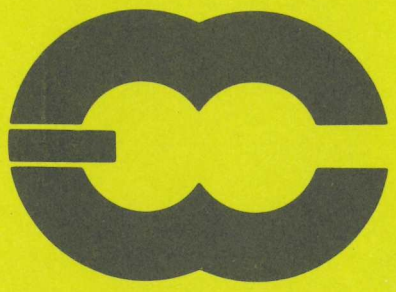

European Centre
for Medium Range
Weather Forecasts

On the FACR (ℓ) Algorithm for the Discrete Poisson Equation

Internal Report 14
Research Dept. September 77.

Centre Européen pour les Prévisions Météorologiques
a Moyen Terme

Europäisches Zentrum Für Mittelfristige Wettervorhersagen

1. Introduction

In a previous paper (Temperton, 1977b), the author compared several direct methods for the solution of the discrete Poisson equation over an N x M rectangular grid, in terms of operation counts, storage requirements, speed and accuracy. The methods considered included FFT-based algorithms, block-cyclic reduction (Buneman's algorithm), and FACR (1) algorithms, in which one preliminary step of block-cyclic reduction is used to halve either the number or the length of the Fourier transforms.

In this paper we consider the FACR (ℓ) algorithm, in which ℓ preliminary steps of block-cyclic reduction are carried out, the reduced system is solved by the FFT method, and the solution is completed by ℓ steps of block back-substitution. Both the basic FFT method and Buneman's algorithm are special cases of the FACR (ℓ) algorithm; with the optimum value of ℓ, the FACR (ℓ) method is probably the fastest known numerically stable algorithm for solving the discrete Poisson equation over a rectangle.

Hockney (1970) first introduced the FACR (ℓ) algorithm, using ℓ preliminary steps of a numerically unstable form of block-cyclic reduction, which nevertheless gave satisfactory results for small values of ℓ. Swarztrauber (1977) used a stable cyclic reduction scheme based on Variant 2 of Buneman's algorithm (Buzbee et al., 1970), derived the optimum value of ℓ, and showed that this gave an operation count asymptotically proportional to $MN \log_2(\log_2 N)$. In this paper we replace Variant 2 of Buneman's algorithm by Variant 1 for the cyclic reduction and repeat Swarztrauber's analysis under a rather different set of assumptions and definitions. Again it is shown that for optimum ℓ the operation count is asymptotically proportional to $MN \log_2(\log_2 N)$; however, it will be demonstrated that for practicable grid sizes the operation count is effectively proportional to MN. It will also be shown that the FACR(ℓ) algorithm arises quite naturally from a study of the relationship between the basic FFT and block-cyclic reduction methods.

$$A = \begin{bmatrix} -4 & 1 & & & & \\ 1 & -4 & 1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & 1 & -4 & 1 & \\ & & & 1 & -4 & \end{bmatrix}$$

and $\underset{\sim}{x}_0 = \underset{\sim}{x}_M = \underset{\sim}{0}$. (Non-homogeneous boundary conditions at $j = 0,M$ can be handled by modifying $\underset{\sim}{b}_1$ and $\underset{\sim}{b}_{M-1}$, and at $i = 0,N$ by modifying the first and last components of each $\underset{\sim}{b}_j$).

We first set out the cyclic odd-even and factorization (CORF) algorithm for solving this system as given by Buzbee et al. (1970).

Define $A^{(0)} = A$, $\underset{\sim}{b}_j^{(0)} = \underset{\sim}{b}_j$, $1 < j \leqslant M-1$.

Then after r reduction steps we have the block-tridiagonal system

$$\underset{\sim}{x}_{j-2^r} + A^{(r)} \underset{\sim}{x}_j + \underset{\sim}{x}_{j+2^r} = \underset{\sim}{b}_j^{(r)} \quad \ldots\ldots\ldots \quad (1)$$

involving only those values of j which are multiples of 2^r, where $A^{(r)}$, $b_j^{(r)}$ are defined recursively by:

$$A^{(r)} = 2I - (A^{(r-1)})^2 \quad (r > 0) \ldots\ldots\ldots\ldots \quad (2)$$

$$\underset{\sim}{b}_j^{(r)} = \underset{\sim}{b}_{j-2^{r-1}}^{(r-1)} + \underset{\sim}{b}_{j+2^{r-1}}^{(r-1)} - A^{(r-1)} \underset{\sim}{b}_j^{(r-1)} . \quad (3)$$

for $j=2^r, 2 \times 2^r, \ldots, M-2^r$.

In particular, after $k = (\log_2 M-1)$ reduction steps the system has been reduced to a single equation:

$$A^{(k)} \underset{\sim}{x}_{2^k} = \underset{\sim}{b}_{2^k}^{(k)} \quad \ldots\ldots\ldots\ldots \quad (4)$$

which can be solved for $\underset{\sim}{x}_{2^k}$. The remainder of the system can then be solved by $k = (\log_2 M-1)$ steps of back-substitution, using decreasing values of r; at each step we use Eq.(1) to obtain $\underset{\sim}{x}_j$ for each j equal to an odd multiple of 2^r, the solution having already been found for all j equal to even multiples of 2^r.

$$\underset{\sim}{x}_{2^k} = S^{-1}\left(\Lambda^{(k)}\right)^{-1} S\underset{\sim}{b}_{2^k}^{(k)} \quad (k=\log_2 M-1) \tag{8}$$

$$\text{and } \underset{\sim}{x}_j = S^{-1}\left(\Lambda^{(r)}\right)^{-1} S\left[\underset{\sim}{b}_j^{(r)} - \underset{\sim}{x}_{j-2^r} - \underset{\sim}{x}_{j+2^r}\right] \tag{9}$$

$$(r=\log_2 M-2,\ldots,1,0; \quad j=2^r,\ 3\times 2^r,\ \ldots,\ M-2^r).$$

We will not discuss the numerical stability of the algorithm defined by Eqs. (7) - (9), but rather note that it requires almost twice as many sine transforms (and their inverses) as necessary. If we first compute

$$\hat{\underset{\sim}{b}}_j^{(0)} = S\underset{\sim}{b}_j, \quad 1 \leq j \leq M-1 \tag{10}$$

then the algorithm becomes:

$$\hat{\underset{\sim}{b}}_j^{(r)} = \hat{\underset{\sim}{b}}_{j-2^{r-1}}^{(r-1)} + \hat{\underset{\sim}{b}}_{j+2^{r-1}}^{(r-1)} - \Lambda^{(r-1)}\hat{\underset{\sim}{b}}_j^{(r-1)} \tag{11}$$

$$(r=1,2,\ldots,\log_2 M-1 \ ; \quad j=2^r, 2\times 2^r, \ldots,\ M-2^r)$$

$$\hat{\underset{\sim}{x}}_{2^k} = \left(\Lambda^{(k)}\right)^{-1} \hat{\underset{\sim}{b}}_{2^k}^{(k)} \quad (k=\log_2 M-1) \tag{12}$$

$$\hat{\underset{\sim}{x}}_j = \left(\Lambda^{(r)}\right)^{-1}\left[\hat{\underset{\sim}{b}}_j^{(r)} - \hat{\underset{\sim}{x}}_{j-2^r} - \hat{\underset{\sim}{x}}_{j+2^r}\right] \tag{13}$$

$$(r=\log_2 M-2,\ldots,1,0; \quad j=2^r,\ 3\times 2^r,\ \ldots,\ M-2^r\ )$$

and finally

$$\underset{\sim}{x}_j = S^{-1}\hat{\underset{\sim}{x}}_j, \quad 1 \leq j \leq M-1 \tag{14}$$

But the algorithm defined by Eqs. (10) - (14) is precisely the "basic FFT" method, with the tridiagonal systems solved by (scalar) cyclic reduction. Thus the block-cyclic reduction method (stabilized by using the factorization of Eq.(6)) and the basic FFT method (with the tridiagonal systems solved by cyclic reduction) are equivalent. By the time the block-cyclic reduction method has been stabilized instead by Buneman's algorithm, and the basic FFT method has been modified to solve the tridiagonal systems by Gaussian elimination, the two resulting algorithms appear quite different; nevertheless, as the above

After ℓ cyclic reduction steps, we have the following system:

$$\underset{\sim}{x}_{j-2^{\ell}} + A^{(\ell)}\underset{\sim}{x}_j + \underset{\sim}{x}_{j+2^{\ell}} = A^{(\ell)}\underset{\sim}{p}_j^{(\ell)} + \underset{\sim}{q}_j^{(\ell)} \tag{17}$$

with $\underset{\sim}{x}_0 = \underset{\sim}{x}_M = \underset{\sim}{0}$, involving only those values of j which are multiples of 2^{ℓ}.

If we then define

$$\underset{\sim}{y}_j = \underset{\sim}{x}_j - \underset{\sim}{p}_j^{(\ell)} \tag{18}$$

$$\underset{\sim}{g}_j = \underset{\sim}{q}_j^{(\ell)} - (\underset{\sim}{p}_{j-2^{\ell}}^{(\ell)} + \underset{\sim}{p}_{j+2^{\ell}}^{(\ell)}), \tag{19}$$

Eq. (17) can be rewritten as

$$\underset{\sim}{y}_{j-2^{\ell}} + A^{(\ell)}\underset{\sim}{y}_j + \underset{\sim}{y}_{j+2^{\ell}} = \underset{\sim}{g}_j \tag{20}$$

for $j = 2^{\ell}, 2x2^{\ell}, \ldots, M-2^{\ell}$, with $\underset{\sim}{y}_0 = \underset{\sim}{y}_M = \underset{\sim}{0}$.

Now defining $\hat{\underset{\sim}{y}}_j = S\underset{\sim}{y}_j$, $\hat{\underset{\sim}{g}}_j = S\underset{\sim}{g}_j$, and using Eq. (6) to factorize $A^{(\ell)}$, Eq. (20) becomes

$$\hat{\underset{\sim}{y}}_{j-2^{\ell}} + \Lambda^{(\ell)}\hat{\underset{\sim}{y}}_j + \hat{\underset{\sim}{y}}_{j+2^{\ell}} = \hat{\underset{\sim}{g}}_j \tag{21}.$$

As $\Lambda^{(\ell)}$ is diagonal, Eq. (21) represents a set of (N-1) independent tridiagonal systems which can easily be solved for $\hat{\underset{\sim}{y}}_j$, $j=2^{\ell}, 2x2^{\ell}, \ldots M-2^{\ell}$. From each $\hat{\underset{\sim}{y}}_j$ we obtain $\underset{\sim}{y}_j = S^{-1}\hat{\underset{\sim}{y}}_j$ and hence $\underset{\sim}{x}_j = \underset{\sim}{y}_j + \underset{\sim}{p}_j^{(\ell)}$.

The remaining $\underset{\sim}{x}_j$ are then found in ℓ steps of back-substitution: for $r = \ell-1, \ell-2, \ldots, 0$ we solve the system

$$A^{(r)}(\underset{\sim}{x}_j - \underset{\sim}{p}_j^{(r)}) = \underset{\sim}{q}_j^{(r)} - (\underset{\sim}{x}_{j-2^r} + \underset{\sim}{x}_{j+2^r}) \tag{22}$$

for $j=2^r, 3x2^r, \ldots, M-2^r$, using once again the factorization of $A^{(r)}$ given by Eq. (5).

4. Operation counts and optimum ℓ

Swarztrauber (1977) presented an algorithm very similar to the one described above, the differences being that the block-cyclic reduction was performed using Variant 2 of Buneman's algorithm (in which the vectors $\underset{\sim}{p}_j^{(r)}$ are eliminated), and the tridiagonal systems were solved by (scalar) cyclic reduction. In deriving an operation count and hence determining the optimum value of ℓ, Swarztrauber defined an operation as consisting of a multiplication or division together with an addition or subtraction, and included only those operations which contributed towards the asymptotic operation count. In this paper we take a somewhat different viewpoint; we count additions and multiplications separately, and include all of them (apart from some lower-order terms of little significance). In fact it turns out that for practicable grid sizes, there are approximately twice as many additions as multiplications, while the asymptotic operation count underestimates the actual operation count by at least 50%.

In deriving an operation count we assume, following Temperton (1977b), that a tridiagonal system of order n can be solved in $2n$ additions and $2n$ multiplications using precomputed coefficients, while a sine transform of order n takes $(1.5 \log_2 n + 2.5)n$ additions and $(\log_2 n - 0.5)\, n$ multiplications.

During the r^{th} preliminary step of cyclic reduction ($1 \leqslant r \leqslant \ell$), we have to solve $(M-2^r)/2$ tridiagonal systems of order $(N-1)$; altogether these contribute approximately ℓMN additions and ℓMN multiplications. Implementation of Eqs. (15) and (16) involves some extra additions : approximately $3MN/2$ for the first reduction ($r=1$), since $\underset{\sim}{p}_j^{(0)} = \underset{\sim}{0}$ for all j, and $6MN/2^r$ for the r^{th} reduction ($2 \leqslant r \leqslant \ell$). The total number of extra additions from this phase is thus approximately $MN\,(3/2 + 6\sum_{r=2}^{\ell} 2^{-r}) = MN\,(9/2 - 6/2^{\ell})$.

After ℓ steps of cyclic reduction, we are left with a system of order $L = (N-1)\,(M/2^{\ell} - 1)$, defined by Eq. (17). Computation of the vectors $\underset{\sim}{g}_j$ takes $2L$ additions; the sine transforms to find

optimum ℓ:

$$[2\ \log_2(\log_2 N) + 10\ ]MN \quad \text{additions}$$
$$\text{and}\ [2\ \log_2(\log_2 N) + 2\ ]MN \quad \text{multiplications.}$$

Several observations are in order here. Firstly, the optimum value of ℓ and the total number of operations per point depend only on N, the length of the Fourier transforms. Secondly, assuming that the range of practicable grid sizes is $16 \leqslant N \leqslant 256$, the operation count for FACR(ℓ) with optimum ℓ is 14-16 additions and 6-8 multiplications per point; hence the FACR(ℓ) algorithm represents a very close approach to the elusive "stable $O(N^2)$ algorithm" (Bank and Rose, 1975; Dorr, 1975). Even for N=4096 the operation count is only 17 additions and 9 multiplications per point. Thirdly, while the actual count for practicable grid sizes is 20-24 operations per point, the "asymptotic" count is $4\ \log_2(\log_2 N)$ or 8-12 operations per point, an underestimate by at least 50%.

It has already been mentioned that this analysis is only valid for $1 \leqslant \ell \leqslant \log_2 M - 1$; in Table 1 the approximate numbers of additions and multiplications per point are presented for the NxN Dirichlet problem with $8 \leqslant N \leqslant 128$ and all possible values of ℓ, including $\ell = 0$ (the basic FFT method) and $\ell = \log_2 N$ (Buneman's algorithm). In deriving Table 1, some lower-order terms, omitted in the foregoing analysis, were included. Notice that for N=8, Buneman's algorithm requires fewer operations than for any $\ell < \log_2 N$. For $N \geqslant 16$, the operation count is minimized at $\ell = 2$ or $\ell = 3$, in accordance with the analysis above; the minimum is quite shallow.

Throughout this discussion we have assumed that all tridiagonal systems are solved by Gaussian elimination using precomputed coefficients. It is of interest to determine the number of such coefficients which are required, since this affects both the storage requirements and the time taken for preprocessing. For the simple tridiagonal systems encountered here, a system of order n requires n precomputed coefficients, which can be calculated with (n-1) additions and n divisions.

requires fewer coefficients (and hence also less preprocessing) than either of the basic methods.

TABLE 2

Number of precomputed coefficients ν required for tridiagonal systems for the N x N Dirichlet problem; tabulated value is $\nu/(N-1)$.

ℓ	N				
	8	16	32	64	128
0	7	15	31	63	127
1	4	8	16	32	64
2	4	6	10	18	34
3	7	8	10	14	22
4	-	15	15	18	22
5	-	-	31	32	34
6	-	-	-	63	64
7	-	-	-	-	127

As outlined in Section 6, the number of coefficients can in fact be further reduced by almost a factor of 2 if a modified form of Gaussian elimination is used for the tridiagonal systems.

TABLE 3

CDC 6600 CPU times (sec.) for the
N x N Dirichlet problem

ℓ	8	16	32	64	128
0	8.41×10^{-3}	3.14×10^{-2}	9.70×10^{-2}	3.80×10^{-1}	1.39×10^{0}
1	4.40×10^{-3}	1.79×10^{-2}	6.20×10^{-2}	2.60×10^{-1}	8.96×10^{-1}
2	2.97×10^{-3}	1.29×10^{-2}	4.79×10^{-2}	1.94×10^{-1}	7.44×10^{-1}
3	1.96×10^{-3}	1.06×10^{-2}	4.40×10^{-2}	1.80×10^{-1}	7.44×10^{-1}
4	-	9.47×10^{-3}	4.18×10^{-2}	1.86×10^{-1}	7.43×10^{-1}
5	-	-	4.79×10^{-2}	1.97×10^{-1}	8.79×10^{-1}
6	-	-	-	2.12×10^{-1}	1.02×10^{0}
7	-	-	-	-	1.07×10^{0}

TABLE 4

IBM 360/195 CPU times (sec.) for the
N x N Dirichlet problem

ℓ	N				
	8	16	32	64	128
0	1.38×10^{-3}	5.75×10^{-3}	2.11×10^{-2}	9.04×10^{-2}	3.60×10^{-1}
1	9.38×10^{-4}	3.97×10^{-3}	1.53×10^{-2}	6.49×10^{-2}	2.63×10^{-1}
2	6.79×10^{-4}	3.22×10^{-3}	1.31×10^{-2}	5.54×10^{-2}	2.26×10^{-1}
3	5.91×10^{-4}	2.93×10^{-3}	1.31×10^{-2}	5.55×10^{-2}	2.28×10^{-1}
4	-	2.92×10^{-3}	1.37×10^{-2}	5.99×10^{-2}	2.49×10^{-1}
5	-	-	1.43×10^{-2}	6.48×10^{-2}	2.76×10^{-1}
6	-	-	-	6.86×10^{-2}	3.02×10^{-1}
7	-	-	-	-	3.20×10^{-1}

TABLE 5

Mean maximum error for the N x N Dirichlet problem
(CDC 6600, old version of sine transform)

ℓ	N				
	8	16	32	64	128
0	4.19×10^{-14}	1.38×10^{-13}	4.24×10^{-13}	2.16×10^{-12}	8.34×10^{-12}
1	3.02×10^{-14}	8.29×10^{-14}	3.59×10^{-13}	1.67×10^{-12}	7.62×10^{-12}
2	2.49×10^{-14}	5.04×10^{-14}	2.35×10^{-13}	7.57×10^{-13}	3.87×10^{-12}
3	2.42×10^{-14}	4.17×10^{-14}	1.09×10^{-13}	4.28×10^{-13}	1.99×10^{-12}
4	-	4.05×10^{-14}	6.55×10^{-14}	2.16×10^{-13}	9.40×10^{-13}
5	-	-	6.73×10^{-14}	1.22×10^{-13}	4.16×10^{-13}
6	-	-	-	1.14×10^{-13}	2.24×10^{-13}
7	-	-	-	-	1.71×10^{-13}

TABLE 6

Mean maximum error for the N x N Dirichlet problem
(CDC 6600, new version of sine transform)

ℓ	N				
	8	16	32	64	128
0	5.68×10^{-14}	1.14×10^{-13}	2.10×10^{-13}	4.30×10^{-13}	8.94×10^{-13}
1	3.38×10^{-14}	7.30×10^{-14}	1.22×10^{-13}	3.17×10^{-13}	5.89×10^{-13}
2	2.42×10^{-14}	4.73×10^{-14}	6.65×10^{-14}	2.05×10^{-13}	3.81×10^{-13}
3	2.42×10^{-14}	4.07×10^{-14}	6.59×10^{-14}	1.46×10^{-13}	2.85×10^{-13}
4	-	4.05×10^{-14}	6.64×10^{-14}	1.17×10^{-13}	2.29×10^{-13}
5	-	-	6.73×10^{-14}	1.11×10^{-13}	1.92×10^{-13}
6	-	-	-	1.14×10^{-13}	1.79×10^{-13}
7	-	-	-	-	1.71×10^{-13}

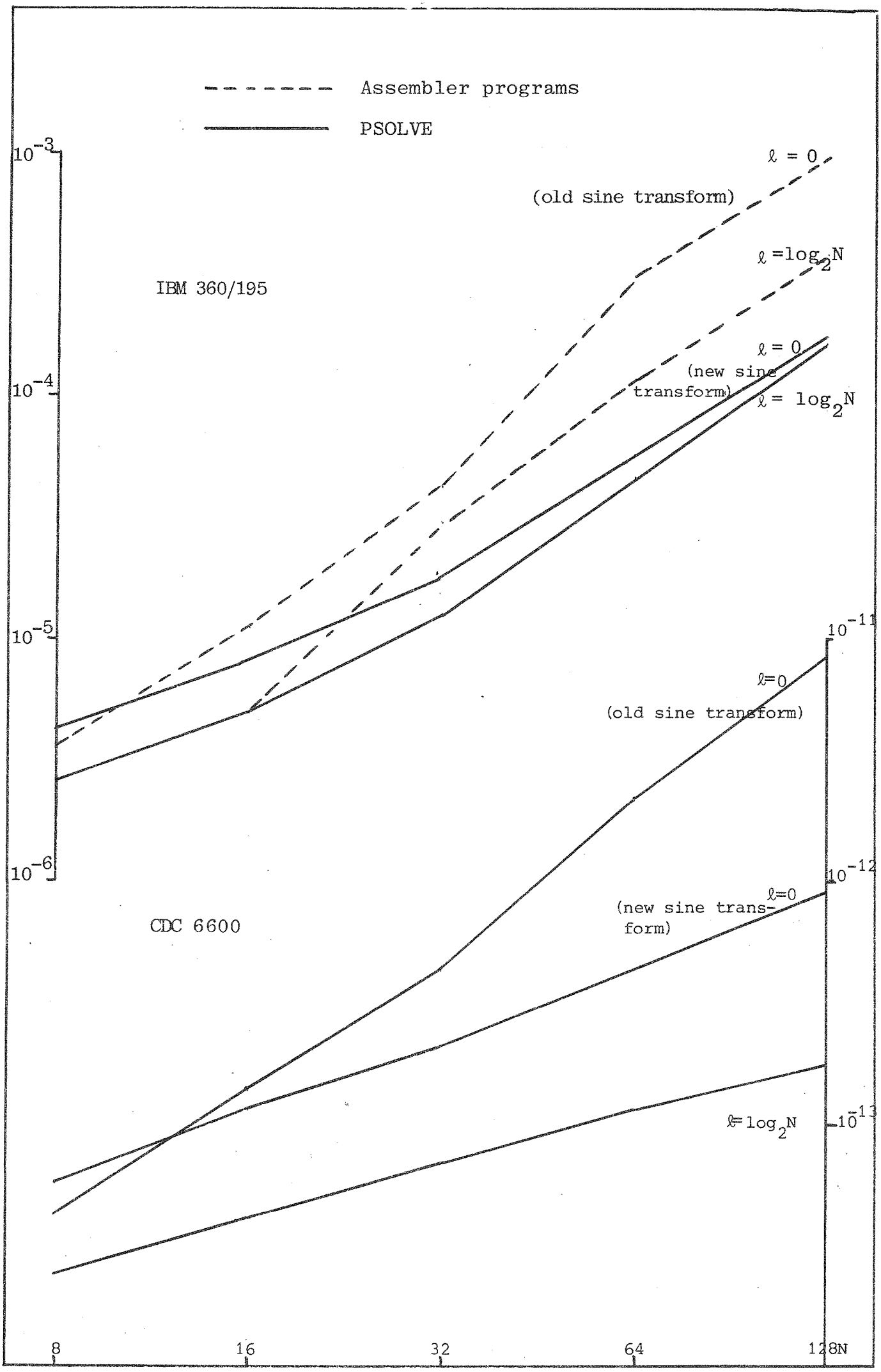

Fig.1 Mean maximum absolute errors

optimum value the coefficient array is much smaller than the solution and right-hand side arrays. As noted by Temperton (1977b), the storage requirement for the coefficients can be almost halved by using "symmetric" Gaussian elimination (Evans and Hatzopoulos, 1976); with this modification available, it appears that using alternative methods for the tridiagonal systems will seldom be worthwhile.

References

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

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

Schumann, U. 1977 Report on the GAMM-Workshop on fast solution methods for the discretized Poisson equation, Karlsruhe, March 3-4, 1977.

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. Audio and Electroacoustics, AU-17, pp. 93-103.

Swarztrauber, P.N. 1977 " The methods of cyclic reduction, Fourier analysis and the FACR algorithm for the discrete solution of Poisson's equation on a rectangle", SIAM Review 19, pp.490-501.

A P P E N D I X : PSOLVE

WARNING: In accordance with the notation specified for the GAMM Workshop on Fast Solution Methods for the Discretized Poisson Equation (for which PSOLVE was originally written), the definitions of M and N are interchanged from those used in the rest of this Report.

Description of algorithm

Subroutine PSOLVE solves the discretized Poisson equation under Dirichlet boundary conditions using a form of the FACR (ℓ) algorithm (Hockney, 1970; Swarztrauber,1977; this report). The ℓ preliminary levels of block-cyclic reduction in the y-direction are performed using Variant 1 of Buneman's algorithm (Buzbee et al., 1970); however, the tridiagonal systems are solved by Gaussian elimination. The real Fourier sine transforms in the x-direction are converted to periodic complex Fourier transforms using the inverse of the procedure of Cooley et al. (1970); the transforms are computed in "radix 4 + 2" arithmetic using a form of Fast Fourier Transform in which the usual re-ordering procedure is eliminated (Temperton,1977a). The resulting tridiagonal systems are solved by Gaussian elimination. The solution of the problem is completed by inverse sine transforms in the x-direction followed by block back-solution in the y-direction.

In most applications, the discretized Poisson equation on a given grid will be solved a number of times with different right-hand sides; consequently, all the trigonometric function values required by the FFT routine, and the coefficients required to solve tridiagonal systems by Gaussian elimination, are computed and stored by preliminary setup routines.

L, the number of preliminary levels of block-cyclic reduction, can take any non-negative integer value provided that N is a multiple of 2^L. If L = 0, then the algorithm consists simply of sine transforms, tridiagonal solutions and inverse sine transforms, without any block-cyclic reduction. If $N = 2^L$, the solution method is equivalent to Buneman's algorithm (Variant 1), and in this case there are no sine transforms, so M need not be a power of 2.

The RHS and Q arrays can be the same (i.e. the solution overwrites the right-hand side) for L = 0); otherwise they must be separate.

```
      S=TRIGS(I+1)
      BETA=-2.0*(2.0-S)
      IF (L.EQ.0) GO TO 80
      DO 70 K=1,L
   70 BETA=2.0-BETA*BETA
   80 CALL TRIDCO(COEFFS(IC),NY,BETA)
   90 IC=IC+NY
      RETURN
      END
```

```
      SUBROUTINE TRIDCO(COEFFS,NY,S)
      DIMENSION COEFFS(NY)
      COEFFS(1)=-1.0/3
      IF (NY.EQ.1) GO TO 20
      DO 10 I=2,NY
   10 COEFFS(I)=-1.0/(S+COEFFS(I-1))
      COEFFS(NY)=-COEFFS(NY)
   20 RETURN
      END
```

```
C     SUBROUTINE "PSOLVE" - SOLVES POISSON"S EQUATION OVER A
C     RECTANGULAR M X N GRID WITH GRIDLENGTH = 1, UNDER DIRICHLET
C     BOUNDARY CONDITIONS.
C
C     THIS ROUTINE USES THE FACR(L) ALGORITHM (SEE P.N. SWARZTRAUBER,
C     THE METHODS OF CYCLIC REDUCTION, FOURIER ANALYSIS AND CYCLIC
C     REDUCTION / FOURIER ANALYSIS FOR THE DIRECT SOLUTION OF POISSON"S
C     EQUATION ON A RECTANGLE").
C
C     BLOCK-CYCLIC REDUCTION IN THE Y-DIRECTION IS PERFORMED USING
C     BUNEMAN"S ALGORITHM VARIANT 1
C     FAST FOURIER TRANSFORMS IN THE X-DIRECTION ARE PERFORMED USING
C     RADIX 4+2 ARITHMETIC WITH A FORM OF FFT REQUIRING NO
C     UNSCRAMBLING
C     TRIDIAGONAL SOLUTIONS ARE PERFORMED BY GAUSSIAN ELIMINATION
C
C     NO. OF POINTS IN X-DIRECTION = (M+1)  (INCLUDING BOUNDARY POINTS)
C     NO. OF POINTS IN Y-DIRECTION = (N+1)  (INCLUDING BOUNDARY POINTS)
C     RHS IS RIGHT-HAND SIDE ARRAY
C     Q IS SOLUTION ARRAY
C     P IS AN AUXILIARY ARRAY WHICH CAN BE "INTERLEAVED" WITH THE Q
C     ARRAY, I.E. (EQUIVALENCE (P(1),Q(M+2))) OR EQUIVALENCE (P(1,1),
C     Q(1,2)))
C     COEFFS IS AN ARRAY OF COEFFICIENTS USED IN SOLVING TRIDIAGONAL
C     SYSTEMS, AND SET UP BY SUBROUTINE "PSETUP"
C     TRIGS IS AN ARRAY OF TRIGONOMETRIC FUNCTION VALUES USED BY THE
C     FFT ROUTINE, AND SET UP BY SUBROUTINE "STRIGS"
C     WORK IS A WORK AREA OF LENGTH M, USED BY THE FFT ROUTINE
C     L IS THE NUMBER OF PRELIMINARY LEVELS OF CYCLIC REDUCTION
C
C     L CAN TAKE ANY VALUE SUCH THAT N IS A MULTIPLE OF 2**L
C     FACR(0) CORRESPONDS TO SIMPLE FFT/TRIDIAGONAL SOLUTIONS/INVERSE
C     FFT METHOD
C     FACR(K) CORRESPONDS TO BUNEMAN"S ALGORITHM (VARIANT 1) IF N=2**K
C
C     RESTRICTIONS:
C     M MUST BE A POWER OF 2, EXCEPT FOR FACR(K) IF N = 2**K
C     N MUST BE A MULTIPLE OF 2**L
C     THE RHS AND Q ARRAYS CAN BE THE SAME FOR FACR(0); OTHERWISE THEY
C     MUST BE SEPARATE.
C
C     ------------------------------------------------------------
C
      SUBROUTINE PSOLVE(RHS,Q,P,COEFFS,TRIGS,WORK,M,N,L)
      DIMENSION RHS(1),Q(1),P(1),COEFFS(1),TRIGS(1),WORK(1)
      MX=M+1
      NX=N+1
      LOT=MX*NX
      MA=M-1
      NA=N-1
C     FIND KK, WHERE M = 2**KK
      LL=2
      KK=1
   10 LL=LL+LL
      KK=KK+1
      IF (LL.LT.M) GO TO 10
C     PARAMETERS FOR FFT ROUTINE
```

```
      NY=NY/2
      IJA=JUMP+1
      DO 150 IJK=IJA,LA,JUMP
      IJ=IJK+1
      DO 100 I=1,MA
      Q(IJ)=P(IJ-LL)+P(IJ+LL)-Q(IJ)
  100 IJ=IJ+1
      IC=ICBASE
      DO 130 IO=1,NN
      IJ=IJK
      G=0.0
      DO 110 I=1,MA
      IC=IC+1
      IJ=IJ+1
      G=COEFFS(IC)*(G+Q(IJ))
  110 Q(IJ)=G
      DO 120 I=2,MA
      IC=IC-1
      IJ=IJ-1
      G=COEFFS(IC)*G-Q(IJ)
  120 Q(IJ)=G
  130 IC=IC+MA-1
      DO 140 I=1,MA
      P(IJ)=P(IJ)+Q(IJ)
      Q(IJ)=Q(IJ-LL)+Q(IJ+LL)-(P(IJ)+P(IJ))
  140 IJ=IJ+1
  150 CONTINUE
  160 ICBASE=ICBASE+ICBASE+MA
  165 IF (LR.EQ.L) GO TO 200
C
C     SOLUTION FOR Q(N/2) BY BUNEMAN'S ALGORITHM
      NN=NN+NN
      IC=ICBASE
      DO 190 IO=1,NN
      IJ=IJA
      G=0.0
      DO 170 I=1,MA
      IC=IC+1
      IJ=IJ+1
      G=COEFFS(IC)*(G+Q(IJ))
  170 Q(IJ)=G
      DO 180 I=2,MA
      IC=IC-1
      IJ=IJ-1
      G=COEFFS(IC)*G-Q(IJ)
  180 Q(IJ)=G
  190 IC=IC+MA-1
      NN=NN/2
      GO TO 215
C
  200 IF (NY.GT.1) GO TO 230
C     SOLUTION FOR Q(N/2) BY SINE TRANSFORM
      CALL FFT(Q(IJA),WORK,Q(IJA),TRIGS,1,M,NFAX,LAST,+1)
      IJ=IJA+1
      IC=ICBASE+1
      DO 210 I=1,MA
      Q(IJ)=COEFFS(IC)*Q(IJ)
```

```
  350 CONTINUE
C
C     BACK-SOLUTION
  355 DO 450 IP=1,LR
      ICBASE=(ICBASE-MA)/2
      IJA=LL+1
      DO 440 IJK=IJA,LA,JUMP
      IJ=IJK+1
      IF (IP.EQ.LR) GO TO 370
      DO 360 I=1,MA
      Q(IJ)=Q(IJ)-(Q(IJ-LL)+Q(IJ+LL))
  360 IJ=IJ+1
      GO TO 390
  370 DO 380 I=1,MA
      Q(IJ)=RHS(IJ)-(Q(IJ-LL)+Q(IJ+LL))
  380 IJ=IJ+1
  390 IC=ICBASE
      DO 420 IQ=1,NN
      IJ=IJK
      G=0.0
      DO 400 I=1,MA
      IC=IC+1
      IJ=IJ+1
      G=COEFFS(IC)*(G+Q(IJ))
  400 Q(IJ)=G
      DO 410 I=2,MA
      IC=IC-1
      IJ=IJ-1
      G=COEFFS(IC)*G-Q(IJ)
  410 Q(IJ)=G
  420 IC=IC+MA-1
      IF (IP.EQ.LR) GO TO 440
      IJ=IJK+1
      DO 430 I=1,MA
      Q(IJ)=P(IJ)-Q(IJ)
  430 IJ=IJ+1
  440 CONTINUE
      JUMP=LL
      LL=LL/2
  450 NN=NN/2
      RETURN
      END
```

```
C
   80 TEMP=WORK(1)-WORK(2)
      WORK(1)=WORK(1)+WORK(2)
      WORK(2)=TEMP
      IF (N.EQ.4) GO TO 30
      K=2
      DO 20 I=3,NH,2
      A0=WORK(I)+WORK(N+2-I)
      A1=WORK(I)-WORK(N+2-I)
      A2=WORK(I+1)+WORK(N+3-I)
      A3=WORK(I+1)-WORK(N+3-I)
      TEMP=A1*TRIGS(K)+A2*TRIGS(K+NQ)
      WORK(I)=A0+TEMP
      WORK(N+2-I)=A0-TEMP
      TEMP=A2*TRIGS(K)-A1*TRIGS(K+NQ)
      WORK(I+1)=TEMP+A3
      WORK(N+3-I)=TEMP-A3
   20 K=K+1
   30 WORK(NH+1)=WORK(NH+1)+WORK(NH+1)
      WORK(NH+2)=WORK(NH+2)+WORK(NH+2)
C
C     POSTPROCESSING - SECOND STAGE
C
      B(1)=0.0
      B(INC+1)=WORK(1)
      J=INK+1
      IF (N.EQ.4) GO TO 50
      DO 40 I=6,N,2
      B(J)=WORK(I-2)
      B(J+INC)=B(J-INC)+WORK(I-3)
   40 J=J+INK
   50 B(J)=WORK(N)
      B(J+INC)=-WORK(2)
      RETURN
      END
```

```
      D(JA+J)=B0+B1
      C(JB+J)=A2-B3
      D(JB+J)=B2+A3
      C(JC+J)=A0-A1
      D(JC+J)=B0-B1
      C(JD+J)=A2+B3
      D(JD+J)=B2-A3
      I=I+INC1
      IF (K.EQ.1) GO TO 20
      TEMPR=C1*C(JB+J)-S1*D(JB+J)
      TEMPI=S1*C(JB+J)+C1*D(JB+J)
      C(JB+J)=TEMPR
      D(JB+J)=TEMPI
      TEMPR=C2*C(JC+J)-S2*D(JC+J)
      TEMPI=S2*C(JC+J)+C2*D(JC+J)
      C(JC+J)=TEMPR
      D(JC+J)=TEMPI
      TEMPR=C3*C(JD+J)-S3*D(JD+J)
      TEMPI=S3*C(JD+J)+C3*D(JD+J)
      C(JD+J)=TEMPR
      D(JD+J)=TEMPI
   20 J=J+INC2
   30 J=J+JUMP
      RETURN
C
C     CODING FOR FACTOR 2 - ASSUMES THAT 2 MUST BE LAST FACTOR, AND
C     HENCE NO ROTATION FACTORS ARE REQUIRED
C
   40 DO 50 L=1,LA
      TEMPR=A(IA+I)-A(IB+I)
      TEMPI=B(IA+I)-B(IB+I)
      C(JA+J)=A(IA+I)+A(IB+I)
      D(JA+J)=B(IA+I)+B(IB+I)
      C(JB+J)=TEMPR
      D(JB+J)=TEMPI
      I=I+INC1
   50 J=J+INC2
      RETURN
      END
```

EUROPEAN CENTRE FOR MEDIUM RANGE WEATHER FORECASTS

Research Department (RD)

Internal Report No. 14

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 Parameterization 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 ECMWF Global Forecasting Model 1: The N24, Non-Adiabatic Experiment
(March 1977)

No. 7 The ECMWF Limited Area Model
(March 1977)

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

No. 9 Documentation for the ECMWF Grid-Point Model
(May 1977)

No.10 Numerical Tests of Parameterization 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 ECMWF Grid-Point Model
(July 1977)

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

No.14 On the FACR(ℓ) Algorithm for the Discrete Poisson Equation
(September 1977)