EUROPEAN CENTRE FOR MEDIUM RANGE WEATHER FORECASTS

TECHNICAL REPORT NO. 3
****

FEBRUARY

1977
****
***
*

MIXED-RADIX FAST FOURIER TRANSFORMS

WITHOUT REORDERING

BY

C. TEMPERTON

EUROPEAN CENTRE FOR MEDIUM RANGE WEATHER FORECASTS

Fitzwilliam House

Skimped Hill

BRACKNELL, BERKS.

UNITED KINGDOM

C O N T E N T S

A B S T R A C T

The Fast Fourier Transform (FFT) is an important tool of computational physics which has found widespread use in meteorological applications. Understanding of the mathematical principles on which the FFT is based is unfortunately less widespread.

This paper presents a relatively simple proof, in terms of matrix algebra, of how the FFT works. The proof is constructive in that it generates not only the usual forms of the FFT, but also some apparently new forms. Two of these have the considerable advantage over more conventional forms that no reordering of the data is required before or after the transform.

Programming strategies for the various forms of FFT are described, and a Fortran routine implementing one of the new forms is included as an Appendix. Timing comparisons indicate that this routine is twice as fast as an otherwise fairly similar program using a conventional FFT algorithm.

1. Introduction

Since its introduction by COOLEY and TUKEY (1965), the Fast Fourier Transform (FFT) has become an indispensable tool of computational physics. Among the more important meteorological applications are the following :

(1) In hemispheric or global gridpoint models based on a latitude-longitude grid, the convergence of meridians towards the poles leads to a severe restriction on the length of the timestep in order to maintain computational stability. This difficulty can be removed by filtering out the higher zonal wavenumbers near the poles at each timestep (HOLLOWAY, SPELMAN and MANABE,1973); the procedure involves large numbers of FOURIER transforms around lines of latitude.

(2) The FFT forms the basis of some of the very fast direct methods for solving the discretized Poisson equation ("Poisson-solvers"). Although these were originally formulated for rectangular grids, they can readily be adapted for use in spherical coordinate systems (SWARZTRAUBER,1974),where for instance they can be used to solve the balance equation (PAEGLE and TOMLINSON,1975). More importantly, they can be used to solve the HELMHOLTZ equations which arise at each timestep in a semi-implicit gridpoint model.

(3) Perhaps the most important application is in spectral prediction models. At each timestep, the nonlinear interactions are computed by transforming certain fields to gridpoint space, carrying out the required multiplications, and then transforming back to wavenumber space (e.g. BOURKE,1972). To obtain a gridpoint field from a set of spherical harmonic coefficients (or vice versa) requires LEGENDRE transforms in the meridional direction, and FOURIER transforms around lines of latitude. In describing their multi-level spectral model, which includes a fair degree of physical parameterization in addition to the dynamics, DALEY et. al.(1976) state that about one third of the total computation time was spent on performing FOURIER transforms. Pseudo-spectral models (MERILEES,1973) also depend heavily on FOURIER transforms.

The ability to transform efficiently between gridpoint and wavenumber space is also useful in problems of horizontal interpolation of data, and in diagnostic studies both of the real atmosphere and of numerical models.

For problems of the size typically encountered in all these applications, the advent of the FFT has brought about an order of magnitude reduction in the computation time required for the FOURIER transforms. It is clearly

important that meteorologists understand how the FFT works, and how it can best be implemented for their particular problems.

A number of different formulations of the FFT have been published; besides the original COOLEY-TUKEY form, we mention here in particular the versions of GENTLEMAN and SANDE (1966), PEASE (1968) and UHRICH (1969).

The FFT is a fast method for computing sums of the form

$$x_j = \sum_{k=0}^{N-1} c_k \exp(2ijk\pi/N), \quad 0 \le j \le N-1 \quad \ldots\ldots(1)$$

where x_j $(0 \le j \le N-1)$ and c_k $(0 \le k \le N-1)$ are complex numbers. Direct implementation of Eq. (1) would clearly require N^2 complex multiplications. If, however, N can be factorized as a product $N = n_1 n_2 \ldots\ldots n_p$, then the FFT reduces the number of complex multiplications from $N.(n_1 n_2 \ldots\ldots n_p)$ to $N.(n_1 + n_2 + \ldots + n_p)$. (In fact we can do considerably better, since it becomes simpler to avoid multiplications by 1). Unfortunately, proofs of how the FFT works (COOLEY and TUKEY 1965, GENTLEMAN and SANDE 1966) tend to involve a plethora of indices, summation signs, "hatted" and "unhatted" variables, and the author suspects he has not been alone in finding them difficult to understand in detail.

Moreover, these proofs tend to pass rather lightly over the fact that the usual forms of the FFT produce their answers in the wrong order, and a permutation is required to unscramble them. If N is factorized as a power of 2, then the required permutation is called "binary reversal" and is easy enough to understand, though rather awkward to program. In the mixed-radix case (N a product of arbitrary factors), the permutation is more difficult both to understand and to program.

A more promising approach (at least for those at home with matrix algebra) is to write Eq. (1) as

$$\underline{x} = W_N \underline{c} \qquad \ldots\ldots\ldots\ldots\ldots(2)$$

where $\underline{x}$ and $\underline{c}$ are vectors with N (complex) elements, and W_N is an N x N matrix, also with complex elements. Note that the elements of $\underline{x}$ and $\underline{c}$ are indexed from 0 to N-1, and the elements of W_N as w_{ij} $(0 \le i \le N-1, \; 0 \le j \le N-1)$; this convention will be used throughout. The matrix W_N is easily defined in terms of its elements: $w_{ij} = \omega^{ij}$, where $\omega = \exp(2i\pi/N)$. The FFT algorithm then becomes equivalent to a factorization of the matrix W_N. This approach was used by PEASE (1968) and THEILHEIMER (1969).

Recently, while writing a mixed-radix FFT program (TEMPERTON, 1976), the author stumbled upon a form which yielded the results in the correct order, thus eliminating

the complicated final permutation step. UHRICH (1969) had previously presented a compact program with this property for $N=2^P$, though with no explanation of how it worked. An effort to explain the workings of the new mixed-radix FFT program yielded a rather simple proof, in matrix form, which generated not only the new form of the FFT, but also a number of other forms, including the COOLEY-TUKEY and GENTLEMAN-SANDE versions, and the formulations of PEASE and UHRICH referred to above. The proof, and the various forms of FFT which it generates, form the subject of this paper.

2. Preliminaries

This section is principally a review of some simple concepts of matrix algebra.

The identity matrix I consists of 1's on the diagonal and 0's elsewhere, and has the property that for any matrix A, IA=AI=A (here and elsewhere we assume that the dimensions of the matrices are such that the matrix multiplication is properly defined; all our matrices will in fact be square). We write the identity matrix of order N as I_N; thus for example

$$I_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

The identity matrix is obviously a diagonal matrix, and we could use the notation $I_3 = \text{diag}(1,1,1)$.

A permutation matrix P also has all its entries 0's and 1's; each row and each column contains just one non-zero element. For example, the following is a permutation matrix of order 5:

$$P = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

$$\text{If } \underline{x} = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \quad \text{then} \quad P\underline{x} = \begin{pmatrix} x_4 \\ x_2 \\ x_3 \\ x_0 \\ x_1 \end{pmatrix}.$$

In general, the rule for premultiplication of a vector $\underline{x}$ by a permutation matrix P is as follows: if $p_{ij}=1$, then element i of $P\underline{x}$ = element j of $\underline{x}$.

We can extend this rule to the multiplication of a matrix A by a permutation matrix P. For premultiplication by P: if $p_{ij}=1$, then row i of PA = row j of A. For postmultiplication by P: if $p_{ij} = 1$, then column j of AP = column i of A.

Example using 3 x 3 matrices:

$$\text{Let } A = \begin{pmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{pmatrix}, \quad P = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix},$$

i.e. $p_{01} = 1$, $p_{12} = 1$, $p_{20} = 1$. Then

$$PA = \begin{pmatrix} a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \\ a_{00} & a_{01} & a_{02} \end{pmatrix}, \quad AP = \begin{pmatrix} a_{02} & a_{00} & a_{01} \\ a_{12} & a_{10} & a_{11} \\ a_{22} & a_{20} & a_{21} \end{pmatrix}.$$

Perhaps a less familiar concept is the KRONECKER product $A \otimes B$ of two matrices (e.g. BELLMAN, 1970), defined as $A \otimes B = (a_{ij} B)$. For example, with the 3 x 3 matrix A defined above and any matrix B,

$$A \otimes B = \begin{pmatrix} a_{00}B & a_{01}B & a_{02}B \\ a_{10}B & a_{11}B & a_{12}B \\ a_{20}B & a_{21}B & a_{22}B \end{pmatrix}.$$

If A and B are both square, and of order N and M respectively, then $A \otimes B$ is a square matrix of order NM. (The KRONECKER product is in fact equally well-defined for rectangular matrices). The value of the notation $\otimes$ is that it provides a compact way of writing the kind of operations involved in the FFT. For example, the "building brick" from which the radix-2 FFT is constructed is the matrix

$$W_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

which replaces a pair of numbers by their sum and difference. Suppose for example we have a vector $\underline{x}$ with 6 elements, and that we apply the operator W_2 to the first, second and third pairs of adjacent numbers. We can write this as

$$\underline{y} = \begin{pmatrix} x_0 + x_1 \\ x_0 - x_1 \\ x_2 + x_3 \\ x_2 - x_3 \\ x_4 + x_5 \\ x_4 - x_5 \end{pmatrix} = \begin{pmatrix} 1 & 1 & & & 0 & \\ 1 & -1 & & & & \\ & & 1 & 1 & & \\ & & 1 & -1 & & \\ 0 & & & & 1 & 1 \\ & & & & 1 & -1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix}$$

$$= \begin{pmatrix} W_2 & & 0 \\ & W_2 & \\ 0 & & W_2 \end{pmatrix} \underline{x},$$

or more compactly as $\underline{y} = (I_3 \otimes W_2)\, \underline{x}$.

Suppose on the other hand that we pair x_i with x_{i+3} $(0 \leq i \leq 2)$ as follows:

$$\underline{y} = \begin{pmatrix} x_0 + x_3 \\ x_1 + x_4 \\ x_2 + x_5 \\ x_0 - x_3 \\ x_1 - x_4 \\ x_2 - x_5 \end{pmatrix} = \left(\begin{array}{cc:cc} 1 & & 1 & \\ & 1 & & 1 \\ & 1 & & -1 \\ 1 & & -1 & \\ & 1 & & -1 \\ & & 1 & -1 \end{array}\right) \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix}$$

$$= \begin{pmatrix} I_3 & I_3 \\ I_3 & -I_3 \end{pmatrix} \underline{x} \quad ,$$

which we can write more compactly as $\underline{y} = (W_2 \otimes I_3)\,\underline{x}$.

Two useful identities involving KRONECKER products are:

$$I_p \otimes I_q = I_{pq}$$

and

$$(I_p \otimes A)(I_p \otimes B)(I_p \otimes C) = I_p \otimes (ABC)$$

where A,B,C are square matrices of the same order.

Readers who refer to the paper by PEASE (1968) should note that his definition of the KRONECKER product is different from the above, and is non-standard.

The transpose A^T of a matrix A is defined as follows:

element (i,j) of A^T = element (j,i) of A, i.e. the matrix is reflected about its diagonal to produce its transpose. If A is symmetric, then $A^T = A$. The product rules are as follows: for the usual product, $(AB)^T = B^T A^T$. For the KRONECKER product, $(A \otimes B)^T = A^T \otimes B^T$. For a permutation matrix P, $P^T = P^{-1}$.

3. The GENTLEMAN-SANDE form of the FFT

We proceed now to the proof of one of the more common forms of the FFT. First, we write out explicitly the form of the matrix W_N of Eq. (2):

$$W_N = \begin{pmatrix} \omega^0 & \omega^0 & \omega^0 & \cdots & \omega^0 \\ \omega^0 & \omega^1 & \omega^2 & \cdots & \omega^{N-1} \\ \omega^0 & \omega^2 & \omega^4 & \cdots & \omega^{2(N-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ \omega^0 & \omega^{N-1} & \omega^{2(N-1)} & \cdots & \omega^{(N-1)(N-1)} \end{pmatrix}$$

where $\omega = \exp(2\pi i/N)$.

Now suppose that N=pq.

Define a diagonal matrix D_q^p of order pq as follows.

First, define a diagonal matrix of order q by

$\Delta_q = \mathrm{diag}\,(1, \omega, \omega^2, \ldots, \omega^{q-1})$, where $\omega = \exp(2\pi i/pq)$,

and from it construct the matrix $D_q^p = \mathrm{diag}\,(I_q, \Delta_q, (\Delta_q)^2, \ldots, (\Delta_q)^{p-1})$

Examples: let $\omega = \exp(2\pi i/6)$. Then

$$D_3^2 = \mathrm{diag}\,(\omega^0, \omega^0, \omega^0, \omega^0, \omega^1, \omega^2)$$

and $$D_2^3 = \mathrm{diag}\,(\omega^0, \omega^0, \omega^0, \omega^1, \omega^0, \omega^2).$$

Also, define a permutation matrix P_q^p by $p_{ij}=1$ if $i=rp+s$ and $j=sq+r$, where $0 \le r \le q-1$ and $0 \le s \le p-1$. To illustrate the form of the matrix P_q^p, we construct an example for p=2, q=3. In the following table, each value of $i (0 \le i \le 5)$ is written in the form rp+s (p=2) and the corresponding value of j=sq+r (q=3) is calculated:

i (=2r+s)	r	s	j(=3s+r)	
0	0	0	0	$p_{00}=1$
1	0	1	3	$p_{13}=1$
2	1	0	1	$p_{21}=1$
3	1	1	4	$p_{34}=1$
4	2	0	2	$p_{42}=1$
5	2	1	5	$p_{55}=1$

Thus the matrix P_3^2 has the following form:

$$P_3^2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

We now state the following important result.

Theorem 1: Let N=pq. Then with the definitions of P_q^p and D_q^p given above,

$$W_N = W_{pq} = P_q^p \,(I_p \otimes W_q)\, D_q^p \,(W_p \otimes I_q) \quad \ldots\ldots\ldots (3)$$

The proof of this theorem is given in Appendix 1.

Eq. (3) is precisely the GENTLEMAN-SANDE formulation of the FFT for two factors, expressed as a matrix factorization and including the final permutation. It can easily be extended to three factors:

$$W_{pqr} = W_{p(qr)} = P^{p}_{qr}\,(I_p \otimes W_{qr})\,D^{p}_{qr}\,(W_p \otimes I_{qr})$$

$$= P^{p}_{qr}\,(I_p \otimes \{P^{q}_{r}\,(I_q \otimes W_r)\,D^{q}_{r}\,(W_q \otimes I_r)\})\,D^{p}_{qr}\,(W_p \otimes I_{qr})$$

$$\therefore\quad W_{pqr} = P^{p}_{qr}\,(I_p \otimes P^{q}_{r})(I_{pq} \otimes W_r)(I_p \otimes D^{q}_{r})(I_p \otimes W_q \otimes I_r)\,D^{p}_{qr}\,(W_p \otimes I_{qr}) \qquad \ldots\ldots\ldots\ldots (4)$$

The generalization to an arbitrary number of factors is given in Appendix 2.

Eq. (4) shows how the permutation matrices collect at the left-hand end of the factorization; this is the feature of the GENTLEMAN-SANDE FFT which becomes difficult in the mixed-radix case.

To clarify the picture, we describe how the computation proceeds for the case $N=8=2^3$. Eq. (4) becomes:

$$W_8 = P^{2}_{4}\,(I_2 \otimes P^{2}_{2})\underbrace{(I_4 \otimes W_2)}\underbrace{(I_2 \otimes D^{2}_{2})(I_2 \otimes W_2 \otimes I_2)}\underbrace{D^{2}_{4}\,(W_2 \otimes I_4)}.$$

A flow diagram for the computation is shown in Fig. 1. The input to each square box consists of two complex numbers a and b; the output consists of the two complex numbers (a+b) and ω^k (a-b), where the integer k is the label on the box. The diagonal multipliers ω^k are combined with the transforms W_2 in this way, rather than as suggested by SINGLETON (1967), since the exponents k then appear in a more convenient order. Notice that the output from each box can overwrite the input, so that the transform part of the computation can be carried out in place, using a single array. In this case, the combined permutation $P^{2}_{4}(I_2 \otimes P^{2}_{2})$ can also be carried out in place, since it consists of pairwise exchanges $1 \leftrightarrow 4$, $3 \leftrightarrow 6$, the remaining elements staying in place. However, in the general case it becomes more difficult to carry out the permutation in place.

The programming strategy for the transform part of the general 3- factor case is as follows: in the first transform we collect p complex numbers N/p locations apart, apply first the transform Wp and then the diagonal multipliers, and overwrite the original locations with the results; one loop takes us through the data, both for indexing the array and for obtaining the diagonal multipliers in the right order. In the second transform we collect q complex numbers N/pq locations apart, apply the transform Wq, and the diagonal multipliers, again overwriting the original locations; this time p loops are required. In the third transform we collect r complex numbers N/pqr locations apart (i.e. they are adjacent), apply the transform Wr

(no diagonal multipliers are required this time), and overwrite the original locations; pq loops are required. It is easy to see how this strategy generalizes to any number of factors.

4. The COOLEY-TUKEY form of the FFT

Since W_N is symmetric, we can transpose both sides of Eq. (4). We use the identities $(AB)^T = B^T A^T$, $(A \otimes B)^T = A^T \otimes B^T$, the easily proved lemma $(P^p_q)^T = P^q_p$, and the fact that the I, W and D matrices of Eq. (4) are all symmetric. We obtain

$$W_{pqr} = (W_p \otimes I_{qr}) D^p_{qr} (I_p \otimes W_q \otimes I_r)(I_p \otimes D^q_r)(I_{pq} \otimes W_r)(I_p \otimes P^r_q) P^{qr}_p ,$$

or exchanging the rôles of p and r,

$$W_{pqr} = (W_r \otimes I_{pq}) D^r_{pq} (I_r \otimes W_q \otimes I_p)(I_r \otimes D^q_p)(I_{qr} \otimes W_p)(I_r \otimes P^p_q) P^{pq}_r \qquad \ldots\ldots\ldots\ldots (5)$$

Eq. (5) is the COOLEY-TUKEY form of the FFT for three factors; the general form is given in Appendix 2. It differs from the GENTLEMAN-SANDE form mainly in that the permutations are carried out before the transform sequence, rather than after it. Using the example $N=8=2^3$ again, we have:

$$W_8 = \underbrace{(W_2 \otimes I_4) D^2_4}\ \underbrace{(I_2 \otimes W_2 \otimes I_2)(I_2 \otimes D^2_2)}\ \underbrace{(I_4 \otimes W_2)}\ (I_2 \otimes P^2_2) P^4_2 .$$

The flow diagram for N=8 is shown in Fig. 2, where we have combined the diagonal and transform operations in a different way from the GENTLEMAN-SANDE form; the input to each diamond-shaped box consists of two complex numbers a and b; the output consists of two complex numbers $a + \omega^k b$ and $a - \omega^k b$, where k is the label on the box. Again the transform part of the program can be carried out in a single array, and the programming strategy for indexing the array and obtaining the diagonal multipliers is just the reverse of that for the GENTLEMAN-SANDE form.

5. PEASE'S form of the FFT

The GENTLEMAN-SANDE and COOLEY-TUKEY forms of the FFT involve a sequence of transforms, each of which has a different "flow diagram" from the others, even when all the factors of N are the same (as in the case $N=2^3$ presented above). PEASE (1968) derived a form such that

for $N=p^k$ (for some factor p, in particular p=2), each transform in the the sequence has the same flow diagram, thus simplifying the programming. DIXON (1976) has also considered this form.

The present analysis can be extended to derive PEASE's form of the FFT as a special case of a more general form for N a product of arbitrary factors. In the general case, each transform in the sequence clearly cannot have the same flow diagram, but they all have the same structure.

We require the following property of the permutation matrices P^p_q .

Theorem 2: For any square matrices Ap and Bq, of order p and q respectively,

$$P^p_q \,(A_p \otimes B_q) = (B_q \otimes A_p)\, P^p_q \qquad \ldots\ldots\ldots(6)$$

and $$(A_p \otimes B_q) = P^q_p \,(B_q \otimes A_p)\, P^p_q \qquad \ldots\ldots\ldots(7)$$

The proof is given in Appendix 1.

We now recall the 3-factor GENTLEMAN-SANDE form of the FFT:

$$Wpqr = P^p_{qr}\,(I_p \otimes P^q_r)(I_{pq} \otimes W_r)(I_p \otimes D^q_r)(I_p \otimes W_q \otimes I_r)\, D^p_{qr}\,(W_p \otimes I_{qr}) \qquad \ldots\ldots\ldots(8)$$

Using (7), we can write:

$$Ipq \otimes Wr = P^r_{pq}\,(W_r \otimes I_{pq})\, P^{pq}_r$$

$$Ip \otimes D^q_r = P^{qr}_p\,(D^q_r \otimes I_p)\, P^p_{qr}$$

and $$(Ip \otimes Wq \otimes Ir) = (I_p \otimes (W_q \otimes I_r)) = P^{qr}_p\,((W_q \otimes I_r) \otimes I_p)\, P^p_{qr}$$

$$= P^{qr}_p\,(W_q \otimes I_{pr})\, P^p_{qr} .$$

Substituting in Eq. (8) and using $P^p_{qr}\, P^{qr}_p = I_{pqr}$, we obtain:

$$Wpqr = P^p_{qr}\,(I_p \otimes P^q_r)\, P^r_{pq}\,(W_r \otimes I_{pq})\, P^{pq}_r\, P^{qr}_p\,(D^q_r \otimes I_p)(W_q \otimes I_{pr})\, P^p_{qr}\, D^p_{qr}\,(W_p \otimes I_{qr}) \qquad \ldots\ldots\ldots(9)$$

An analogue of the identity $P^p_q\ P^q_p = Ipq$, extended to three factors, is

$$P_r^{pq}\ P_p^{qr}\ P_q^{pr} = Ipqr$$

or

$$P_r^{pq}\ P_p^{qr} = P_{pr}^{q} .$$

Hence Eq. (9) becomes:

$$Wpqr = P_{qr}^{p}\,(I_p \otimes P_r^q)\,P_{pq}^{r}\,(W_r \otimes I_{pq})\,P_{pr}^{q}\,(D_r^q \otimes I_p)(W_q \otimes I_{pr})P_{qr}^{p}\,D_{qr}^{p}\,(W_p \otimes I_q \quad \text{.....(10)}$$

This expression for Wpqr has the same final sequence of permutation matrices as the GENTLEMAN-SANDE form, but each transform (using a factor p) is now followed by the permutation matrix $P_{N/p}^{p}$. The general form is given in Appendix 2. Using the example $N=2^3$ again, we have:

$$W_8 = P_4^2\,(I_2 \otimes P_2^2)\,\underbrace{P_4^2\,(W_2 \otimes I_4)}\,\underbrace{P_4^2\,(D_2^2 \otimes I_2)(W_2 \otimes I_4)}\,\underbrace{P_4^2\,D_4^2\,(W_2 \otimes I_4)}$$

in which all the transforms are of the form $(W_2 \otimes I_4)$, followed by a diagonal matrix, followed by P_4^2 (reading the matrix product from right to left). The corresponding flow diagram is shown in Fig. 3, where the notation is the same as for the GENTLEMAN-SANDE case (Fig. 1). Note that the 'internal' permutations are not carried out as separate operations, but are combined with the diagonal and transform matrices by storing the output from each 'box' directly into the appropriate locations. This means that the transforms can no longer be carried out in place; two arrays are required, acting alternately as input and output during the sequence of transforms. Notice that the diagonal multipliers for the second transform of the case $N=2^3$ are required in a different (and perhaps more convenient) order than for the GENTLEMAN-SANDE case, since their matrix representation is $(D_r^q \otimes Ip)$ rather than $(Ip \otimes D_r^q)$.

The programming strategy for each transform is now the same: we collect p complex numbers N/p locations apart, apply first the transform Wp and then the diagonal multipliers, and store the results in adjacent locations in the output array.

Just as the COOLEY-TUKEY FFT is a transposed form of the GENTLEMAN-SANDE FFT, so there is a 'transposed PEASE' FFT. Transposing both sides of Eq. (10) and exchanging the roles of p and r,

$$Wpqr = (W_r \otimes I_{pq})\,D_{pq}^{r}\,P_r^{pq}\,(W_q \otimes I_{pr})(D_p^q \otimes I_r)P_q^{pr}\,(W_p \otimes I_{qr})\,P_p^{qr}\,(I_r \otimes P_q^r)\,P_r^{pq} ,$$

the general form being given in Appendix 2.

Again we use the example $N=2^3$:

$$W_8 = \underbrace{(W_2 \otimes I_4)\, D_4^2\, P_2^4}\ \underbrace{(W_2 \otimes I_4)(D_2^2 \otimes I_2)\, P_2^4}\ \underbrace{(W_2 \otimes I_4)\, P_2^4}\ (I_2 \otimes P_2^2)\, P_2^4 ,$$

for which the flow diagram is given in Fig. 4, where the notation is the same as for the COOLEY-TUKEY formulation (Fig. 2). In the general case, we collect p adjacent complex numbers, apply first the diagonal multipliers and then the transform W_p, and finally store the results N/p locations apart in the output array. The initial permutation sequence, and the way in which the diagonal multipliers are combined with the component transforms, are the same as for the COOLEY-TUKEY form.

6. Mixed-radix FFT without reordering

The four forms of the FFT outlined so far all require a sequence of permutations to be applied either before or after the sequence of transforms; as mentioned in Section 1, this feature becomes somewhat troublesome in the mixed-radix case. In PEASE's form and its transpose (Section 5), simple permutations are also combined with the transforms themselves by suitable indexing, using two arrays alternately. In this Section, we show that this principle can be used to eliminate the troublesome permutation sequence altogether.

Recall the GENTLEMAN-SANDE form for Wpq:

$$W_{pq} = P_q^p\, (I_p \otimes W_q)\, D_q^p\, (W_p \otimes I_q) \qquad \ldots\ldots\ldots(11)$$

Applying the identity given by Eq. (6), we can reorganize the first two factors:

$$W_{pq} = (W_q \otimes I_p)\, P_q^p\, D_q^p\, (W_p \otimes I_q) \qquad \ldots\ldots\ldots(12)$$

The important difference between Eq. (12) and Eq. (11) is made clearer by extending the new representation to three factors:

$$W_{pqr} = W_{p(qr)} = (W_{qr} \otimes I_p)\, P_{qr}^p\, D_{qr}^p\, (W_p \otimes I_{qr})$$

$$= \left(\{(W_r \otimes I_q)\, P_r^q\, D_r^q\, (W_q \otimes I_r)\} \otimes I_p\right) P_{qr}^p\, D_{qr}^p\, (W_p \otimes I_{qr})$$

$$\therefore\ W_{pqr} = (W_r \otimes I_{pq})(P_r^q \otimes I_p)(D_r^q \otimes I_p)(W_q \otimes I_{pr})\, P_{qr}^p\, D_{qr}^p\, (W_p \otimes I_{qr}) \qquad \ldots\ldots\ldots(13)$$

Comparing Eq. (13) with Eq. (4), we see that the permutation matrices have been distributed through the factorization, and that each transform is now of the form $(W_p \otimes I_{N/p})$. The usual example should clarify the situation:

$$W_8 = \underbrace{(W_2 \otimes I_4)}\underbrace{(P_2^2 \otimes I_2)(D_2^2 \otimes I_2)(W_2 \otimes I_4)}\underbrace{P_4^2 D_4^2 (W_2 \otimes I_4)}$$

for which the flow diagram is given in Fig. 5, with the notation as for the GENTLEMAN-SANDE form. In the general case the input to each box consists of p complex numbers from locations N/p apart; in the box the transform Wp is followed by the diagonal multipliers, and the output is stored in locations m apart, starting in the first available location, where m is the product of the factors previously used (m=1 for the first transform). The diagonal multipliers are required in the same order as for PEASE's form. The final transform in the sequence requires no diagonal multipliers and no permutation, and could in fact be carried out in place.

As before, we can transpose both sides of Eq. (13) and exchange the rôles of p and r to obtain:

$$W_{pqr} = (W_r \otimes I_{pq}) D_{pq}^r P_r^{pq} (W_q \otimes I_{pr})(D_p^q \otimes I_r)(P_q^p \otimes I_r)(W_p \otimes I_{qr}) .$$

(The general forms of both this equation and Eq. (13) are given in Appendix 2). For $N=2^3$ we have:

$$W_8 = \underbrace{(W_2 \otimes I_4) D_4^2 P_2^4}\underbrace{(W_2 \otimes I_4)(D_2^2 \otimes I_2)(P_2^2 \otimes I_2)}\underbrace{(W_2 \otimes I_4)}$$

for which the flow diagram is given in Fig. 6, where the notation is as for the COOLEY-TUKEY formulation. Here the input to each box comes from locations N/mp apart, where m is the product of the factors previously used. The output from each box is stored N/p locations apart. This is the formulation used in the program given by UHRICH (1969), extended to the mixed-radix case.

7. Discussion

In this paper we have used factorization of the matrix form of the FFT to generate six different versions of the FFT algorithm. The GENTLEMAN-SANDE form and its transpose the COOLEY-TUKEY form have the advantage that the transform sequence can be carried out using a single array, but a permutation sequence is also required which necessitates extra programming and execution time, and becomes particularly complicated in the mixed-radix case. PEASE's form and its transpose have the advantage that

each of the component transforms has the same structure (in his paper, PEASE (1968) mentioned the suitability of this form for special-purpose FFT hardware design). However, two arrays are now needed, and the permutation sequence is still required. The new form proposed in Section 6, and its transpose the generalized UHRICH form, also require two arrays, but the permutation sequence is eliminated by combining a simple permutation with each component transform; the array indexing required is in fact no more complicated than for the more familiar COOLEY-TUKEY and GENTLEMAN-SANDE forms.

In choosing the most suitable form of the FFT for a particular application, the following considerations are the most important. If core storage is at a premium, then the GENTLEMAN-SANDE or COOLEY-TUKEY form is most appropriate. If simplicity of the component transforms is required, then PEASE's form or its transpose may be the best choice, though the diagonal multipliers still have to be indexed correctly, and the need for a permutation sequence must be borne in mind. In most other circumstances, the forms proposed in Section 6 appear to have clear advantages.

A preliminary study also indicates that, of all the forms of FFT presented here, those proposed in Section 6 may be the most suitable for the new generation of vector-processing computers. Certainly the permutation sequence required by the other forms cannot readily be handled by vector-processing techniques.

Other forms of the FFT can be generated by exchanging the order of the permutation and diagonal matrices (using $P_q^p D_q^p = D_q^q P_q^p$) or by grouping the factors differently, but these minor variations seem to lead either to the same program as before, or to more complicated indexing requirements. It is possible, of course, that quite new variants remain to be discovered. If so, the analysis presented in this paper may provide a basis for finding them.

8. Implementation

A FORTRAN routine has been written at ECMWF to implement the proposed new form of the FFT. To date it has been used for real FOURIER periodic and sine transforms, applying the procedures of COOLEY, LEWIS and WELCH (1970) to convert a real transform of length N to a complex transform of length N/2.

A FORTRAN program (FFFT) to perform a complex fast FOURIER transform of arbitrary length N is presented in Appendix 3. For each factor of N, FFFT calls the subroutine PASS, which is written in such a way that the real and imaginary parts of the complex numbers can be in independent arrays, and

the addressing increment for these arrays can be specified. (These facilities are required when performing multi-dimensional transforms, and in certain circumstances for one-dimensional real transforms).

PASS contains sections of coding for factors 2,3,4,5 and general odd factors $\geq$ 7. SINGLETON (1969) demonstrated that if the factorization of N included a number of 2's, these could be more efficiently treated by grouping them together in pairs; hence the coding for factor 4. In the same paper, SINGLETON also presented an efficient scheme for handling odd prime factors$\geq$5, which has been used in subroutine PASS.

Timing comparisons have been made between the routine FFFT presented here and a FORTRAN FFT routine written by NORMAN BRENNER of MIT, which has also been used at ECMWF. BRENNER's routine is based on the COOLEY-TUKEY version of the FFT; it too includes special coding for factor 4, but not for factor 5.

In Table 1, the respective CPU times on a CDC 6600 are presented for complex transforms of length N, for various values of N. Although the floating point operation count for the transforms is the same for both algorithms, FFFT is consistently twice as fast as BRENNER's program. Some of this increase in speed is due to the fact that FFFT uses a precalculated table of trigonometric function values for the diagonal multipliers; the remainder results from the elimination of the permutation sequence.

Experiments with a special-purpose radix-2 version of the new FFT algorithm indicated that a further increase in speed of around 30% could be achieved by reorganizing the FORTRAN loop structure and placing the vectors in blank COMMON.

9. Acknowledgements

The author wishes to thank ECMWF staff members for reviewing the manuscript, and Miss A. Cara for typing it. Some of this work was carried out at the U.K Meteorological office.

Table 1

CDC 6600 CPU times (in milliseconds) for complex transforms of length N

N	Brenner's routine	New routine (FFFT)
32 = 2 x 4 x 4	5.41	2.86
36 = 3 x 3 x 4	6.62	2.99
48 = 3 x 4 x 4	6.82	3.55
49 = 7 x 7	9.23	4.55
50 = 2 x 5 x 5	10.35	4.33
64 = 4 x 4 x 4	8.78	4.41
96 = 2 x 3 x 4 x 4	14.64	6.86
100 = 4 x 5 x 5	20.30	7.50
121 = 11 x 11	24.03	11.74
128 = 2 x 4 x 4 x 4	19.54	8.85

Appendix 1:

Proof of Theorem 1

We wish to show that

$$W_N = W_{pq} = P_q^p \, (I_p \otimes W_q) \, D_q^p \, (W_p \otimes I_q),$$

where the matrices W, P and D are defined in sections 1 and 3.

Firstly, note that the matrices $(I_p \otimes W_q)$, D_q^p and $(W_p \otimes I_q)$, all of order pq, can all be conveniently partitioned into p^2 square blocks, each block being of order q. For instance,

$$W_p \otimes I_q = \begin{pmatrix} \omega^0 I_q & \omega^0 I_q & \omega^0 I_q & \cdots & \omega^0 I_q \\ \omega^0 I_q & \omega^q I_q & \omega^{2q} I_q & \cdots & \omega^{(p-1)q} I_q \\ \omega^0 I_q & \omega^{2q} I_q & \omega^{4q} I_q & \cdots & \omega^{2(p-1)q} I_q \\ \vdots & \vdots & \vdots & & \vdots \\ \omega^0 I_q & \omega^{(p-1)q} I_q & \omega^{2(p-1)q} I_q & \cdots & \omega^{(p-1)(p-1)q} I_q \end{pmatrix}$$

where $\omega = \exp(2\pi i / pq)$ (and hence $\omega^q = \exp(2\pi i / p)$), leading to the appearance of the factor q in the exponents of ω in the above representation of $(W_p \otimes I_q)$.

Secondly, with this partitioning the matrices $(I_p \otimes W_q)$ and D_q^p are block diagonal, and hence the product

$$\overline{W}_{pq} = (I_p \otimes W_q) \, D_q^p \, (W_p \otimes I_q)$$

can easily be computed block by block.

Let the blocks be indexed in the same way as the individual matrix entries. Then block (m,n) ($0 \le m \le p-1$, $0 \le n \le p-1$) of $(W_p \otimes I_q)$ is $\omega^{mnq} I_q$. The mth diagonal block of D_q^p is $(\Delta_q)^m$, and the mth diagonal block of $(I_p \otimes W_q)$ is W_q. Thus block (m,n) of $\overline{W}_{pq} = (I_p \otimes W_q) D_q^p (W_p \otimes I_q)$ is $\omega^{mnq} W_q (\Delta_q)^m$.

Finallly, premultiplication by the permutation matrix P_q^p permutes the rows of $\overline{W}_{pq}$. Recall that $p_{ij} = 1$ if $i = rp+s$ and $j = sq + r$ ($0 \le r \le q-1$, $0 \le s \le p-1$).

So row $(rp + s)$ of $P_q^p \, \overline{W}_{pq}$ = row $(sq + r)$ of $\overline{W}_{pq}$.

We now show that element (i,j) of $P_q^p \, \overline{W}_{pq}$ = element

(i,j) of Wpq. Suppose $i = rp + s$, $j = kq + \ell$ ($0 \le r \le q-1$, $0 \le s \le p-1$, $0 \le k \le p-1$, $0 \le \ell \le q-1$).

Then element (i,j) = element $(rp + s, kq + \ell)$ of $P_q^p \, \overline{W}_{pq}$

= element $(sq + r, kq + \ell)$ of $\overline{W}pq$

= element (r, ℓ) of block (s,k) of $\overline{W}pq$

= element (r, ℓ) of $\omega^{skq} W_q (\Delta_q)^s$.

Now

$$W_q (\Delta_q)^s = \begin{pmatrix} \omega^0 & \omega^0 & \omega^0 & \dots & \omega^0 \\ \omega^0 & \omega^p & \omega^{2p} & \dots & \omega^{(q-1)p} \\ \omega^0 & \omega^{2p} & \omega^{4p} & \dots & \omega^{2(q-1)p} \\ \vdots & \vdots & \vdots & & \vdots \\ \omega^0 & \omega^{(q-1)p} & \omega^{2(q-1)p} & \dots & \omega^{(q-1)(q-1)p} \end{pmatrix} \begin{pmatrix} \omega^0 & & & 0 \\ & \omega^s & & \\ & & \omega^{2s} & \\ & & & \ddots \\ 0 & & & \omega^{(q-1)s} \end{pmatrix}$$

where $\omega = \exp (2\pi i / pq)$, and hence $\omega^p = \exp(2\pi i / q)$.

So element (r, ℓ) of $\omega^{skq} W_q (\Delta_q)^s$

$= \omega^{skq} \omega^{r\ell p} \omega^{\ell s}$

$= \omega^{rpkq} \omega^{skq} \omega^{r\ell p} \omega^{\ell s}$ since $\omega^{pq} = 1$

$= \omega^{(rp+s)(kq+\ell)}$

$= \omega^{ij}$

= element (i,j) of Wpq = W_N , as required.

Hence $Wpq = P_q^p \, \overline{W}_{pq} = P_q^p (I_p \otimes W_q) D_q^p (W_p \otimes I_q)$.

Proof of Theorem 2

We wish to show that

$$P_q^p (A_p \otimes B_q) = (B_q \otimes A_p) P_q^p .$$

We show that element (i,j) of the matrix on the left hand side of the equation is the same as the corresponding element of the matrix on the right hand side. Let $i = rp+s$, $j = kq+\ell$ $(0 \le r \le q-1, 0 \le s \le p-1, 0 \le k \le p-1, 0 \le \ell \le q-1)$.

Since premultiplication by P_q^p permutes the rows, element $(rp + s , kq + \ell)$ of $P_q^p(Ap \otimes Bq)$ = element $(sq + r, kq + \ell)$ of $Ap \otimes Bq = a_{sk} b_{r\ell}$ (by considering the block structure of $Ap \otimes Bq$). Similarly, since postmultiplication by P_q^p permutes the columns, element $(rp + s, kq + \ell)$ of $(Bq \otimes Ap) P_q^p$ = element $(rp + s, \ell p + k)$ of $Bq \otimes Ap = b_{r\ell} a_{sk}$. Hence Theorem 2 is proved, and the corollary

$$Ap \otimes Bq = P_p^q (B_q \otimes A_p) P_q^p$$

follows immediately.

Appendix 2: General Forms

For the six formulations of the FFT presented above, the general forms are given below for N the product of any number of arbitrary factors. We assume

$$N = \prod_{i=1}^{k} n_i$$

and that the factors are used in the order $n_1, n_2, \ldots, n_k$.

Define

$$m_i = \prod_{j=1}^{i-1} n_j \quad (m_1 = 1), \qquad l_i = N/m_i n_i = \prod_{j=i+1}^{k} n_j \quad (l_k = 1).$$

The notation $\prod_{i=k}^{1}$ for a matrix product indicates that the product is written out in reverse order, e.g.

$$\prod_{i=k}^{1} A_i = A_k A_{k-1} A_{k-2} \cdots A_2 A_1 .$$

Note that $P_1^p = D_1^p = I_p$.

GENTLEMAN-SANDE: $W_N = \prod_{i=1}^{k} \left(I_{m_i} \otimes P_{l_i}^{n_i} \right) \prod_{i=k}^{1} \left\{ \left(I_{m_i} \otimes D_{l_i}^{n_i} \right) \left(I_{m_i} \otimes W_{n_i} \otimes I_{l_i} \right) \right.$

COOLEY-TUKEY: $W_N = \prod_{i=k}^{1} \left\{ \left(I_{l_i} \otimes W_{n_i} \otimes I_{m_i} \right) \left(I_{l_i} \otimes D_{m_i}^{n_i} \right) \right\} \prod_{i=1}^{k} \left(I_{l_i} \otimes P_{n_i}^{m_i} \right)$

PEASE: $W_N = \prod_{i=1}^{k} \left(I_{m_i} \otimes P_{l_i}^{n_i} \right) \prod_{i=k}^{1} \left\{ P_{N/n_i}^{n_i} \left(D_{l_i}^{n_i} \otimes I_{m_i} \right) \left(W_{n_i} \otimes I_{N/n_i} \right) \right\}$

Transposed PEASE:

$$W_N = \prod_{i=k}^{1} \left\{ \left(W_{n_i} \otimes I_{N/n_i} \right) \left(D_{m_i}^{n_i} \otimes I_{l_i} \right) P_{n_i}^{N/n_i} \right\} \prod_{i=1}^{k} \left(I_{l_i} \otimes P_{n_i}^{m_i} \right)$$

New form: $W_N = \prod_{i=k}^{1} \left\{ \left(P_{l_i}^{n_i} \otimes I_{m_i} \right) \left(D_{l_i}^{n_i} \otimes I_{m_i} \right) \left(W_{n_i} \otimes I_{N/n_i} \right) \right\}$

Generalized UHRICH:

$$W_N = \prod_{i=k}^{1} \left\{ \left(W_{n_i} \otimes I_{N/n_i} \right) \left(D_{m_i}^{n_i} \otimes I_{l_i} \right) \left(P_{n_i}^{m_i} \otimes I_{l_i} \right) \right\}$$

Appendix 3: A FORTRAN FFT program

```
C     SUBROUTINE 'FFFT' - PERFORMS A COMPLEX FAST FOURIER TRANSFORM
C     ON A COMPLEX INPUT VECTOR C OF LENGTH N:
C
C     X(J)=SUM(K=0,...,N-1)(C(K)*EXP(2*I*J*K*PI/N)),  J=0,...,N-1,
C     WHERE I=SQRT(-1)
C
C     CALL FFFT(C,X,TRIGS,WORK,IFAX,NFAX,N)
C
C     C IS THE COMPLEX INPUT VECTOR (TREATED HERE AS REAL, SO THAT
C        THE ADDRESS OF THE FIRST IMAGINARY PART CAN BE SUPPLIED TO
C        SUBROUTINE 'PASS')
C     X IS THE COMPLEX OUTPUT VECTOR (SIMILARLY TREATED HERE AS REAL)
C     TRIGS IS A COMPLEX ARRAY OF DIMENSION N, PREVIOUSLY SET UP,
C        CONTAINING THE VALUES TRIGS(K)=EXP(2*I*(K-1)*PI/N),
C        WHERE I=SQRT(-1)
C     WORK IS A WORK AREA OF (COMPLEX) DIMENSION N
C     IFAX IS AN INTEGER ARRAY CONTAINING THE FACTORS OF N
C     NFAX IS THE NUMBER OF FACTORS OF N: THUS
C        N=IFAX(1)*IFAX(2)*...*IFAX(NFAX)
C
      SUBROUTINE FFFT(C,X,TRIGS,WORK,IFAX,NFAX,N)
      DIMENSION C(N),X(N),TRIGS(N),WORK(N),IFAX(NFAX)
      LA=1
      IF (NFAX.GT.1) GO TO 10
C
C     ONLY ONE FACTOR
C
      CALL PASS(C(1),C(2),X(1),X(2),TRIGS,2,2,N,IFAX(1),LA)
      RETURN
C
   10 IF (MOD(NFAX,2).EQ.1) GO TO 20
C
C     FIRST FACTOR, NFAX EVEN
C
      CALL PASS(C(1),C(2),WORK(1),WORK(2),TRIGS,2,2,N,IFAX(1),LA)
      LA=IFAX(1)
      INEXT=50
      GO TO 30
C
C     FIRST FACTOR, NFAX ODD
C
   20 CALL PASS(C(1),C(2),X(1),X(2),TRIGS,2,2,N,IFAX(1),LA)
      LA=IFAX(1)
      INEXT=40
C
C     MAIN LOOP. STATEMENTS 40 & 50 ARE EXECUTED ON ALTERNATE
C     PASSES THROUGH THE LOOP
C
   30 DO 70 L=2,NFAX
      IF (INEXT.EQ.50) GO TO 50
   40 CALL PASS(X(1),X(2),WORK(1),WORK(2),TRIGS,2,2,N,IFAX(L),LA)
      INEXT=50
      GO TO 60
   50 CALL PASS(WORK(1),WORK(2),X(1),X(2),TRIGS,2,2,N,IFAX(L),LA)
      INEXT=40
   60 LA=IFAX(L)*LA
   70 CONTINUE
C
      RETURN
      END
```

```
C      SUBROUTINE 'PASS' - PERFORMS ONE PASS THROUGH DATA AS PART OF
C      COMPLEX FFT ROUTINE
C      A IS REAL INPUT VECTOR
C      B IS IMAGINARY INPUT VECTOR
C      C IS REAL OUTPUT VECTOR
C      D IS IMAGINARY OUTPUT VECTOR
C      TRIGS IS PRECALCULATED TABLE OF SINES AND COSINES
C      INC1 IS ADDRESSING INCREMENT FOR A & B
C      INC2 IS ADDRESSING INCREMENT FOR C & D
C      N IS LENGTH OF VECTORS
C      IFAC IS CURRENT FACTOR OF N
C      LA IS PRODUCT OF PREVIOUS FACTORS
C
       SUBROUTINE PASS(A,B,C,D,TRIGS,INC1,INC2,N,IFAC,LA)
       DIMENSION A(1),B(1),C(1),D(1),TRIGS(1)
       DATA SIN36/0.587785252292473/,COS36/0.809016994374947/,
      *     SIN72/0.951056516295154/,COS72/0.309016994374947/,
      *     SIN60/0.866025403784437/
C
       M=N/IFAC
       IINK=M*INC1
       JINK=LA*INC2
       JUMP=(IFAC-1)*JINK
       IA=1
       JA=1
       IGO=IFAC-1
       IF (IGO.GT.5) IGO=5
       GO TO (10,40,70,100,130),IGO
C
C      CODING FOR FACTOR 2
C
   10  DO 30 K=1,M,LA
       KB=K+K-2
       DO 20 L=1,LA
       IB=IA+IINK
       JB=JA+JINK
       C(JA)=A(IA)+A(IB)
       D(JA)=B(IA)+B(IB)
       C(JB)=A(IA)-A(IB)
       D(JB)=B(IA)-B(IB)
       IA=IA+INC1
       IF (KB.EQ.0) GO TO 20
       TEMPR=C(JB)*TRIGS(KB+1)-D(JB)*TRIGS(KB+2)
       TEMPI=C(JB)*TRIGS(KB+2)+D(JB)*TRIGS(KB+1)
       C(JB)=TEMPR
       D(JB)=TEMPI
   20  JA=JA+INC2
   30  JA=JA+JUMP
       RETURN
C
C      CODING FOR FACTOR 3
C
   40  DO 60 K=1,M,LA
       KB=K+K-2
       KC=KB+KB
       DO 50 L=1,LA
       IB=IA+IINK
```

```
      IC=IB+IINK
      JB=JA+JINK
      JC=JB+JINK
      A1=A(IB)+A(IC)
      B1=B(IB)+B(IC)
      A2=A(IA)-0.5*A1
      B2=B(IA)-0.5*B1
      A3=SIN60*(A(IB)-A(IC))
      B3=SIN60*(B(IB)-B(IC))
      C(JA)=A(IA)+A1
      D(JA)=B(IA)+B1
      C(JB)=A2-B3
      D(JB)=B2+A3
      C(JC)=A2+B3
      D(JC)=B2-A3
      IA=IA+INC1
      IF (KB.EQ.0) GO TO 50
      TEMPR=C(JB)*TRIGS(KB+1)-D(JB)*TRIGS(KB+2)
      TEMPI=C(JB)*TRIGS(KB+2)+D(JB)*TRIGS(KB+1)
      C(JB)=TEMPR
      D(JB)=TEMPI
      TEMPR=C(JC)*TRIGS(KC+1)-D(JC)*TRIGS(KC+2)
      TEMPI=C(JC)*TRIGS(KC+2)+D(JC)*TRIGS(KC+1)
      C(JC)=TEMPR
      D(JC)=TEMPI
   50 JA=JA+INC2
   60 JA=JA+JUMP
      RETURN
C
C     CODING FOR FACTOR 4
C
   70 DO 90 K=1,M,LA
      KB=K+K-2
      KC=KB+KB
      KD=KC+KB
      DO 80 L=1,LA
      IB=IA+IINK
      IC=IB+IINK
      ID=IC+IINK
      JB=JA+JINK
      JC=JB+JINK
      JD=JC+JINK
      A0=A(IA)+A(IC)
      A1=A(IB)+A(ID)
      A2=A(IA)-A(IC)
      A3=A(IB)-A(ID)
      B0=B(IA)+B(IC)
      B1=B(IB)+B(ID)
      B2=B(IA)-B(IC)
      B3=B(IB)-B(ID)
      C(JA)=A0+A1
      D(JA)=B0+B1
      C(JB)=A2-B3
      D(JB)=B2+A3
      C(JC)=A0-A1
      D(JC)=B0-B1
      C(JD)=A2+B3
```

```
      D(JD)=B2-A3
      IA=IA+INC1
      IF (KB.EQ.0) GO TO 80
      TEMPR=C(JB)*TRIGS(KB+1)-D(JB)*TRIGS(KB+2)
      TEMPI=C(JB)*TRIGS(KB+2)+D(JB)*TRIGS(KB+1)
      C(JB)=TEMPR
      D(JB)=TEMPI
      TEMPR=C(JC)*TRIGS(KC+1)-D(JC)*TRIGS(KC+2)
      TEMPI=C(JC)*TRIGS(KC+2)+D(JC)*TRIGS(KC+1)
      C(JC)=TEMPR
      D(JC)=TEMPI
      TEMPR=C(JD)*TRIGS(KD+1)-D(JD)*TRIGS(KD+2)
      TEMPI=C(JD)*TRIGS(KD+2)+D(JD)*TRIGS(KD+1)
      C(JD)=TEMPR
      D(JD)=TEMPI
   80 JA=JA+INC2
   90 JA=JA+JUMP
      RETURN
C
C     CODING FOR FACTOR 5
C
  100 DO 120 K=1,M,LA
      KB=K+K-2
      KC=KB+KB
      KD=KC+KB
      KE=KD+KB
      DO 110 L=1,LA
      IB=IA+IINK
      IC=IB+IINK
      ID=IC+IINK
      IE=ID+IINK
      JB=JA+JINK
      JC=JB+JINK
      JD=JC+JINK
      JE=JD+JINK
      A1=A(IB)+A(IE)
      A2=A(IC)+A(ID)
      A3=A(IB)-A(IE)
      A4=A(IC)-A(ID)
      B1=B(IB)+B(IE)
      B2=B(IC)+B(ID)
      B3=B(IB)-B(IE)
      B4=B(IC)-B(ID)
      A10=A(IA)+A1*COS72-A2*COS36
      A11=B3*SIN72+B4*SIN36
      A20=A(IA)-A1*COS36+A2*COS72
      A21=B3*SIN36-B4*SIN72
      B10=B(IA)+B1*COS72-B2*COS36
      B11=A3*SIN72+A4*SIN36
      B20=B(IA)-B1*COS36+B2*COS72
      B21=A3*SIN36-A4*SIN72
      C(JA)=A(IA)+A1+A2
      D(JA)=B(IA)+B1+B2
      C(JB)=A10-A11
      D(JB)=B10+B11
      C(JC)=A20-A21
      D(JC)=B20+B21
```

```
      C(JD)=A20+A21
      D(JD)=B20-B21
      C(JE)=A10+A11
      D(JE)=B10-B11
      IA=IA+INC1
      IF (KB.EQ.0) GO TO 110
      TEMPR=C(JB)*TRIGS(KB+1)-D(JB)*TRIGS(KB+2)
      TEMPI=C(JB)*TRIGS(KB+2)+D(JB)*TRIGS(KB+1)
      C(JB)=TEMPR
      D(JB)=TEMPI
      TEMPR=C(JC)*TRIGS(KC+1)-D(JC)*TRIGS(KC+2)
      TEMPI=C(JC)*TRIGS(KC+2)+D(JC)*TRIGS(KC+1)
      C(JC)=TEMPR
      D(JC)=TEMPI
      TEMPR=C(JD)*TRIGS(KD+1)-D(JD)*TRIGS(KD+2)
      TEMPI=C(JD)*TRIGS(KD+2)+D(JD)*TRIGS(KD+1)
      C(JD)=TEMPR
      D(JD)=TEMPI
      TEMPR=C(JE)*TRIGS(KE+1)-D(JE)*TRIGS(KE+2)
      TEMPI=C(JE)*TRIGS(KE+2)+D(JE)*TRIGS(KE+1)
      C(JE)=TEMPR
      D(JE)=TEMPI
  110 JA=JA+INC2
  120 JA=JA+JUMP
      RETURN
C
C     CODING FOR GENERAL ODD FACTOR
C
  130 KT=M+M
      LINK=(IFAC-1)*IINK
      MINK=LINK/2
      NN=N+N
      DO 190 K=1,M,LA
      KINK=K+K-2
      DO 180 L=1,LA
      IB=IA+IINK
      IM=IA+MINK
      IZ=IA+LINK
      IQ=IZ
      SUM1=A(IA)
      SUM2=B(IA)
      DO 140 IP=IB,IM,IINK
      TEMPR=A(IP)+A(IQ)
      A(IQ)=A(IP)-A(IQ)
      A(IP)=TEMPR
      SUM1=SUM1+TEMPR
      TEMPI=B(IP)+B(IQ)
      B(IQ)=B(IP)-B(IQ)
      B(IP)=TEMPI
      SUM2=SUM2+TEMPI
  140 IQ=IQ-IINK
      C(JA)=SUM1
      D(JA)=SUM2
      JP=JA+JINK
      JQ=JA+JUMP
      KU=KT
      DO 160 I=IB,IM,IINK
```

```
      SUM1=A(IA)
      SUM2=0.0
      SUM3=B(IA)
      SUM4=0.0
      KV=KU
      IQ=IZ
      DO 150 IP=IB,IM,IINK
      IF (KV.GE.NN) KV=KV-NN
      SUM1=SUM1+A(IP)*TRIGS(KV+1)
      SUM2=SUM2+B(IQ)*TRIGS(KV+2)
      SUM3=SUM3+B(IP)*TRIGS(KV+1)
      SUM4=SUM4+A(IQ)*TRIGS(KV+2)
      KV=KV+KU
  150 IQ=IQ-IINK
      C(JP)=SUM1-SUM2
      D(JP)=SUM3+SUM4
      C(JQ)=SUM1+SUM2
      D(JQ)=SUM3-SUM4
      JP=JP+JINK
      JQ=JQ-JINK
  160 KU=KU+KT
      IA=IA+INC1
      IF (KINK.EQ.0) GO TO 180
      KU=KINK
      JP=JA+JINK
      JQ=JA+JUMP
      DO 170 J=JP,JQ,JINK
      TEMPR=C(J)*TRIGS(KU+1)-D(J)*TRIGS(KU+2)
      TEMPI=C(J)*TRIGS(KU+2)+D(J)*TRIGS(KU+1)
      C(J)=TEMPR
      D(J)=TEMPI
  170 KU=KU+KINK
  180 JA=JA+INC2
  190 JA=JA+JUMP
      RETURN
      END
```

References

Bellman, R. (1970) "Introduction to Matrix Analysis", Mc Graw-Hill, 2nd Edition.

Bourke, W. (1972) "An efficient, one-level, primitive-equation spectral model", Monthly Weather Review 100, 683-689.

Cooley, J. W. and Tukey, J. W. (1965) "An algorithm for the machine calculation of complex Fourier series", Math. Comp. 19, 297-301.

Cooley, J. W. Lewis, P. A. W. and Welch, P. D. (1970) "The Fast Fourier Transform algorithm: programming considerations in the calculation of sine, cosine and Laplace transforms", J. Sound Vib. 12, 315-337.

Daley, R. Girard, C. Henderson, J. and Simmonds, I. (1976) "Short-term forecasting with a multi-level spectral primitive equation model. Part II - hemispheric prognoses and verifications', Atmosphere 14, 117-134.

Dixon, R. (1976) "A special version of the FFT", Met.O 11 Tech. Note 54 (unpublished Meteorological Office Technical Note).

Gentleman, W. M. and Sande, G. (1966) "Fast Fourier Transforms - for fun and profit", 1966 Fall Joint Computer Conference, AFIPS Proc. 29, 563-578.

Holloway, J. L. Spelman, M. J. and Manabe, S. (1973) "Latitude-longitude grid suitable for numerical time integration of a global atmospheric model", Monthly Weather Review 101, 69-78.

Merilees, P. E. (1973) "The pseudospectral approximation applied to the shallow water equations on a sphere", Atmosphere 11, 13-20.

Paegle, J. and Tomlinson, E. M. (1975) "Solution of the balance equation by Fourier transform and Gauss elimination", Monthly Weather Review 103, 528-535.

Pease, M. C. (1968) "An adaptation of the Fast Fourier Transform for parallel processing", JACM 15, 252-264.

Singleton, R. C. (1967) "On computing the Fast Fourier Transform", CACM 10, 647-654.

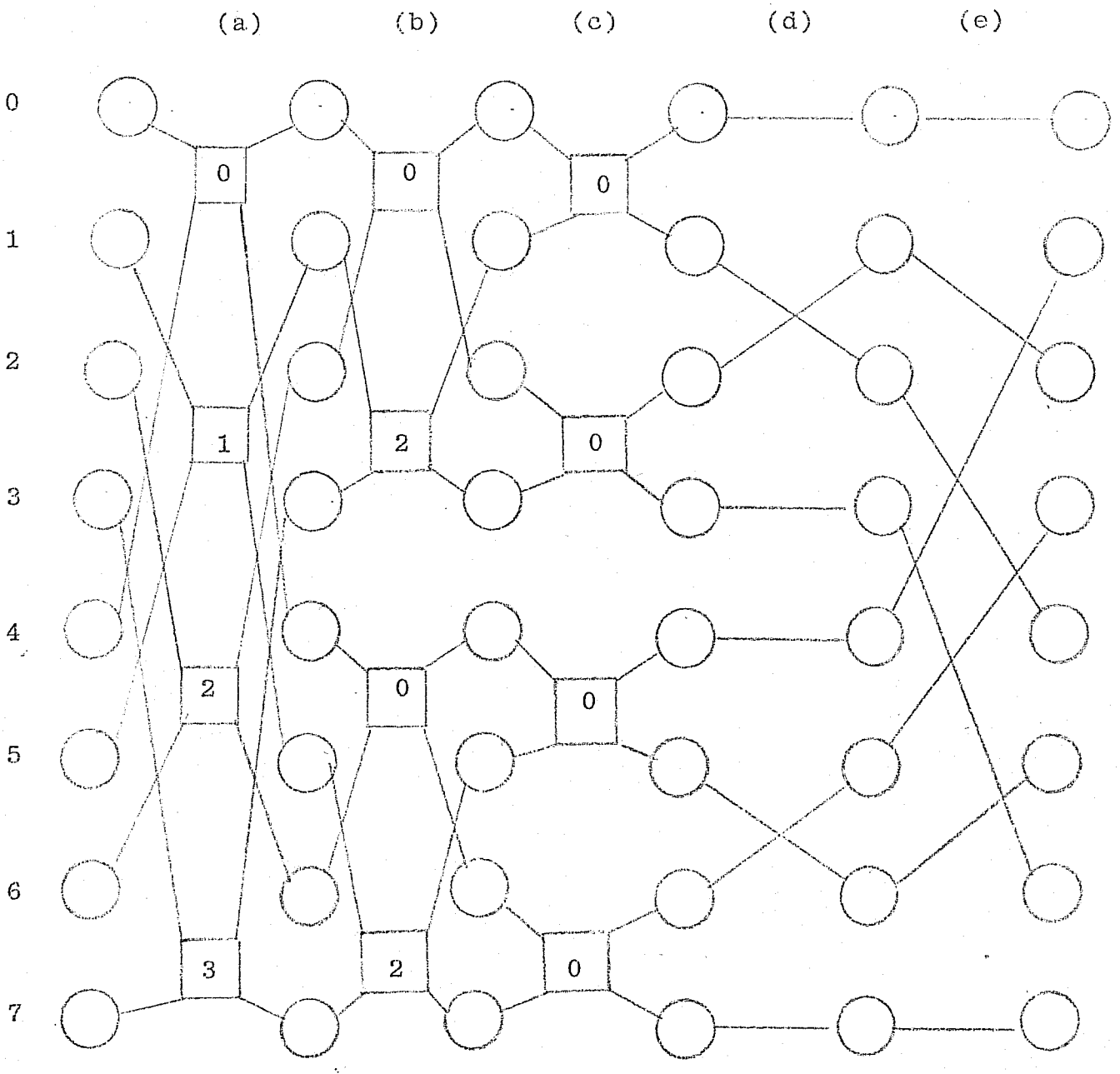
Singleton, R. C. (1969) "An algorithm for computing the mixed radix Fast Fourier Transform", IEEE Transactions on Audio and Electroacoustics, 17, 93-103.

Swarztrauber, P. N. (1974) "The direct solution of the discrete Poisson equation on the surface of a sphere", J. Comp. Phys. 15, 46-54.

Temperton, C. (1976) "An all-purpose mixed-radix Fast Fourier Transform program", Met. O. 2b Tech. Note No. 25 (unpublished Meteorological office Technical Note).

Theilheimer, F. (1969) "A matrix version of the Fast Fourier Transform", IEEE Transactions on Audio and Electroacoustics, 17, 158-161.

Uhrich, M. L. (1969) "Fast Fourier Transforms without sorting", IEEE Transactions on Audio and Electroacoustics, 17, 170-172.



(a) $D_4^2 \quad (W_2 \otimes I_4)$

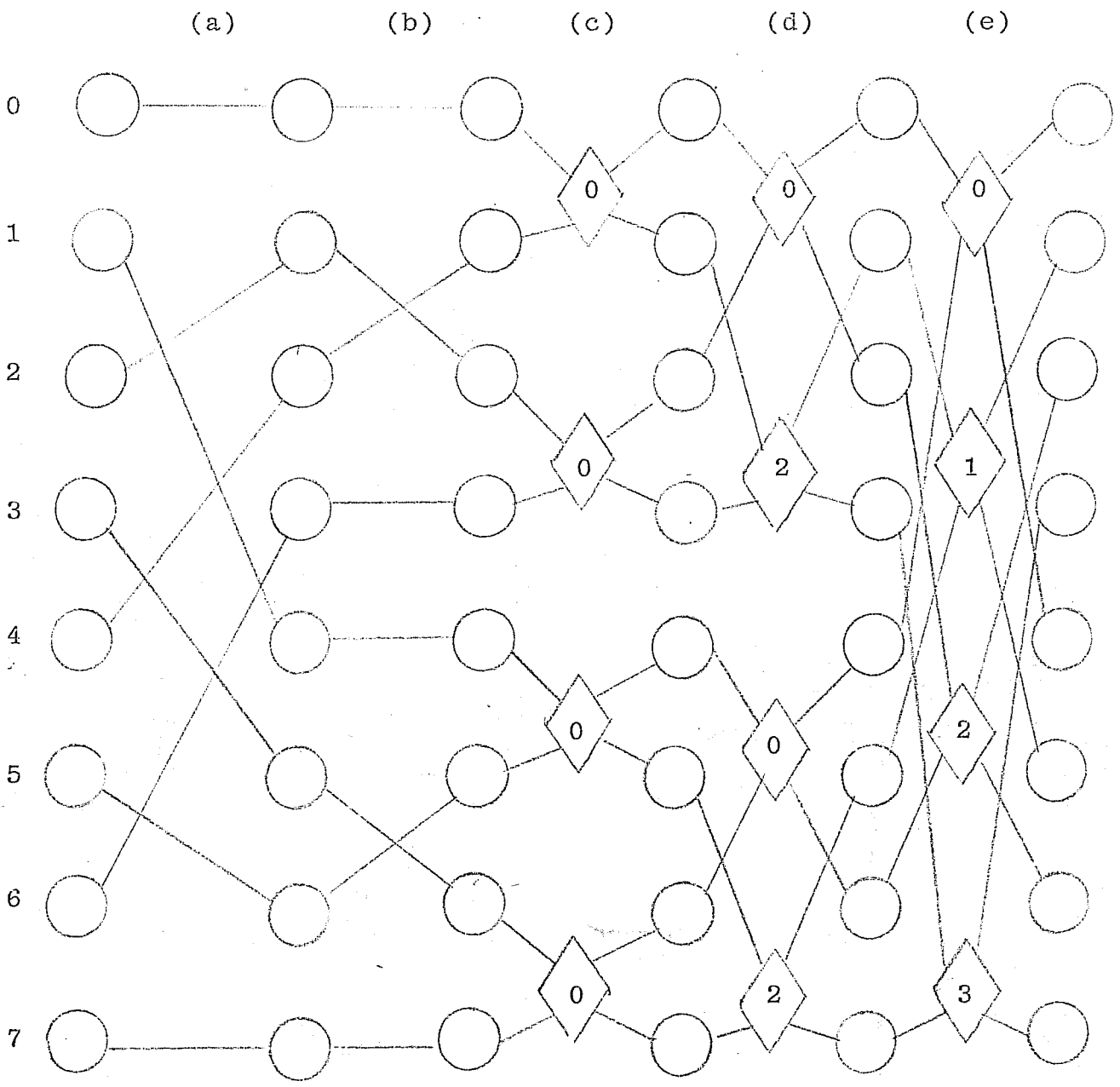
(b) $(I_2 \otimes D_2^2)(I_2 \otimes W_2 \otimes I_2)$

(c) $I_4 \otimes W_2$

(d) $I_2 \otimes P_2^2$

(e) P_4^2

Fig. 1 The Gentleman-Sande FFT for $N=2^3$.



(a) P_2^4

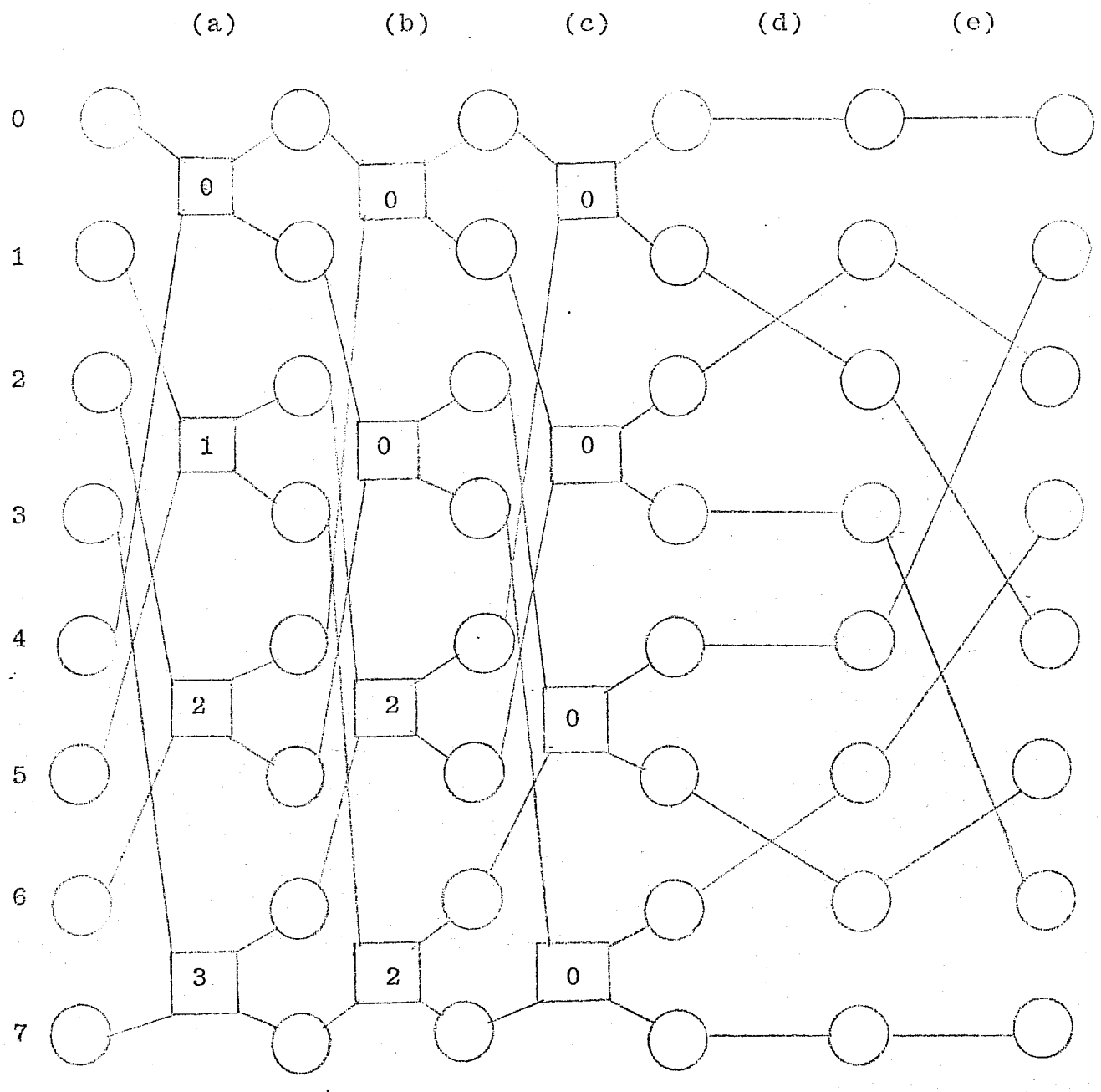
(b) $I_2 \otimes P_2^2$

(c) $I_4 \otimes W_2$

(d) $(I_2 \otimes W_2 \otimes I_2)(I_2 \otimes D_2^2)$

(e) $(W_2 \otimes I_4)\, D_4^2$

Fig. 2 the Cooley-Tukey FFT for $N=2^3$.



(a) $P_4^2 D_4^2 (W_2 \otimes I_4)$

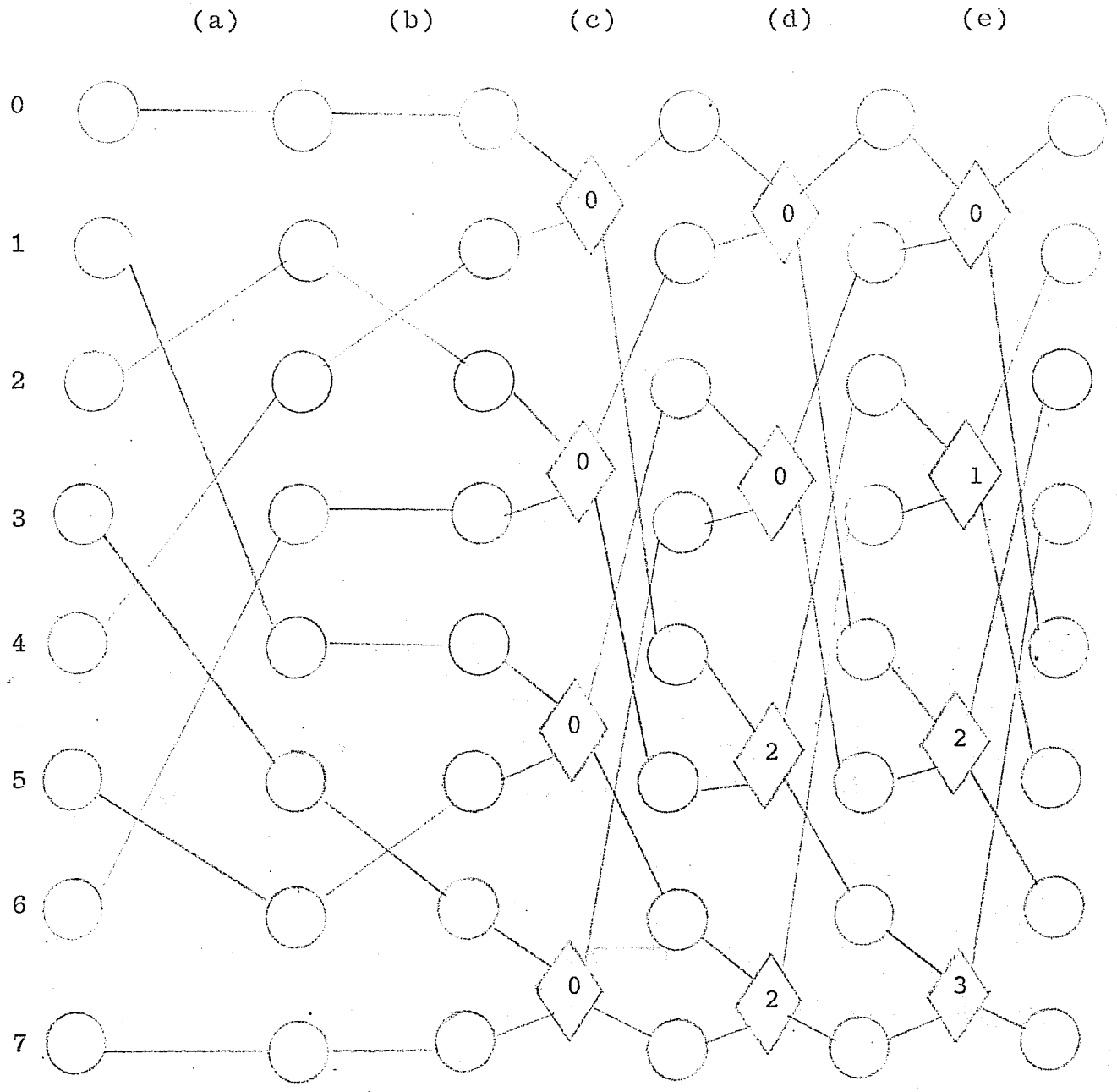
(b) $P_4^2 (D_2^2 \otimes I_2)(W_2 \otimes I_4)$

(c) $P_4^2 (W_2 \otimes I_4)$

(d) $I_2 \otimes P_2^2$

(e) P_4^2

Fig. 3. Pease's FFT for $N=2^3$.



(a) P_2^4

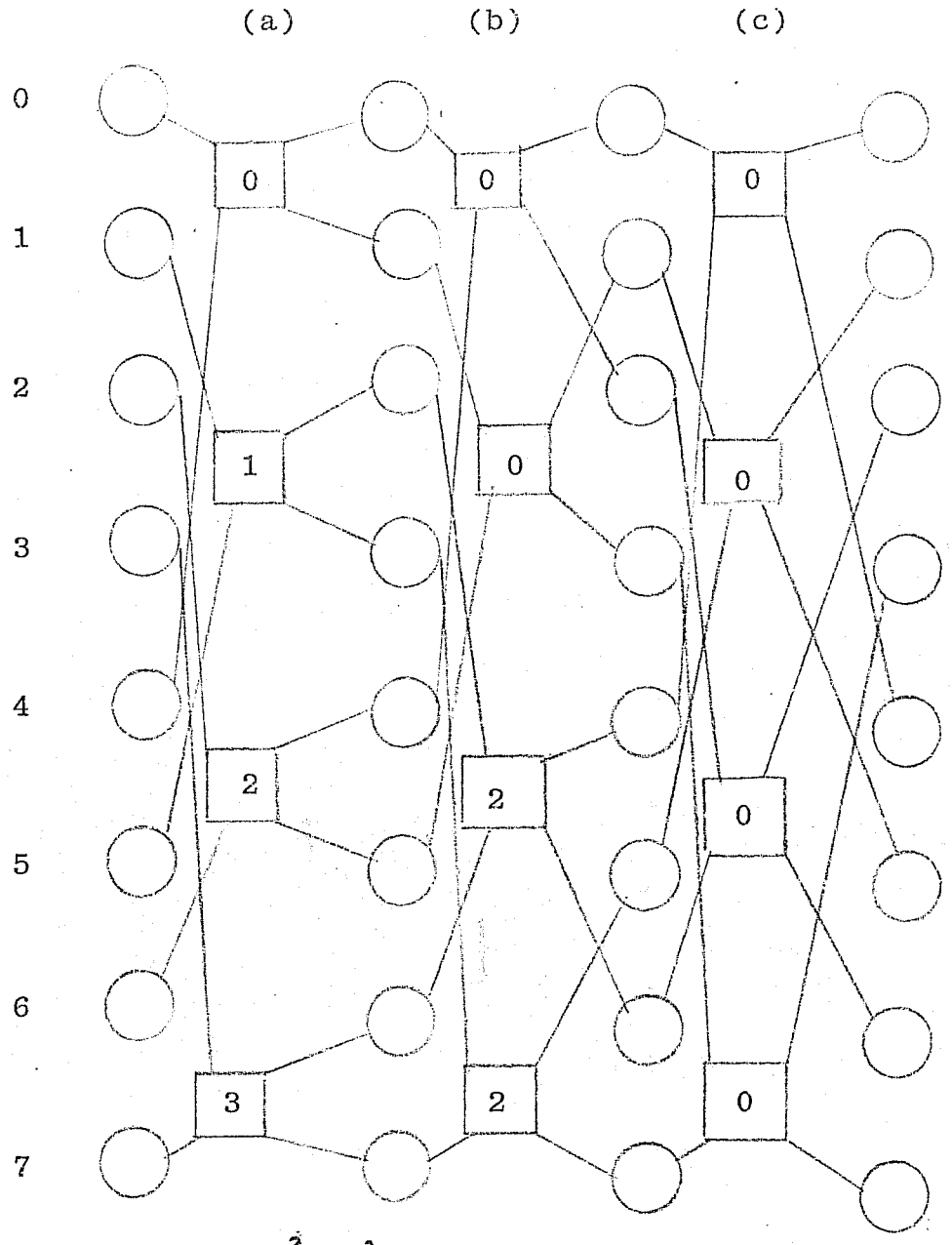
(b) $I_2 \otimes P_2^2$

(c) $(W_2 \otimes I_4) P_2^4$

(d) $(W_2 \otimes I_4)(D_2^2 \otimes I_2) P_4^2$

(e) $(W_2 \otimes I_4) D_4^2 P_2^4$

Fig. 4. The "transposed Pease" FFT for $N=2^3$.

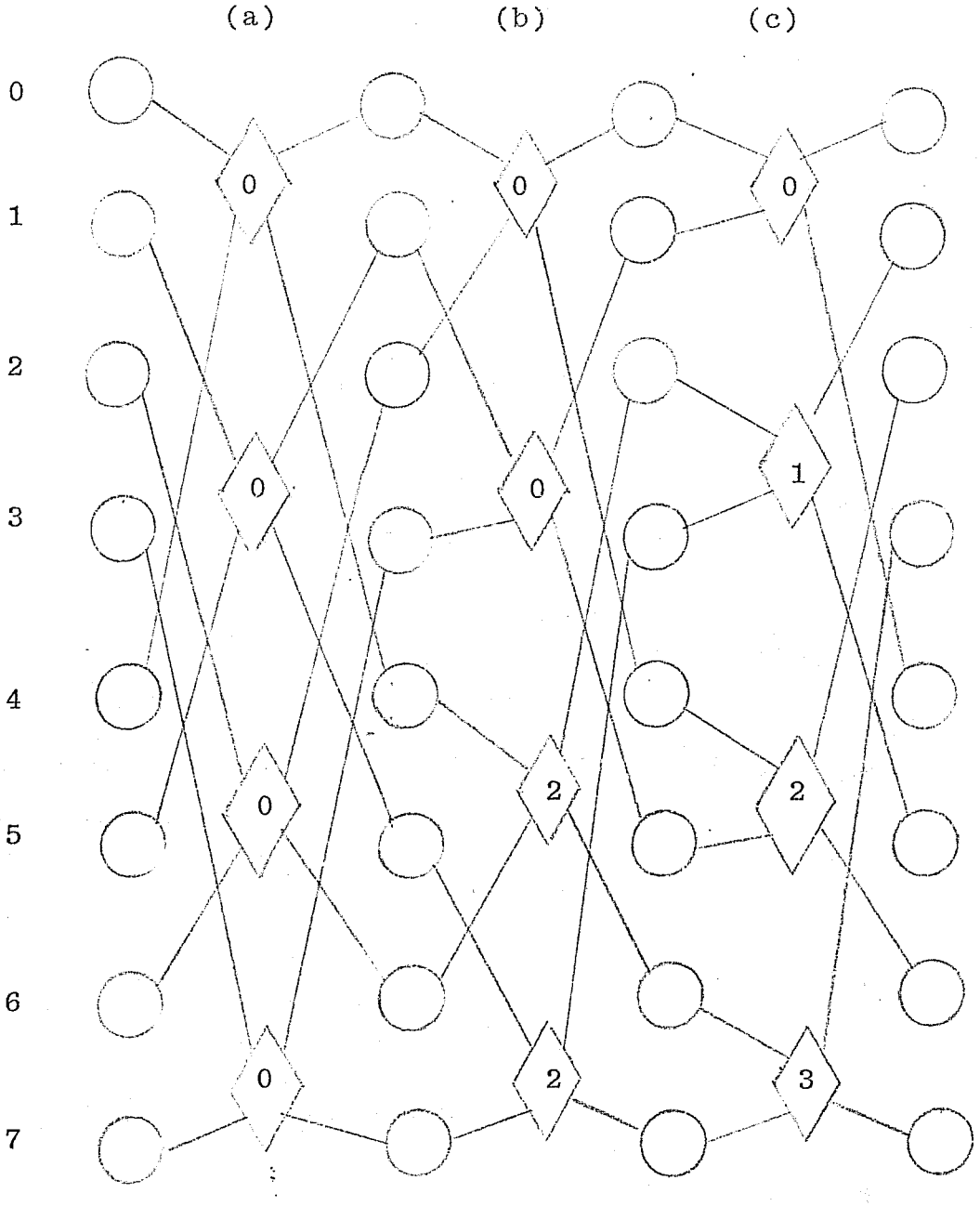


(a) $P_4^2 \, D_4^2 \, (W_2 \otimes I_4)$

(b) $(P_2^2 \otimes I_2)(D_2^2 \otimes I_2)(W_2 \otimes I_4)$

(c) $W_2 \otimes I_4$

Fig. 5. Proposed new form of FFT for $N=2^3$.



(a) $W_2 \otimes I_4$

(b) $(W_2 \otimes I_4)(D_2^2 \otimes I_2)(P_2^2 \otimes I_2)$

(c) $(W_2 \otimes I_4)\, D_4^2\, P_4^2$

Fig. 6. Uhrich's FFT for $N=2^3$.

EUROPEAN CENTRE FOR
MEDIUM RANGE WEATHER
FORECASTS

Research Department (RD)

Technical Report No. 3

No. 1 A Case Study of a Ten Day Prediction

No. 2 The Effect of Arithmetic Precision on some Meteorological Integrations

No. 3 Mixed-Radix Fast Fourier Transforms without reordering