A Direct Method for the Discrete Solution of Separable Elliptic Equations

Paul N. Swarztrauber

SIAM Journal on Numerical Analysis, Volume 11, Issue 6 (Dec., 1974), 1136-1150.

Stable URL:
http://links.jstor.org/sici?sici=0036-1429%28197412%2911%3A6%3C1136%3AADMFTD%3E2.0.CO%3B2-%23

SIAM J. NUMER. ANAL.
Vol. 11, No. 6, December 1974

A DIRECT METHOD FOR THE DISCRETE SOLUTION OF SEPARABLE ELLIPTIC EQUATIONS*

PAUL N. SWARZTRAUBER†

Abstract. This paper extends the direct method of cyclic reduction to linear systems which result from the discretization of separable elliptic equations with Dirichlet, Neumann, or periodic boundary conditions. For an $m \times n$ net, the operation count is proportional to $mn \log_2 n$ and mn storage locations are required.

1. Introduction. In recent years increased attention has been given to direct methods for solving linear systems which result from finite difference approximations to elliptic partial differential equations. A survey of direct methods and a comparison with iterative methods is given by Dorr [4]. Of particular interest are the Fourier and cyclic reduction methods. For Poisson's equation on a rectangle, these methods are desirable from a standpoint of both speed and storage. For an mn net, their operation counts are proportional to $mn \log_2 n$. The storage requirements are minimal since the solution may be returned in the storage occupied by the right side of Poisson's equation. Hockney [5], [6] employs the fast Fourier transform, but prefaces his algorithm with a few steps of the cyclic reduction method due to Golub. Buneman [1] published a stable version of the cyclic reduction algorithm with a FORTRAN computer program. This algorithm, together with generalizations, is studied in detail by Buzbee, Golub and Nielson. The method is extended to certain irregular regions by Buzbee et al. [2].

This paper extends the stabilized cyclic reduction algorithm to separable elliptic equations. This extended algorithm possesses the properties of speed and storage that have distinguished other efficient direct methods.

Section 2 contains a formal development of the algorithm for equations with Dirichlet or Neumann boundary conditions. Several problems which occur in the implementation of the algorithm are resolved in § 3. In § 4 the method is compared with matrix decomposition, and some computational results are presented. In § 5 the algorithm is developed for equations which are subject to periodic boundary conditions. Section 6 contains a proof of a theorem which is fundamental to the method and which is given in § 2.

The methods described in this paper are implemented in a FORTRAN subroutine called BLKTRI which is available from the Computing Facility at the National Center for Atmospheric Research, Boulder, Colorado 80302.

2. Development of the algorithm. Consider the following separable elliptic equation:

$$a(u)x_{uu} + b(u)x_u + c(u)x + d(v)x_{vv} + e(v)x_v + f(v)x = y(u,v), \quad a(u)d(v) > 0, \tag{2.1}$$

This general form is useful in applications when a transformation to self-adjoint form is nontrivial. The discussion is given in terms of two independent variables; however, the method generalizes to higher dimensions. If we discretize (2.1) with

* Received by the editors November 16, 1972, and in revised form August 1, 1973.

† National Center for Atmospheric Research, Boulder, Colorado, which is supported by the National Science Foundation. Now at Geophysical Fluid Dynamics Institute, Florida State University, Tallahassee, Florida 32306.

Dirichlet or Neumann boundary conditions using the usual five-point scheme, then we obtain a linear system of mn equations

$$Ax = y, \tag{2.2}$$

where A is block tridiagonal

$$A = \begin{bmatrix} B_1 & C_1 & & & & 0 \\ A_2 & B_2 & C_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & & A_{n-1} & B_{n-1} & C_{n-1} \\ 0 & & & & A_n & B_n \end{bmatrix}, \tag{2.3}$$

and

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \qquad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}. \tag{2.4}$$

The vectors x_i and y_i are of length m. There is no restriction on m; however, n must have the form $2^k - 1$. The blocks in (2.3) have order m and are of the form

$$A_i = a_i I, \tag{2.5}$$

$$B_i = B + b_i I, \tag{2.6}$$

$$C_i = c_i I, \tag{2.7}$$

where a_i, b_i and c_i are scalars. For the discretization of (2.1) the matrix B is tridiagonal; however, the algorithm is algebraically valid for arbitrary B. Hence, the method may be applied to higher-dimensional problems where B would be block tridiagonal.

The solution is obtained using an extended cyclic reduction algorithm which consists of the following three phases:

(i) *Preprocessing phase.* Results are obtained which depend only on the matrix A. These results may be used repeatedly in phases (ii) and (iii) to obtain additional solutions when only the right side of (2.2) is changed.

(ii) *Reduction phase.* A sequence of linear systems is generated starting with (2.2). At each step about half the unknown vectors x_i are eliminated with the result that each system has a block order of about half the preceding system. This process is continued until a system in the single unknown vector x_{2^k-1} is obtained.

(iii) *Back-substitution phase.* The solution vectors x_i are determined by first solving the final system generated in phase (ii) for x_{2^k-1}. Then the linear systems are used in reverse order. Additional vectors x_i are determined from each system, using the figures for x_i previously determined.

We shall begin our discussion with the reduction phase and defer phase (i) until it becomes evident which calculations should be included in it. The first

system that we generate from (2.2) will have block order $2^{k-1} - 1$ and contain the unknowns $x_2, x_4, \cdots, x_{2^k-2}$. We shall generate this system by eliminating the unknowns x_{i-1} and x_{i+1} between the three block equations corresponding to block rows $i-1$, i and $i+1$ of (2.2). If we multiply these rows by matrices Θ_i, Φ_i, Ψ_i (yet to be determined) and add, then we obtain

$$\begin{aligned}\Theta_i A_{i-1}x_{i-2} + (\Theta_i B_{i-1} + \Phi_i A_i)x_{i-1} + (\Theta_i C_{i-1} + \Phi_i B_i + \Psi_i A_{i+1})x_i \\ + (\Phi_i C_i + \Psi_i B_{i+1})x_{i+1} + \Psi_i C_{i+1}x_{i+2} \\ = \Theta_i y_{i-1} + \Phi_i y_i + \Psi_i y_{i+1}.\end{aligned} \tag{2.8}$$

In order to eliminate x_{i-1} and x_{i+1} we must select Θ_i, Φ_i and Ψ_i such that

$$\Theta_i B_{i-1} + \Phi_i A_i = 0 \tag{2.9}$$

and

$$\Phi_i C_i + \Psi_i B_{i+1} = 0. \tag{2.10}$$

Since the matrices A_i, B_i and C_i commute, this system has an infinite number of solutions. For simplicity we select

$$\Theta_i = A_i B_{i+1}, \tag{2.11}$$

$$\Phi_i = -B_{i-1}B_{i+1} \tag{2.12}$$

$$\Psi_i = C_i B_{i-1}. \tag{2.13}$$

If we substitute these into (2.8) and set

$$A_i^{(1)} = A_i B_{i+1} A_{i-1}, \tag{2.14}$$

$$B_i^{(1)} = A_i B_{i+1} C_{i-1} - B_{i-1}B_{i+1}B_i + C_i B_{i-1} A_{i+1}, \tag{2.15}$$

$$C_i^{(1)} = C_i B_{i-1} C_{i+1} \tag{2.16}$$

and

$$y_i^{(1)} = A_i B_{i+1} y_{i-1} - B_{i-1}B_{i+1}y_i + C_i B_{i-1} y_{i+1}, \tag{2.17}$$

then we obtain

$$A_i^{(1)}x_{i-2} + B_i^{(1)}x_i + C_i^{(1)}x_{i+2} = y_i^{(1)}. \tag{2.18}$$

With $i = 2, 4, \cdots, 2^k - 2$ this system has about half $(2^{k-1} - 1)$ of the unknown vectors x_i that the original system (2.1) has. Also it is block tridiagonal and we can again apply the process described above and obtain the next system in $2^{k-2} - 1$ unknowns $x_4, x_8, \cdots, x_{2^k-4}$. The general reduction algorithm will now be described. It is not the final algorithm but will be used as a basis on which to develop a stable algorithm.

Define $a_1 = c_n = 0$ and for $i = 1, 2, \cdots, n$,

$$A_i^{(0)} = a_i I, \tag{2.19}$$

$$B_i^{(0)} = B + b_i I, \tag{2.20}$$

$$C_i^{(0)} = c_i I \tag{2.21}$$

and

$$y_i^{(0)} = y_i. \tag{2.22}$$

Then for $r = 0, 1, \cdots, k-2$ and $i = 2^{r+1}, 2 \cdot 2^{r+1}, \cdots, (2^{k-r-1} - 1)2^{r+1}$,

$$\hat{A}_i^{(r+1)} = A_i^{(r)} B_{i+2^r}^{(r)} A_{i-2^r}^{(r)}, \tag{2.23}$$

$$\hat{B}_i^{(r+1)} = A_i^{(r)} B_{i+2^r}^{(r)} C_{i-2^r}^{(r)} - B_{i-2^r}^{(r)} B_{i+2^r}^{(r)} B_i^{(r)} + C_i^{(r)} B_{i-2^r}^{(r)} A_{i+2^r}^{(r)}, \tag{2.24}$$

$$\hat{C}_i^{(r+1)} = C_i^{(r)} B_{i-2^r}^{(r)} C_{i+2^r}^{(r)} \tag{2.25}$$

and

$$\hat{y}_i^{(r+1)} = A_i^{(r)} B_{i+2^r}^{(r)} r_{i-2^r}^{(r)} - B_{i-2^r}^{(r)} B_{i+2^r}^{(r)} y_i^{(r)} + B_{i-2^r}^{(r)} C_i^{(r)} y_{i+2^r}^{(r)}. \tag{2.26}$$

Each block equation is now scaled by a matrix $(G_i^{(r+1)})^{-1}$ which is fundamental to the algorithm and will be defined later.

$$A_i^{(r+1)} = (G_i^{(r+1)})^{-1} \hat{A}_i^{(r+1)}, \tag{2.27}$$

$$B_i^{(r+1)} = (G_i^{(r+1)})^{-1} \hat{B}_i^{(r+1)}, \tag{2.28}$$

$$C_i^{(r+1)} = (G_i^{(r+1)})^{-1} \hat{C}_i^{(r+1)} \tag{2.29}$$

and

$$y_i^{(r+1)} = (G_i^{(r+1)})^{-1} \hat{y}_i^{(r+1)}. \tag{2.30}$$

If we define $x_0 = x_{2^{r+1}} = 0$, then for each r we generate a block tridiagonal system

$$A_i^{(r)} x_{i-2^r} + B_i^{(r)} x_i + C_i^{(r)} x_{i+2^r} = y_i^{(r)}, \qquad i = 2^r, 2 \cdot 2^r, \cdots, (2^{k-r} - 1) \cdot 2^r, \tag{2.31}$$

which we can denote as

$$A^{(r)} x^{(r)} = y^{(r)}. \tag{2.32}$$

For $r = k - 1$ the final system (2.32) contains only the vector $x_{2^{k-1}}$:

$$B_{2^{k-1}}^{(k-1)} x_{2^{k-1}} = y_{2^{k-1}}^{(k-1)}. \tag{2.33}$$

The back-substitution phase is initiated by solving (2.33) for $x_{2^{k-1}}$ and then proceeding backward using (2.31). For $r = k-2, k-3, \cdots, 0$ and $i = 2^r, 3 \cdot 2^r, 5 \cdot 2^r, \cdots, (2^{k-r} - 1) \cdot 2^r$,

$$x_i = (B_i^{(r)})^{-1} (y_i^{(r)} - A_i^{(r)} x_{i-2^r} - C_i^{(r)} x_{i+2^r}). \tag{2.34}$$

The vectors x_{i-2^r} and x_{i+2^r} on the right of (2.34) are known from a previous step in the back-substitution.

There are two major difficulties with the algorithm (2.19) through (2.34). The first difficulty is that as r increases the matrices $A_i^{(r)}$, $B_i^{(r)}$ and $C_i^{(r)}$ fill rapidly and consequently storage and computation requirements become excessive. The second difficulty is that the algorithm is unstable. The remainder of this section will resolve these difficulties.

From (2.20) and (2.24) we observe that $B_i^{(0)}$ is linear in B and $B_i^{(1)}$ is a cubic polynomial in B. Further we observe that all the matrices $A_i^{(r)}$, $B_i^{(r)}$ and $C_i^{(r)}$ can be

expressed as polynomials in the single matrix B, if we require that $G_i^{(r+1)}$ be a common divisor of the polynomials $\hat{A}_i^{(r)}$, $\hat{B}_i^{(r)}$ and $\hat{C}_i^{(r)}$. In this event, instead of storing the matrices we may compute and store the zeros of the polynomials that represent them. These matrices will also be referred to as polynomials and all matrix operations will be performed by using their factors. For example, the computation of $(B_i^{(r)})^{-1}y_i^{(r)}$ would be performed by solving a sequence of tridiagonal systems.

From (2.19) through (2.29), the order of $B_i^{(r)}$ would triple at each step if a common divisor $G_i^{(r+1)}$ were not removed from $\hat{A}_i^{(r+1)}$, $\hat{B}_i^{(r+1)}$ and $\hat{C}_i^{(r+1)}$. Hence 3^r tridiagonal systems would be solved in computing $(B_i^{(r)})^{-1}y_i^{(r)}$. The order of the polynomials, and consequently the amount of computation, is reduced as a result of the following theorem.

THEOREM. *In the algorithm* (2.19) *through* (2.29),

$$G_i^{(r+1)} = B_{i-2^{r-1}}^{(r-1)} B_{i+2^{r-1}}^{(r-1)} \tag{2.35}$$

is a common divisor of $\hat{A}_i^{(r+1)}$, $\hat{B}_i^{(r+1)}$ *and* $\hat{C}_i^{(r+1)}$ *for* $r = 1, 2, \cdots, k-2$.

The proof of this theorem is deferred until § 6. In general, $\hat{A}_i^{(1)}$, $\hat{B}_i^{(1)}$ and $\hat{C}_i^{(1)}$ do not have a common divisor and therefore we define $G_i^{(1)} = I$.

Using (2.19), (2.23), (2.27) and (2.35), we find by induction that

$$A_i^{(r)} = \alpha_i^{(r)} B_{i+2^{r-1}}^{(r-1)}, \tag{2.36}$$

where

$$\alpha_i^{(r)} = \prod_{j=i-2^{r+1}}^{i} a_j \tag{2.37}$$

and similarly

$$C_i^{(r)} = \gamma_i^{(r)} B_{i-2^{r-1}}^{(r-1)} \tag{2.38}$$

where

$$\gamma_i^{(r)} = \prod_{j=1}^{i+2^r-1} c_j. \tag{2.39}$$

Now define $B_i^{(-1)} = I$. Using (2.24), (2.28), (2.35), (2.36) and (2.38), we can determine $B_i^{(r)}$ for $r = 0, 1, \cdots, k-2$ and $i = 2^{r+1}, 2\cdot 2^{r+1}, \cdots, (2^{k-r-1}-1)\cdot 2^{r+1}$ by

$$\begin{aligned} B_i^{(r+1)} = (B_{i-2^{r-1}}^{(r-1)} B_{i+2^{r-1}}^{(r-1)})^{-1} (&\alpha_i^{(r)} \gamma_{i-2^r}^{(r)} B_{i+2^{r-1}}^{(r-1)} B_{i-3\cdot 2^{r-1}}^{(r-1)} B_{i+2^r}^{(r)} \\ &- B_{i-2^r}^{(r)} B_i^{(r)} B_{i+2^r}^{(r)} + \alpha_{i+2^r}^{(r)} \gamma_i^{(r)} B_{i-2^{r-1}}^{(r-1)} B_{i+3\cdot 2^{r-1}}^{(r-1)} B_{i-2^r}^{(r)}). \end{aligned} \tag{2.40}$$

From the theorem we conclude that this rational function is a polynomial. These polynomials are monic except for a scalar multiple of $(-1)^r$. The middle term on the right has the highest order; if we subtract the order of the divisor and use induction, then we can determine that the order of $B_i^{(r)}$ is $2^{r+1}-1$. Since there are $2^{k-1}-1$ polynomials for each r, the storage requirement for the zeros is $(2k+5)2^k + k + 5$. For example, if $n = 127$, then $k = 7$ and 1164 locations are needed.

Equations (2.36) and (2.38) enable us to write the entire algorithm in terms of the $B_i^{(r)}$ matrices which eliminate the computation and storage of $A_i^{(r)}$ and $C_i^{(r)}$.

Therefore, the preprocessing stage consists only of computing the zeros of the $B_i^{(r)}$ polynomials using (2.40).

The reduction phase is obtained by substituting (2.36) and (2.38) into (2.26) and using (2.30) and (2.35).

$$y_i^{(r+1)} = (B_{i-2^{r-1}}^{(r-1)} B_{i+2^{r-1}}^{(r-1)})^{-1}(\alpha_i^{(r)} B_{i+2^{r-1}}^{(r-1)} B_{i+2^r}^{(r)} y_{i-2^r}^{(r)} - B_{i-2^r}^{(r)} B_{i+2^r}^{(r)} y_i^{(r)} + \gamma_i^{(r)} B_{i-2^{r-1}}^{(r-1)} B_{i-2^r}^{(r)} y_{i+2^r}^{(r)}). \tag{2.41}$$

The back-substitution phase is obtained by substituting (2.36) and (2.38) into (2.34).

$$x_i = (B_i^{(r)})^{-1}(y_i^{(r)} - \alpha_i^{(r)} B_{i+2^{r-1}}^{(r-1)} x_{i-2^r} - \gamma_i^{(r)} B_{i-2^{r-1}}^{(r-1)} x_{i+2^r}). \tag{2.42}$$

This algorithm (2.41) and (2.42) is unstable; however, it may be stabilized in the following manner. If we define

$$p_i^{(r)} = (B_{i-2^{r-1}}^{(r-1)} B_{i+2^{r-1}}^{(r-1)})^{-1} y_i^{(r)}, \tag{2.43}$$

then substituting (2.43) into (2.41) we eliminate $y_i^{(r)}$ and write the reduction phase in terms of the $p_i^{(r)}$. For $r = 0, 1, \cdots, k-2$ and $i = 2^{r+1}, 2 \cdot 2^{r+1}, \cdots, (2^{k-r-1} - 1) \cdot 2^{r+1}$,

$$p_i^{(r+1)} = \alpha_i^{(r)} (B_{i-2^{r-1}}^{(r-1)})^{-1} q_{i-2^r}^{(r)} + \gamma_i^{(r)} (B_{i+2^{r-1}}^{(r-1)})^{-1} q_{i+2^r}^{(r)} - p_i^{(r)}, \tag{2.44}$$

where

$$q_i^{(r)} = (B_i^{(r)})^{-1} B_{i-2^{r-1}}^{(r-1)} B_{i+2^{r-1}}^{(r-1)} p_i^{(r)}. \tag{2.45}$$

The back-substitution phase (2.42) may also be written in terms of $p_i^{(r)}$ for $r = k-1, k-2, \cdots, 0$ and $i = 2^r, 3 \cdot 2^r, \cdots, (2^{k-r} - 1) \cdot 2^r$.

$$x_i = (B_i^{(r)})^{-1} B_{i-2^{r-1}}^{(r-1)} B_{i+2^{r-1}}^{(r-1)} [p_i^{(r)} - \alpha_i^{(r)} (B_{i-2^{r-1}}^{(r-1)})^{-1} x_{i-2^r} - \gamma_i^{(r)} (B_{i+2^{r-1}}^{(r-1)})^{-1} x_{i+2^r}]. \tag{2.46}$$

3. Implementation of the algorithm. A formal development of the algorithm was given in the previous section; however, there remain several computational difficulties which must be resolved before the algorithm becomes useful. In this section we shall first discuss the calculation of the zeros of the $B_i^{(r)}$ polynomials in the preprocessing phase. Next we shall discuss the implementation of the reduction and back-substitution phases. We shall also include some heuristic remarks about stability and provide certain techniques for reducing the amount of computation.

The zeros of the $B_i^{(r)}$ polynomials may be determined by using a scalar analogue of the matrix recurrence relations given in § 6 or by using (2.40). For reasons of simplicity and economy, the use of (2.40) is preferred. One could determine the zeros by expressing the polynomials $B_i^{(r)}$, $B_i^{(r-1)}$ in terms of their monic basis and then using (2.40) to compute the coefficients of $B_i^{(r+1)}$. Then the zeros could be determined by using current zero-finding techniques. However, both the calculation of the coefficients and the zero-finding techniques are subject to considerable error, with the result that this method is unsatisfactory for polynomials of high order.

A more satisfactory method is to express the polynomials in factored form and to use Newton's method to determine the zeros of $B_i^{(r+1)}$. Using this technique, we have been able to determine the zeros of polynomials of order 511 using single precision on a Control Data 7600 computer.

We shall now proceed to describe in some detail a method for determining the zeros. We assume that the zeros of $B_i^{(l)}$ have been determined for $l \leqq r$. We then define

$$f(x) = \alpha_i^{(r)}\gamma_{i-2^r}^{(r)}B_{i+2^{r-1}}^{(r-1)}B_{i-3\cdot 2^r}^{(r-1)}B_{i+2^r}^{(r)}, \tag{3.1}$$

$$g(x) = B_{i-2^r}^{(r)}B_i^{(r)}B_{i+2^r}^{(r)}, \tag{3.2}$$

$$h(x) = \alpha_{i+2^r}^{(r)}\gamma_i^{(r)}B_{i-2^{r-1}}^{(r-1)}B_{i+3\cdot 2^{r-1}}^{(r-1)}B_{i-2^r}^{(r)}, \tag{3.3}$$

$$F(x) = f(x) - g(x) + h(x) \tag{3.4}$$

and

$$G(x) = B_{i-2^{r-1}}^{(r-1)}B_{i+2^{r-1}}^{(r-1)}. \tag{3.5}$$

From (2.40) a zero of $B_i^{(r+1)}$ must satisfy

$$F(\omega)/G(\omega) = 0. \tag{3.6}$$

If we define x_0 to be an initial approximation to ω, then Newton's method yields the successive approximations

$$x_{n+1} = x_n + \left(\frac{\dot{G}(x_n)}{G(x_n)} - \frac{\dot{F}(x_n)}{F(x_n)}\right)^{-1}. \tag{3.7}$$

When ω has been determined we can then set $G_1(x) = (x - \omega)G(x)$ and obtain a second zero of $F(x)/G(x)$ by applying (3.7) with $G(x)$ replaced by $G_1(x)$. Continuing in this manner we can obtain all the zeros of $F(x)/G(x)$. The quantity $\dot{G}(x)/G(x)$ is given as the sum of the inverse factors of G. $\dot{F}(x)/F(x)$ may be computed in the form

$$\frac{\dot{F}(x)}{F(x)} = \frac{\dfrac{\dot{f}(x)}{f(x)}\dfrac{f(x)}{g(x)} - \dfrac{\dot{g}(x)}{g(x)} + \dfrac{\dot{h}(x)}{h(x)}\dfrac{h(x)}{g(x)}}{\dfrac{f(x)}{g(x)} - 1 + \dfrac{h(x)}{g(x)}}. \tag{3.8}$$

The reason for evaluating $\dot{F}(x)/F(x)$ in this form is to avoid overflow. For higher order polynomials, the value of f, g or h is likely to produce an overflow condition. The calculation of $f(x)/g(x)$ proceeds by alternately multiplying by a factor of $f(x)$ and dividing by a factor of $g(x)$. From (3.1), (3.2) and (3.3) we observe that $g(x)$ has factors in common with $f(x)$ and $h(x)$, which simplifies the calculation of f/g and h/g. If a factor of $f(x)$, $g(x)$, or $h(x)$ is zero, then it can be replaced by the product of its nonzero factors with a corresponding modification of (3.8).

This method of computing the zeros is globally convergent only if the zeros are real and simple. Numerical experiments indicate that this is the case unless the matrix A deviates significantly from being diagonally dominant.

We proceed now to discuss the implementation of the reduction phase given by (2.44) and (2.45). We assume that $B_i^{(r)}$ are matrix polynomials in B. If

in the computation of $q_i^{(r)}$ we repeatedly multiply $p_i^{(r)}$ by the factors of $B_{i-2^{r-1}}^{(r-1)}B_{i+2^{r-1}}^{(r-1)}$ (or by the inverse factors of $B_i^{(r)}$), then the magnitude of the round-off error is approximately multiplied by the largest modulus of the eigenvalues of each factor (or inverse factor). The result would be that (2.44) and (2.45) would be unstable. Therefore, in computing $q_i^{(r)}$, it is necessary to multiply alternately by the inverse of a factor of $B_i^{(r)}$ and by a factor $B_{i-2^{r-1}}^{(r-1)}B_{i+2^{r-1}}^{(r-1)}$. That is, $q_i^{(r)}$ is obtained by defining $z_0 = p_i^{(r)}$ and computing z_j recursively by solving linear systems of the form

$$(B - \theta_j I)z_{j+1} = (B - \phi_j I)z_j, \tag{3.9}$$

where θ_j is a zero of $B_i^{(r)}$ and ϕ_j is a zero of $B_{i-2^{r-1}}^{(r-1)}B_{i+2^{r-1}}^{(r-1)}$. Then $q_i^{(r)}$ is given by $z_{2^{r+1}-1}$. Observe that θ_j and ϕ_j should be selected so that $|\theta_j - \phi_j|$ is as small as possible to ensure that the largest modulus of the eigenvalues of $(B - \theta_j I)^{-1} \cdot (B - \phi_j I)$ will be close to unity and that the roundoff error (and z_j) will not grow.

Because it is not possible to avoid repeated multiplication by inverse factors in computing a term like $\alpha_i^{(r)}(B_{i-2^{r-1}}^{(r-1)})^{-1}q_{i-2^r}^{(r)}$ in (2.44), this term may be subject to error. However, this error is "shifted off" when the term is added to the dominant term in the expression which is $p_i^{(r)}$. The technique used in the reduction phase should also be used in the implementation of the back-substitution phase, i.e., equation (2.46).

The preceding remarks are based mainly on a number of experimental observations which guided the development of the algorithm to its present form. Although the discussion may not be rigorous, we have included it in an attempt to provide some insight into the design of (2.44) and (2.45) and into the dependence of stability on implementation.

We close this section with an observation which results in a significant reduction of computation.

If we define $\hat{z}_{j+1}$ by

$$z_{j+1} = \hat{z}_{j+1} + z_j, \tag{3.10}$$

then substituting (3.10) into (3.9) we obtain

$$(B - \theta_j I)\hat{z}_{j+1} = (\theta_j - \phi_j)z_j. \tag{3.11}$$

When $\hat{z}_{j+1}$ has been obtained from (3.11), then z_{j+1} can be obtained from (3.10). Hence all matrix multiplications can be avoided.

4. Operation counts and experimental results. This section contains operation counts for the reduction and back-substitution phases of the algorithm. In addition, the counts and storage requirements are compared with the method of matrix decomposition which can also be used to solve the system (2.2). The algorithm is also timed for various net sizes and the roundoff error is tabulated.

If we recall that $k = \log_2(n + 1)$, then we can show that the number of tridiagonal systems which must be solved in the reduction phase, using (2.44) and (2.45), is

$$c_r = (2k - 6)(n + 1) + 2k + 6 \tag{4.1}$$

and for the back-substitution phase, using (2.46), we obtain

$$c_b = (2k - 5)(n + 1) + 2k + 5. \tag{4.2}$$

By combining these we obtain the total number of tridiagonal systems which must be solved:

$$c = (4k - 11)(n + 1) + 4k + 11. \tag{4.3}$$

We now assume that the order of B is m. If we only count multiplications and divisions, then $5m$ operations are required to solve a tridiagonal system. Hence, the total number of operations is

$$c_T = 5mc. \tag{4.4}$$

If we neglect the computation of the eigensystem, then the operation count for matrix decomposition is

$$d_T = 2n^2m + 5nm. \tag{4.5}$$

Setting $m = n$ in (4.4) and (4.5), we can develop a comparison of the operation counts (Table 1).

TABLE 1

A comparison of operation counts for an $n \times n$ net

n	Cyclic Reduction	Matrix Decomposition
15	8025	7875
31	49445	64387
63	273105	519939
127	1406525	4177411
255	6909225	33487875

The storage requirements for the eigensystem used in matrix decomposition is n^2 as compared to $(2k - 5)2^k + k + 5$ locations required for the zeros of the $B_i^{(r)}$ polynomials used in cyclic reduction. For a 63×63 net, these numbers are 3969 and 459, respectively.

We now present experimental results which were obtained by solving Poisson's equation on the interior of a sphere and subject to Dirichlet boundary conditions (Table 2). Axisymmetry was assumed, which reduced the domain to two space variables, r (radial) and θ (meridional). The grid is $n \times n$, with n points selected in both independent variables. A solution was first generated randomly and then substituted into the finite difference equations in order to compute the right side. The solution was then recomputed using this right side and the algorithm. The error is the maximum absolute difference between the known and the computed solution. The solution was of the order of magnitude one. The times are in milliseconds and the program was run on a Control Data 7600 computer.

TABLE 2
Computational results for a model problem

n	Preprocess	Solution	Error
15	28	12	7.99×10^{-14}
31	135	54	2.95×10^{-13}
63	621	259	3.63×10^{-12}
127	2807	1214	1.93×10^{-10}

5. Periodic boundary conditions. In this section we discuss the solution of linear systems which result from the descretization of separable elliptic equations which are subject to periodic boundary conditions. These systems are of the form

$$A_p x = y, \tag{5.1}$$

where A_p has the form

$$A_p = \begin{bmatrix} B_1 & C_1 & & & & & A_1 \\ A_2 & B_2 & C_2 & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & & \ddots & \ddots & & \\ & & & A_{n-1} & B_{n-1} & C_{n-1} & \\ C_n & & & & A_n & B_n & \end{bmatrix}. \tag{5.2}$$

We shall assume that n has the form $n = 2^k$ where k is any natural number. This form differs from the form of n given in § 2. The vectors x, y and the matrices A_i, B_i and C_i have the same form as in (2.4) through (2.7). The cyclic reduction algorithm can be adapted to solve (5.1). Proceeding as in (2.8) through (2.17), we compute for $i = 2, 4, \cdots, 2^k - 2$,

$$A_i^{(1)} = A_i B_{i+1} A_{i-1}, \tag{5.3}$$

$$B_i^{(1)} = A_i B_{i+1} C_{i-1} - B_{i-1} B_i B_{i+1} + C_i B_{i-1} A_{i+1}, \tag{5.4}$$

$$C_i^{(1)} = C_i B_{i-1} C_{i+1}, \tag{5.5}$$

$$y_i^{(1)} = A_i B_{i+1} y_{i-1} - B_{i-1} B_{i+1} y_i + C_i B_{i-1} y_{i+1}. \tag{5.6}$$

We then eliminate x_1 and x_{2^k-1} between the first and last two equations of (5.1) and obtain

$$A_{2^k}^{(1)} = A_{2^k} B_1 A_{2^k-1}, \tag{5.7}$$

$$B_{2^k}^{(1)} = A_{2^k} B_1 C_{2^k-1} - B_{2^k-1} B_{2^k} B_1 + C_{2^k} B_{2^k-1} A_1, \tag{5.8}$$

$$C_{2^k}^{(1)} = C_{2^k} B_{2^k-1} C_1 \tag{5.9}$$

and

$$y_{2^k}^{(1)} = A_{2^k} B_1 y_{2^k-1} - B_{2^k-1} B_1 y_{2^k} + C_{2^k} B_{2^k-1} y_1. \tag{5.10}$$

We observe that these equations may be obtained from (5.3) through (5.6) by setting $i = 2^k$ and evaluating the subscripts modulo 2^k. This result holds throughout the development of the algorithm and enables us to write the final algorithm which corresponds to (2.40), (2.44), (2.45) and (2.46). The preprocessing phase consists of computing the roots of the $B_i^{(r)}$ polynomials for $r = 0, 1, 2, \cdots, k-1$ and $i = 2^{r+1}, 2\cdot 2^{r+1}, 3\cdot 2^{r+1}, \cdots, 2^k$.

$$B_i^{(r+1)} = (B_{i-2^{r-1}}^{(r-1)} B_{i+2^{r-1}}^{(r-1)})^{-1}(\alpha_i^{(r)}\gamma_{i-2^r}^{(r)} B_{i-3\cdot 2^{r-1}}^{(r-1)} B_{i-2^r}^{(r)} - B_{i-2^r}^{(r)} B_i^{(r)} B_{i+2^r}^{(r)} + \alpha_{i+2^r}^{(r)}\gamma_i^{(r)} B_{i-2^{r-1}}^{(r-1)} B_{i+3\cdot 2^{r-1}}^{(r-1)} B_{i-2^r}^{(r)}). \tag{5.11}$$

The reduction phase consists of computing $p_i^{(r)}$ for $r = 0, 1, \cdots, k-1$ and $i = 2^{r+1}, 2\cdot 2^{r+1}, 3\cdot 2^{r+1}, \cdots, 2^k$.

$$p_i^{(r+1)} = \alpha_i^{(r)}(B_{i-2^{r-1}}^{(r-1)})^{-1} q_{i-2^r}^{(r)} + \gamma_i^{(r)}(B_{i+2^{r-1}}^{(r-1)})^{-1} q_{i+2^r}^{(r)} - p_i^{(r)}, \tag{5.12}$$

where

$$q_i^{(r)} = (B_i^{(r)})^{-1} B_{i-2^{r-1}}^{(r-1)} B_{i+2^{r-1}}^{(r-1)} p_i^{(r)}. \tag{5.13}$$

The back-substitution phase consists of computing x_i for $r = k, k-1, \cdots, 0$ and $i = 2^r, 3\cdot 2^r, 5\cdot 2^r, \cdots, (2^{k-r}-1)\cdot 2^r$.

$$x_i = (B_i^{(r)})^{-1} B_{i-2^{r-1}}^{(r-1)} B_{i+2^{r-1}}^{(r-1)} [p_i^{(r)} - \alpha_i^{(r)}(B_{i-2^{r-1}}^{(r-1)})^{-1} x_{i-2^r} - \gamma_i^{(r)}(B_{i+2^{r-1}}^{(r-1)})^{-1} x_{i+2^r}]. \tag{5.14}$$

The first step of this back-substitution phase differs from that given in § 2. With $r = k$ then (5.14) reduces to the single equation

$$(\alpha_2^{(k)} B_{2^{k-1}}^{(k-1)} + B_{2^k}^{(k)} + \gamma_{2^k}^{(k)} B_{2^{k-1}}^{(k-1)}) x_{2^k} = B_{2^{k-1}}^{(k-1)} B_{2^{k-1}}^{(k-1)} p_i^{(k)}. \tag{5.15}$$

Hence in order to determine x_{2^k}, the roots of the polynomial on the left-hand side must be computed. This can be simplified somewhat since it can be shown from (5.11) that $B_{2^k}^{(k)}$ has a factor of $B_{2^{k-1}}^{(k-1)}$ which can then be cancelled from (5.15).

6. Proof of the theorem in § 2. In this section we shall prove the theorem including (2.35) given in § 2. We shall also develop a set of recurrence relations which could be used in place of (2.40) to compute the $B_i^{(r)}$ polynomials.

If we set $r = 0$ in (2.23) through (2.29) we obtain

$$A_i^{(1)} = a_i a_{i-1} B_{i+1}^{(0)}, \tag{6.1}$$

$$C_i^{(1)} = c_i c_{i+1} B_{i-1}^{(0)}, \tag{6.2}$$

$$B_i^{(1)} = a_{i+1} c_i B_{i-1}^{(0)} - B_{i-1}^{(0)} B_i^{(0)} B_{i+1}^{(0)} + a_i c_{i-1} B_{i+1}^{(0)} \tag{6.3}$$

and

$$G_i^{(1)} = I. \tag{6.4}$$

If we continue the algorithm for $r = 1, 2$, then a pattern emerges and we are encouraged to look for a recurrence of the form

$$A_i^{(r+1)} = \alpha_i^{(r+1)} B_{i+2^r}^{(r)}, \tag{6.5}$$

$$C_i^{(r+1)} = \gamma_i^{(r+1)} B_{i-2^r}^{(r)}, \tag{6.6}$$

$$B_i^{(r+1)} = P_i^{(r)} B_{i-2^r}^{(r)} + (-1)^{r+1} B_i^{(0)} B_{i-2^r}^{(r)} B_{i+2^r}^{(r)} + Q_i^{(r)} B_{i+2^r}^{(r)} \tag{6.7}$$

$$G_i^{(r+1)} = B_{i-2^{r-1}}^{(r-1)} B_{i+2^{r-1}}^{(r-1)}. \tag{6.8}$$

The significance of this form is that no factorization by $G_i^{(r+1)}$ is required as it is in (2.40). We shall prove by induction that a solution of this form exists and determine the scalars $\alpha_i^{(r+1)}$, $\gamma_i^{(r+1)}$ and the matrices $P_i^{(r)}$ and $Q_i^{(r)}$. We shall also show that $B_{i-2^r}^{(r)} B_{i+2^r}^{(r)}$ is a factor common to $\hat{A}_i^{(r+2)}$, $\hat{B}_i^{(r+2)}$ and $\hat{C}_i^{(r+2)}$, which proves the theorem. For $r = 0$, we see that (6.1) through (6.4) has the desired form (6.5) through (6.8) if we define

$$\alpha_i^{(1)} = a_i a_{i-1}, \tag{6.9}$$

$$\gamma_i^{(1)} = c_i c_{i+1}, \tag{6.10}$$

$$P_i^{(0)} = a_{i+1} c_i I \tag{6.11}$$

and

$$Q_i^{(0)} = a_i c_{i-1} I. \tag{6.12}$$

Equation (6.4) also has the desired form if we define $B_i^{(-1)} = I$.

We shall now assume that (6.5) through (6.8) are valid for $r + 1$ and show that they are valid for $r + 2$. From (2.23), (2.24) and (2.25) we determine that

$$\hat{A}_i^{(r+2)} = \alpha_i^{(r+1)} \alpha_{i-2^{r-1}}^{(r+1)} B_{i-2^r}^{(r)} B_{i+2^r}^{(r)} B_{i+2^{r+1}}^{(r+1)}, \tag{6.13}$$

$$\hat{C}_i^{(r+2)} = \gamma_i^{(r+1)} \gamma_{i+2^{r+1}}^{(r+1)} B_{i-2^r}^{(r)} B_{i+2^r}^{(r)} B_{i-2^{r+1}}^{(r+1)} \tag{6.14}$$

and

$$\begin{aligned} \hat{B}_i^{(r+2)} = {} & \alpha_{i+2^{r+1}}^{(r+1)} \gamma_i^{(r+1)} B_{i+3\cdot 2^r}^{(r)} B_{i-2^r}^{(r)} B_{i-2^{r+1}}^{(r+1)} \\ & + \alpha_i^{(r+1)} \gamma_{i-2^{r+1}}^{(r+1)} B_{i+2^r}^{(r)} B_{i-3\cdot 2^r}^{(r)} B_{i+2^{r+1}}^{(r+1)} \\ & - [P_{i-2^{r+1}}^{(r)} B_{i-3\cdot 2^r}^{(r)} + (-1)^{r+1} B_{i-2^{r+1}}^{(0)} B_{i-3\cdot 2^r}^{(r)} B_{i-2^r}^{(r)} + Q_{i-2^{r+1}}^{(r)} B_{i-2^r}^{(r)}] \\ & \cdot [P_i^{(r)} B_{i-2^r}^{(r)} + (-1)^{r+1} B_i^{(0)} B_{i-2^r}^{(r)} B_{i+2^r}^{(r)} + Q_i^{(r)} B_{i+2^r}^{(r)}] \\ & \cdot [P_{i+2^{r+1}}^{(r)} B_{i+2^r}^{(r)} + (-1)^{r+1} B_{i+2^r}^{(0)} B_{i+2^r}^{(r)} B_{i+3\cdot 2^r}^{(r)} + Q_{i+2^{r+1}}^{(r)} B_{i+3\cdot 2^r}^{(r)}]. \end{aligned} \tag{6.15}$$

We note that $\hat{A}_i^{(r+2)}$ and $\hat{C}_i^{(r+2)}$ have the common factor $G_i^{(r+2)} = B_{i-2^r}^{(r)} B_{i+2^r}^{(r)}$ and hence we shall try to arrange (6.15) so that $\hat{B}_i^{(r+2)}$ also has such a factor.

$$\begin{aligned} \hat{B}_i^{(r+2)} = {} & B_{i-2^r}^{(r)} (\alpha_{i+2^{r+1}}^{(r+1)} \gamma_i^{(r+1)} I - P_i^{(r)} Q_{i+2^{r+1}}^{(r)}) B_{i+3\cdot 2^r}^{(r)} B_{i-2^{r+1}}^{(r+1)} \\ & + B_{i+2^r}^{(r)} (\alpha_i^{(r+1)} \gamma_{i-2^{r+1}}^{(r+1)} I - Q_i^{(r)} P_{i-2^{r+1}}^{(r)}) B_{i-3\cdot 2^r}^{(r)} B_{i+2^{r+1}}^{(r+1)} \\ & - B_{i-2^r}^{(r)} B_{i+2^r}^{(r)} P_{i+2^{r+1}}^{(r)} P_i^{(r)} B_{i-2^{r+1}}^{(r+1)} \\ & - B_{i-2^r}^{(r)} B_{i+2^r}^{(r)} Q_{i-2^{r+1}}^{(r)} Q_i^{(r)} B_{i+2^{r+1}}^{(r+1)} \\ & - B_{i-2^r}^{(r)} B_{i+2^r}^{(r)} (-1)^{r+1} B_{i+2^{r+1}}^{(0)} B_{i+3\cdot 2^r}^{(r)} P_i^{(r)} B_{i-2^{r+1}}^{(r+1)} \\ & - B_{i-2^r}^{(r)} B_{i+2^r}^{(r)} (-1)^{r+1} B_{i-2^{r+1}}^{(0)} B_{i-3\cdot 2^r}^{(r)} Q_i^{(r)} B_{i+2^{r+1}}^{(r+1)} \\ & + B_{i-2^r}^{(r)} B_{i+2^r}^{(r)} (-1)^{r+2} B_i^{(0)} B_{i-2^{r+1}}^{(r+1)} B_{i+2^{r+1}}^{(r+1)}. \end{aligned} \tag{6.16}$$

Now the only terms in (6.16) which do not have $G_i^{(r+2)} = B_{i-2^r}^{(r)} B_{i+2^r}^{(r)}$ as a factor are the first two on the right side. We shall show later, however, that the following factorization can be made:

$$\alpha_{i+2^{r+1}}^{(r+1)} \gamma_i^{(r+1)} I - P_i^{(r)} Q_{i+2^{r+1}}^{(r)} = R_i^{(r)} B_{i+2^r}^{(r)} \tag{6.17}$$

and

$$\alpha_i^{(r+1)} \gamma_{i-2^{r+1}}^{(r+1)} I - Q_i^{(r)} P_{i-2^{r+1}}^{(r)} = S_i^{(r)} B_{i-2^r}^{(r)}, \tag{6.18}$$

where $R_i^{(r)}$ and $S_i^{(r)}$ are yet to be determined. In this event $\hat{A}_i^{(r+2)}$, $\hat{B}_i^{(r+2)}$ and $\hat{C}_i^{(r+2)}$ have the common divisor $G_i^{(r+2)} = B_{i-2^r}^{(r)} B_{i+2^r}^{(r)}$, and using (2.27), (2.28) and (2.29) we obtain the desired form (6.5), (6.6) and (6.7) at $r + 2$:

$$A_i^{(r+2)} = \alpha_i^{(r+2)} B_{i+2^{r+1}}^{(r+1)}, \tag{6.19}$$

$$C_i^{(r+2)} = \gamma_i^{(r+2)} B_{i-2^{r+1}}^{(r+1)} \tag{6.20}$$

and

$$B_i^{(r+2)} = P_i^{(r+1)} B_{i-2^{r+1}}^{(r+1)} + (-1)^{r+2} B_i^{(0)} B_{i-2^{r+1}}^{(r+1)} B_{i+2^{r+1}}^{(r+1)} + Q_i^{(r+1)} B_{i+2^{r+1}}^{(r+1)}, \tag{6.21}$$

where

$$\alpha_i^{(r+2)} = \alpha_i^{(r+1)} \alpha_{i-2^{r+1}}^{(r+1)}, \tag{6.22}$$

$$\gamma_i^{(r+2)} = \gamma_i^{(r+1)} \gamma_{i+2^{r+1}}^{(r+1)} \tag{6.23}$$

and

$$P_i^{(r+1)} = R_i^{(r)} B_{i+3\cdot 2^r}^{(r)} + (-1)^r B_{i+2^{r+1}}^{(0)} B_{i+3\cdot 2^r}^{(r)} P_i^{(r)} - P_{i+2^{r+1}}^{(r)} P_i^{(r)}, \tag{6.24}$$

$$Q_i^{(r+1)} = S_i^{(r)} B_{i-3\cdot 2^r}^{(r)} + (-1)^r B_{i-2^{r+1}}^{(0)} B_{i-3\cdot 2^r}^{(r)} Q_i^{(r)} - Q_{i-2^{r+1}}^{(r)} Q_i^{(r)}. \tag{6.25}$$

We now proceed to prove by induction that the factorizations (6.17) and (6.18) can be made. This is true for $r = 0$ since by (6.9) through (6.12) plus (6.22) and (6.23) we have the left sides of (6.17) and (6.18) are zero. Hence

$$R_i^{(0)} = 0 \tag{6.26}$$

and

$$S_i^{(0)} = 0. \tag{6.27}$$

We shall assume the factorization is true for r and show it is true for $r + 1$. Using (6.24) and (6.25) we obtain

$$\begin{aligned} &\alpha_{i+2^{r+2}}^{(r+2)} \gamma_i^{(r+2)} I - P_i^{(r+1)} Q_{i+2^{r+2}}^{(r+1)} \\ &= \alpha_{i+2^{r+2}}^{(r+2)} \gamma_i^{(r+2)} I - [R_i^{(r)} B_{i+3\cdot 2^r}^{(r)} + (-1)^r B_{i+2^{r+1}}^{(0)} B_{i+3\cdot 2^r}^{(r)} P_i^{(r)} - P_{i+2^{r+1}}^{(r)} P_i^{(r)}] \\ &\quad \cdot [S_{i+2^{r+2}}^{(r)} B_{i+2^r}^{(r)} + (-1)^r B_{i+2^{r+1}}^{(0)} B_{i+2^r}^{(r)} Q_{i+2^{r+2}}^{(r)} - Q_{i+2^{r+1}}^{(r)} Q_{i+2^{r+2}}^{(r)}] \end{aligned} \tag{6.28}$$

or

$$\begin{aligned} &\alpha_{i+2^{r+2}}^{(r+2)} \gamma_i^{(r+2)} I - P_i^{(r+1)} Q_{i+2^{r+2}}^{(r+1)} \\ &\quad = \alpha_{i+2^{r+2}}^{(r+2)} \gamma_i^{(r+2)} I - P_{i+2^{r+1}}^{(r)} P_i^{(r)} Q_{i+2^{r+1}}^{(r)} Q_{i+2^{r+2}}^{(r)} \\ &\qquad - B_{i+3\cdot 2^r}^{(r)} R_i^{(r)} B_{i+2^r}^{(r)} S_{i+2^{r+2}}^{(r)} \\ &\qquad + S_{i+2^{r+2}}^{(r)} P_i^{(r)} (-1)^{r+1} B_{i+2^{r+1}}^{(0)} B_{i+2^r}^{(r)} B_{i+3\cdot 2^r}^{(r)} \end{aligned} \tag{6.29}$$

(*cont.*)

$$
\begin{aligned}
&+ S^{(r)}_{i+2^{r+2}} P^{(r)}_i P^{(r)}_{i+2^{r+1}} B^{(r)}_{i+2^r} \\
&+ R^{(r)}_i Q^{(r)}_{i+2^{r+2}} Q^{(r)}_{i+2^{r+1}} B^{(r)}_{i+3\cdot 2^r} \\
&+ R^{(r)}_i Q^{(r)}_{i+2^{r+2}} (-1)^{r+1} B^{(0)}_{i+2^{r+1}} B^{(r)}_{i+2^r} B^{(r)}_{i+3\cdot 2^r} \\
&+ (-1)^r B^{(0)}_{i+2^{r+1}} P^{(r)}_i Q^{(r)}_{i+2^{r+2}} (-1)^{r+1} B^{(0)}_{i+2^{r+1}} B^{(r)}_{i+2^r} B^{(r)}_{i+3\cdot 2^r} \\
&+ (-1)^r B^{(0)}_{i+2^{r+1}} P^{(r)}_i Q^{(r)}_{i+2^{r+2}} Q^{(r)}_{i+2^{r+1}} B^{(r)}_{i+3\cdot 2^r} \\
&+ (-1)^r B^{(0)}_{i+2^{r+1}} P^{(r)}_i Q^{(r)}_{i+2^{r+2}} P^{(r)}_{i+2^{r+1}} B^{(r)}_{i+2^r}.
\end{aligned}
$$

Using (6.17), (6.18), (6.22) and (6.23) we can determine that the sum of the first two terms on the right side of (6.29) is

$$
S^{(r)}_{i+2^{r+2}} P^{(r)}_i Q^{(r)}_{i+2^{r+1}} B^{(r)}_{i+3\cdot 2^r} + R^{(r)}_i Q^{(r)}_{i+2^{r+2}} P^{(r)}_{i+2^{r+1}} B^{(r)}_{i+2^r}, \tag{6.30}
$$

and since

$$
B^{(r+1)}_{i+2^{r+1}} = P^{(r)}_{i+2^{r+1}} B^{(r)}_{i+2^r} + (-1)^{r+1} B^{(0)}_{i+2^{r+1}} B^{(r)}_{i+2^r} B^{(r)}_{i+3\cdot 2^r} + Q^{(r)}_{i+2^{r+1}} B^{(r)}_{i+3\cdot 2^r} \tag{6.31}
$$

we see that (6.29) has the form

$$
\alpha^{(r+2)}_{i+2^{r+2}} \gamma^{(r+2)}_i I - P^{(r+1)}_i Q^{(r+1)}_{i+2^{r+2}} = R^{(r+1)}_i B^{(r+1)}_{i+2^{r+1}}, \tag{6.32}
$$

where

$$
R^{(r+1)}_i = R^{(r)}_i Q^{(r)}_{i+2^{r+2}} + (-1)^r B^{(0)}_{i+2^{r+1}} Q^{(r)}_{i+2^{r+2}} P^{(r)}_i + S^{(r)}_{i+2^{r+2}} P^{(r)}_i. \tag{6.33}
$$

A similar proof of the factorization (6.18) yields

$$
S^{(r+1)}_i = S^{(r)}_i P^{(r)}_{i-2^{r+2}} + (-1)^r B^{(0)}_{i-2^{r+1}} P^{(r)}_{i-2^{r+2}} Q^{(r)}_i + R^{(r)}_{i-2^{r+2}} Q^{(r)}_i. \tag{6.34}
$$

This completes the proof of the theorem. The $B^{(r)}_i$ polynomials given by (2.40) can also be computed by (6.21) using (6.24), (6.25), (6.33) and (6.34) together with initial polynomials (2.20), (6.11), (6.12), (6.26) and (6.27). From these recurrence relations we can determine that for $r > 0$ the order of the $B^{(r)}_i$ polynomials is $2^{r+1} - 1$, the order of $P^{(r)}_i$, $Q^{(r)}_i$ is $2^{r+1} - 2$, and the order of $R^{(r)}_i$, $S^{(r)}_i$ is $2^{r+1} - 3$.

In general, $B^{(r-1)}_{i+2^{r-1}} B^{(r-1)}_{i-2^{r-1}}$ is not the greatest common divisor of $\hat{A}^{(r+1)}_i$, $\hat{B}^{(r+1)}_i$ and $\hat{C}^{(r+1)}_i$. If the linear system results from the discretization of Poisson's equation in Cartesian coordinates, then additional common divisors exist. However, numerical experiments have shown that for Poisson's equation in spherical coordinates $B^{(r-1)}_{i+2^{r-1}} B^{(r-1)}_{i-2^{r-1}}$ is the greatest common divisor.

Acknowledgments. I wish to thank Professor G. H. Golub for his helpful remarks, and Dr. R. A. Sweet for many helpful ideas and discussions.

REFERENCES

[1] O. Buneman, *A compact non-iterative Poisson solver*, Rep. 294, Stanford University Institute for Plasma Research, Stanford, California, 1969.

[2] B. L. Buzbee, F. W. Dorr, J. A. George and G. H. Golub, *The direct solution of the discrete Poisson equation on irregular regions*, this Journal, 8 (1971), pp. 722–736.

[3] ———, G. H. Golub and C. W. Nielson, *On direct methods for solving Poisson's equations*, this Journal, 7 (1970), pp. 627–656.

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

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

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