SIAM J. NUMER. ANAL.
Vol. 14, No. 4, September 1977

A CYCLIC REDUCTION ALGORITHM FOR SOLVING BLOCK TRIDIAGONAL SYSTEMS OF ARBITRARY DIMENSION*

ROLAND A. SWEET†

Abstract. A generalization of the Buneman variant of cyclic odd–even reduction algorithm for solving finite difference approximations to Poisson's equations is presented. This generalization places no restriction on the block size, n, of the system and computes the solution in $O(n^2 \log_2 n)$ operations.

1. Introduction.

Block tridiagonal linear systems of the form

$$\begin{bmatrix} A & -I & & & \\ -I & A & -I & & \\ & \cdot & \cdot & \cdot & \\ & & -I & A & -I \\ & & & -I & A \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{n-1} \\ v_n \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_{n-1} \\ f_n \end{bmatrix} \tag{1}$$

where each block is an $m \times m$ matrix and each vector v_j and f_j is of length m, result from finite difference approximations to certain elliptic partial differential equations, specifically, equations of the form

$$a(x)\frac{\partial^2 u}{\partial x^2}+b(x)\frac{\partial u}{\partial x}+c(x)u+\frac{\partial^2 u}{\partial y^2}=f(x, y), \qquad a \leqq x \leqq b, \qquad c \leqq y \leqq d,$$

with the solution specified on the boundaries. The rapid solution of such systems of equations has received much attention recently [2], [3], [5], [6], [7], [9], [11], [12]. Two algorithms of importance are the Buneman variant of cyclic reduction [3] and the matrix decomposition method based on the use of the fast Fourier transform (FFT) [3]. Both of these algorithms compute the solution in about $5mn \log_2 n$ operations when n has the form $n = 2^p - 1$ [4].

The Buneman algorithm as originally proposed restricted n to be of the form $n = 2^p - 1$. This author generalized that algorithm to handle any n which was a composite of small primes. However, actual experience demonstrated that efficiency decreased drastically as the primes increased in size. In fact, a computer program based on a prime decomposition of n of the form $n = 2^p 3^q 5^r - 1$ showed that there was an increase of nearly 50% in the execution time between the cases $n = 2^7 - 1 = 127$ and $n = 5^3 - 1 = 124$. Furthermore, to accommodate the more general form of n, an algorithm had to be used which, even when $n = 2^p - 1$, was much less efficient than the Buneman algorithm. It seemed desirable to attempt a generalization of the Buneman algorithm which did not depend on the prime factorization of n at all. Such an algorithm is described in the remainder of this paper.

* Received by the editors April 23, 1976.

† Natural and Physical Sciences Division, University of Colarado at Denver, Denver, Colorado 80202 and National Center for Atmospheric research (sponsored by the National Science Foundation), Boulder, Colorado 80307.

An example is presented in § 2 which illuminates the basic idea of the algorithm. In § 3, we give an algorithm for the solution of system (1) which places no restriction on the number n. In §§ 4 and 5, we describe the modifications to the algorithm which are necessary if the linear system of equations arises from finite difference approximations to the above elliptic equation with Neumann or periodic boundary conditions. The algorithm presented here is a strict generalization of the Buneman algorithm in the sense that if $n=2^p-1$, the algorithm reduces to the Buneman algorithm. Computer programs implementing this algorithm have been developed and will be available in the general elliptic equation package [10] from the National Center for Atmospheric Research.

All computations for this paper were performed on the NCAR Control Data 7600.

2. An example. To motivate the development of the general algorithm given in the next section, we consider solving the block tridiagonal system

$$\begin{bmatrix} A & -I & & & & \\ -I & A & -I & & & \\ & -I & A & -I & & \\ & & -I & A & -I & \\ & & & -I & A & -I \\ & & & & -I & A \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \end{bmatrix}. \tag{2}$$

We perform the following linear combinations: multiply the second row by A and add to it the first and third rows, multiply the fourth row by A and add to it the third and fifth rows, and multiply the sixth row by A and add to it the fifth row. The result is the new system

$$\begin{bmatrix} A^{(1)} & -I & \\ -I & A^{(1)} & -I \\ & -I & B^{(1)} \end{bmatrix} \begin{bmatrix} x_2 \\ x_4 \\ x_6 \end{bmatrix} = \begin{bmatrix} Af_2+f_1+f_3 \\ Af_4+f_3+f_5 \\ Af_6+f_5 \end{bmatrix} = \begin{bmatrix} f_2^{(1)} \\ f_4^{(1)} \\ f_6^{(1)} \end{bmatrix}, \tag{3}$$

where $A^{(1)}=A^2-2I$ and $B^{(1)}=A^2-I$. (We are ignoring for this example the treatment of the right side.) We note that since the block size of the system (2) was an even number, the diagonal blocks of system (3) are not identical. If we try to continue the reduction scheme as above, we must introduce inverse matrices.

Suppose we multiply the second row of system (3) by $A^{(1)}$, add to it the first row and $(B^{(1)})^{-1}A^{(1)}$ times the third row; we obtain the single equation

$$\begin{aligned} Dx_4 &= [(A^{(1)})^2-I-(B^{(1)})^{-1}A^{(1)}]x_4 \\ &= A^{(1)}f_4^{(1)}+f_2^{(1)}+(B^{(1)})^{-1}A^{(1)}f_6^{(1)} \\ &= f_4^{(2)}. \end{aligned} \tag{4}$$

The matrix D multiplying x_4 is no longer a polynomial in the original matrix A but may be expressed as a rational function in A since

$$D=(B^{(1)})^{-1}[A^{(1)}(A^{(1)}B^{(1)}-I)-B^{(1)}]=(B^{(1)})^{-1}B^{(2)}. \tag{5}$$

Furthermore,

$$B^{(1)}=(A-I)(A+I) \tag{6a}$$

and, as will be shown in the next section

$$B^{(2)}=\prod_{i=1}^{6}\left(A-2\cos\frac{i\pi}{7}I\right). \tag{6b}$$

Equation (4) becomes

$$B^{(2)}x_4=B^{(1)}f_4^{(2)},$$

which may be solved readily using the known factorizations given in (6).

The recognition that D in (5) could be written as the quotient of two polynomials in A with known roots motivated the attempt to devise a cyclic reduction scheme for arbitrary block sizes by allowing the last diagonal block in any stage of the algorithm to be a rational function in A rather than a polynomial in A.

3. The general algorithm. We shall now describe the general algorithm for the solution by cyclic reduction of the original system of equations given in (1) where now no restriction is placed on the block size n. Suppose that at the rth step of the reduction algorithm we have the system

$$\begin{bmatrix} A^{(r)} & -I & & & & \\ -I & A^{(r)} & -I & & & \\ \cdot & \cdot & \cdot & & & \\ & -I & A^{(r)} & -I & & \\ & & -I & A^{(r)} & -I & \\ & & & -I & B^{(r)}(C^{(r)})^{-1} \end{bmatrix}\begin{bmatrix} v_h \\ v_{2h} \\ \vdots \\ v_{J_r-2h} \\ v_{J_r-h} \\ v_{J_r}\end{bmatrix} = \begin{bmatrix} A^{(r)}p_h^{(r)}+q_h^{(r)} \\ A^{(r)}p_{2h}^{(r)}+q_{2h}^{(r)} \\ \vdots \\ A^{(r)}p_{J_r-2h}^{(r)}+q_{J_r-2h}^{(r)} \\ A^{(r)}p_{J_r-h}^{(r)}+q_{J_r-h}^{(r)} \\ B^{(r)}(C^{(r)})^{-1}p_{J_r}^{(r)}+q_{J_r}^{(r)} \end{bmatrix}, \tag{7}$$

where $h=2^r$, $J_r=n_rh$, and n_r is the block size. Initially, we have $A^{(0)}=B^{(0)}=A$, $C^{(0)}=I$, $J_0=n_0=n$ and

$$p_j^{(0)}=0, \qquad q_j^{(0)}=f_j, \qquad j=1,2,\cdots,n.$$

Given the system (7) we apply the usual reduction algorithm at first: multiply equation $2j$ by $A^{(r)}$ and add to it equations $2j-1$ and $2j+1$, thus eliminating all unknowns whose index is not a multiple of $2h$. Now, to complete the description of this step of the algorithm we must distinguish two cases.

Case I. n_r an even number. In this case we see that v_{J_r} is an unknown which will not be eliminated. To obtain a new equation for v_{J_r} we multiply the last equation by $A^{(r)}$ and add the next to the last equation to it.

Case II. n_r is an odd number. In this case v_{J_r} is to be eliminated at this step while v_{J_r-h} will not be eliminated. The new equation for v_{J_r-h} is obtained by multiplying the next to the last equation by $A^{(r)}$ and adding to it the preceding equation plus $(B^{(r)})^{-1}C^{(r)}A^{(r)}$ times the last equation.

The above elimination yields the reduced system of equations

$$(8)\quad \begin{bmatrix} A^{(r+1)} & -I & & & \\ -I & A^{(r+1)} & -I & & \\ \cdot & \cdot & \cdot & & \\ & -I & A^{(r+1)} & -I & \\ & & -I & B^{(r+1)}(C^{(r+1)})^{-1} & \end{bmatrix}\begin{bmatrix} v_{2h} \\ v_{4h} \\ \vdots \\ v_{J_{r+1}-2h} \\ v_{J_{r+1}} \end{bmatrix} = \begin{bmatrix} A^{(r+1)}p_{2h}^{(r+1)}+q_{2h}^{(r+1)} \\ A^{(r+1)}p_{4h}^{(r+1)}+q_{4h}^{(r+1)} \\ \vdots \\ A^{(r+1)}p_{J_{r+1}-2h}^{(r+1)}+q_{J_{r+1}-2h}^{(r+1)} \\ B^{(r+1)}(C^{(r+1)})^{-1}p_{J_{r+1}}^{(r+1)}+q_{J_{r+1}}^{(r+1)} \end{bmatrix},$$

where

$$(9)\qquad A^{(r+1)}=(A^{(r)})^2-2I,$$

$$(10)\qquad \begin{aligned} p_j^{(r+1)}&=p_j^{(r)}+(A^{(r)})^{-1}(q_j^{(r)}+p_{j-h}^{(r)}+p_{j+h}^{(r)}),\\ q_j^{(r+1)}&=q_{j-h}^{(r)}+q_{j+h}^{(r)}+2p_j^{(r+1)}, \end{aligned}\qquad j=2h,4h,\cdots,J_{r+1}-2h$$

and, in Case I,

$$(11)\qquad B^{(r+1)}=A^{(r)}B^{(r)}-C^{(r)},\qquad C^{(r+1)}=C^{(r)},$$

$$(12)\qquad p_{J_{r+1}}^{(r+1)}=p_{J_r}^{(r)}+(B^{(r)})^{-1}C^{(r)}(q_{J_r}^{(r)}+p_{J_r-h}^{(r)}),$$

$$(13)\qquad q_{J_{r+1}}^{(r+1)}=q_{J_r-h}^{(r)}+p_{J_{r+1}}^{(r+1)},\qquad J_{r+1}=J_r,$$

while in Case II,

$$(14)\qquad B^{(r+1)}=A^{(r)}(A^{(r)}B^{(r)}-C^{(r)})-B^{(r)},\qquad C^{(r+1)}=B^{(r)},$$

$$(15)\qquad p_{J_{r+1}}^{(r+1)}=p_{J_r-h}^{(r)}+(A^{(r)})^{-1}(q_{J_r-h}^{(r)}+p_{J_r}^{(r)}+p_{J_r-2h}^{(r)}),$$

$$(16)\qquad q_{J_{r+1}}^{(r+1)}=q_{J_r-2h}^{(r)}+p_{J_{r+1}}^{(r+1)}+(B^{(r)})^{-1}C^{(r)}A^{(r)}(q_{J_r}^{(r)}+p_{J_{r+1}}^{(r+1)})$$

$$(17)\qquad J_{r+1}=J_r-h.$$

Using this general reduction scheme the original system of equations (1) may be reduced at step $r=s$ to the single equation

$$(18)\qquad B^{(s)}(C^{(s)})^{-1}v_{J_s}=B^{(s)}(C^{(s)})^{-1}p_{J_s}^{(s)}+q_{J_s}^{(s)}$$

which, as we will show later, may be solved readily. The remaining unknowns are then computed by the usual back substitution process.

Considering (9), (11), and (14), we see that for each r, $A^{(r)}$, $B^{(r)}$ and $C^{(r)}$ are polynomials, say Θ_r, P_r, and Q_r, in the original matrix A. From [3], we already know that $\frac{1}{2}\Theta_r$ is the Chebyshev polynomial of the first kind, T_2r. Suppose that P_r has degree k_r and Q_r has degree l_r. Then, from (11) and (14) we see that

$$k_{r+1}=\begin{cases} k_r+2^r & \text{in Case I,} \\ k_r+2^{r+1} & \text{in Case II,} \end{cases} \tag{19}$$

and

$$l_{r+1}=\begin{cases} l_r & \text{in Case I,} \\ k_r & \text{in Case II.} \end{cases} \tag{20}$$

The usefulness of this algorithm lies in the fact that we can show that P_r and Q_r are, in fact, Chebyshev polynomials of the second kind. First, let us list some properties of these Chebyshev polynomials (see [1, Chap. 22]) in the following lemma.

LEMMA 1. *Let T_m and U_m denote, respectively, the Chebyshev polynomials of the first and second kind. Then*

P1: $T_m(\cos\theta)=\cos m\theta, \qquad U_m(\cos\theta)=\dfrac{\sin(m+1)\theta}{\sin\theta},$

P2: $T_m(x)=\prod_{i=1}^{m}\left(2x-2\cos\dfrac{(2i-1)\pi}{2m}\right), \qquad U_m(x)=\prod_{i=1}^{m}\left(2x-2\cos\dfrac{i\pi}{m+1}\right),$

P3: $2T_mU_n=U_{n+m}+U_{n-m}, \qquad$ *if* $n\geqq m$,

P4: $2T_nT_m=T_{n+m}+T_{n-m}$,

P5: $U_{2m-1}=2T_mU_{m-1}$.

LEMMA 2. $A^{(r)}=2T_{2^r}(\frac{1}{2}A), \quad B^{(r)}=U_{k_r}(\frac{1}{2}A), \quad C^{(r)}=U_{l_r}(\frac{1}{2}A),$

where

$$k_r=2^r+l_r \quad \textit{and} \quad 0\leqq l_r\leqq 2^r-1.$$

Proof. The proof of this lemma is accomplished by induction on r. Throughout, we shall make the substitution

$$A=2\cos\theta.$$

For $r=0$, $B^{(0)}=A=U_1$, $C^{(0)}=I=U_0$, so with $k_0=1$ and $l_0=0$, the lemma is verified. We now assume that the lemma is true for the integer r and verify its truth for the integer $r+1$. We do this by considering the two cases in the reduction scheme.

Case I: From equation (11), $P_{r+1}=2T_{2^r}U_{k_r}-U_{l_r}$ and $Q_{r+1}=Q_r$. So, using P3 of lemma 1, we get

$$P_{r+1}=U_{k_r+2^r}=U_{l_r+2^{r+1}} \quad \text{and} \quad Q_{r+1}=U_{l_r}.$$

Hence, with $l_{r+1} = l_r < 2^{r+1} - 1$ and $k_{r+1} = 2^{r+1} + l_{r+1}$, the lemma is verified for this case.

Case II. From equation (14), we get

$$P_{r+1} = 2T_{2^r}(2T_{2^r}U_{k_r} - U_{l_r}) - U_{k_r} = 2T_{2^r}U_{2^r+k_r} - U_{k_r} = U_{k_r+2^{r+1}}$$

after two applications of identity P3 and $Q_{r+1} = U_{k^r}$. Hence, with $l_{r+1} = k_r \leqq 2^r + 2^r - 1 = 2^{r+1} - 1$ and $k_{r+1} = 2^{r+1} + l_{r+1}$ the lemma is verified completely.

As the algorithm is now stated, if $n = 2^p - 1$, we do not have the Buneman variant of cyclic reduction since the last block equation in step r involves the rational function $B^{(r)}(C^{(r)})^{-1}$ and not the matrix $A^{(r)}$. However, let us assume that $n_0, n_1, \cdots, n_s$ are all odd. Then, using (19) we see that $k_r = 2^{r+1} - 1$, and, hence, $l_r = k_r - 2^r = 2^r - 1$. Using P5 of Lemma 1, we see that

$$B^{(r)} = P_r = U_{2^{r+1}-1} = 2T_{2^r}U_{2^r-1} = A^{(r)}C^{(r)}, \qquad r = 1, 2, \cdots, s,$$

so that, in fact, $B^{(r)}(C^{(r)})^{-1} = A^{(r)}$. Hence, the algorithm is identical to the Buneman variant for the first s steps.

Now suppose that at some step r, k_r (and hence, l_r) is an even number. We may ask: do the polynomials P_r and Q_r have any common roots? From Lemma 1, P1, the roots of $U_m(\cos\theta)$ are given by $\cos(i\pi/(m+1))$, $i = 1, 2, \cdots, m$, so the question is equivalent to the question: for a given even number $k_r = l_r + 2^r$ (with $r > 0$), are there integers i and j such that $1 \leqq j \leqq k_r$, $1 \leqq i \leqq l_r$, and

$$\frac{j}{k_r+1} = \frac{i}{l_r+1}?$$

Assuming this equation holds, we get the condition

$$(j-i)(k_r+1) = j2^r.$$

Now the number $k_r + 1$ is odd and divides the left side of this equation; hence, it divides the right side. But since $k_r + 1$ is odd, it must divide j which is impossible since $1 \leqq j \leqq k_r$. Hence, there are no roots common to both P_r and Q_r.

Using P2 of Lemma 1, we can rewrite (18) as

$$B^{(s)}(v_{J_s} - p_{J_s}^{(s)}) = C^{(s)}q_{J_s}^{(s)} \tag{21}$$

or

$$\prod_{i=1}^{k_s}(A - \lambda_i^{(s)}I)y = \prod_{i=1}^{l_s}(A - \mu_i^{(s)}I)q_{J_s}^{(s)}, \tag{22}$$

where

$$\lambda_i^{(s)} = 2\cos\frac{i}{k_s+1}\pi, \qquad \mu_i^{(s)} = 2\cos\frac{i}{l_s+1}\pi,$$

and $y = v_{J_s} - p_{J_s}^{(s)}$. Equation (22) may be further simplified by using a technique of Swarztrauber [8, p. 1143] which obviates matrix multiplications of the form

$(A-\mu I)q$. The procedure is summarized as:

(23a) 1. Set $z_0 = q_{J_s}^{(s)}$.

2. For $i = 1, 2, \cdots, l_s$, solve the linear system

$$(A-\lambda_i^{(s)}I)\,\delta z_i = (\lambda_i^{(s)} - \mu_i^{(s)})z_{i-1}, \tag{23b}$$

$$z_i = z_{i-1} + \delta z_i.$$

3. For $i = l_{s+1}, l_{s+2}, \cdots, k_s$, solve the linear system

$$(A-\lambda_i^{(s)}I)z_i = z_{i-1}. \tag{23c}$$

(23d) 4. $v_{J_s} = p_{J_s}^{(s)} + z_{k_s}$.

Actually, the factors $(A-\mu_i^{(s)}I)$ on the right side of (22) are not used sequentially as indicated in step 2 above, but rather, are interspersed in such a way that $\mu_i^{(s)}$ is as close as possible to one of the $\lambda_j^{(s)}$. This interspersing reduces considerably the accumulation of rounding errors.

The computation of the last term of (16) is accomplished by an algorithm similar to (23); however, one must take care to merge the roots of Θ_r and Q_r for maximum accuracy.

Furthermore, as in [3], the algorithm may be modified using (10), (12), and (13) to eliminate the $p_j^{(r)}$ vectors except those defined by (15). These must be stored for possible later use. However, there are at most $n \log_2 n$ such vectors so the increased storage requirements are minimal.

A few remarks about the efficiency of this algorithm can be made. By efficiency we mean the number of tridiagonal systems which must be solved as this determines the dominant term in the operation count.

1. The algorithm reduces the block tridiagonal system (1) to a single matrix equation in $[\log_2 n]$ ($[x]$ is greatest integer function) steps. From Lemma 2, we see that $k_r < 2^{r+1}$; hence, the operation count is proportional to $nm \log_2 n$. Figure 1 demonstrates the dependence of the execution time T on the quantity $n^2 \log_2 n$ (in this case, we have taken $m = n$).

2. The least amount of work corresponds to the case $n = 2^{k+1}-1$, i.e., the case where the sequence k_r grows the slowest. The most amount of work occurs when n_0 is even and n_r is odd thereafter, i.e., when $n = 2^{k+1}-2$, as this causes the most rapid growth of the sequence k_r. A quick count shows that the relative increase in the amount of work between these two values of n is about $2/k$ which is exhibited in Fig. 1.

4. Modifications of the algorithm for Neumann boundary conditions.

The linear system of equations to be solved in this case is

$$\begin{bmatrix} A & -2I & & & \\ -I & A & -I & & \\ & \cdot & \cdot & \cdot & \\ & & -I & A & -I \\ & & & -I & \frac{1}{2}A \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{n-1} \\ v_n \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_{n-1} \\ f_n \end{bmatrix}. \tag{24}$$

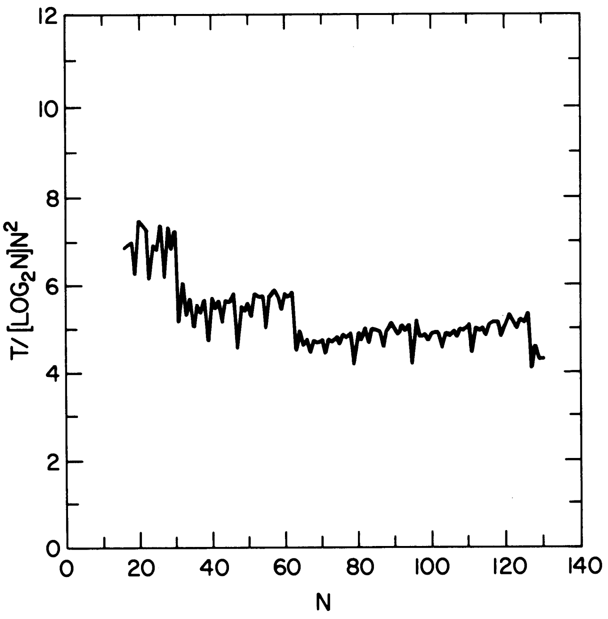

FIG. 1. *Relative computation time for solving equation (1) with $m=n$ and T measured in microseconds*

(Note that we have multiplied the last block equation by $\frac{1}{2}$ for later convenience.) The diagonal matrix in the last equation can be written in the form

$$\tfrac{1}{2}A = A(2I)^{-1} = 2T_1(\tfrac{1}{2}A)(2T_0(\tfrac{1}{2}A))^{-1} = B^{(0)}(C^{(0)})^{-1}.$$

We now proceed with the reduction scheme as described in [3] by taking linear combinations which eliminate the unknowns $v_2, v_4, \cdots$ with even indices. As in the previous section we must distinguish between the two cases—n an even number and n an odd number. We can describe the algorithm as follows. Assume that at step r of the reduction scheme we have the linear system

$$\begin{bmatrix} A^{(r)} & -2I & & & \\ -I & A^{(r)} & -I & & \\ & \cdot & \cdot & \cdot & \\ & -I & A^{(r)} & -I & \\ & & -I & B^{(r)}(C^{(r)})^{-1} \end{bmatrix} \begin{bmatrix} v_1 \\ v_{1+h} \\ \vdots \\ v_{J_r-h} \\ v_{J_r} \end{bmatrix} \tag{25}$$
$$= \begin{bmatrix} A^{(r)}p_1^{(r)}+q_1^{(r)} \\ A^{(r)}p_{1+h}^{(r)}+q_{1+h}^{(r)} \\ \vdots \\ A^{(r)}p_{J_r-h}^{(r)}+q_{J_r-h}^{(r)} \\ B^{(r)}(C^{(r)})^{-1}p_{J_r}^{(r)}+q_{J_r}^{(r)} \end{bmatrix}$$

where $h=1+2^r$ and $J_r=1+n_rh$. (Initially for $r=0$, we have $A^{(0)}=B^{(0)}=A$, $C^{(0)}=2I$, $n_0=n-1$, $p_j^{(0)}=0$, $q_j^{(0)}=f_j$, $j=1,2,\cdots,n$.) The reduction to step $r+1$ is accomplished by eliminating all variables whose index is not of the form $1+2jh$ using the usual reduction given in [3] except at the end where we must distinguish between n_r even and odd.

Case I. n_r is an even number. In this case the variable v_{J_r} is retained and v_{J_r-h} is eliminated. This is accomplished by adding $A^{(r)}$ times the last equation to the preceding equation.

Case II. n_r is an odd number. In this case the variable v_{J_r-h} is retained and v_{J_r} is eliminated. This is accomplished by multiplying the next to the last equation by $A^{(r)}$ and adding to it the preceding equation plus $(B^{(r)})^{-1}C^{(r)}A^{(r)}$ times the last equation.

But this reduction at the end is identical to that described in the previous section. Hence, the recurrence relations given in (11)–(17) apply in this case, too. The relationship between the matrices $B^{(r)}$ and $C^{(r)}$ in this case is given in the following lemma.

LEMMA 3. *In the case of Neumann boundary conditions, we have*

$$A^{(r)}=2T_{2^r}(\tfrac{1}{2}A),\quad B^{(r)}=2T_{k_r}(\tfrac{1}{2}A),\quad C^{(r)}=2T_{l_r}(\tfrac{1}{2}A),$$

where

$$k_r=2^r+l_r,\qquad 0\leqq l_r\leqq 2^r-1.$$

Proof. The proof of this lemma follows exactly as the proof of Lemma 2 using identity P4 of Lemma 1.

This algorithm coincides with that given in [3] when $n=2^{p+1}-1$. For, in this case $n_r=2^{p+1-r}$, an even number, and hence, the reduction for the last unknown at each step is accomplished using Case I. Therefore, $l_r=0$ for each r, so $k_r=2^r$, and $B^{(r)}=A^{(r)}$.

The retention of the unknown v_1 at each step causes an inefficiency near the end of the reduction phase of this algorithm. To describe this suppose that at step s we have reduced from four unknowns to the system

$$\begin{bmatrix} A^{(s)} & -2I \\ -I & B^{(s)}(C^{(s)})^{-1}\end{bmatrix}\begin{bmatrix} v_1 \\ v_{J_s}\end{bmatrix} = \begin{bmatrix} A^{(s)}p_1^{(s)}+q_1^{(s)} \\ B^{(s)}(C^{(s)})^{-1}p_{J_s}^{(s)}+q_{J_s}^{(s)}\end{bmatrix}. \tag{26}$$

We eliminate v_1 by multiplying the second equation by $A^{(s)}$ and adding it to the first equation. The resulting equation for v_{J_s} is

$$M_1(C^{(s)})^{-1}v_{J_s}=M_1(C^{(s)})^{-1}p_{J_s}^{(s+1)}+q_{J_s}^{(s+1)} \tag{27}$$

where

$$M_1=U_{k_s-1}(\tfrac{1}{2}A)U_{2^s-1}(\tfrac{1}{2}A)(A+2I)(A-2I), \tag{28}$$

(the last factor may, depending on the matrix A, be singular),

$$p_{J_s}^{(s+1)}=p_{J_s}^{(s)}+(B^{(s)})^{-1}C^{(s)}(q_{J_s}^{(s)}+p_1^{(s)})$$
$$q_{J_s}^{(s+1)}=q_1^{(s+1)}+2p_{J_s}^{(s)}+2p_{J_s}^{(s+1)}.$$

Equation (27) may be readily solved using the factorization (28) and the algorithm given in (23). The back substitution phase of the algorithms begins at this point.

Now, suppose that at step s we have reduced from either five or six unknowns to the system

$$(29)\quad \begin{bmatrix} A^{(s)} & -2I & \\ -I & A^{(s)} & -I \\ & -I & B^{(s)}(C^{(s)})^{-1} \end{bmatrix} \begin{bmatrix} v_1 \\ v_{1+h} \\ v_{J_s} \end{bmatrix} = \begin{bmatrix} A^{(s)}p_1^{(s)}+q_1^{(s)} \\ A^{(s)}p_{1+h}^{(s)}+q_{1+h}^{(s)} \\ B^{(s)}(C^{(s)})^{-1}p_{J_s}^{(s)}+q_{J_s}^{(s)} \end{bmatrix},$$

where $h=2^s$. We could perform one more reduction step to obtain a system involving v_1 and v_{J_s} and then solve this system as described above. However, it is more efficient (by up to 50%) to reduce the system (29) to a single equation involving v_{1+h}. This is accomplished by multiplying the second equation by $A^{(s)}$ and adding to it the first equation plus $A^{(s)}(B^{(s)})^{-1}C^{(s)}$ times the third equation. The resulting equation in this case is

$$\begin{aligned} M_2(B^{(s)})^{-1}v_{1+h} = M_2(B^{(s)})^{-1}p_{1+h}^{(s)} &+ A^{(s)}(q_{1+h}^{(s)}+p_1^{(s)}+p_{J_s}^{(s)}) \\ &+(B^{(s)})^{-1}A^{(s)}C^{(s)}(q_{J_s}^{(s)}+p_{1+h}^{(s)})+q_1^{(s)}, \end{aligned}$$

or

$$(30)\quad \begin{aligned} v_{1+h}-p_{1+h}^{(s)} = M_2^{-1}[&A^{(s)}B^{(s)}(q_{1+h}^{(s)}+p_1^{(s)}+p_{J_s}^{(s)}) \\ &+A^{(s)}C^{(s)}(q_{J_s}^{(s)}+p_{1+h}^{(s)})+B^{(s)}q_1^{(s)}], \end{aligned}$$

where

$$M_2 = U_{k_s+2^s-1}(\tfrac{1}{2}A)U_{2^s-1}(\tfrac{1}{2}A)(A+2I)(A-2I).$$

The use of M_2^{-1} in (30) is, strictly speaking, not correct since it may be true that the factor $(A-2I)$ of M_2 is singular. But, assuming that the proper orthogonality condition has been satisfied initially for the system (24), a nonunique solution will exist. Computing the right side of (30) may be done efficiently by viewing the solution as the inversion of one matrix, M_2, with three different right sides (the three terms in the brackets). Specifically, we have written a single subroutine which solves the three systems

$$(31a)\quad M_2(A^{(s)}B^{(s)})^{-1}x_1 = q_{1+h}^{(s)}+p_1^{(s)}+p_{J_s}^{(s)}$$

$$(31b)\quad M_2(A^{(s)}C^{(s)})^{-1}x_2 = q_{J_s}^{(s)}+p_{1+h}^{(s)}$$

$$(31c)\quad M_2(B^{(s)})^{-1}x_3 = q_1^{(s)}$$

simultaneously using the algorithm given in (23). The savings here results from factoring each matrix $A-\lambda I$ in equations (23b), (23c) into its LU-decomposition only once and using this for each of the linear systems (31a), (31b), (31c).

Once (30) has been solved for v_{1+h}, the back substitution phase begins.

The algorithm presented here computes the solution of the system (24) in a time which is proportional to $nm[\log_2 n]$ and, as before, the fastest time corresponds to the case $n=2^{p+1}+1$. Figure 2 demonstrates the dependence of the execution time on the quantity $n^2[\log_2 n]$ (where we have taken $m=n$). Note the sharp decrease in the graph for the values of $n=33$, 65, and 129.

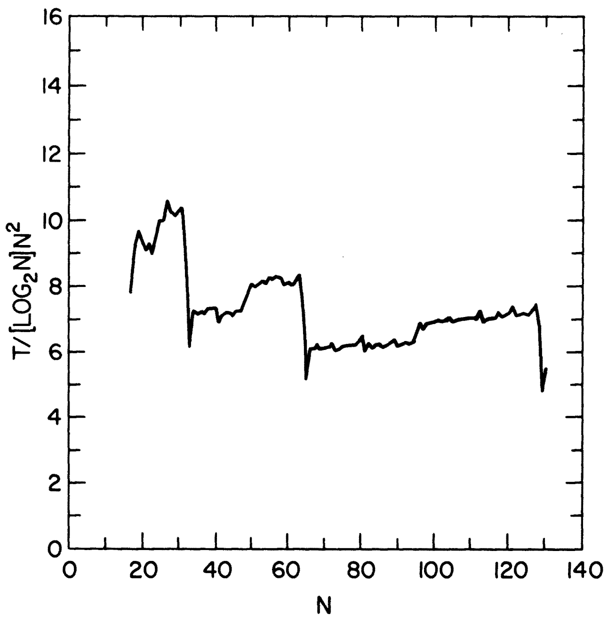

FIG. 2. *Relative computation time for solving equation (24) with $m=n$ and T measured in microseconds*

5. An algorithm for periodic boundary conditions. In this case we must solve the linear system of equations

$$\begin{bmatrix} A & -I & & & -I \\ -I & A & -I & & \\ & \cdot & \cdot & \cdot & \\ & & -I & A & -I \\ -I & & & -I & A \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{n-1} \\ v_n \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_{n-1} \\ f_n \end{bmatrix}. \tag{32}$$

We will show that the solution of this system can be obtained as a solution of two systems of the type already considered. To do this we again consider two cases. (Throughout this section we assume $v_0 = v_n$ and $v_{n+1} = v_1$.)

Case I. n is an even number, say $n = 2l$. We define two new sets of variables by

$$y_j = v_{l-j} - v_{l+j}, \qquad j = 1, 2, \cdots, l-1, \tag{33a}$$

$$z_{j+1} = v_{l-j} + v_{l+j}, \qquad j = 0, 1, 2, \cdots, l. \tag{33b}$$

Using the equations in (32) it is easy to verify that

$$\begin{bmatrix} A & -I & & \\ -I & A & -I & \\ \cdot & \cdot & \cdot & \\ & & -I & A \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{l-1} \end{bmatrix} = \begin{bmatrix} f_{l-1} - f_{l+1} \\ f_{l-2} - f_{l+2} \\ \vdots \\ f_1 - f_{n-1} \end{bmatrix} \tag{34}$$

and

(35)
$$\begin{bmatrix} A & -2I & & & \\ -I & A & -I & & \\ \cdot & \cdot & \cdot & & \\ & & -I & A & -I \\ & & & -I & \frac{1}{2}A \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_l \\ z_{l+1} \end{bmatrix} = \begin{bmatrix} 2f_l \\ f_{l-1}-f_{l+1} \\ \vdots \\ f_1+f_{n-1} \\ f_n \end{bmatrix}$$

The system (34) may now be solved using the algorithm described in § 3 and system (35) by the algorithm of § 4. The vectors v_j are obtained from the vectors y_j and z_j using (33). In fact,

$$v_{l-j}=\tfrac{1}{2}(y_j+z_{j+1}), \qquad v_{l+j}=\tfrac{1}{2}(z_{j+1}-y_j), \qquad j=1,2,\cdots,l-1,$$
$$v_l=\tfrac{1}{2}z_1, \qquad v_n=\tfrac{1}{2}z_l.$$

Case II. n is an odd number. In this case, let $l=(n+1)/2$ and define the new variables

(36a) $$y_j=v_{l-j}-v_{l+j}, \qquad j=1,2,\cdots,l-1,$$

(36b) $$z_{j+1}=v_{l-j}+v_{l+j}, \qquad j=0,1,2,\cdots,l-1.$$

Again from (32) we find

(37)
$$\begin{bmatrix} A & -I & & & \\ -I & A & -I & & \\ \cdot & \cdot & \cdot & & \\ & & -I & A & -I \\ & & & -I & A+I \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{l-1} \\ y_l \end{bmatrix} = \begin{bmatrix} f_{l-1}-f_{l+1} \\ f_{l-2}-f_{l+2} \\ \vdots \\ f_2-f_{n-1} \\ f_1-f_n \end{bmatrix}$$

and

(38)
$$\begin{bmatrix} A & -2I & & & \\ -I & A & -I & & \\ \cdot & \cdot & \cdot & & \\ & & -I & A & -I \\ & & & -I & A-I \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_{l-1} \\ z_l \end{bmatrix} = \begin{bmatrix} 2f_l \\ f_{l-1}+f_{l+1} \\ \vdots \\ f_2+f_{n-1} \\ f_1+f_n \end{bmatrix}.$$

The solution of the system (37) may be obtained using a modification of the algorithm of § 3. In this case we find that the matrices $B^{(r)}$ and $C^{(r)}$ are polynomials $P_r^*(A)$ and $Q_r^*(A)$ where P_r^* and Q_r^* are defined by

$$P_r^*(2\cos\theta)=\frac{\sin(k_r+\frac{1}{2})\theta}{\sin\frac{1}{2}\theta},$$

$$Q_r^*(2\cos\theta)=\frac{\sin(l_r+\frac{1}{2})\theta}{\sin\frac{1}{2}\theta}.$$

Using this fact, we can factor the matrices $B^{(r)}$ and $C^{(r)}$ as

$$B^{(r)} = \prod_{j=1}^{k_r} \left(A - 2\cos\frac{2j\pi}{2k_r+1} I\right),$$

$$C^{(r)} = \prod_{j=1}^{l_r} \left(A - 2\cos\frac{2j\pi}{2l_r+1} I\right)$$

where, again, $k_r = 2^r + l_r$, $0 \leqq l_r \leqq 2^r - 1$.

The solution of system (38) may be obtained using a modification of the algorithm given in § 4. In this case the matrices $B^{(r)}$ and $C^{(r)}$ are polynomials $\tilde{P}_r(A)$ and $\tilde{Q}_r(A)$ where $\tilde{P}_r$ and $\tilde{Q}_r$ are defined by

$$\tilde{P}_r(2\cos\theta) = \frac{\cos(k_r + \frac{1}{2})\theta}{\cos\frac{1}{2}\theta},$$

$$\tilde{Q}_r(2\cos\theta) = \frac{\cos(l_r + \frac{1}{2})\theta}{\cos\frac{1}{2}\theta}.$$

Using these identities, we can factor $B^{(r)}$ and $C^{(r)}$ as

$$B^{(r)} = \prod_{j=1}^{k_r} \left(A - 2\cos\frac{(2j-1)\pi}{2k_r+1} I\right),$$

$$C^{(r)} = \prod_{j=1}^{l_r} \left(A - 2\cos\frac{(2j-1)\pi}{2l_r+1} I\right),$$

where, again, $k_r = 2^r + l_r$, $0 \leqq l_r \leqq 2^r - 1$. The matrices M_1 in (27) and M_2 in (30) in this case factor as

$$M_1 = (A - 2I)U_{2^s-1}(\tfrac{1}{2}A) \prod_{j=1}^{k_s} \left(A - 2\cos\frac{2j\pi}{2k_s+1} I\right)$$

and

$$M_2 = (A - 2I)U_{2^s-1}(\tfrac{1}{2}A) \prod_{j=1}^{k_s+2^s} \left(A - 2\cos\frac{2j\pi}{2k_s+2^{s+1}+1} I\right).$$

Having solved systems (37) and (38), we use (36) to obtain

$$\begin{aligned} v_{l-j} &= \tfrac{1}{2}(z_{j+1} + y_j), \\ v_{l+j} &= \tfrac{1}{2}(z_{j+1} - y_j), \qquad\qquad j = 1, 2, \cdots, l-1, \\ v_l &= \tfrac{1}{2}z_1 \end{aligned}$$

which is the solution of the system (32).

From the discussion in the preceding sections it is clear that the execution time for this algorithm is proportional to $mn[\log_2 n]$. Figure 3 demonstrates this dependence in the case $m = n$. Note that in the case $n = 2^p$, we create two systems (34) and (35). The number of unknowns in this case is $2^{p-1} - 1$ for system (34) and $2^{p-1} + 1$ for system (35). As we have already seen these values give rise to the fastest algorithms so that $n = 2^p$ corresponds to the fastest algorithm for the periodic case even though the algorithm presented in this section is not identical to that presented in [3].

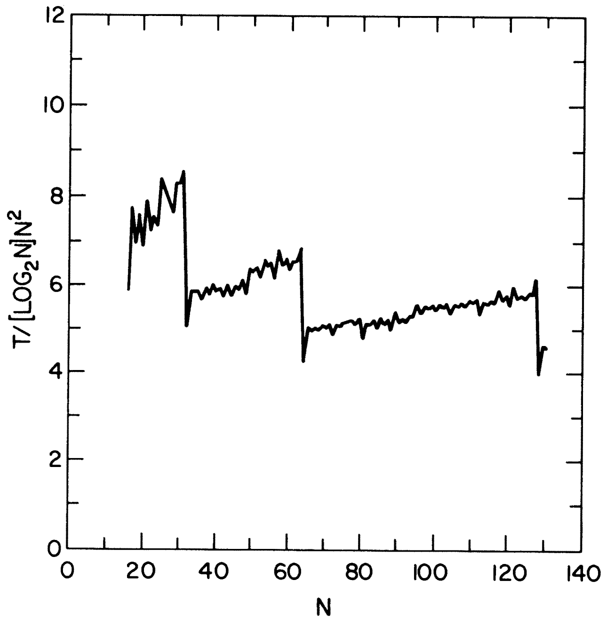

FIG. 3. *Relative computation time for solving equation (32) with $m = n$ and T measured in microseconds*

6. Summary. The algorithm presented in this paper computes the solution to equations (1), (24) or (32) in a time proportional to $mn[\log_2 n]$ for any n. This contrasts sharply with solving the same equations using FFT wherein all computational advantage is lost if n is not a highly composite number.[1] For this reason alone the algorithm presented here appears to be superior to the matrix decomposition method based on the use of FFT.

REFERENCES

[1] M. ABRAMOWITZ AND I. STEGUN, eds., *Handbook of Mathematical Functions*, Applied Mathematics Series No. 55, U.S. Dept. of Commerce, National Bureau of Standards, Washington, D.C., 1964.

[2] R. BANK, *Marching Algorithms for elliptic boundary value problems*, Center for Research in Computing Technology, TR-12-75, Harvard University, Cambridge, MA, 1975.

[3] B. BUZBEE, G. GOLUB AND C. NIELSON, *On direct methods for solving Poisson's equation*, this Journal, 7 (1970), pp. 627–656.

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

[5] R. HOCKNEY, *The potential calculation and some applications*, Methods of Computational Physics, Academic Press, New York, 1969, pp. 136–212.

[6] E. LORENZ, *A rapid procedure for inverting del-square with certain computers*, Monthly Weather Rev., 104 (1976), pp. 961–966.

[1] *Note added in proof.* Numerical experiments with the Bluestein algorithm presented by Brigham, *The Fast Fourier Transform*, Prentice-Hall, 1976, indicate that it is very inefficient for common values of n.

[7] U. SCHUMANN AND R. SWEET, *A direct method for the solution of Poisson's equation with Neumann boundary conditions on a staggered grid of arbitrary size*, J. Computational Phys., 20 (1976), pp. 171–181.

[8] P. SWARZTRAUBER, *A direct method for the discrete solution of separable elliptic equations*, this Journal, 11 (1974), pp. 1136–1150.

[9] ———, *The methods of cyclic reduction, Fourier analysis and cyclic reduction—Fourier analysis for the discrete solution of Poisson's equation on a rectangle*, SIAM Rev., to appear.

[10] P. SWARZTRAUBER AND R. SWEET, *Efficient FORTRAN subprograms for the solution of elliptic partial differential equations*, Tech. Note TN/IA-109, National Center for Atmospheric Research, Boulder, CO, 1975.

[11] R. SWEET, *Direct methods for the solution of Poisson's equation on a staggered grid*, J. Computational Phys., 12 (1973), pp. 422–428.

[12] ———, *A generalized cyclic reduction algorithm*, this Journal, 11 (1974), pp. 506–520.