t_5$; $t_7 = (\sqrt{5}/4)(t_1 + t_3)$;

$a_o = t_6 + t_7$; $a_1 = t_6 - t_7$; $a_2 = x_o - t_5$;

$b_o = -\sin 36^\circ t_2 - \sin 72^\circ t_4$; $b_1 = -\sin 72^\circ t_2 + \sin 36^\circ t_4$

n=6: $t_1 = \sin 60^\circ (x_5 - x_1)$; $t_2 = \sin 60^\circ (x_2 + x_4)$;

$t_3 = x_2 - x_4$; $t_4 = x_1 + x_5$; $t_5 = x_o + \frac{1}{2}\ t_3$; $t_6 = -x_3 - \frac{1}{2}\ t_4$;

$a_o = t_5 - t_1$; $a_1 = x_o - t_3$; $a_2 = t_5 + t_1$;

$b_o = t_6 - t_2$; $b_1 = x_3 - t_4$; $b_2 = t_6 + t_2$

TABLE I

Real operation counts (adds/mults) for $\underset{\sim}{x} = \Omega(K)W_n \underset{\sim}{z}$ or $\underset{\sim}{x} = \overline{W}_n\overline{\Omega}(K)\underset{\sim}{z}$

n	K=0	0<K<M/2	K=M/2
2	2/0	6/4	0/0
3	4/2	16/12	4/2
4	6/0	22/12	6/2
5	12/6	40/28	12/6
6	14/4	46/28	12/4
8	20/2	66/32	22/10

TABLE II

Real operation counts for a real transform of length N

N	conventional		special real	
	adds	mults	adds	mults
180	2156	1104	1714	928
192	2076	832	1654	694
200	2446	1220	2004	1074
216	2552	1224	2020	1012
240	2896	1352	2364	1176
256	2876	1152	2286	942

TABLE III

Times per transform and speeds for multiple real transforms on Cray-1

N	conventional		special real	
	time (μs)	megaflops	time (μs)	megaflops
180	38.9	90	25.8	105
192	38.4	81	25.3	100
200	40.8	94	30.8	107
216	44.5	91	31.3	99
240	51.5	89	36.3	104
256	51.7	78	37.6	95

REFERENCES

1. G.D.BERGLAND, A Fast Fourier Transform Algorithm for real-valued series, C.ACM 11 (1968), 703-710.

2. G.D.BERGLAND, A radix-eight Fast Fourier Transform subroutine for real-valued series, IEEE Trans.Audio and Electroacoustics 17 (1969), 138-143.

3. J.W.COOLEY, P.A.W.LEWIS AND P.D.WELCH, The Fast Fourier Transform algorithm: programming considerations in the calculation of sine, cosine and Laplace transforms, J.Sound Vib. 12 (1970), 315-337.

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

5. W.M. GENTLEMAN AND G. SANDE,Fast Fourier Transforms - for fun and profit, Proc.AFIPS Joint Computer Conference 29 (1966), 563-578.

6. D.G.KORN AND J.J.LAMBIOTTE, Computing the Fast Fourier Transform on a vector computer, Math.Comp. 33 (1979), 977-992.

7. C.TEMPERTON, Fast real Fourier transforms on the Cyber 205, Met.O.11 Technical Note No.155, Meteorological Office, U.K., 1982.

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

9. C.TEMPERTON, A note on prime factor FFT algorithms, submitted to J.Computational Phys.