39

NASA Technical Paper 1282

An Alternating Direction Implicit Method for the Control Data STAR-100 Vector Computer

Jules J. Lambiotte, Jr.

SEPTEMBER 1978

NASA

NASA Technical Paper 1282

An Alternating Direction Implicit Method for the Control Data STAR-100 Vector Computer

Jules J. Lambiotte, Jr.
Langley Research Center
Hampton, Virginia

National Aeronautics
and Space Administration

**Scientific and Technical
Information Office**

1978

SUMMARY

Two difficulties affecting the implementation of the alternating direction implicit (ADI) method on the Control Data STAR-100 computer are discussed. The first is that much of the computation is in the solution of tridiagonal systems of equations, and investigations have shown that parallel or vector tridiagonal solvers are not particularly efficient on the STAR computer until the size of the system is quite large. Second, since the direction of implicitness alternates at each half step in the algorithm, considerable data rearrangement is required to efficiently use a vector algorithm in both directions.

Two parallel algorithms are described which solve M independent systems of tridiagonal equations. Each algorithm solves the M systems more efficiently than a parallel solver used M times on one system. The two algorithms are compared by using STAR-100 timing. It is then shown that the storage requirements for the two algorithms are such that they can be used together in an ADI implementation which has no requirement for data rearrangement. An average vector length is derived for the ADI method which shows that in some instances the method may be very efficient for the STAR-100 computer.

Several other algorithms which result in the need to solve many systems of tridiagonal equations are discussed.

INTRODUCTION

The alternating direction implicit (ADI) method for solving elliptic partial differential equations has proved to be a very effective method for a restricted class of problems. It is also widely used in solving parabolic partial differential equations because it has excellent stability characteristics. Part of the popularity of this method is due to the fact that the main computational work per iteration involves the repeated solution of sets of tridiagonal equations. Whereas this latter characteristic has been one of its strengths on the conventional serial computer, studies have indicated that this may be a weakness on a vector computer. (See ref. 1.) Another difficulty arises if there is a requirement that vector elements be stored in contiguous locations, as on the Control Data STAR-100 computer. There, an algorithm deemed efficient for solving the tridiagonal systems of equations in one direction cannot be used when the direction of implicitness is alternated unless considerable data movement is performed. This movement, corresponding to the transpose of the matrix of grid values, represents an overhead to the algorithm which can be quite costly.

Two approaches to solving the tridiagonal sets of equations are presented. The first is the vectorization of an approach suggested for the ILLIAC IV in reference 2 where each parallel processor is used simultaneously to solve one of the tridiagonal systems. This is referred to as (SS) for simultaneous solution. The second method is motivated from the study of the odd-even reduction parallel algorithm in reference 1 which showed that for a large single system, this parallel approach was considerably better on the STAR-100 vector computer than the other methods considered. This report shows that M independent systems of tridiagonal equations, each of size N, can be linked together to be solved as one system of size MN with few additional computations required. In addition, it is proved that a system artificially coupled in this manner can be solved by the odd-even algorithm in $\log_2 N$ steps instead of the expected $\log_2 MN$ steps. This algorithm is referred to as the (LP) algorithm for linked parallel. It is then possible to show that the storage requirements for the two algorithms ((SS) and (LP)) are such that they can be used together to avoid the necessity of rearranging the data between half steps. It should be noted that the ILLIAC IV has a storage scheme which allows the access of both rows and columns of a matrix so that one can conveniently use (SS) to solve the equations in both directions. Similarly, Texas Instruments Advanced Scientific Computer (ASC) can access noncontiguous locations with a fixed separation but at reduced rates. The average vector length for the ADI method is derived and arguments are made that it can be an efficient algorithm for the STAR computer in some circumstances. Other numerical methods which require the solution of tridiagonal systems (e.g., successive line overrelaxation (SLOR) or Crank-Nicolson) are also discussed.

SYMBOLS AND ABBREVIATIONS

A	matrix of coefficients
ADI	alternating direction implicit method
A_i, B_i, C_i	submatrices in (SS) and SLOR algorithms
a, b, c	diagonals of tridiagonal matrix
$\tilde{a}, \tilde{b}, \tilde{c}$	diagonals of reduced tridiagonal matrix in odd-even reduction algorithm
$\hat{B}_i, \hat{C}_i$	ωB_i and ωC_i, respectively
d	diagonal of U factor in equations (1) to (3)
g	solution to forward substitution in equation (2)
i, j, k	indexing integers
(LP)	linked parallel algorithm
L, U	factors of matrix A
L_2, Q_1, Q_2	submatrices of Q in equation (11)
ℓ	subdiagonal of L factor in equations (1) to (3)
$\hat{\ell}$	average STAR-100 vector length in ADI method

M, N	grid dimensions
N_x, N_y	number of columns and rows, respectively, in grid
$\tilde{N}$	$N/2$
N'	largest power of 2 equal to or greater than N
p	MN
$\tilde{p}$	$p/2$
Q, R	implicit and explicit matrix representations, respectively, in iterative procedure
R_k	ratio of STAR-100 times for ADI algorithms as defined in equation (15)
$r, \hat{r}$	right-hand side vector in matrix equations
$r^{(i)}$	ith subvector from r
SLOR	successive line overrelaxation
(SS)	simultaneous solution algorithm
s, t	vectors defined in equations (5) and (6)
T	block diagonal matrix in which each diagonal submatrix is tridiagonal
T_{GE}	STAR-100 time to solve tridiagonal system by using serial Gauss elimination (minor cycles)
T_i	ith tridiagonal submatrix of T
$T_{LP}(M,N)$	STAR-100 time to solve M independent tridiagonal systems of size N by using (LP) algorithm (minor cycles)
T_{OE}	STAR-100 time to solve a tridiagonal system by using odd-even reduction (minor cycles)
$T_{SS}(M,N)$	STAR-100 time to solve M independent tridiagonal systems of size N by using (SS) algorithm (minor cycles)
$T_T(M,N)$	STAR-100 time to transpose an $M \times N$ matrix (minor cycles)
u	solution vector
$u^{(i)}$	ith subvector of u
$v, v^{(i)}, w, w^{(i)}$	vectors used in equation (17)
x, y, z	coordinate directions

$\alpha,\beta,\gamma,\rho,\sigma,\tau$	vectors of coefficients for finite-difference equations
δ	maximum of N_x and N_y
μ	minimum of N_x and N_y
θ	number of vector operations of length N^2 in ADI implementation
φ	number of vector operations of length N in ADI implementation
ω	relaxation parameter in SLOR

VECTOR TRIDIAGONAL SOLVERS

The usual approach to solve a diagonally dominant system $Au = r$, which is given by

$$\begin{bmatrix} a_1 & b_1 & & & & 0 \\ c_2 & a_2 & b_2 & & & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ 0 & & & \cdot & \cdot & b_{N-1} \\ & & & & c_N & a_N \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ u_N \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ r_N \end{bmatrix}$$

is to apply Gauss elimination (or some equivalent form such as the Thomas algorithm). The algorithm to factor A as $A = LU$ where

$$L = \begin{bmatrix} 1 & & & & & 0 \\ \ell_2 & 1 & & & & \\ & \ell_3 & 1 & & & \\ & & \cdot & \cdot & & \\ & & & \cdot & \cdot & \\ 0 & & & & \cdot & \cdot \\ & & & & \ell_N & 1 \end{bmatrix} \qquad U = \begin{bmatrix} d_1 & b_1 & & & & 0 \\ & d_2 & b_2 & & & \\ & & \cdot & \cdot & & \\ & & & \cdot & \cdot & \\ & & & & \cdot & \cdot \\ 0 & & & & \cdot & b_{N-1} \\ & & & & & d_N \end{bmatrix}$$

is given by

$$\begin{aligned} d_1 &= a_1 \\ \left.\begin{aligned} \ell_i &= c_i/d_{i-1} \\ d_i &= a_i - \ell_i b_{i-1} \end{aligned}\right\} & \qquad (i = 2, \ldots, N) \end{aligned} \tag{1}$$

Then, $Lg = r$ is solved by

$$\begin{aligned} g_1 &= r_1 \\ g_i &= r_i - \ell_i g_{i-1} \qquad (i = 2, \ldots, N) \end{aligned} \tag{2}$$

Finally, $Uu = g$ is solved by

$$\begin{aligned} u_N &= g_N/d_N \\ u_i &= (g_i - u_{i+1}b_i)/d_i \qquad (i = N-1, \ldots, 1) \end{aligned} \tag{3}$$

This is a very efficient algorithm requiring only 8N arithmetic operations. However, when considered for vector (or parallel) computers, the algorithm is unsuitable since the computations are of a recursive nature. The algorithm given by equations (1) to (3) is said to be nonvectorizable since it must be carried out primarily with scalar code on a vector computer (assuming that this is preferable to vectors of length one). Several parallel algorithms that use direct methods have been proposed and investigated in references 1, 3, 4, and 5. Iterative methods have been proposed in references 6 and 7.

A comparison of the predicted performance of most of these methods on the STAR-100 computer is contained in reference 1. There it is shown that, for most situations, the one-dimensional version of cyclic (odd-even) reduction performs the best. Briefly, this algorithm requires $\log_2 N$ major steps to solve a system of size N. At each step, one-half of the unknowns are eliminated, leaving a new tridiagonal system half as large. A good estimate for the STAR-100 timing for this algorithm is

$$T_{OE} \approx 5250 \log_2 N + 46N \tag{4}$$

where T_{OE} is given in units of STAR minor cycles (40 nsec). In figure 1, the ratio of T_{OE} to T_{GE} (the time for a scalar implementation of Gauss elimination on the STAR-100 computer) is plotted as a function of the number of equations N. This algorithm is described in reference 1, and experiments indicate that $T_{GE} \approx 1275 + 255N$. It is clear that even the most promising parallel solver is not relatively efficient until the size of the system is quite large. Since the size of a system using ADI is the number of grid points in one direction of the grid, it is not likely that the parallel algorithm will be useful in a straightforward implementation of ADI.

ADI METHOD

The ADI method is discussed in detail in many references. (See, for example, ref. 8.) Although the method is applicable to three-dimensional problems, this paper will only address the two-dimensional problem except where specifically noted. Briefly, when solving $Au = r$ with the ADI method, the A matrix is expressed as the sum of two matrices; this leads to the two-step procedure to advance the solution from step $2k$ to step $2k + 2$

$$Hu^{2k+1} = s \tag{5}$$

and

$$Vu^{2k+2} = t \tag{6}$$

The vectors s and t are functions of u^{2k} and u^{2k+1}, respectively. The matrices H and V are chosen so that they are invertible, and if the method is to be of practical value, easily invertible. When, for example, A arises from the application of the five-point difference formula to a general second-order elliptic partial differential equation, H corresponds to the finite-difference approximation in the x-direction and V corresponds to derivatives in the y-direction. The matrices H and V may include terms added to the diagonal to increase stability and accelerate convergence. Under suitable permutations, both H and V are tridiagonal. In solving equation (5), the direction of implicitness is said to be in the x-direction; similarly, in solving equation (6), the direction of implicitness is said to be in the y-direction.

For illustrative purposes, consider the implementation of the ADI method for a second-order elliptic partial differential equation with Dirichlet boundary conditions on the model mesh (fig. 2). Let N be the number of interior grid lines in either direction, which here is $N = 3$. Since the order in which the grid points and the coefficients at each grid point are stored in the computer is important, assume that the number of the grid point also indicates its relative position in the pertinent arrays. Therefore, all arrays are stored so that they correspond to rows of the grid. Assume further that the solution has advanced through $2k$ steps and that at step $2k + 1$, the direction of implicitness is in the x-direction. The equation for any point i is of the form

$$\left[\alpha_i u_{i-1} + \beta_i u_i + \gamma_i u_{i+1}\right]^{2k+1} = r_i - \left[\rho_i u_{i-N} + \sigma_i u_i + \tau_i u_{i+N}\right]^{2k}$$

and when all the points for that row of the grid are grouped together, a tridiagonal system of equations results. There is one such system for each row in the grid. The equations for, say, the second interior row are

$$\begin{bmatrix} \beta_{12} & \gamma_{12} & 0 \\ \alpha_{13} & \beta_{13} & \gamma_{13} \\ 0 & \alpha_{14} & \beta_{14} \end{bmatrix} \begin{bmatrix} u_{12} \\ u_{13} \\ u_{14} \end{bmatrix}^{2k+1} = \begin{bmatrix} r_{12} - \alpha_{12}u_{11} - \rho_{12}u_7 - \sigma_{12}u_{12} - \tau_{12}u_{17} \\ r_{13} \qquad\qquad - \rho_{13}u_8 - \sigma_{13}u_{13} - \tau_{13}u_{18} \\ r_{14} - \gamma_{14}u_{15} - \rho_{14}u_9 - \sigma_{14}u_{14} - \tau_{14}u_{19} \end{bmatrix}^{2k} \tag{7}$$

Note that quantities in equation (7) are stored appropriately to allow vector operations of length N in the evaluation of the right side of the equation, with the exception of the subtraction of the boundary value in the equation for the first and last point of each row. Also, each diagonal of the matrix has its elements stored consecutively, as required when using the cyclic reduction parallel algorithm discussed in the section entitled "Vector Tridiagonal Solvers."

All of the independent tridiagonal systems (one for each row) to be solved at step 2k can be grouped into one matrix equation which has the form

$$Qu^{2k+1} = Ru^{2k} + \hat{r}$$

where the matrices Q and R are as shown in figure 3 if it is assumed that any references to the known boundary values are included in $\hat{r}$. It is instructive to view the overall structure because the algorithms to be used in the ADI implementation will be based first upon the computation of $Ru^k + \hat{r}$ (in which long vector operations can be used) and then on a solution to the matrix equation which exploits the structure of Q (as opposed to solving each system one at a time).

When the direction of implicitness is in the y-direction, the resulting equation for the ith point is

$$\left[\rho_i u_{i-N} + \sigma_i u_i + \tau_i u_{i+N}\right]^{2k+2} = r_i - \left[\alpha_i u_{i-1} + \beta_i u_i + \gamma_i u_{i+1}\right]^{2k+1} \qquad (8)$$

When the equations for all points in a column of grid points are grouped together, a tridiagonal matrix arises. The matrix equation for the second column is

$$\begin{bmatrix} \sigma_8 & \tau_8 & 0 \\ \rho_{13} & \sigma_{13} & \tau_{13} \\ 0 & \rho_{18} & \sigma_{18} \end{bmatrix} \begin{bmatrix} u_8 \\ u_{13} \\ u_{18} \end{bmatrix}^{2k+2} = \begin{bmatrix} r_8 - \rho_8 u_3 - \alpha_8 u_7 - \beta_8 u_8 - \gamma_8 u_9 \\ r_{13} - \alpha_{13} u_{12} - \beta_{13} u_{13} - \gamma_{13} u_{14} \\ r_{18} - \tau_{18} u_{23} - \alpha_{18} u_{17} - \beta_{18} u_{18} - \gamma_{18} u_{19} \end{bmatrix}^{2k+1} \qquad (9)$$

Observe that the right side of equation (9) cannot be evaluated with vectors of length N, nor are the diagonals of the tridiagonal system stored consecutively. When all of the independent systems at step 2k + 1 are grouped, the matrix equation is of the form

$$Qu^{2k+2} = Ru^{2k+1} + \hat{r}$$

where Q and R now have the structure shown in figure 4. In order to have the same structure as in figure 3, one would need, for instance, u_8, followed by u_{13}, followed by u_{18}, etc. This, of course, corresponds to the transpose of the original storage scheme. On a serial computer, the different structure presents no problem, at least not when all the data are contained in central memory. The arrays of coefficients and the previous solution vector can be stored in a two-dimensional array and accessed from there one at a time. However, on a vector computer, one apparently must transpose the arrays in order to use the same algorithm in both steps. As an alternative to the data movement, two parallel approaches to solving the systems of tridiagonal equations are described. It is then shown that when used together, they remove the necessity to rearrange the data.

SIMULTANEOUS SOLUTION VECTORIZATION

Since each of the tridiagonal systems is independent of the others, any algorithm for solving one of the tridiagonal systems becomes a parallel algorithm when it is used to perform the identical operations on the corresponding coefficients of each system. Hence, with M systems, each vector operation would be of length M. The most obvious algorithm to use is the serial algorithm in equations (1) to (3). This is the general approach used in reference 2 and it is referred to here as (SS). The only difficulty is to determine what storage arrangement relative to the direction of implicitness is required. In reference 2, the coefficients for any one system are stored in one of the processors, and each processor solves the system whose coefficients are stored in its memory.

Matrix equation (9) arises when the direction of implicitness is opposite the direction of storage. A particular scalar operation for column two, say ρ_{13}/σ_8, must also be executed for columns one and three and will occur as ρ_{12}/σ_7 and ρ_{14}/σ_9. Hence, the scalar operation for each system can be performed as a one-vector operation of length M. The conclusion is that (SS) works when the direction of implicitness is opposite the direction of storage (or numbering) of the grid. A quick glance at equation (7) shows that the same approach would not work when the direction of implicitness corresponds to the numbering of the grid.

Another view of this method can be obtained by examining the structure of Q and R shown in figure 4. The matrix equation is of the form

$$\begin{bmatrix} A_1 & B_1 & 0 \\ C_2 & A_2 & B_2 \\ 0 & C_3 & A_3 \end{bmatrix} \begin{bmatrix} u^{(1)} \\ u^{(2)} \\ u^{(3)} \end{bmatrix} = \begin{bmatrix} r^{(1)} \\ r^{(2)} \\ r^{(3)} \end{bmatrix}$$

where A_i, B_i, and C_i are diagonal matrices of size M, $u^{(i)}$ refers to the ith row of unknowns, and $r^{(i)}$ is the corresponding values of the right-hand side. If the diagonal of each diagonal matrix is stored as a vector and the obvious rules for multiplying, inverting, adding, and subtracting diagonal matrices are observed, it is clear that there is a 1-to-1 correspondence with the operations in the scalar algorithm that is given by equations (1) to (3) except that each operation is now a vector operation of length M. This algorithm has been coded in STAR FORTRAN and timed. For M independent systems of equations, each of size N,

$$T_{SS}(M,N) \approx 1700N + 7.5NM \qquad (10)$$

Since N refers to the size of the system (the number of points in the direction of implicitness), $T_{SS}(M,N)$ is smaller when the direction of implicitness is along the side of the rectangle that has the smallest number of grid points.

LINKED PARALLEL VECTORIZATION

Now consider another approach to solving the M independent tridiagonal systems, each of size N. When the direction of implicitness corresponds to the direction in which the grid points have been ordered, the storage is correct for using the parallel odd-even reduction algorithm. (See eq. (7).) However, since the size of each system is the number of columns of the grid (still assuming the grid is ordered by rows), the algorithm would not appear to perform particularly well unless a very fine grid size is used. However, again it is instructive to view the overall structure as given in figure 3. If each of the M independent tridiagonal systems is denoted by $T_i u^{(i)} = r^{(i)}$, the resulting matrix equation $Tx = r$ is given by

$$\begin{bmatrix} T_1 & & & & 0 \\ & T_2 & & & \\ & & \cdot & & \\ & & & \cdot & \\ & & & & \cdot \\ 0 & & & & T_M \end{bmatrix} \begin{bmatrix} u^{(1)} \\ u^{(2)} \\ \cdot \\ \cdot \\ \cdot \\ u^{(M)} \end{bmatrix} = \begin{bmatrix} r^{(1)} \\ r^{(2)} \\ \cdot \\ \cdot \\ \cdot \\ r^{(M)} \end{bmatrix}$$

It is clear that the matrix T is also tridiagonal, of size $p = NM$, and can be solved with the odd-even reduction algorithm. The advantages of solving T as one larger system of size p are clear from figure 1. In addition, the following theorem proves that it is not necessary to carry out all of the $\log_2 p$ steps of the algorithm due to the special structure of T.

Theorem: If the tridiagonal matrix T consists of M independent tridiagonal systems of size N, the odd-even parallel algorithm can be terminated after $\log_2 N$ steps.

Proof: For simplicity, assume N is a power of two. Denote the diagonals of T by c, a, and b, and their ith component by c_i, a_i, and b_i. Note that for these uncoupled systems,

$$c_{1+(k-1)N} = 0 \qquad (k = 2, 3, \ldots, M) \tag{11}$$

and

$$b_{N+(k-1)N} = 0 \qquad (k = 1, 2, \ldots, M - 1) \tag{12}$$

At each step of the algorithm, half of the unknowns are eliminated, leaving a reduced tridiagonal system half as large as the original. Consider the first reduced system after one step of the algorithm. Call the diagonals of this system $\tilde{c}$, $\tilde{a}$, and $\tilde{b}$. Then, as shown in reference 1,

$$\tilde{c}_j = \frac{-c_{2j}c_{2j-1}}{a_{2j-1}} \qquad \left(j = 2, 3, \ldots, \frac{p}{2} \equiv \tilde{p}\right)$$

and

$$\tilde{b}_j = \frac{-b_{2j}b_{2j+1}}{a_{2j+1}} \qquad \left(j = 1, 2, \ldots, \frac{p}{2} - 1 = \tilde{p} - 1\right)$$

Now, in particular, when $j = 1 + (k - 1)\tilde{N}$ for $k = 2, 3, \ldots, M$ and $\tilde{N} = N/2$, then

$$c_{2j-1} = c_{2+2(k-1)\tilde{N}-1} = c_{1+(k-1)N} = 0$$

Also, when $j = \tilde{N} + (k - 1)\tilde{N}$ for $k = 1, 2, \ldots, M - 1$, then

$$b_{2j} = b_{2\tilde{N}+2(k-1)\tilde{N}} = b_{N+(k-1)N} = 0$$

Since $\tilde{c}_j$ and $\tilde{b}_j$ are zero for those $M - 1$ values of j, the $\tilde{p} \times \tilde{p}$ system still has $M - 1$ zero superdiagonal and subdiagonal coefficients separating the independent systems. The same argument shows that after $\log_2 N$ steps, the reduced system, which is now $M \times M$, still has $M - 1$ subdiagonal and superdiagonal zeroes. Hence, it is diagonal.

The assumption that N was a power of two was made only to allow the precise identification of the $M - 1$ zero subdiagonal and superdiagonal elements in the reduced system. It is clear from equations (11) and (12) that the zero off-diagonal elements must be propagated for any N. The elimination procedure can be stopped after $\log_2 N'$ steps where N' is the smallest power of 2 equal to or greater than N. Q.E.D.

The approximate timing for the M systems of size N is

$$T_{LP}(M,N) = 5200 \log_2 N + 46MN \tag{13}$$

MIXED ADI ALGORITHM

The discussion concerning the (SS) and (LP) algorithms indicates that (SS) is appropriate when the direction of implicitness is opposite the direction in which the grid is ordered and that (LP) is appropriate when the direction of implicitness is the same as the grid ordering. This indicates that a mixed algorithm can be used which will remove the necessity for rearranging the data.

Assume that the grid is ordered by rows [columns] of the grid. Then

(1) When implicit in the x-direction, use the (LP) [(SS)] algorithm

(2) When implicit in the y-direction, use the (SS) [(LP)] algorithm

Once the grid is ordered, the direction in which (SS) and (LP) are used is determined. When the grid is square, the storage selection makes no difference. However, when the number of rows N_y does not equal the number of columns N_x, the grid ordering should be selected to minimize the sum of the time for each algorithm. Let $\mu = \min(N_x,N_y)$ and $\delta = \max(N_x,N_y)$. Then if the grid is ordered in the direction corresponding to the maximum dimension, the time $T(ADI)$ for the mixed algorithm is

$$\begin{aligned} T(ADI) &= T_{LP}(\mu,\delta) + T_{SS}(\delta,\mu) \\ &= 5250 \log_2 \delta + 1700\mu + 53.5\delta\mu \qquad (14) \end{aligned}$$

If the grid is ordered along the minimum dimension

$$\begin{aligned} T(ADI) &= T_{LP}(\delta,\mu) + T_{SS}(\mu,\delta) \\ &= 5250 \log_2 \mu + 1700\delta + 53.5\delta\mu \end{aligned}$$

Since $\delta \geqq \mu$

$$\frac{5250 \log_2 \delta + 1700\mu}{5250 \log_2 \mu + 1700\delta} \leqq 1$$

for any $\mu \geqq 8$. Thus, the grid should be ordered consecutively along the largest dimension to yield the time given in equation (14).

The choice between the mixed algorithm and the more conventional algorithm (that is, (SS) in both directions with matrix transposes) is problem dependent. The advantage for the mixed algorithm occurs when the region is nonsquare and/or when more than one data array must be transposed. This can be seen from table I which gives the comparison for several combinations of M and N by using equations (10) and (13). In that table

$$R_k = \frac{T_{SS}(\mu,\delta) + T_{SS}(\delta,\mu) + 2kT_T(\mu,\delta)}{T_{LP}(\mu,\delta) + T_{SS}(\delta,\mu)} \qquad (15)$$

is the ratio of the times for the conventional to the mixed algorithm where k is the number of data arrays which must be transposed at each half step and $T_T(M,N)$ is the time to transpose an $M \times N$ matrix.

The use of different algorithms in different directions also applies to the three-dimensional problem. Consider the ADI method applied to a cube such as shown in figure 5. Assuming a dimension of N, the corresponding algorithm would then be

(1) Implicit in the z-direction: Use (LP) algorithm on one system of size N^3

(2) Implicit in the x-direction: Use (SS) algorithm N times, once for each square grid in the x-z plane. Vector operations are of length N

(3) Implicit in the y-direction: Use (SS) algorithm once on all N^2 tridiagonal systems. Vector operations are of length N^2

AVERAGE VECTOR LENGTH IN ADI

Even though the average vector length in the solution to the M triadiagonal systems is only M with (SS), the length for the entire algorithm may be considerably higher because the coefficients α_i, β_i, γ_i, ρ_i, σ_i, and τ_i in equation (8) may be all evaluated with vectors of length MN since the equations are independent. In many cases, this computation may dominate the time so that much of the work involves long vectors. To estimate the average vector length $\hat{\ell}$, assume that $M = N$ and that θ vector operations of length N^2 are performed to evaluate the coefficients; also assume that φN vector operations of length N are required to solve the resulting systems, as in the (SS) algorithm. Then

$$\hat{\ell} = \frac{\theta N^2 + \varphi N(N)}{\theta + \varphi N} \approx \left(\frac{\varphi + \theta}{\varphi}\right) N$$

Counting a multiplication as one operation, an addition as one-half, and a division as two, $\varphi = 7.5$ so that

$$\hat{\ell} \approx \left(\frac{7.5 + \theta}{7.5}\right) N$$

The value for θ is at least 4.5 to evaluate $Ru^k + b$ and, of course, can be considerably larger for nontrivial coefficient evaluation.

The evaluation of the coefficients and the right-hand side with vectors which are $O(MN)$ requires the boundary points to be included in the vector of unknowns. To have a valid equation for a point i on the boundary, set

$$\rho_i = \tau_i = \alpha_i = \beta_i = \gamma_i = 0 \qquad \sigma_i = 1 \qquad r_i = \hat{u}_i$$

in, for instance, equation (8), where $\hat{u}_i$ is the fixed boundary value for the ith point.

OTHER APPLICATIONS

SLOR

The two algorithms can, of course, be used in any application where there exist a number of independent tridiagonal systems to solve. It may be necessary, however, to impose an ordering other than the usual row-by-row ordering on the grid to obtain the independence required for these algorithms. For instance, the usual row by row ordering of the grid in figure 2 yields a block tridiagonal matrix C_i, A_i, B_i where C_i and B_i are diagonal of size N and each A_i is tridiagonal. When the successive line overrelaxation (SLOR) method is applied to this matrix, the following matrix equation must be solved at each step where $\hat{C}_i = \omega C_i$ and ω is the relaxation factor:

$$\begin{bmatrix} A_1 & & & & \\ \hat{C}_2 & A_2 & & 0 & \\ & \cdot & \cdot & & \\ & & \cdot & \cdot & \\ & 0 & & \cdot & \cdot \\ & & & \hat{C}_N & A_N \end{bmatrix} \begin{bmatrix} u^{(1)} \\ u^{(2)} \\ \cdot \\ \cdot \\ \cdot \\ u^{(N)} \end{bmatrix} = \begin{bmatrix} r^{(1)} \\ r^{(2)} \\ \cdot \\ \cdot \\ \cdot \\ r^{(N)} \end{bmatrix}$$

Each $u^{(i)}$ is the updated solution on the ith row of the grid. Here the tridiagonal equations are not independent and must be solved one at a time. However, as pointed out in reference 2, if the odd rows are improved first, followed by the even rows, the resulting task requires two solutions of $N/2$ independent systems of equations. Here the matrix equation is

$$\left[\begin{array}{cccccc:cccccc} A_1 & & & & & & & & & & & \\ & A_3 & & & & & & & & & & \\ & & \cdot & & & & & & & & & \\ & & & \cdot & & & & & & & & \\ & & & & \cdot & & & & & & & \\ & & & & & A_{N-1} & & & & & & \\ \hdashline \hat{C}_2 & \hat{B}_2 & & & & & A_2 & & & & & \\ & \hat{C}_4 & \hat{B}_4 & & & & & A_4 & & & & \\ & & \cdot & \cdot & & & & & \cdot & & & \\ & & & \cdot & \cdot & & & & & \cdot & & \\ & & & & \cdot & \cdot & & & & & \cdot & \\ & & & & & \hat{C}_N & & & & & & A_N \end{array}\right] \left[\begin{array}{c} u^{(1)} \\ u^{(3)} \\ \cdot \\ \cdot \\ \cdot \\ u^{(N-1)} \\ \hdashline u^{(2)} \\ u^{(4)} \\ \cdot \\ \cdot \\ \cdot \\ u^{(N)} \end{array}\right] = \left[\begin{array}{c} r^{(1)} \\ r^{(3)} \\ \cdot \\ \cdot \\ \cdot \\ r^{(N-1)} \\ \hdashline r^{(2)} \\ r^{(4)} \\ \cdot \\ \cdot \\ \cdot \\ r^{(N)} \end{array}\right] \qquad (16)$$

where also $\hat{B}_i = \omega B_i$. The matrix in equation (16) is of the form $Qv = w$ given by

$$\left[\begin{array}{c|c} Q_1 & 0 \\ \hline L_2 & Q_2 \end{array}\right]\left[\begin{array}{c} v^{(1)} \\ \hline v^{(2)} \end{array}\right] = \left[\begin{array}{c} w^{(1)} \\ \hline w^{(2)} \end{array}\right] \tag{17}$$

which is solved by

$$v^{(1)} = Q_1^{-1}w^{(1)}$$

$$v^{(2)} = Q_2^{-1}\left(w^{(2)} - L_2v^{(1)}\right)$$

The matrix results from the ordering shown in figure 6(a) and has the structure shown in figure 7(a). With this ordering, (LP) is appropriate. As in the ADI method, each of the grid points involved in the $N/2$ independent systems could be numbered columnwise so that (SS) is appropriate. This ordering and the resulting Q matrix are shown in figure 6(b) and figure 7(b).

Three-Dimensional Crank-Nicolson

As a final example, consider a three-dimensional problem that uses a Crank-Nicolson differencing procedure. (See, for example, ref. 9.) In figure 8, the differencing scheme is shown for the point (i,j,k) looking down on the x-y plane at the level $z = k$. At the point (i,j,k), there is an implicit reference to the points immediately above and below at $(i,j,k + 1)$ and $(i,j,k - 1)$, respectively. There is also reference to the points marked by a cross, as well as some values directly above and below them in the z-direction. Now, assuming the solution is specified on the boundary, one could start at the point denoted by A by grouping all the equations from each plane at that point. A tridiagonal system of equations results for finding the solution at that column of points in the z-direction. Since the boundary values are known, the system is completely specified with the given differencing. One could now systematically proceed to step 1 position in the y-direction, solve that system, step again, etc., until all unknowns corresponding to $x = 1$ are determined. With this approach, each tridiagonal system must be solved one at a time. It can be seen, however, that once the solution at $x = 1$, $y = 1$ is known, the differencing allows the two systems denoted by B to be specified. They can be solved by using either the (SS) or (LP) algorithms. Then, as shown, the three systems denoted by C can be solved at the next step, etc. Ultimately, one has M systems of size N where N is the number of grid points in the z-direction.

In the ADI method, both (SS) and (LP) were used due to the desire to remove the transpose requirement. In the latter two examples, either algorithm could be used.

CONCLUDING REMARKS

Several weaknesses of a straightforward implementation of the ADI method on a vector computer have been pointed out. Two algorithms have been presented which solve M independent systems of tridiagonal equations of size N in a reasonably efficient manner in spite of the fact that no known tridiagonal solver solves any one system particularly efficiently. It is shown that if the (SS) algorithm is used when the direction of implicitness is opposite the direction of storage of the grid values and the (LP) algorithm is used at alternate steps, then the apparent necessity to rearrange storage at alternate steps is eliminated.

The mixed algorithm is most likely to be an advantage when the grid is rectangular and/or when several transposes are required. This is because the greater speed of the (SS) algorithm is less important in these instances. In the case of a rectangular grid, it is shown that the grid should be ordered consecutively along the largest dimension.

An average vector length for the ADI method is derived which shows that, in spite of the shortness of the vector lengths used in the solution to the tridiagonal systems, the algorithm may still have a relatively large average vector length if the evaluation of the coefficients involves a lot of computations.

Other algorithms which require the solution of numerous tridiagonal systems of equations have been discussed and orderings have been shown which lead to an uncoupling of the system.

Langley Research Center
National Aeronautics and Space Administration
Hampton, VA 23665
July 14, 1978

REFERENCES

1. Lambiotte, Jules J., Jr.; and Voigt, Robert G.: The Solution of Tridiagonal Linear Systems on the CDC STAR-100 Computer. ACM Trans. Math. Software, vol. 1, no. 4, Dec. 1975, pp. 308-329.

2. Ericksen, James H.: Iterative and Direct Methods for Solving Poisson's Equation and Their Adaptability to ILLIAC IV. CAC Doc. No. 60, Univ. of Illinois at Urbana-Champaign, Dec. 10, 1972.

3. Lambiotte, Jules J., Jr.: The Solution of Linear Systems of Equations on a Vector Computer. Ph. D. Diss., Univ. of Virginia, 1975.

4. Stone, Harold S.: An Efficient Parallel Algorithm for the Solution of a Tridiagonal Linear System of Equations. J. Assoc. Comput. Mach., vol. 20, no. 1, Jan. 1973, pp. 27-38.

5. Stone, Harold S.: Parallel Tridiagonal Equation Solvers. ACM Trans. Math. Software, vol. 1, no. 4, Dec. 1975, pp. 289-307.

6. Heller, D. E.; Stevenson, D. K.; and Traub, J. F.: Accelerated Iterative Methods for the Solution of Tridiagonal Systems on Parallel Computers. J. Assoc. Comput. Mach., vol. 23, no. 4, Oct. 1976, pp. 636-654.

7. Traub, J. F., ed.: Complexity of Sequential and Parallel Numerical Algorithms. Academic Press, 1973.

8. Young, David M.: Iterative Solution of Large Linear Systems. Academic Press, 1971.

9. Dwyer, Harry A.: Solution of a Three-Dimensional Boundary-Layer Flow With Separation. AIAA J., vol. 6, no. 7, July 1968, pp. 1336-1342.

TABLE I.- STAR-100 TIMES FOR ADI METHOD

Grid size	$T_{SS}(M,N)$, sec	$T_{LP}(M,N)$, sec	$T_T(M,N)$, sec	R_1	R_5
N = 32, M = 32	0.0026	0.0030	0.0005	1.1	1.8
N = 32, M = 64	0.0030	0.0048	0.0010	1.3	2.3
N = 64, M = 32	.0053	.0051			
N = 128, M = 32	0.0106	0.0091	0.0019	1.5	2.7
N = 32, M = 128	.0038	.0085			
N = 64, M = 64	0.0060	0.0089	0.0019	1.1	2.1

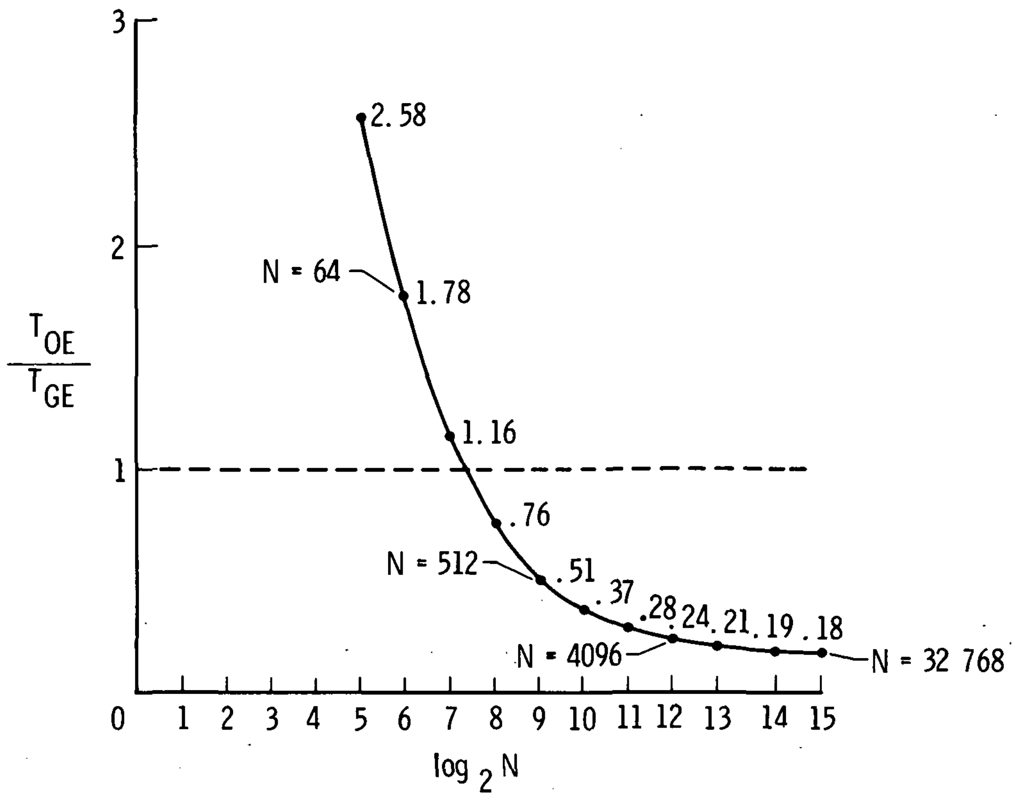


Figure 1.- Comparison of odd-even reduction with Gauss elimination for tridiagonal system of order N.

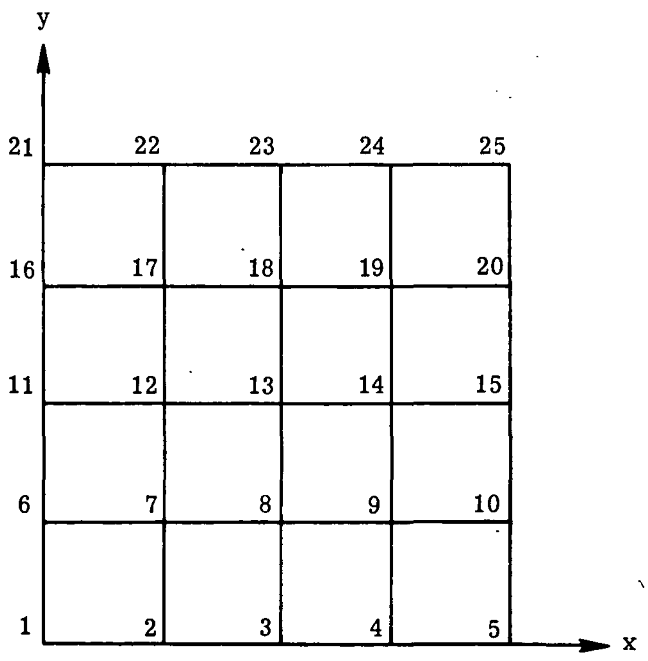


Figure 2.- Model grid.

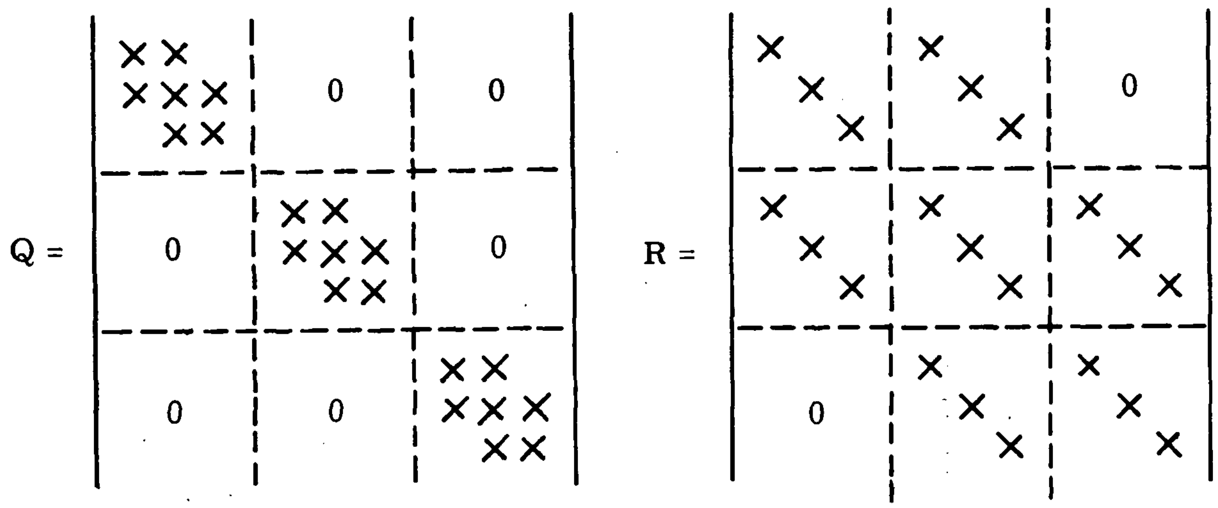


Figure 3.- Structure of ADI matrices when implicit in x-direction.

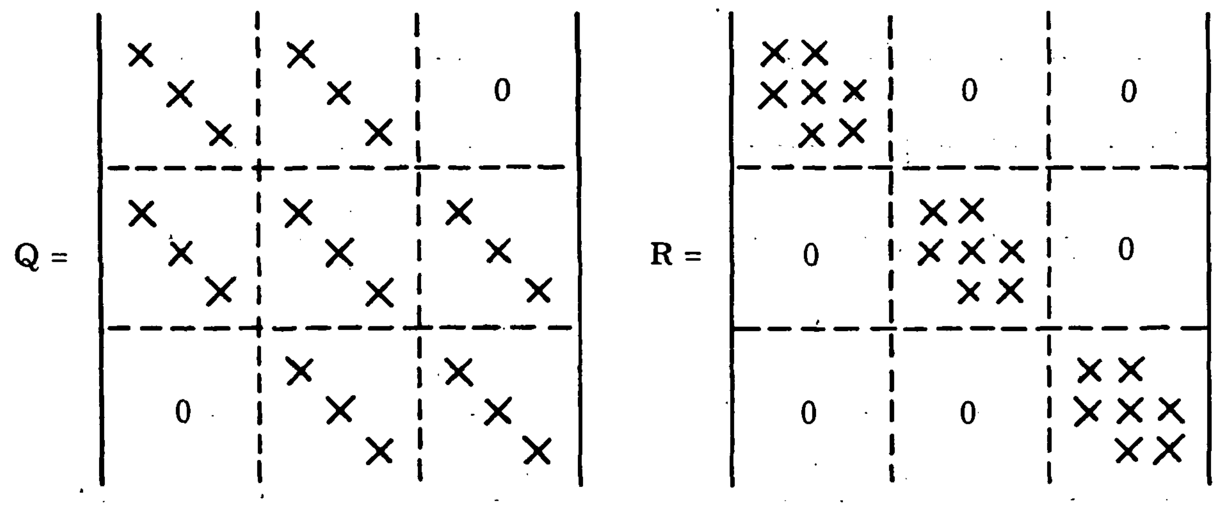


Figure 4.- Structure of ADI matrices when implicit in y-direction.

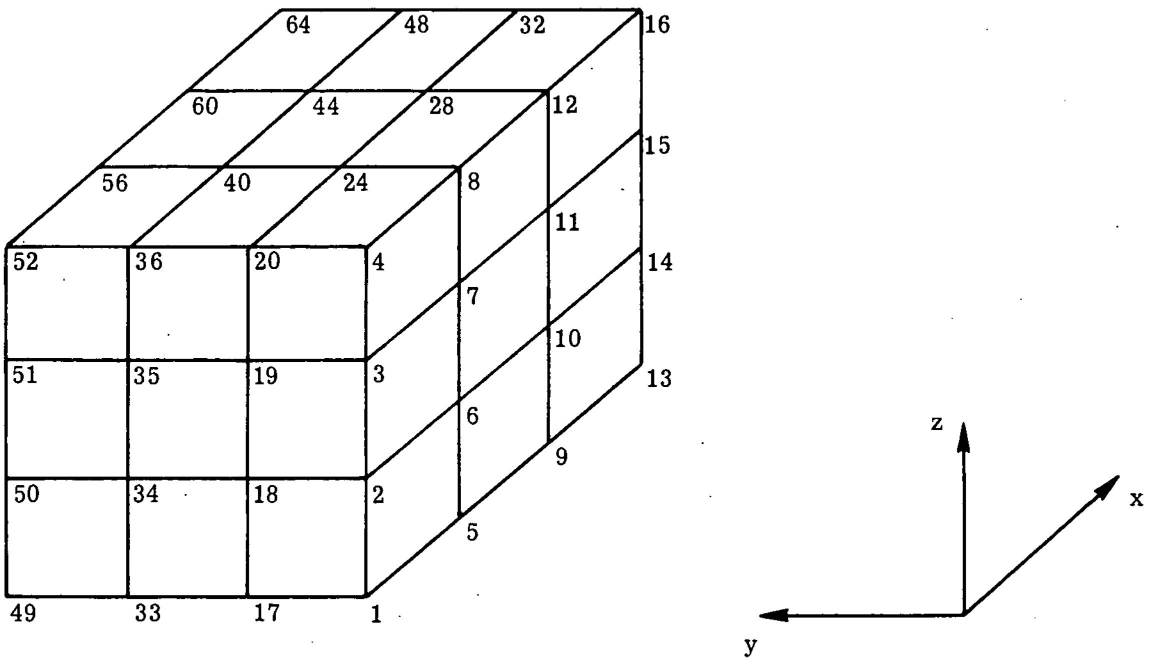
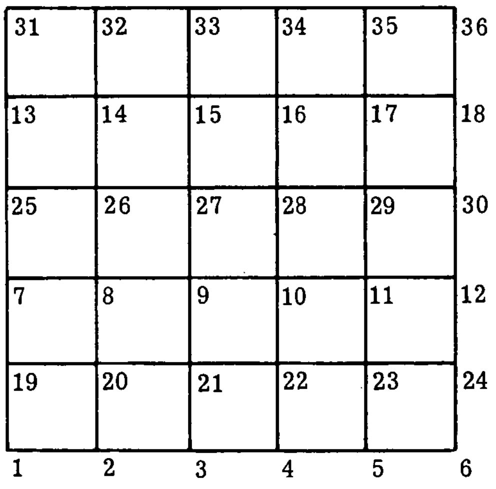
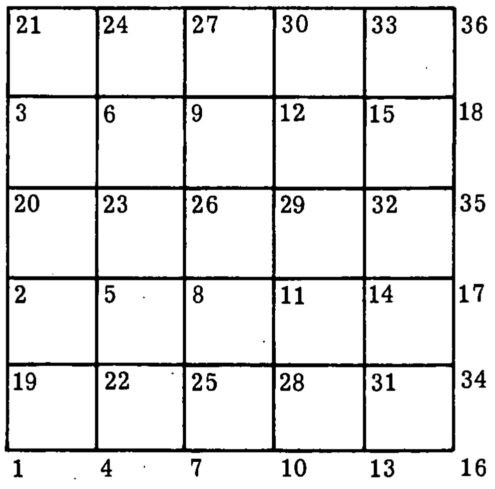


Figure 5.- Model cube.

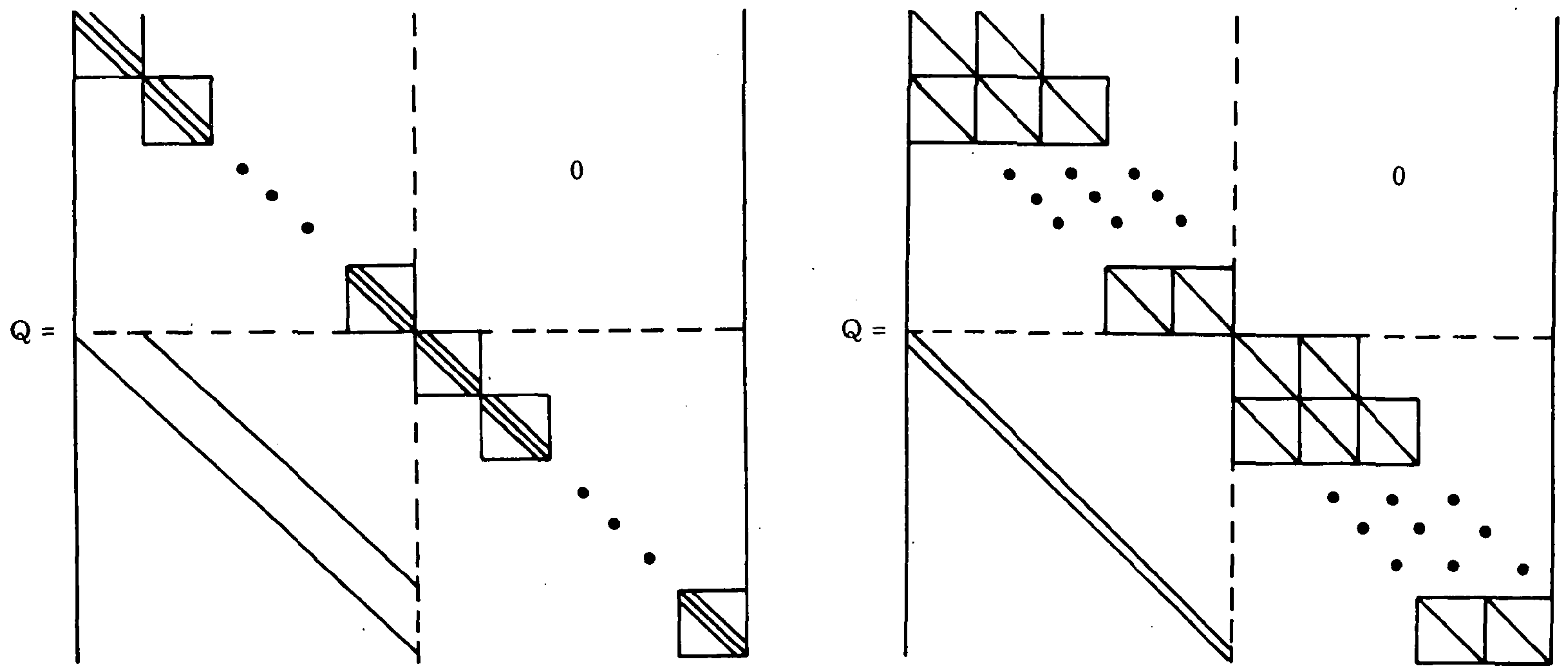


(a) Ordering for (LP) with SLOR.



(b) Ordering for (SS) with SLOR.

Figure 6.- Mesh orderings for SLOR.



(a) (LP).

(b) (SS).

Figure 7.- Structure of Q for SLOR with (LP) and (SS).

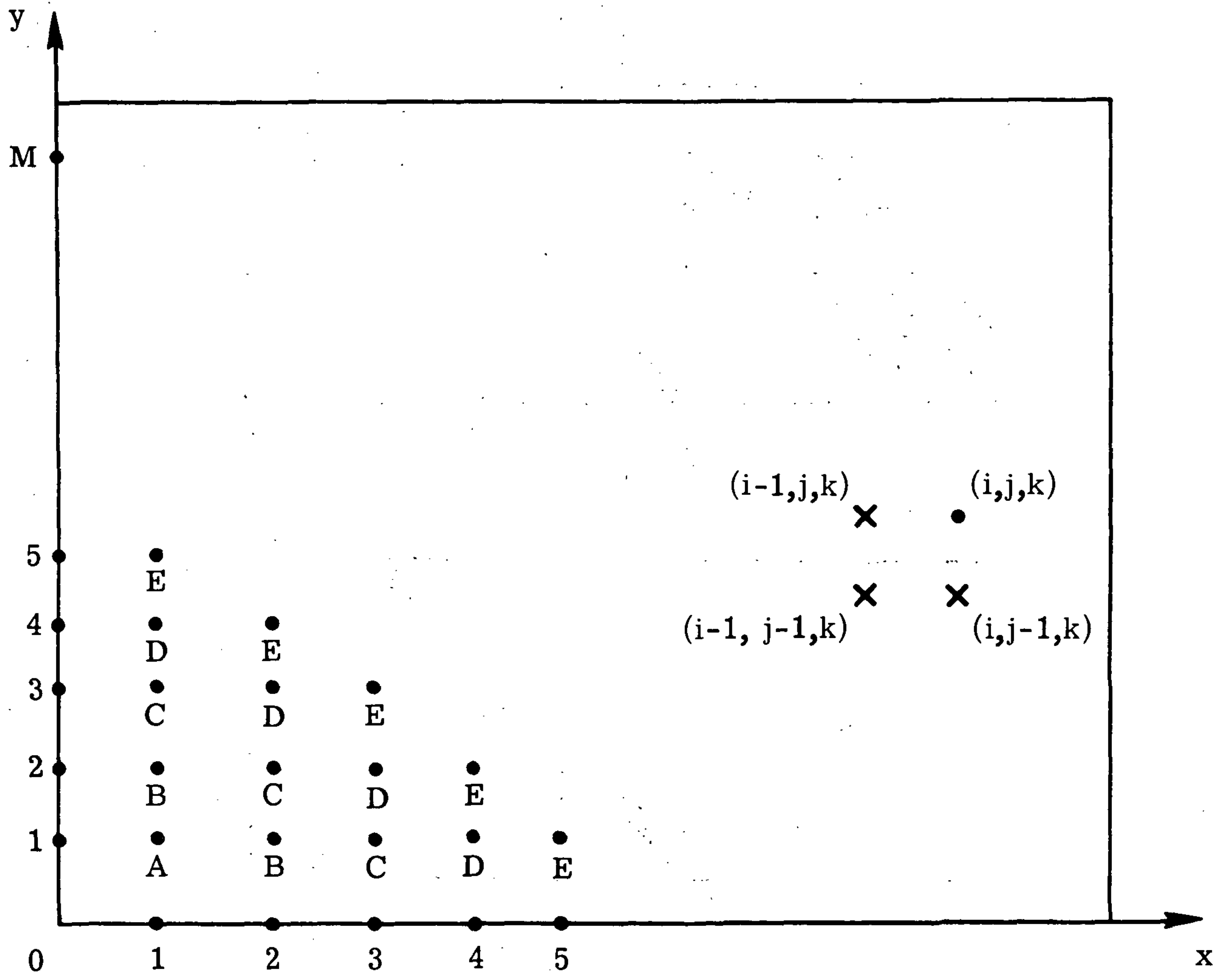


Figure 8.- Three-dimensional Crank-Nicolson differencing.

1. Report No. NASA TP-1282	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle AN ALTERNATING DIRECTION IMPLICIT METHOD FOR THE CONTROL DATA STAR-100 VECTOR COMPUTER		5. Report Date September 1978
		6. Performing Organization Code
7. Author(s) Jules J. Lambiotte, Jr.		8. Performing Organization Report No. L-12287
		10. Work Unit No. 505-15-33-02
9. Performing Organization Name and Address NASA Langley Research Center Hampton, VA 23665		11. Contract or Grant No.
		13. Type of Report and Period Covered Technical Paper
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20546		14. Sponsoring Agency Code
15. Supplementary Notes		

16. Abstract

An implementation of the alternating direction implicit (ADI) method for the Control Data STAR-100 computer is presented and analyzed. Two parallel algorithms, both of which are most efficient when used to solve many independent tridiagonal systems of equations, are discussed relative to their usefulness in an ADI implementation on the STAR-100 computer. It is shown that it may be desirable to alternate between the parallel algorithms as the direction of implicitness is alternated in order to eliminate the data rearrangement which would otherwise be required. The applicability of the two parallel tridiagonal solvers to several other numerical algorithms is also discussed.

17. Key Words (Suggested by Author(s)) STAR-100 Vector computer ADI Parallel tridiagonal solver		18. Distribution Statement Unclassified - Unlimited Subject Category 64	
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 22	22. Price* $4.00

* For sale by the National Technical Information Service, Springfield, Virginia 22161