International Journal of Computer Mathematics
CMMSE SI 2007

New Parallel Symmetric SOR Preconditioners by Multi-Type Partitioning

Dexuan Xie[1]

[1] *Department of Mathematical Sciences,, University of Wisconsin, Milwaukee, WI 53201, USA*

emails: dxie@uwm.edu

Abstract

A new parallel symmetric successive over-relaxation (PSSOR) preconditioner is proposed in this paper by the multi-type partition techniques introduced in SIAM J. Scientific Computing 20, 2006, pp. 1513-1533. In a general matrix expression, it is proved to be symmetric and positive-definite if the coefficient matrix of a linear system is symmetric and positive-definite. It is also proved to be equivalent to the SSOR preconditioner using the multi-type ordering. Thus, it works for the preconditioned conjugate gradient method (PCG) and can be analyzed by the classic SOR theory. Numerical tests on an anisotropic model problem show that the PSSOR preconditioner can make PCG to have a faster rate of convergence and better parallel performances than the red-black SSOR preconditioner. They also confirm that the PSSOR preconditioner can have a rate of convergence that is nearly the same as the classic sequential SSOR preconditioner when the problem has large anisotropy.

Key words: preconditioned conjugate gradient, SSOR preconditioner, domain decomposition, multi-type partition, parallel computing
MSC 2000: AMS 65Y05 (65F10)

1 Introduction

The symmetric successive over-relaxation (SSOR) preconditioner is a widely used classic preconditioner for solving a large scale sparse linear system that arises from a finite element or finite difference approximation to an elliptic boundary value problem [5, 7, 14]. It is well known that the effects of the SSOR preconditioner on the convergence rate of the preconditioned conjugate gradient method (PCG) depend on the ordering of the mesh points on which the linear system is defined. The natural ordering leads to a sequential SSOR preconditioner, with which PCG has the fastest convergence rate, but is difficult to be implemented on a parallel computer. The red-black (or multicolor) ordering [14] can overcome this difficulty, but may seriously reduce the convergence rate of PCG. Several other ordering techniques, which include the wavefront ordering

[1], the local column-wise ordering [8], the many-color ordering [2, 6], and several domain decomposition orderings [3, 9, 10], have been proposed, yielding various parallel SSOR preconditioners. Most of these parallel SSOR preconditioners were shown to be effective depending on computational environments and problems to be solved. Some of them were found to be more efficient than the red-black SSOR preconditioner in both convergence and parallel performance [9]. On current MIMD parallel computers, domain decomposition ordering techniques are well known to have superior interprocessor data communications. Thus, they are often used to develop parallel algorithms. On the other hand, they may reduce the convergence rate of a sequential method to be parallelized when the number of subdomains increases.

To overcome low convergence rates, a new domain decomposition technique called the multi-type partition (or ordering) was introduced in [13]. By the multi-type partition, a new block parallel SOR (BPSOR) method was proposed in [13], along with three particular multi-type BPSOR methods for solving the five-point like linear systems (in 2D) and the seven-point like linear systems (in 3D). All of them were proved to 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". In the point form, BPSOR is reduced to the point parallel SOR method proposed in [12]. Due to the success of the multi-type partition in developing parallel SOR methods, it is interesting to study the application of the multi-type partition in developing new parallel SSOR preconditioners. The purpose of this paper is to define a new parallel SSOR (PSSOR) preconditioner using the multi-type partition, and study its effects on the rate of convergence of PCG.

In particular, this paper obtains a general matrix expression of the PSSOR preconditioner. It then shows that the PSSOR preconditioner is symmetric and positive-definite (SPD) if the coefficient matrix of the linear system is SPD. It also proves that the PSSOR preconditioner is equivalent to the SSOR preconditioner using multi-type ordering. Furthermore, it shows that the PSSOR preconditioning equation can be solved simply by one symmetric BPSOR iteration with an initial guess of zero. Here the symmetric BPSOR method will be defined in this paper. These theoretical results guarantee that the PSSOR preconditioner can be applied to PCG as easily as the classic SSOR preconditioner, and can be analyzed in the framework of the classic SOR theory [14].

To study the effects of the PSSOR preconditioner on the convergence rate of PCG, two particular PSSOR preconditioners (the 2-type and 3-type PSSOR preconditioners), the red-black SSOR preconditioner, and the sequential SSOR preconditioner are constructed for solving a linear system that arises from a five-point finite difference approximation to an anisotropic model problem. Through computing and comparing their condition numbers, it is found that the two PSSOR preconditioners have much smaller condition numbers than the red-black SSOR preconditioner. For the problem with large anisotropy, the condition number of the 2-type PSSOR preconditioner is found to be the same as that of the sequential SSOR preconditioner. Thus, the PSSOR preconditioner is expected to make PCG converge faster than the red-black SSOR preconditioner, and a rate of convergence that is comparable to the classic sequential SSOR

preconditioner.

To study the parallel performance of the PSSOR preconditioner, a parallel PCG program package is developed in FORTRAN 77 and MPI, which includes the 2-type and 3-type PSSOR preconditioners and the red-black SSOR preconditioner. It is tested for solving an anisotropic model problem on a MIMD parallel computer (the SGI Origin 2000 with 16 processors at the University of Wisconsin-Milwaukee). Numerical results show that with the PSSOR preconditioner, PCG has a much faster convergence rate and much better parallel performance (in terms of the total number of iterations and computer CPU time) than with the red-black SSOR preconditioner. They also show that for the problem of large anisotropy, the PCG using the 2-type PSSOR preconditioner has nearly the same rate of convergence as the PCG using the sequential SSOR preconditioner. In other words, the convergence rate of the parallel PCG using the PSSOR preconditioner can be independent of the number of processors.

The remainder of the paper is organized as follows: Section 2 reviews the multi-type partition and the related block linear system. Section 3 defines and analyzes the PSSOR preconditioner. Section 4 constructs the 2-type and 3-type PSSOR preconditioners for solving the model problem, and compares their condition numbers with that of the red-black and classic SSOR preconditioners. Section 5 studies the parallel performance of PSSOR preconditioners. Finally, conclusions are made in Section 6.

2 The multi-type partition and the related block linear system

This section gives the multi-type partition and the related block linear system a short review. For more details, see [13]. Here the linear system, $Au = f$, is supposed to arise from a finite element or finite difference approximation to an elliptic boundary value problem with mesh domain Ω_h, where both u and f are vectors, and A denotes a SPD and consistently ordered matrix with entries a_{ij}. If $a_{ij} \neq 0$ with $i \neq j$, mesh point i is said to be connected to mesh point j. Two subdomains of Ω_h are said to be connected if one subdomain has at least one mesh point connected to a mesh point from the other subdomain.

In the multi-type partition, the mesh domain Ω_h is partitioned into p disjoint subdomains, $\Omega_{h,j}$ for $j = 1, 2, \ldots, p$, if the linear system is solved on p processors of a parallel computer. Here each subdomain is only connected to its neighboring subdomains, and the work amounts related to subdomain $\Omega_{h,j}$ will be assigned to processor j for calculations. Moreover, the mesh points of each subdomain are grouped into t different types according to the following three rules: (i) connected types must be adjacent; (ii) no adjacent types are the same type; and (iii) every interior type has at least one adjacent type located in a neighboring subdomain. Here t is a positive integer determined by the connection information among neighboring subdomains, which is less than or equal to the number of neighboring subdomains. For clarity, the multi-type partition with t types will be referred to as the t-type partition. An illustration of the 2-type and 3-type partitions is given in Figures 1 and 2, respectively.

Let $\Omega_{h,\mu}^i$ denote the ith type of subdomain μ for $\mu = 1, 2, \ldots, p$ and $i = 1, 2, \ldots, t$. When these tp type subdomains are ordered in the natural ordering (i.e., from $i = 1$ to t for each μ from 1 to p), the linear system $Au = f$ can be written in the block matrix form with

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

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

$$\text{and } A_{\mu\nu} = \begin{cases} \begin{bmatrix} 0 & 0 & \cdots & 0 \\ A_{\mu\nu}^{21} & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ A_{\mu\nu}^{t1} & \cdots & A_{\mu\nu}^{t,t-1} & 0 \end{bmatrix} & \text{for } \mu < \nu, \\ \begin{bmatrix} 0 & A_{\mu\nu}^{12} & \cdots & A_{\mu\nu}^{1t} \\ 0 & 0 & \ddots & \vdots \\ \vdots & \vdots & \ddots & A_{\mu\nu}^{t-1,t} \\ 0 & 0 & \cdots & 0 \end{bmatrix} & \text{for } \mu > \nu. \end{cases} \tag{3}$$

Here $\mathcal{U}_\mu^i$ denotes a sub-vector of u defined on the subdomain $\Omega_{h,\mu}^i$, $A_{\mu\mu}^{ii}$ is the sub-matrix of A defined on $\Omega_{h,\mu}^i$, and $A_{\mu\nu}^{ij}$ with $\mu \neq \nu$ and $i \neq j$ is the sub-matrix that indicates the connection of $\Omega_{h,\mu}^i$ with $\Omega_{h,\nu}^j$. Clearly, if $\Omega_{h,\mu}^i$ is not adjacent to $\Omega_{h,\nu}^j$ with $\mu \neq \nu$ and $i \neq j$, then $A_{\mu\nu}^{ij} = 0$. In particular, it can be claimed that $A_{\mu\nu}^{ij} = 0$ for all $i \leq j$ if $\mu < \nu$ and for all $i \geq j$ if $\mu > \nu$. This gives the form of $A_{\mu\nu}$ with $\mu \neq \nu$ in (3).

3 The PSSOR preconditioner

This section defines the PSSOR preconditioner based on the block form of matrix A given in (2). It then discusses its basic properties generally.

Let D_μ and L_μ be the diagonal and strictly lower triangular matrices, respectively, of $A_{\mu\mu}$ so that $A_{\mu\mu} = D_\mu - L_\mu - L_\mu^T$ for $\mu = 1, 2, \ldots, p$. Here T denotes the transpose of a matrix. The block matrix A is split into the sum

$$A = D - B - B^T - N - N^T, \tag{4}$$

where D and B are two block diagonal matrices defined by

$$D = \text{diag}(D_1, D_2, \ldots, D_p) \quad \text{and} \quad B = \text{diag}(L_1, L_2, \ldots, L_p), \tag{5}$$

and N is a strictly lower block triangular matrix defined by

$$N = -\begin{bmatrix} 0 & 0 & \cdots & 0 \\ A_{21} & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ A_{p1} & \cdots & A_{p(p-1)} & 0 \end{bmatrix}. \tag{6}$$

Based on the sum in (4), the PSSOR preconditioner is defined by

$$M = \frac{1}{\omega(2-\omega)}[D - \omega(B + N^T)]D^{-1}[D - \omega(B^T + N)], \tag{7}$$

where the relaxation parameter $\omega \in (0, 2)$.

If $\omega = 1$, the PSSOR preconditioner can be simplified as

$$M = [D - B - N^T]D^{-1}[D - B^T - N)], \tag{8}$$

which is called the parallel symmetric Gauss-Seidel preconditioner.

Theorem 1 *If the matrix A defined in (2) is symmetric and positive-definite, so is the PSSOR preconditioner M. Here M is defined in (7).*

Proof: Set $\bar{R} = [D - \omega(B + N^T)]D^{-1}[D - \omega(B^T + N)]$. Since $M = \frac{1}{\omega(2-\omega)}\bar{R}$, one only needs to show that $\bar{R}$ is SPD .

Clearly, $\bar{R}$ is symmetric. From (5) it can be seen that D consists of the main diagonal entries of A, which are positive since A is positive-definite. Thus, $x^T Dx > 0$ and $x^T D^{-1}x > 0$ for nonzero vector x.

Set $\Lambda = D - \omega(B^T + N)$. From (4) it follows that $\Lambda + \Lambda^T = (2-\omega)D + \omega A$. Hence, for any nonzero vector x, and $\omega \in (0, 2)$,

$$x^T \Lambda x = \frac{1}{2}x^T(\Lambda + \Lambda^T)x = \frac{1}{2}[(2-\omega)x^T Dx + \omega x^T Ax] > 0.$$

So Λ is non-singular, and thus $\bar{R} = \Lambda^T D^{-1}\Lambda$ is positive definite. The proof is completed.

Since M is SPD, the inverse of M exists and can be found as below:

$$M^{-1} = \omega(2-\omega)[D - \omega(B^T + N)]^{-1}D[D - \omega(B + N^T)]^{-1}. \tag{9}$$

But, in practice, it is rare to find the inverse of M. Instead, the solution of the preconditioning equation $Mz = r$, $z = M^{-1}r$, can be found simply by one symmetric BPSOR (SBPSOR) iteration with an initial guess of zero. Here each SBPSOR iteration is defined by two half-iterations: the first one is the BPSOR method and the other one is the BPSOR method using the reverse order, which will be called the backward BPSOR method for clarity. From [13] it has been known that the BPSOR method has the iterative form

$$\begin{aligned} u^{(k+1)} &= [D - \omega(B + N^T)]^{-1}[(1-\omega)D + \omega(B^T + N)]u^{(k)} \\ &+ \omega[D - \omega(B + N^T)]^{-1}r, \quad k = 0, 1, 2, \ldots, \end{aligned} \tag{10}$$

where $u^{(0)}$ is an initial guess. By reversing the ordering that is used in implementing each BPSOR iteration, it is easy to obtain the iterative expression of the backward BPSOR method as below:

$$\begin{aligned} u^{(k+1)} &= [D-\omega(B^T+N)]^{-1}[(1-\omega)D+\omega(B+N^T)]u^{(k)} \\ &+ \omega[D-\omega(B^T+N)]^{-1}r, \quad k=0,1,2,\ldots. \end{aligned} \tag{11}$$

With $u^{(0)}=0$, the first BPSOR iterate $u^{(1)}=\omega[D-\omega(B+N^T)]^{-1}r$. Then, one backward BPSOR iteration starting at $u^{(1)}$ immediately gives the solution $z=M^{-1}r$. Hence, the PSSOR preconditioning equation $Mz=r$ can be solved easily by one SBPSOR iteration.

The pt type subdomains $\{\Omega^i_{h,\mu}\}$ of the t-type partition can also be ordered from $\mu=1$ to p for each value of type i from 1 to t while the original ordering is retained within each type subdomain. Such an ordering is called the multi-type ordering. In the multi-type ordering, the sub-vectors $\{\mathcal{U}^i_j\}$ are reordered in the form

$$\hat{u}=\begin{pmatrix} \hat{\mathcal{U}}_1 \\ \hat{\mathcal{U}}_2 \\ \vdots \\ \hat{\mathcal{U}}_t \end{pmatrix} \text{ with } \hat{\mathcal{U}}_i=\begin{pmatrix} \mathcal{U}^i_1 \\ \mathcal{U}^i_2 \\ \vdots \\ \mathcal{U}^i_p \end{pmatrix}.$$

Clearly, there exists a permutation matrix, P, such that $\hat{u}=Pu$ with u being given in (1). In terms of P, the reordered linear system by the multi-type ordering can be expressed as $\hat{A}\hat{u}=\hat{f}$ with $\hat{A}=PAP^T, \hat{u}=Pu$, and $\hat{f}=Pf$. Thus, the SSOR preconditioner using the t-type ordering, which is also called the t-type SSOR preconditioner, can be obtained as below:

$$\hat{M}=\frac{1}{\omega(2-\omega)}[\hat{D}-\omega\hat{L}]\hat{D}^{-1}[\hat{D}-\omega\hat{L}^T], \tag{12}$$

where $\hat{D}$ and $\hat{L}$ are diagonal and strictly lower triangular matrices, respectively, satisfying $\hat{A}=\hat{D}-\hat{L}-\hat{L}^T$.

Theorem 2 *If M and $\hat{M}$ are the PSSOR preconditioner and the t-type SSOR preconditioner defined in (7) and (12), respectively, then there exists a permutation matrix P such that*

$$\hat{M}=PMP^T. \tag{13}$$

Proof: Clearly, there exists the permutation matrix P such that $\hat{u}=Pu$. In terms of P, it is easy to see that $\hat{D}=PDP^T$. With $u^{(0)}=0$, the first BPSOR iterate $u^{(1)}$ for solving $Au=f$ becomes

$$u^{(1)}=\omega[D-\omega(B+N^T)]^{-1}f.$$

Multiplying the above expression by P from the left-hand side and using the identities $P^TP=I$, $Pf=\hat{f}$, and $PDP^T=\hat{D}$ give that $\hat{u}^{(1)}=\omega[\hat{D}-\omega P(B+M)P^T]^{-1}\hat{f}$. On

the other hand, since $\hat{u}^{(0)} = Pu^{(0)} = 0$, the first SOR iterate for solving the reordered linear system $\hat{A}\hat{u} = \hat{f}$ becomes $\hat{u}^{(1)} = \omega[\hat{D} - \omega\hat{L}]^{-1}\hat{f}$. Combining these two expressions of $\hat{u}^{(1)}$ yields the identity

$$[\hat{D} - \omega P(B + N^T)P^T]^{-1}\hat{f} = [\hat{D} - \omega\hat{L}]^{-1}\hat{f} \quad \text{for all nonzero vector } \hat{f}.$$

From the above identity it follows that $[\hat{D}-\omega P(B+N^T)P^T]^{-1} = [\hat{D}-\omega\hat{L}]^{-1}$, which can be simplified as $\hat{L} = P(B+N^T)P^T$. Thus, combing it with $PDP^T = \hat{D}$ and $P^T = P^{-1}$ gives

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

Multiplying both sides of the above identity by the constant $1/[\omega(2-\omega)]$ yields $\hat{M} = PMP^T$. This completes the proof.

Identity (13) implies that the PSSOR preconditioner M has the same eigenvalues as the t-type SSOR preconditioner $\hat{M}$. In this sense, M is said to be equivalent to $\hat{M}$. Thus, the analysis of the PSSOR preconditioner can be done simply by applying the classic SSOR theory [14] to the t-type SSOR preconditioner.

4 PSSOR preconditioners for an anisotropic model problem

This section studies the effects of the PSSOR preconditioner on the convergence rate of PCG. To do so, it introduces the 2-type and 3-type partitions and orderings. It then constructs two particular PSSOR preconditioners, the 2-type and 3-type PSSOR preconditioners, for solving an anisotropic model problem. For comparison, the sequential SSOR preconditioner and the red-black SSOR preconditioner are also constructed. The effect studies are then carried out through condition number calculations and comparisons.

The anisotropic model problem is given as below:

$$\begin{cases} -(au_{xx} + bu_{yy}) = f(x,y) & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega, \end{cases} \tag{14}$$

where a and b are positive constants with $a \geq b$, $\Omega = (0,1) \times (0,1)$, and $\partial\Omega$ denotes the boundary of Ω. By the five-point finite difference formula, the model problem can be approximated as the following linear system

$$2(1 + \frac{b}{a})u_{ij} - (u_{i+1,j} + u_{i-1,j}) - \frac{b}{a}(u_{i,j+1} + u_{i,j-1}) = \frac{h^2}{a}f_{ij}, \tag{15}$$

where $i, j = 1, 2, \ldots, n-1, h = 1/n$ with $n > 1$, $f_{ij} = f(ih, jh)$, u_{ij} denotes an approximation of the solution $u(ih, jh)$, and $u_{i0} = u_{in} = u_{0i} = u_{ni} = 0$ for $i = 0, 1, 2, \ldots, n$.

In the 2-type partition, the mesh domain Ω_h is partitioned into p strips along one dimension of the space. For the model problem (15), each strip contains at least two

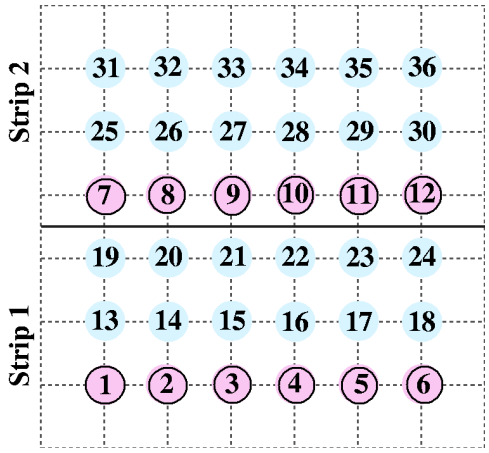

Figure 1: The 2-type partition and ordering for the model problem (15) with $h = 1/7$, $p = 2$ and $t = 2$. The 1st type mesh points are marked in circles.

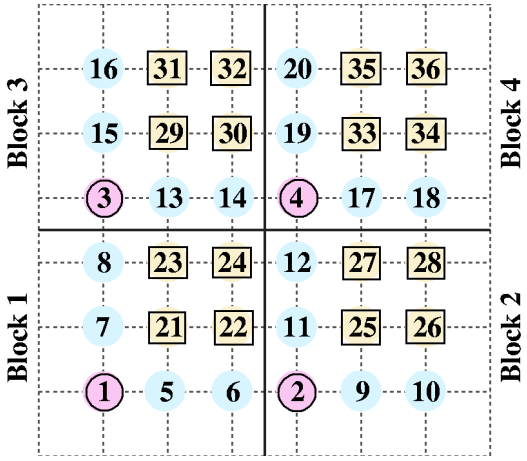

Figure 2: The 3-type partition and ordering for the model problem (15) with $h = 1/7$, $p = 2 \times 2$ and $t = 3$. The 3rd type mesh points are marked in squares.

mesh lines, and the grid points of each strip are divided into two types, $\Omega^1_{h,\mu}$ and $\Omega^2_{h,\mu}$, since each strip has two neighboring strips (except the two boundary strips). The first type $\Omega^1_{h,\mu}$ consists of the mesh points of the first mesh line of each strip, and the second type $\Omega^2_{h,\mu}$ consists of the remaining mesh points. The 2-type ordering is then defined by first numbering the mesh points of type 1 and then type 2 in the natural ordering (left to right and bottom to top within each strip) across strips.

Figure 1 illustrates the construction of the 2-type partition for the mesh domain Ω_h with $h = 1/7$, $p = 2$ and $t = 2$. Here each strip contains three mesh lines, and the mesh points are numbered from 1 to 36 in the 2-type ordering. The mesh points of types 1 and 2 are numbered from 1 to 12 and from 13 to 36, respectively.

In the 3-type partition, the mesh domain Ω_h is partitioned into p blocks along two dimensions of the space. Here $p = \tilde{m} \times \tilde{n}$ if there are $\tilde{m}$ blocks in one dimension, and $\tilde{n}$ blocks in the other dimension. For the model problem (15), each block is connected to its neighboring blocks from the left, right, bottom, and top. Thus, the grid points of each block are divided into three types ($t = 3$) such that each type is connected to two neighboring blocks.

An illustration of the 3-type partition is given in Figure 2 for the mesh domain Ω_h with $h = 1/7$, $p = 2 \times 2$ and $t = 3$. In this example, the 3-type ordering is marked out from 1 to 36 in the figure. Types 1, 2 and 3 contain the mesh points numbered from 1 to 4, 5 to 20, and 21 to 36, respectively. Note that type 2 subdomain, $\Omega^2_{h,\mu}$, consists of the mesh points of the first horizontal mesh line and the first vertical mesh line of block $\Omega_{h,\mu}$ except for the first mesh point. The reason to select type 2 in this way is based on the following two considerations: (i) the iterates on these mesh points can be

updated as a group, and (ii) the iterates defined on type 2 mesh points can be sent and received efficiently as two arrays (one is related to the bottom neighboring block, and the other to the left neighboring block).

Based on the 2-type partition, the block matrix A of (2) becomes a tridiagonal block matrix in the form

$$A = \begin{bmatrix} A_{11} & A_{12} & & \\ A_{21} & A_{22} & \ddots & \\ & \ddots & \ddots & A_{p-1,p} \\ & & A_{p,p-1} & A_{pp} \end{bmatrix} \text{ with } A_{\mu\mu} = \begin{bmatrix} A^{11}_{\mu\mu} & A^{12}_{\mu\mu} \\ A^{21}_{\mu\mu} & A^{22}_{\mu\mu} \end{bmatrix}, \tag{16}$$

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

In particular, for the model problem (15), it can be found that

$$A^{11}_{\mu\mu} = \mathcal{B}, \quad A^{12}_{\mu\mu} = -cI, \quad A^{21}_{\mu,\mu+1} = -cI,$$

$$A^{22}_{\mu\mu} = \begin{bmatrix} \mathcal{B} & -cI & & \\ -cI & \mathcal{B} & \ddots & \\ & \ddots & \ddots & -cI \\ & & -cI & \mathcal{B} \end{bmatrix}, \quad \mathcal{B} = \begin{bmatrix} \alpha & -1 & & \\ -1 & \alpha & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & \alpha \end{bmatrix},$$

where $\alpha = 2(1 + b/a), c = b/a$, I is an $(n-1) \times (n-1)$ identity matrix, $\mathcal{B}$ is an $(n-1) \times (n-1)$ tridiagonal matrix, $A^{22}_{\mu\mu}$ is an $\ell \times \ell$ block matrix if $\Omega^2_{h,\mu}$ contains ℓ mesh lines, and $\mu = 1, 2, \ldots, p$.

Based on the 3-type partition, the block matrix A of (2) becomes a five-diagonal block matrix with nonzero entries lying on the main diagonal, the second off-diagonal, and the $(\tilde{m}+1)$th off-diagonal. For the model problem (15) with $h = 1/7$, the block matrix A with $p = 2 \times 2$ and $t = 3$ can be found in the form

$$A = \begin{bmatrix} \mathcal{A} & \mathcal{G} & \mathcal{H} & \mathbf{0} \\ \mathcal{G}^T & \mathcal{A} & \mathbf{0} & \mathcal{H} \\ \mathcal{H}^T & \mathbf{0} & \mathcal{A} & \mathcal{G} \\ \mathbf{0} & \mathcal{H}^T & \mathcal{G}^T & \mathcal{A} \end{bmatrix} \text{ with } \mathcal{A} = \begin{bmatrix} A^{11} & A^{12} & \mathbf{0} \\ A^{12^T} & A^{22} & A^{23} \\ \mathbf{0} & A^{23^T} & A^{33} \end{bmatrix}, \quad A^{11} = \alpha, \tag{17}$$

$$A^{22} = \begin{bmatrix} \alpha & -1 & 0 & 0 \\ -1 & \alpha & 0 & 0 \\ 0 & 0 & \alpha & -c \\ 0 & 0 & -c & \alpha \end{bmatrix}, \quad A^{33} = \begin{bmatrix} \alpha & -1 & -c & 0 \\ -1 & \alpha & 0 & -c \\ -c & 0 & \alpha & -1 \\ 0 & -c & -1 & \alpha \end{bmatrix},$$

$$A^{23} = \begin{bmatrix} -c & 0 & 0 & 0 \\ 0 & -c & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix}, \qquad A^{12} = [-1 \quad 0 \quad -c \quad 0],$$

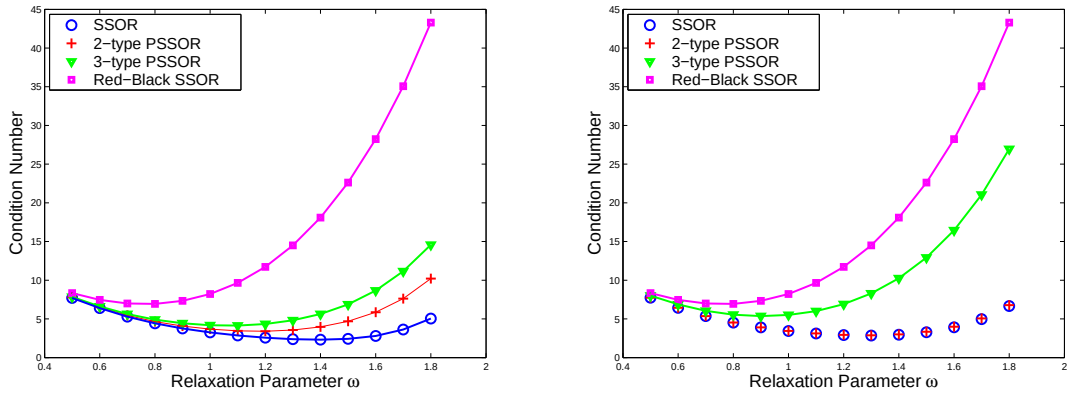

Figure 3: Comparison of the condition numbers of the PSSOR preconditioners with that of the sequential and red-black SSOR preconditioners for the model problem (15) with $h = 1/7$. The left plot for $a = b = 1$, and the right plot for $a = 10, b = 1$.

$\mathcal{G} = (g_{ij})$ is a 9×9 matrix with only three nonzero entries given by $g_{31} = g_{74} = g_{95} = -1$, and $\mathcal{H} = (h_{ij})$ is a 9×9 matrix with only three nonzero entries given by $h_{51} = h_{82} = h_{93} = -c$.

With the two matrix forms of A given in (16) and (17), the 2-type and 3-type PSSOR preconditioners are constructed according to formula (7), respectively. The sequential SSOR preconditioners is constructed from the matrix A of (16) according to the formula

$$M = \frac{1}{\omega(2-\omega)}(D - \omega L)D^{-1}(D - \omega L^T),$$

where D is the diagonal matrix of A and L is the strictly lower triangular matrix such that $A = D - L - L^T$. Furthermore, the red-black matrix form of A is obtained by reordering the matrix A of (16) in the red-black ordering, from which the red-black SSOR preconditioner is then constructed.

From the PCG theory [4] it is known that the effect of a preconditioner M on the convergence rate of the PCG method can be studied directly by evaluating the condition number of $M^{-1}A$. The smaller the condition number, the faster the convergence rate [4]. For the model problem with $h = 1/7$, the condition numbers of the above four preconditioners were calculated on MATLAB, and reported in Figure 3.

Figure 3 shows that the 2-type and 3-type PSSOR preconditioners have much smaller condition numbers than the red-black SSOR preconditioner, while the SSOR preconditioner has the smallest condition number. Hence, the PSSOR preconditioner is more effective than the red-black SSOR preconditioner while the sequential SSOR preconditioner is the most effective among them. Figure 3 also shows that the condition number is a quadratic function of ω. Hence, for each preconditioner, there exists an optimal value of ω at which the smallest condition number is reached.

From the right plot (with $a = 10$ and $b = 1$) of Figure 3 it can be seen that the 2-type PSSOR preconditioner has almost the same condition number as the SSOR

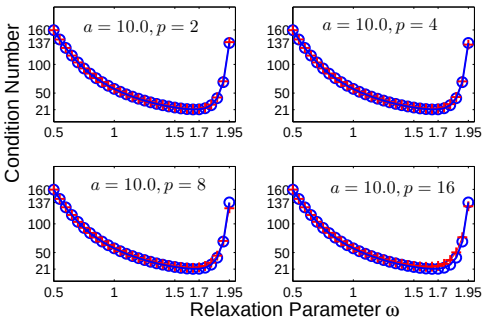

Figure 4: Condition number comparisons between the parallel 2-type PSSOR preconditioner (in +) and the sequential SSOR preconditioner (in ∘) for the model problem (15) with $h = 1/33$, $a = 10$ and $p = 2, 4, 8, 16$.

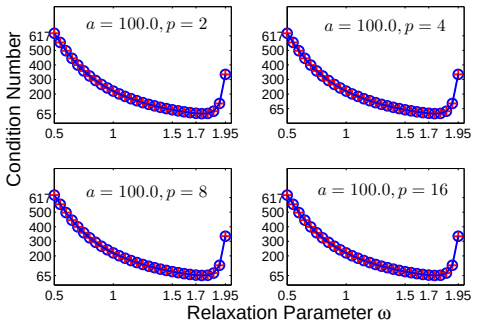

Figure 5: Condition number comparisons between the parallel 2-type PSSOR preconditioner and the sequential SSOR preconditioner for the model problem (15) with $h = 1/65$, $a = 100$ and $p = 2, 4, 8, 16$.

preconditioner. To further explore this phonomania, more condition numbers were calculated on two large grid domains ($h = 1/33$ and $h = 1/65$) with $a = 10, 100$ and $p = 2, 4, 8, 16$. The numerical results were reported in Figures 4 and 5. These two figures demonstrate that the condition numbers became larger on a larger grid domain but the parallel 2-type PSSOR preconditioner still almost retained the condition numbers of the sequential SSOR preconditioner for different values of p.

5 Parallel performance of PSSOR preconditioner

To investigate the parallel performance of the PSSOR preconditioner, a parallel program of PCG was developed in Fortran 77 and MPI (the Message Passing Interface library) [11], which includes the 2-type and 3-type PSSOR preconditioners and the red-black SSOR preconditioner. Numerical experiments were made on a MIMD parallel computer (the SGI Origin 2000 computer at the University of Wisconsin-Milwaukee, which has 16 R12000 400 MHz processors) for solving the model problem (15) with $f = 1.0$ and $h = 1/517$, where four anisotropic ratios a/b were set by $a = 1, 10, 100, 1000$ for $b = 1$. All the numerical tests used an initial guess of zero, and the same iteration stop rule in which the relative residue norm is less than 10^{-6}. In addition, the optimal values of the relaxation parameter ω were used in these tests, which were determined by experiments. The program was compiled using the optimization level $O2$. The CPU time was measured by the MPI function MPI_Wtime, which returns the wall time in seconds. All calculations were done with double precision. The sequential PCG method using the SSOR preconditioner was programmed in Fortran 77 and implemented on one processor of the computer, where the Fortran 77 time function $etime()$ was used to measure the CPU time.

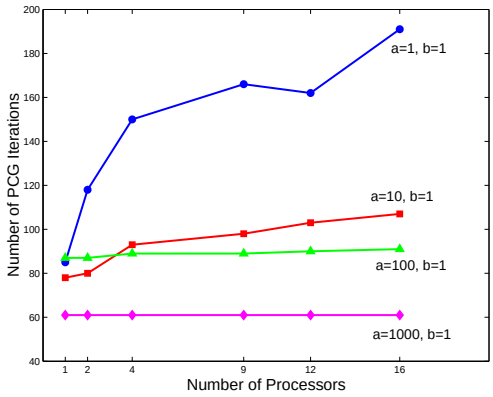

Figure 6: Convergence dependence of the PCG using the PSSOR preconditioner on the number of processors and the anisotropic ratio a/b.

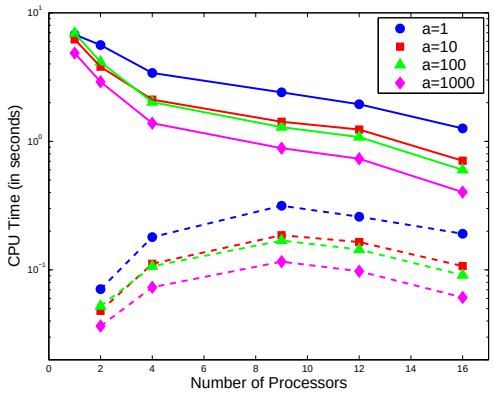

Figure 7: Parallel performance of the PCG using the PSSOR preconditioner. The dash lines indicate the total interprocessor data communication time.

Figure 6 shows that the PCG using the PSSOR preconditioner can have a faster convergence speed for a larger value of the ratio a/b. Also, it shows that the number of processors has less effects on the convergence rate for a larger anisotropic ratio of a/b.

Figure 7 displays the parallel performances of the PCG using the PSSOR preconditioner. It shows that the total CPU time is a decreasing function of the number of processors. Compared to the total CPU time, the time consumed by interprocessor data communication is very small. A speedup of 12 was obtained on the 16 processors compared to the sequential PCG using the sequential SSOR preconditioner.

Figures 8 and 9 compare the parallel performance of the PCG using the 2-type and 3-type PSSOR preconditioners with that of the PCG using the red-black SSOR preconditioner. Here $a = 10$ and $b = 1$. The block partitions of two by two, three by three and four by four blocks were used in the construction of the 3-type PSSOR preconditioner when the tests were implemented on four, nine and sixteen processors, respectively. From these two figures it can be seen that the PSSOR preconditioner accelerated the convergence speed of PCG (in terms of the total number of PCG iterations determined by the iteration stop rule) for about 3 to 6 times compared to the red-black SSOR preconditioner. Moreover, the PSSOR preconditioner reduced the total CPU time and interprocessor data communication time of the PCG using the red-black SSOR preconditioner by a factor of 3 to 6. These numerical results confirm that the PSSOR preconditioner is much more effective than the red-black SSOR preconditioner. In addition, these two figures also show that the 2-type PSSOR preconditioner was more effective than the 3-type PSSOR preconditioner in enhancing the convergence rate and parallel performance of PCG.

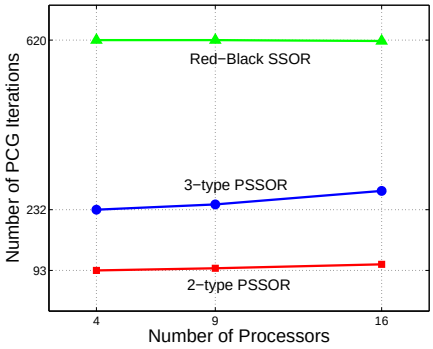

Figure 8: Convergence comparison of the PCG using the 2-type and 3-type PSSOR preconditioners with the PCG using the red-black SSOR preconditioner.

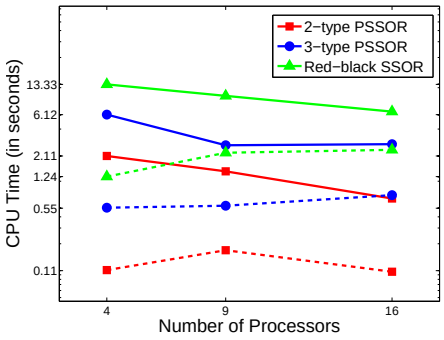

Figure 9: Parallel performance comparison of the PCG using the 2-type and 3-type PSSOR preconditioners with the PCG using the red-black SSOR preconditioner. The dash lines indicate the total interprocessor data communication time.

6 Conclusions

A new parallel symmetric SOR preconditioner by the multi-type partition, the PSSOR preconditioner, has been well defined and analyzed for a general block linear system. According to the general mathematical analysis, the PSSOR preconditioner can be easily and effectively applied to PCG to accelerate the convergence. This paper also has well studied the effects of the PSSOR preconditioner on the convergence rate of PCG through the two particular PSSOR preconditioners, the 2-type and 3-type PSSOR preconditioners, for solving the anisotropic model problem, showing that the PSSOR preconditioner can make PCG to have a much faster rate of convergence than the red-black SSOR preconditioner while the 2-type PSSOR preconditioner can retain the rate of convergence of the PCG using the classic sequential SSOR preconditioner when the problem has a large anisotropy. Finally, the numerical experiments on a MIMD parallel computer have confirmed that the PCG using the PSSOR preconditioner can be much more efficient than the PCG using the red-black SSOR preconditioner in both convergence rate and parallel performance in terms of the total number of PCG iterations and the total computer CPU time.

In the future, it is planned to compare the PSSOR preconditioner with other parallel SSOR preconditioners. The parallel performances of the PSSOR preconditioner will be further investigated for solving a three dimensional elliptic boundary value problem on some new computational environments (e.g., a cluster of PC systems with each PC containing multi-cores processors). Through these new studies, the application range of the PSSOR preconditioner is expected to be significantly expanded.

Acknowledgements

The author thanks the two anonymous referees for their valuable comments and suggestions. This project was supported by the National Science Foundation through grant DMS-0241236.

References

[1] Ashcraft, C. and Grimes, R., 1988, On vectorizing incomplete factorization and SSOR preconditioners, *SIAM J. Sci. Statist. Comput.*, **9**, 122-151.

[2] Block, U., Frommer, A., and Mayer, G., 1990, Block colouring schemes for the SOR method on local memory parallel computers. *Parallel Computing*, **14**, 61-75.

[3] Farhat, C., 1986, Multiprocessors in computational mechanics. Ph.D. thesis, Civil Engineering Department, University of California, Berkeley, CA.

[4] Golub, G. H. and van Loan, C. F., 1996, *Matrix Computations*. (3rd edn), (Baltimore: John Hopkins University Press).

[5] Hackbusch, W., 1994, *Iterative Solution of Large Sparse Systems of Equations*. (New York: Springer-Verlag).

[6] Harrar, D. and Ortega, J., 1989, Multicoloring with lots of colors. In *Proc. Third Internat. conference Supercomputing*, pages 1–6, Grete, Greece.

[7] Lippert, Th., 1999, Parallel SSOR preconditioner for lattice QCD. *Parallel Computing*, **25**, 1357-1370.

[8] Melhem, R., 1987, Towards efficient implementations of preconditioned conjugate methods on vector supercomputers. *Internat. J. Supercomput. Appl.*, **1**, 70-98.

[9] Stotland, S. A. and Ortega, J. M., 1997, Orderings for parallel conjugate gradient preconditioners. *SIAM J. Sci. Comput.*, **18**, 854-868.

[10] van der Vorst, H. A., 1987, Large tridiagonal and block tridiagonal linear systems on vector and parallel computers. *Parallel Computing*, **5**, 45-54.

[11] William Gropp, W., Lusk, E., and Skjellum, A., 1999, *Using MPI: Portable Parallel Programming with the Message-Passing Interface*, (2nd edn), Cambridge, Massacjisetts, MIT Press.

[12] Xie, D. and L. Adams, L., 1999, New parallel SOR method by domain partitioning. *SIAM J. Sci. Comput*, **20**, 2261-2281.

[13] Xie, D., 2006, A new block parallel SOR method and its analysis. *SIAM J. Sci. Comput*, **27**, 1513-1533.

[14] Young, D. M., 1971, *Iterative Solution of Large Linear System*, (New York: Academic press).