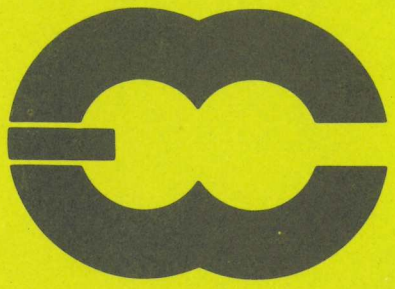


European Centre
for Medium Range
Weather Forecasts

Direct Methods for the Solution of the Discrete Poisson Equation: Some Comparisons

Internal Report 13
Research Dept.

July 77

Centre Européen pour les Prévisions Météorologiques
a Moyen Terme

Europäisches Zentrum Für Mittelfristige Wettervorhersagen

Direct Methods for the Solution of the Discrete Poisson Equation : Some Comparisons.

By

Clive Temperton

European Centre for Medium Range Weather Forecasts, Bracknell

ECMWF
LIBRARY
596

Internal Report No. 13

Research Department

July 1977

N O T E :

This paper has not been published and should be regarded as an Internal Report from ECMWF Research Department.

Permission to quote from it should be obtained from the Deputy Director, Head of Research, at E.C.M.W.F.

ECMWF LIBRARY

1. Introduction

During the past ten years, it has become increasingly popular to solve the discretized Poisson equation (and related problems) by direct rather than iterative methods. Such methods were at first applied only to the simple Poisson equation on a rectangular N x M grid, where N was restricted to the form $N = 2^k$ (or $N = 3 \times 2^k$ in Hockney's (1965) important early paper). More recently, direct methods have been extended to other regular regions (Swarztrauber,1974; Swarztrauber and Sweet, 1973), to arbitrary N (Schumann and Sweet, 1976; Sweet, 1974), to irregular regions (Buzbee et al., 1971; Buzbee and Dorr, 1974; Temperton, 1977b), and to general separable elliptic equations (Swarztrauber, 1974). Some applications of Poisson-solvers to numerical weather prediction have been documented in earlier reports (Temperton 1977a, 1977b).

The algorithms developed for the simple Poisson equation fall into two apparently distinct categories : those based on Fourier decomposition in one dimension, using Fast Fourier Transform (FFT) techniques, and those based on block-cyclic reduction (Buneman's algorithm). Both approaches are documented in detail in the paper by Buzbee et al. (1970). The two approaches can be combined in various ways, and in a forthcoming report (Temperton, 1977c) the relationship between them will be demonstrated. It is natural to ask which of the two approaches, or which combination of them, leads to the fastest algorithms; from the available literature, the answer is not clear. For example, Sweet (1973) found that for a particular problem, Buneman's algorithm was at least twice as fast as a method based on FFT, while Fischer et al.(1974) found FFT methods to be the faster. The reasons for this confusion lie in the variety of methods available for carrying out component parts of the algorithms (principally Fast Fourier Transforms and the solution of tridiagonal systems), and in the assumptions made by different authors (e.g. whether or not any coefficients required in solving tridiagonal systems have been precalculated). Hockney (1977) has also noted the influence of program design, compilers and computer architecture on the relative performance of different algorithms. When comparing operation counts rather than execution times, there are further pitfalls; Hockney (1965) pointed out the importance of counting additions as well as multiplications, and the danger of including only the highest-order terms. Unfortunately, not all subsequent authors have followed this advice.

This paper has as its main aims the following : to establish reasonably accurate operation counts for a number of variants of the direct methods applied to a simple problem; to outline some alternative variants; to offer hints on how certain methods may be implemented most efficiently; to compare execution times; and also to compare the relative accuracy of different methods.

We consider the discretized Poisson equation on a rectangular N x M grid (i,j), $o \le i \le N$, $o \le j \le M$, and for simplicity we assume a unit gridlength in each direction, so that the equation has the simple form:

$$x_{i-1,j} + x_{i+1,j} + x_{i,j-1} + x_{i,j+1} - 4x_{i,j} = b_{i,j} \ldots \qquad (1)$$

where the boundary conditions at i=o,N are either Dirichlet :

$$x_{o,j} = x_{N,j} = o \quad , \qquad o \leq j \leq M,$$

or periodic:

$$x_{o,j} = x_{N,j} \quad , \qquad o \leq j \leq M,$$

and the boundary conditions at j = o,M are Dirichlet:

$$x_{i,o} = x_{i,M} = o \quad , \qquad o \leq i \leq N.$$

We also assume that N is a power of 2, and that FFT's and block-cyclic reductions will normally be performed in the i-direction. Finally, we assume that the same problem has to be solved a number of times with different right-hand sides, and that sufficient core storage is available to precalculate and store any required coefficients; in some cases we will also wish to store both the right-hand side and the solution array separately. The consequences of altering some of these assumptions will also receive mention.

2. Preliminaries

Very careful operation counts for some of the algorithms which follow show that the number of additions or multiplications required is typically of the form

$$(M-1) \left\{ K_1 N \log_2 N + K_2 N + K_3 \log_2 N + K_4 \right\}$$

where, however, the third and fourth terms within the brackets are small compared to the first two, and tend to depend on programming details. We are therefore justified in neglecting them and simply establishing $K_1 \log_2 N + K_2$, the number of additions or multiplications per point.

As mentioned previously, the direct methods outlined here depend heavily on Fast Fourier Transforms and the solution of tridiagonal systems; we first establish operation counts for these procedures.

The experiments reported in Section 7 used a general-purpose Assembler Language FFT package with the following important characteristics:

(a) the number of data points (N) can be any number for a complex transform, and any even number for a real transform;

(b) all trigonometric function values are precalculated and picked out as required during the transform;

(c) the results emerge in the correct order, in contrast to most FFT programs which require somewhat tortuous logic to unscramble the transform (Sweet (1973) found that this feature of most FFT programs significantly increased the execution time of an FFT-based Poisson-solver);

(d) factors of 2 are (as far as possible) grouped together into factors of 4.

The mixed-radix complex FFT was modelled on the algorithm given by Singleton (1969) with appropriate modifications to eliminate reordering (Temperton,1977a) and to pick out trigonometric function values rather than calculating them every time they are required. The additional manipulations required to use the FFT for real transforms are due to Cooley, Lewis and Welch (1970).

For general even N, factorized in the form

$$N = 2^p\, 3^q\, 4^r\, 5^s\, m_1{}^{t_1}\, m_2{}^{t_2} \ldots \qquad (2)$$

where $p \geq 1$ and m_i represents a general odd factor, the operation count for a real periodic transform is

$$N\,\{1.5p + 2.67q + 2.75r + 4s + 0.5 \sum_i (m_i + 5)t_i\} \text{ additions,}$$

$$N\,\{p + 2q + 1.5r + 3.2s + 0.5 \sum_i (m_i + 2)t_i - 1\} \text{ multiplications,}$$

while a real sine transform requires an extra 2.5N additions and 0.5N multiplications.

In the present analysis, we are restricting N to be a power of 2, and for simplicity we assume that the FFT is used in pure radix-2 mode, i.e. r=0 in Eq. (2), so that our operation counts become the following:
for a periodic transform,
$1.5 \log_2 N$ additions and $(\log_2 N - 1)$ multiplications per point;
for a sine transform,
$(1.5 \log_2 N + 2.5)$ additions and $(\log_2 N - 0.5)$ multiplications per point.
Even for the radix-2 FFT, these counts are not quite optimal; Bergland (1968) has indicated a slightly lower operation count for the periodic case by modifying the complex FFT algorithm directly rather than applying the procedures of Cooley, Lewis and Welch (1970) to manipulate the real data into an artificial complex series before (or after) using the complex FFT. Hockney (1970) has similarly achieved a slightly reduced operation count.

We turn our attention now to the solution of tridiagonal systems. In the present problem, these are almost all of the simple form

$$x_{j-1} + \lambda x_j + x_{j+1} = b_j, \quad 1 \leq j \leq M-1 \qquad (3)$$

with $x_0 = x_M = 0$, and $|\lambda| > 2$. We use the following algorithm based on Gaussian elimination and given (for general diagonally dominant tridiagonal systems) by Varga (1962,p.195):

$$w_1 = \lambda^{-1}; \quad w_j = (\lambda - w_{j-1})^{-1}, \quad 2 \leq j \leq M-1;$$

$$g_1 = w_1 b_1; \quad g_j = w_j (b_j - g_{j-1}), \quad 2 \leq j \leq M-1;$$

$$x_{M-1} = g_{M-1}; \quad x_j = g_j - w_j x_{j+1}, \quad M-2 \geq j \geq 1.$$

Provided that the coefficients w_j have been precalculated and stored, this algorithm requires only 2 additions and 2 multiplications per unknown. Other possibilities requiring less coefficient storage but more arithmetic include cyclic reduction (Hockney 1965,1970), methods based on the Toeplitz structure of the tridiagonal matrix (Fischer et al., 1974), and on the factorization of the matrix into two rectangular matrices (Evans, 1972). An interesting way of halving the number of coefficients while requiring no additional arithmetic arises from the "symmetric Gaussian" or "folding" algorithm recently proposed by Evans and Hatzopoulos (1976).

3. The Basic FFT method

For clarity we derive briefly the basic FFT method for Poisson's equation with Dirichlet boundary conditions at $i = 0,N$. On each row, we express the solution values (including the prescribed boundary values) as sums of Fourier sine coefficients:

$$x_{i,j} = \sum_{k=1}^{N-1} \hat{x}_{k,j} \sin (ik\pi/N), \quad 0 \le i \le N, \quad 0 \le j \le M \qquad (4)$$

Substituting in Eq. (1), we obtain (after considerable manipulation) (N-1) tridiagonal systems each of the form:

$$\hat{x}_{k,j-1} + \lambda_k \hat{x}_{k,j} + \hat{x}_{k,j+1} = \hat{b}_{k,j}, \quad 1 \le k \le N-1, \quad 1 \le j \le M-1 \qquad (5)$$

where $\lambda_k = 2 \cos (k\pi/N) - 4$

and

$$\hat{b}_{k,j} = (2/N) \sum_{i=1}^{N-1} b_{i,j} \sin (ik\pi/N) \qquad (6)$$

The solution procedure thus has three stages. In the first stage, a sine transform is applied on each row of the right-hand side field to obtain the coefficients $\hat{b}_{k,j}$. Eq.(5) is then solved for each value of k. In the third stage, an inverse sine transform is applied on each row to implement Eq.(4) and obtain the solution. From Section 2, stages 1 and 3 each require $(1.5\log_2 N + 2.5)$ additions and $(\log_2 N - 0.5)$ multiplications per point, using a radix-2 FFT; while stage 2 requires 2 additions and 2 multiplications per point. The operation count for the whole algorithm is thus $(3\log_2 N + 7)$ additions and $(2\log_2 N + 1)$ multiplications per point.

For periodic boundary conditions at $i = 0,N$ we use the full periodic transform:

$$x_{i,j} = \sum_{k=0}^{N/2} \bar{x}_{k,j} \cos (2ik\pi/N) + \sum_{k=1}^{N/2-1} \hat{x}_{k,j} \sin (2ik\pi/N) \qquad (7)$$

and obtain (N/2+1) systems of the form

$$\bar{x}_{k,j-1} + \lambda_k \bar{x}_{k,j} + \bar{x}_{k,j+1} = \bar{b}_{k,j}, \quad 0 \leq k \leq N/2, \quad 1 \leq j \leq M-1$$

where $\lambda_k = 2 \cos (2k\pi/N) - 4$ (8)

and $\bar{b}_{k,j} = [E(k)/N] \sum_{i=0}^{N-1} b_{i,j} \cos (2ik\pi/N)$

with $E(0) = E(N/2) = 1$, otherwise $E(k) = 2$; together with $(N/2-1)$ systems of the same form as Eq. (5) but with $\lambda_k, 1 \leq k \leq N/2-1$, as in Eq. (8) and

$$\hat{b}_{k,j} = (2/N) \sum_{i=0}^{N-1} b_{i,j} \sin (2ik\pi/N).$$

This time the operation count is $(3\log_2 N+2)$ additions and $(2\log_2 N+1)$ multiplications per point.

The number of coefficients used in solving tridiagonal systems is $(N-1)(M-1)$ or $(N/2+1)(M-1)$ depending on whether the boundary conditions at i=0,N are Dirichlet or periodic. The solution can overwrite the right-hand side field; the only work space required is an area of N locations used by the FFT routine outlined in Section 2.

4. Block-cyclic reduction

Details of the block-cyclic reduction method (Buneman's algorithm) are given by Buzbee et al.(1970), and need not be repeated here. There are two variants; Variant 1 requires two separate arrays, while in Variant 2 the solution can overwrite the original right-hand side field, using no additional work space.

To establish the operation count for Variant 1 with Dirichlet boundary conditions at i=0,N, we first consider the number of tridiagonal systems which have to be solved. There are $(\log_2 N-1)$ steps in the reduction phase; in the rth step we solve $(N/2^r-1)$ systems of the form $A^{(r-1)} x = b$, where $A^{(r-1)}$ is the product of 2^{r-1} tridiagonal matrices (see Buzbee et al.(1970) for details). Solving for $x_{N/2}$ requires N/2 tridiagonal solutions. There are then $(\log_2 N-1)$ steps of the back-solution phase; in the rth step we solve 2^r systems of the form $A^{(s-1)} \underset{\sim}{x} = \underset{\sim}{b}$, where $s = \log_2 N - r$ and $A^{(s-1)}$ is the product of 2^{s-1} tridiagonal matrices. Altogether there are $N(\log_2 N-1)+1$ tridiagonal systems to be solved, each of order (M-1), requiring approximately $(2\log_2 N-2)$ additions and $(2\log_2 N-2)$ multiplications per point.

Secondly we consider the extra additions required to calculate the vectors $\underset{\sim}{p}_i^{(r+1)}$ and $\underset{\sim}{q}_i^{(r+1)}$ at each stage, using the notation of Buzbee et al. (1970) but replacing j by i to indicate that we are performing the block-cyclic reduction in the i-direction. In the first step, $\underset{\sim}{p}_i^{(1)}$ $(i=2,4,\ldots,N-2)$ is given directly by the solution of the tridiagonal system $A\underset{\sim}{p}_i^{(1)} = \underset{\sim}{b}_i$, while $3(M-1)$ additions are required to calculate each corresponding $\underset{\sim}{q}_i^{(1)}$. In the remaining $(\log_2 N-2)$ reduction steps, $\underset{\sim}{p}_i^{(r)}$ and $\underset{\sim}{q}_i^{(r)}$ $(i=2^r, 2^{r+1}, \ldots N-2^r)$ require $3(M-1)$ additions each. The total number of extra additions during the reduction phase is approximately $4.5\,(M-1)N$. During the back-solution phase, the solution of each vector $\underset{\sim}{x}_i$ requires $3(M-1)$ additions, except during the final step when, for all odd i, $\underset{\sim}{x}_i$ requires only $2(M-1)$ additions. The total of extra additions during the back-solution phase is thus $2.5(M-1)N$, and the operation count for the whole algorithm is $(2\log_2 N+5)$ additions and $(2\log_2 N-2)$ multiplications per point. This is less than the corresponding operation count for the basic FFT algorithm given in Section 3.

Turning now to Variant 2, the number of tridiagonal systems to be solved is exactly the same. The vectors $\underset{\sim}{p}_i^{(r+1)}$ are eliminated; the equation given by Buzbee et al.(1970) for $\underset{\sim}{q}_i^{(r+1)}$ appears to involve $12(M-1)$extra additions, but this can be reduced to $8(M-1)$ by defining

$$\bar{\underset{\sim}{q}}_i^{(r)} = \underset{\sim}{q}_{i-2h}^{(r)} - \underset{\sim}{q}_{i-h}^{(r-1)} + \underset{\sim}{q}_i^{(r)} - \underset{\sim}{q}_{i+h}^{(r-1)} + \underset{\sim}{q}_{i+2h}^{(r)}$$

$$\underset{\sim}{q}_i^{(r+1)} = \bar{\underset{\sim}{q}}_i^{(r)} + (A^{(r)})^{-1}\left[\underset{\sim}{q}_{j-3h}^{(r-1)} + \underset{\sim}{q}_{j+3h}^{(r-1)} - \bar{\underset{\sim}{q}}_i^{(r)} - \underset{\sim}{q}_i^{(r)}\right]$$

where

$h = 2^{r-1}$, and in fact this reduces to $3(M-1)$ in the first step since we have simply

$$\underset{\sim}{q}_i^{(1)} = \underset{\sim}{b}_{i+1} + \underset{\sim}{b}_{i-1} - 2A^{-1}\underset{\sim}{b}_i \,.$$

The total of extra additions in the reduction phase is thus approximately $5.5\,(M-1)\,N$. Back-solution for $\underset{\sim}{x}_i$ (i even) requires $5(M-1)$ extra additions and $(M-1)$ extra multiplications- for i odd there are only $2(M-1)$ extra additions (and no multiplications), a total of $3.5\,(M-1)\,N$ extra additions and $0.5\,(M-1)N$ extra multiplications during back-solution. The operation count for the whole algorithm is thus $(2\log_2 N+7)$ additions and $(2\log_2 N-1.5)$ multiplications per point, an increase over Variant 1 of only 2 additions and 0.5 multiplications per point. Thus the estimate of Buzbee et al.(1970) that Variant 2 requires approximately twice as many additions is seen to be excessively pessimistic.

For periodic boundary conditions at i=0,N the reduction and back-solution phases each have $\log_2 N$ steps; again, the details are given by Buzbee et al.(1970). In this case the operation count for Variant 1 is $(2\log_2 N+9)$ additions and $(2\log_2 N+2)$ multiplications per point. Comparing this with the counts for the corresponding FFT method in Section 3, the number of multiplications is always larger, while for $N < 256$ the number of additions is also larger. Variant 2 again requires an extra 2 additions and 0.5 multiplications per point.

The number of coefficients used in solving tridiagonal systems is (N-1)(M-1) or (N+1)(M-1), depending on whether the boundary conditions at i=0,N are Dirichlet or periodic. When using Variant 1, the vectors $p_i^{(r)}$ and $q_i^{(r)}$ $r \geq 1$, can share an array of dimension (M-1)N which becomes the solution array, so that the original right-hand side field is not destroyed. Alternatively, if the vectors $q_i^{(r)}$ overwrite the right-hand side, then the auxiliary array containing the vectors $p_i^{(r)}$ need only be half the size of the main array, so that Variant 1 only requires 50% more array space than Variant 2.

5. FACR (1) methods

(a) Hockney's FACR (1,j) algorithm.

Hockney (1965) pointed out that half the transforms in the basic FFT method could be eliminated by first performing one stage of cyclic reduction in the j-direction, i.e. by eliminating all $x_{i,j}$ with j odd. The resulting equation is

$$x_{i,j-2} - x_{i-2,j} + 8x_{i-1,j} - 16x_{i,j} + 8x_{i+1,j} - x_{i+2,j} + x_{i,j+2}$$

$$= b^{(1)}_{i,j} = b_{i,j-1} - b_{i-1,j} + 4b_{i,j} - b_{i+1,j} + b_{i,j+1} \quad \ldots \qquad (9).$$

Again we can write $x_{i,j}$ as in Eq. (4) (this time for even j only) and we obtain (for Dirichlet boundary conditions at i = 0,N) (N-1) tridiagonal systems each of the form:

$$\hat{x}_{k,j-2} + \lambda_k \hat{x}_{k,j} + \hat{x}_{k,j+2} = \hat{b}^{(1)}_{k,j}, \quad 1 \leq k \leq N-1, \quad j=2,4,\ldots,M-2 \qquad (10)$$

where $\lambda_k = 2 - 4\,(\cos(k\pi/N) - 2)^2$

and

$$\hat{b}^{(1)}_{k,j} = (2/N) \sum_{i=1}^{N-1} b^{(1)}_{i,j} \sin(ik\pi/N) \qquad (11)$$

Thus we first obtain $b^{(1)}_{i,j}$ for even j using Eq.(9); this requires 2 additions and 0.5 multiplications per point (of the whole grid). Next we perform the sine transform using Eq. (11) for even j, requiring $(0.75\log_2 N+1.25)$ additions and $(0.5\log_2 N-0.25)$ multiplications per point. The tridiagonal systems (Eq. 10) require 1 addition and 1 multiplication per point; the inverse sine transforms (Eq.(4)) for even j require $(0.75\log_2 N+1.25)$ additions and $(0.5\log_2 N-0.25)$ multiplications per point.

Finally the solution $x_{i,j}$ for odd j is obtained by solving tridiagonal systems along rows, taking 2 additions and 1 multiplication per point. The total operation count for the whole algorithm is $(1.5\log_2 N+7.5)$ additions and $(\log_2 N+2)$ multiplications per point.

Comparing with previous operation counts for Dirichlet boundary conditions, we see that Hockney's algorithm (using Gaussian elimination for the tridiagonal systems) is always faster than the basic FFT method; the number of additions is less than in Buneman's algorithm for $N > 32$, and the number of multiplications is less for $N > 16$.

For periodic boundary onditions at i=0,N a completely analogous algorithm can be derived; the only difficulty is that a cyclic tridiagonal system of order N has to be solved for each odd-numbered line. A number of techniques are available, some of which were discussed by Temperton (1975); Algorithm 4 of that paper requires the least computation, namely 3N additions and 2.5N multiplications, but as N is here assumed to be a power of 2, cyclic reduction is almost as efficient, and requires little or no coefficient storage.

Using Gaussian elimination for tridiagonal systems and cyclic reduction for cyclic tridiagonal systems, Hockney's algorithm for periodic boundary conditions at i=0,N takes $(1.5\ \log_2 N+6)$ additions and $(\log_2 N+1.5)$ multiplications per point, the smallest operation count so far obtained.

For Hockney's method, the number of coefficients required for the tridiagonal solutions is $(M/2-1)(N-1)$ or $(M/2-1)(N/2+1)$, depending on whether the boundary conditions at i=0,N are Dirichlet or periodic. As in the case of the basic FFT method, the solution can overwrite the right-hand side, and the only work space required is for FFT's.

(b) FACR(1,i)

In comparison with the basic FFT method, Hockney's algorithm halves the number of Fourier transforms required by forming a set of equations involving $x_{i,j}$ for even j only. An alternative strategy is to halve the length of the Fourier transforms by forming an analogous set involving $x_{i,j}$ for even i only. The resulting equations are of the form:

$$x_{i-2,j} - x_{i,j-2} + 8x_{i,j-1} - 16x_{i,j} + 8x_{i,j+1} - x_{i,j+2} + x_{i+2,j}$$

$$= b^{(1)}_{i,j} = b_{i-1,j} - b_{i,j-1} + 4b_{i,j} - b_{i,j+1} + b_{i+1,j} \quad \ldots(14)$$

On each line we now have $(N/2+1)$ values of $x_{i,j}$ (including the boundary points), and these can be expressed as sums of $(N/2-1)$ sine coefficients:

$$x_{2i,j} = \sum_{k=1}^{N/2-1} \hat{x}_{k,j} \sin(2ik\pi/N), \quad 0 \le i \le N/2, \quad 0 \le j \le M \qquad (15)$$

Substituting in Eq.(14), we obtain N/2-1 pentadiagonal systems, each of which can be factorized into the form

$$\left.\begin{aligned} \hat{y}_{k,j-1} + \lambda_{N-k}\,\hat{y}_{k,j} + \hat{y}_{k,j+1} &= \hat{b}^{(1)}_{k,j} \\ \hat{x}_{k,j-1} + \lambda_k \hat{x}_{k,j} + \hat{x}_{k,j+1} &= \hat{y}_{k,j} \end{aligned}\right\} \qquad (16)$$

where $\lambda_k = 2\cos(k\pi/N) - 4$, and

$$\hat{b}^{(1)}_{k,j} = (4/N) \sum_{i=1}^{N/2-1} b^{(1)}_{2i,j} \sin(2ik\pi/N) \qquad (17)$$

The algorithm thus proceeds as follows: the right-hand side of Eq.(14) is calculated at all gridpoints with i even, taking 2 additions and 0.5 multiplications per point. Eq. (17) is then implemented; this involves (M-1) sine transforms each of length N/2, and takes ($0.75\log_2 N + 0.5$) additions and ($0.5\log_2 N - 0.75$) multiplications per point. The solution of (N/2-1) pentadiagonal systems takes 2 additions and 2 multiplications per point. (M-1) inverse sine transforms each of length N/2 are then performed to obtain $x_{2i,j}$, requiring ($0.75\log_2 N + 0.5$) additions and ($0.5\log_2 N - 0.75$) multiplications per point. Finally, $x_{i,j}$ for j odd is obtained by solving simple tridiagonal systems in the j-direction which (including setting up the right-hand sides) takes 2 additions and 1 multiplication per point. The operation count for the whole algorithm is ($1.5\log_2 N + 7$) additions and ($\log_2 N + 2$) multiplications, a saving of just half an addition per point over Hockney's algorithm. For N=16, the FFT method with one preliminary level of cyclic reduction in i takes the same number of additions and multiplications as Buneman's algorithm; for N >16, the number is less.

For periodic boundary conditions at i=0,N a similar algorithm can be derived, with an operation count of ($1.5\log_2 N + 4.5$) additions and ($\log_2 N + 0.5$) multiplications per point. The improvement over Hockney's algorithm is 1.5 additions and 1 multiplication per point, the gain being greater in the periodic case since the final step involves strictly tridiagonal systems rather than cyclic tridiagonal systems as in Hockney's algorithm.

The number of coefficients required for the tridiagonal systems is the same as for the basic FFT method. Again, the solution can overwrite the right-hand side, and the only work space required is for FFT's.

(c) FACR(1,i+j) (diagonal cyclic reduction)

Yet another way of combining the FFT method with one step of cyclic reduction is included here, not because it leads to a more efficient algorithm but because it has some interesting aspects, and because it has applications in the direct solution of Poisson's equation over irregular regions (Temperton, 1977b). In Sections 5(a) and 5(b) we halved the number of unknowns by eliminating $x_{i,j}$ either for i odd or for j odd. A third alternative is to eliminate $x_{i,j}$ for (i+j) odd, resulting in the following equation:

$$x_{i,j+2} + x_{i,j-2} + x_{i-2,j} + x_{i+2,j} + 2(x_{i-1,j+1} + x_{i-1,j-1} + x_{i+1,j+1} + x_{i+1,j-1}) - 12x_{i,j} = b^{(1)}_{i,j}$$

$$= 4\, b_{i,j} + b_{i,j+1} + b_{i,j-1} + b_{i+1,j} + b_{i-1,j} \qquad (18)$$

with appropriate modifications near the boundaries. The retained points lie on alternate diagonals, as shown in Fig. 1, which also indicates the nine-point operator represented by the left-hand side of Eq. (18). For even j, we introduce the same sine summation as for FACR (1,i):

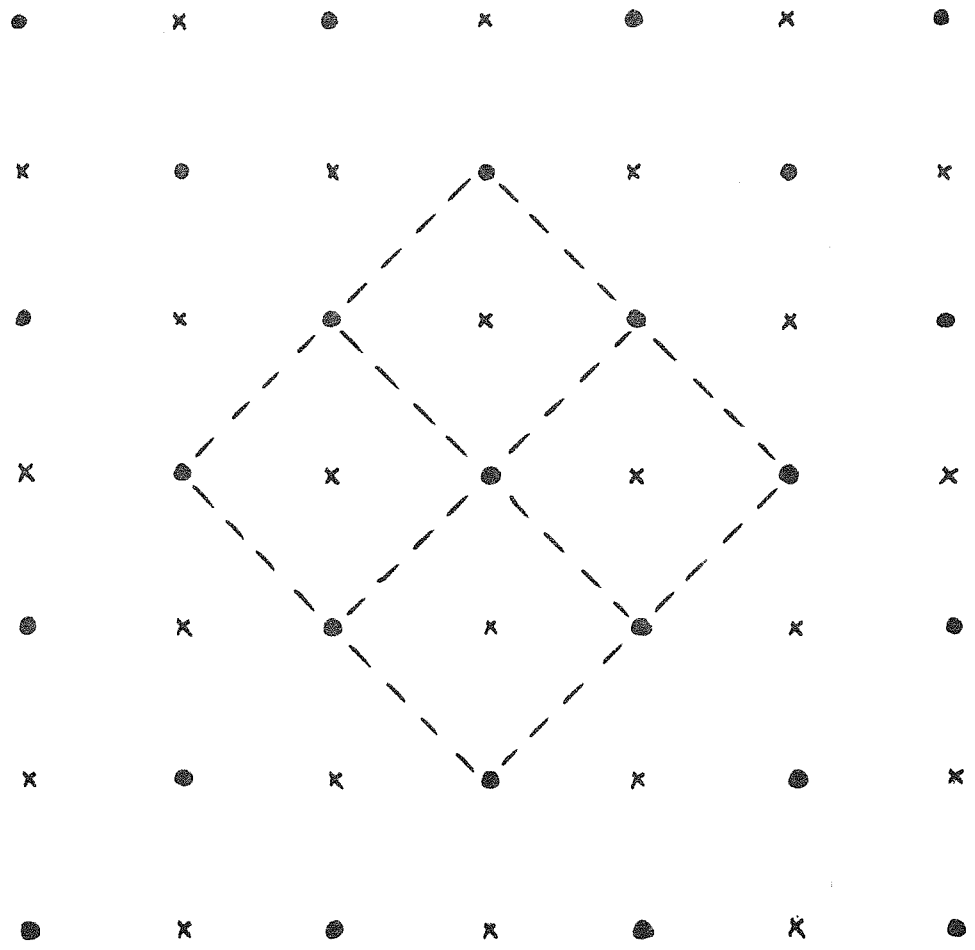
$$x_{2i,j} = \sum_{k=1}^{N/2-1} \hat{x}_{k,j} \sin(2ik\pi/N), \quad 0 \le i \le N/2, \quad j \text{ even} \qquad (19)$$

while for odd j, we introduce the following modified summation:

$$x_{2i-1,j} = \sum_{k=1}^{N/2} \hat{x}_{k,j} \sin((2i-1)k\pi/N), \quad 0 \le i \le N/2-1, \quad j \text{ odd} \qquad (20)$$

Notice that this series has an extra term (k=N/2). Introducing (19) and (20) into Eq. (18) and performing the usual manipulations, we obtain (N/2-1) pentadiagonal systems of order (M-1), which factorize into the form:

$$\left.\begin{aligned} \hat{y}_{k,j-1} + \lambda'_k \hat{y}_{k,j} + \hat{y}_{k,j+1} &= \hat{b}^{(1)}_{k,j} \\ \hat{x}_{k,j-1} + \lambda'_k \hat{x}_{k,j} + \hat{x}_{k,j+1} &= \hat{y}_{k,j} \end{aligned}\right\} \qquad (21)$$



● retained points

× eliminated points

FIGURE 1 : "Diagonal" cyclic reduction

where $\lambda'_k = 2\cos(k\pi/2N) - 4$, $\lambda''_k = 2\cos(k\pi/2N) + 4$,

$$\hat{b}^{(1)}_{k,j} = (4/N)\sum_{i=1}^{N/2-1} b^{(1)}_{2i,j} \sin(2ik\pi/N) \quad \ldots \qquad (22)$$

for j even, and

$$\hat{b}^{(1)}_{k,j} = (4/N)\sum_{i=1}^{N/2} b^{(1)}_{2i-1,j} \sin((2i-1)k\pi/N) \ldots \qquad (23)$$

for j odd.

In addition, for k=N/2 we obtain a tridiagonal system of order M/2 of a rather special form:

$$\left.\begin{aligned} -15\hat{x}_{k,1} + \hat{x}_{k,3} &= \hat{b}^{(1)}_{k,1} \\ \hat{x}_{k,j-2} - 14\hat{x}_{k,j} + \hat{x}_{k,j+2} &= \hat{b}^{(1)}_{k,j} \quad ,j=3,5,\ldots,M-3 \\ \hat{x}_{k,M-3} - 15\hat{x}_{k,M-1} &= \hat{b}^{(1)}_{k,M-1} \end{aligned}\right\} \qquad (24)$$

where

$$\hat{b}^{(1)}_{k,j} = -(4/N)\sum_{i=1}^{N/2} (-1)^i b^{(1)}_{2i-1,j} \qquad (25)$$

The algorithm is thus implemented as follows: First, $b^{(1)}_{i,j}$ is determined, using Eq. (18), at all points with (i+j) even. The Fourier sine coefficients $\hat{b}^{(1)}_{k,j}$ are then computed using a simple sine transform Eq.(22), on even lines and a shifted sine transform, Eqs. (23) and (25), on odd lines. The pentadiagonal systems (21) and the tridiagonal system (24) are solved for the sine coefficients $\hat{x}_{k,j}$. The solution at points with (i+j) even is then obtained using a simple inverse sine transform (19) on even lines and the inverse shifted sine transform (20) on odd lines.

Finally the solution at points with (i+j) odd is determined from the scalar equation

$$x_{i,j} = \frac{1}{4}(x_{i+1,j} + x_{i-1,j} + x_{i,j+1} + x_{i,j-1} - b_{i,j})$$

since all the quantities on the right-hand side are by now known.

The manipulations required to convert a real shifted sine transform into a real periodic transform are given by Swarztrauber (1977).

The operation count, including the manipulations for the shifted transforms, is $(1.5\log_2 N + 7.5)$ additions and $(\log_2 N + 2.25)$ multiplications per point, very similar to the counts for Hockney's FACR (1,j) algorithm and for FACR (1,i). The algorithm can be modified to eliminate $x_{i,j}$ for (i+j) even; the roles of odd and even lines are then interchanged.

An analogous algorithm can also be derived for periodic boundary conditions at i=0,N, with an operation count of $(1.5 \log_2 N+5)$ additions and $(\log_2 N+2)$ multiplications per point.

The process of diagonal cyclic reduction is similar to the first step of the "total reduction" method of Schröder and Trottenberg (1973). It should be possible to continue the process, eliminating half the diagonals at each stage, and thus to generate a "diagonal" variant of block-cyclic reduction. It is interesting that in this application of block-cyclic reduction, the blocks are of varying size.

6. Summary of operation counts

In Table 1 we summarize the operation counts for all the algorithms outlined in Sections 3 - 5. We remarked in Section 4, that for Dirichlet boundary conditions at i=0,N Buneman's algorithm appeared to be faster than the basic FFT method, while for periodic boundary conditions at i=0,N the reverse was true, except for very large values of N. (Note here the misleading effect of comparing "asymptotic" operation counts). Combining the FFT method with one step of cyclic reduction to halve the length of the transforms, we obtain a scheme which requires fewer operations than Buneman's algorithm for $N > 16$ under Dirichlet boundary conditions at i=0,N, and for all N under periodic boundary conditions at i=0,N.

It is worth reiterating at this point that we have been solving all the tridiagonal systems by Gaussian elimination, using precalculated coefficients. If this is ruled out by lack of space, cyclic reduction may be used instead. To solve a tridiagonal system of order M by cyclic reduction requires approximately 4M additions and 2M multiplications, an increase over Gaussian elimination of 2M additions. Now the number of tridiagonal systems to be solved in Buneman's algorithm is approximately $N(\log_2 N-1)$ for Dirichlet boundary conditions at i=0,N, and $N(\log_2 N+1)$ for periodic boundary conditions at i=0,N; while in the FFT-based algorithms the number is approximately N, or at most 3N/2. The consequences of changing from Gaussian elimination to cyclic reduction for simple tridiagonal systems are thus as follows: for either Variant of Buneman's algorithm, an extra $(2\log_2 N-2)$ (Dirichlet) or $(2\log_2 N+2)$ (periodic) additions per point; for basic FFT or FACR $(1,i\pm j)$, 2 extra additions per point; for Hockney's FACR (1,j) algorithm, an extra 2 (Dirichlet) or 1 (periodic) additions per point; for FACR (1,i), 3 extra additions per point. In this situation, the use of FFT-based methods appears even more advantageous.

TABLE 1

Summary of operation counts

B.c.'s at i=0,N	Dirichlet		periodic	
	adds	mults	adds	mults
Buneman (Variant 1)	$2\log_2 N+5$	$2\log_2 N-2$	$2\log_2 N+9$	$2\log_2 N+2$
Buneman (Variant 2)	$2\log_2 N+7$	$2\log_2 N-1.5$	$2\log_2 N+11$	$2\log_2 N+2.5$
Basic FFT	$3\log_2 N+7$	$2\log_2 N+1$	$3\log_2 N+2$	$2\log_2 N+1$
Hockney's FACR (1,j)	$1.5\log_2 N+7.5$	$\log_2 N+2$	$1.5\log_2 N+6$	$\log_2 N+1.5$
FACR(1,i)	$1.5\log_2 N+7$	$\log_2 N+2$	$1.5\log_2 N+4.5$	$\log_2 N+0.5$
FACR(1,i+j)	$1.5\log_2 N+7.5$	$\log_2 N+2.25$	$1.5\log_2 N+5$	$\log_2 N+2$

7. Numerical results

In the preceding sections, we have considered a number of algorithms in terms of operation counts; here we turn to their actual implementation on a computer. Now although it is clear that direct methods for the solution of Poisson's equation are much more efficient than simple iterative methods such as successive overrelaxation, they are considerably harder to program, and for their superiority to be fully realised it is important that they be efficiently coded. The author's personal preference is for the use of a low-level language, and the timings reported here relate to programs written in IBM Assembler Language. This preference was reinforced by the following experiment.

Two Fortran subroutines were written to implement Buneman's algorithm (Variant 1) for Poisson's equation on a rectangle with Dirichlet boundary conditions. One treated all the arrays as two-dimensional (i.e., doubly-subscripted), while the second treated them as one-dimensional (singly-subscripted). The two subroutines were then each compiled at levels G,H and H+ (the latter including special optimizing features for the 360/195 on which the programs were run). The Fortran style was intended to be as helpful as possible to the compiler, and as efficient as possible at run time (e.g., no branches to subroutines, or from one section of the program to another,apart from simple loops).

TABLE 2

Execution times (in msec) for 32x32 problem
by Buneman's
algorithm (Variant 1)

Language	Compiler	2-d indexing	1-d indexing
Fortran	G	58.5	39.6
Fortran	H	14.9	10.5
Fortran	H+	14.2	9.4
Assembler			5.33

The six resulting programs, together with a corresponding Assembler Language subroutine, were then timed on the Dirichlet problem with N=M=32. The results are given in Table 2. The conclusions are clear; even the best Fortran subroutine took 75% longer than the Assembler version. If a high-level language must be employed, the indexing should be one-dimensional and the best possible compiler should be used.

Assembler Language versions of four algorithms described in Sections 3,4 and 5 (the basic FFT algorithm, Buneman's algorithm Variant 1, Hockney's FACR (1,j) algorithm and the FACR (1,i) algorithm) were timed on N x N problems, with N ranging from 8 to 128, and with both Dirichlet and periodic boundary conditions at i=0,N. The FFT package (Section 2) was used in its 'radix 4+2' mode, thus slightly improving on the operation counts given in Section 6. Also, the cyclic tridiagonal systems arising in Hockney's algorithm with periodic boundary conditions at i=0,N were solved using Algorithm 4 of Temperton (1975). The results are shown in Table 3 (Dirichlet boundary conditions at i=0,N) and Table 4 (periodic boundary conditions at i=0,N).

Several points are worthy of note. For Dirichlet boundary conditions, the fastest algorithm was Buneman's for $N \leq 32$, and Hockney's for $N \geq 64$. For periodic boundary conditions, the fastest was Hockney's throughout the range. These results are in line with the comparisons given in Section 6, though for Dirichlet boundary conditions the value of N at which Hockney's algorithm becomes faster than Buneman's is somewhat larger than predicted, evidently because of the extra addressing arithmetic required by the FFT. FACR (1,i) is disappointing in comparison with Hockney's algorithm, especially for small values of N; again the reason lies in the extra addressing arithmetic for the FFT. In terms of total execution time it is clearly faster to do M/2 transforms of length N rather than M transforms of length N/2, though the latter has a slightly lower floating point operation count. This suggests that all the FFT-based algorithms could be made more efficient by performing the transforms in parallel rather than one at a time. Finally, notice that, in agreement with the timings reported by Hockney (1970), the execution time for each algorithm is roughly proportional to N^2, the total number of points.

TABLE 3

Execution times (sec) for Dirichlet b.c.'s at i=0,N

Method	N = 8	N = 16	N = 32	N = 64	N = 128
Buneman	2.33×10^{-4}	1.12×10^{-3}	5.33×10^{-3}	2.54×10^{-2}	1.58×10^{-1}
Basic FFT	5.60×10^{-4}	2.24×10^{-3}	8.81×10^{-3}	3.94×10^{-2}	1.64×10^{-1}
FACR (1,j)	3.55×10^{-4}	1.42×10^{-3}	5.61×10^{-3}	2.45×10^{-2}	1.02×10^{-1}
FACR (1,i)	5.33×10^{-4}	1.69×10^{-3}	6.57×10^{-3}	2.61×10^{-2}	1.14×10^{-1}

TABLE 4

Execution times (sec) for periodic b.c.'s at i=0,N

Method	N = 8	N = 16	N = 32	N = 64	N = 128
Buneman	3.89×10^{-4}	2.28×10^{-3}	1.10×10^{-2}	5.24×10^{-2}	4.44×10^{-1}
Basic FFT	4.79×10^{-4}	2.10×10^{-3}	7.68×10^{-3}	3.35×10^{-2}	1.55×10^{-1}
FACR(1,j)	3.26×10^{-4}	1.40×10^{-3}	5.31×10^{-3}	2.23×10^{-2}	1.01×10^{-1}
FACR(1,i)	5.07×10^{-4}	1.70×10^{-3}	7.21×10^{-3}	2.71×10^{-2}	1.36×10^{-1}

We consider now the question of the accuracy of the various algorithms. For each value of N and each set of boundary conditions, a random number generator was used to set up ten NxN "true" solutions with values in the interval $[-1,+1]$, from which corresponding right-hand sides were computed. Each algorithm was then used to recover the solution from the right-hand side, and the computed solution was compared with the original field to determine the maximum absolute point error. (Note that the computation of the right-hand side is itself a source of round-off error). The maximum errors, meaned over ten solutions in each case, are shown in Tables 5 and 6.

TABLE 5

Mean maximum errors for Dirichlet b.c.'s at i=0,N

Method	N = 8	N = 16	N = 32	N = 64	N = 128
Buneman	2.56×10^{-6}	4.95×10^{-6}	2.98×10^{-5}	1.28×10^{-4}	6.05×10^{-4}
Basic FFT	3.60×10^{-6}	1.10×10^{-5}	4.18×10^{-5}	3.10×10^{-4}	9.65×10^{-4}
FACR(1,j)	4.04×10^{-6}	2.55×10^{-5}	8.96×10^{-5}	4.90×10^{-4}	1.57×10^{-3}
FACR(1,i)	5.82×10^{-6}	1.43×10^{-5}	5.38×10^{-5}	2.23×10^{-4}	8.03×10^{-4}

TABLE 6

Mean maximum errors for periodic b.c.'s at i=0,N

Method	N = 8	N = 16	N = 32	N = 64	N = 128
Buneman	3.49×10^{-6}	7.33×10^{-6}	4.59×10^{-5}	2.30×10^{-4}	9.95×10^{-4}
Basic FFT	4.42×10^{-6}	1.61×10^{-5}	7.05×10^{-5}	2.83×10^{-4}	1.10×10^{-3}
FACR(1,j)	5.00×10^{-6}	1.90×10^{-5}	7.27×10^{-5}	2.98×10^{-4}	1.14×10^{-3}
FACR(1,i)	4.76×10^{-6}	1.88×10^{-5}	7.34×10^{-5}	2.99×10^{-4}	1.09×10^{-3}

Buneman's algorithm is the most accurate, while Hockney's is generally the least accurate, though in practical terms the difference between them is unlikely to be important. The variant of Hockney's algorithm described by Temperton(1977c), in which the preliminary cyclic reduction step is "stabilized", yields smaller errors than the form used here.

For the Dirichlet problem, the algorithm of Cooley, Lewis and Welch (1976) was used to convert a real sine transform of length N to a real periodic transform of length N, and hence to a complex periodic transform of length N/2. Since the sine transform is its own inverse (apart from a scaling factor), one can alternatively "invert" their algorithm, and since these experiments were run it has been discovered that the inverted procedure gives smaller round-off errors. See Temperton (1977c) for some comparisons obtained on CDC 6600. Nevertheless, the round-off error for the basic FFT method remained greater than that for Buneman's algorithm.

Tables 5 and 6 also show that the errors are roughly proportional to N^2. For Buneman's algorithm at least, this behaviour appears to be computer-dependent; compare with the results reported by Temperton (1977c) on a CDC 6600. The results of Tables 5 and 6 also show that for large grids (say $N > 128$), single-precision arithmetic on an IBM machine may give insufficiently accurate results.

8. Extensions

In Section 6, we noted the consequences of limiting the available core storage so that tridiagonal systems could no longer be solved by Gaussian elimination using precomputed coefficients; it was shown that under these circumstances the advantages of FFT-based methods over block-cyclic reduction became more pronounced. We now consider the effects of relaxing some of the constraints on the problem itself which were laid down in Section 1.

If the gridlengths in the i and j directions are unequal, the algorithms require only slight modification; the number of extra multiplications per point ranges from zero to two, depending on the algorithm used. It is generally more efficient to scale the problem to give unit gridlength in the j-direction, so that the tridiagonal systems to be solved in that direction retain the form of Eq. (3), with 1's on the sub- and superdiagonals of the corresponding tridiagonal matrices.

The operation counts for Neumann boundary conditions at i=0,N or at j=0,M are the same as for Dirichlet boundary conditions. Periodic boundary conditions at j=0,M require the solution of periodic tridiagonal systems, and again it is the FFT-based algorithms, with fewer such systems to solve,which require less extra computation.

The most general form of elliptic equation which can readily be handled by the techniques developed in this paper is

$$\nabla^2\phi + \beta_j \delta_y \phi - \kappa_j \phi = r_{i,j} \qquad (26)$$

where ∇^2, δ_y are the finite-difference analogues of $\nabla^2, \partial/\partial y$, and β κ, are functions of j only. Hockney's algorithm and FACR(1,i+j) become less straightforward, but for the remaining algorithms the only changes are to the tridiagonal systems in the j-direction. These now have the more general form.

$$\mu_j x_{j-1} + \lambda_j x_j + \nu_j x_{j+1} = b_j. \quad 1 \leq j \leq M-1$$

with $x_0 = x_M = 0$. It is possible (though not very easy) to solve such a system by cyclic reduction (Heller,1976); more suitable is Gaussian elimination using precomputed coefficients. This requires 2 additions, 3 multiplications and 2 precomputed coefficients per unknown. With the increased operation count for the solution of each tridiagonal system, it is again the FFT-based methods, involving only about N such systems, which score over the block-cyclic reduction method. For the basic FFT method, the change from the simple Poisson Eq.(1) to the more general Eq.(26) increases the operation count by only one multiplication per point, though it doubles the number of precomputed coefficients required for Gaussian elimination.

Swarztrauber (1974) has shown that block-cyclic reduction can be extended to equations even more general than Eq.(26), though the resulting algorithm is very complicated. Some extension is also possible for the FFT methods; see for example Le Bail (1972).

Finally in this section we relax the restriction that N be a power of 2. Sweet (1974) has extended Buneman's algorithm to the case $N=2^p3^q5^r\ldots$; the operation count rises rather rapidly as larger factors of N are included. A more flexible, simpler and more efficient approach is to use FFT methods with a mixed-radix FFT (see Section 2). For particularly awkward values of N, e.g. large prime numbers, Temperton (1977b) has suggested an FFT-based algorithm in which the rectangle is embedded in a larger one with a more convenient value of N, using a capacity-matrix technique.

However, for general N the most efficient method may not after all be FFT-based. Schumann and Sweet (1976) have recently proposed a new block-cyclic reduction algorithm which has an operation count approximately proportional to $MN\log_2 N$ for arbitrary N. Although their algorithm is for a grid staggered with respect to the boundaries, a similar algorithm can be derived for the grid specified in this paper.

9. Conclusions

Some of the most important conclusions of this study may be summarized as follows:

(1) In calculating operation counts for direct methods, it is important to include both additions and multiplications, and to include terms of order MN as well as those of order $MN\log_2 N$.

(2) Under the assumptions of Section 1, Buneman's algorithm (block-cyclic reduction) is faster than the basic FFT method for the Dirichlet problem, but the introduction of periodic boundary conditions in one direction tends to reverse the position.

(3) Buneman's algorithm (especially Variant 2) can be implemented with fewer operations than quoted, for example, by Hockney (1970).

(4) FACR (1) algorithms, combining the FFT method with one preliminary level of cyclic reduction, are faster than either of the basic methods except for the Dirichlet problem on small grids, for which Buneman's algorithm remains the fastest. A forthcoming report (Temperton, 1977c) will examine FACR(ℓ) algorithms, in which the FFT method is combined with ℓ preliminary levels of cyclic reduction to give a further increase in speed.

(5) The cyclic reduction step for the FACR(1) algorithm can be incorporated in at least three different ways.

(6) Programming details are important; in particular, Fortran routines should treat the arrays as one-dimensional rather than two-dimensional, even when a sophisticated compiler is available.

(7) Buneman's algorithm is generally the most accurate of those considered here, but the differences are unlikely to be important, at any rate for small grids.

(8) For large grids, single-precision arithmetic may be inadequate on IBM machines.

(9) More complicated problems, and storage restrictions, tend to favour the FFT method since it requires fewer tridiagonal solutions than Buneman's algorithm.

10. Acknowledgement

Most of this work was completed while the author was at the U.K. Meteorological Office, Bracknell.

REFERENCES

Bergland, G.D. 1968 "A Fast Fourier Transform algorithm for real-valued series", Comm. ACM, 11, pp. 703-710.

Buzbee, B.L., Golub, G.H. and Nielson, C.W. 1970 "On direct methods for solving Poisson's equations", SIAM J.Numer.Anal. 7, pp.627-656.

Buzbee,B.L., Dorr,F.W., George,J.A. and Golub,G.H. 1971 "The direct solution of the discrete Poisson equation on irregular regions", SIAM J.Numer.Anal.8, pp.722-736.

Buzbee,B.L. and Dorr,F.W. 1974 "The direct solution of the biharmonic equation on rectangular regions and the Poisson equation on irregular regions", SIAM J.Numer.Anal.11,pp.753-763.

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, pp.315-337.

Evans,D.J. 1972 "An algorithm for the solution of certain tridiagonal systems of linear equations", Computer J. 15, pp.356-359.

Evans,D.J. and Hatzopoulos,M. 1976 "The solution of certain banded systems of linear equations using the folding algorithm", Computer J. 19, pp.184-187.

Fischer, D., Golub,G. Hald,O., Leiva,C. and Widlund,O. 1974 "On Fourier-Toeplitz methods for separable elliptic problems", Math.Comp. 28, pp.348-368.

Heller, D. 1976 "Some aspects of the cyclic reduction algorithm for block tridiagonal linear systems", SIAM J.Numer.Anal. 13,pp.484-495.

Hockney, R.W. 1965 "A fast direct solution of Poisson's equation using Fourier analysis", J.ACM 12, pp.95-113.

Hockney,R.W. 1970 "The potential calculation and some applications", Methods in Computational Physics, 9, pp.135-211.

REFERENCES

Hockney,R.W. 1977 "Computers,compilers and Poisson-solvers", to appear.

Le Bail,R.C. 1972 "Use of Fast Fourier Transforms for solving partial differential equations in physics", J.Comp.Phys. 9, pp.440-465.

Schröder,J. and Trottenberg,U. 1973 "Reduktionsverfahren für Differenzengleichungen bei Randwertaufgaben: I ", Numer.Math. 22, pp.37-68.

Schumann,U. and Sweet,R.A. 1976 "A direct method for the solution of Poisson's equation with Neumann boundary conditions on a staggered grid of arbitrary size", J.Comp.Phys. 20, pp.171-182.

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

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

Swarztrauber,P.N. 1974b "A direct method for the discrete solution of separable elliptic equations", SIAM J.Numer.Anal. 11,pp.1136-1150.

Swarztrauber,P.N. 1977 "The methods of cyclic reduction, Fourier analysis and cyclic reduction-Fourier analysis for the discrete solution of Poisson's equation on a rectangle", to appear in SIAM Review,July 1977.

Swarztrauber,P.N. and Sweet,R.A. 1973 "The direct solution of the discrete Poisson equation on a disk", SIAM J.Numer.Anal. 10, pp.900-907.

Sweet,R.A. 1973 "Direct methods for the solution of Poisson's equation on a staggered grid", J.Comp.Phys. 12, pp.422-428.

REFERENCES

Sweet,R.A.	1974	"A generalized cyclic reduction algorithm", SIAM J.Numer.Anal. 11,pp.506-520.
Temperton, C.	1975	"Algorithms for the solution of cyclic tridiagonal systems", J.Comp.Phys. 19, pp.317-323.
Temperton, C.	1977a	"Mixed-radix Fast Fourier Transforms without reordering", ECMWF Technical Report No. 3.
Temperton, C.	1977b	"An improved algorithm for the direct solution of Poisson's equation over irregular regions", ECWMF Research Dept. Internal Report No. 5.
Temperton, C.	1977c	"On the FACR (ℓ) algorithm for the discrete Poisson equation", ECMWF Research Dept. Internal Report No. 14.
Varga, R.S.	1962	"Matrix Iterative Analysis", Prentice-Hall, Englewood Cliffs,N.J.

EUROPEAN CENTRE FOR
MEDIUM RANGE WEATHER
FORECASTS

Research Department (RD)
Internal Report No. 13

No. 1 Users Guide for the G.F.D.L. Model
(November 1976)

No. 2 The Effect of Replacing Southern Hemispheric Analyses by Climatology on Medium Range Weather Forecasts
(January 1977)

No. 3 Test of a Lateral Boundary Relaxation Scheme in a Barotropic Model
(February 1977)

No. 4 Parameterisation of the surface fluxes
(February 1977)

No. 5 An Improved Algorithm for the Direct Solution of Poisson's Equation over Irregular Regions
(February 1977)

No. 6 Comparative Extended Range Numerical Integrations with the E.C.M.W.F. Global Forecasting Model 1: The N24, Non-Adiabatic Experiment
(March 1977)

No. 7 The E.C.M.W.F. Limited Area Model
(March 1977)

No. 8 A Comprehensive Radiation Scheme designed for Fast Computation
(May 1977)

No. 9 Documentation for the E.C.M.W.F. Grid Point Model
(May 1977)

No. 10 Numerical Tests of Parameterisation Schemes at an Actual Case of Transformation of Arctic Air
(June 1977)

No. 11 Analysis Error Calculations for the FGGE
(June 1977)

No. 12 Normal Modes of a Barotropic Version of the E.C.M.W.F. Gridpoint Model
(July 1977)

No. 13 Direct Methods for the Solution of the Discrete Poisson Equation : Some Comparisons
(July 1977)