SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 2, April 1986

010

IS SOR COLOR-BLIND?*

LOYCE M. ADAMS† AND HARRY F. JORDAN‡

Abstract. The work of Young in 1950, see Young [1950], [1971], showed that the Red/Black ordering and the natural rowwise ordering of matrices with Property A, such as those arising from the 5-point discretization of Poisson's equation, lead to SOR iteration matrices with identical eigenvalues. With the advent of parallel computers, multicolor point SOR schemes have been proposed for more complicated stencils on 2-dimensional rectangular grids, see Adams and Ortega [1982] for example, but to our knowledge, no theory has been provided for the rate of convergence of these methods relative to that of the natural rowwise scheme.

New results show that certain matrices may be reordered so the resulting multicolor SOR matrix has the same eigenvalues as that for the original ordering. In addition, for a wide range of stencils, we show how to choose multicolor orderings so the multicolor SOR matrices have the same eigenvalues as the natural rowwise SOR matrix. The strategy for obtaining these orderings is based on "data flow" concepts and can be used to reach Young's conclusions above for the 5-point stencil.

The importance of these results is threefold. Firstly, a constructive and easy means of finding these multicolorings is a direct consequence of the theory; secondly, multicolor SOR methods can be found that have the same rate of convergence as the natural rowwise SOR method for a wide range of stencils used to discretize partial differential equations; and thirdly, these multicolor SOR methods can be efficiently implemented on a wide class of parallel computers.

Keywords. multi-color ordering, SOR, data flow, parallel processing
AMS (MOS) subject classifications. 65, 68

1. Introduction. The successive overrelaxation (SOR) iterative method can be used to solve a linear system of equations,

$$A\mathbf{u} = \mathbf{b} \tag{1}$$

and is guaranteed to converge if the matrix A is symmetric and positive definite and the relaxation factor, ω, is in the interval $0 < \omega < 2$. If we express A as,

$$A = D - L - U \tag{2}$$

where D, L, and U are the diagonal, strictly lower and upper triangular parts of A respectively, the SOR iteration matrix, $\mathcal{L}_\omega$, is given by (3).

$$\mathcal{L}_\omega = (D - \omega L)^{-1}(\omega U + (1-\omega)D). \tag{3}$$

If we reorder the equations in (1) to get the system $\hat{A}\hat{\mathbf{u}} = \hat{\mathbf{b}}$, the resulting SOR matrix $\hat{\mathcal{L}}_\omega$ is not guaranteed to have the same eigenvalues as $\mathcal{L}_\omega$ of (3) and hence the convergence rates of the two SOR schemes may be different.

The matrix A frequently arises from the discretization of an elliptic partial differential equation on a rectangular region by a local stencil and by numbering the grid nodes in the natural rowwise fashion (left to right, bottom to top). For example, for Poisson's equation on a rectangle, we can use the 5-point stencil and number the grid as shown in Fig. 1. Young [1950], [1971] showed that the Red/Black ordering of the nodes as shown in Fig. 2 and the natural rowwise ordering of Fig. 1 led to SOR iteration matrices with the same eigenvalues. Knowing that these SOR matrices have the same eigenvalues is important; since for parallel computers we can choose the

* Received by the editors March 18, 1984, and in revised form November 21, 1985.

† Institute for Computer Applications in Science and Engineering, Hampton, Virginia, and Department of Mathematics, University of California, Los Angeles, California 90024. This research paper was supported by the National Aeronautics and Space Administration under contracts NAS1-17070 and NAS1-17130 while the author was in residence at ICASE, NASA Langley Research Center, Hampton, Virginia 23665.

‡ University of Colorado, Boulder, Colorado 80309.

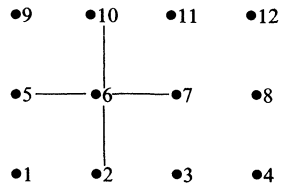

FIG. 1. *5-point stencil and natural rowwise ordering.*

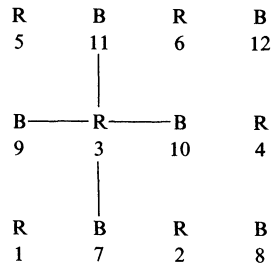

FIG 2. *5-point stencil and red/black ordering.*

Red/Black ordering instead of the natural rowwise ordering without a degradation in the asymptotic convergence rate. More parallelism is achieved with this ordering since nodes of the same color are not neighbors, which implies all Red and then all Black nodes may be updated simultaneously.

With the advent of parallel computers, multicolor point SOR schemes have been proposed for more complicated stencils on 2-dimensional rectangular grids, see Adams and Ortega [1982] for example, but to our knowledge no theory has been provided for the rate of convergence of these methods relative to that of the natural rowwise scheme. This paper will show that the geometry of the stencil can be used to derive a coloring for the region such that the multicolor and natural rowwise SOR iteration matrices have the same eigenvalues. The fact that this is true for a class of stencils containing those of interest in partial differential equations means that highly parallel iterative methods can be formulated which converge equally as well as their sequential counterparts.

In § 2, we describe how trying to implement natural rowwise SOR for a 9-point stencil on the Denelcor HEP, Patel and Jordan [1984], led to a strategy based on data flow ideas for finding orderings that are highly parallel and that produce the same iterates as the natural rowwise iteration. The ideas in § 2 are formalized in § 3 where we prove that the SOR iteration with the orderings generated by the data flow strategy has the same asymptotic rate of convergence as the SOR iteration with particular multicolor orderings. In § 4 we show how to find these multicolor orderings for a wide range of stencils. In § 5, we describe some interesting implementation issues for multicolor SOR on various parallel computers. Finally, in conclusion, we summarize our results, mention generalizations of our ideas to block SOR and multiple equations per grid point, and list unanswered questions.

2. Parallelizing the natural rowwise SOR algorithm. In the multiple instruction stream, or MIMD, environment such as that of the Denelcor HEP, it is useful to investigate speeding up the sequential rowwise SOR computation by using multiple instruction streams called processes. If an arbitrary number of processes are available in the computing environment, we want to investigate a MIMD algorithm that implements a parallel SOR iteration that is equivalent to the natural rowwise SOR algorithm.

The important ideas that this approach yields can best be seen by a specific example. We consider the 9-point stencil shown in Fig. 3. The natural rowwise ordering of a rectangular grid imposes the following update rule, or data flow dependencies, for the unknown at the center of the stencil:

NR *stencil rule.* The value at the center of the stencil for iteration $k+1$ can be calculated after the values to the left of and below center (backward neighbors) have been calculated on iteration $k+1$ and the values to the right and above center (forward neighbors) have been calculated on iteration k.

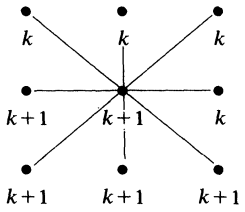

FIG. 3. *9-point stencil.*

This rule is depicted in Fig. 3 for the 9-point stencil.

To formulate the NR stencil rule in the language of data flow (Ackerman [1982]), we look at the set of $n \times m \times K$ updates implied by the rule for unknowns $u^{(k)}(i,j)$ at grid point (i,j) on iteration k, $k=1,\cdots,K$, $i=1,\cdots,n$, and $j=1,\cdots,m$. We will use (i,j) to indicate the point in the ith row and jth column of the grid. These define a computation in a value oriented, or single-assignment, computational model with initial values $u^{(0)}$ and boundary values considered as constants. In this model all values are viewed as having an independent existence with no concept of updating a storage location, so that $n \times m \times K$ storage locations would be needed to store the $n \times m$ unknowns on the K iterations. It is clear that a sequential rowwise sweep for iteration 1 followed by one for iteration 2, and so on, is not the only scheduling of equation evaluations consistent with rule NR. For example, with the 9-point stencil, as soon as $u^{(1)}(1,1)$ and $u^{(1)}(1,2)$ have been evaluated, the computations for $u^{(1)}(1,3)$ and $u^{(1)}(2,1)$ can proceed simultaneously and when $u^{(1)}(1,2)$, $u^{(1)}(2,1)$ and $u^{(1)}(2,2)$ have been computed the iteration 2 value $u^{(2)}(1,1)$ can be produced. If we assume an arbitrarily large number of processors and schedule each computation as early as possible, then many computations, possibly associated with different iterations, will occur simultaneously at points spanning the region.

In general, to cast this $n \times m \times K$ scheduling problem into the form of an iteration on an $n \times m$ region, with only $n \times m$ storage locations required, it is convenient to require symmetry of the stencil as explained below. For a structurally symmetric stencil, like our example in Fig. 3, a point (i,j) is a forward neighbor of all its backward neighbors and a backward neighbor of all its forward neighbors. Thus, by the time it is possible to compute $u^{(k+1)}(i,j)$, the value $u^{(k)}(i,j)$ has been used to compute $u^{(k+1)}$ for all backward neighbors and $u^{(k)}$ for all forward neighbors and these are the only computations which require it. Thus only the "current" iteration value is needed at a specific point (i,j) and the re-use of an $n \times m$ storage array is possible for all the schedulings satisfying the NR rule. The "current" iteration number may be different for different nodes of the region as a function of the specific schedule. We will assume a structurally symmetric stencil henceforth and use rule NR to refer to the $(k+1)$st update of the point at the center of the stencil.

For the 9-point stencil and for many others of interest in PDEs this data flow scheme of scheduling updates as early as allowed by rule NR will be shown to lead to multicolor iterative methods. If each computation takes one time unit then a subset R of grid points will be associated with computations scheduled at time t. At $t+1$ a new subset B of nodes will be computed. If these successively scheduled subsets are disjoint, exhaust the region, and the time difference between subsequent iterations is the same constant for all nodes, then we will show that the resulting time schedule has the form of a multicolor iteration.

To continue our example, we consider the 6 by 5 grid of interior nodes that results from discretizing an 8 by 7 grid with the 9-point stencil with boundary values assumed

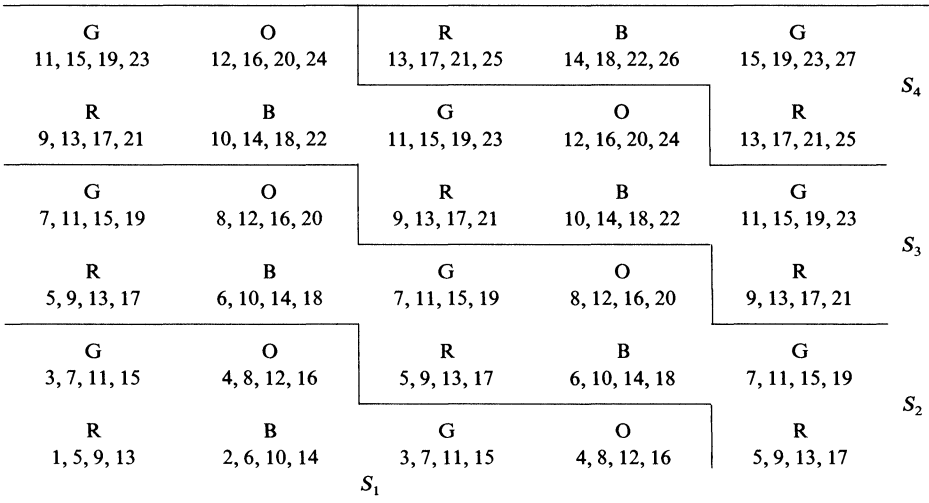

FIG. 4. *Earliest times for the 9-point stencil.*

to be known. This grid is shown in Fig. 4 where the numbers below each node indicate the earliest times that consecutive iterations can begin at the node according to the NR stencil rule if the computation at a node requires one time unit. Notice that each node updates every $\Delta t = 4$ time units. The solid lines in Fig. 4 separate the nodes into four disjoint sets, denoted S_1, S_2, S_3, and S_4 with set S_1 containing nodes that update for the first time during times 1 to Δt, S_2 containing nodes with first update times in the interval $\Delta t + 1$ to $2\Delta t$, etc. These sets will be formally defined later. The following statements can be made from Fig. 4 by observing these earliest update times. Note that at times (1, 2, 3, 4) iteration 1 can be done on the nodes in S_1, at times (5, 6, 7, 8) iteration 1 can be done on the nodes in S_2 and iteration 2 on the nodes in S_1; at times (9, 10, 11, 12) iteration 1 on set S_3, iteration 2 on S_2 and iteration 3 on S_1 can be done; at times (13, 14, 15, 16) iteration 1 on S_4, iteration 2 on S_3, iteration 3 on S_2, and iteration 4 on S_1 can be done; at times (17, 18, 19, 20) iteration 2 on S_4, iteration 3 on S_3 and iteration 4 on S_2 can be done; at times (21, 22, 23, 24) iteration 3 on S_4, iteration 4 on S_3 can be done; and finally at times (25, 26, 27) iteration 4 can be done on S_4.

At this point, it is instructive to remark on the difference between a *data flow scheme* of assigning times to the nodes and an *ordering* of the nodes as reflected by the order of the equations of A in (1). For the 30 nodes (30 equations) in Fig. 4, the scheme of assigning update times at a node to be the earliest times for which the data for the node was available according to the NR Stencil Rule resulted in 15 unique times being assigned to the nodes on the first iteration; whereas, an ordering must assign the integers 1 to 30 to the nodes. However, there are orderings that are consistent with the NR Stencil Rule that can be constructed by considering the update times for the first iteration that resulted from the data flow scheme. We define these orderings in Definition 1.

DEFINITION 1. An NR *data flow ordering* of the nodes of a grid is an ordering by the first earliest update times according to the NR rule with ties being resolved arbitrarily.

There are many orderings of the 30 equations of A in (1) that do not violate the NR stencil rule, but do not have the same update times for the first iteration as shown in Fig. 4. One such ordering is the natural rowwise, left to right, bottom to top ordering. All orderings that do not violate the NR Stencil Rule are nonmigratory permutations

of one another, as described in Young [1971], and will result in SOR matrices with equal eigenvalues. However, it is the NR data flow orderings that are useful in proving our theorems about particular multicolor SOR iterations.

A crucial observation is that at times (13, 14, 15, 16), nodes from all four sets are being updated and that all nodes which can be updated at a particular time, say 13, are not neighbors. Hence, if we color all nodes Red that are updated at time 13, all nodes Black that are updated at time 14, all nodes Green that are updated at time 15, and all nodes Orange that are updated at time 16 as shown in Fig. 4, it is clear that the schedule of updates for four iterations of SOR consistent with the NR Stencil Rule *contains* one iteration (times 13, 14, 15, 16) of a multicolor SOR iteration (R/B/G/O-SOR) as defined by Adams and Ortega [1982]. If we order the colors R/B/G/O as 1/2/3/4, a closer examination of Fig. 4 also reveals that nodes of color i in S_i are connected only to nodes of colors greater than i in S_{i-1} and only to nodes of colors less than i in S_{i+1}. Also nodes in S_i are not connected to any nodes in sets greater than $i+1$ or less than $i-1$.

A strategy for proving that the SOR iteration matrix that results from ordering the Red equations first, followed by the Black, Green, and then Orange equations will have the same eigenvalues as the natural rowwise SOR iteration matrix has evolved from this example. We now formalize this strategy.

3. Theory. The data flow example of the last section leads us to begin with the following definitions.

DEFINITION 2. A *multicolor*, or c-color, matrix is a $c \times c$ block matrix of the form

$$M = \begin{bmatrix} D_1 & X_{12} & \cdot & \cdot & X_{1c} \\ X_{21} & D_2 & \cdot & \cdot & X_{2c} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ X_{c1} & \cdot & \cdot & \cdot & D_c \end{bmatrix}$$

where the D_i are diagonal matrices, each associated with a different color, and the X_{ij} are arbitrary.

DEFINITION 3. A *multicolor T matrix* is a block tridiagonal matrix of the form,

$$T_M = \begin{bmatrix} M_1 & L_1 & & & \\ U_1 & M_2 & L_2 & & \\ & U_2 & \cdot & & \\ & & \cdot & M_{s-1} & L_{s-1} \\ & & & U_{s-1} & M_s \end{bmatrix}$$

where M_i, $2 \leqq i \leqq s-1$, are multicolor matrices with c colors numbered $1, \cdots, c$ respectively; M_1 is a multicolor matrix with $c-f+1$ colors numbered $f, \cdots, c$ respectively; M_s is a multicolor matrix with $e < c$ colors numbered $1, \cdots, e$ respectively. In addition, the matrices L_i, $2 \leqq i \leqq s-2$ are strictly lower $c \times c$ block triangular matrices; the matrix L_1 is a $(c-f+1) \times (c)$ block matrix with blocks $B_{ij} = 0$ if $j \geqq i+f-1$; the matrix L_{s-1} is a $c \times e$ block matrix with blocks $B_{ij} = 0$ if $j \geqq i$; and the matrices U_i, $i = 1, \cdots, s-1$ have the transposed form of the matrices L_i, $i = 1, \cdots, s-1$ respectively.

It is easy to show that the matrix that results from permuting the rows and columns of T_M to bring nodes of color j from all M_i together into the same block D_j is a multicolor matrix with c colors ordered $1, \cdots, c$ respectively. Such a matrix is called the *multicolor matrix associated with* T_M.

In relation to Fig. 4 with R/B/G/O representing colors 1/2/3/4, the blocks M_i, $i = 1, \cdots, 4$ correspond to the four sets S_i, $i = 1, \cdots, 4$, with the nodes in a given block ordered by colors, the value of $c = 4$, the value of $f = 1$ and the value of $e = 3$. The matrix T_M has the following form where D_{ij} indicates the nodes of color i from block M_j.

$$T_M = \begin{bmatrix}
\begin{bmatrix} D_{11} & \times & \times & \times \\ \times & D_{21} & \times & \times \\ \times & \times & D_{31} & \times \\ \times & \times & \times & D_{41} \end{bmatrix} \begin{bmatrix} 0 & & & \\ \times & 0 & & \\ \times & \times & 0 & \\ \times & \times & \times & 0 \end{bmatrix} & & & \\
\begin{bmatrix} 0 & \times & \times & \times \\ & 0 & \times & \times \\ & & 0 & \times \\ & & & 0 \end{bmatrix} \begin{bmatrix} D_{12} & \times & \times & \times \\ \times & D_{22} & \times & \times \\ \times & \times & D_{32} & \times \\ \times & \times & \times & D_{42} \end{bmatrix} \begin{bmatrix} 0 & & & \\ \times & 0 & & \\ \times & \times & 0 & \\ \times & \times & \times & 0 \end{bmatrix} & & \\
\begin{bmatrix} 0 & \times & \times & \times \\ & 0 & \times & \times \\ & & 0 & \times \\ & & & 0 \end{bmatrix} \begin{bmatrix} D_{13} & \times & \times & \times \\ \times & D_{23} & \times & \times \\ \times & \times & D_{33} & \times \\ \times & \times & \times & D_{43} \end{bmatrix} \begin{bmatrix} 0 & & & \\ \times & 0 & & \\ \times & \times & 0 & \\ \times & \times & \times & \end{bmatrix} & \\
\begin{bmatrix} 0 & \times & \times & \times \\ & 0 & \times & \times \\ & & 0 & \times \\ & & & \end{bmatrix} \begin{bmatrix} D_{14} & \times & \times \\ \times & D_{24} & \times \\ \times & \times & D_{34} \end{bmatrix}
\end{bmatrix}.$$

The associated 4-color matrix is

$$M = \begin{bmatrix}
\begin{bmatrix} D_{11} & & & \\ & D_{12} & & \\ & & D_{13} & \\ & & & D_{14} \end{bmatrix} & & & \\
& \begin{bmatrix} D_{21} & & & \\ & D_{22} & & \\ & & D_{23} & \\ & & & D_{24} \end{bmatrix} & & \\
& & \begin{bmatrix} D_{31} & & & \\ & D_{32} & & \\ & & D_{33} & \\ & & & D_{34} \end{bmatrix} & \\
& & & \begin{bmatrix} D_{41} & & \\ & D_{42} & \\ & & D_{43} \end{bmatrix}
\end{bmatrix}.$$

We now prove our major theorem.

THEOREM 1. *A multicolor* T *matrix and its associated multicolor matrix have* SOR *iteration matrices with the same eigenvalues.*

Proof. Let $\mathcal{L}_{T,\omega}$ be the SOR matrix for the multicolor T matrix; $\mathcal{L}_{c\omega}$ be the SOR matrix for the multicolor matrix and P be the permutation matrix that transforms the multicolor T ordering to the multicolor ordering.

Definition 3 guarantees that a multicolor T matrix is also a banded matrix with semibandwidth equal to $c-1$ with the blocks D_{ij} on the diagonal being diagonal blocks. This means that consecutive iterations of SOR can occur for every node every c time units. Let $Q_i^{(k)}$ represent the set of earliest times, t, that the nodes in blocks D_j in block M_i can be updated on iteration k. Specifically,

$$
\begin{aligned}
&Q_1^{(k)} = \{t \mid (k-1)c+1 \leqq t \leqq kc-f+1\},\\
&Q_i^{(k)} = \{t \mid (k+i-2)c-f+2 \leqq t \leqq (k+i-1)c-f+1\},\ 2 \leqq i \leqq s-1,\\
&Q_s^{(k)} = \{t \mid (k+s-2)c-f+2 \leqq t \leqq (k+s-2)c-f+1+e\},
\end{aligned} \tag{4}
$$

and it follows that iterations $s-1, s-2, \cdots, 1$ can be done on the nodes in blocks M_i, $i=1,2,\cdots,s-1$ respectively before iteration 1 is started on the nodes in block M_s. Also, we can conclude from (4) that at the c times $(s+i-2)c-f+2$ to $(s+i-1)c-f+1$ inclusively, a multicolor $\mathscr{L}_{c,\omega}$ iteration with colors numbered $1,\cdots,c$ can be done. This suggests that we consider the matrix $\mathscr{L}_{T,\omega}$ in the factored form,

$$\mathscr{L}_{T,\omega} = \mathscr{L}_s \mathscr{L}_{s-1} \cdots \mathscr{L}_2 \mathscr{L}_1 \tag{5}$$

where the $N \times N$ matrix $\mathscr{L}_i$ represents one SOR iteration on the nodes in block M_i of the multicolor T matrix. In particular, let

(1) n_i be the number of nodes in block M_i,

(2) $I_{(ij)}$ be the $N \times N$ matrix with a 1 in the diagonal position of the row associated with the jth node of block M_i and zeros elsewhere;

(3) $B_{(ij)}$ be the $N \times N$ matrix with the row associated with the jth node of block M_i being equal to the row of the Jacobi iteration matrix, $B = D^{-1}(L+U)$, associated with this node and all other rows being zero.

Then $\mathscr{L}_i$ can be written as

$$\mathscr{L}_i = \prod_{j=1}^{n_i} (\omega B_{(ij)} + I - \omega I_{(ij)}) \tag{6}$$

and has the form

$$\mathscr{L}_i = \begin{bmatrix} I_1 & & & & & & \\ & I_2 & & & & & \\ & & I_{i-1} & & & & \\ & & X & X & X & & \\ & & & & & I_{i+1} & \\ & & & & & & I_s \end{bmatrix}. \tag{7}$$

Now, $k+s-1$ iterations of multicolor T SOR can be expressed in terms of k iterations of multicolor SOR as

$$\mathscr{L}_{T,\omega}^{k+s-1} = R P^T \mathscr{L}_{c,\omega}^k P S \tag{8}$$

where

$$
\begin{aligned}
R &= (\mathscr{L}_s \cdots \mathscr{L}_3 \mathscr{L}_2) \cdots (\mathscr{L}_s \mathscr{L}_{s-1} \mathscr{L}_{s-2})(\mathscr{L}_s \mathscr{L}_{s-1})(\mathscr{L}_s),\\
S &= (\mathscr{L}_1) \cdots (\mathscr{L}_{s-2} \cdots \mathscr{L}_2 \mathscr{L}_1)(\mathscr{L}_{s-1} \cdots \mathscr{L}_2 \mathscr{L}_1).
\end{aligned} \tag{9}
$$

Since $\det(\mathscr{L}_{T,\omega}) = (1-\omega)^N$ where N is the size of $\mathscr{L}_{T,\omega}$, it follows that $\mathscr{L}_{T,\omega}$ is nonsingular if $\omega \neq 1$. Hence the factors in (5) and R in (8) are nonsingular whenever $\omega \neq 1$.

So, for $\omega \neq 1$, it follows from (8) that

$$\mathcal{L}_{T,\omega}^{k+s-1} = RP^T \mathcal{L}_{c,\omega}^k PR^{-1} RS. \tag{10}$$

Since the multicolor T matrix is block tridiagonal, the matrix $\mathcal{L}_i$ commutes with $\mathcal{L}_j$ if $j \neq i-1, i+1$ as is easily seen by (7). By applying this fact to the product RS in (10), we get

$$RS = \mathcal{L}_{T,\omega}^{s-1}. \tag{11}$$

Hence, for $\omega \neq 1$,

$$\mathcal{L}_{T,\omega}^k = RP^{-1} \mathcal{L}_{c,\omega}^k PR^{-1} \tag{12}$$

and it follows that $\mathcal{L}_{T,\omega}$ and $\mathcal{L}_{c,\omega}$ have the same eigenvalues.

Finally, the case $\omega = 1$ follows because the eigenvalues of a matrix are continuous functions of the coefficients of the matrix, and the theorem is proved.

Now, consider the five sets that would be formed in Fig. 4 by letting $f = 4$. This results in a grouping of nodes with the earliest first update times of 1, 2–5, 6–9, 10–13, and 14–15 into sets 1 to 5 respectively. This corresponds to assigning the numbers 1/2/3/4 to the colors B/G/O/R respectively with the other values in Definition 2 being $c = 4$ and $e = 2$. The resulting T_M matrix is the same as that for the assignment of 1/2/3/4 to the colors R/B/G/O in the previous example; however, the block structure is different, indicating different associated multicolor matrices. Also, we could let $f = 3$ and then $f = 2$ and effectively describe G/O/R/B and O/R/B/G orderings of the equations. This discussion leads to the following corollary.

COROLLARY 1. *If the multicolor T matrix results from a* NR *data flow ordering of the grid points, the c multicolor* SOR *matrices that arise by letting f in Definition* 2 *vary from* 1 *to c have the same eigenvalues as the natural rowwise* SOR *matrix.*

Proof. From Theorem 1 we conclude that the c multicolor SOR matrices have the same eigenvalues as the multicolor T SOR matrix. However, the multicolor T ordering is the NR data flow ordering which is just a nonmigratory permutation of the natural rowwise ordering and the corollary follows.

For our example, the R/B/G/O, B/G/O/R, G/O/R/B, and the O/R/B/G SOR matrices have the same eigenvalues as the natural rowwise SOR matrix, and thus an iteration done with any of these multicolor matrices will converge at the same asymptotic rate as the iteration using the sequential rowwise matrix.

The question arises whether there are other four color orderings for the 9-point stencil of Fig. 3 that lead to multicolor and natural rowwise SOR matrices that have the same eigenvalues. We provide a partial answer to this question with Corollary 2.

COROLLARY 2. *The multicolor* SOR *matrix associated with the matrix T_M in Definition* 3 *and the multicolor* SOR *matrix that results from ordering the equations of T_M in reverse order have the same eigenvalues whenever T_M is symmetric.*

Proof. Let A_1, $\mathcal{L}_{1,\omega}$ and A_2, $\mathcal{L}_{2,\omega}$ be the respective multicolor T and multicolor T SOR matrices for the forward and reverse orders respectively. Then by Theorem 1, the multicolor SOR matrices have the same eigenvalues as $\mathcal{L}_{1,\omega}$ and $\mathcal{L}_{2,\omega}$ respectively. It remains to show that $\mathcal{L}_{1,\omega}$ and $\mathcal{L}_{2,\omega}$ have the same eigenvalues.

Now, let

$$A_1 = D_1 - L_1 - U_1$$

and

$$A_2 = D_1 - L_2 - U_2$$

where D_i, L_i, and U_i are defined by (2).

If P is the permutation matrix from the forward to the reverse ordering, we have,

$$P^T D_1 P = D_2; \quad P^T L_1 P = U_2, \quad P^T U_1 P = L_2$$

and

$$P^T \mathcal{L}_{1,\omega} P = (D_2 - \omega U_2)^{-1}(\omega L_2 + (1-\omega) D_2).$$

But since A_2 is symmetric, $U_2 = L_2^T$ and

$$P^T \mathcal{L}_{1,\omega} P = [\omega U_2 + (1-\omega) D_2)(D_2 - \omega L_2)^{-1}]^T.$$

Since a square matrix has the same eigenvalues as its transpose, and the eigenvalues of a product AB equals the eigenvalues of BA, it follows from (3) that $\mathcal{L}_{1,\omega}$ and $\mathcal{L}_{2,\omega}$ have the same eigenvalues and the corollary follows.

Therefore, for Fig. 4, the O/G/B/R, R/O/G/B, B/R/O/G, G/B/R/O, and the natural rowwise orderings have SOR matrices with the same eigenvalues whenever the matrix A of (1) is symmetric.

Obviously, there are other 4-color topologies of a rectangular grid that is discretized with the 9-point stencil of Fig. 3. For example, consider the coloring shown in Fig. 5.

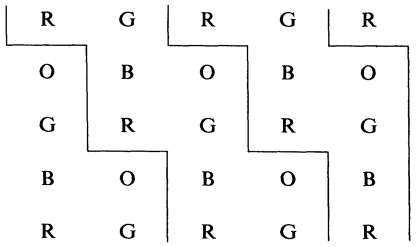

FIG. 5. *Columnwise coloring for the 9-point stencil.*

With the sets indicated in Fig. 5, the R/B/G/O, B/G/O/R, G/O/R/B, O/R/B/G orderings lead to SOR matrices with the same eigenvalues as the natural columnwise (left to right, bottom to top) SOR matrix. Another example, that was given in Adams [1982] is shown in Fig. 6 along with the associated earliest update times for the first two iterations. These update times were obtained by considering two different update rules for the grid. The first rule applies to the R and G points and is identical to the rule in Fig. 3 except the data used from the west neighbor is from iteration k instead of $k+1$. The second rule applies to the O and B points and is like Fig. 3 except the data used from the east neighbor is from iteration $k+1$ instead of iteration k.

The sets indicated above show that $f = 1$, and $e = 4$. Also, the value of f can not be changed and still maintain a multicolor T matrix since all colors must be present in each set for the earliest times shown above. Theorem 1 can be applied to Fig. 6 to prove that R/B/G/O SOR has the same eigenvalues as a multicolor T SOR matrix that is natural rowwise-like in the sense that we update nodes in set 1 before those in set 2, etc., but within sets the ordering is R/B/G/O and is not a nonmigratory permutation of the natural rowwise ordering.

So far, we have shown that the 9-point stencil has multicolor orderings with SOR matrices having eigenvalues identical to those of the SOR matrix for the natural rowwise ordering, the natural columnwise ordering, and a rowwise-like ordering. Next, we turn to the practical questions.

G 11, 15	O 12, 16	G 11, 15	O 12, 16
R 9, 13	B 10, 14	R 9, 13	B 10, 14
G 7, 11	O 8, 12	G 7, 11	O 8, 12
R 5, 9	B 6, 10	R 5, 9	B 6, 10
G 3, 7	O 4, 8	G 3, 7	O 4, 8
R 1, 5	B 2, 6	R 1, 5	B 2, 6

FIG. 6. *Rowwise-like coloring for the 9-point stencil.*

(1) What stencils have multicolor and NR SOR matrices with the same eigenvalues?

(2) How do we find these multicolor orderings?

4. Special stencils and NR-equivalent multicolor orderings. In this section, we define a class of stencils for which a discretized rectangular domain has multicolor orderings with SOR matrices that have the same eigenvalues as the natural rowwise (NR) SOR matrix. In addition, we show how to find these colorings and illustrate the procedure for several well-known stencils. We begin with the following definitions.

DEFINITION 4. A stencil S is a pattern of neighbors for a given node $(\bar{i}, \bar{j})$. The stencil has $(0, 0)$ as its center node and is defined relative to $(\bar{i}, \bar{j})$ as follows: (p, q) is said to be a node in stencil S if for all (i, j) in the region, the node $(i+p, j+q)$ is a neighbor of (i, j) provided it is also in the region.

DEFINITION 5. A stencil that is structurally symmetric about node $(0, 0)$ is a SO-stencil. That is, if node (i, j) is in the stencil, then node $(-i, -j)$ is also in the stencil.

We note that a symmetric stencil leads to a symmetric nonzero pattern in the matrix A of (1), but does not necessarily lead to numerical symmetry of A.

Now, recall from the 9-point stencil example of the last section that every node could be updated every c time units, where c was the number of colors. This fact was reflected in the definition of a multicolor T matrix and was necessary to prove that the R/B/G/O, B/G/O/R, G/O/R/B, and O/R/B/G SOR matrices for Fig. 4 had the same eigenvalues as the NR SOR matrix. However, update rules based on other orderings, as shown in Figs. 5 and 6, also lead to constant update intervals but may not be equivalent to the NR ordering. We now seek a class of stencils for which a constant updating increment is both necessary and sufficient to find multicolor orderings that are equivalent to the NR scheme.

The first step in this direction is to consider stencils for which consecutive nodes in a given row of the grid update one time unit apart. This is ensured by requiring that a SO stencil contain node $(0, 1)$, and likewise, node $(0, -1)$. Such a stencil is called a $(0, 1)$-SO stencil, and the result is proven in Theorem 2.

THEOREM 2. *If a rectangular grid is discretized with a* $(0, 1)$-SO *stencil and an iteration is done at the earliest time according to rule* NR *and requires one time unit to*

complete, then iteration k for node $(i, j+1)$ *begins one time unit later than iteration k for node* (i, j).

Proof. The proof is by induction on k. For each value of k we induct on i and for each value of i we induct on j. The proof is given in its entirety in the Appendix.

We remark that we have been able to find multicolor orderings for SO stencils with the properties that nodes update every c units of time and the SOR matrices have the same eigenvalues as the NR scheme, but do not contain node (0, 1). However, this requirement that a node have an "east" and "west" neighbor is not restrictive for stencils that are commonly used for PDEs as will be illustrated later.

The next step is to further restrict the stencil so that a node updates every c time units. The NR rule says that the time a node can update on iteration $k+1$ is a function of the times its backward neighbors were updated on iteration $k+1$ and the times its forward neighbors were updated on iteration k. However, the times the nodes update on the first iteration are determined by the backward neighbor times only. Since stencils of interest contain a node in row -1, it is convenient to consider stencils for which a node in row -1 is the last backward node to be updated. This will be true for node $(-1, \alpha)$ if nodes strictly below the x-axis and strictly above line L_1 in Fig. 7 are excluded from the stencil as indicated by the darkened nodes in Fig. 7. Node $(-1, \alpha)$ is called the "controlling" backward neighbor. Similarly, we require the last forward neighbor to be updated to be above the x-axis in row γ and column β, $\beta \geqq 0$. This will be true for node (γ, β) if we do not allow nodes above line L_2 to be in the stencil as shown in Fig. 7. The nodes $(-3, 0)$ and $(-3, 1)$ are excluded by symmetry of the stencil since $(3, 0)$ and $(3, -1)$ are excluded as forward neighbors. Node (γ, β) is called the "controlling" forward neighbor. Again, this is not a real restriction for practical stencils. Fig. 7 was the motivation for our major definition below.

DEFINITION 6. A SO-stencil that contains nodes $(0, 1)$, $(-1, \alpha)$, and (γ, β) with $\alpha, \beta \geqq 0$ and $\gamma > 0$ but does not contain nodes

$$\{(y, x), y \leqq -1 | x > -(\alpha+1)y - 1\}$$

or

$$\{(y, x), y \geqq 0 | x > -(\alpha+1)y + \gamma(\alpha+1) + \beta\}$$

is an (α, β, γ)-SO Stencil.

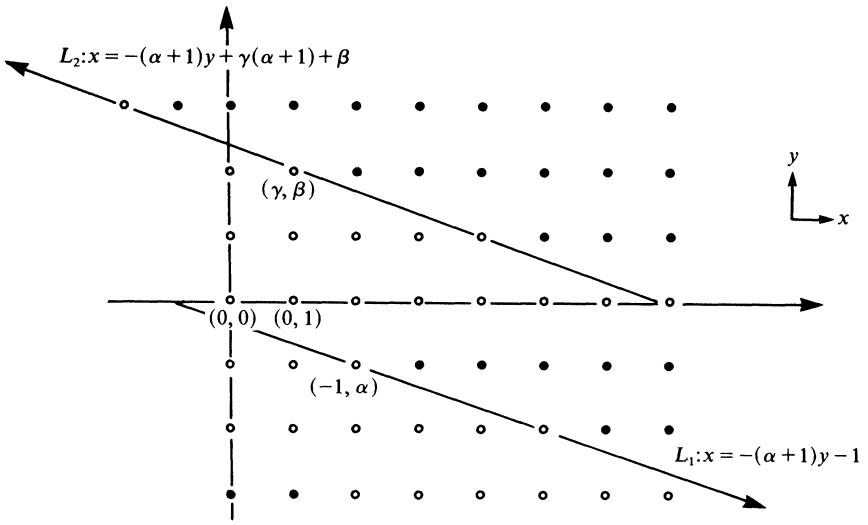

FIG. 7. *Excluded backward and forward neighbors.*

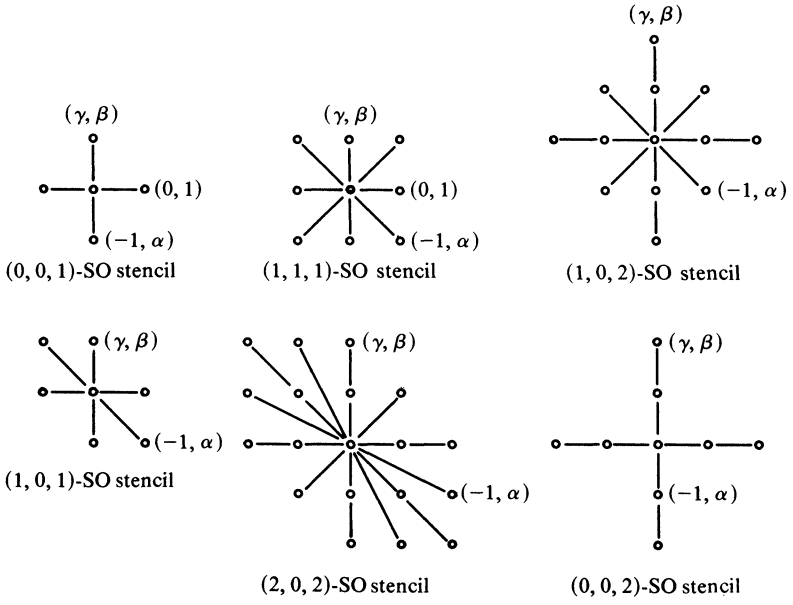

FIG. 8. *The classification of some common stencils.*

The classification of six commonly used stencils as (α, β, γ)-SO stencils is shown in Fig. 8. Hence, we see that it is quite an easy task to find the values of α, β, and γ for a given stencil that satisfy Definition 6. It is also easy to construct many stencils for given values of α, β, and γ.

In Theorem 3 below, we express the value of c in terms of α, β, and γ and describe how to color the grid so that c equals the number of colors.

THEOREM 3. *If a rectangular grid is discretized with an* (α, β, γ)-SO *stencil, the matrix A of* (1) *that results from the* NR *data flow ordering is a multicolor T matrix with*

$$c = \gamma(\alpha+1)+(\beta+1)$$

colors. Furthermore, if the first node is colored f and node (i, j) *is color*

$$[t^{(1)}(i,j)+f-2] \bmod c+1$$

then the blocks M_1, M_r, $r=2, \cdots, s-1$, *and* M_s *contain the nodes in set* S_1, S_r, $r=2, \cdots, s-1$, *and* S_s *below*:

$$S_1 = \{(i,j) | 1 \leqq t^{(1)}(i,j) \leqq c-f+1\},$$

$$S_r = \{(i,j) | (r-1)c-f+2 \leqq t^{(1)}(i,j) \leqq rc-f+1\}, \qquad 2 \leqq r \leqq s,$$

$$S_s = \{(i,j) | (s-1)c-f+2 \leqq t^{(1)}(i,j) \leqq (s-1)c-f+1+e\}.$$

Proof. It is sufficient to prove that all nodes can update every c time units, since the above specifications for M_r, $r=1 \cdots s$, and the coloring rule follow from this fact. The proof is an induction on k and for each k we induct on i and use Theorem 2 to replace the j induction. The proof is given in the Appendix.

Theorem 3 and Corollary 1 show that the c multicolor SOR matrices gotten by letting f vary from 1 to c have the same eigenvalues as the SOR matrix associated with the NR data flow ordering.

5. Implementation of parallel SOR. The data flow view of rowwise sequential SOR as a collection of computations to be scheduled as early as possible makes it clear that a parallel SOR algorithm is possible for a shared memory MIMD machine. It was the investigation of such an algorithm on the HEP computer which stimulated this paper (Patel and Jordan [1984]). The fact that stencils of interest lead to the multicolor scheme which updates fixed, disjoint and exhaustive subsets of the grid points means that implementation on an SIMD or vector computer is also simple. The degree of parallelism of the algorithm, number of processes in MIMD, or vector length in SIMD is $(n \times m)/c$ where c is the number of colors. Since the stencils of interest are limited in extent, multicolor SOR is also suitable for MIMD machines in which data communication costs are significant, such as the processor arrays discussed in Adams [1982].

In implementing multicolor SOR on a vector computer, c vectors of length $(n \times m)/c$ are updated cyclically. Since the vectors consist of subsets of region points and are fixed throughout the iteration, it is appropriate to reorder the data structure to bring all points of a given color into a single vector. If the $n \times m$ array is stored rowwise in memory, the regularity of the coloring pattern for a given stencil on the right sized grid may make it possible to form vectors with a constant stride between elements without reordering the data structure. Some vector architectures will handle constant stride vectors directly. The computation to update the vector of, say, red nodes will be a linear combination of the vectors for nodes of other colors. Two or more shifted versions of a vector may be used, since the stencil centered on a red point may include more than one, say, black neighbor. Other comments on SOR in connection with vector computers can be found in Buzbee et al. [1977] and Adams [1983].

In a shared memory MIMD implementation, the structuring of data is unimportant. Instead, the main issue is that of sychronizing processes so that a scheduling consistent with rule NR results. Potentially, the process updating point (i, j) for the kth time would have to verify that all backward neighbors have been updated k times and all forward neighbors $k-1$ times. However, for (α, β, γ)-SO stencils, it is sufficient to verify that point$(-1, \alpha)$ has been updated k times and point (γ, β) updated $k-1$ times. Synchronization at one point in the forward direction and one point in the backward direction is all that is needed. With shared memory, it is useful to think of the processes moving across the array following a wave of computation. The HEP algorithm mentioned above started with order n parallelism by letting one process sweep each row with the sweeps being as simultaneous as possible under the synchronization rules. Observing that sweeps for subsequent iterations could be started before the current one finished led to the multicolor algorithm with parallelism $(n \times m)/c$. Synchronization on the HEP was done by the producer/consumer mechanism with a computation consuming values from neighbors $(-1, \alpha)$ and (γ, β) and producing two new values, one for each of these neighbors.

On an MIMD machine where the processors are arranged in an array, it is no longer appropriate to let the processes move through the array of processors. A single processor would perform computations for a fixed subregion, preferably containing an equal number of points of each color. Synchronization becomes a by-product of the communications required to pass new iterates to other processors which require them. The storage organization places a node into the local memory of the processor responsible for updating it. This situation is thoroughly covered in Adams [1982].

6. Conclusions. The results of this paper give the practitioner in numerical partial differential equations the assurance that one can use highly parallel multicolor SOR

methods that have exactly the same convergence properties that are associated with the rowwise sequential method, and that these particular multicolor orderings are easily constructed. Thus, in this sense SOR is color-blind.

There are, however, colorings, like those in Figs. 5 and 6, which decouple the stencil on the region which have SOR matrices with the same eigenvalues as columnwise and other, less regular, orderings. The relationship of SOR with these orderings to the natural rowwise SOR is still an open question in general. However, for the 5-point stencil of Fig. 8 the Red/Black, Black/Red, natural rowwise, and natural columnwise orderings all have SOR iteration matrices with equal eigenvalues which corroborates Young's results that were obtained by the use of consistently ordered matrices. In addition, for symmetric matrices, our results show that for the 6-point $(1, 0, 1)$-SO stencil of Fig. 8 all 6 orderings of the unique 3-color pattern which decouples the stencil lead to NR equivalent SOR matrices. This means that columnwise and rowwise SOR have the same asymptotic convergence rate for this stencil. For the 9-point $(1, 1, 1)$-SO of Fig. 8, for symmetric matrices, we only exhibited 8 orderings using 4 colors that were equivalent to the rowwise natural ordering. These 8 are a small fraction of the evaluation orderings on several 4-color topologies which decouple the stencil on the grid.

The multicolor T matrix defined here does not appear to lead to the determination of an optimal relaxation factor, ω, for SOR by relating the eigenvalues to the Jacobi iteration matrix as was the case for Young's T matrices. Nevertheless, if the matrix A in (1) is a Stieltjes matrix, a "good" estimate of ω can be found so that any ordering (and hence multicolor orderings) will yield a convergence rate which is at least one-half the convergence rate associated with the optimal ordering. This result is due to Kahan and can be found in Young [1971].

The development also makes clear some of the issues involved in implementing multicolor SOR. Insensitivity to a cyclic permutation of the colors and the equivalence of a reversed color cycle to a backwards rowwise sweep for symmetric matrices are cases in point. A particularly nice correspondence is that of the points $(-1, \alpha)$ and (γ, β) to the backward and forward synchronization points of the shared memory MIMD implementation.

The technique of using data flow ideas to find orderings clearly extends to regions of three or more dimensions as does the shared memory MIMD approach of engaging multiple processes in simultaneous rowwise sweeps. The equivalence of a fast scheduling of the data flow operations to a multicolor scheme will again depend somewhat on stencil geometry so the idea of an (α, β, γ)-SO stencil will need to be generalized.

The extension to block SOR can be done by defining a block multicolor T matrix and by considering the (α, β, γ)-SO stencil to represent connectivity between blocks of nodes. The convergence rate of block SOR for certain block multicolor orderings of mesh problems leading to irreducible Stieltjes matrices has been compared to a 2-line SOR scheme by O'Leary [1983]. Our results can be used to show that her interesting P^3 ordering leads to a block multicolor T matrix for which the associated block multicolor SOR matrix has the same eigenvalues as a block columnwise SOR matrix.

For point SOR with multiple equations per node a multicolor scheme can be formulated in which equation 1 is evaluated for all points of one color, followed by equation 2 for that color, etc., before moving to the next color. An argument equivalent to the current one can be used to show the SOR matrix has the same eigenvalues as that for a rowwise sequential sweep of the grid points with all equations being evaluated at a point in the same order as for the color groups in the parallel method.

Appendix.

Proof of Theorem 2. Let S represent the stencil

$t^{(k)}(i,j)$ = the earliest time that iteration k can begin for node (i,j),

$S_R = \{i | (i,l) \in S \text{ for some } l \geqq 0\}$,

$S_C = \{j < 0 | (0,j) \in S\}$,

$\Delta_j = \max_{l \geqq 0} l$ where $(j,l) \in S$.

The proof is by induction on k.

The proof for $k=1$ is by induction on i. For $i=1$, the (0, 1)-SO stencil guarantees that $t^{(1)}(1,j+1) = t^{(1)}(1,j)+1$. We assume that

$$t^{(1)}(g,j+1) = t^{(1)}(g,j)+1$$

for $g < i$ and prove that

$$t^{(1)}(i,j+1) = t^{(1)}(i,j)+1. \tag{A.1}$$

Now,

$$t^{(1)}(i,j+1) = \max\{\max_{\{h \leqq j | h-j-1 \in S_C\}} [t^{(1)}(i,h)+1], \max_{\{g<i | g-i \in S_R\}} [t^{(1)}(g,j+1)+\Delta_{g-1}+1]\}. \tag{A.2}$$

If $\{g < i | g - i \in S_R\}$ is empty, then (A.1) follows immediately from an induction on j; otherwise, (A.2) can be written as

$$t^{(1)}(i,j+1) = \max\{\max_{\{h \leqq j | h-j-1 \in S_C\}} t^{(1)}(i,h), \max_{\{g<i | g-i \in S_R\}} [t^{(1)}(g,j)+\Delta_{g-i}+1]+1\}. \tag{A.3}$$

Since

$$t^{(1)}(i,1) = \max_{\{g<i | g-i \in S_R\}} [t^{(1)}(g,1)+\Delta_{g-i}+1],$$

(A.1) follows from an induction on j.

Secondly, we assume $t^{(k)}(i,j+1) = t^{(k)}(i,j)+1$ and prove

$$t^{(k+1)}(i,j+1) = t^{(k+1)}(i,j)+1. \tag{A.4}$$

Now,

$$t^{(k+1)}(i,j+1) = \max\{\max_{\{h \leqq j | h-j-1 \in S_C\}} [t^{(k+1)}(i,h)]+1, \max_{\{g<i | g-i \in S_R\}} [t^{(k+1)}(g,j+1)+\Delta_{g-i}+1], \max_{\{g \geqq i | g-i \in S_R\}} [t^{(k)}(g,j+1)+\Delta_{g-i}+1]\}. \tag{A.5}$$

Since $\{g \geqq i | g - i \in S_R\}$ is not empty for (0, 1)-SO stencils, the last term may be written as

$$\max_{\{g \geqq i | g-i \in S_R\}} [t^{(k)}(g,j)+\Delta_{g-i}+1]+1. \tag{A.6}$$

Now, if $\{g < i | g - i \in S_R\}$ is empty, (A.4) follows from an induction on j; otherwise

the second term in (A.5) can be written as

$$\max_{\{g<i|g-i\in S_R\}} [t^{(k+1)}(g,j)+\Delta_{g-i}+1]+1 \tag{A.7}$$

and (A.4) follows from an induction on j and the theorem is proved.

Proof of Theorem 3. It is sufficient to prove that all nodes can update every c time units, since the block definitions and the coloring rule given in the theorem trivially follow this fact. Since an (α, β, γ)-SO stencil is also a $(0, 1)$-SO stencil, Theorem 1 implies that we only need to prove that

$$t^{(k+1)}(i,1)-t^{(k)}(i,1)=\gamma(\alpha+1)+(\beta+1)=c \tag{A.8}$$

for $k\geqq 1$.

Let S_R, S_C, and Δ_j be as defined in the proof of Theorem 2. Now, the first iteration can begin at node $(i, 1)$ at time,

$$t^{(1)}(i,1)=\max_{\{g<i|g-i\in S_R\}} [t^{(1)}(g,1)+\Delta_{g-1}+1]. \tag{A.9}$$

By the definition of an (α, β, γ)-SO stencil,

$$\Delta_{g-i}\leqq\begin{cases}(i-g)(\alpha+1)-1 & \text{if } g-i<0,\\ (\gamma+i-g)(\alpha+1)+\beta & \text{if } g-i\geqq 0\end{cases} \tag{A.10}$$

with equality when $g-i=-1$ and when $g-i=\gamma$.

By using (A.10), (A.9) can be written as

$$t^{(1)}(i,1)=\max\{\max_{\{g<i-1|g-i+1\in S_R\}} [t^{(1)}(g,1)+\Delta_{g-i}+1],\ t^{(1)}(i-1,1)+(\alpha+1)\} \tag{A.11}$$

and if we choose $t^{(1)}(1,1)=1$, it follows from an induction on i that

$$t^{(1)}(i,1)=1+(i-1)(\alpha+1). \tag{A.12}$$

Now,

$$\begin{aligned}t^{(2)}(i,1)-t^{(1)}(i,1)=\max\{&\max_{\{g<i|g-i\in S_R\}} [t^{(2)}(g,1)-t^{(1)}(i,1)+\Delta_{g-i}+1],\\ &\max_{\{g\geqq i|g-i\in S_R\}} [t^{(1)}(g,1)-t^{(1)}(i,1)+\Delta_{g-i}+1]\}.\end{aligned} \tag{A.13}$$

When $i=1$,

$$t^{(2)}(1,1)-t^{(1)}(1,1)=\max\{[t^{(1)}(\gamma+1,1)+\beta],\ \max_{\{g\geqq 1|g-1\in S_R\}} [t^{(1)}(g,1)+\Delta_{g-i}]\}$$

and $t^{(2)}(1,1)-t^{(1)}(1,1)=1+\gamma(\alpha+1)+\beta$ from (A.10), (A.12), and from an induction of i in (A.13), (A.8) follows for $k=1$.

Next, assume (A.8) is true for $k-1$, and prove true for k.

Now,

$$\begin{aligned}t^{(k+1)}(i,1)-t^{(k)}(i,1)=\max\{&\max_{\{g<i|g-i\in S_R\}} [t^{(k+1)}(g,1)-t^{(k-1)}(i,1)-t+\Delta_{g-i}+1],\\ &\max_{\{g\geqq i|g-i\in S_R\}} [t^{(k-1)}(g,1)-t^{(k-1)}(i,1)+\Delta_{g-i}+1]\}.\end{aligned} \tag{A.14}$$

When $i=1$,

$$t^{(k+1)}(1,1)-t^{(k)}(1,1)=\max_{\{g\geqq 1|g-1\in S_R\}} [t^{(k-1)}(g,1)-t^{(k-1)}(1,1)+\Delta_{g-1}+1]$$

and

$$t^{(k+1)}(1,1) - t^{(k)}(1,1) = t^{(k)}(1,1) - t^{(k-1)}(1,1) = c$$

from (A.10) and (A.12). Hence (A.14) is true for $i = 1$, and by induction on i in equation (A.14), the theorem is proven.

Acknowledgments. Thanks are due Nisheeth Patel for the practical side stimulus which led to a discussion between one of us, Burton Smith, Dennis Gannon, Kenneth Batcher and Gary Rodrigue at a forum in Oregon, expertly hosted by Bill Buzbee of Los Alamos and George Michael of Lawrence Livermore National Laboratories. Enthusiastic and valuable support was provided by Milton Rose, Robert Voigt, and Merrell Patrick both personally and through the ICASE working environment. James Ortega also contributed discussion and enthusiasm for the work.

REFERENCES

W. B. ACKERMAN, *Data flow languages*, Computer, 15 (1982), pp. 15-25.

L. M. ADAMS, *Iterative algorithms for large sparse linear systems on parallel computers*, Ph.D. dissertation, University of Virginia; also published as NASA CR-166027, NASA Langley Research Center, Hampton, VA, November, 1982.

L. M. ADAMS AND J. M. ORTEGA, *A multi-color* SOR *method for parallel computation*, Proc. 1982 International Conference on Parallel Processing, Bellaire, MI, August 1982, pp. 53-58.

L. ADAMS, *An M-step preconditioned conjugate gradient method for parallel computation*, Proc. 1983 International Conference on Parallel Processing, Bellaire, MI, August 1983, pp. 36-43.

B. L. BUZBEE, G. H. GOLUB AND J. A. HOWELL, *Vectorization for the* CRAY-1 *of some methods for solving elliptic difference equations*, High Speed Computer and Algorithm Organization, D. J. Kuck, D. H. Lawrie and A. A. Sameh, eds., Academic Press, New York, 1977, pp. 255-272.

DIANNE P. O'LEARY, *Ordering schemes for parallel processing of certain mesh problems*, this Journal, 5 (1984), pp. 620-632.

N. R. PATEL AND H. F. JORDAN, *A parallelized point successive over-relaxation method on a multiple instruction multiple data stream computer*, submitted to Parallel Computing.

D. YOUNG, *Iterative methods for solving partial differential equations of elliptic type*, Doctoral thesis, Harvard Univ., Cambridge, MA, 1950.

———, *Iterative Solution of Large Linear Systems*, Academic Press, New York, 1971.