JOURNAL OF COMPUTATIONAL PHYSICS 25, 319–331 (1977)

Direct Solutions for Poisson's Equation in Three Dimensions

ROBERT B. WILHELMSON AND JAMES H. ERICKSEN

Laboratory for Atmospheric Research, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801

Received June 28, 1976

Two-dimensional Poisson problems are commonly solved by one of three direct methods: cyclic reduction (CR), Fourier analysis (FA), or a combination of FA and CR (FACR). It has been shown that FACR requires the least amount of computation and CR the most. All of these methods employ one-dimensional solvers embedded in the algorithms for solving the two-dimensional problem. These methods can be used to solve three-dimensional systems if the one-dimensional solver is replaced with a two-dimensional solver. In three dimensions FACR and FA require almost the same number of operations if the two-dimensional solver is FACR. Further, the three-dimensional CR using CR to solve the two-dimensional systems requires a much larger number of operations in comparison with any of the other approaches considered. This information is illustrated using a staggered grid with Neumann boundary conditions. The operation counts for this problem are derived so that they are applicable even if the number of mesh points in any direction is small. The FFT algorithm required when using FA is presented. The FA algorithm is simpler to code than FACR. A comparison of run times between CR and FA (using CR to solve the two-dimensional systems) is given for several mesh sizes. The results agree with the operation count comparisons. Some input–output considerations for coding problems which require auxiliary storage are also discussed.

1. INTRODUCTION

Direct methods for solving the discrete Poisson equation over a two-dimensional rectangular domain have been discussed extensively in the literature. Commonly used methods include cyclic reduction (CR), Fourier analysis (FA), and a combination of cyclic reduction and Fourier analysis (FACR). Swarztrauber [6] has reviewed these methods for solving $U_{xx} + U_{yy} = f(x, y)$ on a $N \times M$ nonstaggered mesh with various boundary conditions. He gives asymptotic operations counts for Dirichlet boundary conditions of $3N \log_2 M$ for CR, $2N \log_2 M$ for FA, and $3N \log_2 \log_2 M$ for FACR, where $M = 2^m - 1$. These asymptotic operations counts also appear to be valid for periodic ($M = 2^m$) and Neumann ($M = 2^m + 1$) boundary conditions. They indicate FACR to be superior, followed by Fourier analysis and then cyclic reduction.

The main purposes of this paper are to discuss how these methods can be extended effectively to three dimensions and how their operation counts compare in three

ISSN 0021–9991

dimensions. An increasing number of fluid flow models now require three-dimensional Poisson solutions. This includes one of the author's cumulus cloud models [8]. The particular problem considered here is

$$U_{xx} + U_{yy} + U_{zz} = f(x, y, z) \qquad \text{in } R, \tag{1a}$$

$$U_n = 0 \qquad \text{on the boundary of } R, \tag{1b}$$

where R is a rectangular box defined by $R = (x, y, z)$: $0 \leqslant x \leqslant X$, $0 \leqslant y \leqslant Y$, $0 \leqslant z \leqslant Z$, and n represents the normal derivative. A staggered grid is used because of its convenience in fluid flow problems. It is defined by $R_{i,j,k} = (x_i, y_j, z_k)$: $x_i = (i - 1/2)\,\Delta x$, $i = 0, 1, 2,..., L + 1$, $y_j = (j - 1/2)\,\Delta y$, $j = 0, 1, 2,..., M + 1$, $z_k = (k - 1/2)\,\Delta z$, $k = 0, 1, 2,..., N + 1$ where $L\,\Delta x = X$, $M\,\Delta y = Y$, and $N\,\Delta z = Z$. The seven point star used to approximate (1a) is

$$\begin{aligned}&[u_{i+1,j,k} + u_{i-1,j,k}]/(\Delta x)^2 + [u_{i,j+1,k} + u_{i,j-1,k}]/(\Delta y)^2 \\ &\quad + [u_{i,j,k+1} + u_{i,j,k-1}]/(\Delta z)^2 - 2[1/(\Delta x)^2 + 1/(\Delta y)^2 + 1/(\Delta z)^2]\, u_{i,j,k} = f_{i,j,k}\,,\end{aligned} \tag{2}$$

where u represents the finite difference solution. The Neumann boundary conditions are approximated by second order differences (e.g., $[u_{1,j,k} - u_{0,j,k}]/\Delta x = 0$ at the $i = \frac{1}{2}$ boundary). If $U_n \neq 0$ then f can be modified so that the boundary condition is zero (e.g., see [9]).

The general algorithm for solving (2) consists of replacing the one-dimensional solvers in the two-dimensional CR, FA, and FACR algorithms by two-dimensional solvers. These two-dimensional solvers could be CR, FA, or FACR. Several of the possible combinations are discussed. Operation counts for selected dimensions are derived keeping in mind that L, M, and N are frequently small, i.e., $\leqslant 65$. An operation is defined as an add or subtract plus a multiply or divide. Then, input–output algorithms that can be coupled with the outlined methods are discussed since even with dimensions of 65, auxiliary storage is often needed. Finally, some general comments will be made and some actual test cases presented.

2. Methods for Solving Poisson's Equation

2.1. *A Cyclic Reduction Method* (*CR*)

By multiplying (2) by $(\Delta z)^2$, the equation for the kth x–y plane can be written

$$u_{k-1} + Ju_k + u_{k+1} = (\Delta z)^2 f_k\,.$$

This type of block tridiagonal system can be solved using CR. After r levels of reduction the equation for the kth x–y plane is

$$-u_{k-2h} + J^{(r)}u_k - u_{k+2h} = J^{(r-1)} f_k^{(r-1)} + f_{k-h}^{(r-1)} + f_{k+h}^{(r-1)} = f_k^{(r)} \tag{3}$$

where

$$
\begin{aligned}
f_k^{(0)} &= -(\Delta z)^2 f_k, \qquad h = 2^{r-1}, \qquad J^{(0)} = -J, \\
J^{(r)} &= [J^{(r-1)}]^2 - 2I \\
&= \prod_{j=1}^{2^r} (J^{(0)} - 2\cos[(2j-1)\,\pi/2^{r+1}]\, I),
\end{aligned} \tag{4}
$$

$$
J = \begin{bmatrix}
A+C & C & & & & \\
C & A & C & & & \\
 & C & A & C & & \\
 & & \ddots & \ddots & \ddots & \\
 & & & C & A & C \\
 & & & & C & A+C
\end{bmatrix}
$$

and A is the L square matrix

$$
A = \begin{bmatrix}
b+a & a & & & & \\
a & b & a & & & \\
 & a & b & a & & \\
 & & \ddots & \ddots & \ddots & \\
 & & & a & b & a \\
 & & & & a & b+a
\end{bmatrix},
$$

$$
\begin{aligned}
b &= -2[1 + (\Delta z/\Delta x)^2 + (\Delta z/\Delta y)^2], \\
a &= (\Delta z/\Delta x)^2, \qquad C = (\Delta z/\Delta y)^2 I.
\end{aligned}
$$

In this paper the mesh size in z has been constrained to $N = 2^{n+1} + 1$, where the z direction has been arbitrarily chosen for the reduction process. Round-off error becomes a problem when computing the right-hand side of (3). The Buneman variant eliminates this problem. Sweet [7] presents this method using

$$f_k^{(r)} = J^{(r)} p_k^{(r)} + q_k^{(r)} \tag{5}$$

where

$$q_k^{(0)} = f_k^{(0)}, \qquad p_k^{(0)} = 0,$$

$$q_k^{(1)} = q_{k-1}^{(0)} + q_{k+1}^{(0)} + 2p_k^{(1)}, \qquad p_k^{(1)} = [J^{(0)}]^{-1} q_k^{(0)}$$

and for $r > 1$

$$q_k^{(r)} = q_{k-h}^{(r-1)} + q_{k+h}^{(r-1)} + 2p_k^{(r)},$$

$$p_k^{(r)} = [J^{(r-1)}]^{-1}\,[p_{k-h}^{(r-1)} + p_{k+h}^{(r-1)} + q_k^{(r-1)}] + p_k^{(r-1)}.$$

The general recurrence relationships for $p_k^{(r)}$ and $q_k^{(r)}$ can be written so that only the $q_k^{(r)}$'s need to be stored. For example, initially

$$q_k^{(1)} = q_{k+1}^{(0)} + q_{k-1}^{(0)} + 2[J^{(0)}]^{-1} q_k^{(0)} \tag{6}$$

for $k = 3, 5, 7,..., N-2$. One can then determine $q_k^{(1)}$ for a given k using (6) and the solution, X, of the two-dimensional system $J^{(0)}X = q_k^{(0)}$. This first level of reduction requires solving $(N+1)/2$ two-dimensional systems as indicated in Table I. This includes the special cases for $k = 1$ and $k = N$ which involve different J's but the same formula (6) provided that planes indexed outside the subscript limits are considered to be zero. The next $n - 1$ levels of reduction can be written

$$\begin{aligned} q_k^{(r+1)} &= q_{k+2h}^{(r)} - q_{k+h}^{(r-1)} + q_k^{(r)} - q_{k-h}^{(r-1)} + q_{k-2h}^{(r)} \\ &\quad + [J^{(r)}]^{-1} [-q_{k+3h}^{(r-1)} + q_{k+2h}^{(r)} - q_{k+h}^{(r-1)} + 2q_k^{(r)} - q_{k-h}^{(r-1)} \\ &\quad + q_{k-2h}^{(r)} - q_{k-3h}^{(r-1)}] \end{aligned} \tag{7}$$

for $h = 2^{r-1}$, $k = 1, 1 + 4h, 1 + 8h,..., N$ and $r = 1, 2,..., n-1$. Again for $k = 1$ and N the J's are specially defined and indexed elements outside subscript limits are zero. Here 2^r two-dimensional systems need to be solved for each k as seen from (7) and the product (4). The resulting number of two-dimensional problems for each reduction level is $2^r\lceil N/2^{r+1}\rceil$ where $\lceil G\rceil$ is the smallest integer greater than or equal to G. This can be approximated as $(N+1)/2$. The next steps involve solving for $q^{(n+1)}$, u_1, u_i, and u_N where $i = 1 + 2^n$ and the u_k's for $k = 1,..., N$ form the solution to (2). A back substitution process involving known u_k's can then be used to fill out the solution according to Sweet's basic algorithm. The number of two-dimensional solutions required is given in Table I along with the approximate number for the entire three-dimensional solution, $(N-1)(\log_2(N-1)+2)$. Note that the term 2 adds significantly to the two-dimensional system count for smaller N. These systems can be solved by any two-dimensional method that solves $L \times M$ systems for the x and y boundary conditions.

Assuming the CR method as given by Sweet [7] is used and reduction is performed in the y direction where $M = 2^m + 1$, then $(M-1)(\log_2(M-1)+2)$ tridiagonal systems of arbitrary dimension L need to be solved per two-dimensional system. Each tridiagonal system requires $3L$ operations. The total operation count for solving the three-dimensional system is then $3L(M-1)(N-1)(\log_2(M-1)+2)$ $(\log_2(N-1)+2)$.

2.2. *Fourier Analysis (FA)*

Fourier analysis can also be used in the z direction to reduce (2) to N separate two-dimensional systems. The reduction is accomplished by analyzing f using

$$\bar{f}_s = \sum_{k=1}^{N} f_k \cos[(k - 1/2)\, \Delta z(s-1)\, \pi/(N\Delta z)], \tag{8}$$

TABLE I

A Breakdown is Given of the Number of Two-Dimensional Calculations Needed to Solve a Three-Dimensional Poisson Problem. Also a Breakdown of Input/Output by Planes is Given for an Out of Core Problem

Reduction or expansion level	No. of planes calculated	No. of two-dimensional solutions per level of reduction	Planes of data required to solve for a new plane in reduction–expansion process	Planes read or written in solving for a new plane	Same as previous column but saving planes required for the next new plane calculation in core
$q^{(1)}$	$(N+1)/2$	$(N+1)/2$	3	4	3
$q^{(r+1)}, r = 1, 2, \ldots, n-1$	$[N/2^{r+1}]$	$2^r[N/2^{r+1}]$	7	8	5
$q^{(n+1)}$	1	$(N-1)/2$	5	6	4
u_{1+2^n}	1	$N-1$	3	4	0
u_1, u_N	2	$N-1$	3	4	0
$u_k, k = 1+2^r,$ $1+3\cdot 2^r, \ldots, N-2^r,$ $r = n-1, n-2, \ldots, 1$	$(N-1)/2^{r+1}$	$(N-1)/2$	5	6	5
$u_k, r = 0$	$(N-1)/2$	$(N-1)/2$	3	4	3
The total number of two-dimensional solutions $(N-1)(\log_2(N-1)+2)$ required to solve a three dimensional system			Average I/O per plane	11	8

where $\bar{f}_s$ and f_k are two-dimensional x–y arrays. The resulting N two-dimensional systems can be written as

$$\frac{\bar{u}_s(i+1,j)+\bar{u}_s(i-1,j)}{(\Delta x)^2}+\frac{\bar{u}_s(i,j+1)+\bar{u}_s(i,j-1)}{(\Delta y)^2}$$

$$-2\left[\left(\frac{1}{\Delta x}\right)^2+\left(\frac{1}{\Delta y}\right)^2+\left(\frac{1}{\Delta z}\right)^2(1-\cos[(s-1)\,\pi/N])\right]\bar{u}_s(i,j)=\bar{f}_s(i,j) \tag{9}$$

for $s = 1, 2, \ldots, N$. The systems represented by (9) are solved for $\bar{u}_s$ using a two-dimensional Poisson solver. The $\bar{u}_s$ values are then transformed into the solution u_k using Fourier synthesis, i.e.,

$$u_k = \sum_{s=1}^{N} E_s\bar{u}_s \cos[(k-1/2)\,\Delta z(s-1)\,\pi/(N\Delta z)], \tag{10}$$

$$\begin{aligned} E_s &= 1/N \qquad \text{if } s = 1\\ &= 2/N \qquad \text{otherwise.}\end{aligned}$$

The transforms used for Fourier analysis (8) and Fourier synthesis (10) are dependent on the boundary conditions in the z direction. To calculate these transforms efficiently the transforms are converted into standard Fourier transforms for which fast Fourier transform (FFT) routines are readily available. Cooley *et al.* [1], used this approach in calculating transforms for Neumann, Dirichlet, and periodic boundary conditions on a nonstaggered grid. Their standard transform was the complex FFT routine. Appropriate pre- and postprocessing was developed. Swarztrauber [6] extended this work to include combinations of the above boundary conditions in the same direction. Pre- and postprocessing for Neumann boundary conditions on a staggered grid can also be developed in a similar manner. They follow.

Fourier analysis (8) is initiated using the preprocessing

$$\begin{aligned} v_1 &= 2f_1, \qquad v_N = 2f_N,\\ v_{2k} &= f_{2k}+f_{2k+1},\\ v_{2k+1} &= f_{2k}-f_{2k+1}. \end{aligned} \tag{11}$$

The v's are supplied to a library routine for periodic Fourier synthesis using the equation

$$\begin{aligned}\bar{v}_s = [v_1+(-1)^{s-1}v_N]/2 + \sum_{k=1}^{N/2-1} [v_{2k}\cos[2k(s-1)\,\pi/N]\\ + v_{2k+1}\sin[2k(s-1)\,\pi/N]].\end{aligned} \tag{12}$$

Postprocessing of $\bar{v}_s$ is required to get $\bar{f}_s$, i.e.,

$$\begin{aligned}\bar{f}_1 &= \bar{v}_1,\\ \bar{f}_s &= [(a+b)\bar{v}_s-(a-b)\bar{v}_{N-s+2}]/2\end{aligned} \tag{13}$$

for $s = 2,\ldots, N$, $a = \sin[(s-1)\pi/2N]$, and $b = \cos[(s-1)\pi/2N]$. This completes the analysis.

After solving the two-dimensional systems (9) Fourier synthesis on $\bar{u}_s$ will give the solution using (10). The preprocessing needed to compute u_k from $\bar{u}_s$ is

$$\begin{aligned} \bar{w}_1 &= \bar{u}_1, \\ \bar{w}_s &= [(a+b)\bar{u}_s + (a-b)\bar{u}_{N-s+2}]. \end{aligned} \tag{14}$$

The values of $\bar{w}_s$ are supplied to a library routine for periodic Fourier analysis using the equations

$$\begin{aligned} Nw_1 &= \sum_{s=1}^{N} \bar{w}_s, \qquad Nw_N = \sum_{s=1}^{N} \bar{w}_s(-1)^{s-1}, \\ Nw_{2k} &= \sum_{s=1}^{N} \bar{w}_s \cos[2k(s-1)\,\pi/N], \\ Nw_{2k+1} &= \sum_{s=1}^{N} \bar{w}_s \sin[2k(s-1)\,\pi/N]. \end{aligned} \tag{15}$$

Finally, postprocessing gives

$$\begin{aligned} u_1 &= w_1, \qquad u_N = w_N, \\ u_{2k} &= w_{2k} + w_{2k+1}, \\ u_{2k+1} &= w_{2k} - w_{2k+1}. \end{aligned} \tag{16}$$

Routines for periodic Fourier analysis (15) and synthesis (12) are available in many subroutine libraries and require $N \log_2 N/2$ multiplies and $N(1.5 \log_2 N/2 + 1)$ adds for an N-element transform. If these routines are not available in the reader's library they can be written using a library complex FFT routine and the pre- and post-processing outlined in Cooley, *et al.* [1]. The pre- and post-processing given by (11) and (13) require $2LMN$ multiples and $2LMN$ adds. Thus, Fourier analysis for Neumann boundary conditions on a staggered grid requires LM transforms (12) and the pre- and post-processing, (11) and (13), for a total of $LMN(\log_2 N + 1)$ multiplies and $1.5LMN(\log_2 N + 1)$ adds. The synthesis, (14)–(16), requires the same number of multiplies and adds. The time for both the analysis and synthesis can then be approximated by $2LMN(\log_2 N + 1)$ operations provided adds require considerably less time than multiplies.

If the two-dimensional systems are solved using FFT in y ($M = 2^m$) followed by solving M tridiagonal systems of arbitrary size L the total operation count becomes $LMN(2 \log_2 N + 2 \log_2 M + 7)$ where three of the seven account for the tridiagonal solutions. The two-dimensional systems might also be solved using cyclic reduction and expansion requiring approximately $3L(M-1)(\log_2(M-1) + 2)$ operations per system. The total operation count would then increase to $LMN(2\log_2 N + 2) + 3L(M-1)\,N(\log_2(M-1)+2)$. Another possibility is to solve the two-dimensional systems using FACR as discussed in the next section.

2.3. *Fourier Analysis and Cyclic Reduction* (*FACR*)

Hockney [2] has developed the FACR algorithm for solving the Poisson equation. It combines Fourier analysis and cyclic reduction to improve the solution speed over either method individually. Five basic steps are required: (1) perform l levels of cyclic reduction; (2) Fourier analyze the reduced system; (3) solve the resulting tridiagonal systems; (4) Fourier synthesize the reduced system; and (5) perform l levels of expansion. Consider first for clarity the solution of a two-dimensional $M \times N$ system using FACR where $M = 2^m$ and $N = 2^l k + 1$. Each level of reduction and expansion in steps 1 and 5 requires solving approximately N tridiagonal systems of length M (see Table I where the two-dimensional systems there are one-dimensional tridiagonal systems here). After l levels of reduction the resulting system is $M \times N'$, where $N' = 2^{-l}(N-1)+1$. This resulting system can be solved by Fourier analyzing and synthesizing in M (steps 2 and 4) and solving tridiagonal systems of length N' (step 3). The total operation count is then $3M(lN) + MN'(2\log_2 M + 5)$ where lN is the number of tridiagonal solutions for steps 1 and 5. For simplicity one can assume $N' \approx 2^{-l}N$ with slight favoritism to the scheme so that the operation count can be written $MN(3l + 2^{-l}(2\log_2 M + 5))$. The optimum l equals the greatest lower integer bound of $\log_2[(2\log_2 M + 5)/3]$. For $M = 64$, the optimum l_{opt} equals 2 and the operation count per point is $0.5\log_2 M + 7.25 = 10.25$. If the Fourier method is used ($l = 0$) the operation count per point would be higher, i.e., $2\log_2 M + 5 = 17$ (see Table III). The FACR algorithm can be used to solve the two-dimensional systems that occur in the Fourier method discussed in the last section. The operation count for the three-dimensional solver then becomes $LMN[2\log_2 N + 2 + 3l + 2^{-l}(2\log L + 5)]$.

When FACR is applied to a three-dimensional problem, steps 1, 3, and 5 require the solution of two-dimensional systems. For l levels of reduction and expansion in z (steps 1 and 5) $lLMN(3l' + 2^{-l'}(2\log_2 L + 5))$ operations are required if FACR is used to solve the associated two-dimensional x–y systems. Here l' is the number of reductions in y for solving these two-dimensional systems. Fourier analysis and synthesis in x (steps 2 and 4) requires $2^{-l}LMN(2\log_2 L + 2)$ operations. In step 3, y–x systems are solved by performing $l'' - l$ reductions and expansions in z. This gives a total of l'' levels of reduction and expansion because l levels are performed in steps 1 and 5. The reduced systems of dimension $M \times 2^{-l''}N$ are solved using FA. Step 3 requires $LMN(3(l''-l) + 2^{-l''}(2\log_2 M + 5))$ operations. The total operation count then becomes $LMN[l(3l' + 2^{-l'}(2\log_2 L + 5)) + 2^{-l}(2\log_2 L + 2) + 3(l''-l) + 2^{-l''}(2\log_2 M + 5)]$. The dimensions are restricted to $L = 2^i$, $M = 2^m$, and $N = 2^{l''}k + 1$ where k is an integer. Steps 1 and 5 involve l' reductions and expansions in y while step 3 requires the use of a FFT in y. Usually this would require M to be $2^{l'}j + 1$ and 2^m, respectively. By using Schumann and Sweet's algorithm [3] for reduction with arbitrary dimensions this approach can be used with $M = 2^m$. Any increase in the number of operations is ignored here. The three-dimensional FACR does not seem practical to code since there is only a small difference between the operation counts for FACR and FA using FACR and since the FACR algorithm is more complicated (see Table III).

3. Some Input–Output Considerations

Solutions to even small three-dimensional problems often require auxiliary storage. Input–output algorithms for moving data between core and auxiliary memory are then needed. Here a discussion of I/O is given with the intent of reducing the number of reads and writes necessary to solve (1). The common read or write unit considered is a plane.

Cyclic reduction can be performed by calculating $q_k^{(r)}$ and u_k planes. The number of planes that need to be calculated for each level of reduction is given in Table I in the second column and the number of planes of data required, in the fourth. The average number of planes read and written for each of the N solution (u_k) planes is 11. The I/O can be reduced if planes used in the previous plane calculation are saved in core. The average number of planes read or written then reduces to eight as seen from column 6 of Table I. Further reduction of the I/O can be accomplished by reading blocks of four consecutive planes. When two of these blocks are in core the first two levels of reduction can be performed on all but the last plane in the second block. After this is accomplished the first block is written and the next block is read. Then the first two levels of reduction are performed on these data. The process continues until all blocks have been read. This eliminates all the I/O for the second reduction. Similarly all the I/O for the next to last expansion can be eliminated. This reduces the I/O to $5\frac{1}{2}$ read or writes per plane.

The I/O for the Fourier method can in general be accomplished by (1) reading x–z (or y–z) planes, Fourier analyzing each in z and writing out the results; (2) reading and forming x–y planes, solving the two-dimensional x–y systems, and writing the results; and (3) forming x–z (or y–z) planes, Fourier synthesizing each in z and writing out the solution. The difficulty here is that if results from step 1 are written as x–z (or y–z) planes it will be hard to efficiently form x–y planes for step two. The reverse holds between step 2 and 3. One possible solution to this dilemma is to Fourier analyze as many x–z (or y–z) planes as possible in step 1 and then write then out as x–y–z blocks. A better solution exists if the two-dimensional solutions in step 2 are themselves solved using Fourier analysis rather than cyclic reduction. In the first step x–z planes can be read and Fourier analyzed in both x and z. Further, the elimination phase of the tridiagonal solver can be performed in y and the result stored. All x–z planes can be processed consecutively from $i = 1, 2, \ldots, L$ in this way. The second step is to perform the back substitution of the tridiagonal solver and Fourier synthesize as i proceeds from $L, L-1, \ldots, 1$. This method requires four reads or writes per plane and requires only two x–z planes in memory at any time.

TABLE II

Operation Counts for Solving the Two- and Three-Dimensional Poisson Equation[a]

	Method		L	M	N
Three dimensions	CR	$3L(M-1)(N-1)[\log_2(M-1)+2][\log_2(N-1)+2]$	A	2^j+1	2^k+1
	FA with FA	$LMN(2\log_2 M + 2\log_2 N + 7)$	A	2^j	2^k
	FA with CR	$3L(M-1)N(\log_2(M-1)+2) + 2LMN(\log_2 N + 1)$	A	2^j+1	2^k
	FA with FACR	$LMN[2\log_2 N + 2 + 3l + 2^{-l}(2\log_2 L + 5)]$	2^r	$2^l k + 1$	2^k
	FACR	$LMN[l(3l' + 2 - l'(2\log_2 L + 5)) + 2^{-l}(2\log_2 L + 2) + 3(l'' - l) + 2^{-l''}(2\log_2 M + 5)]$	2^r	2^j	$2^l k + 1$
Two dimensions	CR	$3M(N-1)[\log_2(N-1)+2]$		A	2^k+1
	FA	$MN(2\log_2 M + 5)$		2^j	A
	FACR	$MN(3l + 2^{-l}(2\log_2 M + 5))$		2^j	$2^l k + 1$

[a] The second through fourth methods differ in how the two-dimensional x–y systems are solved after Fourier analyzing in z. The appropriate dimensions are indicated. The symbol A represents an arbitrary dimension and i, j, k, and l are integers.

It is quite convenient since it is based on consecutive access of x–z planes that can be stored consecutively in auxiliary storage.

This is also true if FA is used with FACR; however, more data are needed in core. Suppose two blocks of 2^l x–z planes are contained in core where l is the number of levels of reduction. The general algorithm for the first half of the solution where blocks $i-1$ and i are in core is (1) perform Fourier analysis on the z rows in block i, (2) perform the levels of reduction on the y dimension in block i, (3) perform Fourier analysis on the reduced x rows in block i, (4) perform the forward elimination on the y direction in block i, (5) write block $i-1$ and read block $i+1$. After passing through the data the process can be reversed to give the solution.

TABLE III

Actual Operation Counts per Point for $2^i = 2^j = 2^k$

	Method \ $2^i = 2^j = 2^k$	16	32	64	128	256
Three dimensions	CR	95.67	138.23	186.14	239.25	297.67
	FA with FA	23	27	31	35	39
	FA with CR	26.94	32.36	37.63	42.79	47.88
	FA with FACR ($l_{\text{opt}} = 2$)	19.25	21.75	24.25	26.75	29.25
	FACR ($l = 1, l''_{\text{opt}} = l'_{\text{opt}} = 2$	20.5	22.5	24.5	26.5	28.5
Two dimensions	CR	16.94	20.36	23.63	26.79	29.88
	FA	13	15	17	19	21
	FACR ($l_{\text{opt}} = 2$)	9.25	9.75	10.25	10.75	11.25

4. Conclusions

The operation counts for the various methods previously discussed are summarized in Table II. Table III contains some actual calculations using these formulas where all the dimensions are the same, i.e., $2^i = 2^j = 2^k$. For the two-dimensional problem cyclic reduction (CR) takes about twice the time of FACR with optimal l while Fourier analysis (FA) takes roughly the average of CR and FACR. In three dimensions a noticeable change occurs. Cyclic reduction takes roughly five times as long as FACR. Further, FACR is not consistently the fastest scheme. Fourier analysis in z with FACR used to solve the resulting two-dimensional systems is fastest when the dimensions are smaller than 128. FA with FACR is recommended for solving three-dimensional problems because of its speed in relation to CR and its simplicity in relation to FACR. If a two-dimensional FACR program is not available any direct two-dimensional solver may be substituted with a slight sacrifice in speed as shown in Table III.

Several test cases were run on the CDC 7600 at the National Center for Atmospheric Research using CR and FA with CR. The results are given in Table IV. A FACR code was not written. The theoretical ratios compare quite favorably with the calculated ratios despite the fact that neither code is fully optimized. This helps to confirm the reasonableness of the nonasymptotic operational counts obtained in this paper.

TABLE IV

A Comparison of Derived Operational Counts with Actual Timings Using a CDC 7600[a]

Method	Mesh size	Operation count/point	R	Timing/ point	R
CR	$30 \times 65 \times 65$	186.14	5.0	24.0	4.9
FA with CR	$30 \times 65 \times 64$	37.63		4.9	
CR	$15 \times 65 \times 65$	186.14	5.0	25.0	5.0
FA with CR	$15 \times 65 \times 64$	37.63		5.0	
CR	$15 \times 33 \times 65$	165.42	4.8	22.0	4.8
FA with CR	$15 \times 33 \times 64$	34.36		4.6	

[a] The timing/point is in units of 10^{-5} sec and R is the ratio of the preceding counts or timings.

Although the discussion has been limited to a staggered grid with Neumann boundary conditions the general results should apply to Dirichlet or periodic boundary conditions and also to nonstaggered grid problems. This is because the operation counts for these problems should be similar to those given in this paper.

The methods discussed in this paper can be generalized. Cyclic reduction can be used to solve separable elliptic equations as discussed by Swarztrauber [5]. If Fourier analysis is used only the separable coefficients of the dimensions analyzed need be constant (e.g., see [8]). Further, cyclic reduction and Fourier analysis can be performed on grid sizes other than $2^k + 1$ and 2^k, respectively. Schumann and Sweet [3] have discussed this for the former and Singleton [4] for the latter.

Acknowledgments

This research has been sponsored by the National Science Foundation under Grant ATM 75-19336.

REFERENCES

1. J. W. COOLEY, P. A. W. LEWIS, AND P. D. WELCH, *J. Sound Vib.* **12** (1970), 315.
2. R. W. HOCKNEY, The Potential calculation and some applications, *in* "Methods of Computational Physics" (B. Alder, S. Fernback, and M. Rotenberg, Eds.), Vol. 9, p. 136, Academic Press, New York, 1969.
3. U. SCHUMANN AND R. A. SWEET, *J. Comput. Phys.* **20** (1976), 171.
4. R. C. SINGLETON, *IEEE Trans. Audio Electroacoustics* **17** (1969), 93.
5. P. N. SWARZTRAUBER, *SIAM J. Numer. Anal.* **11** (1974), 1136.
6. P. N. SWARZTRAUBER, The methods of cyclic reduction, Fourier analysis for the discrete solution of Poisson's equation on a rectangle, *SIAM Rev.* **19** (1977), 490.
7. R. A. SWEET, *J. Comput. Phys.* **12** (1973), 422.
8. R. WILHELMSON, *J. Atmos. Sci.* **31** (1974), 1629.
9. G. P. WILLIAMS, *J. Fluid Mech.* **37** (1969), 727.