2. It guarantees similarity of the iteration matrices of the multicolor method and the original one, while in [1] only equality of spectra is demonstrated.

3. It provides a way to obtain eigenvectors of the multicolor iteration matrix from those of the original one and to construct the similarity matrix connecting these iteration matrices.

4. It applies to structurally nonsymmetric stencils.

5. It applies to general update methods, including point and block relaxation methods and methods other than SOR.

An example showing the applicability of the present coloring method to unstructured grids is given. A comparison of coloring methods for the Gauss-Seidel relaxation in multigrid is presented.

The present approach yields a new point of view and insight which may be useful in future research of update methods.

Acknowledgment. I wish to thank Dr. Irad Yavneh for valuable comments and for introducing Method B in Section 4.

References


and the boundary conditions

\[ u_n = 0 \quad x = 0, \ y = 0 \quad \text{or} \quad z = 0 \]
\[ Du_n + 0.5u = 0 \quad x = 30, \ y = 30 \quad \text{or} \quad z = 30 \]

(where \( \bar{n} \) is the outer normal vector). The finite volume discretization of [4] is used. However, in light of [2], it cannot be applied directly to the original PDE, since this results in strong coupling between domains which are only weakly connected in the PDE and, consequently, in an inconsistent scheme. Hence, it is applied instead to the modified PDE \(-\nabla(D\nabla u) = f\), where

\[
\varepsilon = \frac{d_x + d_y}{2} \min(d_x/d_x, d_y/d_y) \\
D(x, y, z) = \begin{cases} 
\varepsilon & \min(|x - 14| + |y - 14|, |x - 14| + |z - 14|, |y - 14| + |z - 14|) \leq 30/(N_0 - 1) \\
D(x, y, z) & \text{otherwise}
\end{cases}
\]

Although this discretization results in a 7-coefficient stencil, the stencils of the coarse grids are of 27 coefficients. Hence, an 8-color ordering is needed for them.

The right hand side \( f \) is that of [4] [5]. The initial guess is zero. A uniform \( N_0 \times N_0 \times N_0 \) fine grid, with \( N_0 = 31 \), is used. A \( V(1,1) \) multigrid cycle is used. The Gauss-Seidel relaxation, colored by Method A or Method B, is used as a smoothing procedure. Multigrid is iterated until the \( l_2 \) norm of the residual is reduced by 6 orders. The convergence factor displayed in Table 2 is the \( l_2 \) norm of the final residual divided by that of the former one.

Table 2: A comparison of coloring methods for the Gauss-Seidel smoother in multigrid.

<table>
<thead>
<tr>
<th>( d_x )</th>
<th>( d_y )</th>
<th>Method A</th>
<th>Method B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>.126</td>
<td>.138</td>
</tr>
<tr>
<td>( 10^3 )</td>
<td>1</td>
<td>.148</td>
<td>.187</td>
</tr>
<tr>
<td>( 10^6 )</td>
<td>1</td>
<td>.145</td>
<td>.188</td>
</tr>
</tbody>
</table>

We have found, though, that for the 2-d "staircase" problem (Example IV in [2]) with \( N_0 = 65 \) Method B is superior to Method A; the convergence factors of Black Box Multigrid are .161 and .132, respectively, when methods A and B are used. It seems that junction points in \( D \) are treated better by Method A, while Method B is suitable to more simple kinds of discontinuities.

5 Discussion

For general nonsingular linear update methods, a coloring method is introduced for which the resulting iteration matrix is similar to the original one. This result is stronger than that of [1] in the following senses.

1. It applies to general methods and is independent of the definition of stencils and grids.
4 Applications to Multigrid

Here we compare two coloring methods for the Gauss-Seidel relaxation used as a smoothing procedure in multigrid. Let \( d \) be a positive integer denoting the dimension of the problem. Let the grid be a \( d \)-dimensional hypercube of \( N_0^d \) nodes, each of which is denoted by a \( d \)-dimensional vector \( \vec{i} = (i_0, i_1, \ldots, i_{d-1}) \) (where each component \( i_j \) is an integer between \( 0 \) and \( N_0 - 1 \)). The stencil is a \( d \)-dimensional hypercube including \( 3^d \) coefficients. The coloring methods are:

- **Method A:**
  \[
  (n + 1)\text{st color} = \{ \vec{i} | \sum_{j=0}^{d-1} 2^j i_j = n \mod 2^d \}, \quad n = 0, 1, \ldots, 2^d - 1.
  \]

- **Method B:**
  \[
  (\sum_{j=0}^{d-1} 2^j n_j + 1)\text{st color} = \\
  = \{ \vec{i} | i_l = n_l \mod 2, \quad l = 0, 1, \ldots, d - 1 \}, \\
  \quad n_m = 0, 1, \quad m = 0, 1, \ldots, d - 1.
  \]

Method A is obtained from the lexicographical ordering by the coloring method of Section 2.3. Method B was introduced to me by Irad Yavneh; it has the advantage of being easily vectorizable.

We apply a four-level implementation of the Black Box Multigrid method of Dendy ([3] [4]) to the problem

\[
- \nabla (D \nabla u) = f \quad \text{in } \Omega \equiv (0, 30) \times (0, 30) \times (0, 30),
\]

with

\[
\begin{align*}
  j(t) &= \begin{cases} 
    0 & 0 < t < 14 \\
    1 & 14 < t < 30 
  \end{cases}, \\
  D(x, y, z) &= \begin{cases} 
    d_x & (x, y, z) \in \Omega, \quad j(x) + j(y) + j(z) \mod 2 = 0 \\
    d_y & (x, y, z) \in \Omega, \quad j(x) + j(y) + j(z) \mod 2 = 1 \\
    0 & (x, y, z) \not\in \Omega
  \end{cases},
\end{align*}
\]
Theorem 1 \[ L = L_2^{-1} L_{mc} L_2. \]

Corollary 1 Given an eigenvector of \( L \), an eigenvector of \( L_{mc} \) with the same eigenvalue is obtained by applying Algorithm 2 to it until step \( T \) is completed.

Proof: From Theorem 1, we have \( L v = \lambda v \Rightarrow L_{mc} L_2 v = \lambda L_2 v. \)

The matrix \( L_2 \) can be also obtained explicitly by applying Algorithm 2 until step \( T \) to the columns of the identity matrix.

3 Applications to Unstructured Grids

Here we apply the above coloring method to the SOR relaxation of a problem defined on the two-dimensional grid

\[
\begin{array}{cccccc}
1 & 2 & 3 & 4 & 5 \\
6 & 7 & 8 & 9 & 10 \\
11 & 12 & 13 & 14 & 15 \\
16 & 17 & 18 & 19 & 20 \\
21 & 22 & 23 & 24 & 25 \\
26 \\
\end{array}
\]

with the usual 9-point stencil ([1]) for variables 1–25 and assuming that the 26th one depends on variables 21–25. This stencil is obtained, for example, from a linear finite element discretization of a second order elliptic linear PDE on the grid depicted in Figure 1 with boundary conditions of the second or third kind. With \( T \) being the minimal integer satisfying (7), the results of our coloring method are displayed in Table 1. This coloring is not optimal in the sense of minimizing the number of colors. In fact, Methods A and B of Section 4 (with the first and second coordinate there corresponding to the vertical and horizontal direction, respectively) provide coloring methods which use four colors only. However, these coloring methods do not have the property in Theorem 1. Furthermore, the coloring of a graph using a minimum number of colors is known to be an NP-Complete problem; hence, one cannot expect our general coloring method to be optimal in this sense.
2.3 The Coloring Method

We are now able to define the coloring method. Let the first color be the set of indices \( i \) for which \( v_i \) is updated in step \( T + 1 \) of Algorithm 2. Define the \( j \)th color, \( j = 2, 3, \ldots \), to be the set of indices \( i \) for which \( v_i \) is updated in step \( T + j \) of Algorithm 2, minus the first, second, \ldots and \((j - 1)\)st colors. From the construction, the colors are disjoint. Furthermore, they have the following properties.

**Lemma 6** If the \( i \)th color (\( i \geq 1 \)) is empty, so are all the successive ones.

**Proof:** Let \( t = T + i \) and \( \Gamma = \{ j \mid k_j(t - 1) = k_j(T) \} \). Assume that \( \Gamma \) is not empty. Let

\[
n = \min\{ j \in \Gamma \mid k_j(T) = \min_{i \in \Gamma} k_i(T) \}.
\]

From (3)-(4), \( v_n \) is updated in step \( t \), which contradicts the assumption that the \( i \)th color is empty. Consequently, \( \Gamma \) is empty and the lemma follows from the definition of the colors.

**Lemma 7** Let \( c \) be the number of non-empty colors. Then

\[
\bigcup_{j=1}^{c} \text{\( j \)th color} = \{1, 2, \ldots, N\}.
\]

**Proof:** From Lemma 4, we have

\[
\bigcup_{j=1}^{\infty} \text{\( j \)th color} = \{1, 2, \ldots, N\}.
\]

From Lemma 6 and the definition of \( c \) we have that the first \( c \) colors are non-empty, which implies that all the others are. □

Consider the multicolor update method

**Algorithm 4** For \( n = 1, 2, \ldots, c \) do: for every \( i \) in the \( n \)th color, do

\[
x_i \leftarrow \sum_{j \in \Theta_i} a_{i,j} f_j + \sum_{j \in \Omega_i} \gamma_{i,j} x_j.
\]

Let \( I_{mc} \) denote the iteration matrix of Algorithm 4, namely, Algorithm 4 with \( f \equiv 0 \) is represented by \( x \leftarrow I_{mc} x \). Let \( w \) be an arbitrary \( N \)-dimensional initial vector. From Lemmas 1 and 7, the \( i \)th component of \( w \) is updated in \( L_{mc} L_2 w \) exactly \( k_i(T) + 1 \) times. Furthermore, Lemma 1 guarantees that the updates are synchronous. From that and (11), we have

\[
L_3 L_{mc} L_2 = L_3 L_2 L = L^{k+1}.
\]

Since, by assumption, \( L \) is nonsingular, we have from (11) that \((L^{-k} L_3) L_2 \) equals the identity matrix. From this and (12), we have

\[
L = L^{-k} L_3 L_{mc} L_2 = L_2^{-1} L_{mc} L_2.
\]

This gives
Algorithm 3 For $t = T + 1, T + 2, \ldots$ do: for $i = 1, 2, \ldots, N$, if

\begin{align}
    j \in \tilde{\Omega}_i \text{ and } j < i & \Rightarrow k_i(t-1) < k_j(t-1) \quad \text{(8)} \\
    j \in \tilde{\Omega}_i \text{ and } j > i & \Rightarrow k_i(t-1) \leq k_j(t-1) \quad \text{(9)} \\
    k_i(t-1) & < k \quad \text{(10)}
\end{align}

then update $u_i$ (using $f \equiv 0$ in (1)).

Let $u^{(t)}$ denote the vector $u$ right after step $t$ of Algorithm 3. Let $\tau, \tau > T$, be the first integer for which $k_i(\tau) \geq k, 1 \leq i \leq N$. From lemma 4, $\tau$ exists. Clearly, Algorithms 2 and 3 uniquely define linear operators $L_2 : v \to v^{(T)}$ and $L_3 : u \to u^{(\tau)}$. We now show that Algorithm 3, when applied to $L_2 v$, is synchronous.

Lemma 5 Let Algorithm 3 be applied to $L_2 v$, and let $m_i(t), t \geq T$, denote the total number of updates of $v_i$ in steps $1, 2, \ldots, T$ of Algorithm 2 and steps $T + 1, T + 2, \ldots, t$ of Algorithm 3. Then $m_i(t) = \min(k, k_i(t)), 1 \leq i \leq N$. Furthermore, right after step $t$ of Algorithm 3, we have $v_i = (L^{m_i(t)} v)_i, 1 \leq i \leq N$.

Proof: The first part of the lemma is proved by induction on $t, t \geq T$. For $t = T$, it follows from the definition of $k_i(t)$ in Algorithm 2. Assume that it holds for $t-1$ and consider step $t$. Assume first that (10) holds. Then, since (8)–(9) is equivalent to (3)–(4), we have

$$m_i(t) = m_i(t-1) + 1 = k_i(t-1) + 1 = k_i(t)$$

if they hold and

$$m_i(t) = m_i(t-1) = k_i(t-1) = k_i(t)$$

if they do not. Assume now that (10) does not hold. Let $s$ be the last step for which it does hold, namely, $k_i(s-1) < k$. From the above, we have $m_i(s) = k_i(s) = k$. Since $s < t$, we have from (10) that $m_i(t) = k$.

For the second part of the lemma, consider the case for which $v_i$ is updated, namely, (8)–(10) holds. Since (8)–(9) implies (5)–(6), one may use (10) and the first part of the lemma to make the substitution $k \leftarrow m$ in (5)–(6) and obtain

\begin{align}
    j \in \tilde{\Omega}_i \text{ and } j < i & \Rightarrow m_j(t-1) = m_i(t-1) + 1 \\
    j \in \tilde{\Omega}_i \text{ and } j > i & \Rightarrow m_j(t-1) = m_i(t-1),
\end{align}

which means synchronization. □

From Lemma 5 we have the useful result

$$L_3 L_2 = L^k.$$ (11)
Lemma 1 Right after step $t$ of Algorithm 2, $v_i = (L^{k_i(t)}v)_i$, $1 \leq i \leq N$.

Proof: From (2) and (3)–(4), we have that $v_i$ is updated in step $t$ if and only if

\begin{align*}
j \in \Omega_i \text{ and } j < i & \Rightarrow k_j(t-1) = k_i(t-1) + 1 \\
j \in \Omega_i \text{ and } j > i & \Rightarrow k_j(t-1) = k_i(t-1),
\end{align*}

(5) which guarantees synchronization. \(\square\)

The next lemma gives some more information about the dynamics of Algorithm 2.

Lemma 2 For any $t \geq 1$, there is at least one index $i$ for which $v_i$ is updated in step $t$.

Proof: Let

\[ n = \min \{1 \leq j \leq N \mid k_j(t-1) = \min_{1 \leq i \leq N} k_i(t-1) \}. \]

From (3)–(4), $v_n$ is updated in step $t$. \(\square\)

The next lemma shows that each step in Algorithm 2 can be done in parallel.

Lemma 3 If $j \in \Omega_i$, then $v_i$ and $v_j$ are not updated together at any step $t$ ($t \geq 1$).

Proof: Since $j \in \Omega_i \subset \Omega_j$, we have by (2) $i \in \Omega_j$. Without loss of generality, assume $j < i$. Assume that $v_i$ and $v_j$ are updated together for some step $t \geq 1$. Then, from (3), $k_j(t-1) > k_i(t-1)$. But from (4) we have $k_j(t-1) \leq k_i(t-1)$, which is a contradiction. \(\square\)

Next, we show that Algorithm 2 does not stagnate for any of the variables.

Lemma 4 $k_i(t) \rightarrow_{t \rightarrow \infty} \infty$, $1 \leq i \leq N$.

Proof: For each $i$, $\{k_i(t)\}_{t=0}^\infty$ is a monotonically nondecreasing sequence of integers. Assume that there exists an index $i$ for which $k_i(t) \rightarrow_{t \rightarrow \infty} M_i < \infty$. From (2) and (3)–(4), there exist such constants $M_j$ also for every $j \in \Omega_i$ and, hence, also for every $j$ in the connectivity domain of $i$

\[ \Gamma_i = \{m \mid \exists i_1, i_2, \ldots, i_n \text{ s.t. } n \geq 1, i_1 = i, i_n = m, i_{l+1} \in \Omega_i, l = 1, 2, \ldots, n-1\} \cup \{i\}. \]

Let $\tau$ be so large that $k_j(\tau) = M_j \forall j \in \Gamma_i$. Let

\[ n = \min \{j \in \Gamma_i \mid M_j = \min_{i \in \Gamma_i} M_i\}. \]

From (3)–(4), $v_n$ is updated in step $\tau + 1$, which contradicts the definition of $\tau$. \(\square\)

We now define the "complementary" algorithm of Algorithm 2, in a sense which will be made clear below. Let $v^{(t)}$ denote the vector $v$ right after step $t$ of Algorithm 2. Let $T$ be a fixed nonnegative integer. Normally we would like to choose $T$ satisfying

\[ k_i(T+1) > 0, \ 1 \leq i \leq N. \]

(7) From Lemma 4, this is possible. Let

\[ k = \max_{1 \leq i \leq N} k_i(T) + 1. \]

For an arbitrary $N$-dimensional initial vector $u$, define
solution of three-dimensional problems are presented. In Section 5 concluding remarks are made.

2 The Coloring Method and Analysis

2.1 The Update Method

Let $N$ be a positive integer. For $1 \leq i \leq N$, let $\Omega_i$ and $\Theta_i$ be some index sets and $a_{i,j}$ and $\gamma_{i,j}$ $(1 \leq j \leq N)$ nonvanishing scalars. Consider the update method

\textbf{Algorithm 1} For $i = 1, 2, \ldots, N$ do:

\[ x_i \leftarrow \sum_{j \in \Theta_i} a_{i,j} f_j + \sum_{j \in \Omega_i} \gamma_{i,j} x_j, \]

(1)

where $x$ and $f$ are $N$-dimensional vectors. For a fixed $i$, the substitution "$\leftarrow$" in (1) is called an update of the $i$th component of $x$. Such update methods are commonly used for the iterative solution of linear systems. For example, when $\Theta_i = \{i\}$ $(1 \leq i \leq N)$, $\Omega_1 = \{2\}$, $\Omega_N = \{N - 1\}$, $\Omega_i = \{i - 1, i + 1\}$ $(2 \leq i \leq N - 1)$, $a_{i,i} \equiv 1/2$ and $\gamma_{i,j} \equiv 1/2$, Algorithm 1 represents a Gauss-Seidel iteration for the solution of $Ax = f$, where $A = \text{tridiag}(-1, 2, -1)$. More complicated choices of the above parameters may lead to point and block relaxation methods for multi-dimensional problems.

2.2 General Data-Flow Algorithms

Let

\[ \bar{\Omega}_i = \{1 \leq j \leq N \mid j \in \Omega_i \text{ or } i \in \Omega_j\}. \]

From this definition, we have

\[ j \in \bar{\Omega}_i \Leftrightarrow i \in \bar{\Omega}_j. \]

(2)

Let $L$ denote the iteration matrix of Algorithm 1, namely, Algorithm 1 with $f \equiv 0$ is represented by $x \leftarrow Lx$. Assume that $L$ is nonsingular. Let $v$ be an arbitrary $N$-dimensional initial vector. The following algorithm is a general form of the data flow parallel implementation presented in [1].

\textbf{Algorithm 2} Let $k_i(0) = 0$, $1 \leq i \leq N$. For $t = 1, 2, \ldots$ do: for $i = 1, 2, \ldots, N$, if

\[ j \in \bar{\Omega}_i \text{ and } j < i \Rightarrow k_i(t - 1) < k_j(t - 1) \]

(3)

\[ j \in \bar{\Omega}_i \text{ and } j > i \Rightarrow k_i(t - 1) \leq k_j(t - 1) \]

(4)

then update $v_i$ (using $f \equiv 0$ in (1)) and set $k_i(t) = k_i(t - 1) + 1$; else, set $k_i(t) = k_i(t - 1)$.

Note that $k_i(t)$ is the number of times $v_i$ was updated in steps $1, 2, \ldots, t$. In the following lemma, it is shown that the updates in Algorithm 2 are done in a synchronous manner, namely, they are equivalent to those of Algorithm 1 with $f \equiv 0$. 

2
COLORING UPDATE METHODS

Yair Shapira
Computer Science Department, Technion – Israel Institute of Technology, Haifa 32000, Israel.

Abstract

For linear nonsingular update methods, a coloring method is introduced for which the multicolor iteration matrix is similar to the original one. It is general in the sense that its definition is independent of grids and stencils. A method for transforming eigenvectors of the original iteration matrix to those of the multicolor one is introduced. Applications to unstructured grids and to multigrid solution of three-dimensional problems are presented.

1 Introduction

The Successive Over Relaxation (SOR) method introduced in [6] is one of the most popular methods for the solution of linear systems of equations. Although it is inferior to multigrid for certain problems arising from PDEs (see, for example, [2]) it has the advantage of being suitable to parallel implementations. Maximum parallelism is often obtained when the equations are relaxed color by color, where a color is a set of variables which are not explicitly dependent of each other in the system of equations. This order is also often more stable than the standard one, which is especially important when considering indefinite problems and using SOR as a smoother in multigrid methods.

In [1] Adams and Jordan show that, for linear systems arising from two-dimensional problems involving structurally symmetric stencils, the iteration matrix of lexicographically ordered SOR and that of multicolor SOR (for certain coloring) have the same spectrum. This, however, does not imply similarity of the iteration matrices. In this note, an algebraic framework, which is more general in the sense that neither stencils nor grids are needed, is considered. For general update methods, including point and block SOR, colors are defined such that the iteration matrix for the resulting multicolor method is similar to that of the original one. The result of [1] is obtained as a special case, provided that the iteration matrix of the lexicographically ordered SOR is nonsingular (this requirement is not an essential restriction; in fact, one may consider the situation after \( n_0 \) relaxations, where \( n_0 \) is the maximal size of a nilpotent Jordan block of the iteration matrix, and restrict the operators to the subspace obtained by excluding the (semi-) eigenvectors corresponding to these blocks).

In Section 2 the coloring method is introduced and analyzed. In Section 3 an application to a two-dimensional unstructured grid is presented. In Section 4 applications to multigrid