NASA TECHNICAL NOTE **NASA TN D-7545**

NASA TN D-7545

VECTORIZATION ON THE STAR COMPUTER OF SEVERAL NUMERICAL METHODS FOR A FLUID FLOW PROBLEM

by Jules J. Lambiotte, Jr., and Lona M. Howser

Langley Research Center
Hampton, Va. 23665

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION • WASHINGTON, D. C. • JULY 1974

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TN D-7545		

4. Title and Subtitle	5. Report Date
VECTORIZATION ON THE STAR COMPUTER OF SEVERAL NUMERICAL METHODS FOR A FLUID FLOW PROBLEM	July 1974
	6. Performing Organization Code
7. Author(s)	8. Performing Organization Report No.
Jules J. Lambiotte, Jr., and Lona M. Howser	L-9339
	10. Work Unit No.
9. Performing Organization Name and Address	501-06-01-11
NASA Langley Research Center Hampton, Va. 23665	11. Contract or Grant No.
	13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address	Technical Note
National Aeronautics and Space Administration Washington, D.C. 20546	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

A reexamination of some numerical methods is considered in light of the new class of computers which use vector streaming to achieve high computation rates. A study has been made of the effect on the relative efficiency of several numerical methods applied to a particular fluid flow problem when they are implemented on a vector computer. The method of Brailovskaya, the alternating direction implicit method, a fully implicit method, and a new method called partial implicitization have been applied to the problem of determining the steady-state solution of the two-dimensional flow of a viscous incompressible fluid in a square cavity driven by a sliding wall.

The characteristics of the Control Data STAR computer have been used in this study. The timing of vector operations has been considered to develop order of computation concepts for the STAR computer.

Results were obtained on the Control Data 6600 computer system for three mesh sizes and a comparison was made of the methods for serial computation. The methods were vectorized for the STAR computer and expected timings were used to compare one iteration of each vectorized version as a function of grid size. The methods which are explicit in form are shown to vectorize better than the implicit methods in the sense that they allow the use of large vectors in the computations. This advantage becomes less important as the number of grid points increases. Two implementations of the alternating direction implicit method are presented, one of which uses a proposed parallel algorithm for solving a tridiagonal system of equations. This algorithm is shown to possess undesirable characteristics with respect to the STAR computer. Another disadvantage of the alternating direction implicit method, poor program locality in a paging environment, is pointed out and a possible solution is proposed.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Vector computer Parallel computation Fluid flow	Unclassified – Unlimited STAR Category 19

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified	57	$3.75

*For sale by the National Technical Information Service, Springfield, Virginia 22151

CONTENTS

VECTORIZATION ON THE STAR COMPUTER OF SEVERAL NUMERICAL METHODS FOR A FLUID FLOW PROBLEM

By Jules J. Lambiotte, Jr., and Lona M. Howser
Langley Research Center

SUMMARY

A reexamination of some numerical methods is considered in light of the new class of computers which use vector streaming to achieve high computation rates. A study has been made of the effect on the relative efficiency of several numerical methods applied to a particular fluid flow problem when they are implemented on a vector computer. The method of Brailovskaya, the alternating direction implicit method, a fully implicit method, and a new method called partial implicitization have been applied to the problem of determining the steady-state solution of the two-dimensional flow of a viscous incompressible fluid in a square cavity driven by a sliding wall.

The characteristics of the Control Data STAR computer have been used in this study. The timing of vector operations has been considered to develop order of computation concepts for the STAR computer.

Results were obtained on the Control Data 6600 computer system for three mesh sizes and a comparison was made of the methods for serial computation. The methods were vectorized for the STAR computer and expected timings were used to compare one iteration of each vectorized version as a function of grid size. The methods which are explicit in form are shown to vectorize better than the implicit methods in the sense that they allow the use of large vectors in the computations. This advantage becomes less important as the number of grid points increases. Two implementations of the alternating direction implicit method are presented, one of which uses a proposed parallel algorithm for solving a tridiagonal system of equations. This algorithm is shown to possess undesirable characteristics with respect to the STAR computer. Another disadvantage of the alternating direction implicit method, poor program locality in a paging environment, is pointed out and a possible solution is proposed.

INTRODUCTION

The introduction of the Control Data STAR vector streaming computer necessitates the reevaluation of many numerical methods presently being used on a serial computer. The relative efficiency of known methods may change when they are used on a vector computer. Also, new methods will be, and have been, formulated for use on the advanced computers. The process of organizing the data and calculations within a numerical method so that the operations performed take advantage of the STAR vector instructions available is referred to as the vectorization of the method. This report presents the results of a study into the effect of vectorization on several numerical methods currently being used for fluid flow problems. Also included in the study is a new method proposed for the STAR computer.

A natural classification of finite-difference methods for a time-dependent solution to a fluid flow problem is either as an explicit or implicit method. An explicit method expresses the updated solution variable at each grid point at time $t + \Delta t$ as a function of previously computed information. These methods are relatively easy to formulate but have the disadvantage of requiring a small time step to maintain numerical stability. An implicit method expresses a relationship between all or some of the solution variables at the updated time simultaneously; this gives rise to the necessity of solving a set of simultaneous equations. The implicit algorithms normally have no stability restrictions in theory but are more difficult to use in an efficient manner.

The two-dimensional flow of a viscous incompressible fluid in a square cavity driven by a sliding wall was chosen as a model problem. Both explicit and implicit methods were used on a serial computer (Control Data 6600 computer system) to obtain results for three grid sizes. The methods chosen for this problem were the method of Brailovskaya (BR), a two-step explicit method; the alternating direction implicit method (ADI); a fully implicit method (FI); and a new method by Randolph A. Graves, Jr., called partial implicitization (PI). After obtaining results on the serial computer, these methods, with some variations and exceptions, were then coded for the STAR computer using a FORTRAN-like language which has vector instructions. Timings were then obtained based on estimates supplied by Control Data Corporation. These timings give a sample of the effect of vectorization on the relative efficiency of the several methods.

SYMBOLS

ADI alternating direction implicit method

$A_{i,j}, B, C_{i,j}$ coefficients in tridiagonal matrix

BR method of Brailovskaya

$D_{i,j}, E_{i,j}, F, G_{i,j}, H_{i,j}$ coefficients of finite-difference equation for $\zeta_{i,j}$

DV(M) degree of vectorization by approach M

$Dx_{i,j} = \psi_{i+1,j} - \psi_{i-1,j}$

$Dy_{i,j} = \psi_{i,j+1} - \psi_{i,j-1}$

$dx_{i,j} = \frac{N_{Re}\,\Delta t}{4} Dx_{i,j}$

$dy_{i,j} = \frac{N_{Re}\,\Delta t}{4} Dy_{i,j}$

FI fully implicit method

g amplification factor

h spacing between grid points

I column number in ADI formulation

i,j grid location

J row number in ADI formulation

K time step

k order of the number of vector computations

$K_{i,j}, I_{i,j}, M_{i,j}, N_{i,j}, O_{i,j}$ quantities in PI solution (eq. (19))

k_1, k_2 constants

L number of results per clock

l order of the average length of a vector

l' length of vector

M	a particular vectorization approach
m	number of serial computations
$N = n + 1$	
n	number of grid lines in each direction
$N_A = \frac{n(\log_2 n - 1) + 1}{\log_2 n}$	
N_{Re}	Reynolds number
$\overline{O}_l(k)$	vectorization of a task in which $O(k)$ vector operations are performed involving vectors whose average length is $O(l)$
PI	method of partial implicitization
RD	Stone's algorithm
RT	repeated tasks
$R_{i,j}$	right-hand side of tridiagonal system of equations
s	vector startup time in clocks
ST	implementation of ADI using Stone's algorithm
T	vector timing in clocks
t	time
u_o	velocity of sliding wall
x,y	coordinates
ζ	vorticity
$\bar{\zeta}$	vorticity at intermediate step

$$\rho = \frac{\Delta t}{h^2}$$

$$\sigma_1 = \frac{\partial \psi}{\partial y} \frac{\Delta t N_{Re}}{2h}$$

$$\sigma_2 = \frac{\partial \psi}{\partial x} \frac{\Delta t N_{Re}}{2h}$$

ψ stream function

STATEMENT OF THE PROBLEM

The problem chosen was to find the steady-state solution for the flow of a viscous incompressible fluid in a square cavity driven by a sliding wall as shown in figure 1. Since it was the purpose of this report only to compare several methods when applied to a representative problem, this particular problem was chosen because of its relative simplicity and the availability of previous results (refs. 1 and 2).

The governing equations are written in a time-dependent form and the solution process is a time-marching procedure to the steady-state solution. By introducing the stream function $\psi(x,y)$ and vorticity $\zeta(x,y)$, the governing equations become, after suitably nondimensionalizing and scaling the time by a factor N_{Re},

$$\nabla^2 \psi = -\zeta \tag{1}$$

$$\frac{\partial \zeta}{\partial t} + N_{Re}\left[\left(\frac{\partial \psi}{\partial y}\right)\left(\frac{\partial \zeta}{\partial x}\right) - \left(\frac{\partial \psi}{\partial x}\right)\left(\frac{\partial \zeta}{\partial y}\right)\right] = \nabla^2 \zeta \tag{2}$$

$$\psi = 0 \qquad \text{(for all walls)} \tag{3a}$$

$$\frac{\partial \psi}{\partial n} = 0 \qquad \text{(for stationary walls)} \tag{3b}$$

$$\frac{\partial \psi}{\partial n} = -1 \qquad \text{(for moving wall)} \tag{3c}$$

The value of 100 is used for the Reynolds number in all computations.

The boundary values for ζ are computed from equation (1) and the boundary conditions for ψ. (See refs. 1 and 2 and the discussion on pp. 6 and 7 for more complete details.)

SERIAL SOLUTIONS

Several methods were used on a CDC 6600 computer to obtain results. The unit square was divided into an equally spaced n by n grid network and the differential equations (1) and (2) were expressed in a finite-difference form. In all methods, central differences were used for the discretization yielding an $O(h^2)$ spatial approximation to the original equations where $h = \frac{1}{n+1}$. For example,

$$\frac{\partial \psi}{\partial x}(x_i, y_j) = \frac{\psi(x_i + \Delta x, y_j) - \psi(x_i - \Delta x, y_j)}{2h} + O(h^2) \tag{4}$$

In the notation used herein (fig. 1),

$$\psi_{i,j} = \psi(i\Delta x, j\Delta y) = \psi(ih, jh)$$

so that equation (4) becomes

$$\left.\frac{\partial \psi}{\partial x}\right|_{i,j} = \frac{\psi_{i+1,j} - \psi_{i-1,j}}{2h} + O(h^2) \tag{5}$$

Similarly, the Laplacian operator becomes

$$\left.\nabla^2 \psi\right|_{i,j} = \frac{\psi_{i+1,j} + \psi_{i-1,j} + \psi_{i,j+1} + \psi_{i,j-1} - 4\psi_{i,j}}{h^2} + O(h^2) \tag{6}$$

For the purposes of this report, the Poisson equation (1) was considered to be an auxiliary equation and was solved in an identical fashion for each method; therefore, its solution time was not included in the timings presented. It was solved in all cases by a fully implicit method. This involved the solution of a positive definite banded system of equations and was achieved with a banded Cholesky decomposition scheme.

Figure 2 shows the program flow chart. The initial ζ^0 was taken as n copies of a vector which was order of magnitude correct with the results of Mills (ref. 1) at the center line of the grid. For a given estimate of ζ^K, equation (1) can be solved for ψ^K (originally $K = 0$). Now ζ^{K+1} can be computed on the four boundaries. Let $N = n + 1$; then, the four boundary equations are as follows:

Right boundary,

$$\zeta_{N,j} = \frac{-2\psi_{N-1,j}}{h^2} \tag{7a}$$

Left boundary,

$$\zeta_{0,j} = \frac{-2\psi_{1,j}}{h^2} \tag{7b}$$

Lower boundary,

$$\zeta_{i,0} = \frac{-2\psi_{i,1}}{h^2} \tag{7c}$$

Upper boundary,

$$\zeta_{i,N} = \frac{2}{h^2}\left(h - \psi_{i,N-1}\right) \tag{7d}$$

These equations are derived by assuming the existence of an imaginary point outside the boundary and using the governing equations at the boundary to eliminate it. Figure 3 illustrates this procedure for the right boundary.

The computation of ζ^{K+1} at the interior points is now performed by one of the previously mentioned methods. The finite-difference form for equation (2) is

$$\frac{\zeta_{i,j}^{K+1} - \zeta_{i,j}^{K}}{\Delta t} + N_{Re}\left[\frac{Dy_{i,j}}{2h}\left(\frac{\zeta_{i+1,j} - \zeta_{i-1,j}}{2h}\right) - \frac{Dx_{i,j}}{2h}\left(\frac{\zeta_{i,j+1} - \zeta_{i,j-1}}{2h}\right)\right]$$

$$= \frac{\zeta_{i+1,j} - 2\zeta_{i,j} + \zeta_{i,j+1}}{h^2} + \frac{\zeta_{i,j+1} - 2\zeta_{i,j} + \zeta_{i,j-1}}{h^2} \tag{8}$$

where

$$Dy_{i,j} = \psi_{i,j+1} - \psi_{i,j-1}$$

$$Dx_{i,j} = \psi_{i+1,j} - \psi_{i-1,j}$$

The time superscript has been deliberately deleted from the vorticity ζ since this is a function of the various methods. All ψ values are assumed to be ψ^K.

Brailovskaya's Method

Brailovskaya proposed a two-step method in which intermediate vorticity values $\bar{\zeta}^{K+1}_{i,j}$ are computed from equation (8) by using ζ from time step K (see, for example, ref. 3) and then the intermediate values are inserted into the convective terms. The equations for this problem become

$$\bar{\zeta}^{K+1}_{i,j} = \zeta^{K}_{i,j} + \frac{\Delta t}{h^2}\left(\zeta^{K}_{i+1,j} + \zeta^{K}_{i-1,j} - 4\zeta^{K}_{i,j} + \zeta^{K}_{i,j+1} + \zeta^{K}_{i,j-1}\right)$$

$$- \frac{N_{Re}\,\Delta t}{4h^2}\left[\left(Dy_{i,j-1}\right)\left(\zeta^{K}_{i+1,j} - \zeta^{K}_{i-1,j}\right) - \left(Dx_{i,j}\right)\left(\zeta^{K}_{i,j+1} - \zeta^{K}_{i,j-1}\right)\right] \qquad (9)$$

$$\zeta^{K+1}_{i,j} = \zeta^{K}_{i,j} + \frac{\Delta t}{h^2}\left(\zeta^{K}_{i+1,j} + \zeta^{K}_{i-1,j} - 4\zeta^{K}_{i,j} + \zeta^{K}_{i,j+1} + \zeta^{K}_{i,j-1}\right)$$

$$- \frac{N_{Re}\,\Delta t}{4h^2}\left[\left(Dy_{i,j-1}\right)\left(\bar{\zeta}^{K+1}_{i+1,j} - \bar{\zeta}^{K+1}_{i-1,j}\right) - \left(Dx_{i,j}\right)\left(\bar{\zeta}^{K+1}_{i,j+1} - \bar{\zeta}^{K+1}_{i,j-1}\right)\right] \qquad (10)$$

Brailovskaya's method (BR) is an explicit method and is stability limited. Carter (ref. 3) has analyzed the stability of BR on the Navier-Stokes equations. Adapting the present problem to his analysis yields the stability criterion

$$\Delta t \leq 0.205h^2$$

The method itself is comparatively simple to implement. Note that the work involved is $O(n^2)$ per time step since equations (9) and (10) are evaluated for each of the n^2 grid points.

Alternating Direction Implicit Method

The alternating direction implicit method (ADI) uses two difference equations at each point in alternate sweeps through the grid (ref. 4). At $t = 2K + 1$, equation (8) is written, a row at a time, with spatial derivatives implicit in the x-direction and explicit in the y-direction. Thus, equation (8) becomes

$$\zeta^{2K+1}_{i,j} = \zeta^{2K}_{i,j} + \frac{\Delta t}{h^2}\left[\left(\zeta^{2K+1}_{i+1,j} - 2\zeta^{2K+1}_{i,j} + \zeta^{2K+1}_{i-1,j}\right) + \left(\zeta^{2K}_{i,j+1} - 2\zeta^{2K}_{i,j} + \zeta^{2K}_{i,j-1}\right)\right]$$

$$- \frac{N_{Re}\,\Delta t}{4h^2}\left[Dy_{i,j}\left(\zeta^{2K+1}_{i+1,j} - \zeta^{2K+1}_{i-1,j}\right) - Dx_{i,j}\left(\zeta^{2K}_{i,j+1} - \zeta^{2K}_{i,j-1}\right)\right] \qquad (11)$$

Multiplying by h^2 and gathering terms yields from equation (11) a system of equations for each horizontal line of points in the grid. Now, let $y = J\,\Delta y$ for the Jth row. Then,

$$\begin{bmatrix} B & C_{1,J} & & & & & \\ A_{2,J} & B & C_{2,J} & & & & \\ & A_{3,J} & B & \cdot & & & \\ & & \cdot & \cdot & \cdot & & \\ & & & \cdot & \cdot & \cdot & \\ & & & & \cdot & C_{n-1,J} & \\ & & & & A_{n,J} & B \end{bmatrix} \begin{bmatrix} \zeta_{1,J}^{2K+1} \\ \zeta_{2,J}^{2K+1} \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \zeta_{n,J}^{2K+1} \end{bmatrix} = \begin{bmatrix} R_{1,J} \\ R_{2,J} \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ R_{n,J} \end{bmatrix} \tag{12}$$

where if

$$dx_{i,j} = \frac{N_{Re}\,\Delta t}{4} Dx_{i,j} \tag{12a}$$

$$dy_{i,j} = \frac{N_{Re}\,\Delta t}{4} Dy_{i,j} \tag{12b}$$

then,

$$B = -\left(2\Delta t + h^2\right) \tag{12c}$$

$$C_{i,J} = \Delta t - dy_{i,J} \tag{12d}$$

$$A_{i,J} = \Delta t + dy_{i,J} \tag{12e}$$

$$R_{i,J} = \zeta_{i,J}^{2K}\left(2\Delta t - h^2\right) + \zeta_{i,J+1}^{2K}\left(-\Delta t - dx_{i,J}\right) + \zeta_{i,J-1}^{2K}\left(-\Delta t + dx_{i,J}\right) \tag{12f}$$

Both $R_{i,1}$ and $R_{i,n}$ have an extra term since differencing about $\zeta_{1,J}$ and $\zeta_{n,J}$ includes the known values on the left and right boundaries, respectively. They are modified from equation (12f) as follows:

$$R_{1,J} = R_{1,J} - \zeta_{0,J}^{2K+1} A_{1,J} \tag{12g}$$

$$R_{n,J} = R_{n,J} - \zeta_{n+1,J}^{2K+1} C_{n,J} \tag{12h}$$

This system of equations is often solved by Thomas' algorithm, which is equivalent to a Gaussian elimination factorization of the matrix without pivoting. The steps of the algorithm, dropping the J subscript for simplicity, are

$$
\left.
\begin{aligned}
w_1 &= B \\
P_1 &= C_1/w_1 \\
Q_1 &= R_1/w_1 \\
&\left.
\begin{aligned}
w_j &= B - A_j P_{j-1} \\
Q_j &= \frac{R_j - A_j Q_{j-1}}{w_j} \\
P_j &= C_j/w_j
\end{aligned}
\right\} \quad (j = 2, 3, \ldots, n-1) \\
w_n &= B - A_n P_{n-1} \\
Q_n &= \frac{R_n - A_n Q_{n-1}}{w_n}
\end{aligned}
\right\} \qquad (13a)
$$

Then,

$$
\left.
\begin{aligned}
\zeta_n &= Q_n \\
\zeta_j &= Q_j - P_j \zeta_{j+1} \qquad (j = n-1, n-2, \ldots, 1)
\end{aligned}
\right\} \qquad (13b)
$$

For each row of the grid ($J = 1, 2, \ldots, n$), a similar structured system is generated which is similarly solved. Each row is solved independently of the other. This fact is taken advantage of when the solution process is set up for the STAR computer.

When the direction for the next time step $2K + 2$ is alternated so that the implicitness is in the discretization of the derivatives in the y-direction, the following equation, which is similar to equation (8), is obtained:

$$\zeta_{i,j}^{2K+2} = \zeta_{i,j}^{2K+1} + \frac{\Delta t}{h^2}\left[\left(\zeta_{i+1,j}^{2K+1} - 2\zeta_{i,j}^{2K+1} + \zeta_{i-1,j}^{2K+1}\right) + \left(\zeta_{i,j+1}^{2K+2} - 2\zeta_{i,j}^{2K+2} + \zeta_{i,j-1}^{2K+2}\right)\right]$$

$$- \frac{N_{Re}\,\Delta t}{4h^2}\left[Dy_{i,j}\left(\zeta_{i+1,j}^{2K+1} - \zeta_{i-1,j}^{2K+1}\right) - Dx_{i,j}\left(\zeta_{i,j+1}^{2K+2} - \zeta_{i,j-1}^{2K+2}\right)\right] \tag{14}$$

Now, the equations for the Ith column (that is, $x = I\,\Delta x$) become

$$\begin{bmatrix} B & C_{I,1} & & & & & \\ A_{I,2} & B & C_{I,2} & & & & \\ & A_{I,3} & B & \cdot & & & \\ & & \cdot & \cdot & \cdot & & \\ & & & \cdot & \cdot & \cdot & \\ & & & & \cdot & \cdot & C_{I,n-1} \\ & & & & & A_{I,n} & B \end{bmatrix} \begin{bmatrix} \zeta_{I,1}^{2K+2} \\ \zeta_{I,2}^{2K+2} \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \zeta_{I,n}^{2K+2} \end{bmatrix} = \begin{bmatrix} R_{I,1} \\ R_{I,2} \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ R_{I,n} \end{bmatrix} \tag{15}$$

where using equations (12a), (12b), and (12c)

$$C_{I,j} = \Delta t + dx_{I,j} \tag{15a}$$

$$A_{I,j} = \Delta t - dx_{I,j} \tag{15b}$$

$$R_{I,j} = \zeta_{I,j}^{2K+1}\left(2\Delta t - h^2\right) + \zeta_{I+1,j}^{2K+1}\left(-\Delta t + Dy_{I,j}\right) + \zeta_{I-1,j}^{2K+1}\left(-\Delta t - Dy_{I,j}\right) \tag{15c}$$

Modifying as previously yields

$$R_{I,1} = R_{I,1} - \zeta_{I,0}^{2K+2}\,A_{I,1} \tag{15d}$$

$$R_{I,n} = R_{I,n} - \zeta_{I,n+1}^{2K+2}\,C_{I,n} \tag{15e}$$

The amount of computation involved in the solution of equation (12) by Thomas' algorithm is $O(n)$ and, hence, the computation per time step for the n systems is $O(n^2)$. The changing of directions presents added programing complexity but the alternation of direction is necessary since it is this process that gives the unconditional stability after two equal time steps.

Performing a linearized stability analysis (ref. 4) for this problem results in

$$\zeta_{i,j}^{2K+2} = g\zeta_{i,j}^{2K}$$

The amplification factor g is given by

$$g = \left[\frac{\left(1 - 4\rho \sin^2 \frac{k_1 \Delta x}{2}\right) - 2i\sigma_1 \sin k_1 \Delta x}{\left(1 + 4\rho \sin^2 \frac{k_1 \Delta x}{2}\right) + 2i\sigma_1 \sin k_1 \Delta x}\right]\left[\frac{\left(1 - 4\rho \sin^2 \frac{k_2 \Delta y}{2}\right) + 2i\sigma_2 \sin k_2 \Delta y}{\left(1 + 4\rho \sin^2 \frac{k_2 \Delta y}{2}\right) - 2i\sigma_2 \sin k_2 \Delta y}\right]$$

where k_1 and k_2 are constants and

$$\rho = \frac{\Delta t}{h^2} \qquad \sigma_1 = \frac{\partial \psi}{\partial y} \frac{\Delta t N_{Re}}{2h} \qquad \sigma_2 = \frac{\partial \psi}{\partial x} \frac{\Delta t N_{Re}}{2h}$$

The Von Neumann condition for stability is $|g| \leq 1$. This condition is satisfied since g is composed of two factors and each factor is of the form

$$f = \frac{a + ib}{e - ib}$$

where $|a| \leq |e|$. Hence, it can be shown that $|f| \leq |1|$.

Fully Implicit Method

For the fully implicit method (FI) the values of $\zeta_{i,j}$ in equation (8) are taken at time $K + 1$; this results in the following equation for the i,j point:

$$D_{i,j}\,\zeta^{K+1}_{i-1,j} + E_{i,j}\,\zeta^{K+1}_{i+1,j} + G_{i,j}\,\zeta^{K+1}_{i,j+1} + H_{i,j}\,\zeta^{K+1}_{i,j-1} + F\,\zeta^{K+1}_{i,j} = -h^2\zeta^{K}_{i,j} \qquad (16)$$

where

$$D_{i,j} = \Delta t + dy_{i,j} \qquad G_{i,j} = \Delta t + dx_{i,j} \qquad F = -4\Delta t - h^2$$

$$E_{i,j} = \Delta t - dy_{i,j} \qquad H_{i,j} = \Delta t - dx_{i,j}$$

Equations (16) satisfies the Von Neumann condition for stability since here the amplification factor g is

$$g = \frac{1}{1 + 4\rho\left(\sin^2 \frac{k_1 \Delta x}{2} + \sin^2 \frac{k_2 \Delta y}{2}\right) + i\left(2\sigma_1 \sin k_1 \Delta x - 2\sigma_2 \sin k_2 \Delta y\right)} \qquad (17)$$

and clearly $|g| \leq 1$ since g is of the form $g = \frac{1}{a + ib}$ where $a \geq 1$.

The use of FI introduces several computational burdens. The resulting matrix is n^2 by n^2 and when a row by row ordering scheme is used, it becomes a banded matrix. This matrix is said to have a bandwidth of n and when banded programing techniques are used, the order of computation is $O(n^4)$ and the required storage is $O(n^3)$ which is considerably higher than that required by ADI. It should also be noted that although the band itself is sparse, it quickly fills so that after the elimination of the first n variables (one row) the submatrix for the next n variables is full. Thus, it is not possible to take advantage of sparsity within the band. Figure 4 shows the band structure and the fill that occurs after the first n variables are eliminated.

The reasons FI was considered in spite of these disadvantages are as follows:

(1) Recent advances in sparse matrix theory reduce these computation and storage figures to $O(n^3)$ and $O(n^2 \log_2 n)$, respectively. (See ref. 5.)

(2) The solution procedures for the tridiagonal systems in ADI appeared nonvectorizable.

(3) It was considered possible that FI might have better convergence properties than ADI (require fewer steps to reach steady state).

Method of Partial Implicitization

The method of partial implicitization (PI) has recently been proposed by Graves (ref. 6). In this method he has been able to express $\zeta_{i,j}^{n+1}$ explicitly in terms of past information and at the same time retain the stability characteristics of a fully implicit method.

The derivation for the stated problem proceeds in the following manner. Observe from equation (16), which is repeated for convenience,

$$D_{i,j}\,\zeta_{i-1,j}^{K+1} + E_{i,j}\,\zeta_{i+1,j}^{K+1} + G_{i,j}\,\zeta_{i,j+1}^{K+1} + H_{i,j}\,\zeta_{i,j-1}^{K+1} + F\zeta_{i,j}^{K+1} = -h^2\zeta_{i,j}^{K}$$

that the five grid points included in the general equation form the familiar star or cross pattern with $\zeta_{i,j}$ at the center. Based upon the presumption that it is these four neighbors that exert the most influence upon the solution at that point, the general equation for each of the four neighbors is also written. References to grid points within the star for $\zeta_{i,j}$ are made implicitly, whereas those outside are expressed explicitly. The resulting five equations can be expressed in matrix form as follows:

$$\begin{bmatrix} F & 0 & H_{i,j+1} & 0 & 0 \\ 0 & F & E_{i-1,j} & 0 & 0 \\ G_{i,j} & D_{i,j} & F & E_{i,j} & H_{i,j} \\ 0 & 0 & D_{i+1,j} & F & 0 \\ 0 & 0 & G_{i,j-1} & 0 & F \end{bmatrix} \begin{bmatrix} \zeta_{i,j+1}^{K+1} \\ \zeta_{i-1,j}^{K+1} \\ \zeta_{i,j}^{K+1} \\ \zeta_{i+1,j}^{K+1} \\ \zeta_{i,j-1}^{K+1} \end{bmatrix} = \begin{bmatrix} K_{i,j} \\ L_{i,j} \\ M_{i,j} \\ N_{i,j} \\ O_{i,j} \end{bmatrix} \tag{18}$$

where

$$K_{i,j} = -h^2\zeta_{i,j+1}^{K} - D_{i,j+1}\,\zeta_{i-1,j+1}^{K} - E_{i,j+1}\,\zeta_{i+1,j+1}^{K} - G_{i,j+1}\,\zeta_{i,j+2}^{K}$$

$$L_{i,j} = -h^2\zeta_{i-1,j}^{K} - D_{i-1,j}\,\zeta_{i-2,j}^{K} - G_{i-1,j}\,\zeta_{i-1,j+1}^{K} - H_{i-1,j}\,\zeta_{i-1,j-1}^{K}$$

$$M_{i,j} = -h^2\zeta_{i,j}^{K}$$

$$N_{i,j} = -h^2\zeta_{i+1,j}^{K} - E_{i+1,j}\,\zeta_{i+2,j}^{K} - G_{i+1,j}\,\zeta_{i+1,j+1}^{K} - H_{i+1,j}\,\zeta_{i+1,j-1}^{K}$$

$$O_{i,j} = -h^2\zeta_{i,j-1}^{K} - D_{i,j-1}\,\zeta_{i-1,j-1}^{K} - E_{i,j-1}\,\zeta_{i+1,j-1}^{K} - H_{i,j-1}\,\zeta_{i,j-2}^{K}$$

Matrix equation (18) is solved for $\zeta_{i,j}^{K+1}$ by using Cramer's rule. The result is

$$\zeta_{i,j}^{K+1} = \frac{FM_{i,j} - \left(N_{i,j}\,E_{i,j} + O_{i,j}\,H_{i,j} + L_{i,j}\,D_{i,j} + K_{i,j}\,G_{i,j}\right)}{F^2 - \left(E_{i,j}\,D_{i+1,j} + H_{i,j}\,G_{i,j-1} + D_{i,j}\,E_{i-1,j} + G_{i,j}\,H_{i,j+1}\right)} \tag{19}$$

Equation (19) can be used for all the interior points except the points adjacent to the boundary. Although it is possible in such a case to simply remove the equation for the boundary point from the system and rederive the expression for $\zeta_{i,j}^{K+1}$, it is desirable for the vector operation of the STAR computer to maintain formula (19) for all points. This objective can be accomplished by modifying some of the appropriate constants in matrix equation (18). For example, let $\zeta_{i,j}$ be a point adjacent to the top boundary. Then $\zeta_{i,j+1}^{K+1}$ is known. The resulting four equations to be solved can be expressed in the same format as matrix equation (18) as follows:

$$\begin{bmatrix} 1 & 0 & H_{i,j+1} & 0 & 0 \\ 0 & F & E_{i-1,j} & 0 & 0 \\ G_{i,j} & D_{i,j} & F & E_{i,j} & H_{i,j} \\ 0 & 0 & D_{i+1,j} & F & 0 \\ 0 & 0 & G_{i,j-1} & 0 & F \end{bmatrix} \begin{bmatrix} \zeta^{K+1}_{i,j+1} \\ \zeta^{K+1}_{i-1,j} \\ \zeta^{K+1}_{i,j} \\ \zeta^{K+1}_{i+1,j} \\ \zeta^{K+1}_{i,j-1} \end{bmatrix} = \begin{bmatrix} K_{i,j} \\ L_{i,j} \\ M_{i,j} \\ N_{i,j} \\ O_{i,j} \end{bmatrix} \tag{20}$$

where the following changes are made to arrays of coefficients and right-hand-side constants given in matrix equation (18):

First,

$$M_{i,j} = M_{i,j} - \zeta^{K+1}_{i,j+1} G_{i,j} \tag{20a}$$

then,

$$G_{i,j} = 0 \tag{20b}$$

$$H_{i,j+1} = 0 \tag{20c}$$

$$K_{i,j} = \zeta^{K+1}_{i,j+1} \tag{20d}$$

The system of matrix equation (20) is now correct for the point near the boundary and equation (19) can be used for this point also. Similar logical changes can be made for points near the other boundaries and near the corners. The amount of computation for PI is $O(n^2)$ but requires about twice as many operations as ADI; however, the explicit nature of PI allows it to vectorize much better than ADI on the STAR computer and, in fact, the simplicity of the PI form may make it popular on a serial computer.

The stability of this method has been verified for the two-dimensional heat equation and by Graves for Burgers' equation. The grid sizes run for this problem showed no stability constraints and in fact demonstrated a seemingly complete insensitivity to Δt.

Results of Serial Computations

The results from the computations done on the CDC 6600 computer using the four methods are presented in table I. The best results are reported for each method. The following observations can be made regarding these results:

(1) Despite the fact that the FI took fewer total steps to reach convergence, its large computation time heavily negated this slight advantage. Since it did not show any great advantage over the other two stable methods, it will be omitted from consideration in the rest of this report.

(2) ADI and BR took approximately the same amount of time per step and PI required about twice that amount.

(3) Neither PI nor ADI greatly reduced the total number of steps to convergence. On the average they took about one-third as many steps. Interestingly, PI had the characteristic of being insensitive to the size of Δt selected if Δt was greater than some number. For any Δt larger than this value, steady state was reached in the same number of steps. ADI always reached a point at which a larger Δt would cause the results to diverge toward infinity.

(4) As n doubled, the ADI, BR, and PI methods required about four times as many steps.

(5) Of the three methods under consideration, ADI performed the best and PI and BR performed about the same.

VECTORIZATION OF THE PROBLEM

General Characteristics of Vector Timing

The STAR computer obtains an effective increase in computational speed by streaming consecutively stored data from the memory, through pipeline processing units, and back to memory so that the elapsed time between the production of successive results is much less than the time from beginning to end of any one computation. This process requires that the data be organized into a vector format (that is, stored in consecutive locations in memory) and that the computations use STAR vector instructions.

Since a comparison of different vector implementations is desired, it is necessary to first look at the general timing for a vector operation and understand its implications. The general form, given in clocks (1 clock = 40 nanoseconds), is

$$T = s + \frac{l'}{L}$$

where

T	time for operation, clocks
s	startup time, clocks
l'	length of vector
L	number of results per clock

Some representative STAR timings are given in table II. These timings are based on an unpublished preliminary STAR timing summary (Aug. 15 '72) supplied by Control Data Corporation and, inasmuch as they are preliminary timings, are subject to change.

The startup time represents an inefficiency in the use of vector instructions. Obviously, the timing is best for a particular computation if it is performed with long vectors and few startups, if possible.

Since the timing for a computation depends not only upon how many results are generated but also on whether they were performed with a few long vectors or with many short vectors, it is convenient to introduce some notation and definitions to describe the vector implementation in an order sense.

Presume that there is a computational task to perform which has associated with it a parameter n which in some way characterizes the size of the task. In discussing quantities related to the computation of the task, the concept of order of magnitude at infinity with respect to n is used. Specify that $f(n) = O(g(n))$ (read as "f and g are of the same order of magnitude for large n") if $\lim_{n\to\infty} \frac{f(n)}{g(n)}$ exists and $\lim_{n\to\infty} \frac{f(n)}{g(n)} = K$ where $0 < K < \infty$.

For compactness of notation n is suppressed and $f = O(g)$ is written if the parameter involved is clearly defined. Note that in the statement $f = O(g)$, g is not uniquely defined by f; that is, $5n^3 + 6n^2 + 12 = O(5n^3 + 6n^2) = O(5n^3) = O(10n^3) = O(n^3)$. In the examples given herein, the simplest expression is always used; that is, $5n^3 + 6n^2 + 12 = O(n^3)$.

Consider now the implementation of the presumed computational task. In the following discussion, m, ℓ, and k are assumed to be functions of n, and the usual implementation that carries out the computations on a serial computer or on the STAR computer without vector operations (referred to as the scalar mode) is assumed to require $O(m)$ calculations. There may be many vector implementations (vectorizations) of the task and it is assumed that a particular vectorization M requires $O(k)$ vector operations whose average length is $O(\ell)$. The vector order of computation of such a vectorization M is denoted by $\overline{O}_\ell(k)$.

Definition: The degree of vectorization by vectorization M, denoted by $DV(M)$, is m/k.

Again, since m and k are not unique, $DV(M)$ is not unique, and $DV(M)$ can be any $f(n)$ which is $O(m/k)$. As before, the simplest form is used.

On an ideal parallel computer (one which has an infinite number of processors which operate in parallel), $DV(M)$ could be called the speedup ratio. On the STAR computer, $DV(M)$ is only an indication of the speedup.

Definition: Vectorization M is a consistent vectorization if $\ell k = O(m)$; otherwise, it is called inconsistent.

An inconsistent vectorization is hence one that produces a higher order of total results than the serial algorithm it replaces. Since the high-order term for the timing of the scalar algorithm is $k_1 m$ and for the inconsistent algorithm is $k_2 \ell k$ (for some constants k_1 and k_2), it is guaranteed that as $n \to \infty$, the scalar algorithm becomes better than the vectorized algorithm. Naturally, the value of n for which this happens depends upon the coefficients involved and the relationship between $k\ell$ and m. There are examples of parallel algorithms proposed which, if vectorized, would not be consistent. They are designed with the ideal parallel computer as the model and assume that timing is proportional to the number of vector operations involved and independent of the length of the vectors. This assumption is not valid for the STAR computer. However, there may be regions of vector lengths where these algorithms can be useful but with reduced gains. The recursive doubling algorithm of Stone (ref. 7) is an example of an inconsistent vectorization and is discussed in more detail subsequently.

A consistent vectorization is considered to be optimal if there is no other consistent vectorization of the task whose degree of vectorization is of higher order. Certainly this is true of vectorization M if DV(M) = m. It should be emphasized that the term "optimal" refers only to the vectorization of the particular task in question. If that task is only a part of the overall solution procedure, then an approach in which that task vectorizes optimally need not be the best approach to take. For example, if the vectorization of several iterative methods is considered, and specifically the vectorization of the computations involved in one iteration of each method, it is possible that one iteration of a method with poor convergence rates can be vectorized optimally, whereas a method with good convergence rates may have a vectorization of lesser degree. The method with the optimal vectorization is not necessarily the best approach to use since it requires more iterations.

To illustrate the use of these terms, consider the vectorization of two approaches for adding two n by n matrices, namely,

$$C = A + B$$

Vectorization M_1: Let each column of the matrices be a vector. Then, n vector adds of length n yield

$$T_1 = n\left(33 + \frac{n}{2}\right) = 33n + \frac{n^2}{2}$$

This vectorization is $\bar{O}_n(n)$, and $DV(M_1) = \frac{n^2}{n} = n$.

Vectorization M_2: Treat the whole matrix as one vector of length n^2. Then,

$$T_2 = 33 + \frac{n^2}{2}$$

This vectorization is $\overline{O}_{n^2}(1)$, and $DV(M_2) = \frac{n^2}{1} = n^2$. Note that both M_1 and M_2 vectorizations are consistent and M_2 is optimal.

The following features about timing are illustrated by this example.

(1) The serial order of computation $O(n^2)$ is reflected in both timings.

(2) Vectorization M_2 is superior to M_1 because it has fewer startup times. Note, however, that this difference shows up in a lower order term of the timing. Therefore, as $n \to \infty$, the two vectorizations become essentially equivalent in an order sense since $\frac{T_1}{T_2} \to 1$. For smaller values of n, the lower terms are more important and $\frac{T_1}{T_2} \to n$.

(3) Since scalar timings for an $C(n^2)$ task would be $T_s = kn^2$, it has no lower order terms and for small values of n could be competitive with vector operations.

(4) For consistent vectorizations, DV(M) is a meaningful rough comparison of the vectorization in terms of computations.

In many applications it will be possible to specify or describe the efficiency of the vectorization of subtasks of the total problem but very difficult to determine the best approach to the overall solution of the problem. For instance, in the problem in this report, the emphasis has been on comparing the vectorization of the task of advancing the solution one step in time. The methods can be compared on this basis, but to specify, in general, the best method when one considers the total number of steps required for each method is difficult, if not impossible, and usually quite problem dependent. Therefore, results have been given in terms of what is reasonably constant for most problems of this type, namely, the computation time required to perform one step toward steady state. Results will be given for total solution time for the three grid sizes computed on the CDC 6600 computer.

The vectorization of the various methods for this problem provides an excellent cross section of the philosophies involved in the selection of a method for the STAR computer. First, there is BR which runs slowly on a serial computer but vectorizes optimally. Then, there is ADI which outperforms BR serially but seemingly does not do well on the STAR computer because of the serial nature of the solution of a tridiagonal system. Two vectorizations of ADI are presented in this report. The first implementation ST utilizes a new parallel algorithm by Stone which was formulated to solve each tridiagonal system in a parallel fashion. This vectorization is found to be inconsistent.

The second implementation involves no new mathematics but only the recognition that the n tridiagonal systems are identical in form and independent of each other. Thus, the standard algorithm of Thomas can be used to solve all n systems at the same time. This approach is referred to as RT for repeated tasks. In this implementation, the importance of data storage in STAR is emphasized. Finally, there is PI. This stable method is particularly well suited for the STAR computer since the solution is computed by an explicit-type formula which vectorizes very effectively. The vectorization of PI and BR demonstrates the importance of control vectors.

Each vectorized version is compared with the others through estimated STAR timings. The relative speeds of the vectorized versions are compared with the relative speeds of their serial counterparts to demonstrate the effect of vectorization on the methods and also to quantify some of the order concepts developed earlier in the report.

Assumed FORTRAN Extensions

The programs presented are coded in a FORTRAN-like language with extensions for vector operations. Since the language which will actually be used has not been finalized, the code given here is only presumed to be representative of the final version. Several liberties will be taken with the code in order to make it more readable. These will be pointed out. A description of the FORTRAN extensions used follows.

Implied DO

A sequence of elements from an array A can be specified by an implied reference A(M1:M2:M3) where M1, M2, M3 have the same meaning as they do in the DO statement DO 50 I = M1, M2, M3. All vector operations must involve consecutive locations in core. Therefore, it is presumed that the reference A(I:J), which represents A(I), A(I+1), . . ., A(J), will generate vector operations, whereas A(I:J:2), which represents A(I), A(I+2), . . ., A(J), will generate scalar code. It is also possible to use implied DO references within multidimensional arrays as long as the reference occurs only in one of the indices, for example, A(I,1:M) or A(1:N,J). Note also that if the STAR computer stores arrays by columns first, then the latter reference is to consecutive locations and therefore can be considered to be a vector, whereas the former reference cannot.

BIT

BIT is a type statement identifying a variable or array of variables each to be one bit long.

CTRL

The STAR computer has in its hardware instruction set the capability to use a control vector with its normal vector instructions. The control vector is a string of bits

where each consecutive bit corresponds to a consecutive element of a vector generated or computed in some vector operation. If a bit is a 1, the corresponding element of the result vector is stored. If the bit is a 0, this computation is not stored but merely discarded. The assumed FORTRAN code to use this feature will be

```
A = B  .CTRL.  (EX)
```

where EX is an expression giving rise to an array or vector of results, and B is a bit vector declared in the BIT statement. As an example of its use, consider the following program:

```
DIMENSION A1(6), A2(6), C(6)
BIT B(6)
C = B  .CTRL.  (A1 + A2)
END
```

Assume that A1, A2, C, and B have the following data before computation:

C = [2,2,2,2,2,2]

A1 = [3,3,3,3,3,3]

A2 = [4,4,4,4,4,4]

B = [1,1,0,1,0,1]

Then after computation,

C = [7,7,2,7,2,7]

This feature is desirable in boundary value problems because it allows one to include the boundary points in the arrays of variables and thereby form a vector which includes all grid points. One can then use this long vector in an expression which is valid for the interior points, but not for the boundaries, and yet which does not destroy the good information at the boundaries by overstoring it with a quantity computed using an invalid equation.

Two-dimensional arrays are assumed to be stored by the STAR compiler by columns. For clarity in reading, the capability to reference a two-dimensional array as if it were singly dimensioned is assumed.

Vectorization of Brailovskaya's Method

The equations for BR are of the form

$$\zeta_{i,j}^{K+1} = F_{i,j}\,\zeta_{i,j}^{K} + D_{i,j}\,\zeta_{i-1,j}^{K} + E_{i,j}\,\zeta_{i+1,j}^{K} + H_{i,j}\,\zeta_{i,j-1}^{K} + G_{i,j}\,\zeta_{i,j+1}^{K} \quad \begin{pmatrix} i = 1,n \\ j = 1,n \end{pmatrix}$$

Since each value of vorticity and each coefficient used in the right-hand side of the equation are the result of a calculation at the last time step, it is possible to perform the calculations with vectors of length n^2 by using the following procedure. Let Z be a FORTRAN array containing the values for time K. First, compute the coefficients $F_{i,j}$, using vectors of length n^2, and store the result in the array F. Next, perform the vector multiplication F*Z and store the result in the temporary vector T1. Similarly, compute D*Z with the proper offset in Z and store in T2. Then, add T1 and T2. Continue in this fashion until the entire equation has been computed. This vectorization is obviously $\overline{O}_{n^2}(1)$ and is an optimal vectorization for the task. The coding in appendix A does not perform the computations in precisely this order since it is possible to take advantage of similarities in the two steps of BR.

The STAR FORTRAN coding uses vectors of length n^2 with the storage arrangement as shown in figure 5(a), where the elements are stored consecutively by columns beginning in the lower left-hand corner. The grid as shown in the figure includes the boundaries so that all the information needed to compute the interior points is contained in the vector. All the vectors needed in the computation are used in this manner and since it is necessary to give the starting and the ending location of the vector, the proper offsets must be computed for the vector instructions. This is done at the beginning of the coded example. To compute the first result, Z(MC), four points are needed. The subscripts of these points are used as the beginning subscripts for the implied DO notation. The last result computed is Z(NC). The subscripts of the points it needs are used as the ending subscripts in the implied DO notation. The results are computed in order from Z(MC) to Z(NC).

A bit control vector is used to prohibit storing results on the boundaries. The bits in the control vector corresponding to the boundaries have the value 0, and the remaining bits corresponding to the interior points have the value 1. (See fig. 5(b).)

The STAR FORTRAN coding for the Brailovskaya method is given in appendix A.

Vectorization of Method of Partial Implicitization

Since the general equation for PI has the same form as for BR, it is again an $\overline{O}_{n^2}(1)$ vectorization and, hence, is optimal. In order to maintain an order of n^2 vectorization for the method of partial implicitization, it is necessary to use equation (19) for the computation of all the interior points. In the evaluation of $K_{i,j}$, $L_{i,j}$, $N_{i,j}$, and $O_{i,j}$ appearing in equation (19), the points $\zeta_{i,j+2}$, $\zeta_{i-2,j}$, $\zeta_{i+2,j}$, and $\zeta_{i,j-2}$ are needed. For $\zeta_{i,j}$ adjacent to the boundaries, these points do not exist; therefore, two columns are appended to the original grid points, one on the left side and one on the right side. Now the vector contains an appropriate number of elements and thus equation (19) can always be used without referencing nonexistent points. It does not matter what the contents of the two appended columns are because, as noted earlier, for the points adjacent to the

boundaries, equation (19) is modified as described by equations (20a) to (20d) so that the terms containing the nonexistent points are multiplied by a factor of 0 or are reevaluated. In the FORTRAN program, the terms D, E, G, H, K, L, M, N, and O are all evaluated by using vector instructions of length n^2. Next, the modifications are made according to equations (20a) to (20d). Some of the modifications can use vector instructions of length n, whereas others will be scalar code. Then, equation (19) is computed by using vector instructions of length n^2.

The STAR FORTRAN coding uses vectors with the storage arrangement as shown in figure 6(a), where the elements are stored consecutively by columns beginning in the lower left-hand corner. All vectors have the same length even though they may not be filled completely. This is useful when computing subscripts because corresponding elements in the vectors have the same relative locations. To compute each result, the twelve closest points are needed in the equation. The results are computed in order beginning with Z(MC) and ending with Z(NC).

A bit control vector is used to prohibit storing the results on the boundaries. The bits corresponding to the boundaries and the two appended columns have the value 0 and the remaining interior bits corresponding to the interior points have the value 1. (See fig. 6(b).)

The boundaries are computed as they are in Brailovskaya's method; therefore, the code is not repeated in this example. The STAR FORTRAN coding for the partial implicitization method is given in appendix B.

Vectorization of Alternating Direction Implicit Method by Repeated Tasks

As stated previously, the use of ADI gives rise to n different systems of equations, each tridiagonal and each independent of the results of the others. The task of solving any one of these systems is serial in nature as evidenced by the recursive nature of equations (13a) and (13b). However, it is possible to obtain a degree of parallelism by noting that each task has n-fold repetitiveness in that the operations required to solve the first system are repeated for the other n-1 systems. Therefore, by correctly arranging the coefficients in storage, Thomas' algorithm can be used in the vector mode. For instance, if C_i is assumed to be a STAR vector composed of the following elements (see fig. 7)

$$C_i = \begin{bmatrix} C_{i,1} \\ C_{i,2} \\ \cdot \\ \cdot \\ \cdot \\ C_{i,n} \end{bmatrix} \qquad (i = 1, 2, \ldots, n)$$

and the same is done for A_i, D_i, then Thomas' algorithm can be used in the same form as equations (13a) and (13b) except that each operation is now a vector operation of length n. This vectorization is then on $\overline{O}_n(n)$ vectorization of an $O(n^2)$ task.

To actually implement this idea, one has to be certain that the coefficients that are to be used as vectors are stored consecutively. Since the coefficients must be computed and each is the result of an operation on elements of indexed arrays, it seems possible and is desirable to compute and store the coefficients by using vector operations.

In order to illustrate the importance of making the correct decision about data organization in vectorizing the problems, consider the specific grid in figure 5(a) for n = 5. Assume that the FORTRAN vector PSI and ZETA contain

$$\text{PSI} = \left[\psi_1,\ \psi_2,\ \psi_3, \ldots,\ \psi_{49}\right]$$

and

$$\text{ZETA} = \left[\zeta_1,\ \zeta_2,\ \zeta_3, \ldots,\ \zeta_{49}\right]$$

and that the first step is implicit in the y-direction. Then, for example,

$$C_1 = \begin{bmatrix}\Delta t\\ \Delta t\\ \Delta t\\ \Delta t\\ \Delta t\end{bmatrix} + \begin{bmatrix}\text{CON}\\ \text{CON}\\ \text{CON}\\ \text{CON}\\ \text{CON}\end{bmatrix} * \left\{ \begin{bmatrix}\psi_{16}\\ \psi_{23}\\ \psi_{30}\\ \psi_{37}\\ \psi_{44}\end{bmatrix} - \begin{bmatrix}\psi_{2}\\ \psi_{9}\\ \psi_{16}\\ \psi_{23}\\ \psi_{30}\end{bmatrix} \right\}$$

where $\text{CON} = N_{Re} * \Delta t/4$.

However, none of these operations are vector since the ψ's in the operations indicated are not stored consecutively. However, if the step is taken implicit in the x-direction, the following computations for C_1 are obtained:

$$C_1 = \begin{bmatrix}\Delta t\\ \Delta t\\ \Delta t\\ \Delta t\\ \Delta t\end{bmatrix} - \begin{bmatrix}\text{CON}\\ \text{CON}\\ \text{CON}\\ \text{CON}\\ \text{CON}\end{bmatrix} * \left\{ \begin{bmatrix}\psi(10)\\ \psi(11)\\ \psi(12)\\ \psi(13)\\ \psi(14)\end{bmatrix} - \begin{bmatrix}\psi(8)\\ \psi(9)\\ \psi(10)\\ \psi(11)\\ \psi(12)\end{bmatrix} \right\}$$

All these operations are vector operations for ψ stored as indicated. The conclusion is that for the assumed implicitness in the x-direction, it is necessary that the two-dimensional array of ψ and ζ variables be stored by columns of the grid. The opposite is true when the implicitness is in the y-direction. This last fact forces a rearrangement of the PSI array each time the direction is alternated. The rearrangement is a fairly expensive task but is necessary in using this vectorization of the ADI method. Note that in the discussion of Stone's algorithm, the opposite correlation between direction of implicitness and direction of storage is desirable.

It should be pointed out that the rearrangement of the vectors is not only expensive computationally, it also could be slowed considerably because of the paging system of storage in the STAR computer. Information is brought from the disk to core in pages. If, in one sweep, a row of vorticities is on one page, then a column is on many different pages. The necessity to bring many pages in and out of core to reference a column represents an overhead to the rearrangement that is not shown in the vector timings and could be quite significant.

The following comments help to make the STAR FORTRAN coding for the ADI method presented in appendix C more readable:

(1) By inserting just a small bit of logic and changing a few signs, essentially the same code can be used for the solution in both directions.

(2) Two arrays DER1 and DER2 are used to store the ψ derivatives.

$$\text{DER2(I)} = \text{PSI(I + 1)} - \text{PSI(I - 1)} \quad * \text{CON}$$

$$\text{DER1(I)} = \text{PSI(I + N)} - \text{PSI(I - N)} \quad * \text{CON}$$

where, when implicit in x-direction,

$$\text{CON} = \frac{N_{Re}\,\Delta t}{4}$$

and when implicit in y-direction,

$$\text{CON} = -\frac{N_{Re}\Delta t}{4}$$

Therefore, when implicit in x-direction,

$$\text{DER2(I)} = \frac{N_{Re} h\,\Delta t}{2}\,\psi_y(I)$$

$$\text{DER1(I)} = \frac{N_{Re} h\,\Delta t}{2}\,\psi_x(I)$$

After the rearrangement of the ψ vector for the implicitness in the y-direction

$$DER2(I) = \frac{-N_{Re}h\,\Delta t}{2}\psi_x(I)$$

$$DER1(I) = \frac{-N_{Re}h\,\Delta t}{2}\psi_y(I)$$

(3) When the ψ's are stored by columns of the grid, the DER arrays do not include the left and right boundaries. The top and bottom values are computed only because it is desirable to do the computation on long vectors (vectors which include all the interior grid points).

(4) All two-dimensional FORTRAN arrays are assumed to be stored consecutively by columns so that when they are used in computation, only implied DO in the first index generates vector code.

(5) Since Thomas' algorithm in the scalar form only requires $A_{i+1}, C_{i+1}, R_{i+1}$ after computation is finished with A_i, C_i, R_i, it is possible to have just one A, C, and R vector at each step of the algorithm.

(6) In Thomas' algorithm, Q_j and P_j are computed with a division by w_j. Since vector division is slow compared with multiplication, the two divisions have been replaced with two multiplications by $1/w_j$.

Vectorization of Alternating Direction Implicit Method Using Stone's Algorithm

Stone has proposed a parallel algorithm (RD) for the direct solution of a tridiagonal system of equations. He notes that in the factorization of the matrix A into the product of a lower triangular matrix L and an upper triangular matrix U, the resulting equations are recursive and of the form $x_i = b_i x_{i-1} + c_i x_{i-2}$ for the factorization and $x_i = b_i x_{i-1} + c_i$ for the forward and back substitutions. These operations involve $O(n)$ calculations when done serially. Stone uses recursive doubling to perform the calculations in $\log_2 n$ vector operations. Recursive doubling is the effective subdivision of work in a task into subtasks which have similar form. A simple example is the calculation of the sum of n numbers. This is merely x_n in the sequence given by $x_1 = a_1$; $x_i = x_{i-1} + a_i$ where $i = 2, 3, \ldots, n$. The calculating sequence is illustrated as follows for $n = 4$:

Initially,

$$x = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix}$$

Step 1,

$$x = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} + \begin{bmatrix} 0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} a_1 \\ a_1 + a_2 \\ a_2 + a_3 \\ a_3 + a_4 \end{bmatrix}$$

Step 2,

$$x = \begin{bmatrix} a_1 \\ a_1 + a_2 \\ a_2 + a_3 \\ a_3 + a_4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ a_1 \\ a_1 + a_2 \end{bmatrix} = \begin{bmatrix} a_1 \\ a_1 + a_2 \\ a_1 + a_2 + a_3 \\ a_1 + a_2 + a_3 + a_4 \end{bmatrix}$$

The computation of the vector order of calculation provides some interesting results and are presented here for the summation problem. Let $n = 2^{\ell}$. The first step of the parallel algorithm is a vector add of length $n - 1$; the second step is an add of length $n - 2$; the third step is an add of $n - 4$; and the kth step is a vector add of $n - 2^{k-1}$. There are $\ell = \log_2 n$ such steps. Therefore, the average vector length is given by

$$N_A = \frac{1}{\log_2 n} \sum_{k=0}^{\ell-1} \left(n - 2^k\right) = \frac{1}{\log_2 n}\left[n\ell - \left(2^{\ell} - 1\right)\right]$$

$$= \frac{1}{\log_2 n}\left(n \log_2 n - n + 1\right) = \frac{n\left(\log_2 n - 1\right) + 1}{\log_2 n}$$

Now as n gets increasingly large, $N_A \to n$. Thus, the vector order of computation is $\overline{O}_n\left(\log_2 n\right)$ for a task which is $O(n)$ serially. This means that work which is $O\left(n \log_2 n\right)$ is being done in the vector mode and, although each computation is being done more quickly in the vector mode, there will be some value of n for which this approach is not beneficial and can be beaten even with scalar coding.

Table III contains the estimated timing for the solution for one tridiagonal system with N equations using scalar coding and recursive doubling. Scalar coding is seen to be faster than the parallel algorithm for very small systems ($N < 32$) and for very large systems ($N > 8192$). This is not very surprising. For large values of N, the $O\left(N \log_2 N\right)$ order of work in the vector mode takes longer than the $O(N)$ scalar order of work. When the vectors are short, the startup time for the vector operations is important and accounts

for the ratio in that region. Even in the region where Stone's algorithm is faster, it is only slightly better since once the vectors get long enough to make the startup less important, the higher order of work is being felt. It is concluded that although the recursive doubling approach is probably effective on a computer such as the ILLIAC IV, its advantages on the STAR computer are much less.

The recursive doubling formulas for the solution of the tridiagonal system are not easily presented. A detailed analysis is presented in reference 7 as well as a FORTRAN-like algorithm for carrying out the procedure. That algorithm has been used in appendix D with the assumption of a capability of zero and negative indices.

An interesting feature of the vectorization process for this approach is that the desired relationship between direction of implicitness and the storing of the variables is the exact opposite from that desired for the RT vectorization. Here, since vector operations involve the coefficients of the particular tridiagonal system being solved, it is desired that the unknowns for that line be stored consecutively. For example, in solving implicitly in the y-direction, it is necessary that PSI = $[\psi_1, \psi_2, \psi_3, \ldots, \psi_{49}]$ because here the vector

$$C = \begin{bmatrix} \Delta t \\ \Delta t \\ \Delta t \\ \Delta t \\ \Delta t \end{bmatrix} + \begin{bmatrix} CON \\ CON \\ CON \\ CON \\ CON \end{bmatrix} * \left\{ \begin{bmatrix} \psi_{16} \\ \psi_{17} \\ \psi_{18} \\ \psi_{19} \\ \psi_{20} \end{bmatrix} - \begin{bmatrix} \psi_2 \\ \psi_3 \\ \psi_4 \\ \psi_5 \\ \psi_6 \end{bmatrix} \right\}$$

and all computations involve STAR vectors for PSI as indicated.

Results of Vectorized Methods

Table IV gives a summary of the vectorizations for the four methods. BR and PI both are optimal degree vectorizations. Of the two ADI vectorizations, clearly the vectorization using repeated tasks RT is better. Table V gives a summary of the timings for the several vectorized methods. The graph of these formulas, as a function of n, appears in figure 8. Since BR was the minimum for all n, each time has been normalized to a value of 1.0 for BR. Again, it should be emphasized that this graph reflects only the amount of time required to perform one step of the various algorithms and does not include convergence rates.

The graph reflects the following interesting concepts related to the vectorizations:

(1) The ratio of the two ADI methods to BR varies with n. This type of variation does not exist for a serial computation where the ratio is a constant. This variation is, of course, due to the different degrees of vectorization obtainable in the methods.

(2) The greater values of RT, when n is small, are due to the poorer vectorization in the method. Recall from an earlier discussion that the effect of the degree of vectorization on timing is most significant when n is small.

(3) As $n \to \infty$, the value of RT/BR becomes nearly constant. This constant is just the ratio of the high-order terms in the timing. (See table V.)

(4) The ratio PI/BR is essentially constant since both PI and BR are n^2 degree vectorizations.

(5) ST has the general shape of the tridiagonal timings computed earlier. When n is small, ST is greater due to the weak vectorization $\left(n/\log_2 n\right)$. As n gets larger, this fact becomes less important, but then the inconsistency of the vectorization $\left(n^2 \log_2 n \text{ term}\right)$ begins to dominate.

It is clear that RT would be superior to ST for all n regardless of convergence rates since they are vectorizations of the same method. The choice between the others, of course, will be influenced by the total number of steps required to reach steady state. Table VI presents the predicted normalized computer run time of the three best vectorizations for the grid sizes for which the number of steps to convergence (table I) are known. It is of interest to note that ADI, which was the fastest serial method, is now the slowest with respect to the STAR computer. The PI and BR vectorizations are the fastest and are nearly equivalent since the longer time per iteration of PI has been offset by its fewer steps to convergence. It should be noted that if the trend shown in table I in going from $n = 13$ to $n = 27$ continues (that is, as n doubled, each method took about four times as many steps), ADI with RT will approach the other two in STAR timing since its vectorization becomes relatively better as $n \to \infty$. (See fig. 8.) However, as n gets large, the poor program locality referred to earlier becomes more important even though it doesn't show up in the timings.

Although it is currently impossible to say how important this effect might be, an alternative is suggested for the ADI here to offset this effect should it prove to be large. The poor program locality is caused by the need to transpose the vorticity and stream function values. If this can be avoided, then so is the locality problem. It is recalled from the discussion of the two ADI vectorizations that ADI with RT required storage of the grid opposite to the direction of implicitness, whereas ADI with ST required storage in the same direction of implicitness. The following algorithm may be worth considering for the ADI: Let the grid be stored columnwise.

(1) At step 2K + 1, let the implicitness be in the x-direction. Use the RT vectorization.

(2) At step 2K + 2, let the implicitness be in the y-direction. Then, use ST or scalar solution of the tridiagonal system, or whichever tridiagonal solver proves best to solve each column individually. (Note that even with using the scalar code to solve the systems, some vectorization is present since the coefficients in the matrix equation can be computed in the vector streaming mode.)

(3) Go to step (1).

The necessity for the transpose has been removed and, hence, the locality has been improved. Also, a costly operation $(8n^2)$ has been eliminated from the timings for the method. The timing for such a method would be approximately the average of the two ADI vectorizations minus $8n^2$ for the transpose operation.

CONCLUDING REMARKS

From the viewpoint of seeing as many facets of the vectorization process as possible, the following benefits were obtained from this study of a specified fluid flow problem:

(1) Examples of optimal vectorizations (Brailovskaya's method (BR) and method of partial implicitization (PI)) and the importance of control vectors in achieving this optimality.

(2) Examples of two completely different approaches to vectorizing a sequential problem (solving a series of tridiagonal systems of equations):

(a) The first approach (ST) uses a new mathematical algorithm by Stone to induce vectorization. It also demonstrates the effect of an inconsistent vectorization on the STAR computer.

(b) The second approach (RT) takes advantage of the repeated and independent nature of the task to obtain the vectorization using the usual serial algorithm. Both approaches serve to illustrate the importance of data management.

(3) An example of a new method (that is, PI) whose theoretical properties are most advantageous on the STAR computer.

(4) A feeling, in an order of magnitude sense, for the effect of different degrees of vectorization.

(5) The dependence on the number of grid points of the relative efficiency of the several methods.

(6) A suggested approach to eliminate the locality problem in the alternating direction implicit method (ADI).

Although it is not reasonable to draw global conclusions about results generated from only one problem for only three different grid sizes, some conclusions can be reached about the three methods (BR, PI, and RT) for this problem. In a comparison of the two stable (theoretically) methods, PI performed almost as well as ADI as regards number of steps and since the vectorization is better for a small number of grid lines in each direction and has no locality problems for a large number of grid lines in each direction, PI would be preferred. PI has a slight advantage over BR in the vectorized form but BR will be easier to adapt for less regular regions. However, the theoretical stability characteristics of PI makes it an interesting method to consider for use with the STAR computer.

Langley Research Center,
National Aeronautics and Space Administration,
Hampton, Va., March 29, 1974.

APPENDIX A

A STAR CODING FOR THE BRAILOVSKAYA METHOD

The program listing for Brailovskaya's method is presented in this appendix and uses an assumed FORTRAN-like language with extensions for vector operations.

```
C
C            BRAILOVSKAYA   METHOD
C
C    A ONE-DIMENSIONED VECTOR IS USED
C
C    ZO     RESULT VORTICITY VECTOR AT TIME, T
C    ZBAR   INTERMEDIATE RESULT VECTOR
C    Z      RESULT VECTOR AT TIME, T + DELT
C    NSQ    TOTAL NUMBER OF ELEMENTS IN VECTOR  (INCLUDES BOUNDARIES)
C    N      NUMBER OF ELEMENTS IN ONE COLUMN OF GRID
C    BZ     BIT CONTROL VECTOR WHICH PROHIBITS STORAGE ON BOUNDARIES
C             THE Z BOUNDARIES ARE COMPUTED FIRST, WHEN THE INTERIOR Z ELEMENTS
C             ARE COMPUTED BZ DOES NOT ALLOW THE NEW Z ELEMENTS TO BE STORED
C             AT THE BOUNDARIES
C    PSI    STREAM FUNCTION VECTOR
C    TEMP,      T2, T3   ARE TEMPORARY VECTORS USED IN THE CALCULATIONS
C    H      =1/(N-1)
C
      DIMENSION Z(NSQ), ZO(NSQ), ZBAR(NSQ), PSI(NSQ), TEMP(NSQ)
     1 , T2(NSQ), T3(NSQ)
      BIT  BZ(NSQ)
C
C  COMPUTE CONSTANTS
C
      HSQ = H*H
      CON1 = DELT/HSQ
      CON3 = R * DELT/ (4.0*HSQ)
      CON5 = 2.0/HSQ
      NM1=N-1
C
C  COMPUTE OFFSETS
C
C  BEGINNING SUBSCRIPT
      MB = N+1
      MC = N+2
      MA = N+3
      MR = MC+N
C  ENDING SUBSCRIPT
      NC = (N-1)*N - 1
      NB = NC-1
      NA = NC+1
      NR = NC+N
      NL = NC-N
  100 CONTINUE
C
```

APPENDIX A – Concluded

```
C COMPUTE PSI
C
C EVALUATE BOUNDARY CONDITIONS
      Z(2:NM1) = -CON5* PSI(2:NM1)
      NN = NA+2
      N1 = NL+3
      Z(NN:NR) = -CON5* PSI(N1:NC)
      ZBAR(2:NM1) = Z(2:NM1)
      ZBAR(NN:NR) = Z(NN:NR)
      N1M1 = N1-1
      Z(MB:N1M1:N) =-CON5*PSI(MC:NL:N)
      NN= 2*N
      N1 = NN+1
      Z(NN:NA:N) = CON5*(-PSI(N1:NC:N) + H)
      ZBAR(MB:N1M1:N) = Z(MB:N1M1:N)
      ZBAR(NN:NA:N) = Z(NN:NA:N)
C
C EVALUATE TEMPORARY VECTORS
C
      T2(MC:NC)= PSI(MA:NA) - PSI(MB:NB)
      T3(MC:NC)= PSI(MR:NR) - PSI(2:NL)
      TEMP(MC:NC)= ZO(MC:NC) + CON1* (ZO(MR:NR) - 4.0*ZO(MC:NC) + ZO(2:NL)
     1  +ZO(MA:NA) + ZO(MB:NB))
C
C COMPUTE INTERMEDIATE STEP
C
      ZBAR(MC:NC) = BZ(MC:NC) .CTRL. ( TEMP(MC:NC) - CON3* (T2(MC:NC)*
     1  (ZO(MR:NR)-ZO(2:NL))  -T3(MC:NC)* (ZO(MA:NA)-ZO(MB:NB)) ))
C
C COMPUTE SOLUTION RESULT
C
      Z(MC:NC) = BZ(MC:NC) .CTRL. ( TEMP(MC:NC) - CON3* (T2(MC:NC) *
     1  (ZBAR(MR:NR)-ZBAR(2:NL)) - T3(MC:NC)*(ZBAR(MA:NA)-ZBAR(MB:NB))))
      END
```

APPENDIX B

A STAR CODING FOR THE METHOD OF PARTIAL IMPLICITIZATION

The program listing for the method of partial implicitization is presented in this appendix and uses an assumed FORTRAN-like language with extensions for vector operations.

```
      HSQ = H**2
      CON = R*DELT * H /2.0
      F = -HSQ -4.0*DELT
      FSQ = F**2
      DX(MC:NC)= CON * (PSI(NR:NR) - PSI(ML:NL))
      DY(MC:NC)= CON * (PSI(MA:NA) -PSI(MB:NB))
      D(MC:NC) = DY(MC:NC) + DELT
      E(MC:NC) =-DY(MC:NC) + DELT
      G(MC:NC) = DX(MC:NC) + DELT
      H(MC:NC) =-DX(MC:NC) + DELT
      TEMP(MLM1:NRM1)= -HSQ* ZC(MLM1:NRM1)
C
C COMPUTE COEFFICIENTS AND TERMS USED IN THE GENERAL EQUATION
C
      DZ(2:NLP1)=D(ML:NA)*ZO(2:NLP1)
      EZ(MRM1:N2R)=E(MB:NR)*ZO(MRM1:N2R)
      GZ(MLP1:NRP1)=G(ML:NR)*ZC(MLP1:NRP1)
      HZ(MLM1:NRM1)=H(ML:NR)*ZC(MLM1:NRM1)
      AK(MC:NC) = TEMP(MA:NA) -DZ(MLP1:NLP1) - EZ(MRP1:NRP1)- GZ(M2A:N2A)
      AL(MC:NC) = TEMP(ML:NL) -EZ(M2L:N2L) -GZ(MLP1:NLP1) -HZ(MLM1:NLM1)
      AN(MC:NC) = TEMP(MR:NR) -EZ(M2R:N2R) -GZ(MRP1:NRP1) -HZ(MRM1:NRM1)
      AM(MC:NC) = TEMP(MC:NC)
      J(MC:NC) = TEMP(MB:NB) - DZ(MLM1:NLM1) -EZ(MPM1:NRM1)- HZ(M2B:N2B)
C
      DE(MC:NR) =D(MC:NR) *E(ML:NC)
      HG(MC:NA) =H(MC:NA) *G(MB:NC)
      DENJ4(MC:NC)= DE(MR:NR) + HG(MC:NC) + DE(MC:NC) + HG(MA:NA)
      ANUM(MC:NC) = AM(MC:NC)*E(ML:NC) + AC(MC:NC)*H(MC:NC)+AL(MC:NC)*D(MC:NC)
     1 + AK(MC:NC)* G(MC:NC)
C
C COMPUTE  COEFFICIENTS FOR THE POINTS ADJACENT TO THE BOUNDARIES
C      (TWO-DIMENSIONAL NOTATION IS USED HERE FOR EASIER READING)
C
C FOUR CORNERS
C       TOP LEFT
      AM(N-1,3)= AM(N-1,3) -G(N-1,3)*ZO(N,3) -D(N-1,3)*ZC(N-1,2)
      G(N-1,3)=0.0
      D(N-1,3)=0.0
C       TOP RIGHT
      AM(N-1,N)= AM(N-1,N) -G(N-1,N)*ZO(N,N) -E(N-1,N)*ZC(N-1,N+1)
      E(N-1,N) =0.0
C       BOTTOM LEFT
      AM(2,3)= AM(2,3)-D(2,3)*ZO(2,2) -H(2,2)*ZO(1,3)
      D(2,3)=0.0
      H(2,3)=0.0
```

APPENDIX B – Continued

```
C
C          METHOD OF PARTIAL IMPLICITIZATION
C
C    ONE-DIMENSIONED VECTOR IS USED
C
C    ZO     RESULT VORTICITY VECTOR AT TIME, T
C    Z      RESULT VECTOR AT TIME, T + DELT
C    N      NUMBER OF ELEMENTS IN ONE COLUMN OF GRID
C    N2     TOTAL NUMBER OF ELEMENTS IN VECTOR    = N**2 + 2N
C                (INCLUDES BOUNDARIES AND APPENDED COLUMNS)
C    PSI    STREAM FUNCTION VECTOR
C    H      =1/(N-1)
C    DX,DY,D,E,G,H,TEMP,AK,AL,AM,AN,O,DENOM,ANUM      TEMPORARY VECTORS USED AS
C                TERMS IN THE GENERAL EQUATION
C    BZ     BIT CONTROL VECTOR WHICH PROHIBITS STORAGE ON THE BOUNDARIES
C                THE Z BOUNDARIES ARE COMPUTED FIRST, WHEN THE INTERIOR Z ELEMENTS
C                ARE COMPUTED BZ DOES NOT ALLOW THE NEW Z ELEMENTS TO BE STORED
C                AT THE BOUNDARIES
C
      DIMENSION Z(N2),ZO(N2),TEMP(N2),DY(N2),DX(N2),D(N2),E(N2),G(N2),H(N2),
     1  AK(N2),AL(N2),AM(N2),AN(N2),O(N2),DENOM(N2),ANUM(N2)
     1  ,BZ(N2),EZ(N2),GZ(N2),HZ(N2),CE(N2),HG(N2)
      BIT  BZ(N2)
C
C COMPUTE OFFSETS
C
C  BEGINNING SUBSCRIPT
C
      MC = 2*N+2
      MB = MC-1
      MLP1= MA-N
      ML = MC-N
      MLM1= MB-N
      MRM1= MB+N
C
C   ENDING SUBSCRIPT
C
      NC = N**2 - 2
      NA = NC+1
      NR = NC+N
      N2K = NR+N
      NRP1= NR+1
      NRM1= NR-1
      NLP1= NL+1
C
```

APPENDIX B – Concluded

```
C       BOTTOM RIGHT
        AM(2,N)= AM(2,N)- E(2,N)*ZO(2,N+1)- H(2,N)*ZO(1,N+1)
        E(2,N  )=0.0
        H(2,N  )=0.0
C
C BOTTOM POINTS ADJACENT TO BOUNDARY   AND  TOP
C
        DO 200 J=4,N-1
        AM(2,J) = A1(2,J) -H(2,J)* ZO(1,J)
        H(2,J) = 0.0
        AM(N-1,J) = A1(N-1,J) - G(N-1,J) *ZO(N,J)
        G(N-1,J) =0.0
  200 CONTINUE
C
C RIGHT  AND LEFT POINTS ADJACENT TO BOUNDARIES
C
        DO 210 I=3,N-2
        AM(I,3)=  AM(I,3) -D(I,3)*ZO(I,2)
        D(I,3)=0.0
        AM(I,N)= AM(I,N)- E(I,N)*ZO(I,N+1)
        E(I,N)=0.0
  210 CONTINUE
C
C COMPUTE GENERAL EQUATION
C
        Z(MC:NC)=BZ(MC:NC) .CTRL.((F*AM(MC:NC)-ANUM(MC:NC))/(FSQ-DENOM(MC:NC)))
        END
```

APPENDIX C

A STAR CODING FOR THE ALTERNATING DIRECTION IMPLICIT METHOD BY REPEATED TASKS

The program listing for ADI by repeated tasks is presented in this appendix and uses an assumed FORTRAN-like language with extensions for vector operations.

```
C
C       ADI BY REPEATED TASKS
C
C      PSI    STREAM FUNCTION VECTOR
C      Z      VORTICITY VECTOR NOW COMPUTING
C      DER1,DER2  PSI DERIVATIVE VECTORS
C      A,B,C,D    COEFFICINTS IN MATRIX EQUATION
C                 B IS A CONSTANT
C      G,F,W      TEMP VECTORS AS IN THOMAS ALG
C      T          TEMPORARY FOR VORTICITY
C
C      N       NUMBER OF INTERIOR MESH PTS IN 1 LINE
C      R       REYNOLDS NUMBER
C      H       SPATIAL GRID SPACING
C      DT      TIME STEP
       DIMENSION PSI(M,M),Z(M,M),DER1(M,N),DER2(M,N),
      1  A(N),C(N),D(N),W(N),F(N-1,N),G(N,N),T(N,N)
       LOGICAL COLWISE
       HSQ=H*H
       CON=DT*R/4.
       CON1=-2./HSQ
       B=-2.*DT-HSQ
       NP1 = N+1
       M=N+2
       NM1=N-1
       CON2= 2.*DT-HSQ
       COLWISE=.T.
C
C      COLWISE=.T. IMPLIES PSI,Z STORED BY COLS OF GRID
C                  THUS,IMPLICIT IN X-DIRECTION
   11 CONTINUE
C
C      COMPUTE BOUNDARY VALUES OF Z
C
       IF (.NOT. COLWISE) GO TO 10
C
C      PSI,Z STORED BY GRID COLS
C      Z(2:NP1,1) REFERS TO FIRST GRID COLUMN
C
       Z(2:NP1,1)=CON1*PSI(2:NP1,2)
       Z(2:NP1,M)=CON1*PSI(2:NP1,NP1)
       DO 5 J=2,NP1
       Z(1,J)=CON1*PSI(2,J)
     5 Z(M,J)=-CON1*(H-PSIN(NP1,J))
       GO TO 9
```

APPENDIX C – Concluded

```
   10 CONTINUE
C
C      PSI,Z STORED BY GRID ROWS
C      Z(2:NP1,1) REFERS TO FIRST GRID ROW
C
       Z(2:NP1,1)=CON1*PSI(2:NP1,2)
       Z(2:NP1,M)=-CON1* (H-PSI(2:NP1,NP1))
       DO 4 J=2,NP1
       Z(1,J)=CON2* PSI(2,J)
    4  Z(M,J)=CON2* PSI(NP1,J)
    5  CONTINUE
       DER1(1:M*N)=(PSI(2*M+1:M*M)-PSI(1:M*M-2*M ))*CON
       DER2(1:M*N)=(PSI(M+2:M*M-M+1)-PSI(M+M*M-M-1))*CON
C
C      IF COLWISE,DER1=(R*DT*H/2)*D/DX(PSI)
C      IF NOT COLWISE,DER1=-(R*DT*H/2)*D/DY(PSI)
C
C      BEGIN TRI-DIAG SETUP AND SOL
C
       CON3=1./8
       W(1:N)=CON3
       D(1:N)=Z(2:NP1,2)*CON2 + Z(3:M,2)*(-DER1(2:N+1,1)-DT)
     1     +Z(1:N,2)*(DER1(2:NP1,1)-DT)-Z(2:NP1,1)*( DER2(2:NP1,1)+DT)
       G(1:N,1)=D(1:N)*W(1:N)
       DO  1  J= 1,NM1
       C(1:N)= DT-DER2(2:NP1,J)
       F(1:N,J) =C(1:N)*W(1:N)
       D(1:N)=Z(2:NP1,J+2)*CON2+Z(3:M,J+2)*(-DER1(2:NP1,J+1)-DT)
     1       +Z(1:N,J+2)*DER1(2:NP1,J+1)-DT)
       IF( J  .EQ. NM1) D(1:N)=D(1:N)-Z(2:NP1,M)*(-DER2(2:NP1,N)+DT)
       A(1:N)=DT+DER1(2:NP1,J+1)
       TEMP= B- A(1:N)*F(1:N,J)
       W(1:N)=1./TEMP(1:N)
       G(1:N,J+1)=(D(1:N)-A(1:N)*G(1:N,J))*W(1:N)
    1  CONTINUE
C
C      BEGIN BACK SUBSTITUTION
C
       T(1:N,N)= G(1:N,N)
       DO  2  J=1,NM1
       L= N-J
       LP1 =L+1
    2  T(1:N,L)= G(1:N,L)-F(1:N,L) * T(1:N,LP1)
C
C      REARRANGE  Z
C
       DO  3  I =2,NP1
    3  Z(2:NP1,I)= T (I,1:N)
       COLWISE= .NOT. COLWISE
       CON=-CON
C
C      IF CON GT 0   TEST FOR CONVERGENCE
C      CALL POISSON SOLVER  TO  GET  PSI
C
       GO  TO  11
       END
```

APPENDIX D

A STAR CODING FOR THE ALTERNATING DIRECTION IMPLICIT METHOD USING STONE'S ALGORITHM

The program listing for ADI using Stone's algorithm is presented in this appendix and uses an assumed FORTRAN-like language with extensions for vector operations.

```
C
C
C     ADI BY  STONES RECURSIVE DOUBLING
C
C     PSI    STREAM FUNCTION VECTOR
C     Z      VORTICITY VECTOR NOW COMPUTING
C     DER1,DER2  PSI DERIVATIVE VECTORS
C     A,B,C,D    COEFFICIENTS IN MATRIX EQUATION
C                B IS A CONSTANT
C     T          TEMPORARY FOR VORTICITY
C     QI,QIM1,QIM2,AC,TEMP,X,Y    TEMPORARY VECTORS NEEDED FOR
C                                 RECURSIVE DOUBLING
C     U    UPPER DIAGONAL OF FACTORED MATRIX
C     URECIP   RECIPROCAL OF EACH ELEMENT IN LOWER DIAGONAL
C              OF FACTORED MATRIX
C     N       NUMBER OF INTERIOR MESH PTS IN 1 LINE
C     R       REYNOLDS NUMBER
C     H       SPATIAL GRID SPACING
C     DT      TIME STEP
      DIMENSION PSI(M,M),Z(M,M),DER1(M,N),DER2(M,N),
     1  A(N),C(N),D(N),AC(N),QI(N),QIM1(0:N),QIM2(-1:N),TEMP(N)
     2  URECIP(N),M(N),X(N),Y(N), T(N,N)
      REAL M
      LOGICAL COLWISE
      HSQ=H*H
      CON=DT*R/4.
      CON1=-2./HSQ
      NP1 = N+1
      M=N+2
      NM1=N-1
      B=-2.*DT-HSQ
      BSQ= B*B
      CON2= 2.*DT-HSQ
      COLWISE=.T.
C
C     COLWISE=.T. IMPLIES PSI,Z STORED BY COLS OF GRID
C                   THUS,IMPLICIT IN   Y-DIRECTION
   11 CONTINUE
C
C     COMPUTE BOUNDARY VALUES OF Z
C
      IF (.NOT. COLWISE) GO TO 10
C
C     PSI,Z STORED BY GRID COLS
C     Z(2:NP1,1) REFERS TO FIRST GRID COLUMN
```

APPENDIX D — Continued

```
      I=1
   21 CONTINUE
      I=I+I
      IF( I  .GT. N/2) GO TO 22
      TEMP(I-1:N)=QI(I-1:N)*QIM1(0:N-I+1)
     1     +AC(1:N-I+2)*QIM2(I-1:N)*QIM2(-1:N-I)
      QIM1(I:N)= QI(I:N)*CIM1(0:N-I)+AC(1:N-I+1)*QIM2(-1:N-I-1)
     1    *QIM1(I:N)
      QIM2 (I-1:N) =TEMP(I-1: N)
      QI( I+1: N)= B*QI*1(I:N-1)+AC(I+1:N)*CIM2(I-1:N-2)
      GO  TO  21
   22 CONTINUE
      URECIP(2:N)=Q(1:N-1)/Q(2:N)
      M(2: N)=A(2:N)*URECIP(1:N-1)
C
C     FORWARD  SUBSTITUTION
C
      Y(1:N)= D(1:N)
      TEMP(2:N)=-M(2:N)
      I=1
   20 CONTINUE
      IF( I.GT. N/2) GO TO 30
      Y(I+1:N)=Y(I+1:N)+ Y(1:N-I)*TEMP(I+1:N)
      TEMP(I+1:N)=TEMP(I+1:N)*TEMP(1:N-I)
      I=I +I
      GO  TO   20
C
C     BACK SUBSTITUTION
C
   30 CONTINUE
      X(1:N)= Y(1:N)*URECIP(1:N)
      TEMP(1:N-1)=-C(1:N-1)*URECIP(1:N)
      I=1
   40 CONTINUE
      IF( I .GT. N/2) GO TO 50
      X(1:N-I)= X(1:N-I) + X(I+1:N)*TEMP(1:N-I)
      TEMP(1:N-I)= TEMP(1:N-I) * TEMP (I+1:N)
      I=I +I
      GO  TO  40
   50 T(1:N,J)=X(1:N)
C
C     INTERCHANGE ROWS AND COLS
C
      DO  3  I=2,NP1
    3 Z(2:N-1,I)= T(I-1,1:N)
C
      Z(2:NP1,1)=CON1*PSI(2:NP1,2)
      Z(2:NP1,M)=CON1*PSI(2:NP1,NP1)
      DO 5 J=2,NP1
      Z(1,J)=CON1*PSI(2,J)
    5 Z(M,J)=-CON1*(H-PSIN(NP1,J))
      GO TO 9
   10 CONTINUE
C
```

APPENDIX D – Concluded

```
C     PSI,Z STORED BY GRID ROWS
C     Z(2:NP1,1) REFERS TO FIRST GRID ROW
C
      Z(2:NP1,1)=CON1*PSI(2:NP1,2)
      Z(2:NP1,M)=-CON1* (H-PSI(2:NP1,NP1))
      DO 4 J=2,NP1
      Z(1,J)=CON2* PSI(2,J)
    4 Z(M,J)=CON2* PSI(NP1,J)
    9 CONTINUE
      DER1(1:M*N)=(PSI(2*M+1:M*M)-PSI(1:M*M-2*M ))*CON
      DER2(1:M*N)=(PSI(M+2:M*M-M+1)-PSI(M:M*M-M-1))*CON
C
C     IF COLWISE,DER1=(R*DT*H/2)*D/DX(PSI)
C     IF NOT COLWISE,DER1=-(R*DT*H/2)*D/DY(PSI)
C
C     BEGIN TRI-DIAG SETUP AND SOL
C
      DO  1  J=1,N
      JP1=J+1
      JP2=J+2
      D(1:N)=Z(2:NP1,JP1)*CON2 +Z(2:NP1,JP2)*(DER(2:NP1,J)-DT)
     1       +Z(2:NP1,J)*(-DER2(2:NP1,J)-DT)
      D(1)=D(1)-Z(1,JP1)*(-DER1(2,J)+DT)
      D(N)=D(N)- Z(M,JP1)*(DER1(NP1,J)+DT)
      C(1 NM1)=DT +DER1(2:N,J)
      A(2:N)= DT- DER1(3 NP1,J)
      AC(2:N)= -A(2:N)*C(1:NM1)
      AC(1)=0.
      QIM2(-1: N)=1.
      QIM1(1:N)= B
      QIM1(0)=1.
      QI( 2:N) = BSQ + AC(2: N)
      QI(1) =B
C
C     BEGIN FACTORIZATION
C
      COLWISE= .NOT. COLWISE
      CON=-CON
C
C     IF CON GT 0   TEST FOR CONVERGENCE
C     CALL POISSON SOLVER  TO  GET  PSI
C
      GO  TO  11
      END
```

REFERENCES

1. Mills, Ronald D.: Numerical Solutions of the Viscous Flow Equations for a Class of Closed Flows. J. Roy. Aeronaut. Soc., vol. 69, no. 658, Oct. 1965, pp. 714-718; Correction, vol. 69, no. 660, Dec. 1965, p. 880.

2. Burggraf, Odus R.: Analytical and Numerical Studies of the Structure of Steady Separated Flows. J. Fluid Mech., vol. 24, pt. 1, Jan. 1966, pp. 113-151.

3. Carter, James E.: Numerical Solutions of the Navier-Stokes Equations for the Supersonic Laminar Flow Over a Two-Dimensional Compression Corner. NASA TR R-385, 1972.

4. Richtmyer, Robert D.; and Morton, K. W.: Difference Methods for Initial-Value Problems. Second ed., Interscience Publ., c.1967.

5. George, J. Alan: Block Elimination on Finite Element Systems of Equations. Sparse Matrices and Their Applications, Donald J. Rose and Ralph A. Willoughby, eds., Plenum Press, 1972, pp. 101-114.

6. Graves, Randolph A., Jr.: Partial Implicitization. J. Comput. Phys., vol. 13, no. 3, Nov. 1973, pp. 439-444.

7. 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.

TABLE I.- SERIAL COMPUTER RESULTS

Method	n	Number time steps	Time steps normalized by ADI time steps	CDC 6600 CPU time per step, sec	Total CPU time, sec
BR	13	136	2.83	0.0165	2.24
	20	414	3.23	.037	15.32
	27	560	2.86	.065	36.4
PI	13	57	1.19	0.036	2.05
	20	152	1.19	.083	12.62
	27	223	1.14	.142	31.67
ADI	13	48	1	0.018	.86
	20	128	1	.041	5.25
	27	196	1	.075	14.7
FI	13	34	0.71	0.39	13.3
	20	86	.67	1.81	155.7
	27	Not computed			

TABLE II.- ESTIMATED TIMINGS FOR STAR 64-BIT VECTOR OPERATIONS

Operation	Time in clocks
Add, subtract	$33 + \frac{l'}{2}$
Multiply	$38 + l'$
Divide	$87 + 2l'$
Transmit	$30 + \frac{l'}{2}$
Transmit index list	$34 + 8l'$

TABLE III.- RECURSIVE DOUBLING AND SCALAR TIMINGS FOR TRIDIAGONAL SYSTEM

Recursive doubling:

$$T_{RD} = 15N_A \log_2 n + 9.5n - 10N_A + 51 \log_2 n - 3$$

Scalar coding:

$$T_S = 190n$$

n	8	16	32	64	128	256	512	1024	2048	4096	8192	16384
$\log_2 n$	3	4	5	6	7	8	9	10	11	12	13	14
T_{RD}/T_S	1.5	1.1	0.86	0.72	0.67	0.66	0.70	0.75	0.81	0.88	0.95	1.03

TABLE IV.- SUMMARY OF VECTORIZATIONS

Method	Order	Degree
BR	$\overline{O}_{n^2}(1)$	n^2
PI	$\overline{O}_{n^2}(1)$	n^2
ADI using repeated tasks	$\overline{O}_n(n)$	n
ADI using Stone's algorithm	$\overline{O}_n(n \log_2 n)$	$n/\log_2 n$

TABLE V.- TIMINGS FOR THE VECTORIZATIONS

Method	Formula for STAR time in clocks	Number of vector operations and length of vector									
		Add, subtract	Length	Multiply	Length	Divide	Length	Transmit	Length	Transmit indexed list	Length
BR	$15.5n^2 + 799$	15	n^2	8	n^2						
PI	$29.5n^2 + 4n + 1709$	2	n	2	n			2	n		
		26	n^2	14	n^2	1	n^2	1	n^2		
RT	$25.5n^2 + 722n + 142$	9n	n	8n	n	n	n			n	n
		2	n^2	2	n^2						
ST	$15nN_A \log_2 n + 25.5n^2 - 10nN_A + 651n \log_2 n + 305n + 142$	7n	n	7n	n	n	n	4n	n	n	n
		2	n^2	2	n^2						
		$2n \log_2 n$	N_A	$4n \log_2 n$	N_A			$n(\log_2 n - 1)$	N_A		
		$3n(\log_2 n - 1)$	N_A	$8n(\log_2 n - 1)$	N_A						

TABLE VI.- PREDICTED NORMALIZED STAR COMPUTER RUN TIME†

n	BR	PI	ADI with RT
13	136	113	192
20	414	295	448
27	560	430	612

†All entries = (No. steps) * (Normalized time per step)

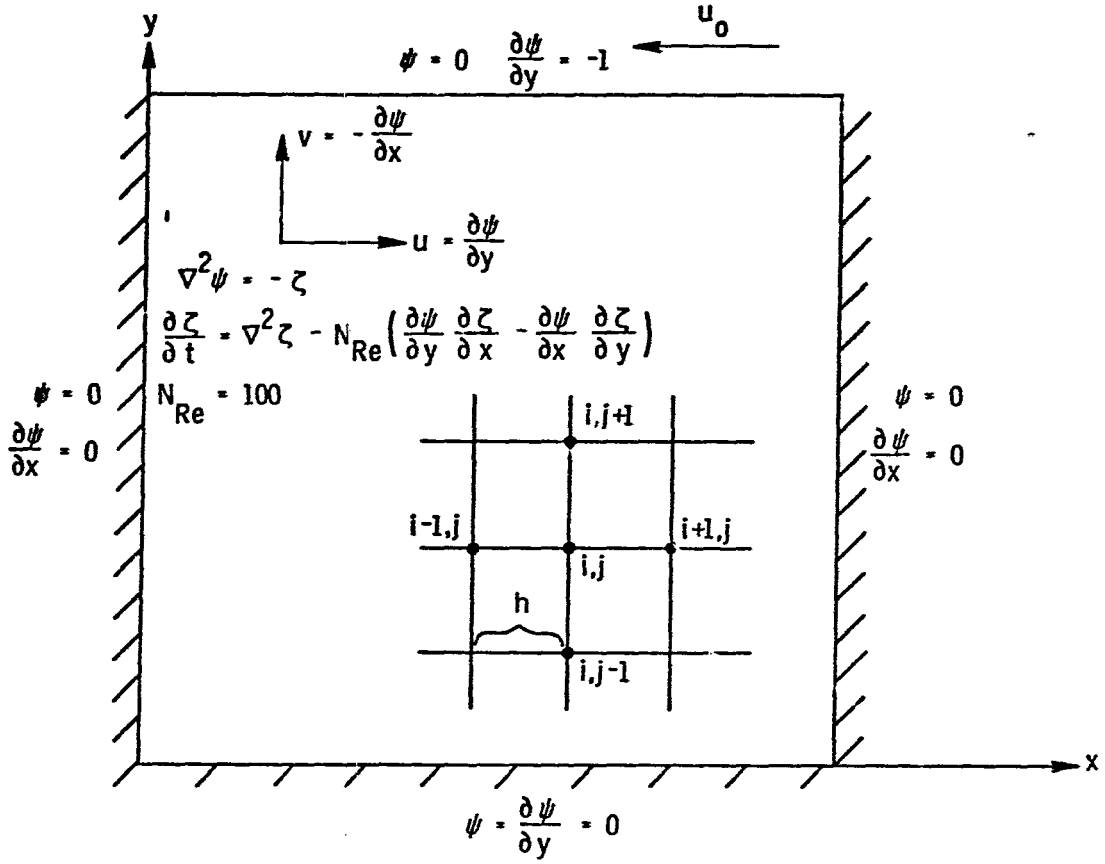


Figure 1.- Geometry and equations for specified problem.

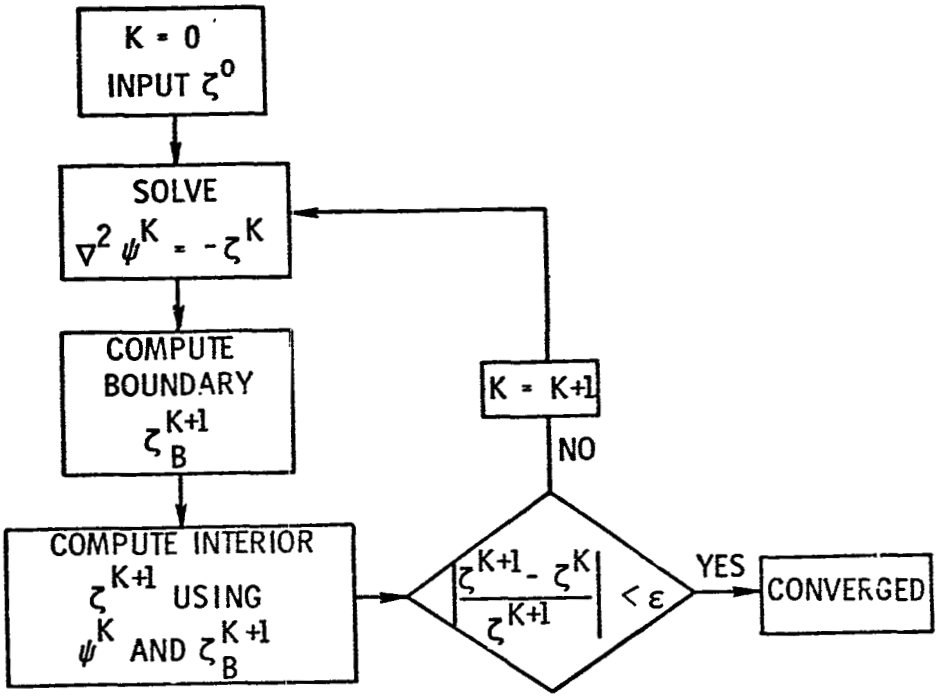
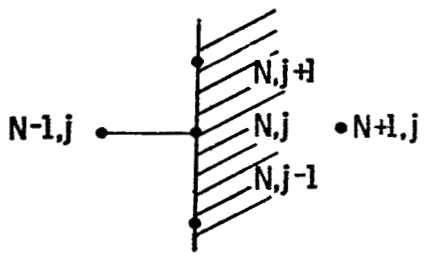


Figure 2.- Program flow chart.



(1) $\frac{\partial \psi}{\partial x}(N,j) = 0$

(2) $\nabla^2 \psi (N,j) = -\zeta_{N,j}$

From (1), $\psi_{N+1,j} = \psi_{N-1,j}$

From (2), $\frac{\psi_{N-1,j} + \psi_{N+1,j}}{h^2} = -\zeta_{N,j}$

Therefore, $\zeta_{N,j} = \frac{-2\psi_{N-1,j}}{h^2}$

Figure 3.- Representative derivation of ζ boundary conditions.

X Original nonzero element
1 Fill from elimination of first n variables

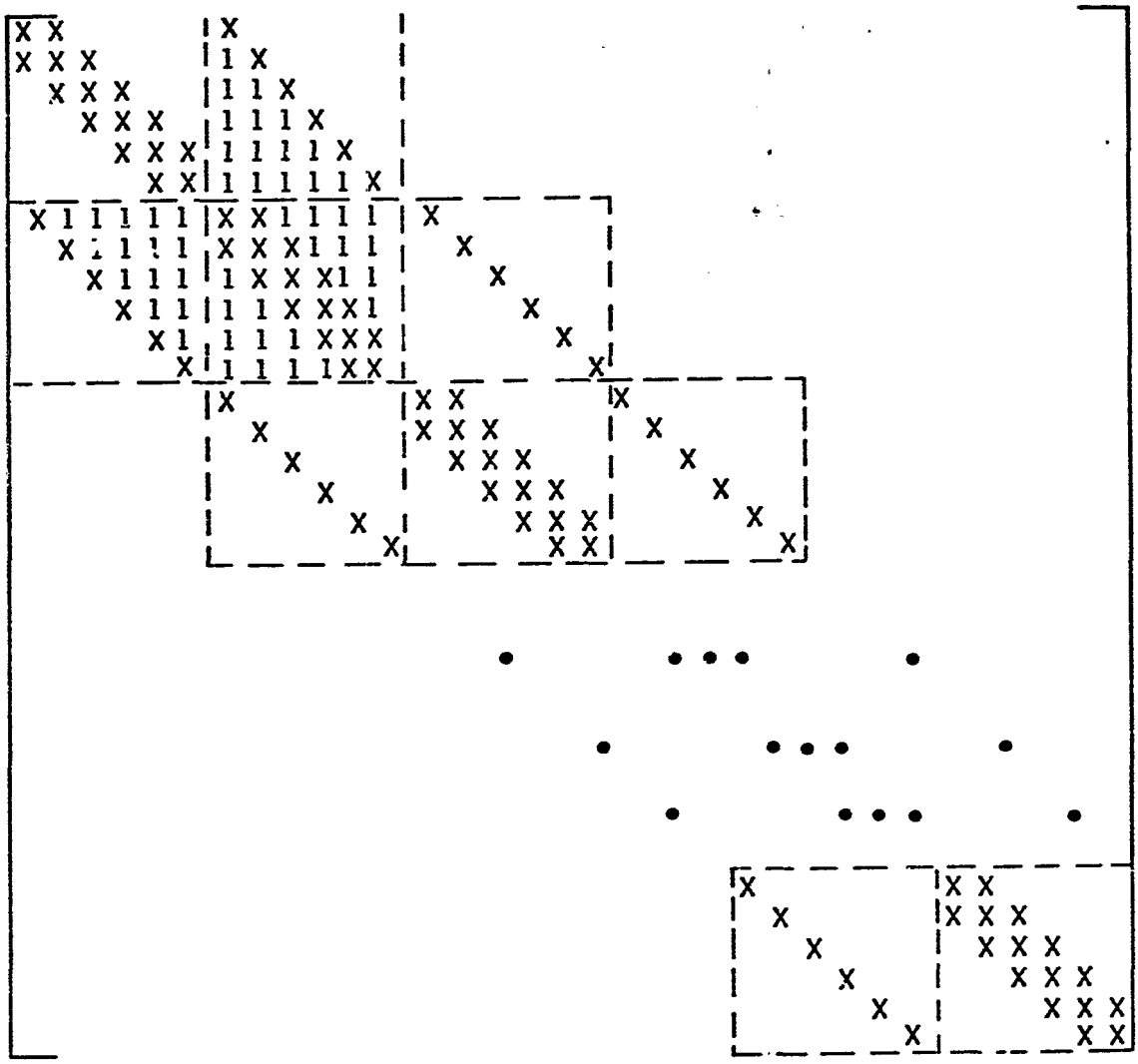
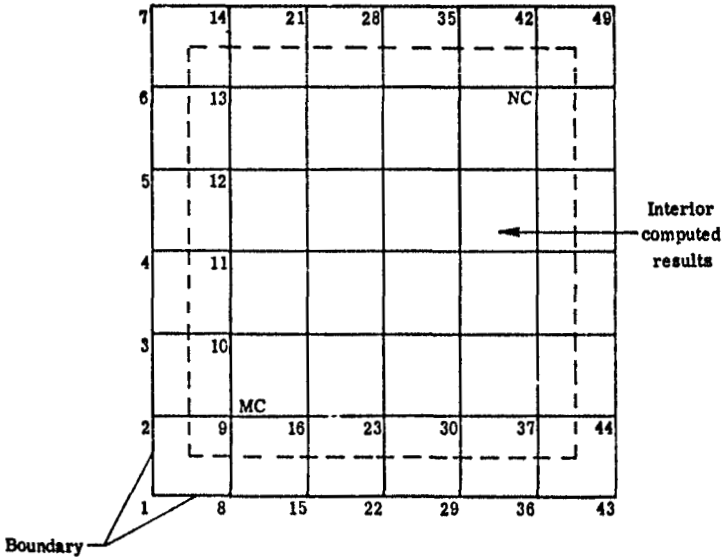


Figure 4.- Banded system with fully implicit method (FI) after elimination of first n variables.

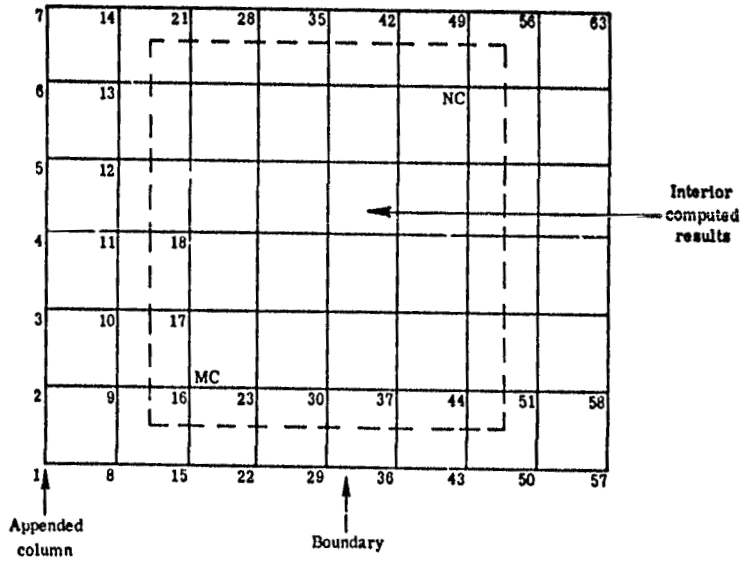


(a) Grid arrangement.

0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	0	0	0	0	0	0

(b) Bit control vector.

Fig. 5.- Grid arrangement and bit control vector for Brailovskaya's method (BR) for n = 5.



(a) Grid arrangement.

0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	0	0
0	0	1	1	1	1	1	0	0
0	0	1	1	1	1	1	0	0
0	0	1	1	1	1	1	0	0
0	0	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0

(b) Bit control vector.

Figure 6.- Grid arrangement and bit control vector for method of partial implicitization (PI) for $n = 5$.

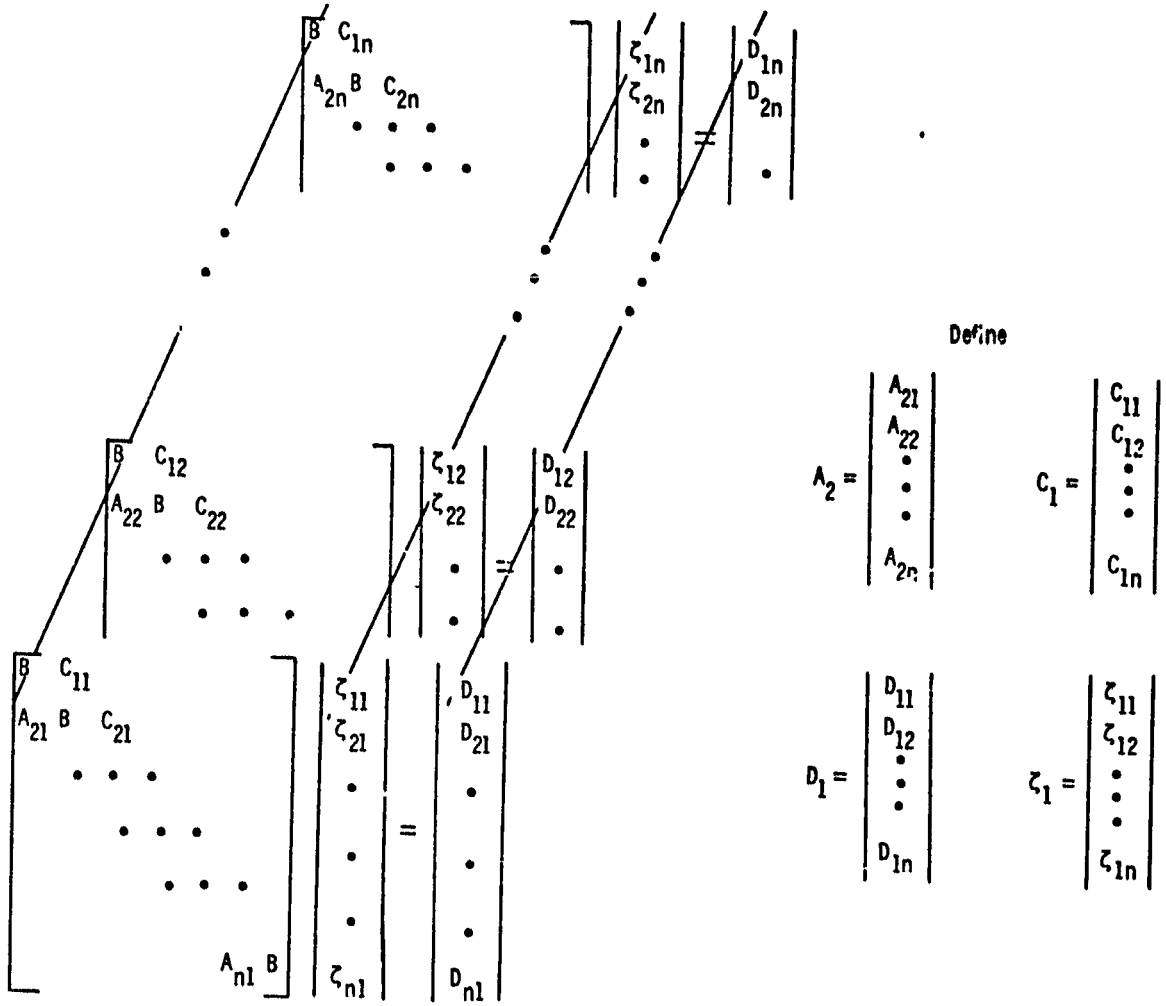


Figure 7.- Vectorization of alternating direction implicit method (ADI) by repeated tasks (RT).

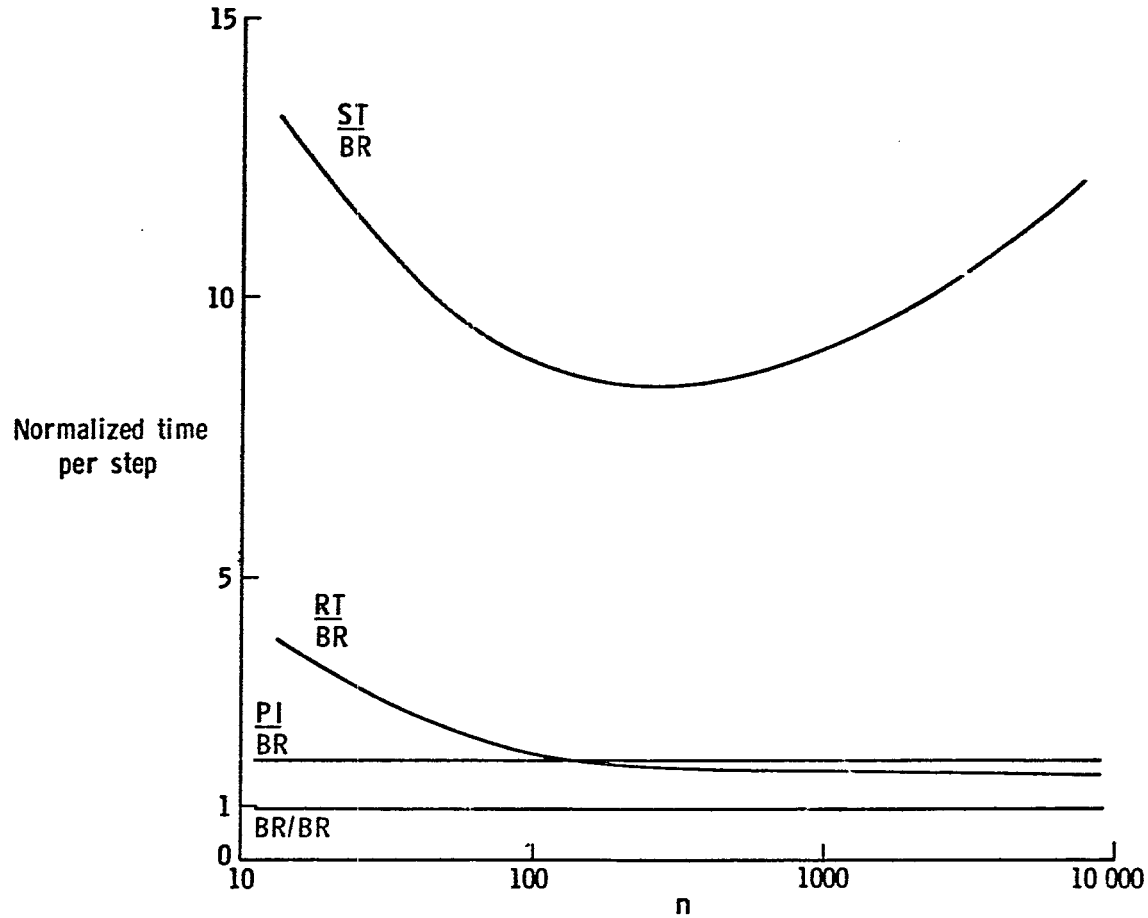


Figure 8.- Vectorized time per step.

END

DATE

FILMED

SEP 16 1974