SIAM REVIEW
Vol. 19, No. 3, July 1977

THE METHODS OF CYCLIC REDUCTION, FOURIER ANALYSIS AND THE FACR ALGORITHM FOR THE DISCRETE SOLUTION OF POISSON'S EQUATION ON A RECTANGLE*

PAUL N. SWARZTRAUBER†

Abstract. The methods of cyclic reduction and Fourier analysis are reviewed together with the FACR algorithm which combines the two methods. It is shown that the asymptotic operation count of the FACR algorithm for an $n \times n$ grid is $O(n^2 \log_2 \log_2 n)$ which compares with $O(n^2 \log_2 n)$ for either cyclic reduction or Fourier analysis used independently. In addition to the description of the methods, the treatment of the standard boundary conditions for the Fourier method is given and the efficient use of the fast Fourier transform is also discussed. Computational results are presented for grid sizes up to 1023×1023.

1. Introduction. The primary purpose of this paper is to review the efficient direct methods for solving the discrete Poisson equation on a rectangle. In particular, we shall redirect some attention to the FACR (l) algorithm [13], which combines the methods of $\underline{F}$ourier $\underline{a}$nalysis and $\underline{c}$yclic $\underline{r}$eduction. In this paper it is shown that if $l \simeq \log_2 \log_2 n - 1$, then the asymptotic operation count for an $n \times n$ grid is $O(n^2 \log_2 \log_2 n)$, compared to $O(n^2 \log_2 n)$ for either cyclic reduction or Fourier analysis used independently. The FACR (l) algorithm also requires minimal storage and, therefore, possesses the attributes which have distinguished the efficient direct methods.

Preceding a discussion of the FACR (l) algorithm in § 4, the methods of cyclic reduction and Fourier analysis are reviewed in §§ 2 and 3, respectively. Section 2 contains Buneman's second stable variant of the cyclic reduction algorithm which requires half the storage of the first variant and has the same asymptotic operation count which is derived at the end of the section. Much additional information about cyclic reduction is contained in [5], including algorithms for irregular regions and for Neumann and periodic boundary conditions. The notation in this paper is consistent with that used in [5] in order to facilitate cross reference.

Originally the method of cyclic reduction was restricted to the solution of Poisson's equation on a rectangle but it has subsequently been extended to a significantly broader class of problems. The solution on irregular regions is considered in [5], [6] and [7]. Also in [7], the method is applied to the biharmonic equation. In [14] and [16], the method is applied to Poisson's equation on the surface of the sphere and on the disk, respectively. Both coordinate singularities and singular operators are treated. In [15], the method is extended to separable elliptic equations. In [8], the method of cyclic reduction is combined with D'yakonov iteration for solving any self-adjoint, strongly elliptic equation. The Fourier and cyclic reduction methods are also implicitly found in certain of the

* Received by the editors September 24, 1975, and in revised form January 24, 1976.

† National Center for Atmospheric Research, Boulder, Colorado, 80303, which is supported by the National Science Foundation. The work was written in part while the author was a visiting lecturer at the Geophysical Fluid Dynamics Institute, Florida State University, Tallahassee, Florida 32306. This paper is contribution number 125 of the Geophysical Fluid Dynamics Institute.

marching techniques of Bank [1] and Rose [2] and in the Fourier–Toeplitz method [11].

The method of Fourier analysis is presented in § 3 together with a description of the method for the standard boundary conditions (periodic, Dirichlet–Dirichlet, Neumann–Neumann, Dirichlet–Neumann). Also discussed is the pre- and postprocessing of the data so that standard fast Fourier transform (FFT) packages can be used efficiently. The section closes with a derivation of the asymptotic operation count and a comparison with the operation count for cyclic reduction.

The FACR (l) algorithm is given in § 4. The algorithm begins with $l \simeq \log_2 \log_2 n - 1$ steps of cyclic reduction. The resulting equations are then solved via the method of Fourier analysis and the remaining unknowns are obtained by l steps of the back substitution phase. The asymptotic operation count is determined and is compared with the methods discussed in §§ 2 and 3. The section closes with an intuitive discussion of the algorithm. Section 5 contains a brief discussion of the data organization for large grids and experimental results are presented for grid sizes up to 1023×1023.

The methods presented in this paper are most frequently used and, likely, best understood in the context of solving Poisson's equation on a rectangle; however, we shall present them in the context of solving a large sparse linear system of equations, since the discretization of the Poisson's equation leading to the linear system can be found elsewhere [5].

2. Cyclic reduction.

In this section we consider the method of cyclic reduction for the solution of the large sparse linear system

$$M\mathbf{x} = \mathbf{y}, \tag{2.1}$$

where M is block tridiagonal with block order $m-1$,

$$M = \begin{bmatrix} A & I & & & & \\ I & A & I & & & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & \cdot & \cdot & \cdot \\ & & & I & A & I \\ & & & & I & A \end{bmatrix} \tag{2.2}$$

and

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{x}_{m-1} \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{y}_{m-1} \end{bmatrix}. \tag{2.3}$$

The form of M given in (2.2) corresponds to the system obtained by discretizing Poisson's equation subject to Dirichlet boundary conditions on at least two opposite sides of the rectangle. The matrix A is tridiagonal with scalar elements. Its order (and the length of the vectors $\mathbf{x}_j$, $\mathbf{y}_j$) is $n-1$, n, or $n+1$ depending on the boundary condition on the remaining sides of the rectangle (see § 3).

We now state, without derivation, Buneman's variant of the cyclic reduction algorithm for solving the system (2.1). A derivation of the algorithm, together with his first variant, plus algorithms for the standard boundary conditions, are contained in [5]. The algorithm is simplified by assuming that m has the form $m=2^{k+1}$ for some integer k, but this restriction has been removed [18], [19]. The algorithm consists of three phases: preprocessing, reduction, and backsubstitution.

Preprocessing. The matrices $A^{(r)}$ are computed from $A^{(0)}=A$, using the recurrence

$$A^{(r)}=2I-(A^{(r-1)})^2, \qquad r=1,\cdots,k. \tag{2.4}$$

We shall show later how to avoid the explicit computation and storage of the $A^{(r)}$.

Reduction. The vectors $\mathbf{q}_j^{(r)}$ are computed starting with

$$\mathbf{q}_j^{(0)}=\mathbf{y}_j, \qquad j=1,2,\cdots,m-1, \tag{2.5a}$$

$$\mathbf{q}_j^{(1)}=\mathbf{q}_{j-1}^{(0)}+\mathbf{q}_{j+1}^{(0)}-2A^{-1}\mathbf{q}_j^{(0)}, \qquad j=2,4,\cdots,m-2, \tag{2.5b}$$

The remaining $\mathbf{q}_j^{(r)}$ are determined for $r=2,\cdots,k$ and $j=2^r, 2\cdot 2^r,\cdots,m-2^r$ by

$$\begin{aligned}\mathbf{q}_j^{(r)}&=\mathbf{q}_{j-2h}^{(r-1)}-\mathbf{q}_{j-h}^{(r-2)}+\mathbf{q}_j^{(r-1)}-\mathbf{q}_{j+h}^{(r-2)}+\mathbf{q}_{j+2h}^{(r-1)}\\&\quad+(A^{(r-1)})^{-1}(\mathbf{q}_{j-3h}^{(r-2)}-\mathbf{q}_{j-2h}^{(r-1)}+\mathbf{q}_{j-h}^{(r-2)}-2\mathbf{q}_j^{(r-1)}\\&\quad+\mathbf{q}_{j+h}^{(r-2)}-\mathbf{q}_{j+2h}^{(r-1)}+\mathbf{q}_{j+3h}^{(r-2)}),\end{aligned} \tag{2.6}$$

where $h=2^{r-2}$.

Backsubstitution. If we define $\mathbf{q}_{j-1/2}^{(-1)}+\mathbf{q}_{j+1/2}^{(-1)}-\mathbf{q}_j^{(0)}=\mathbf{x}_0=\mathbf{x}_m=0$, then the solution vectors $\mathbf{x}_j$ are given for $r=k,\cdots,0$ and $j=2^r, 3\cdot 2^r,\cdots,m-2^r$ by

$$\mathbf{x}_j=\tfrac{1}{2}(\mathbf{q}_{j-2h}^{(r-1)}+\mathbf{q}_{j+2h}^{(r-1)}-\mathbf{q}_j^{(r)})+(A^{(r)})^{-1}(\mathbf{q}_j^{(r)}-\mathbf{x}_{j-4h}-\mathbf{x}_{j+4h}). \tag{2.7}$$

Note that $\mathbf{x}_{j-4h}$ and $\mathbf{x}_{j+4h}$ which appear on the right side are computed at a previous step in the backsubstitution and thus are known. We may also note that the $\mathbf{y}_j$ can be overwritten by the $\mathbf{q}_j^{(r)}$, which can in turn be overwritten by the $\mathbf{x}_j$. Hence, no additional storage is required. Buneman's second variant, given above, requires half the storage of his first variant, at the cost of some additional computation (which becomes negligible as r increases, as it does not contribute to the asymptotic operation count).

There remains a computational difficulty which is discussed only briefly here, since the details are available in [5]. We note that although A is sparse and easily inverted, the matrices $A^{(r)}$ fill rapidly; as a result, the algorithm becomes expensive in terms of both storage and computation. However, we also note that each $A^{(r)}$ is a polynomial, say $p_{2^r}(A)$, of degree 2^r in A. Let $p_{2^r}(x)$ denote the scalar analogue of $p_{2^r}(A)$; then $p_{2^r}(x)$ is given recursively from $p_1(x)=x$ and

$$p_{2^r}(x)=2-[p_{2^{r-1}}(x)]^2.$$

If we make the substitution $x = -2\cos\theta$, then

$$p_{2^r}(x) = -2\cos 2^r\theta$$

or

$$p_{2^r}(x) = -C_{2^r}(x).$$

where $C_{2^r}(x)$ is the Chebyshev polynomial with zeros

$$\alpha_\nu^{(r)} = 2\cos(2\nu-1)\frac{\pi}{2^{r+1}}, \qquad \nu = 1, \cdots, 2^r.$$

Therefore $p_{2^r}(A)$ can be expressed in factored form

$$p_{2^r}(A) = -\prod_{\nu=1}^{2^r}(A-\alpha_\nu^{(r)}I), \qquad r>0. \tag{2.8}$$

Hence, instead of storing the matrix $A^{(r)}$, only the zeros $\alpha_\nu^{(r)}$ are stored, requiring only $2m$ locations. Further, all matrix computations can be performed using the factored form of $A^{(r)}$.

For example, consider the equation (2.7) for $r = k$:

$$\mathbf{x}_{m/2} = \tfrac{1}{2}(\mathbf{q}_{m/4}^{(k-1)} + \mathbf{q}_{3\cdot m/4}^{(k-1)} - \mathbf{q}_{m/2}^{(k)}) + (A^{(k)})^{-1}\mathbf{q}_{m/2}^{(k)}. \tag{2.9}$$

Then, $\mathbf{x}_{m/2}$ is determined by first defining $\mathbf{w}_0 = \mathbf{q}_{m/2}^{(k)}$ and computing $\mathbf{w}_\nu$ recursively by solving a sequence of tridiagonal systems.

$$(A-\alpha_\nu^{(k)}I)\mathbf{w}_\nu = \mathbf{w}_{\nu-1}, \qquad \nu = 1, \cdots, m/2; \tag{2.10}$$

then

$$\mathbf{x}_{m/2} = \tfrac{1}{2}(\mathbf{q}_{m/4}^{(k-1)} + \mathbf{q}_{3\cdot m/4}^{(k-1)} - \mathbf{q}_{m/2}^{(k)}) + \mathbf{w}_{m/2}. \tag{2.11}$$

In this manner, all computations can be made in terms of sparse matrices.

We close this section with a derivation of the operation count for the cyclic reduction algorithm. We define an operation as consisting of a multiplication or division plus an addition or subtraction. We shall consider only those computations that contribute to the asymptotic count. In the reduction phase, a contribution is obtained from the second line in (2.6). Since $A^{(r-1)}$ is a polynomial of order $2^{(r-1)}$ in A, the evaluation of $\mathbf{q}_j^{(r)}$ will require the solution of $2^{(r-1)}$ tridiagonal systems. If we assume the order of A to be n, then $3n2^{(r-1)}$ operations are required since the codiagonal elements of A are 1. Since there are $m/2^r - 1$ $q_j^{(r)}$, the asymptotic count for one cycle of reduction is $\frac{3}{2}$nm. In a similar manner, the asymptotic operation count for one cycle of the backsubstitution phase (2.7) is $\frac{3}{2}$nm. Since there are k cycles of reduction and backsubstitution, the total operation count for cyclic reduction is $3nm\log_2 m$. If the cyclic reduction is performed in the other direction, the operation count becomes $3mn\log_2 n$. Therefore, the reduction should be performed in the direction which minimizes these counts. For an $n\times n$ grid, the operation count is $3n^2\log_2 n$.

3. Fourier analysis. In this section we shall discuss the method of Fourier analysis which may also be used to efficiently solve the system of linear equations (2.1). We noted previously that for the method of cyclic reduction, a detailed description of the algorithms for the standard boundary conditions was available elsewhere [5]. However, for the method of Fourier analysis, a comparable reference is not available. Therefore, we shall consider the treatment of periodic, Dirichlet–Dirichlet, Dirichlet–Neumann, and Neumann–Neumann boundary conditions. The method is efficient only when used in conjunction with the fast Fourier transform; hence, we discuss how the FFT can be adapted to take advantage of symmetries associated with each boundary condition. The section closes with an operation count for the method of Fourier analysis and a comparison with the operation count for cyclic reduction.

Referring now to the system (2.1), let Q denote the matrix whose columns are the eigenvectors of A. Then, $Q^{-1}AQ = \operatorname{diag}(\lambda_{i_1}, \lambda_{i_1+1}, \cdots, \lambda_{i_2})$, where it will be shown that the subscripts i_1, i_2 depend on the boundary conditions. When A is symmetric and Q is normalized, then $Q^{-1} = Q^T$; however, for Neumann boundary conditions, A is not symmetric. If we define $\bar{\mathbf{x}}_j = Q^{-1}\mathbf{x}_j$ and compute $\bar{\mathbf{y}}_j = Q^{-1}\mathbf{y}_j$, then, substituting into (2.1) and multiplying by Q^{-1}, we obtain

$$\bar{\mathbf{x}}_{j-1} + Q^{-1}AQ\bar{\mathbf{x}}_j + \bar{\mathbf{x}}_{j+1} = \bar{\mathbf{y}}_j, \qquad j = 1, \cdots, m-1. \tag{3.1}$$

Let $\bar{x}_{\nu,j}$ and $\bar{y}_{\nu,j}$ denote the νth component of $\bar{\mathbf{x}}_j$ and $\bar{\mathbf{y}}_j$, respectively; then for each $\nu = i_1, \cdots, i_2$ we obtain a tridiagonal system of order $m-1$.

$$\bar{x}_{\nu,j-1} + \lambda_\nu \bar{x}_{\nu,j} + \bar{x}_{\nu,j+1} = \bar{y}_{\nu,j}, \qquad j = 1, \cdots, m-1. \tag{3.2}$$

Hence, the $\bar{x}_{\nu,j}$ can be easily determined by solving a sequence of independent tridiagonal systems. The solution is then given by the inverse Fourier transform $\mathbf{x}_j = Q\bar{\mathbf{x}}_j$.

This completes the general discussion of the Fourier method; however, it remains to describe the treatment of the various boundary conditions. Since the following discussion is independent of j, its use will be discontinued and $x_{i,j}$, $\bar{x}_{\nu,j}$, $y_{i,j}$ and $\bar{y}_{\nu,j}$ will be referred to simply as x_i, $\bar{x}_\nu$, y_i, and $\bar{y}_\nu$. Also, $\mathbf{x}_j$, $\mathbf{y}_j$ will be referred to as $\mathbf{x}$ and $\mathbf{y}$.

Periodic. If A results from the discretization of Poisson's equation subject to periodic boundary conditions, then A has the form

$$A = \rho^2 \begin{bmatrix} -2\alpha & 1 & & & & 1 \\ 1 & -2\alpha & 1 & & & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & \cdot & \cdot & \cdot \\ & & & 1 & -2\alpha & 1 \\ 1 & & & & 1 & -2\alpha \end{bmatrix}_{n \times n} \tag{3.3}$$

where ρ is the ratio of the grid spacings and $\alpha = (1+\rho^2)/\rho^2$. The unknowns are x_i

for $i = 1, \cdots, n$. The Fourier transform $\bar{\mathbf{y}} = Q^{-1}\mathbf{y}$ is given by

(3.4a) $$\bar{y}_1 = \frac{2}{n} \sum_{i=1}^{n} y_i,$$

(3.4b) $$\bar{y}_{2\nu} = \frac{2}{n} \sum_{i=1}^{n} y_i \cos \nu i \frac{2\pi}{n}, \qquad \nu = 1, 2, \cdots, \frac{n}{2} - 1,$$

(3.4c) $$\bar{y}_{2\nu+1} = \frac{2}{n} \sum_{i=1}^{n} y_i \sin \nu i \frac{2\pi}{n}, \qquad \nu = 1, 2, \cdots, \frac{n}{2} - 1$$

(3.4d) $$\bar{y}_n = \frac{2}{n} \sum_{i=1}^{n} y_i (-1)^i.$$

The inverse Fourier transform $\mathbf{x} = Q\bar{\mathbf{x}}$ is given by

(3.5) $$x_i = \frac{1}{2}\bar{x}_1 + \sum_{\nu=1}^{n/2-1} \left(\bar{x}_{2\nu} \cos \nu i \frac{2\pi}{n} + \bar{x}_{2\nu+1} \sin \nu i \frac{2\pi}{n} \right) + \frac{1}{2}\bar{x}_n(-1)^i, \qquad i = 1, \cdots, n.$$

The eigenvalues of A for use in (3.2) are

(3.6a) $$\lambda_1 = -2,$$

(3.6b) $$\lambda_{2\nu} = \lambda_{2\nu+1} = -2\left(1 + 2\rho^2 \sin^2 \nu \frac{\pi}{n}\right), \qquad \nu = 1, 2, \cdots, \frac{n}{2} - 1,$$

(3.6c) $$\lambda_n = -2(1 + 2\rho^2).$$

Dirichlet–Dirichlet. If Dirichlet boundary conditions are specified at both boundaries, then the unknowns are x_i for $i = 1, \cdots, n-1$ and A has the form

(3.7) $$A = \rho^2 \begin{bmatrix} -2\alpha & 1 & & & & \\ 1 & -2\alpha & 1 & & & \\ \cdot & \cdot & \cdot & & & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & 1 & -2\alpha & 1 \\ & & & & 1 & -2\alpha \end{bmatrix}_{(n-1)\times(n-1)} .$$

The Fourier transform $\bar{\mathbf{y}} = Q^{-1}\mathbf{y}$ is given by

(3.8) $$\bar{y}_\nu = \frac{2}{n} \sum_{i=1}^{n-1} y_i \sin \nu i \frac{\pi}{n}, \qquad \nu = 1, \cdots, n-1.$$

The inverse Fourier transform $\mathbf{x} = Q\bar{\mathbf{x}}$ is

(3.9) $$x_i = \sum_{\nu=1}^{n-1} \bar{x}_\nu \sin \nu i \frac{\pi}{n}, \qquad i = 1, \cdots, n-1.$$

The eigenvalues of A for use in (3.2) are

$$\lambda_\nu = -2\left(1+2\rho^2 \sin^2 \nu\frac{\pi}{2n}\right), \qquad \nu = 1, \cdots, n-1. \tag{3.10}$$

Neumann–Neumann. If Neumann boundary conditions are specified at both boundaries, then the unknowns are x_i for $i = 0, \cdots, n$ and A has the form

$$A = \rho^2 \begin{bmatrix} -2\alpha & 2 & & & & \\ 1 & -2\alpha & 1 & & & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & \cdot & \cdot & \cdot \\ & & & 1 & -2\alpha & 1 \\ & & & & 2 & -2\alpha \end{bmatrix}_{(n+1)\times(n+1)} \tag{3.11}$$

The Fourier transform $\bar{\mathbf{y}} = Q^{-1}\mathbf{y}$ is given by

$$\bar{y}_\nu = \frac{1}{n} y_0 + \frac{2}{n} \sum_{i=1}^{n-1} y_i \cos \nu i \frac{\pi}{n} + \frac{1}{n}(-1)^\nu y_n, \qquad \nu = 0, \cdots, n. \tag{3.12}$$

The inverse Fourier transform $\mathbf{x} = Q\bar{\mathbf{x}}$ is

$$x_i = \tfrac{1}{2}\bar{x}_0 + \sum_{\nu=1}^{n-1} \bar{x}_\nu \cos \nu i \frac{\pi}{n} + \tfrac{1}{2}(-1)^i \bar{x}_n, \qquad i = 0, \cdots, n. \tag{3.13}$$

The eigenvalues of A for use in (3.2) are

$$\lambda_\nu = -2\left(1+2\rho^2 \sin^2 \nu\frac{\pi}{2n}\right), \qquad \nu = 0, \cdots, n. \tag{3.14}$$

Dirichlet–Neumann. If Dirichlet boundary conditions are specified at $i = 0$ and Neumann conditions at $i = n$, then the unknowns are x_i, $i = 1, \cdots, n$, and A has the form

$$A = \rho^2 \begin{bmatrix} -2\alpha & 1 & & & & \\ 1 & -2\alpha & 1 & & & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & \cdot & \cdot & \cdot \\ & & & 1 & -2\alpha & 1 \\ & & & & 2 & -2\alpha \end{bmatrix}_{n\times n}. \tag{3.15}$$

The Fourier transform $\bar{\mathbf{y}} = Q^{-1}\mathbf{y}$ is given by

$$\bar{y}_\nu = \frac{1}{n}(-1)^{\nu+1} y_n + \frac{2}{n} \sum_{i=1}^{n-1} y_i \sin (2\nu-1) i \frac{\pi}{2n}, \qquad \nu = 1, \cdots, n. \tag{3.16}$$

The inverse Fourier transform $\mathbf{x} = Q\bar{\mathbf{x}}$ is

$$x_i = \sum_{\nu=1}^{n} \bar{x}_\nu \sin (2\nu-1) i \frac{\pi}{2n}, \qquad i = 1, \cdots, n. \tag{3.17}$$

The eigenvalues of A for use in (3.2) are

$$\lambda_\nu = -2\left[1+2\rho^2 \sin^2 (2\nu-1)\frac{\pi}{4n}\right], \qquad \nu = 1, \cdots, n. \tag{3.18}$$

The most commonly available FFT programs are for periodic real or complex data. These programs can be adapted to perform the transforms given above by employing suitable pre- and postprocessing of the data before and after calling the FFT package. The procedures for the cosine transforms of even data and sine transforms of odd data are given in [9]. The procedures for all the remaining transforms will not be given; however, we will describe a method for the transform (3.17) which is not given in [9].

From (3.17) we can obtain

$$\begin{aligned}(x_i+x_{n-i}) &\cos i\frac{\pi}{2n} + (x_i - x_{n-i}) \sin i\frac{\pi}{2n} \\ &= \bar{x} + \sum_{\nu=1}^{n/2-1} \left[(\bar{x}_{2\nu+1} - \bar{x}_{2\nu}) \cos \nu i\frac{2\pi}{n} + (\bar{x}_{2\nu+1} + \bar{x}_{2\nu}) \sin \nu i\frac{2\pi}{n}\right] \\ &\quad + \bar{x}_n(-1)^{i+1}.\end{aligned} \tag{3.19}$$

Hence, if we preprocess $\bar{x}_\nu$ by computing

$$\bar{a}_1 = 2\bar{x}_1, \tag{3.20a}$$

$$\bar{a}_{2\nu} = \bar{x}_{2\nu+1} - \bar{x}_{2\nu}, \qquad \nu = 1, \cdots, n/2-1, \tag{3.20b}$$

$$\bar{a}_{2\nu+1} = \bar{x}_{2\nu+1} + \bar{x}_{2\nu}, \tag{3.20c}$$

$$\bar{a}_n = -2\bar{x}_n, \tag{3.20d}$$

then, the real periodic transform (3.5) can be used to compute

$$a_i = \tfrac{1}{2}\bar{a}_1 + \sum_{\nu=1}^{n/2-1}\left(\bar{a}_{2\nu}\cos \nu i\frac{2\pi}{n} + \bar{a}_{2\nu+1} \sin \nu i\frac{2\pi}{n}\right) + \tfrac{1}{2}\bar{a}_n(-1)^i. \tag{3.21}$$

But, from (3.19) and (3.21), we obtain

$$a_i = (x_i + x_{n-i}) \cos i\frac{\pi}{2n} + (x_i - x_{n-i}) \sin i\frac{\pi}{2n} \tag{3.22}$$

or

$$x_i = \tfrac{1}{2}(a_i + a_{n-i}) \sin i\frac{\pi}{2n} + \tfrac{1}{2}(a_i - a_{n-i}) \cos i\frac{\pi}{2n}. \tag{3.23}$$

Hence, once the a_i are determined from (3.21), they can be postprocessed by (3.23) to determine the x_i. Note that only a real transform of length n is required. Should only a complex FFT be available, then pre- and postprocessing algorithms which perform the efficient transform of real data using the complex FFT can be found in [3].

We close this section with some comments on the operation count of the Fourier method. The only computations which contribute to the asymptotic operation count are the Fourier transforms $\bar{\mathbf{y}} = Q^{-1}\mathbf{y}$ and $\mathbf{x} = Q\bar{\mathbf{x}}$, as the solution of (3.2) requires $O(mn)$ operations and does not contribute to the asymptotic operation count.

A real transform of length n has an asymptotic operation count of $n \log_2 n$. Since m transforms are required for both the transform and the inverse transform, the total asymptotic operation count is $2mn \log_2 n$. When compared with the count of $3mn \log_2 n$ for cyclic reduction, we see that the method of Fourier analysis has the potential for being more efficient. However, since the counts are close the final comparison will likely depend on a number of factors, including the particular program, the compiler, and the effect of low-order terms in the operation counts for smaller values of m and n.

4. Fourier analysis-cyclic reduction. In this section we shall describe the FACR (l) algorithm [13] which combines the methods of §§ 2 and 3 to produce a third method whose asymptotic operation count is less than either Fourier analysis or cyclic reduction used independently. Dirichlet boundary conditions on all sides will be assumed; however, the other standard boundary conditions can be treated by selecting the appropriate transform given in § 3.

The FACR (l) algorithm begins with l steps of cyclic reduction in which the $\mathbf{q}_j^{(r)}$ are computed using (2.5) and (2.6) for $r = 1, \cdots, l$, where $l < k$ and is yet to be determined. With $r = l$, (2.7) can be written as the linear system

$$\mathbf{x}_{j-2^l} + A^{(l)}\mathbf{x}_j + \mathbf{x}_{j+2^l} = \tfrac{1}{2}A^{(l)}(\mathbf{q}_{j-2^{l-1}}^{(l-1)} + \mathbf{q}_{j+2^{l-1}}^{(l-1)} - \mathbf{q}_j^{(l)}) + \mathbf{q}_j^{(l)} \tag{4.1}$$

for $j = 2^l, \cdots, m - 2^l$. But this system can be solved using the Fourier method. Define $\bar{\mathbf{x}}_j = Q^{-1}\mathbf{x}_j$ and compute $\bar{\mathbf{q}}_j = Q^{-1}\mathbf{q}_j$. Then substituting into (4.1) and multiplying by Q^{-1}, we obtain

$$\begin{aligned}\bar{\mathbf{x}}_{j-2^l} + Q^{-1}A^{(l)}Q\bar{\mathbf{x}}_j + \bar{\mathbf{x}}_{j+2^l} = \tfrac{1}{2}Q^{-1}A^{(l)}Q& \\ \cdot (\bar{\mathbf{q}}_{j-2^{l-1}}^{(l-1)} + \bar{\mathbf{q}}_{j+2^{l-1}}^{(l)} - \bar{\mathbf{q}}_j^{(l)}) + \mathbf{q}_j^{(l)}&.\end{aligned} \tag{4.2}$$

However, since $A^{(l)} = p_{2^l}(A)$, equation (4.2) reduces to $i_2 - i_1 + 1$ independent tridiagonal systems, of order $m/2^l - 1$.

$$\begin{aligned}&\bar{x}_{\nu,j-2^l} + p_{2^l}(\lambda_i)\bar{x}_{\nu,j} + \bar{x}_{\nu,j+2^l} \\ &\qquad = \tfrac{1}{2}p_{2^l}(\lambda_\nu)(\bar{q}_{\nu,j-2^l}^{(l-1)} + \bar{q}_{\nu,j+2^{l-1}}^{(l-1)} - \bar{q}_{\nu,j}^{(l)}) + \bar{q}_{\nu,j}^{(l)}\end{aligned} \tag{4.3}$$

where $j = 2^l, 2 \cdot 2^l, \cdots, m - 2^l$. Once $\bar{x}_{\nu,j}$ is determined from these systems then $\mathbf{x}_j$ can be obtained from the inverse transform $\mathbf{x}_j = Q\bar{\mathbf{x}}_j$. Note, however, that only every (2^l)th vector $\mathbf{x}_j$ has been determined. The remaining $\mathbf{x}_j$ are determined by l cycles of the backsubstitution phase, i.e., for $r = l - 1, \cdots, 0$, using (2.7).

The FACR (l) algorithm given in [13] is very similar to the algorithm which has just been described. However, we shall show that l can be chosen so as to reduce the asymptotic operation count below that of either cyclic reduction or Fourier analysis. To this end, we make use of the operation counts given in §§ 2 and 3 and, again, consider only those operations which contribute to the asymptotic count. Since l cycles of both the reduction and backsubstitution are required, the count for this portion of the algorithm is $3mnl$. The Fourier analysis portion requires $2 \cdot m/2^l$ transforms of length n for a count of $(mn/2^{l-1}) \log_2 n$. Hence, the total count is $C_l = 3mnl + 2^{1-l}mn \log_2 n$. This expression is minimized at $l = \log_2 \log_2 n + \log_2(\frac{2}{3} \ln 2)$ or $l \simeq \log_2 \log_2 n - 1$. With this value of l, we refer to FACR (l) simply as the FACR algorithm. Substituting into C_l, we obtain an asymptotic operation count of $3mn \log_2 \log_2 n$ for the FACR algorithm.

In practice it is likely that the exact choice of l will depend on a number of factors, including the particular program, the compiler, I/O considerations for large grids, and the low-order contributions to the operation count for small grids.

In the preceding discussion, it is not intuitively evident why the change to Fourier analysis in the middle of cyclic reduction is so effective in reducing the operation count. We close this section with a few remarks which may provide some insight. In the course of cyclic reduction, one must solve several systems of the form

$$\prod_{j=1}^{2^r} (A - \alpha_j I)\mathbf{x} = \mathbf{y}. \tag{4.4}$$

This system can be solved as a sequence of 2^r tridiagonal systems as described in § 2. If A has order n, then each system requires $3n$ operations and the total asymptotic count is $3n2^r$. Alternately, we may solve the system by the Fourier method which requires the transform of $\mathbf{y}$, the solution of a diagonal system, followed by the inverse transform to determine $\mathbf{x}$. The asymptotic count for this method is $2n \log_2 n$. Thus, if r is such that $3n2^r$ is less than $2n \log_2 n$, then (4.4) should be solved as a sequence of tridiagonal systems as in the method of cyclic reduction. However, if $3n2^r$ is greater than $2n \log n$, then the Fourier method is more efficient. The crossover point is for $r \simeq \log_2 \log_2 n + \log_2 \frac{2}{3}$.

5. Experimental results. Three programs were written, all of which solved Poisson's equation subject to Dirichlet boundary conditions on all sides. The programs were written and timed in FORTRAN. The cyclic reduction and Fourier analysis programs were "in core" and thus the times given for n larger than 127 are extrapolated in Table 1 below. The extrapolations are based on the formula $t = c_1n^2 \log_2 n + c_2n^2$ where c_1 and c_2 were determined from the actual times for $n = 63$ and $n = 127$.

The FACR program was an "out of core" program. To accommodate the large grid sizes, the large core memory (LCM) of the NCAR Control Data 7600 was used. Also, some disk storage was required. Only $2^{l+1} - 1$ vectors $\mathbf{y}_j$ were maintained in small core memory (SCM); this permitted l cycles of reduction (see (2.5), (2.6)). The right side of (4.2) was computed and stored in LCM. The next $2^{l+1} - 1$ vectors $\mathbf{y}_j$ were placed in SCM and the cycle was repeated. Once all the right sides of (4.2) were computed, then the solution of (4.3) could be obtained without significant storage manipulation. The data flow for the backsubstitution was essentially the same as the flow for the reduction phase. A value of $l = 2$ was optimal for $n = 63$ and 127, and $l = 3$ was optimal for $n = 255$, 511, and 1023. Only computational times are given in Table 1. To provide perspective on these times, they may be compared with 0.011 seconds which is required for one iteration of successive overrelaxation (SOR) on a 63×63 grid. Thus, the solution via the FACR algorithm requires roughly the same amount of time as five SOR iterations.

TABLE 1

7600 Computation times in seconds

n	Cyclic reduction	Fourier analysis	FACR
63	0.073	0.062	0.052
127	0.335	0.283	0.228
255	1.512*	1.272*	1.156
511	6.736*	5.648*	4.420
1023	29.696*	24.832*	15.977

* Extrapolated times.

TABLE 2

Asymptotic operation counts

Cyclic Reduction	$3n^2 \log_2 n$
Fourier Analysis	$2n^2 \log_2 n$
FACR	$3n^2 \log_2 \log_2 n$

REFERENCES

[1] R. E. BANK, *Marching algorithms for elliptic boundary value problems*, Ph.D. thesis, Harvard Univ., Cambridge, Mass., 1975.

[2] R. E. BANK AND D. J. ROSE, *An $O(n^2)$ method for solving constant coefficient boundary value problems in two dimensions*, SIAM J. Numer. Anal., 12 (1975), pp. 529–540.

[3] E. O. BRIGHAM, *The Fast Fourier Transform*, Prentice-Hall, Englewood Cliffs, N.J., 1974.

[4] O. BUNEMAN, *A compact non-iterative Poisson solver*, Rep. 294, Stanford University Institute for Plasma Research, Stanford, Calif., 1969.

[5] B. L. BUZBEE, G. H. GOLUB AND C. W. NIELSON, *On direct methods for solving Poisson's equations*, SIAM J. Numer. Anal., 7 (1970), pp. 627–656.

[6] B. L. BUZBEE, F. W. DORR, J. A. GEORGE AND G. H. GOLUB, *The direct solutions of the discrete Poisson equation on irregular regions*, SIAM J. Numer. Anal., 8 (1971), pp. 722–736.

[7] B. L. BUZBEE AND F. W. DORR, *The direct solution of the biharmonic equation on rectangular regions and the Poisson equation on irregular regions*, Ibid., 11 (1974), pp. 753–763.

[8] P. CONCUS AND G. H. GOLUB, *Use of fast direct methods for the efficient numerical solution of nonseparable elliptic equations*, Ibid., 10 (1973), pp. 1103–1120.

[9] J. W. COOLEY, P. A. W. LEWIS AND P. D. WELCH, *The fast Fourier transform algorithm: Programming considerations in the calculation of sine, cosine, and Laplace transforms*, J. Sound Vib., 12 (1970), pp. 315–337.

[10] F. W. DORR, *The direct solution of the discrete Poisson equation on a rectangle*, this Review, 12 (1970), pp. 248–263.

[11] D. FISCHER, G. GOLUB, O. HALD, C. LEIVA AND O. WIDLUND, *On Fourier–Toeplitz methods for separable elliptic problems*, Math. Comp., 28 (1974), pp. 349–368.

[12] R. W. HOCKNEY, *A fast direct solution of Poissons equation using Fourier analysis*, J. Assoc. Comput. Mach., 8 (1965), pp. 95–113.

[13] ———, *The potential calculation and some applications*, Methods of Computational Physics, vol. 9, B. Adler, S. Fernback and M. Rotenberg, eds., Academic Press, New York and London, 1969, pp. 136–211.

[14] P. N. SWARZTRAUBER, *The direct solution of the discrete Poisson equation on the surface of a sphere*, J. Comput. Phys., 15 (1974), pp. 46–54.
[15] ———, *A direct method for the discrete solution of separable elliptic equations*, SIAM J. Numer. Anal., 11 (1974), pp. 1136–1150.
[16] P. N. SWARZTRAUBER AND R. A. SWEET, *The direct solution of the discrete Poisson equation on a disk*, Ibid., 10 (1973), pp. 900–907.
[17] R. A. SWEET, *Direct methods for the solution of Poisson's equation on a staggered grid*, J. Comput. Phys., 12 (1973), pp. 422–428.
[18] ———, *A generalized cyclic reduction algorithm*, SIAM J. Numer. Anal., 11 (1974), pp. 506–520.
[19] ———, *Cyclic reduction algorithm for solving Poisson's equation on a grid of arbitrary size*, SIAM J. Numer. Anal., to appear.