A computer-oriented description of the Peaceman–Rachford ADI method

By W. A. Murray* and M. S. Lynn†

The Peaceman–Rachford ADI technique is a well-known method for solving, say, large-scale sets of linear equations arising out of the numerical solution by difference methods of second-order elliptic and parabolic partial differential equations. The present paper demonstrates how the algorithm can be conveniently arranged for computational purposes, and analyses the problems concerning the use of serial-access auxiliary storage equipment.

1. Introduction

This paper is to a large extent expository, and is mainly concerned with demonstrating how the Peaceman–Rachford alternating direction implicit method (ADI) can be conveniently arranged into a form which is in general more suitable for computer programming purposes than the mathematical statement of the algorithm. The Peaceman–Rachford (1955) ADI method is a well-known iterative technique for solving large-scale sets of linear equations such as those arising, for example, in the numerical solution by difference methods of elliptic and parabolic partial differential equations [see, for example, Varga (1962), Birkhoff, Varga and Young (1962), Wachspress (1963)].

The derivation of the problems we shall consider is outlined in Section 2. We confine our attention to a "model problem", namely to the solution of the sets of equations arising out of a difference-method solution of the second-order linear elliptic equation (2.1) in two space dimensions, defined on a rectangle, with Dirichlet boundary conditions on the perimeter. A word of explanation is here in order: the analysis we present is by no means restricted in principle to such specialized problems—thus, for example, the techniques may be readily extended to non-rectangular regions, to higher-order problems with more general boundary conditions, to non-uniform grids and, with a corresponding increase in complexity of the algorithm and of its exposition, to a greater number of space dimensions. However, the ideas we wish to exhibit are, we feel, sufficiently well illustrated by the relatively simple model problem.

The Peaceman–Rachford method is defined in Section 3. The difficulty that arises, and which principally motivates this paper, is that the statement (3.1) and (3.2) of the algorithm, while convenient from the point of view of mathematical analysis of such considerations as convergence properties, turns out to be somewhat inconvenient for the purpose of programming the method for a computer, in particular when the size of the problem is so large that all the relevant data cannot be contained within the limits of the high-speed memory, and the use of auxiliary storage becomes necessary.

This discussion is mainly oriented towards serial-access auxiliary storage devices (e.g. magnetic tapes), although similar considerations apply (see Section 5) to random-access devices (e.g. discs). In Section 4 it is shown how the method can be conveniently re-arranged into a form amenable to the use of serial-access equipment, such that operations in "alternating directions", referred to in the name of this class of methods, can in fact be replaced by mathematically equivalent operations in one direction only. This, incidentally, removes a misconception that sometimes seems to arise, namely that ADI methods are not suitable for problems so large as to necessitate auxiliary storage.

Finally in Section 5 we indicate, in the context of magnetic-tape auxiliary storage, how this re-arrangement of the algorithm can be efficiently implemented for a large variety of computer configurations. Since the latter clearly vary considerably in detail, this section is necessarily non-specific and can only exemplify certain general principles.

2. Derivation and properties of equations

In this paper we shall exemplify the ideas we are presenting by considering the systems of linear algebraic equations derived in the numerical solution by difference methods of the linear, second-order, self-adjoint partial differential equation

$$-\frac{\partial}{\partial x}\left[\rho_1(x, y)\frac{\partial U}{\partial x}\right] - \frac{\partial}{\partial y}\left[\rho_2(x, y)\frac{\partial U}{\partial y}\right] + \sigma(x, y)U = \phi(x, y), \quad (x, y)\varepsilon R, \tag{2.1}$$

where R is the interior of the rectangle formed by the lines $x = a, b$ $(a < b)$, $y = \alpha, \beta$ $(\alpha < \beta)$; together with Dirichlet boundary conditions of the form

$$U = \gamma(x, y), \quad (x, y)\varepsilon\Gamma = \bar{R} - R, \tag{2.2}$$

where $\bar{R}$ is the closure of R.

In addition to whatever regularity conditions are necessary to ensure the existence and uniqueness of a solution to (2.1) and (2.2), we necessarily assume that

$$\rho_i(x, y) > 0 \ (i = 1, 2); \quad \sigma(x, y) \geqslant 0; \quad (x, y)\varepsilon R. \tag{2.3}$$

* *Mathematics Division, National Physical Laboratory, Teddington, Mddx.*

† Now at the *IBM Systems Research and Development Center, Kirkeby Center,* 10889 *Wilshire Boulevard, Los Angeles* 24, *California.*

In the numerical solution of this problem, we cover R by a uniform grid, $R(h, k)$, of mesh lines:

$$\begin{aligned} x_i &= a + ih, \quad i = 0(1)n + 1\dagger \\ y_j &= \alpha + jk, \quad j = 0(1)m + 1\dagger \end{aligned} \tag{2.4}$$

where $b - a = (n + 1)h, \quad \beta - \alpha = (m + 1)k.$ (2.5)

Let

$$u_{ij} = U(x_i, y_j), \quad i = 0(1)n + 1, \quad j = 0(1)m + 1 \tag{2.6}$$

and define difference operators

$H\Big($approximating the operator

$$-\frac{\partial}{\partial x}\left[\rho_1\frac{\partial}{\partial x}\right] + \tfrac{1}{2}\sigma \text{ in (2.1)}\Big)$$

and $V\Big($approximating the operator $-\dfrac{\partial}{\partial y}\left[\rho_2\dfrac{\partial}{\partial y}\right] + \tfrac{1}{2}\sigma\Big)$

by

$$\left.\begin{aligned} Hu_{ij} &= -h_{ij}u_{i-1,j} + H_{ij}u_{ij} \\ &\quad - h_{i+1,j}u_{i+1,j} \\ Vu_{ij} &= -v_{ij}u_{i,j-1} + V_{ij}u_{ij} \\ &\quad - v_{i,j+1}u_{i,j+1} \end{aligned}\right\}, i = 1(1)n, j = 1(1)m \quad \begin{matrix}(2.7)\\ \\ \\(2.8)\end{matrix}$$

where

$$h_{ij} = (k/h)\rho_1(x_i - \tfrac{1}{2}h, y_j);$$
$$H_{ij} = h_{ij} + h_{i+1,j} + \tfrac{1}{2}hk\sigma(x_i, y_j) \tag{2.9}$$
$$v_{ij} = (h/k)\rho_2(x_i, y_j - \tfrac{1}{2}k);$$
$$V_{ij} = v_{ij} + v_{i,j+1} + \tfrac{1}{2}hk\sigma(x_i, y_j). \tag{2.10}$$

We now identify u_{ij} with an *approximation* to $U(x_i, y_j)$ satisfying the discretized form of (2.1):

$$Hu_{ij} + Vu_{ij} = hk\phi_{ij}, \quad i = 1(1)n, \quad j = 1(1)m \tag{2.11}$$

where

$$\phi_{ij} = \phi(x_i, y_j), \tag{2.12}$$

together with the boundary conditions

$$u_{ij} = \gamma_{ij} = \gamma(x_i, y_j), \quad (x_i, y_j)\varepsilon\Gamma. \tag{2.13}$$

If we now define the vector $\boldsymbol{u}_H$ by

$$\boldsymbol{u}_H = (u_{11}, u_{21}, \ldots, u_{n1}, u_{12}, u_{22}, \ldots, u_{n2}, \ldots, u_{1m}, u_{2m}, \ldots, u_{nm})^* \tag{2.14}$$

(the subscript H indicates that the ordering is by rows, i.e., horizontally; * denotes (conjugate) transpose) and let $\boldsymbol{H}$ and $\boldsymbol{V}$ be the matrices corresponding to the difference operators defined in (2.7) and (2.8), respectively, determined by this ordering, we may re-write (2.11) and (2.13) as the single matrix equation

$$\boldsymbol{A}\boldsymbol{u}_H = \boldsymbol{f}_H \tag{2.15}$$

where $\boldsymbol{A} = \boldsymbol{H} + \boldsymbol{V}$ (2.16)

and $\boldsymbol{f}_H$ is the vector consisting of the $\{hk\phi_{ij}\}$ together with any terms contributed by the boundary data (2.13).

With this notation and ordering, the matrix $\boldsymbol{H}$ has the following block-diagonal (***diag***) form:

$$\boldsymbol{H} = \boldsymbol{diag}\,\{\boldsymbol{H}_1, \ldots, \boldsymbol{H}_m\} \tag{2.17}$$

where $\boldsymbol{H}_j$ $(j = 1(1)m)$ is the $n \times n$, irreducible, tri-diagonal Stieltjes matrix (Varga, 1962) of the form

$$\boldsymbol{H}_j = \text{tri-diag}\,\{-h_{ij}, H_{ij}, -h_{i+1,j}\} \tag{2.18}$$

$$\overset{\text{(defn)}}{=} \begin{bmatrix} H_{1j} & -h_{2j} & & & 0 \\ -h_{2j} & H_{2j} & -h_{3j} & & \\ \cdot & \cdot & \cdot & & \\ & \cdot & \cdot & \cdot & \\ & -h_{ij} & H_{ij} & -h_{i+1,j} & \\ & & \cdot & \cdot & \cdot \\ & & & \cdot & \cdot \\ 0 & & & -h_{nj} & H_{nj} \end{bmatrix}. \tag{2.19}$$

For later convenience we re-define h_{1j} and $h_{n+1,j}$ by

$$h_{1j} = h_{n+1,j} = 0, \quad j = 1(1)m. \tag{2.20}$$

Furthermore, $\boldsymbol{V}$ has the block tri-diagonal (***tri-diag***) form:

$$\boldsymbol{V} = \boldsymbol{tri\text{-}diag}\,\{-\boldsymbol{v}_j, \boldsymbol{V}_j, -\boldsymbol{v}_{j+1}\} \tag{2.21}$$

$$\overset{\text{(defn)}}{=} \begin{bmatrix} \boldsymbol{V}_1 & -\boldsymbol{v}_2 & & & \boldsymbol{0} \\ -\boldsymbol{v}_2 & \boldsymbol{V}_2 & -\boldsymbol{v}_3 & & \\ \cdot & \cdot & \cdot & & \\ & \cdot & \cdot & \cdot & \\ & -\boldsymbol{v}_j & \boldsymbol{V}_j & -\boldsymbol{v}_{j+1} & \\ & & \cdot & \cdot & \cdot \\ & & & \cdot & \cdot \\ \boldsymbol{0} & & & -\boldsymbol{v}_m & \boldsymbol{V}_m \end{bmatrix}$$

where $\boldsymbol{v}_j$, $\boldsymbol{V}_j$ $(j = 1(1)m)$ are the $n \times n$ diagonal matrices defined by

$$\left.\begin{aligned} \boldsymbol{v}_j &= \text{diag}\,\{v_{1j}, \ldots, v_{nj}\} \\ \boldsymbol{V}_j &= \text{diag}\,\{V_{1j}, \ldots, V_{nj}\} \end{aligned}\right\}, j = 1(1)m \tag{2.22}$$

and where, in this definition and for later convenience, we re-define v_{i1} and $v_{i,m+1}$ by

$$v_{i1} = v_{i,m+1} = 0, \quad i = 1(1)n \tag{2.23}$$

so that

$$\boldsymbol{v}_1 = \boldsymbol{v}_{m+1} = \boldsymbol{0}. \tag{2.24}$$

We note that, in the above notation, the subscripts j and i do *not* refer to matrix elements in the usual manner but to row and column positions, respectively, of the mesh $R(h, k)$.

Finally, we let (the meaning of the superscript, t, will become apparent in the following section)

$$\boldsymbol{u}_{jH}^{(t)} = (u_{1j}^{(t)}, u_{2j}^{(t)}, \ldots, u_{nj}^{(t)})^* \tag{2.25}$$

† The notation $i = 0(1)n + 1$ is equivalent to $i = 0, 1, 2, \ldots, n + 1$.

so that

$$\boldsymbol{u}_H^{(t)} = (\boldsymbol{u}_{1H}^{(t)*}, \boldsymbol{u}_{2H}^{(t)*}, \ldots, \boldsymbol{u}_{mH}^{(t)*})^*, \tag{2.26}$$

and define $$\boldsymbol{f}_j = (f_{1j}, f_{2j}, \ldots, f_{nj})^* \tag{2.27}$$

so that $$\boldsymbol{f}_H = (\boldsymbol{f}_1^*, \boldsymbol{f}_2^*, \ldots, \boldsymbol{f}_m^*)^*. \tag{2.28}$$

3. The Peaceman–Rachford (P–R) method

The particular alternating direction implicit (ADI) method we shall consider is the original one due to Peaceman and Rachford (1955) where, given an arbitrary starting vector $\boldsymbol{u}_H^{(0)}$, a sequence of vectors $\{\boldsymbol{u}_H^{(t)}\}$ is defined iteratively from†

$$\left.\begin{aligned}(\boldsymbol{V} + r_t\boldsymbol{I})\boldsymbol{u}_H^{(t+\frac{1}{2})} &= \boldsymbol{f}_H - (\boldsymbol{H} - r_t\boldsymbol{I})\boldsymbol{u}_H^{(t)} \quad (3.1)\\ (\boldsymbol{H} + r_t\boldsymbol{I})\boldsymbol{u}_H^{(t+1)} &= \boldsymbol{f}_H - (\boldsymbol{V} - r_t\boldsymbol{I})\boldsymbol{u}_H^{(t+\frac{1}{2})} \quad (3.2)\end{aligned}\right\} t = 0, 1, 2, \ldots.$$

In the above, $\boldsymbol{u}_H^{(t+\frac{1}{2})}$ is an intermediate vector, and $\{r_t\}$ $(t = 0, 1, 2, \ldots)$ is a sequence of parameters chosen to accelerate the convergence of $\boldsymbol{u}_H^{(t)}$ towards $\boldsymbol{u}_H$; in practice, a finite sequence of T parameters is used cyclically, i.e.,

$$r_{t+T} = r_t, \quad t = 0, 1, 2, \ldots. \tag{3.3}$$

We shall not, in this paper, consider how the $\{r_t\}$ $(t = 0(1)T - 1)$ are chosen, nor, in fact, conditions under which the method converges to the solution of (2.15) or how fast it converges. For a full discussion of these and related problems see, for example, any of the references at the end of this paper.

We draw the reader's attention to a fundamental requirement of ADI methods, which is exhibited in (3.1) and (3.2), namely, the need to solve sets of equations at each half iteration in order to obtain subsequent iterates. The essential feature of these sets of equations is, however, that they only involve matrices containing three non-zero diagonals and hence (see below) such systems are simple to solve compared with the original system (2.15). Problems which are more complex than that defined in Section 2 (e.g. partial differential equations of order higher than the second) may lead to more than three non-zero diagonals. Nevertheless, the resulting systems still maintain band form and are simple to solve relative to the original system. We would point out once again that one of the objects of this paper is to illustrate how all the arithmetic operations implied by (3.1) and (3.2), including the solution of the above sets of equations, can be carried out while maintaining the row-ordering of the vectors $\{\boldsymbol{u}_H^{(t)}\}$. As we have already mentioned in the introduction, this point has important relevance when serial-access equipment is used for auxiliary storage—this will be considered in more detail in Section 5.

† Here we have defined the method for a particular ordering of the vector of approximate solutions $\{u_{ij}\}$, namely the row-ordering. However, it may clearly be defined independently of any ordering.

4. The numerical procedure

We begin this section by obtaining the Choleski or square-root decompositions of the matrices $\boldsymbol{V} + r_t\boldsymbol{I}$ and $\boldsymbol{H} + r_t\boldsymbol{I}$. Considering first $\boldsymbol{V} + r_t\boldsymbol{I}$, we let

$$\boldsymbol{L}_V(t) = \textbf{tri-diag}\,\{\boldsymbol{Q}_j^V(t), \boldsymbol{P}_j^V(t), \boldsymbol{0}\} \tag{4.1}$$

$$\overset{\text{(defn)}}{=} \begin{bmatrix} \boldsymbol{P}_1^V(t) & \boldsymbol{0} & & & & \boldsymbol{0}\\ \boldsymbol{Q}_2^V(t) & \boldsymbol{P}_2^V(t) & \boldsymbol{0} & & & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & \boldsymbol{Q}_j^V(t) & \boldsymbol{P}_j^V(t) & \boldsymbol{0} & \\ & & & \cdot & \cdot & \cdot \\ & & & & \cdot & \cdot \\ \boldsymbol{0} & & & & \boldsymbol{Q}_m^V(t) & \boldsymbol{P}_m^V(t) \end{bmatrix} \tag{4.2}$$

where $\boldsymbol{P}_j^V(t)$ and $\boldsymbol{Q}_j^V(t)\,(j = 1(1)m)$ are the $n \times n$ diagonal matrices defined by

$$\boldsymbol{P}_j^V(t) = \text{diag}\,\{1/p_{1j}^V(t), 1/p_{2j}^V(t), \ldots, 1/p_{nj}^V(t)\}, \quad j = 1(1)m \tag{4.3}$$

$$\boldsymbol{Q}_j^V(t) = \text{diag}\,\{q_{1j}^V(t), q_{2j}^V(t), \ldots, q_{nj}^V(t)\}, \quad j = 1(1)m \tag{4.4}$$

and the $\{p_{ij}^V(t)\}$, $\{q_{ij}^V(t)\}$ are defined by the recurrence relations

$p_{i0}^V(t) = 0, \quad i = 1(1)n$ (for notational convenience)

$$\left.\begin{aligned} q_{ij}^V(t) &= -v_{ij}p_{i,j-1}^V(t)\\ p_{ij}^V(t) &= \{V_{ij} + r_t - [q_{ij}^V(t)]^2\}^{-\frac{1}{2}}\end{aligned}\right\}, \; i = 1(1)n, \; j = 1(1)m\dagger. \tag{4.5}$$

From the positive-definiteness of $\boldsymbol{V} + r_t\boldsymbol{I}$, it can readily be shown that all the quantities involved in (4.3), (4.4) and (4.5) actually exist and, in fact, $p_{ij}^V(t) > 0$ $(i = 1(1)n, j = 1(1)m)$. For later arithmetic convenience, however, we prefer to define these quantities in this "inverse" manner.

The reader may verify that

$$\boldsymbol{L}_V(t)\boldsymbol{L}_V^*(t) = \boldsymbol{V} + r_t\boldsymbol{I} \tag{4.6}$$

so that $\boldsymbol{L}_V(t)$ is the lower triangular matrix appearing in the Choleski or square-root decomposition of $\boldsymbol{V} + r_t\boldsymbol{I}$.

Similarly, let

$$\boldsymbol{L}_H(t) = \textbf{diag}\,\{\boldsymbol{L}_{1H}(t), \ldots, \boldsymbol{L}_{mH}(t)\} \tag{4.7}$$

where

$$\boldsymbol{L}_{jH}(t) = \text{tri-diag}\,\{q_{ij}^H(t), 1/p_{ij}^H(t), 0\}$$

$$\overset{\text{(defn)}}{=} \begin{bmatrix} 1/p_{1j}^H(t) & 0 & & 0\\ q_{2j}^H(t) & 1/p_{2j}^H(t) & & \\ & \cdot & \cdot & \\ & & \cdot & \cdot \\ 0 & & q_{nj}^H(t) & 1/p_{nj}^H(t) \end{bmatrix}, \; j = 1(1)m \tag{4.8}$$

† Here and henceforth, when one running index expression follows another we imply that *all* values of the first expression are taken for *each* value of the second.

and the $\{p_{ij}^H(t)\}$, $\{q_{ij}^H(t)\}$ satisfy the recurrence relationships

$$\left.\begin{aligned} p_{0j}^H(t) &= 0 \\ \left.\begin{aligned} q_{ij}^H(t) &= -h_{ij}p_{i-1,j}^H(t) \\ p_{ij}^H(t) &= \{H_{ij} + r_t - [q_{ij}^H(t)]^2\}^{-\frac{1}{2}} \end{aligned}\right\}, &\ i = 1(1)n \end{aligned}\right\}, j = 1(1)m. \tag{4.9}$$

Again, $L_H(t)$ is the lower triangular matrix satisfying

$$L_H(t)L_H^*(t) = H + r_tI \tag{4.10}$$

representing the Choleski decomposition of $H + r_tI$. As before, all the quantities involved in (4.7), (4.8) and (4.9) do exist.

We now define vector sequences $\{w_H^{(t)}\}$ and $\{z_H^{(t)}\}$ by

$$\left.\begin{aligned} w_H^{(t+1)} &= L_V^*(t)u_H^{(t+\frac{1}{2})} \\ z_H^{(t+1)} &= L_H^*(t)u_H^{(t+1)} \end{aligned}\right\}, \quad t = 0, 1, 2, \ldots. \tag{4.11}$$

We may then re-write (3.1) and (3.2) as

$$\left.\begin{aligned} w_H^{(t+1)} &= L_V^{-1}(t)\{f_H - (H - r_tI) \\ &\qquad [L_H^*(t-1)]^{-1}z_H^{(t)}\} \\ z_H^{(t+1)} &= L_H^{-1}(t)\{f_H - (V - r_tI) \\ &\qquad [L_V^*(t)]^{-1}w_H^{(t+1)}\} \end{aligned}\right\}, \quad t = 0, 1, 2, \ldots \begin{matrix}(4.12)\\ \\ (4.13)\end{matrix}$$

where

$$z_H^{(0)} = L_H^*(-1)u_H^{(0)} = L_H^*(T-1)u_H^{(0)}, \tag{4.14}$$

so that (4.12) has an obvious meaning for $t = 0$.

The inverse notation used in the defining relations (4.12) and (4.13) in fact means that, during actual computation, the corresponding sets of linear equations are to be solved by a simple back or forward substitution.

Let us examine the sequence of operations involved in executing (4.12) and (4.13).

$$\boxed{\text{I. } u_H^{(t)} = [L_H^*(t-1)]^{-1}z_H^{(t)}:}$$

From (4.7), this is equivalent to

$$u_{jH}^{(t)} = [L_{jH}^*(t-1)]^{-1}z_{jH}^{(t)}, \quad j = 1(1)m \tag{4.15}$$

where $z_H^{(t)} = (z_{1H}^{(t)}{}^*, z_{2H}^{(t)}{}^*, \ldots, z_{mH}^{(t)}{}^*)^*$. From (4.8), this in turn is equivalent to the *back* substitutions

$$\left.\begin{aligned} u_{nj}^{(t)} &= p_{nj}^H(t-1)z_{nj}^{(t)} \\ u_{ij}^{(t)} &= p_{ij}^H(t-1)z[_{ij}^{(t)} - q_{i+1,j}^H(t-1)u_{i+1,j}^{(t)}], \\ &\qquad\qquad i = n - 1(-1)1 \end{aligned}\right\}, \quad j = 1(1)m \tag{4.16}$$

where the $\{p_{ij}^H(t-1)\}$, $\{q_{ij}^H(t-1)\}$ are defined in (4.9).

$$\boxed{\text{II. } \tilde{u}_H^{(t)} \text{ (say)} = f_H - (H - r_tI)u_H^{(t)}:}$$

Rewriting this in component form,

$$\tilde{u}_{ij}^{(t)} = f_{ij} + \{h_{ij}u_{i-1,j}^{(t)} + (r_t - H_{ij})u_{ij}^{(t)} + h_{i+1,j}u_{i+1,j}^{(t)}\}, \\ i = 1(1)n, \quad j = 1(1)m \tag{4.17}$$

where appropriate terms are suppressed for $i = 1$ and $i = n$.

$$\boxed{\text{III. } w_H^{(t+1)} = [L_V(t)]^{-1}\tilde{u}_H^{(t)}:}$$

From (4.2), this is effected by the block *forward* substitution:

$$\begin{aligned} w_{1H}^{(t+1)} &= [P_1^V(t)]^{-1}\tilde{u}_{1H}^{(t)} \\ w_{jH}^{(t+1)} &= [P_j^V(t)]^{-1}\{\tilde{u}_{jH}^{(t)} - Q_j^V(t)w_{j-1,H}^{(t+1)}\}, \quad j = 2(1)m, \end{aligned} \tag{4.18}$$

or, on account of (4.3) and (4.4):

$$\begin{aligned} w_{i1}^{(t+1)} &= p_{i1}^V(t)\tilde{u}_{i1}^{(t)}, \quad i = 1(1)n \\ w_{ij}^{(t+1)} &= p_{ij}^V(t)[\tilde{u}_{ij}^{(t)} - q_{ij}^V(t)w_{i,j-1}^{(t+1)}], \quad i = 1(1)n,\ j = 2(1)m. \end{aligned} \tag{4.19}$$

$$\boxed{\text{IV. } u_H^{(t+\frac{1}{2})} = [L_V^*(t)]^{-1}w_H^{(t+1)}:}$$

Similarly to III, only using a *back* instead of a forward substitution, $u_H^{(t+\frac{1}{2})}$ is obtained from the recurrence relations

$$\begin{aligned} u_{im}^{(t+\frac{1}{2})} &= p_{im}^V(t)w_{im}^{(t+\frac{1}{2})}, \quad i = 1(1)n \\ u_{ij}^{(t+\frac{1}{2})} &= p_{ij}^V(t)[w_{ij}^{(t+1)} - q_{i,j+1}^V(t)u_{i,j+1}^{(t+\frac{1}{2})}], \\ &\qquad i = 1(1)n, \quad j = m - 1(-1)1. \end{aligned} \tag{4.20}$$

$$\boxed{\text{V. } \tilde{u}_H^{(t+\frac{1}{2})} = f_H - (V - r_tI)u_H^{(t+\frac{1}{2})}:}$$

That is, using a convenient ordering of the subscript j,

$$\begin{aligned} \tilde{u}_{ij}^{(t+\frac{1}{2})} = f_{ij} &+ [v_{ij}u_{i,j-1}^{(t+\frac{1}{2})} + (r_t - V_{ij})u_{ij}^{(t+\frac{1}{2})} \\ &+ v_{i,j+1}u_{i,j+1}^{(t+\frac{1}{2})}], \quad i = 1(1)n, \quad j = m(-1)1 \end{aligned} \tag{4.21}$$

where appropriate terms are suppressed for $j = m$ and $j = 1$.

$$\boxed{\text{VI. } z_H^{(t+1)} = [L_H(t)]^{-1}\tilde{u}_H^{(t+\frac{1}{2})}:}$$

Using (4.7) and (4.8), this becomes (again ordering j conveniently)

$$\begin{aligned} z_{1j}^{(t+1)} &= p_{ij}^H(t)\tilde{u}_{1j}^{(t+\frac{1}{2})} \\ z_{ij}^{(t+1)} &= p_{ij}^H(t)[\tilde{u}_{ij}^{(t+\frac{1}{2})} - q_{ij}^H(t)z_{i-1,j}^{(t+1)}], \\ &\qquad i = 2(1)n, \quad j = m(-1)1. \end{aligned} \tag{4.22}$$

The reader may convince himself that the operations I–VI may be carried out in sequential order by proceeding through the mesh alternately upwards (i.e., for $j = 1(1)m$) and downwards (i.e., for $j = m(-1)1$), processing each *row* in turn, so that serial-access auxiliary storage equipment may be used without impediment. This is illustrated schematically in the flow chart in **Fig. 1.**

Intuitively, however, it would seem that this scheme would suffer from being input/output bound† since essentially the same data is being transferred repeatedly to and from (say) magnetic tapes for each of the

† We are assuming the use of a computer in which input/output can proceed in parallel (i.e. concurrently) with arithmetic computation. At a given time, such a computer is said to be *input/output bound* (or conversely, *computation bound*) if arithmetic processing is held up on account of data transfers to and from peripheral equipment (and conversely).

operations I–VI, while comparatively little arithmetic is being performed. This, of course, will depend upon the detailed features of the machine, in particular the number of tape-units, the ratio of read/write speeds to arithmetic speeds, and whether or not the tapes can be read backwards as well as forwards (reversible tapes). We shall consider this in more detail in Section 5. We now show, however, that we can merge I, II, and III into one combined operation, and similarly merge IV, V, and VI into one combined operation such that only one upward and one downward sweep of the mesh is necessay to complete one iteration, and the amount of data transfers is consequently reduced.

In order to merge I, II, and III, we note that the operation in II corresponding to the jth row can be performed immediately the operation in I corresponding to the jth row has been executed. Similarly, the operation in III corresponding to the jth row can be performed immediately the operation in II corresponding to the jth row has been executed, provided that we proceed *upwards* through the mesh in the order $j = 1(1)m$. Thus the operations of I, II, and III for a given fixed j can be combined into one operation, $A_j^{(t)}$ (see **Fig. 2**), and hence I, II, and III are equivalent to the execution of $A_j^{(t)}$ for $j = 1(1)m$.

Similar consideration of IV, V, and VI shows that they are equivalent to performing the operation $B_j^{(t)}$ proceeding *downwards* through the mesh in the order $j = m(-1)0$, where $B_j^{(t)}$ is described in **Fig. 3.**

We repeat that the method now consists of executing $A_j^{(t)}$ for $j = 1(1)m$, that is, sweeping upwards through the mesh row by row, and then executing $B_j^{(t)}$ for $j = m(-1)0$, that is, sweeping back downwards through the mesh. We also note that $A_j^{(t)}$ essentially transforms $z_{jH}^{(t)}$ into $w_{jH}^{(t+1)}$, and that $B_j^{(t)}$ transforms $w_{jH}^{(t+1)}$ into $z_{j+1,H}^{(t+1)}$; the vectors $u_{jH}^{(t)}$, $u_{jH}^{(t+\frac{1}{2})}$, $\tilde{u}_{jH}^{(t)}$, $\tilde{u}_{j+1,H}^{(t+\frac{1}{2})}$ are only intermediate vectors.

5. Programming considerations—data handling

In the previous section, we showed how the operations required to perform one iteration of the ADI algorithm can be conveniently reduced to an upward sweep followed by a downward sweep through the mesh operating upon rows successively; we intimated that the algorithm in this form is intuitively consistent with the requirements of the transmission of the data to and from serial-access auxiliary storage in the sense that all the data needed for $A_1^{(t)}$ will be accessible first, followed by all the data needed for $A_2^{(t)}$ and so forth, and similarly for the $B_j^{(t)}$ on the downward sweep with decreasing order of the subscript. It might at first appear, from the fact that the operations $A_j^{(t)}$ are performed in increasing order of the subscript, while the operations $B_j^{(t)}$ are performed in decreasing order of the subscript, that this may be inconsistent with the requirements of, say, non-reversible magnetic tapes which can only be read in one direction, but this difficulty can usually be overcome by employing tape-splitting techniques.

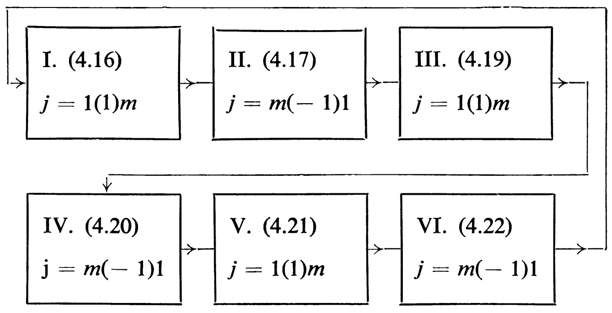

Fig. 1.—Flow diagram of ADI operations showing maintenance of row-ordering

$$u_{ij}^{(t)} = p_{ij}^H(t-1)[z_{ij}^{(t)} - q_{i+1,j}^H(t-1)u_{i+1,j}^{(t)}], \quad i = n(-1)1$$
$$\tilde{u}_{ij}^{(t)} = f_{ij} + [h_{ij}u_{i-1,j}^{(t)} + (r_t - H_{ij})u_{ij}^{(t)} + h_{i+1,j}u_{i+1,j}^{(t)}], i = n(-1)1$$
$$w_{ij}^{(t+1)} = p_{ij}^V(t)[\tilde{u}_{ij}^{(t)} - q_{ij}^V(t)w_{i,j-1}^{(t+1)}], \quad i = n(-1)1$$

where

$$q_{n+1,j}^H(t-1) = u_{n+1,j}^{(t)} = h_{n+1,j} = u_{0j}^{(t)} = h_{1,j} = 0, \quad j = 1(1)m$$
$$w_{i0}^{(t+1)} = 0, \quad i = 1(1)n$$

Fig. 2.—Operation $A_j^{(t)}$, combining the operations of I, II, and III at the *j*th stage

$$u_{ij}^{(t+\frac{1}{2})} = p_{ij}^V(t)[w_{ij}^{(t+1)} - q_{i,j+1}^V(t)u_{i,j+1}^{(t+\frac{1}{2})}], \quad i = 1(1)n\ (j \neq 0)$$
$$\tilde{u}_{i,j+1}^{(t+\frac{1}{2})} = f_{i,j+1} + [v_{i,j+1}u_{ij}^{(t+\frac{1}{2})} + (r_t - V_{i,j+1})u_{i,j+1}^{(t+\frac{1}{2})}$$
$$+ v_{i,j+2}u_{i,j+2}^{(t+\frac{1}{2})}], \ i = 1(n)n\ (j \neq m)$$
$$z_{i,j+1}^{(t+1)} = p_{i,j+1}^H(t)[\tilde{u}_{i,j+1}^{(t+\frac{1}{2})} - q_{i,j+1}^H(t)z_{i-1,j+1}^{(t)}], \ i = 1(1)n\ (j \neq m)$$

where

$$q_{i,m+1}^V(t) = u_{i,m+1}^{(t+\frac{1}{2})} = v_{i,m+1} = 0, \quad i = 1(1)n$$
$$z_{0,j+1}^{(t+1)} = 0, \qquad j = 0(1)m - 1.$$

Fig. 3.—Operation $B_j^{(t)}$, combining the operations of IV, V, and VI at the *j*th stage

[We note parenthetically that data stored on magnetic tapes usually consists of blocks of numerical information (records) separated by gaps. The write instruction usually specifies an area of consecutive main memory locations, the contents of which are serially transferred to the tape thus forming a block. The read instruction initiates the transfer of the contents of a block to consecutive locations in main memory. Some tape units (reversible) can read the tape whether it is moving forwards or backwards past the reading-head; such backward reading may result in the data being filed in main memory in reverse order to that resulting from forward reading, leading to additional, although minor, indexing problems. A useful feature of reversible tape units, particularly in connection with iterative procedures, is that it may avoid the use of the "tape-splitting" technique, in which, say, the first half of a given set of blocks, which are required sequentially, is stored on one tape, and the second half on a second tape, one being rewound while the other is being referenced and vice-versa (see Section 5(a))].

Although in this paper we are emphasizing serial-access auxiliary storage, such as magnetic tape units, much of what is said is equally applicable to slow (compared to main memory) random-access storage (discs, drums, and slow-access core storage), where the transfer of blocks of sequentially stored data is comparable, in principle and in speed, to the use of serial-access equipment. We shall indeed henceforth assume that the auxiliary storage consists of magnetic tape units, since the referencing problems are then more acute than those associated with block transfers to and from random-access storage; later, the emphasis will be placed on reversible tape units.

In this section we examine some of the questions that arise in connection with implementing the above requirements. Then it is the programmer's aim to arrange the computation, working areas in main memory (for input, output and computation), the spacing of the input and output commands throughout the program, and the distribution of data between the available tape units such that the input of data for, say, $A_{j+1}^{(t)}$, the computation of $A_j^{(t)}$, and the output of the results of $A_{j-1}^{(t)}$ (and similarly for $B_{j-1}^{(t)}$, $B_j^{(t)}$, $B_{j+1}^{(t)}$) all take place simultaneously and flow continuously so that the machine is tape/computation bound (see footnote page 169) to the minimum degree. Naturally the details of this depend crucially upon individual computer-system characteristics, e.g., arithmetic speeds, rate of transfer of data to and from magnetic tapes, the amount of high-speed storage available, the number of tape units (and whether they are reversible), and the number of tape units that can be referenced simultaneously. These characteristics vary so considerably from machine to machine that no general discussion is really feasible, each machine requiring its own particular program. However, there are a few general principles that can usefully be considered, and a discussion of these principles will be the object of the remainder of this paper.

We start by recalling that for the operation $A_j^{(t)}(B_j^{(t)})$ we transform the vector $z_j^{(t)}(w_j^{(t+1)})$ into $w_j^{(t+1)}$ $(z_{j+1}^{(t+1)})$ and require the data as set out in **Table 1** for each operation. (Note the slight variant of our previous notation.)

There is a considerable degree of flexibility in the way in which the vectors of the "decomposition elements", $\boldsymbol{p}_j^H(t)$, $\boldsymbol{q}_j^H(t)$, $\boldsymbol{p}_j^V(t)$, $\boldsymbol{q}_j^V(t)$, can be manipulated, and this has important bearing upon the programmer's strategy in achieving the aims referred to above. We now discuss several alternatives in some detail, and will finally exemplify one of these alternatives in connection with a program presently being written for the English Electric—Leo KDF9 computer of the National Physical Laboratory.

(*a*) *Computation of all decomposition elements before iterating*

In equation (3.3) we referred to the cyclic use of a finite set of parameters $r_0, r_1, \ldots, r_{T-1}$, so that

$$\boldsymbol{p}_j^H(t+T) = \boldsymbol{p}_j^H(t), \quad t = 0, 1, 2, \ldots, \quad j = 1(1)m, \quad (5.1)$$

Table 1

Quantities required at typical stages of the upward and downward sweep

$A_j^{(t)}$	$B_j^{(t)}$
$\boldsymbol{z}_j^{(t)} = (z_{1j}^{(t)}, \ldots, z_{nj}^{(t)})^*$	$\boldsymbol{w}_j^{(t+1)}$
$\boldsymbol{p}_j^H(t-1) = (p_{0j}^H(t-1), \ldots, p_{nj}^H(t-1))^*$	$\boldsymbol{p}_j^V(t)$
$\boldsymbol{q}_j^H(t-1) = (q_{1j}^H(t-1), \ldots, q_{n+1,\,j}^H(t-1))^*$	$\boldsymbol{q}_{j+1}^V(t)$
$\boldsymbol{p}_j^V(t) = (p_{1j}^V(t), \ldots, p_{nj}^V(t))^*$	$\boldsymbol{p}_{j+1}^H(t)$
$\boldsymbol{q}_j^V(t) = (q_{1j}^V(t), \ldots, q_{nj}^V(t))^*$	$\boldsymbol{q}_{j+1}^H(t)$
$\hat{\boldsymbol{h}}_j = (h_{1j}, \ldots, h_{n+1,\,j})^*$	$\hat{\boldsymbol{v}}_{j+1}, \hat{\boldsymbol{v}}_{j+2}$
$\hat{\boldsymbol{H}}_j = (H_{1j}, \ldots, H_{nj})^*$	$\hat{\boldsymbol{V}}_{j+1}$
$\hat{\boldsymbol{v}}_j = (v_{1j}, \ldots, v_{nj})^*$	$\hat{\boldsymbol{h}}_{j+1}$
$\hat{\boldsymbol{V}}_j = (V_{1j}, \ldots, V_{nj})^*$	$\hat{\boldsymbol{H}}_{j+1}$
$\boldsymbol{f}_j = (f_{1j}, \ldots, f_{nj})^*$	$\boldsymbol{f}_{j+1}$
The above being used to compute	
$\boldsymbol{w}_j^{(t+1)} = (w_{1j}^{(t+1)}, \ldots, w_{nj}^{(t+1)})^*$	$\boldsymbol{z}_{j+1}^{(t+1)}$

and similarly for the remainder of the decomposition elements. Thus an obvious technique would be to compute all the decomposition elements (of which, in all, there are $(4n+2)mT$) at the beginning of the program, storing them on magnetic tape and using them as required during the iteration. It is clear that, for handling the decomposition elements, at least two tape units, and tape-splitting techniques will be required to achieve continuous operation. In the event that reversible tape units are available and the parameters are used cyclically in the order $r_0, r_1, \ldots, r_{T-1}, r_{T-2}, \ldots, r_1, r_0$, only one tape unit is required.

If only two tape units are in fact available for storage of these elements, then, since $4n+2$ elements must be read during the operation $A_j^{(t)}$ or $B_j^{(t)}$ (assuming of course, that the remaining quantities required (see Table 1) can be read in parallel), a crude estimate of the arithmetic which must be performed in the same time indicates that a high reading rate (in comparison with arithmetic speeds) would be required if we are to avoid being tape-bound. Alternatively, more than two tape units would be needed so that shorter block-lengths can be achieved by sharing the data, and, if necessary, extending the use of tape-splitting, thereby reducing tape-bound time (again assuming that parallel reading of all tape units is possible).

Thus, at the expense of handling large quantities of data, this is the most efficient ADI scheme from the point of view of minimizing the number of arithmetic operations during iteration.

(b) *Totally redundant computation of decomposition elements during iterations*

Scheme (*a*) above required either a relatively high transfer rate or a large number of tape units. If neither of these requirements can be met, it may still be possible to avoid being tape-bound, and even to achieve a reduction in the total time needed to complete an iteration, although this, as we shall see, will entail redundant computation.

This redundant computation essentially consists of recomputing all of the decomposition elements as they are required and thereby taking advantage neither of the periodicity exemplified by (5.1), nor of the fact that the same quantities are used a half-iteration apart (an important modification of this scheme which takes advantage of the latter is discussed in Section 5(*c*)). However, although the amount of arithmetic is increased, the amount of data which must be transferred from tape is reduced (as will be seen below), and the object, using either this scheme or the modification discussed in Section 5(*c*), is to achieve a balance between the two and hence an overall saving in time.

For any j, we see from the recurrence relations (4.9) that the elements of $\mathbf{p}_j^{(H)}(t-1)$ and $\mathbf{q}_j^{(H)}(t-1)$ for $A_j^{(t)}$ and of $\mathbf{p}_{j+1}^H(t)$ and $\mathbf{q}_{j+1}^H(t)$ for $B_j^{(t)}$ can be readily obtained when required. With regard to the calculation of the elements of $\mathbf{p}_j^V(t)$ and $\mathbf{q}_j^V(t)$ for the execution of $A_j^{(t)}$, the recurrence relations (4.5) can be used as they stand provided that $\mathbf{p}_{j-1}^V(t)$ is available; this in fact is the case since $\mathbf{p}_{j-1}^V(t)$ has just been computed and used in the execution of $A_{j-1}^{(t)}$. For $B_j^{(t)}$, we require $\mathbf{p}_j^V(t)$ and $\mathbf{q}_{j+1}^V(t)$, and we must re-arrange the recurrence relations (4.5) into a slightly different form, noting that $\mathbf{p}_{j+1}^V(t)$ has been similarly computed during the execution of $B_{j+1}^{(t)}$:

$$\left.\begin{aligned} q_{i,j+1}^V(t) &= \{V_{i,j+1} + r_t \\ &\quad - [p_{i,j+1}^V(t)]^{-2}\}^{\frac{1}{2}} \\ p_{i,j}^V(t) & \\ &= -\,q_{i,j+1}^V(t)/v_{i,j+1} \end{aligned}\right\}, i=1(1)n, \quad j=m-1(-1)1, \quad (5.2)$$

the recursion being initiated with the available $\mathbf{p}_m^V(t)$ which has been computed during $A_m^{(t)}$ and also used during $B_m^{(t)}$ (N.B. $\mathbf{q}_{m+1}^V(t)=0$, see Fig. 3).

This technique is the extreme opposite of that discussed in Section 5(*a*) and could be used where either very few tape units can be referenced in parallel or where the reading speed is extremely slow in relation to arithmetic speeds. In the next subsection we discuss examples of how a compromise can be achieved between the two extremes so far described.

(c) *Partially redundant computation of decomposition elements during iterations*

One such compromise, clearly, is to output onto tape the $\mathbf{p}_j^V(t)$ and $\mathbf{q}_j^V(t)$ after they have been used in $A_j^{(t)}$ and then read them in again when they are required for $B_j^{(t)}$ and $B_{j-1}^{(t)}$, respectively; similarly $\mathbf{p}_{j+1}^H(t)$ and $\mathbf{q}_{j+1}^H(t)$ are output after $B_j^{(t)}$ and read in again for use in $A_j^{(t+1)}$. Thus, it is only necessary to recompute half of the decomposition elements at each stage.

There are many possible configurations along these lines and it is likely that an efficient, smoothly flowing

$$u_{ij}^{(t)} = \overrightarrow{p_{ij}^H}(t-1)[\overrightarrow{z_{ij}^{(t)}} + \overrightarrow{h_{i+1,j}}\overrightarrow{p_{ij}^H}(t-1)u_{i+1,j}^{(t)}], \quad i=n(-1)1$$

$$\tilde{u}_{ij}^{(t)} = \overrightarrow{f_{ij}} + [h_{ij}u_{i-1,j}^{(t)} + (r_t - \overrightarrow{H_{ij}})u_{ij}^{(t)} + h_{i+1,j}u_{i+1,j}^{(t)}], \quad i=n(-1)1$$

$$q_{ij}^V(t) = -\,\overrightarrow{v_{ij}}p_{i,j-1}^V(t), \quad i=n(-1)1$$

$$p_{ij}^V(t) = \{\overrightarrow{V_{ij}} + r_t - [q_{ij}^V(t)]^2\}^{-\frac{1}{2}}, \quad i=n(-1)1$$

$$\overleftarrow{w_{ij}^{(t+1)}} = \overleftarrow{p_{ij}^V}(t)[\tilde{u}_{ij}^{(t)} - q_{ij}^V(t)w_{i,j-1}^{(t+1)}], \quad i=n(-1)1$$

where

$$u_{n+1,j}^{(t)} = h_{n+1,j} = u_{0j}^{(t)} = h_{1j} = 0, \quad j=1(1)m$$

$$p_{i0}^V(t) = v_{i1} = w_{i0}^{(t+1)} = 0, \quad i=1(1)n.$$

Fig. 4.—$\widetilde{A}_j^{(t)}$, $j=1(1)m$

Modified form of operation $A_j^{(t)}$ showing partially redundant computation of decomposition elements. The arrows above quantities relate to input/output requirements (see Section 5(c))

$$u_{ij}^{(t+\frac{1}{2})} = \overrightarrow{p_{ij}^V}(t)[\overrightarrow{w_{ij}^{(t+1)}} + \overrightarrow{v_{i,j+1}}\overrightarrow{p_{ij}^V}(t)u_{i,j+1}^{(t+\frac{1}{2})}],$$
$$i=1(1)n \text{ (also for } j=m)$$

$$\tilde{u}_{i,j+1}^{(t+\frac{1}{2})} = \overrightarrow{f_{i,j+1}} + [v_{i,j+1}u_{i,j}^{(t+\frac{1}{2})} + (r_t - \overrightarrow{V_{i,j+1}})u_{i,j+1}^{(t+\frac{1}{2})} + v_{i,j+2}u_{i,j+2}^{(t+\frac{1}{2})}]$$

$$p_{0,j+1}^H(t) = 0$$

$$\left.\begin{aligned} q_{i,j+1}^H(t) &= -\,\overrightarrow{h_{i,j+1}}p_{i-1,j+1}^H(t) \\ p_{i,j+1}^H(t) &= \{\overrightarrow{H_{i,j+1}} + r_t - [q_{i,j+1}^H(t)]^2\}^{-\frac{1}{2}}, \end{aligned}\right\} i=1(1)n$$

$$z_{i,j+1}^{(t+1)} = \overleftarrow{p_{i,j+1}^H}(t)[\tilde{u}_{i,j+1}^{(t+\frac{1}{2})} - q_{i,j+1}^H(t)\overleftarrow{z_{i-1,j+1}^{(t+1)}}], \quad i=1(1)n$$

where

$$v_{i,m+1} = u_{i,m+1}^{(t+\frac{1}{2})} = 0, \quad i=1(1)n$$
$$z_{0,j+1}^{(t+1)} = 0, \quad j=m-1(-1)0.$$

Fig. 5.—$\widetilde{B}_j^{(t)}$, $j=m-1(-1)0$

Modified form of operation $B_j^{(t)}$ showing partially redundant computation of decomposition elements. The arrows above quantities relate to input/output requirements (see Section 5(c))

program can be achieved for a large variety of machine installations. To exemplify this, we will outline certain aspects of a preliminary version of a program currently being written for the KDF9 of the National Physical Laboratory. This machine is presently equipped with four reversible, independently buffered tape units (permitting parallel referencing) whose transfer-rate is relatively slow compared with arithmetic speeds; the scheme we describe is dovetailed to these properties.

In this scheme we output $\mathbf{p}_j^V(t)$ but not $\mathbf{q}_j^V(t)$ along with the intermediate vector $\mathbf{w}_j^{(t+1)}$ after completion of $A_j^{(t)}$. These quantities are input before $B_j^{(t)}$, ready for use during the latter, when we first compute $\mathbf{q}_{j+1}^V(t)$, using the second equation of (4.5), and later compute $\mathbf{p}_{j+1}^H(t)$ and $\mathbf{q}_{j+1}^H(t)$ using (4.9); having completed $B_j^{(t)}$, we output $\mathbf{p}_{j+1}^H(t)$ along with $\mathbf{z}_{j+1}^{(t+1)}$. Thus we can see that $\mathbf{p}_j^H(t)$ is available for $A_j^{(t+1)}$ and can be used to

Relative address ..	$W + 1$	...	$W + i - 1$	$W + i$	$W + i + 1$	...	$W + n$
Contents of W-area ..	$w^{(t+1)}_{1,j-1}$	...	$w^{(t+1)}_{i-1,j-1}$	$w^{(t+1)}_{i,j}$	$w^{(t+1)}_{i+1,j}$	...	$w^{(t+1)}_{n,j}$

Fig. 6.—Contents of the W-area of memory for index i during the execution of the last expression of $\tilde{A}^{(t)}_j$

recompute $q^H_j(t)$ by the second equation of (4.9). We also compute $p^V_j(t + 1)$ and $q^V_j(t + 1)$ during $A^{(t+1)}_j$, the former being output, thereby completing the cycle. We can thus detail the modified operations $\tilde{A}^{(t)}_j$ and $\tilde{B}^{(t)}_j$ as shown in **Figs. 4** and **5.**

The arrows above the various elements relate to input–output requirements in the following way: a right-pointing arrow ($\rightarrow$) means that this is the first time the quantity above which it appears is being referenced and indicates that the entire corresponding vector must have been read in before the expression is executed; a left-pointing arrow ($\leftarrow$) above a quantity means that after the expression in which the quantity appears has been executed, the entire corresponding vector may be written out (this does not mean that the quantity is not referenced again in succeeding operations; e.g., $w^{(t+1)}_j$, computed during $\tilde{A}^{(t)}_j$, is referenced again in $\tilde{A}^{(t)}_{j+1}$).

We point out that the double-subscripting in $\tilde{A}^{(t)}_j$ and $\tilde{B}^{(t)}_j$, while convenient for a clear statement of the algorithm, is, of course, meaningless in the context of a computer program, since the program references the same locations in memory which contain corresponding elements of similar vectors for successive row operations; for example, in **Fig. 6**, we show the contents of the W-area of memory, where W is the sequence of consecutive memory locations into which the vector $w^{(t+1)}_j$ is accumulated, for a given index i of the last expression of $\tilde{A}^{(t)}_j$. Similar considerations hold for the other data vectors. It may be necessary to create two or more working areas, which are to be used cyclically, for similar quantities, one being referenced for computation during a given row operation, the other being tied up with output of results of previous row operations or the input of data for succeeding row operations.

With this program and machine configuration, the time taken to execute the operation $\tilde{A}^{(t)}_j$ or $\tilde{B}^{(t)}_j$ is very nearly the same as that taken to read or write a block of $\frac{5}{2}n$ numbers from or onto tape; therefore, provided we distribute the data amongst the four tape units in the manner set out in **Figs. 7** and **8**, where only $\frac{5}{2}n$ numbers need be transferred to or from the same tape unit during one such operation, we will be very close to our desired aim of an optimum balance between computation and input/output.

We note that it is necessary to divide the vectors f_j into two subvectors

$$f^V_j = (f_{1j}, f_{2j}, \ldots, f_{[n/2],j})^* \qquad (5.3)$$

and

$$f^H_j = (f_{[n/2]+1,j}, \ldots, f_{n,j})^* \qquad (5.4)$$

where f^V_j is stored in the same block as $\hat{v}_j$ and $\hat{V}_j$ on a tape we call the V-matrix tape, and f^H_j is stored in the same block as $\hat{h}_j$ and $\hat{H}_j$ on a tape we call the H-matrix tape.

Fig. 7 schematically depicts the state of the tapes during the operation $\tilde{A}^{(t)}_j$, and in particular shows those quantities that have been read and used or computed and output during preceding operations, those that are in the process of being read or output, and those that have yet to be read. Similarly, in Fig. 8 the same features are exhibited for the operation $\tilde{B}^{(t)}_j$ on the return sweep.

From Figs. 7 and 8 we also see the value of reversible tapes; as indicated by the arrows, the solution tapes (those containing $p^H_j(t - 1)$, $z^{(t)}_j$, etc.) are always read backwards and written forwards, whilst the matrix-tapes, which are never written upon, are read in both directions. All necessity for tape-splitting and rewinding of tapes is thereby avoided. We emphasize once again that reverse reading may invert the order of elements in the main memory thereby resulting in slightly complex indexing problems.

6. Conclusions

We have seen how the arithmetic operations of the Peaceman–Rachford ADI method can be conveniently arranged to allow for the requirements of serial-access (or slow random-access) auxiliary storage, and how various schemes, possibly involving redundant computation, can probably be devised to ensure efficient, continuously flowing computer programs.

It should be mentioned, however, that these schemes give rise to certain problems in connection with testing for the convergence of the iterative procedure. With iterative processes, usually one accumulates some norm, $||d^{(t)}||$, say, of the displacement vector $d^{(t)} = u^{(t-1)} - u^{(t)}$, and terminates the process when $||d^{(t)}||$ is sufficiently small. In the above description of the algorithm, however, it may be inconvenient to accumulate $||d^{(t)}||$, since only $z^{(t)}$ and $w^{(t+1)}$, or $w^{(t+1)}$ and $z^{(t+1)}$, are really available simultaneously. If extra auxiliary storage is not available, some alternative controls may have to be devised; for example, these might be based upon the accumulation of norms of randomly-selected subvectors of $d^{(t)}$, and, when a sequence of these are sufficiently small, a special procedure may be entered to accumulate and test the norm of the entire current displacement vector $d^{(t)}$.

Note added in proof: It has recently come to our attention that the possibility of conveniently row-ordering ADI methods for serial-access auxiliary storage has been independently discovered by W. J. van de Lindt (1964):

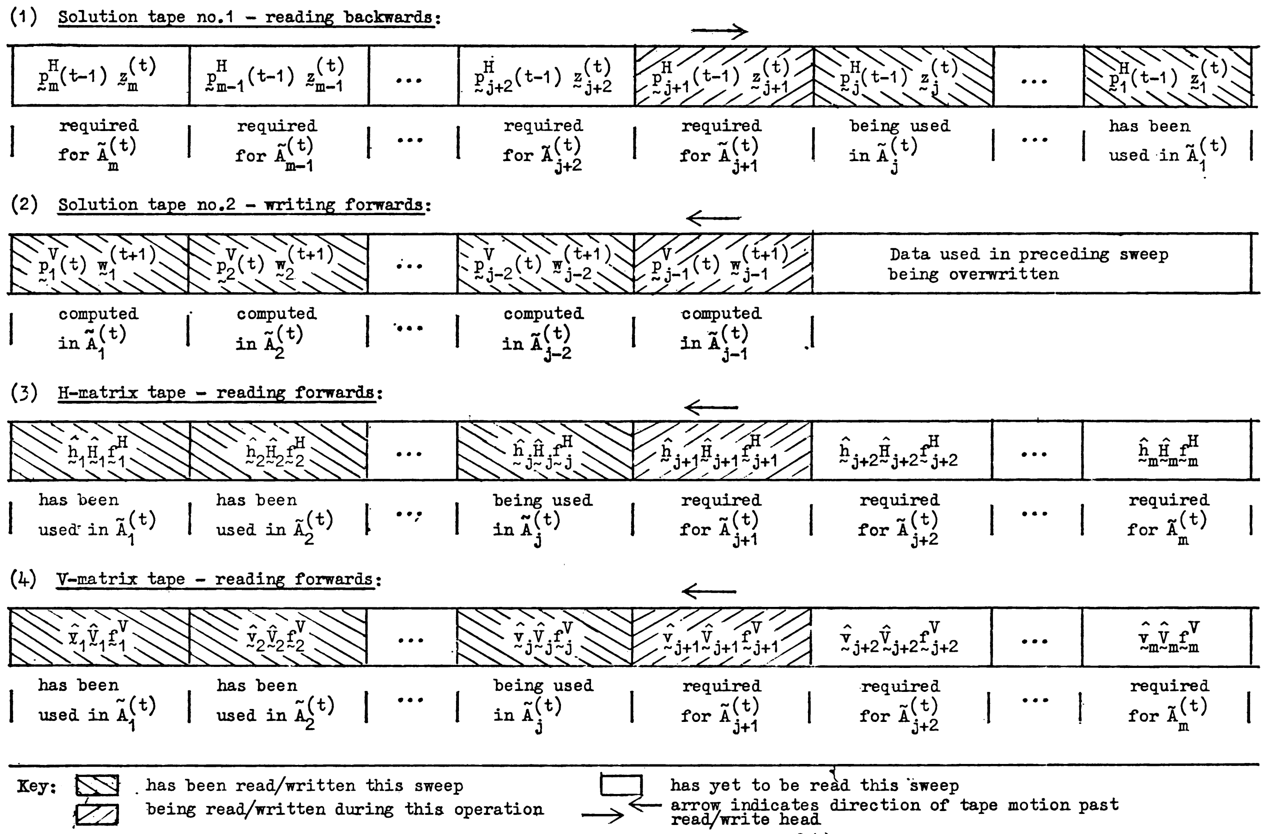
(1) Solution tape no.1 - reading backwards:
$p_m^H(t-1)\ z_m^{(t)}$ required for $\tilde{A}_m^{(t)}$
$p_{m-1}^H(t-1)\ z_{m-1}^{(t)}$ required for $\tilde{A}_{m-1}^{(t)}$
$p_{j+2}^H(t-1)\ z_{j+2}^{(t)}$ required for $\tilde{A}_{j+2}^{(t)}$
$p_{j+1}^H(t-1)\ z_{j+1}^{(t)}$ required for $\tilde{A}_{j+1}^{(t)}$
$p_j^H(t-1)\ z_j^{(t)}$ being used in $\tilde{A}_j^{(t)}$
$p_1^H(t-1)\ z_1^{(t)}$ has been used in $\tilde{A}_1^{(t)}$
(2) Solution tape no.2 - writing forwards:
$p_1^V(t)\ w_1^{(t+1)}$ computed in $\tilde{A}_1^{(t)}$
$p_2^V(t)\ w_2^{(t+1)}$ computed in $\tilde{A}_2^{(t)}$
$p_{j-2}^V(t)\ w_{j-2}^{(t+1)}$ computed in $\tilde{A}_{j-2}^{(t)}$
$p_{j-1}^V(t)\ w_{j-1}^{(t+1)}$ computed in $\tilde{A}_{j-1}^{(t)}$
Data used in preceding sweep being overwritten
(3) H-matrix tape - reading forwards:
$\hat{h}_1\hat{H}_1 f_1^H$ has been used in $\tilde{A}_1^{(t)}$
$\hat{h}_2\hat{H}_2 f_2^H$ has been used in $\tilde{A}_2^{(t)}$
$\hat{h}_j\hat{H}_j f_j^H$ being used in $\tilde{A}_j^{(t)}$
$\hat{h}_{j+1}\hat{H}_{j+1} f_{j+1}^H$ required for $\tilde{A}_{j+1}^{(t)}$
$\hat{h}_{j+2}\hat{H}_{j+2} f_{j+2}^H$ required for $\tilde{A}_{j+2}^{(t)}$
$\hat{h}_m\hat{H}_m f_m^H$ required for $\tilde{A}_m^{(t)}$
(4) V-matrix tape - reading forwards:
$\hat{v}_1\hat{V}_1 f_1^V$ has been used in $\tilde{A}_1^{(t)}$
$\hat{v}_2\hat{V}_2 f_2^V$ has been used in $\tilde{A}_2^{(t)}$
$\hat{v}_j\hat{V}_j f_j^V$ being used in $\tilde{A}_j^{(t)}$
$\hat{v}_{j+1}\hat{V}_{j+1} f_{j+1}^V$ required for $\tilde{A}_{j+1}^{(t)}$
$\hat{v}_{j+2}\hat{V}_{j+2} f_{j+2}^V$ required for $\tilde{A}_{j+2}^{(t)}$
$\hat{v}_m\hat{V}_m f_m^V$ required for $\tilde{A}_m^{(t)}$
Key: has been read/written this sweep
being read/written during this operation
has yet to be read this sweep
arrow indicates direction of tape motion past read/write head

Fig. 7.—Schematic representation of tapes during $\tilde{A}_j^{(t)}$

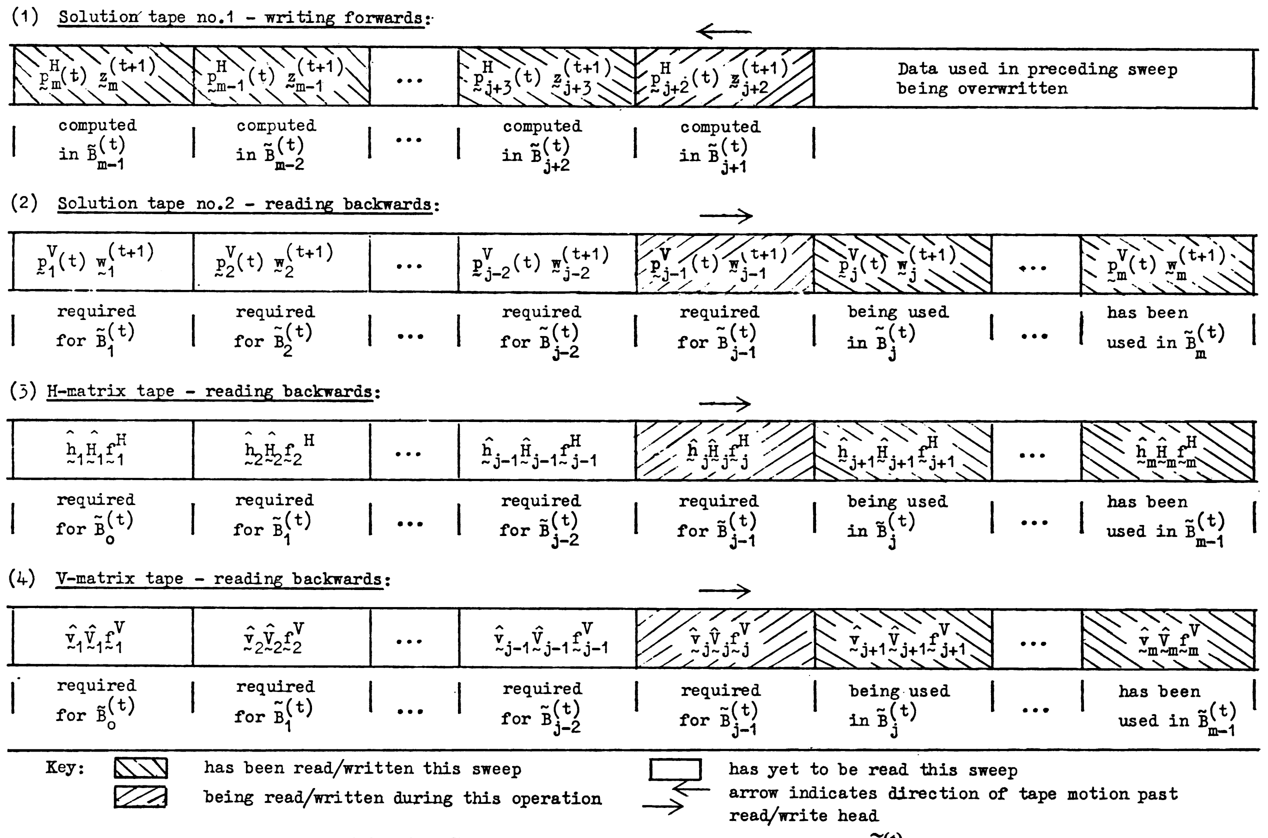
(1) Solution tape no.1 - writing forwards:
$p_m^H(t)\ z_m^{(t+1)}$ computed in $\tilde{B}_{m-1}^{(t)}$
$p_{m-1}^H(t)\ z_{m-1}^{(t+1)}$ computed in $\tilde{B}_{m-2}^{(t)}$
$p_{j+3}^H(t)\ z_{j+3}^{(t+1)}$ computed in $\tilde{B}_{j+2}^{(t)}$
$p_{j+2}^H(t)\ z_{j+2}^{(t+1)}$ computed in $\tilde{B}_{j+1}^{(t)}$
Data used in preceding sweep being overwritten
(2) Solution tape no.2 - reading backwards:
$p_1^V(t)\ w_1^{(t+1)}$ required for $\tilde{B}_1^{(t)}$
$p_2^V(t)\ w_2^{(t+1)}$ required for $\tilde{B}_2^{(t)}$
$p_{j-2}^V(t)\ w_{j-2}^{(t+1)}$ required for $\tilde{B}_{j-2}^{(t)}$
$p_{j-1}^V(t)\ w_{j-1}^{(t+1)}$ required for $\tilde{B}_{j-1}^{(t)}$
$p_j^V(t)\ w_j^{(t+1)}$ being used in $\tilde{B}_j^{(t)}$
$p_m^V(t)\ w_m^{(t+1)}$ has been used in $\tilde{B}_m^{(t)}$
(3) H-matrix tape - reading backwards:
$\hat{h}_1\hat{H}_1 f_1^H$ required for $\tilde{B}_0^{(t)}$
$\hat{h}_2\hat{H}_2 f_2^H$ required for $\tilde{B}_1^{(t)}$
$\hat{h}_{j-1}\hat{H}_{j-1} f_{j-1}^H$ required for $\tilde{B}_{j-2}^{(t)}$
$\hat{h}_j\hat{H}_j f_j^H$ required for $\tilde{B}_{j-1}^{(t)}$
$\hat{h}_{j+1}\hat{H}_{j+1} f_{j+1}^H$ being used in $\tilde{B}_j^{(t)}$
$\hat{h}_m\hat{H}_m f_m^H$ has been used in $\tilde{B}_{m-1}^{(t)}$
(4) V-matrix tape - reading backwards:
$\hat{v}_1\hat{V}_1 f_1^V$ required for $\tilde{B}_0^{(t)}$
$\hat{v}_2\hat{V}_2 f_2^V$ required for $\tilde{B}_1^{(t)}$
$\hat{v}_{j-1}\hat{V}_{j-1} f_{j-1}^V$ required for $\tilde{B}_{j-2}^{(t)}$
$\hat{v}_j\hat{V}_j f_j^V$ required for $\tilde{B}_{j-1}^{(t)}$
$\hat{v}_{j+1}\hat{V}_{j+1} f_{j+1}^V$ being used in $\tilde{B}_j^{(t)}$
$\hat{v}_m\hat{V}_m f_m^V$ has been used in $\tilde{B}_{m-1}^{(t)}$
Key: has been read/written this sweep
being read/written during this operation
has yet to be read this sweep
arrow indicates direction of tape motion past read/write head

Fig. 8.—Schematic representation of tapes during $\tilde{B}_j^{(t)}$

"Nuclear reactor calculations with the group diffusion equations on digital computers." Doctoral dissertation, Technische Hogeschool, Delft.

7. Acknowledgements

The authors would like to acknowledge with gratitude the helpful comments of E. L. Albasiny, C. W. Clenshaw and G. F. Miller.

The above work was carried out as part of the research programme of the National Physical Laboratory, and is published with the permission of the Director of the Laboratory.

References

Birkhoff, G., Varga, R. S., and Young, D. (1962). "Alternating direction implicit methods," *Advances in Computers*, Vol. 3, p. 189.

Peaceman, D. W., and Rachford, H. H., Jr. (1955). "The numerical solution of parabolic and elliptic differential equations," *J. Soc. Indust. Appl. Math.*, Vol. 3, p. 28.

Varga, R. S. (1962). *Matrix Iterative Analysis*, New Jersey: Prentice-Hall.

Wachspress, E. L. (1963). "Extended application of alternating direction implicit iteration model problem theory," *J. Soc. Indust. Appl. Math.*, Vol. 11, p. 994.

Correspondence

An impossible program

To the Editor,
The Computer Journal.

Dear Sir,

Strachey's letter* under this title gave rise to some discussion among my colleagues, some of it along the lines of ApSimon's letter,† and it recalled to me similar discussions in my school-days about the validity of *reductio ad absurdum* proofs in geometry. If ApSimon's objections were valid, they would apply to all proofs of this sort, and invalidate a fair part of the structure of mathematics from Euclid onward.

Much more interesting to me is a corollary which I have not seen stated elsewhere. Consider the following program:

```
begin integer $a, b, c, m$;
    for $m := 3, m + 1$ while true do
        for $a := 1$ step 1 until $m$ do
            begin for $b := 1$ step 1 until $a$ do
                    for $c := a$ step 1 until $a + b$ do
                        if $a^m + b^m = c^m$ then go to out;
                if $a > 2$ then begin
                for $b := 1$ step 1 until $m$ do
                  for $c := m$ step 1 until $m + b$ do
                    if $m^a + b^a = c^a$ then go to out end
                        if a
            end for a and for m;
    out: end program
```

This, if I have not made any errors in writing it, is a rather inefficient search program for a counter-example to Fermat's last theorem, covering all possible cases in a single denumerable infinity (this is important). Application of the function T to this program would constitute a proof or a refutation of this theorem. (The inefficiency of the program is irrelevant for this purpose; one might well imagine that simplicity would be more helpful.) A similar technique would apply immediately to a great number of unproved conjectures in number theory. Alas, that it cannot be done in this way!

Yours sincerely,
Bryan Higman.

Windy Sayles,
Felden,
Hemel Hempstead.
7 May 1965.

* This *Journal*, January 1965, p. 313.
† *Ibid.*, April 1965, p. 72.

To the Editor,
The Computer Journal.

Dear Sir,

We wish to bring to the notice of your correspondent, Mr. ApSimon ("An impossible program", this *Journal*, April 1965, p. 72), the method of proof by *reductio ad absurdum*, in which a hypothesis is proved false by assuming its truth and deriving a contradiction. In this instance, the hypothesis is:

(i) There is (i.e., it is possible to write) a Boolean function with a routine as argument whose value is **true** if the routine terminates and **false** if it does not.

Call such a function $T[R]$.

From (i) it follows, as in Strachey's letter (this *Journal*, January, 1965, p. 313):

(ii) There is a routine P which terminates if $T[P] =$ **false**, and fails to if $T[P] =$ **true**.

This is a contradiction. Hence the hypothesis (i) is false, which was to be proved.

Yours faithfully,
W. F. Lunnon,
C. F. J. Outred.

Department of Computer Science,
The University,
Manchester 13.
26 May 1965.

[*Correspondence contd. on next page*]