SIAM J. SCI. COMPUT.
Vol. 27, No. 5, pp. 1513–1533

A NEW BLOCK PARALLEL SOR METHOD AND ITS ANALYSIS*

DEXUAN XIE†

Abstract. As the development of the PSOR method (a point parallel SOR method by mesh domain partitioning proposed in [*SIAM J. Sci. Comput.*, 20 (1999), pp. 2261–2281], this paper introduces a new mesh domain partition and ordering (the multitype partition and ordering), and proposes a new block parallel SOR (BPSOR) method for numerically solving 2-dimensional (2D) or three-dimensional (3D) elliptic boundary problems. A general mathematical analysis shows that the BPSOR method can have the same asymptotic convergence rate as the corresponding sequential block SOR method if the coefficient matrix of the block linear system is "consistently ordered." It also shows that the original sequential ordering can be maintained in the parallel implementation of the BPSOR method so that the BPSOR method can be effectively applied to solve complex elliptic boundary problems. Furthermore, three particular multitype orderings are proposed based on strip and block mesh partitions, which lead to three effective BPSOR methods for solving the five-point like linear systems (in two dimensions) and the seven-point like linear systems (in three dimensions). In addition, it is shown that the PSOR method can be generated from BPSOR if each block equation is solved approximately by only one point SOR iteration. Thus, the PSOR method is well defined without involving any interprocessor data communication operations, and extended to solving three-dimensional (3D) problems. Finally, numerical results are presented which confirm the theoretical results and show that BPSOR has a good parallel performance on a parallel MIMD computer.

Key words. block SOR, PSOR, domain decomposition, parallel computing

AMS subject classifications. 65Y05, 65F10

DOI. 10.1137/040604777

1. Introduction.

The successive overrelaxation (SOR) method [28] is an important classic iterative method for solving large-scale sparse linear systems arising from a finite element or finite difference discretization of the elliptic boundary problem. It is also widely used to produce efficient preconditioners [10, 14, 28] and robust smoothers [10, 22, 25, 27]. Due to the efficiency and simplicity in implementation, the SOR method is often used in scientific computing. For example, in protein simulations, three widely used biomolecular simulation packages, CHARMM [5], DelPha [19], and UHBD [7], include the SOR method as the default solver for the linearized Poisson–Boltzmann equation (an elliptic Dirichlet boundary problem with discontinuous coefficients), which models the electrostatic potential energy of a solvate protein system through treating the solvent and the protein as two different continuum dielectric materials [6, 13, 20]. However, the SOR method using the natural ordering is sequential. It must be parallelized to be implemented on a parallel computer.

Various parallel SOR methods have been developed by using multicolor ordering schemes and domain decomposition techniques [1, 2, 4, 8, 9, 12, 15, 16, 17, 18, 23, 24, 26, 28]. Usually, a parallel SOR method defined by multicolor techniques can have a faster convergence rate but may be more difficult to apply for solving complex elliptic boundary problems than a parallel SOR method constructed from a domain decomposition. To improve the convergence rate of a domain decomposition algorithm, a novel mesh partition strategy was proposed in [26] which led to an efficient parallel

*Received by the editors March 4, 2004; accepted for publication (in revised form) October 19, 2004; published electronically February 3, 2006. This work was supported by the National Science Foundation through grant DMS-0241236.

http://www.siam.org/journals/sisc/27-5/60477.html

†Department of Mathematical Sciences, University of Wisconsin, Milwaukee, WI 53201-0413 (dxie@uwm.edu).

SOR method called the PSOR method. The PSOR method was shown to retain the convergence rate of the sequential SOR method [26]. However, the PSOR method was only considered as a point iterative algorithm for solving 2-dimensional (2D) elliptic boundary value problems, and its definition involved a usage of interprocessor data communication operations.

As the development of the PSOR method, this paper proposes a new domain decomposition scheme called the multitype partition. The idea of the multitype partition can be briefly described as follows: the mesh domain Ω_h is first divided into p subdomains, $\{\Omega_{h\mu}\}_{\mu=1}^{p}$, and then the mesh points of each subdomain are grouped into t types, $\{\Omega_{h,\mu}^{i}\}_{i=1}^{t}$, such that each type is coupled only with different adjacent types and at least one of the adjacent types is located in a neighboring subdomain. Here p is the number of processors, and t is less than or equal to the number of neighboring subdomains. The multitype ordering is then defined by numbering the tp type subdomains $\{\Omega_{h,\mu}^{i}\}$ from $\mu = 1$ to p for $i = 1$ to t, while the original ordering is kept within each type subdomain.

This paper then proposes a new block parallel SOR (BPSOR) method by using the multitype partition. A general mathematical analysis shows that the BPSOR method can have the same convergence rate as the corresponding sequential block SOR method if the coefficient matrix of the block linear system is "consistently ordered." (Note that a consistently ordered matrix in this paper usually refers to the blocks of the matrix being consistently ordered. See Definition 9-2.2 in [11] for details.) This paper also introduces three particular multitype partitions based on strip and block mesh partitioning for solving 5-point like (in two dimensions) and 7-point like (in three dimensions) finite difference equations. The three corresponding BPSOR methods are shown, separately, to have the same convergence rate as the corresponding sequential block SOR method.

Two matrix forms of the BPSOR iterative expression are obtained in this paper. One is formulated in the original sequential ordering, and the other in the multitype ordering. With the matrix form formulated by the multitype ordering, the BPSOR method is shown to be equivalent to the normal block SOR method for solving the reordered block linear system by the multitype ordering. Hence, the BPSOR method can be analyzed in the framework of the classic SOR theory [28]. On the other hand, with the matrix form formulated by the original sequential ordering, the BPSOR method is shown to be an iterative method for solving the original block linear system. This implies that the BPSOR method can be implemented easily in parallel on an MIMD parallel computer without any reordering scheme of the unknowns.

In addition, this paper shows that the point PSOR method can be defined as an inexact BPSOR method in which each block equation is solved approximately by only one point sequential SOR iteration. As an inexact BPSOR method, the PSOR method is now well defined without involving any interprocessor data communication operations and extended to solving 3-dimensional (3D) problems. Since the BPSOR theory is developed based on the work of PSOR in [26], some notation and analysis techniques from [26] are kept in this paper for consistency.

Numerical experiments were made on a parallel MIMD computer, the SGI (Silicon Graphics, Inc.) Origin 2000 at the University of Wisconsin-Milwaukee, which has 16 R12000 400 MHz processors. The test model problem is a 7-point finite difference approximation to the Poisson equation on a unit cubic domain. The parallel program is written in FORTRAN 77 and MPI [3]. Each block equation was solved either by the sparse LDL^T subroutine from the SGI's Scientific Computing Software Library or

simply by the sequential point SOR method. Numerical results confirm that BPSOR has the same convergence rate as the corresponding sequential block SOR method. They also show that BPSOR has good parallel performances. For examples, speedups around 13 to 16 were obtained on the sixteen processors of the SGI Origin 2000 for solving the model problem with $h = 1/65$ and $1/129$. Moreover, numerical results show that BPSOR can significantly improve the convergence rate and parallel performance of the point PSOR method. For example, BPSOR is found to take about 16 times fewer iterations and about 16 times less interprocessor communication time than the point PSOR method for the model problem with grid size $h = 1/65$ on eight processors. Here the number of iterations is determined by an iteration termination rule. Even with the point SOR method as a solver of each block equation, test results show that the total CPU time of BPSOR can be less than that of PSOR. Clearly, the total CPU time of BPSOR can be further reduced if each block equation of BPSOR is solved by a more efficient iterative algorithm (such as the multigrid method).

It is possible to generate efficient parallel preconditioners and robust parallel smoothers from the BPSOR method through a proper selection of a local solver, a mesh domain partition, and an accuracy of local solutions. Also, the multitype ordering can be extended to parallelize the sequential Schwartz multiplicative domain decomposition method [21]. Such research topics will be investigated in the sequential papers.

The remainder of the paper is organized as follows. Section 2 introduces the multitype partition and the BPSOR method. Section 3 analyzes the BPSOR method. Section 4 defines three particular and useful BPSOR methods based on strip and block partitionings. Numerical results are presented in section 5.

2. The multitype partition and the BPSOR method.

2.1. The block linear system by the multitype partition. Consider a large-scale system of linear equations that arises from a finite element or finite difference approximation to a 2D or 3D elliptic Dirichlet boundary problem. Let the mesh domain Ω_h be a set of grid points on which the unknowns of the linear system are defined with a grid size of h. To solve the linear system in parallel on p processors of a multiprocessor computer, the mesh domain Ω_h is divided into p disjoint subdomains $\Omega_{h,\mu}$ with $\mu = 1, 2, \ldots, p$ such that

$$\Omega_h = \Omega_{h,1} \cup \Omega_{h,2} \cup \cdots \cup \Omega_{h,p} \quad \text{and} \quad \Omega_{h,\mu} \cap \Omega_{h,\nu} = \emptyset \quad \text{for } \mu \neq \nu. \tag{2.1}$$

Based on domain decomposition (2.1), the linear system can be formulated into p block linear equations:

$$\sum_{\nu=1}^{p} A_{\mu\nu} \mathcal{U}_\nu = \mathcal{F}_\mu, \quad \mu = 1, 2, \ldots, p, \tag{2.2}$$

where $\mathcal{U}_\nu$ is the vector of unknowns defined on Ω_ν, $\mathcal{F}_\mu$ is a given right-hand side vector on $\Omega_{h,\mu}$, $A_{\mu\mu}$ is a nonsingular matrix defined on $\Omega_{h,\mu}$, and $A_{\mu\nu}$ denotes the matrix that indicates the connection of Ω_μ with Ω_ν. If the matrix $A_{\mu\nu}$ with $\mu \neq \nu$ contains at least one nonzero entry, Ω_μ is said to be connected to Ω_ν. Two subdomains are referred to as two neighboring subdomains if they are connected and the intersection of their closures is nonempty. It is assumed that the matrix $A_{\mu\nu}$ with $\mu \neq \nu$ is nonzero if and only if Ω_μ and Ω_ν are two neighboring subdomains.

According to the connection information of each subdomain with its neighboring subdomains, the grid points of each subdomain can be grouped into t different types

such that each type is connected only to its different adjacent types, and at least one of the adjacent types is located in a neighboring subdomain. Here the number t is less than or equal to the number of neighboring subdomains, and two types are called two adjacent types only if they are connected and have a nonempty closure intersection. If $\Omega_{h,\mu}^i$ denotes the ith type of the μth subdomain, then the multitype partition of the mesh domain Ω_h is defined by

$$\Omega_{h,\mu} = \Omega_{h,\mu}^1 \cup \Omega_{h,\mu}^2 \cup \cdots \cup \Omega_{h,\mu}^t \quad \text{and} \quad \Omega_{h,\mu}^i \cap \Omega_{h,\mu}^j = \emptyset \quad \text{for } i \neq j \tag{2.3}$$

for $\mu = 1, 2, \ldots, p$.

Based on the multitype partition (2.3), a system of tp block linear equations is formulated as

$$\sum_{\nu=1}^{p} \sum_{j=1}^{t} A_{\mu\nu}^{ij} \mathcal{U}_\nu^j = \mathcal{F}_\mu^i, \quad i = 1, 2, \ldots, t \text{ and } \mu = 1, 2, \ldots, p, \tag{2.4}$$

where $\mathcal{U}_\mu^i$ and $\mathcal{F}_\mu^i$ denote the vectors defined on $\Omega_{h,\mu}^i$, $A_{\mu\mu}^{ii}$ is a nonsingular matrix defined on $\Omega_{h,\mu}^i$, and $A_{\mu\nu}^{ij}$ denotes the matrix that indicates the connections of $\Omega_{h,\mu}^i$ with $\Omega_{h,\nu}^j$. In the multitype partition, the submatrix $A_{\mu\nu}^{ij}$ with $\mu \neq \nu$ and $i \neq j$ is nonzero only if $\Omega_{h,\mu}^i$ and $\Omega_{h,\nu}^j$ are two adjacent types. Specifically, $A_{\mu\nu}^{ij} = 0$ for all $i \leq j$ if $\mu < \nu$ and for all $i \geq j$ if $\mu > \nu$ so that the block linear system of (2.4) can be simplified as

$$\sum_{j=1}^{t} A_{\mu\mu}^{ij} \mathcal{U}_\mu^j + \sum_{\nu=\mu+1}^{p} \sum_{j=1}^{i-1} A_{\mu\nu}^{ij} \mathcal{U}_\nu^j + \sum_{\nu=1}^{\mu-1} \sum_{j=i+1}^{t} A_{\mu\nu}^{ij} \mathcal{U}_\nu^j = \mathcal{F}_\mu^i, \tag{2.5}$$

where $\mu = 1, 2, \ldots, p$ and $i = 1, 2, \ldots, t$. Many submatrices $A_{\mu\nu}^{ij}$ in (2.5) may still be zero, but they are retained in (2.5) to include various multitype partitions.

The linear system can be expressed in a matrix equation if a global ordering of the unknowns is given. A natural way to order the tp-type subdomains $\{\Omega_{h,\mu}^i\}$ is to order them from $i = 1$ to t for each value of μ from 1 to p. Such a global ordering will be referred to as the natural ordering.

In the natural ordering, the block linear system (2.5) can be expressed in the matrix form

$$Au = f, \tag{2.6}$$

where u and f are two collective vectors defined by

$$u = \begin{pmatrix} \mathcal{U}_1 \\ \mathcal{U}_2 \\ \vdots \\ \mathcal{U}_p \end{pmatrix} \text{ with } \mathcal{U}_\mu = \begin{pmatrix} \mathcal{U}_\mu^1 \\ \mathcal{U}_\mu^2 \\ \vdots \\ \mathcal{U}_\mu^t \end{pmatrix}, \; f = \begin{pmatrix} \mathcal{F}_1 \\ \mathcal{F}_2 \\ \vdots \\ \mathcal{F}_p \end{pmatrix} \text{ with } \mathcal{F}_\mu = \begin{pmatrix} \mathcal{F}_\mu^1 \\ \mathcal{F}_\mu^2 \\ \vdots \\ \mathcal{F}_\mu^t \end{pmatrix}, \tag{2.7}$$

and A is a $tp \times tp$ partitioned block matrix defined by

$$A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1p} \\ A_{21} & A_{22} & \cdots & A_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ A_{p1} & A_{p2} & \cdots & A_{pp} \end{bmatrix} \text{ with } A_{\mu\mu} = \begin{bmatrix} A_{\mu\mu}^{11} & A_{\mu\mu}^{12} & \cdots & A_{\mu\mu}^{1t} \\ A_{\mu\mu}^{21} & A_{\mu\mu}^{22} & \cdots & A_{\mu\mu}^{2t} \\ \vdots & \vdots & \ddots & \vdots \\ A_{\mu\mu}^{t1} & A_{\mu\mu}^{t2} & \cdots & A_{\mu\mu}^{tt} \end{bmatrix}, \tag{2.8}$$

and

$$A_{\mu\nu} = \begin{cases} \begin{bmatrix} 0 & 0 & \cdots & 0 \\ A_{\mu\nu}^{21} & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ A_{\mu\nu}^{t1} & \cdots & A_{\mu\nu}^{t,t-1} & 0 \end{bmatrix} & \text{for } \mu < \nu, \\ \begin{bmatrix} 0 & A_{\mu\nu}^{12} & \cdots & A_{\mu\nu}^{1t} \\ 0 & 0 & \ddots & \vdots \\ \vdots & \vdots & \ddots & A_{\mu\nu}^{t-1,t} \\ 0 & 0 & \cdots & 0 \end{bmatrix} & \text{for } \mu > \nu. \end{cases} \tag{2.9}$$

Here the block matrix A is assumed to be sparse, symmetric positive definite, or consistently ordered.

2.2. The sequential block SOR method by the *t*-type partition. Let the block matrix A be split into the form

$$A = D - L - U, \tag{2.10}$$

where L and U are the two strictly block lower- and upper-triangular matrices, respectively, and D is the block diagonal matrix. They are defined as

$$D = \operatorname{diag}(D_1, D_2, \ldots, D_p) \text{ with } D_\mu = \operatorname{diag}(A_{\mu\mu}^{11}, A_{\mu\mu}^{22}, \ldots, A_{\mu\mu}^{tt}), \tag{2.11}$$

$$L = -\begin{bmatrix} L_1 & 0 & \cdots & 0 \\ A_{21} & L_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ A_{p1} & \cdots & A_{p(p-1)} & L_p \end{bmatrix}, \quad U = -\begin{bmatrix} U_1 & A_{12} & \cdots & A_{1p} \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & U_{p-1} & A_{(p-1)p} \\ 0 & \cdots & 0 & U_p \end{bmatrix},$$

where the abbreviation $\operatorname{diag}(\alpha_1, \alpha_2, \ldots, \alpha_m)$ denotes the diagonal matrix with the diagonal elements $\alpha_1, \alpha_2, \ldots, \alpha_m$, and

$$L_\mu = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ A_{\mu\mu}^{21} & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ A_{\mu\mu}^{t1} & \cdots & A_{\mu\mu}^{t,t-1} & 0 \end{bmatrix}, \quad U_\mu = \begin{bmatrix} 0 & A_{\mu\mu}^{12} & \cdots & A_{\mu\mu}^{1t} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & A_{\mu\mu}^{t-1,t} \\ 0 & \cdots & 0 & 0 \end{bmatrix}. \tag{2.12}$$

In terms of the above matrices, the block SOR method by the t-type partition is defined by

$$u^{(k+1)} = M_{SOR} u^{(k)} + \omega[D - \omega L]^{-1} f, \tag{2.13}$$

where $u^{(k)}$ is a block vector in the form

$$u^{(k)} = \begin{pmatrix} \mathcal{U}_1^{(k)} \\ \mathcal{U}_2^{(k)} \\ \vdots \\ \mathcal{U}_p^{(k)} \end{pmatrix} \quad \text{with} \quad \mathcal{U}_\mu^{(k)} = \begin{pmatrix} \mathcal{U}_\mu^{1,(k)} \\ \mathcal{U}_\mu^{2,(k)} \\ \vdots \\ \mathcal{U}_\mu^{t,(k)} \end{pmatrix} \quad \text{for } \mu = 1, 2, \ldots, p, \tag{2.14}$$

$\mathcal{U}_\mu^{i,(k)}$ is the kth iterate on $\Omega_{h,\mu}^i$, and M_{SOR} is the SOR iteration matrix defined by

$$M_{SOR} = [D - \omega L]^{-1}[(1-\omega)D + \omega U]. \tag{2.15}$$

See [28] for the details on the analysis of the SOR method.

However, the block SOR method is a sequential iterative method. In fact, on each type subdomain, Ω_μ^i, the component expression of the block SOR method (2.13) can be written as

$$\begin{aligned}\mathcal{U}_\mu^{i,(k+1)} = \omega(A_{\mu\mu}^{ii})^{-1}\Bigg[\mathcal{F}_\mu^i - \sum_{j=1}^{i-1} A_{\mu\mu}^{ij}\mathcal{U}_\mu^{j,(k+1)} - \sum_{j=i+1}^{t} A_{\mu\mu}^{ij}\mathcal{U}_\mu^{j,(k)} \\ - \sum_{\nu=\mu+1}^{p}\sum_{j=1}^{i-1} A_{\mu\nu}^{ij}\mathcal{U}_\nu^{j,(k)} - \sum_{\nu=1}^{\mu-1}\sum_{j=i+1}^{t} A_{\mu\nu}^{ij}\mathcal{U}_\nu^{j,(k+1)}\Bigg] + (1-\omega)\mathcal{U}_\mu^{i,(k)},\end{aligned} \tag{2.16}$$

where $i = 1, 2, \ldots, t$, $\mu = 1, 2, \ldots, p$, and $k = 0, 1, 2, \ldots$. From (2.16) it can be seen that the current new iterate $\mathcal{U}_\mu^{i,(k+1)}$ depends on the previous new iterates $\{\mathcal{U}_\nu^{j,(k+1)}\}_{\nu=1}^{\mu-1}$, which are located on other subdomains, $\{\Omega_{h,\nu}\}_{\nu=1}^{\mu-1}$. Thus, the block SOR method by the multitype partition is sequential in terms of subdomains $\{\Omega_{h,\nu}\}_{\nu=1}^{p}$.

2.3. The BPSOR method. However, the sequential block SOR method by the multitype partition can be easily parallelized by using the multitype ordering, which orders the subdomains $\{\Omega_{h,\mu}^i\}$ from $\mu = 1$ to p for each value of i from 1 to t. In detail, the subdomains $\{\Omega_{h,\mu}^1\}_{\mu=1}^p$ of type 1 is first ordered from $\mu = 1$ to p, then the subdomains $\{\Omega_{h,\mu}^2\}_{\mu=1}^p$ of type 2, and so on, through the subdomains $\{\Omega_{h,\mu}^t\}_{\mu=1}^p$ of type t. Within each type subdomain, the original natural ordering of mesh points is maintained so that each block equation of (2.5) is kept. The multitype ordering with t types will be simply referred to as the t-type ordering.

In the t-type ordering, the iterates $\mathcal{U}_\nu^{j,(k+1)}$ with $j = 1, 2, \ldots, i-1$ and $\nu = \mu, \mu+1, \ldots, p$ will be computed before the calculation of $\{\mathcal{U}_\mu^{i,(k+1)}\}_{\mu=1}^p$. Hence, the p iterates $\{\mathcal{U}_\mu^{i,(k+1)}\}_{\mu=1}^p$ can be defined by

$$\begin{aligned}\mathcal{U}_\mu^{i,(k+1)} = \omega(A_{\mu\mu}^{ii})^{-1}\Bigg[\mathcal{F}_\mu^i - \sum_{\nu=\mu}^{p}\sum_{j=1}^{i-1} A_{\mu\nu}^{ij}\mathcal{U}_\nu^{j,(k+1)} - \sum_{\nu=1}^{\mu}\sum_{j=i+1}^{t} A_{\mu\nu}^{ij}\mathcal{U}_\nu^{j,(k)}\Bigg] \\ + (1-\omega)\mathcal{U}_\mu^{i,(k)} \quad \text{for} \quad \mu = 1, 2, \ldots, p, \quad i = 1, 2, \ldots, t.\end{aligned} \tag{2.17}$$

Here $k = 0, 1, 2, \ldots$, $\mathcal{U}_\nu^{j,(0)}$ is a given initial guess, and $\omega \in (0, 2)$. Since all the iterates on the right-hand side of the above equations are given or computed in the previous steps, the iterative expressions of $\{\mathcal{U}_\mu^{i,(k+1)}\}_{\mu=1}^p$ in (2.17) are completely independent. In addition, these p iterates are defined on p different subdomains, $\{\Omega_{h,\mu}\}_{\mu=1}^p$, respectively. Hence, when each subdomain is assigned to one processor, they can be calculated in parallel on p processors. Therefore, the iterative method defined by (2.17) is a parallel scheme based on the domain decomposition (2.1). For clarity, it will be called the block parallel SOR (BPSOR) method.

A parallel implementation scheme of BPSOR is described in Algorithm 1. Here processor μ contains both subdomain Ω_μ and its ghost boundary, which is a set of the mesh points of the neighboring subdomains that are connected to Ω_μ for $\mu = 1, 2, \ldots, p$.

ALGORITHM 1 (for parallel implementation of BPSOR). *Let u_μ^i be an array to hold the BPSOR iterate $\mathcal{U}_\mu^{i,(k)}$, and $\mathcal{U}_\mu^{i,(0)}$ be a given initial guess. On Processor μ for $\mu = 1$ to p in parallel, do the following steps.*

1. *Set $u_\mu^i := \mathcal{U}_\mu^{i,(0)}$ for $i = 1, 2, \ldots, t$, and $k = 1$.*
2. *Update $\{u_\mu^i\}_{i=1}^t$ by the for-loop on i:*
   for $i = 1, 2, \ldots, t$ do
   (a) *Solve $A_{\mu\mu}^{ii} v_\mu^i = f_\mu^i$ for v_μ^i, where f_μ^i is defined by*

$$f_\mu^i := \mathcal{F}_\mu^i - \sum_{j=1}^{i-1} \sum_{\nu=\mu}^{p} A_{\mu\nu}^{ij} u_\nu^j - \sum_{j=i+1}^{t} \sum_{\nu=1}^{\mu} A_{\mu\nu}^{ij} u_\nu^j.$$

   (b) *Set $u_\mu^i := \omega v_\mu^i + (1-\omega) u_\mu^i$.*
   (c) *Update the ghost boundary values that relate to u_μ^i by interprocessor communication operations.*
   endfor (i)
3. *Set $k := k + 1$ and return to Step 2 for the next step of iteration.*

3. The analysis of the BPSOR method.

3.1. BPSOR as an iterative method using the natural ordering. From Algorithm 1 it can be seen that the BPSOR iterates $\{\mathcal{U}_\mu^{i,(k)}\}$ are calculated from $i = 1$ to t within each processor. Thus, the pt BPSOR iterates $\{\mathcal{U}_\mu^{i,(k)}\}$ can be formulated as a block vector in the natural ordering.

THEOREM 3.1. *If the BPSOR iterates $\{\mathcal{U}_\mu^{i,(k)}\}$ defined in* (2.17) *are formulated as a block vector $u^{(k)}$ in the form* (2.14)*, then the BPSOR method can be regarded as an iterative method using the natural ordering, and it has the following matrix iterative expression:*

$$u^{(k+1)} = M_{BPSOR} u^{(k)} + \omega [D - \omega(B + M)]^{-1} f \quad \text{for} \quad k = 0, 1, 2, \ldots, \tag{3.1}$$

where $\omega \in (0, 2)$, M_{BPSOR} is the BPSOR iteration matrix defined by

$$M_{BPSOR} = [D - \omega(B + M)]^{-1} [(1 - \omega) D + \omega(C + N)], \tag{3.2}$$

D is the block diagonal matrix given in (2.11)*, B and C are the two block diagonal matrices defined by*

$$B = -\operatorname{diag}(L_1, L_2, \ldots, L_p) \quad \text{and} \quad C = -\operatorname{diag}(U_1, U_2, \ldots, U_p), \tag{3.3}$$

and M and N are block triangular matrices defined by

$$N = L - B \quad \text{and} \quad M = U - C. \tag{3.4}$$

Here L, U, L_μ, and U_μ are given in (2.11) *and* (2.12)*.*

Proof. With (3.4), the matrices L and U can be expressed by

$$L = N + B \quad \text{and} \quad U = M + C. \tag{3.5}$$

Thus, the regular splitting (2.10) becomes

$$A = D - B - C - M - N. \tag{3.6}$$

An iterative method using the natural ordering means the one for solving the linear system $Au = f$ given in (2.6). With the splitting formula (3.6), an iterative sequence $\{u^{(k)}\}$ using the natural ordering can be defined by

$$Du^{(k+1)} = (B + M)u^{(k+1)} + f + (C + N)u^{(k)}, \quad k = 0, 1, 2, \ldots.$$

Introducing the relaxation parameter ω into the above iterative sequence gives

$$Du^{(k+1)} = \omega[(B + M)u^{(k+1)} + f + (C + N)u^{(k)}] + (1 - \omega)u^{(k)}. \tag{3.7}$$

Solving the above equation for $u^{(k+1)}$ yields the matrix iterative expression (3.1). Expanding the matrix expression (3.1) into the block component expressions in terms of $\{\mathcal{U}_\mu^{i,(k)}\}$ yields the BPSOR iterative expressions in (2.17). Thus, the block iterative sequence defined by (3.7) is exactly the BPSOR method (2.17). Hence, the expression (3.1) is also the matrix iterative expression of the BPSOR method. Therefore, the BPSOR method can be regarded as an iterative method using the natural ordering. This completes the proof of the theorem.

Because of (3.5), the BPSOR iterative expression (3.1) can be changed to the sequential block SOR iterative expression (2.13) by simply switching the positions of N and M. Such a matrix-switch operation is completed in step (c) of Algorithm 1. In this sense, BPSOR can be regarded as a parallel version of the block SOR method (2.13), which is generated from a novel usage of interprocessor communication operations. □

3.2. BPSOR as the block SOR using the multitype ordering. Let P be a permutation matrix defined by $\hat{u} = Pu$ for u in the form (2.7), and $\hat{u}$ in the following form:

$$\hat{u} = \begin{pmatrix} \hat{\mathcal{U}}_1 \\ \hat{\mathcal{U}}_2 \\ \vdots \\ \hat{\mathcal{U}}_t \end{pmatrix} \text{ with } \quad \hat{\mathcal{U}}_i = \begin{pmatrix} \mathcal{U}_1^i \\ \mathcal{U}_2^i \\ \vdots \\ \mathcal{U}_p^i \end{pmatrix} \text{ for } 1 \le i \le t. \tag{3.8}$$

Obviously, by the t-type ordering, the block linear system $Au = f$ in (2.6) can be reordered into an equivalent block linear system. In terms of P, the reordered linear system can be expressed as

$$\hat{A}\hat{u} = \hat{f}, \tag{3.9}$$

where $\hat{A} = PAP^T$, $\hat{u} = Pu$, and $\hat{f} = Pf$. The matrix form of $\hat{A}$ can be found as

$$\hat{A} = \begin{bmatrix} \hat{A}_{11} & \hat{A}_{12} & \cdots & \hat{A}_{1t} \\ \hat{A}_{21} & \hat{A}_{22} & \cdots & \hat{A}_{2t} \\ \vdots & \ddots & \ddots & \vdots \\ \hat{A}_{t1} & \cdots & \hat{A}_{t,t-1} & \hat{A}_{tt} \end{bmatrix} \tag{3.10}$$

with $\hat{A}_{ij}$ being defined by

$$\hat{A}_{ij} = \begin{cases} \operatorname{diag}(A_{11}^{ii}, A_{22}^{ii}, \ldots, A_{pp}^{ii}) & \text{for} \quad i = j, \\ \begin{bmatrix} A_{11}^{ij} & A_{12}^{ij} & \cdots & A_{1p}^{ij} \\ 0 & A_{22}^{ij} & \cdots & A_{2p}^{ij} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & A_{pp}^{ij} \end{bmatrix} & \text{for} \quad i > j, \\ \begin{bmatrix} A_{11}^{ij} & 0 & \cdots & 0 \\ A_{21}^{ij} & A_{22}^{ij} & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ A_{p1}^{ij} & A_{p2}^{ij} & \cdots & A_{pp}^{ij} \end{bmatrix} & \text{for} \quad i < j. \end{cases} \tag{3.11}$$

THEOREM 3.2. *If the BPSOR iterates* $\{\mathcal{U}_\mu^{i,(k)}\}$ *defined in* (2.17) *are arranged as a block vector* $\hat{u}^{(k)}$ *in the form* (3.8)*, then the BPSOR method defined in* (2.17) *is equivalent to the block SOR method using the multitype ordering, and it has the following matrix iterative expression:*

$$\hat{u}^{(k+1)} = \hat{M}_{SOR}\hat{u}^{(k)} + \omega(\hat{D} - \omega\hat{L})^{-1}\hat{f}, \tag{3.12}$$

where $\hat{M}_{SOR}$ *is the iteration matrix defined by*

$$\hat{M}_{SOR} = (\hat{D} - \omega\hat{L})^{-1}[(1-\omega)\hat{D} + \omega\hat{U}], \tag{3.13}$$

$\hat{D}, \hat{L}$, *and* $\hat{U}$ *are the block diagonal, strictly lower-triangular, and strictly upper-triangular matrices satisfying*

$$\hat{A} = \hat{D} - \hat{L} - \hat{U}, \tag{3.14}$$

and $\hat{A}$ *is the block matrix given in* (3.10).

Proof. By the splitting of $\hat{A}$ in (3.14), it is easy to find that the block SOR method using the multitype ordering has the matrix iterative expression (3.12). In terms of $\mathcal{U}_\mu^{i,(k)}$, the block component expressions of the matrix form (3.12) turn out to be the same as the iterative expressions of the BPSOR method in (2.17). Hence, the BPSOR method is equivalent to the block SOR method using the multitype ordering for solving the reordered linear system (3.9). This completes the proof of the theorem. ∎

3.3. The relationship between the two iteration matrices. Two matrix iterative expressions, (3.1) and (3.12), have been formulated for the BPSOR method by the natural ordering and the t-type ordering. The expression of (3.1) indicates that the BPSOR method can be easily implemented in parallel since the original natural ordering is maintained. From the expression of (3.12) it implies that the BPSOR method can be analyzed under the framework of the classic SOR theory [28].

THEOREM 3.3. *Let* M_{BPSOR} *and* $\hat{M}_{SOR}$ *be the two BPSOR iteration matrices given in* (3.2) *and* (3.13)*, and let* P *be the permutation matrix defined in* (3.8)*. Then*

$$\hat{M}_{SOR} = PM_{BPSOR}P^T. \tag{3.15}$$

Proof. The key step in proving (3.15) is to show the following two identities:

$$\hat{L} = P(B+M)P^T \quad \text{and} \quad \hat{U} = P(C+N)P^T. \tag{3.16}$$

The proof of the first one is given below.

By multiplying both sides of expression (3.1) by P, (3.1) can be rewritten as

$$\begin{aligned}\hat{u}^{(k+1)} = {} & [\hat{D} - \omega P(B+M)P^T]^{-1}[(1-\omega)\hat{D} + \omega P(C+N)P^T]\hat{u}^{(k)} \\ & + \omega[\hat{D} - \omega P(B+M)P^T]^{-1}\hat{f},\end{aligned} \tag{3.17}$$

where the identities $P^TP = I$, $Pu^{(k)} = \hat{u}^{(k)}$, $Pf = \hat{f}$, and $PDP^T = \hat{D}$ have been used. The following two expressions of $\hat{u}^{(1)}$ are then produced from (3.17) and (3.12) by setting $\hat{u}^{(0)} = 0$:

$$\hat{u}^{(1)} = \omega[\hat{D} - \omega P(B+M)P^T]^{-1}\hat{f} \quad \text{and} \quad \hat{u}^{(1)} = \omega[\hat{D} - \omega\hat{L}]^{-1}\hat{f}.$$

Combining them as one identity yields

$$[\hat{D} - \omega P(B+M)P^T]^{-1}\hat{f} = [\hat{D} - \omega\hat{L}]^{-1}\hat{f}.$$

Since $\hat{f}$ is arbitrary, the above identity follows as $[\hat{D}-\omega P(B+M)P^T]^{-1} = [\hat{D}-\omega\hat{L}]^{-1}$. Simplifying it gives $\hat{L} = P(B+M)P^T$.

To prove the second identity of (3.16), substituting $P(B+M)P^T$ by $\hat{L}$ in (3.17) gives

$$\hat{u}^{(k+1)} = [\hat{D} - \omega\hat{L}]^{-1}[(1-\omega)\hat{D} + \omega P(C+N)P^T]\hat{u}^{(k)} + \omega[\hat{D} - \omega\hat{L}]^{-1}\hat{f}.$$

Combining the above expression with (3.12) yields

$$[(1-\omega)\hat{D} + \omega P(C+N)P^T]\hat{u}^{(k)} = [(1-\omega)\hat{D} + \omega U]\hat{u}^{(k)} \quad \text{for all } k \geq 0.$$

Since the above identity holds for any vector of $\hat{u}^{(k)}$, it implies that

$$(1-\omega)\hat{D} + \omega P(C+N)P^T = (1-\omega)\hat{D} + \omega U.$$

Simplifying again gives that $\hat{U} = P(C+N)P^T$.

Finally, with (3.16) and $\hat{D} = PDP^T$,

$$\begin{aligned}\hat{M}_{SOR} &= (\hat{D} - \omega\hat{L})^{-1}[(1-\omega)\hat{D} + \omega\hat{U}] \\ &= (PDP^T - \omega P(B+M)P^T)^{-1}[(1-\omega)PDP^T + \omega P(C+N)P^T] \\ &= P(D - \omega(B+M))^{-1}[(1-\omega)D + \omega(C+N)]P^T = PM_{BPSOR}P^T. \end{aligned} \tag{3.18}$$

This completes the proof of Theorem 3.3. ∎

3.4. Convergence of the BPSOR method. The main convergence results of the BPSOR method are presented in the following theorem.

THEOREM 3.4. *Let M_J, M_{SOR}, and M_{BPSOR} be, respectively, the iteration matrices of the block Jacobi method, the block SOR method, and the BPSOR method. If the block matrix A defined in* (2.8) *is symmetric positive definite, then the BPSOR iterates $u^{(k)}$ converge to the exact solution $u^* = A^{-1}f$ as $k \to \infty$ if and only if its relaxation parameter ω satisfies $0 < \omega < 2$.*

Furthermore, if A is consistently ordered, M_J has real eigenvalues, and the spectral radius $\rho(M_J) < 1$, then

$$\rho(M_{BPSOR}(\omega)) = \rho(M_{SOR}(\omega)) \tag{3.19}$$

$$= \begin{cases} \frac{1}{4}\left[\omega\rho(M_J) + \sqrt{\omega^2\rho(M_J)^2 - 4(\omega-1)}\right]^2 & \text{for } 0 < \omega \le \omega_b, \\ \omega - 1 & \text{for } \omega_b \le \omega < 2, \end{cases} \tag{3.20}$$

and

$$\omega_b = \frac{2}{1 + \sqrt{1 - \rho(M_J)^2}}. \tag{3.21}$$

Here $\rho(\cdot)$ denotes the spectral radius of a matrix, which is defined as the maximum of the moduli of the eigenvalue of the matrix, M_{SOR} and M_{BPSOR} are given in (2.15) *and* (3.2)*, and $M_J = L + U$ with L and U being given in* (2.11)*.*

Proof. Since the reordered matrix $\hat{A}$ defined in (3.10) satisfies $\hat{A} = PAP^T$ and A is symmetric positive definite, then $\hat{A}$ is symmetric positive definite. Also, from Theorem 4.1 in [26] it follows that $\hat{A}$ is consistently ordered if A is consistently ordered. Moreover, in Theorem 3.2, the BPSOR method has been shown to be equivalent to the block SOR method for solving the reordered linear system (3.9). Hence, from the classic SOR theory [28] it can be shown that the BPSOR iterates $\hat{u}^{(k)}$ converge to the exact solution $\hat{u}^*$ if and only if the relaxation parameter ω satisfies $0 < \omega < 2$, and both the spectral radius $\rho(\hat{M}_{SOR})$ and the optimal relaxation value $\hat{\omega}_b$ can be expressed as

$$\rho(\hat{M}_{SOR}) = \begin{cases} \frac{1}{4}\left[\omega\rho(\hat{M}_J) + \sqrt{\omega^2\rho(\hat{M}_J)^2 - 4(\omega-1)}\right]^2, & 0 < \omega \le \hat{\omega}_b, \\ \omega - 1, & \hat{\omega}_b \le \omega < 2, \end{cases} \tag{3.22}$$

and

$$\hat{\omega}_b = \frac{2}{1 + \sqrt{1 - \rho(\hat{M}_J)^2}}, \tag{3.23}$$

where $\hat{M}_J = \hat{L} + \hat{U}$ is the Jacobi iteration matrix of the reordered linear system (3.9), and $\hat{u}^* = Pu^*$ with u^* being the exact solution of the original linear system (2.6).

On the other hand, from the classic SOR theory [28] it follows that $\rho(M_{SOR})$ and ω_b have the same expressions as given in (3.20) and (3.21). That $\hat{M}_J = PM_JP^T$ follows easily from (3.16). Thus, $\rho(\hat{M}_J) = \rho(M_J)$. As a result, the expressions (3.22) and (3.23) become the same as the expressions (3.20) and (3.21), resulting in $\rho(\hat{M}_{SOR}) = \rho(M_{SOR})$ for $0 < \omega < 2$. Furthermore, from (3.15) it follows that $\rho(M_{BPSOR}) = \rho(\hat{M}_{SOR})$. Consequently,

$$\rho(M_{BPSOR}) = \rho(\hat{M}_{SOR}) = \rho(M_{SOR}).$$

Finally, the BPSOR iterate $u^{(k)} = P^T\hat{u}^{(k)} \to P^T\hat{u}^* = u^*$ as $k \to \infty$. This completes the proof. ∎

Two different iterative methods are said to have the same asymptotic rate of convergence if they have the same spectral radius [28]. Hence, (3.19) implies that the BPSOR method has the same asymptotic rate of convergence as the corresponding sequential block SOR method.

3.5. The point PSOR method as an inexact BPSOR scheme. To reduce the computing cost, the block equations of the BPSOR method can be solved approximately by an iterative method. Different iterative solvers with different approximation accuracies lead to different inexact BPSOR methods. In particular, the point PSOR method proposed in [26] can be regarded as an inexact BPSOR method in which each block equation is solved by only one point SOR iteration.

To show the above claim, let $\{\mathcal{U}_\mu^{i,(k)}\}$ be a sequence of the inexact BPSOR iterates in which each block equation is solved by only one point SOR iteration. Then, $\mathcal{U}_\mu^{i,(k+1)}$ is defined by

$$\mathcal{U}_\mu^{i,(k+1)} = \omega(D_\mu^i)^{-1}[f_\mu^i + L_\mu^i \mathcal{U}_\mu^{i,(k+1)} + U_\mu^i \mathcal{U}_\mu^{i,(k)}] + (1-\omega)\mathcal{U}_\mu^{i,(k)}, \tag{3.24}$$

where D_μ^i, L_μ^i, and U_μ^i denote the diagonal, strictly lower-triangular, and strictly upper-triangular matrices satisfying $A_{\mu\mu}^{ii} = D_\mu^i - L_\mu^i - U_\mu^i$, f_μ^i is a vector defined by

$$f_\mu^i = \mathcal{F}_\mu^i - \sum_{\nu=\mu}^{p}\sum_{j=1}^{i-1} A_{\mu\nu}^{ij}\mathcal{U}_\nu^{j,(k+1)} - \sum_{\nu=1}^{\mu}\sum_{j=i+1}^{t} A_{\mu\nu}^{ij}\mathcal{U}_\nu^{j,(k)},$$

$\mu = 1, 2, \ldots, p$, and $i = 1, 2, \ldots, t$. In terms of $A_{\mu\nu}$, $\mathcal{U}_\mu^{(k)}$, L_μ, and U_μ, in (2.9), (2.14), and (2.12), the block vector $f_\mu = ((f_\mu^1)^T, (f_\mu^2)^T, \ldots, (f_\mu^t)^T)^T$ can be expressed as

$$f_\mu = \mathcal{F}_\mu - \sum_{\nu=\mu+1}^{p} A_{\mu\nu}\mathcal{U}_\nu^{(k+1)} - L_\mu\mathcal{U}_\mu^{(k+1)} - \sum_{\nu=1}^{\mu-1} A_{\mu\nu}\mathcal{U}_\nu^{(k)} - U_\mu\mathcal{U}_\mu^{(k)}. \tag{3.25}$$

By using (3.25) and the three block diagonal matrices, $\tilde{D}_\mu = \mathrm{diag}(D_\mu^1, D_\mu^2, \ldots, D_\mu^t)$, $\tilde{B}_\mu = \mathrm{diag}(L_\mu^1, L_\mu^2, \ldots, L_\mu^t)$, and $\tilde{C}_\mu = \mathrm{diag}(U_\mu^1, U_\mu^2, \ldots, U_\mu^t)$, the t expressions of $\{\mathcal{U}_\mu^{i,(k+1)}\}_{i=1}^t$ in (3.24) can be formulated as one vector form:

$$\begin{aligned}\mathcal{U}_\mu^{(k+1)} = \omega(D_\mu)^{-1}\Bigg[&\mathcal{F}_\mu + (\tilde{B}_\mu + L_\mu)\mathcal{U}_\mu^{(k+1)} + (\tilde{C}_\mu + U_\mu)\mathcal{U}_\mu^{i,(k)} \\ &- \sum_{\nu=\mu+1}^{p} A_{\mu\nu}\mathcal{U}_\nu^{(k+1)} - \sum_{\nu=1}^{\mu-1} A_{\mu\nu}\mathcal{U}_\nu^{(k)}\Bigg] + (1-\omega)\mathcal{U}_\mu^{(k)}.\end{aligned} \tag{3.26}$$

Further, define three block diagonal matrices as

$$\tilde{D} = \mathrm{diag}(\tilde{D}_1, \tilde{D}_2, \ldots, \tilde{D}_p),\ \tilde{B} = \mathrm{diag}(\tilde{L}_1, \tilde{L}_2, \ldots, \tilde{L}_p),\ \tilde{C} = \mathrm{diag}(\tilde{U}_1, \tilde{U}_2, \ldots, \tilde{U}_p),$$

where $\tilde{L}_\mu = \tilde{B}_\mu + L_\mu$ and $\tilde{U}_\mu = \tilde{C}_\mu + U_\mu$, which are the strictly lower- and upper-triangular matrices satisfying $A_{\mu\mu} = \tilde{D}_\mu - \tilde{L}_\mu - \tilde{U}_\mu$. By using them, the p expressions in (3.26) can be written as the matrix form

$$u^{(k+1)} = \omega\tilde{D}^{-1}[f + (\tilde{B} + M)u^{(k+1)} + (\tilde{C} + N)u^{(k)}] + (1-\omega)u^{(k)},$$

where the matrices M and N are given in (3.4). Solving the above equation for $u^{(k+1)}$ yields the matrix iterative expression of the inexact BPSOR method:

$$\begin{aligned}u^{(k+1)} = &[\tilde{D} - \omega(\tilde{B} + M)]^{-1}[(1-\omega)\tilde{D} + \omega(\tilde{C} + N)]u^{(k)} \\ &+ \omega[\tilde{D} - \omega(\tilde{B} + M)]^{-1}f.\end{aligned} \tag{3.27}$$

On the other hand, from [26] it is known that the point PSOR method has the same matrix expression as the one above. Hence, the inexact BPSOR method is exactly the point PSOR method defined in [26].

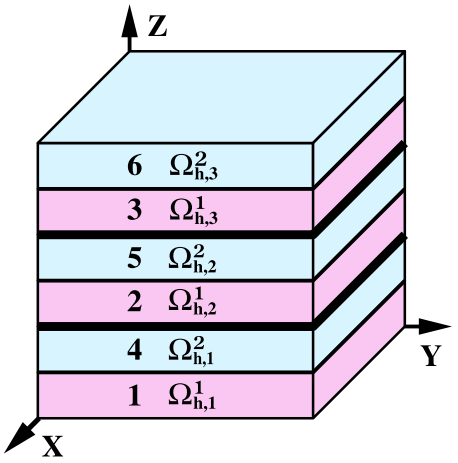

FIG. 4.1. *The 2-type partition based on a 3-strip partitioning of a cubic domain ($p = 3$ and $t = 2$). Each strip contains two types, $\Omega^1_{h,\mu}$ and $\Omega^2_{h,\mu}$ for $\mu = 1, 2, 3$, which are numbered in the 2-type ordering.*

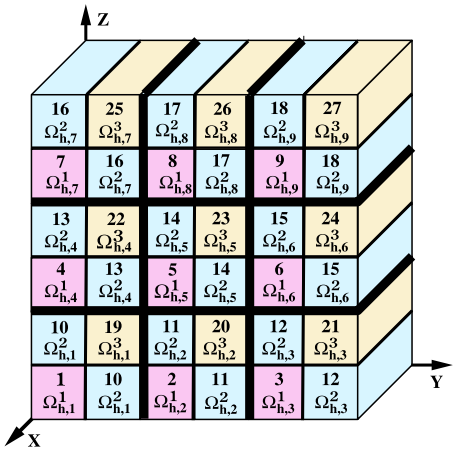

FIG. 4.2. *The 3-type partition based on a 2D 9-block partitioning of a cubic domain ($p = 9$ and $t = 3$). Each block contains three types, $\Omega^1_{h,\mu}$, $\Omega^2_{h,\mu}$, and $\Omega^3_{h,\mu}$ for $\mu = 1, 2, \ldots, 9$, which are numbered in the 3-type ordering.*

4. Three BPSOR methods based on strip and block partitionings.

4.1. The 2-type BPSOR method by strip partitioning. The 2-type BPSOR method is the BPSOR method by the 2-type partition based on a strip partition of the mesh domain Ω_h. For finite element and finite difference equations, the mesh-domain Ω_h is partitioned into p strips along one direction of the space such that each interior strip has only two coupled neighboring strips. Thus, each strip $\Omega_{h,\mu}$ contains only two types, $\Omega^1_{h,\mu}$ and $\Omega^2_{h,\mu}$. An illustration of the 2-type partition and ordering is given in Figure 4.1. Here a cubic domain is partitioned into three strips ($p = 3$), the two types are colored in two different colors, and each type is numbered in the 2-type ordering.

In the case of the 2-type partition ($t = 2$), the block matrix A in (2.8) becomes a tridiagonal block matrix with $A_{ij} = 0$ for $|i - j| \geq 2$. The nonzero off-diagonal submatrices of each row have the following forms:

$$A_{\mu,\mu+1} = \begin{bmatrix} 0 & 0 \\ A^{21}_{\mu,\mu+1} & 0 \end{bmatrix} \quad \text{and} \quad A_{\mu+1,\mu} = \begin{bmatrix} 0 & A^{12}_{\mu+1,\mu} \\ 0 & 0 \end{bmatrix}. \tag{4.1}$$

By setting $A^{12}_{1,0} = A^{21}_{p,p+1} = 0$, the BPSOR method by the strip partitioning can be expressed as

$$\begin{aligned} \mathcal{U}^{1,(k+1)}_\mu &= \omega(A^{11}_{\mu\mu})^{-1}[\mathcal{F}^1_\mu - A^{12}_{\mu,\mu-1}\mathcal{U}^{2,(k)}_{\mu-1} - A^{12}_{\mu,\mu}\mathcal{U}^{2,(k)}_\mu] \\ &\quad + (1-\omega)\mathcal{U}^{1,(k)}_\mu, \quad \mu = 1, 2, \ldots, p \quad \text{for Type 1,} \end{aligned} \tag{4.2}$$

$$\begin{aligned} \mathcal{U}^{2,(k+1)}_\mu &= \omega(A^{22}_{\mu\mu})^{-1}[\mathcal{F}^2_\mu - A^{21}_{\mu,\mu}\mathcal{U}^{1,(k+1)}_\mu - A^{21}_{\mu,\mu+1}\mathcal{U}^{1,(k+1)}_{\mu+1}] \\ &\quad + (1-\omega)\mathcal{U}^{2,(k)}_\mu, \quad \mu = 1, 2, \ldots, p \quad \text{for Type 2.} \end{aligned} \tag{4.3}$$

Obviously, the p iterative expressions of (4.2) are completely independent; so are the p iterative expressions of (4.3). Hence, the 2-type BPSOR method is a parallel iterative scheme by the strip partitioning.

In the 2-type ordering, the reordered matrix $\hat{A}$ of (3.10) becomes a 2×2 block matrix in the form

$$\hat{A} = \begin{bmatrix} \hat{A}_{11} & \hat{A}_{12} \\ \hat{A}_{21} & \hat{A}_{22} \end{bmatrix},$$

where $\hat{A}_{11} = \text{diag}(A_{11}^{11}, A_{22}^{11}, \ldots, A_{pp}^{11})$, $\hat{A}_{22} = \text{diag}(A_{11}^{22}, A_{22}^{22}, \ldots, A_{pp}^{22})$,

$$\hat{A}_{12} = \begin{bmatrix} A_{11}^{12} & & & \\ A_{21}^{12} & A_{22}^{12} & & \\ & \ddots & \ddots & \\ & & A_{p,p-1}^{12} & A_{pp}^{21} \end{bmatrix}, \text{ and } \hat{A}_{21} = \begin{bmatrix} A_{11}^{21} & A_{12}^{21} & & \\ & A_{22}^{21} & \ddots & \\ & & \ddots & A_{p-1,p}^{21} \\ & & & A_{pp}^{21} \end{bmatrix}.$$

Obviously, the above $\hat{A}$ is consistently ordered since it is a form of block red-black matrix [11]. Hence, the 2-type BPSOR method has the same convergence rate as the corresponding block SOR method.

4.2. The 3-type BPSOR method by 2D block partitioning. The 3-type BPSOR method is the BPSOR method by the 3-type partition based on the 2D block partitioning. For a linear system that arises from a 5-point-like or 7-point-like finite difference approximation to a 2D or 3D elliptic boundary problem, the mesh domain Ω_h is partitioned into p blocks $\Omega_{h,\mu}$ along two dimensions of the space such that each interior block is only connected to the four neighboring blocks from the left, right, top, and bottom sides. Each block is then divided into three types $\Omega_{h,\mu}^1, \Omega_{h,\mu}^2$, and $\Omega_{h,\mu}^3$ for $\mu = 1, 2, \ldots, p$. An illustration of the 3-type partition is given in Figure 4.2. Here a cubic domain is partitioned into nine blocks along the y-axis and z-axis directions, and the types labeled as $\Omega_{h,\mu}^i$ for $\mu = 1, 2, \ldots, 9$ and $i = 1, 2, 3$ are numbered in the 3-type ordering. Note that each second type $\Omega_{h,\mu}^2$ consists of two small blocks since the iterates defined on these two small blocks can be calculated simultaneously without affecting the definition of BPSOR.

In the case of the 3-type partition, the block matrix A given in (2.8) becomes a 5-diagonal block matrix such that the BPSOR method defined in (2.17) can be simplified as

$$\begin{aligned}
\mathcal{U}_\mu^{1,(k+1)} &= \omega(A_{\mu\mu}^{11})^{-1}[\mathcal{F}_\mu^1 - A_{\mu\mu}^{12}\mathcal{U}_\mu^{2,(k)} - A_{\mu,\mu-1}^{12}\mathcal{U}_{\mu-1}^{2,(k)} - A_{\mu,\mu-m}^{12}\mathcal{U}_{\mu-m}^{2,(k)}] \\
&\quad + (1-\omega)\mathcal{U}_\mu^{1,(k)}, \quad \mu = 1, 2, \ldots, p, \quad \text{for Type 1,} \\
\mathcal{U}_\mu^{2,(k+1)} &= \omega(A_{\mu\mu}^{22})^{-1}[\mathcal{F}_\mu^2 - A_{\mu\mu}^{21}\mathcal{U}_\mu^{1,(k+1)} - A_{\mu\mu}^{23}\mathcal{U}_\mu^{3,(k)} - A_{\mu,\mu-1}^{23}\mathcal{U}_{\mu-1}^{3,(k)} \\
&\quad - A_{\mu,\mu+1}^{21}\mathcal{U}_{\mu+1}^{1,(k+1)} - A_{\mu,\mu+m}^{21}\mathcal{U}_{\mu+m}^{1,(k+1)} - A_{\mu,\mu-m}^{23}\mathcal{U}_{\mu-m}^{3,(k)}] \\
&\quad + (1-\omega)\mathcal{U}_\mu^{2,(k)}, \quad \mu = 1, 2, \ldots, p, \quad \text{for Type 2,} \\
\mathcal{U}_\mu^{3,(k+1)} &= \omega(A_{\mu\mu}^{33})^{-1}[\mathcal{F}_\mu^3 - A_{\mu\mu}^{32}\mathcal{U}_\mu^{2,(k+1)} - A_{\mu,\mu+1}^{32}\mathcal{U}_{\mu+1}^{2,(k+1)} - A_{\mu,\mu+m}^{32}\mathcal{U}_{\mu+m}^{2,(k+1)}] \\
&\quad + (1-\omega)\mathcal{U}_\mu^{3,(k)}, \quad \mu = 1, 2, \ldots, p, \quad \text{for Type 3,}
\end{aligned} \tag{4.4}$$

where $A_{\mu j}^{12} = A_{\mu j}^{23} = 0$ for $j \leq 0$ and $A_{\mu k}^{21} = A_{\mu k}^{32} = 0$ for $k \geq p$. Obviously, the p expressions of each type in (4.4) are completely independent. Hence, the 3-type BPSOR method is a parallel scheme by the 2D block partitioning.

Under the 3-type ordering, the reordered matrix $\hat{A}$ of (3.10) becomes a 3×3 block matrix in the form

$$\hat{A} = \begin{bmatrix} \hat{A}_{11} & \hat{A}_{12} & 0 \\ \hat{A}_{21} & \hat{A}_{22} & \hat{A}_{23} \\ 0 & \hat{A}_{32} & \hat{A}_{33} \end{bmatrix}, \tag{4.5}$$

where $\hat{A}_{ii} = \mathrm{diag}(A^{ii}_{11}, A^{ii}_{22}, \ldots, A^{ii}_{pp})$ for $i = 1, 2, 3$, $\hat{A}_{12}$, $\hat{A}_{23}, \hat{A}_{21}$, and $\hat{A}_{32}$ have nonzero entries only on the main diagonal, the second off-diagonal, and the mth off-diagonal. That is, $\hat{A}_{12}$ and $\hat{A}_{23}$ have the forms

$$\hat{A}_{12} = \begin{bmatrix} A^{12}_{11} & & & & & \\ A^{12}_{21} & A^{12}_{22} & & & & \\ A^{12}_{m+1,1} & \ddots & \ddots & & & \\ & \ddots & A^{12}_{p-1,p-2} & A^{12}_{p-1,p-1} & & \\ & & A^{12}_{p,p-m} & A^{12}_{p,p-1} & A^{12}_{pp} \end{bmatrix},$$

$$\hat{A}_{23} = \begin{bmatrix} A^{23}_{11} & & & & & \\ A^{23}_{21} & A^{23}_{22} & & & & \\ A^{23}_{m+1,1} & \ddots & \ddots & & & \\ & \ddots & A^{23}_{p-1,p-2} & A^{23}_{p-1,p-1} & & \\ & & A^{23}_{p,p-m} & A^{23}_{p,p-1} & A^{23}_{pp} \end{bmatrix};$$

$\hat{A}_{21} = \hat{A}^T_{12}$ and $\hat{A}_{32} = \hat{A}^T_{23}$ if A is symmetric.

By the definition of the consistently ordered matrix as the one given in Definition 9-2.2 of [11], it can be shown that the reordered matrix $\hat{A}$ in (4.5) is consistently ordered. In fact, the index set $S = \{1, 2, \ldots, 3p\}$ since $\hat{A}$ for a $3p\times 3p$ block matrix of $\hat{A}$ has the partition $S = S_1 \cup S_2 \cup S_3$, where $S_1 = \{1, 2, \ldots, p\}$, $S_2 = \{p+1, p+2, \ldots, 2p\}$, and $S_3 = \{2p + 1, 2p + 2, \ldots, 3p\}$, which satisfies the definition of the consistently ordered matrix [11, 28]. That is, if the (i, j)th entry of $\hat{A}$ is nonzero and $i \in S_k$, then $j \in S_{k+1}$ for $j > i$ and $j \in S_{k-1}$ if $j < i$.

Since $\hat{A}$ is consistently ordered, from the classic SOR theory it follows that the 3-type BPSOR method has the same convergence rate as the corresponding block SOR method.

4.3. The 6-type BPSOR method by the 3D block partitioning. The 6-type BPSOR method is the BPSOR method by the 6-type partition based on the 3D block partitioning. For the linear system arising from a 7-point-like finite difference approximation to a 3D elliptic boundary problem, a 3D mesh domain is partitioned into p blocks along the three dimensions of the space such that each block is only connected to its six neighboring blocks that come from the left, right, bottom, top, back, and front sides. Each block is then divided into six types Ω^i_μ with $i = 1, 2, \ldots, 6$ and $\mu = 1, 2, \ldots, p$ as illustrated in Figure 4.3. An example of the 6-type partition is given in Figure 4.4. Here a cubic domain is partitioned into eight blocks along the x-, y-, and z-axis directions. To view the labels and ordering numbers of each type, this 6-type partition is cut into four portions along the x-axis direction so that the 48 types $\Omega^i_{h,\mu}$ with $\mu = 1, 2, \ldots, 8$ and $i = 1, 2, \ldots, 6$ of this 6-type partition can be displayed in the four plots of Figure 4.5.

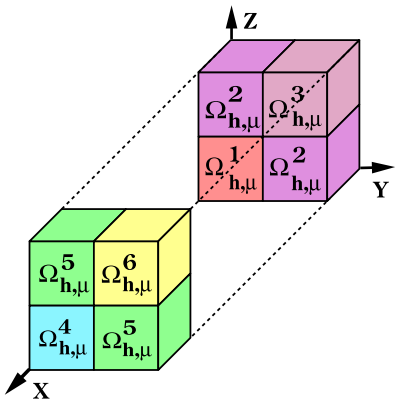

FIG. 4.3. *The six types of each block for the* 6*-type partition in Figure* 4.4*. The six types are labeled as* $\Omega^i_{h,\mu}$ *for* $i = 1, 2, \ldots, 6$*, and the front three types have been pushed away from the back three types to display the labels.*

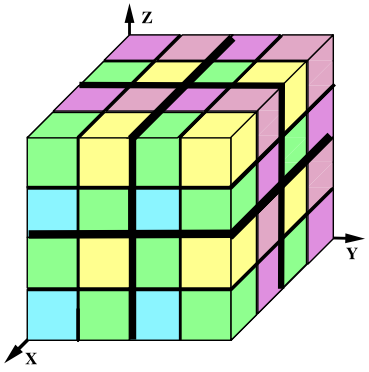

FIG. 4.4. *The* 6*-type partition based on a* $2 \times 2 \times 2$ *block partitioning of a cubic domain. The six types of each block are illustrated in Figure* 4.3*. See Figure* 4.5 *for the* 6*-type ordering.*

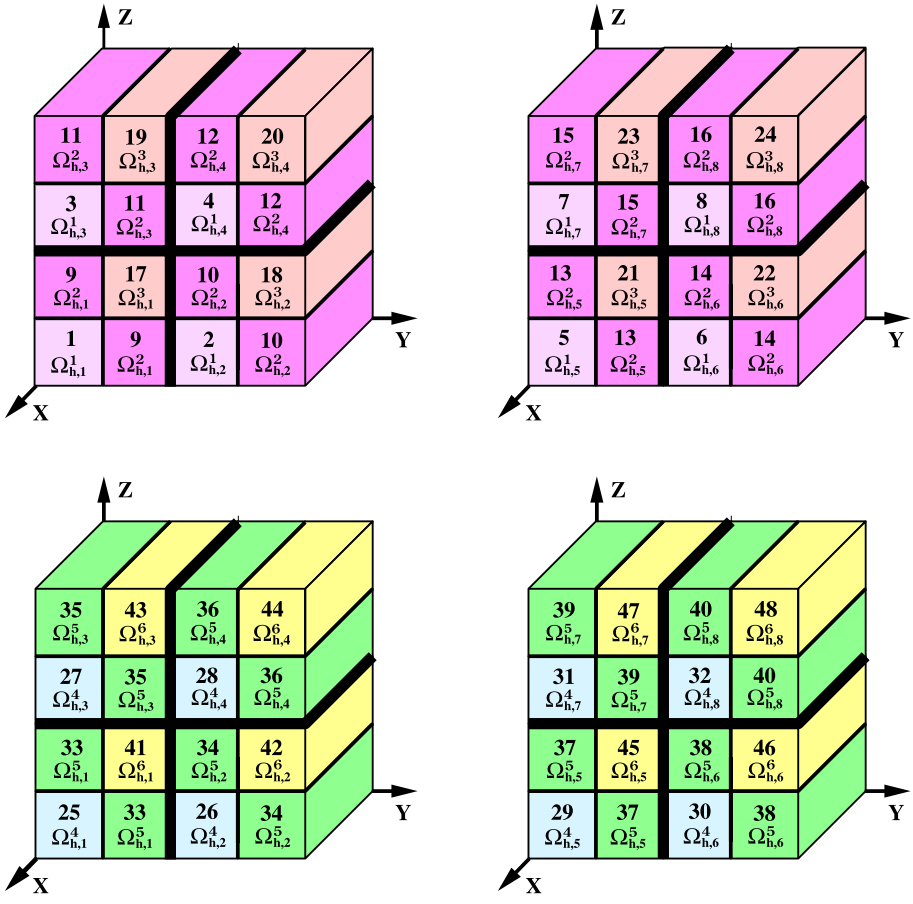

FIG. 4.5. *The* 6*-type ordering and the* 48 *types* $\Omega^i_{h,\mu}$ *with* $\mu = 1, 2, \ldots, 8$ *and* $i = 1, 2, \ldots, 6$ *of the* 6*-type partition in Figure* 4.4*. Here the* 6*-type partition consists of four type layers, which are drawn separately in the four plots to display the label and ordering number of each type.*

In the case of the 6-type partition, the block matrix A in (2.8) is reduced to a seven diagonal block matrix so that the iterative expression of the 6-type BPSOR method can be obtained directly from the general expression (2.17). In the 6-type ordering, the reordered matrix $\hat{A}$ in (3.10) is consistently ordered. In fact, since $\hat{A}$ is a $6p \times 6p$ block matrix, the index set $S = \{1, 2, \ldots, 6p\}$ is considered. It can be found that S has the following partition:

$$S = S_1 \cup S_2 \cup S_3 \cup S_4,$$

where $S_1 = \{1, 2, \ldots, p\}, S_2 = \{p+1, p+2, \ldots, 2p, 3p+1, 3p+2, \ldots, 4p\}, S_3 = \{2p+1, 2p+2, \ldots, 3p, 4p+1, 4p+2, \ldots, 5p\}$, and $S_4 = \{5p+1, 5p+2, \ldots, 6p\}$. It then can be shown that the partition of S satisfies the definition of a consistently ordered matrix. Since $\hat{A}$ is a consistently ordered matrix, the classic SOR theory follows that the 6-type BPSOR method has the same convergence rate as the corresponding sequential block SOR method.

5. Numerical results. Numerical experiments were made on the SGI Origin 2000 at the University of Wisconsin-Milwaukee, which contains 16 R12000 processors. A linear system for numerical tests comes from a 7-point finite difference approximation to a 3D Poisson boundary value problem on the unit cubic domain with a boundary value of zero. The mesh domain Ω_h consists of the interior mesh points (x_i, y_j, z_k) with $x_i = (i-1)h$, $y_j = (j-1)h$, and $z_k = (k-1)h$ for $i, j, k = 2, 3, \ldots, n$. Here $h = 1/n$. By setting $u_{i,j,k}$ as the numerical approximation to the Poisson solution at mesh point (x_i, y_j, z_k), the linear system is defined as

$$\begin{aligned} 6u_{i,j,k} - u_{i+1,j,k} - u_{i-1,j,k} - u_{i,j+1,k} - u_{i,j-1,k} \\ - u_{i,j,k-1} - u_{i,j,k+1} = h^2 f_{i,j,k}, \end{aligned} \tag{5.1}$$

where $i, j, k = 2, 3, \ldots, n$, $f_{i,j,k}$ denotes the value of the right-hand side function at mesh point (x_i, y_j, z_k), and the boundary values are $u_{i1} = u_{i,n+1} = u_{1i} = u_{n+1,i} = 0$ for $1 \le i \le n+1$. In numerical tests, $f_{i,j,k}$ was set as one.

The parallel program was written in FORTRAN 77 and MPI (the Message Passing Interface) [3]. The program was compiled by using the optimization level $O2$. The CPU time was measured by the MPI function MPI_Wtime, which returns the wall time in seconds. The sequential SOR program in FORTRAN 77 was implemented on one processor of the computer. Its CPU time was measured by the F77 function $etime()$. The iteration termination rule is given by

$$\|f - Au^{(k)}\| < 10^{-6}, \tag{5.2}$$

where $u^{(k)}$ denotes the kth iterate for solving the linear system $Au = f$.

5.1. Solution of each block linear equation. Each BPSOR iteration requires us to solve tp block equations, $A^{ii}_{\mu\mu} v^i_\mu = f^i_\mu$ for $i = 1, 2, \ldots, t$ and $\mu = 1, 2, \ldots, p$. Since each block equation is located within one processor, it can be solved accurately by a sequential direct method or approximately by a sequential iterative method. In this numerical test, the sequential point SOR method, and the sparse LDL^T method from the SGI's Scientific Computing Software Library were used to solve block equations. Here a point SOR iterate $v^{i,(k)}_\mu$ was accepted as an approximate solution of block equation $A^{ii}_{\mu\mu} v^i_\mu = f^i_\mu$ when it met the termination rule: $\|f^i_\mu - A^{ii}_{\mu\mu} v^{i,(k)}_\mu\| \le 10^{-8}$ for all $i = 1, 2, \ldots, t$, $\mu = 1, 2, \ldots, p$. For simplicity, the point SOR method

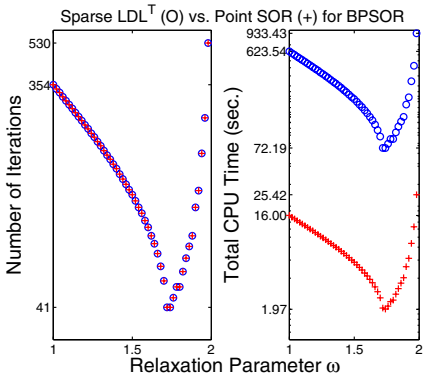

FIG. 5.1. *Comparisons of the convergence and performance of the BPSOR method using the LDL^T direct solver for each block equation with that using the sequential point SOR iterative solver for each block equation. The tests were made on eight processors for the model problem* (5.1) *with* $h = 1/65$.

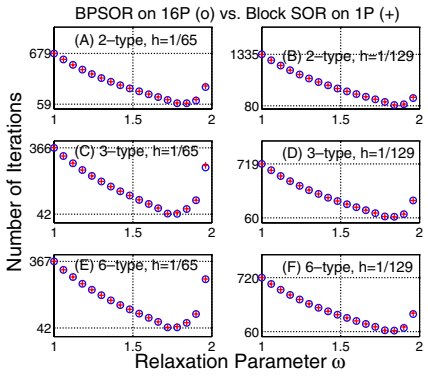

FIG. 5.2. *Comparisons of the convergence rates of the three particular BPSOR methods defined by the* 2*-type,* 3*-type, and* 6*-type partitions on 16 processors with that of the three corresponding sequential block SOR methods on one processor for solving the model problem* (5.1) *with* $h = 1/65$ *and* $h = 1/129$.

used $\omega = 1.54$ (which may not be optimal) for solving all the block equations. The BPSOR method was carried out on eight processors based on an 8-strip partition along the z-axis direction. The numerical results were plotted in Figure 5.1. They show that with the point SOR method, the BPSOR method can have almost the same number of iterations determined by (5.2) as that with the sparse LDL^T direct solver, while can take much less CPU time (about 40 times less in this test).

5.2. Convergence comparisons. Figure 5.2 compares the total numbers of iterations of the three particular BPSOR methods defined by the 2-type, 3-type, and 6-type partitions with that of the three corresponding sequential block SOR methods for solving the model problem (5.1) with $h = 1/65$ and $h = 1/129$, respectively. The BPSOR methods were implemented on the sixteen processors of the SGI Origin 2000. The total number of iterations was determined by the termination rule (5.2). The 2-type, 3-type, and 6-type partitions were constructed based on the strip partitioning of order $1 \times 1 \times 16$, the 2D-block partitioning of $1 \times 4 \times 4$, and the 3D-block partitioning of $2 \times 2 \times 4$, respectively. The size of each block equation was selected to be almost the same. Each block equation was solved approximately by the sequential point SOR method with $\omega = 1.54$ for the 2-type case and $\omega = 1.7$ for the 3-type and 6-type cases. From Figure 5.2 it can be seen that all the three BPSOR methods have almost the same numbers of iterations as the three corresponding sequential block SOR methods. This confirms the conclusion of Theorem 3.4; i.e., the BPSOR method has the same asymptotic rate of convergence as the corresponding sequential block SOR method.

5.3. Parallel performances of BPSOR. Figure 5.3 displays the CPU time data of the above BPSOR and block SOR methods and the speedups of the three BPSOR methods as functions of relaxation parameter ω between 1 and 1.98. The interprocessor data communication time spent by BPSOR were also reported in the figure as dashed lines. Here for each value the speedup was calculated as the ratio of the total CPU time of the sequential block SOR method on one processor to that of the BPSOR method on sixteen processors. From Figure 5.3 it can be seen that all the

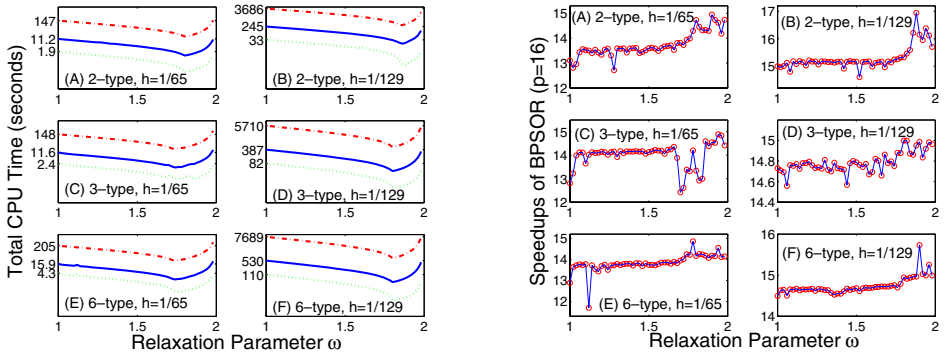

FIG. 5.3. *Performance comparisons between the three particular BPSOR methods* (the solid lines) *defined by the* 2*-type,* 3*-type, and* 6*-type partitions on* 16 *processors and the corresponding three sequential block SOR methods* (the dash-dotted lines) *on one processor for solving the model problem* (5.1) *with* $h = 1/65$ *and* $h = 1/129$. The dotted lines *indicate the interprocessor communication times. The speedups of the BPSOR methods are also presented.*

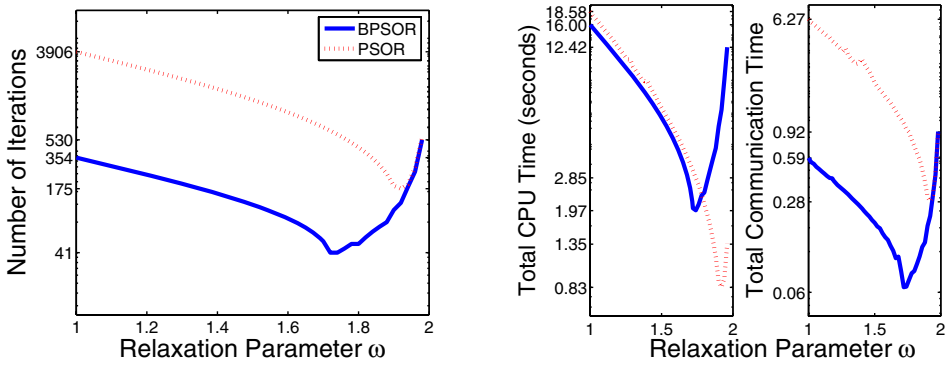

FIG. 5.4. *Performance comparisons of BPSOR with PSOR on eight processors of the SGI Origin 2000 based on an* 8*-strips partition of the cubic mesh domain with* $h = 1/65$.

three BPSOR methods have the speedups between 12 to 16 on the sixteen processors of the SGI Origin 2000, which indicate that the BPSOR methods have good parallel performances.

5.4. Parallel performance comparisons with PSOR. The parallel performance of the BPSOR method was compared with that of the point PSOR method in Figure 5.4. Both BPSOR and PSOR were defined on the same 8-strip partition along the z-axis, and implemented on eight processors of the SGI Origin 2000. The first plot of Figure 5.4 shows that BPSOR took about 10 to 16 times fewer iterations than PSOR from $\omega = 1$ to $\omega = 1.76$ to satisfy the termination rule (5.2), confirming that BPSOR has a much faster rate of convergence than the point PSOR method. As a result, as displayed in the third plot of Figure 5.4, the BPSOR method took about 10 to 16 times less CPU time in interprocessor data communications than the point PSOR method. The second plot of Figure 5.4 also shows that BPSOR took up to 53 percent less total CPU time than PSOR for the values of ω from 1 to 1.76. Furthermore, the first plot shows that the convergence rate of BPSOR is less sensitive

to the value selection of ω than that of the point PSOR method. Obviously, the total performance of BPSOR can be further improved significantly when a more efficient sequential iterative method (such as the multigrid method [22]) is used to solve each block equation.

Acknowledgments. The author thanks Professor Loyce Adams and the anonymous referees for their valuable comments.

REFERENCES

[1] L. M. Adams and H. F. Jordan, *Is SOR color-blind?*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 490–506.
[2] L. M. Adams and J. M. Ortega, *A multi-color SOR method for parallel computation*, Proceedings of the 1982 International Conference on Parallel Processing, Bellaire, MI, 1982, pp. 53–58.
[3] *MPI: A Message-Passing Interface Standard*, University of Tennessee, Knoxville, Tennessee, June 1995.
[4] C. Ashcraft and R. Grimes, *On vectorizing incomplete factorization and SSOR preconditioners*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 122–151.
[5] B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan, and M. Karplus, *CHARMM: A program for macromolecular energy, minimization, and dynamics calculations*, J. Comp. Chem., 4 (1983), pp. 187–217.
[6] C. J. Cramer and D. G. Truhlar, *Implicit solvation models: Equilibria, structure, spectra, and dynamics*, Chem. Rev., 99 (1999), pp. 2161–2200.
[7] M. E. Davis, J. D. Madura, B. A. Luty, and J. A. McCammon, *Electrostatics and diffusion of molecules in solution: Simulations with the University of Houston, Brownian dynamics program*, Comput. Phys. Comm., 62 (1991), pp. 187–197.
[8] I. S. Duff and G. A. Meurant, *The effect of orderings on preconditioned conjugate gradients*, BIT, 29 (1989), pp. 635–647.
[9] C. Farhat, *Multiprocessors in Computational Mechanics*, Ph.D. thesis, Civil Engineering Department, University of California at Berkeley, Berkeley, CA, 1986.
[10] W. Hackbusch, *Iterative Solution of Large Sparse Systems of Equations*, Springer-Verlag, New York, 1994.
[11] L. A. Hageman and D. M. Young, *Applied Iterative Methods*, Academic Press, San Diego, 1981.
[12] D. Harrar and J. M. Ortega, *Multicoloring with lots of colors*, in Proceedings of the Third International Conference on Supercomputing, Crete, Greece, 1989, pp. 1–6.
[13] B. Honig and A. Nicholls, *Classical electrostatics in biology and chemistry*, Science, 268 (1995), pp. 1144–1149.
[14] T. Lippert, *Parallel SSOR preconditioner for lattice QCD*, Parallel Comput., 25 (1999), pp. 1357–1370.
[15] R. Melhem, *Towards efficient implementations of preconditioned conjugate methods on vector supercomputers*, Internat. J. Supercomput. Appl., 1 (1987), pp. 70–98.
[16] J. M. Ortega, *Orderings for conjugate gradient preconditionings*, SIAM J. Optim., 1 (1991), pp. 565–582.
[17] E. L. Poole and J. M. Ortega, *Multicolor ICCG methods for vector computers*, SIAM J. Numer. Anal., 24 (1987), pp. 1394–1418.
[18] S. A. Stotland and J. M. Ortega, *Orderings for parallel conjugate gradient preconditioners*, SIAM J. Sci. Comput., 18 (1997), pp. 854–868.
[19] W. Rocchia, E. Alexov, and B. Honig, *Extending the applicability of the nonlinear Poisson-Boltzmann equation: Multiple dielectric constants and multivalent ions*, J. Phys. Chem. B, 105 (2001), pp. 6507–6514.
[20] B. Roux and T. Simonson, *Implicit solvent models*, Biophys. Chem., 78 (1999), pp. 1–20.
[21] B. F. Smith, P. E. Bjørstad, and W. D. Gropp, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, Cambridge, UK, 1996.
[22] K. Stüen and U. Trottenberg, *Multigrid methods: Fundamental algorithms, model problem analysis and applications*, in Multigrid Methods, Lecture Notes in Math. 960, Springer-Verlag, Berlin, New York, 1982, pp. 1–176.
[23] E. F. van de Velde, *Concurrent Scientific Computing*, Springer-Verlag, New York, 1994.

[24] H. A. VAN DER VORST *Large tridiagonal and block tridiagonal linear systems on vector and parallel computers*, Parallel Comput., 5 (1987), pp. 45–54.
[25] J. WANG, *Convergence analysis without regularity assumptions for multigrid algorithms based on SOR smoothing*, SIAM J. Numer. Anal., 29 (1992), pp. 987–1001.
[26] D. XIE AND L. ADAMS, *New parallel SOR method by domain partitioning*, SIAM J. Sci. Comput., 20 (1999), pp. 2261–2281.
[27] I. YAVNEH, *On red-black SOR smoothing in multigrid*, SIAM J. Sci. Comput., 17 (1996), pp. 180–192.
[28] D. M. YOUNG, *Iterative Solution of Large Linear Systems*, Academic Press, New York, 1971.