SchurRAS: A restricted version of the overlapping Schur complement preconditioner

Zhongze Li [*] Yousef Saad [†]

May 17, 2004

Abstract

This paper presents a preconditioner based on solving approximate Schur complement systems with overlapping restricted additive Schwarz methods (RAS). The construction of the preconditoner, called SchurRAS, is as simple as in the standard RAS. The communication requirements for each application of the preconditioning operation are similar with those of the standard RAS approach. In the particular case when the degree of overlap is one, then SchurRAS and RAS involve exactly the same communication volume per step. In addition, SchurRAS has the same degree of parallelism as RAS. In some numerical experiments with a model problem, the convergence rate of the method was found to be similar to that of the Multiplicative Schwarz (MS) method. The Schur based RAS usually outperforms the standard RAS both in terms of iteration count and CPU time. For a few two dimensional scaled problems, SchurRAS was about twice as fast as the stardard RAS on 64 processors.

1 Introduction

In the past few decades, *incomplete LU* (ILU) preconditioners, combined with Krylov subspace acceleration, have yielded techniques which offer a good compromise between robustness and efficiency when solving general large sparse linear systems of equations, see, e.g., [12, 17]. The extension of these techniques to parallel computing environments has been done in two distinct ways. The first is to extract parallelism via a reordering of the unknowns. The second is to resort to domain decomposition ideas. Multicoloring is a standard way of uncovering parallel tasks in an algorithm such as Gaussian elimination. For example, parallel ILU-type preconditioners based on graph partitioning and multicoloring were recently presented [8, 6, 11]. Constructing a parallel ILU preconditioner with multicoloring starts with a coloring of the interface variables. The elimination proceeds with the interior nodes first, and this phase is perfectly parallel. Then the interface variables are eliminated in certain order which is based on the colors of the interfaces. This basic idea was already exploited for a parallel implementation of ILU(0), in 1994 [10] (see also the 1996 edition of [12]).

Concerning the second class of methods mentioned above, Domain Decomposition Methods (DDM) have given rise to a general-purpose strategy for constructing global preconditioners for sparse linear systems [12, 17]. These methods were initially developed for solving Partial Differential Equations (PDEs), but there is an increasing interest in DDM-like preconditioners for general sparse linear systems, obtained from a purely algebraic viewpoint, see, e.g., [5, 2]. In

[*]Institute of Computational Mathematics and Scientific/Engineering Computing, Academy of Mathematics and System Sciences, Chinese Academy of Sciences, China, lzz@lsec.cc.ac.cn. Work supported by the State Key Laboratory of Scientific and Engineering Computing and by the Joint Research Institute for Applied Mathematics between Peking University and Hong Kong Baptist University.

[†]Department of Computer Science & Engineering, the University of Minnesota, U.S.A., saad@cs.umn.edu. Work supported by the NSF under grant ACI-0305120 and by the Minnesota Supercomputing Institute.

domain decomoposition methods, the initial problem domain is partitioned into a number of subdomains, possibly with overlapping interfaces. Such extensions lead to *Distributed sparse systems* [12], whereby the equations and associated unknowns are assigned to processor with the help of a graph partitioning approach. Using this viewpoint, many of the domain decomposition methods can be retrofitted to the solution of distributed sparse linear systems. The method then is to compute an approximate solution from solutions in the subdomains – exploiting knowledge about the operator, discretization, and other properties of the problem, when possible. In the simplest case of the Additive Schwarz Methods (ASM), the iterative process consists of simply solving independent problems and updating residuals in each domain after each iteration. Variants of ASM are obtained by varying the amount of overlap and the subdomain solvers. If the subdomains do not overlap the procedure results in a method that essentially amounts to a *Block Jacobi* preconditioner. At this point it is important to note that Schwarz-type procedures are straightforward to extend to general sparse linear systems, at least from an algorithmic point of view. Though the body of theory that is available in DDM for PDEs does not easily extend without additional information, domain decomposition gives rise to fairly efficient methods for sparse linear systems, which have the advantage of being easy to implement [2].

In [3] Cai and Sarkis introduced an effective variant of the Schwarz procedure, called the Restricted Additive Schwarz (RAS) method for the case where subdomains are overlapped and each subdomain problem is approximately solved with incomplete factorization. The initial motivation that lead to the discovery of for RAS was to reduce communication costs – by omitting half of the communication in the standard overlapping AS method. However, it turns out that this is only incidental to the efficiency of the method because this omition has the unexpected effect of also reducing the number of iterations.

In practice, ASM-type preconditioners are preferable to multicoloring ILU preconditioners in that they yield a better performance in most cases [4, 1]. ASM usually outperforms non-overlapping Block Jacobi preconditioners both in terms of iteration counts and execution times because the block Jacobi preconditioner ignores all inter-domain couplings. To improve the performance for non-overlapping domain decomposition, Schur complement-based preconditioners were advocated as alternatives to the simple block Jacobi techniques [13, 9]. Though Schur complement preconditioners take fewer iterations to converge, they lead to higher execution times to converge in some cases because of the additional costs incurred by the inner iterations when solving the Schur complement systems.

This paper presents a technique (SchurRAS) which in effect, brings together the best of ASM on the one hand and Schur complement methods on the other. As in RAS, the new algorithm combines overlapping and non-overlapping iterations. However, these iterations now involve Schur complements, i.e., they work on interface variables.

The rest of the paper is organized as follows. Section 2 introduces distributed sparse systems and Schwarz procedures, as well as Schur complements. Section 3 presents the restricted version of the Schur complement preconditioner, Section 4 gives some details on the parallel implementation of SchurRAS. Numerical experiments are shown in Section 5 and some conclusions are drawn in Section 6.

2 Distributed systems and distributed preconditioners

When solving a linear system $Ax = b$ on a distributed memory computer it is common to start by partitioning the adjacency graph of the coefficient matrix A by a graph partitioner [7]. Recall that the *adjacency graph* of A is a graph $G = (V, E)$, where V is the set of all vertices, representing the unknowns of the system, and E is the set of all edges representing the binary relation "a_{ij} is a nonzero entry" between vertices i and j [12]. Assuming that the matrix A has a symmetric nonzero pattern, the adjacency graph G is undirected. The partitioning of G will result in p non-overlapping subgraphs $V_1, \cdots, V_p$. Typically, the group of equations associated with the set V_i will be assigned to processor i, along with the corresponding unknwons. Thus, each processor holds a set of equations (rows of the linear system) and vector components associated with these rows. Note that we assume a 'vertex-based' partitioning. This means that vertices are assigned to processors (instead of edges or elements in a finite element problem). In this situation, unknowns can be categorized into three classes: (1) Interior variables which are coupled only with local variables by the equations; (2) Local interface variables which are coupled with non-local (external) variables as well as local variables; and (3) External interface variables which are unknowns in other processors are coupled with local variables. Figure 1 shows an illustration.

When putting together the equations assigned to a given processor, one can observe that the coefficient matrix consists of two pieces: the first called A_i, acts on local variables and the other, called X_i acts on remote variables. Thus, the global linear system $Ax = b$ can be rewritten in each processor as

$$A_i x_i + X_i x_{i,ext} = b_i, \tag{1}$$

where x_i are the vector of local unknowns, $x_{i,ext}$ are the external interface variables, b_i is the corresponding local part of the right hand side vector. The above system is referred to as the "local system".

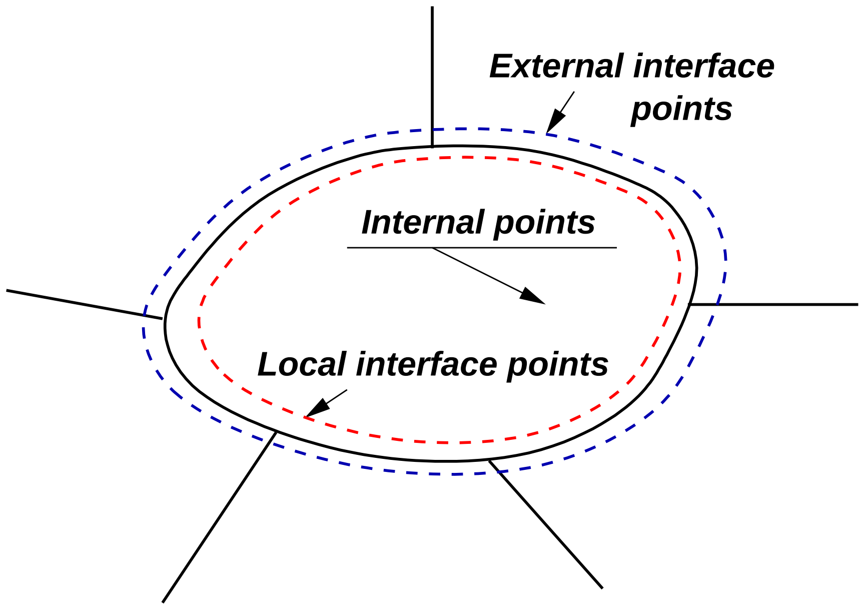

Figure 1: A local view of a distributed sparse matrix.

For convenience, a local reordering of the data is often exploited by splitting the local vector x_i into two parts: the subvector u_i of interior nodes followed by the subvector v_i of local interface nodes. The local right hand side vector b_i is conformally split into the subvectors f_i and g_i, so that:

$$x_i = \begin{pmatrix} u_i \\ v_i \end{pmatrix} \ ; \quad b_i = \begin{pmatrix} f_i \\ g_i \end{pmatrix} \ . \tag{2}$$

With this reordering, equation (1) takes the form:

$$\underbrace{\begin{pmatrix} B_i & E_i \\ F_i & C_i \end{pmatrix}}_{A_i} \begin{pmatrix} u_i \\ v_i \end{pmatrix} + \underbrace{\begin{pmatrix} 0 \\ \sum_{j \in N_i} E_{ij} v_j \end{pmatrix}}_{X_i x_{i,ext}} = \begin{pmatrix} f_i \\ g_i \end{pmatrix} . \tag{3}$$

Here, N_i is the set of indices for subdomains that are adjacent to the subdomain i. The term $E_{ij}v_j$ is the contribution to the local equation from neighboring subdomain j. The zero block in the second term of the left-hand side of (3) is due to the fact that local internal nodes are not coupled with external nodes.

2.1 Schwarz procedures for distributed systems

Let the global domain Ω be decomposed into p non-overlapping subdomains Ω_i, obtained from a partitioning of V into p non-overlapping subgraphs V_i^0. Here the superscript 0 is added to indicate that the domains do not overlap. Each subdomain Ω_i can then be extended to a larger subdomain Ω_i^1 to form one-level overlapped subdomain by including all immediate neighboring nodes of V_i^0. We may similarly define δ-level overlapped subdomain Ω_i^δ with the set of nodes V_i^δ. Associated with each subdomain Ω_i^δ, we can define the restriction operator

$$\begin{array}{cccc} R_{i,\delta}: & \Omega & \longrightarrow & \Omega_i^\delta \\ & z & \longrightarrow & z_{i,\delta} \end{array}$$

where $z_{i,\delta} = R_{i,\delta}z$ is the vector of Ω_i^δ which consists only of components of z belonging to Ω_i^δ. It is common to use the transpose $R_{i,\delta}^T$ as the corresponding *extension* operator

$$\begin{array}{cccc} R_{i,\delta}^T: & \Omega_i^\delta & \longrightarrow & \Omega \\ & z_{i,\delta} & \longrightarrow & z \end{array}$$

where $z = R_{i,\delta}^T z_{i,\delta}$ is the zero-extension of a vector $z_{i,\delta}$ on Ω_i^δ to Ω. The local matrix $A_{i,\delta}$ associated with the subdomain Ω_i^δ can be expressed as $A_{i,\delta} = R_{i,\delta}^T A R_{i,\delta}$. The additive Schwarz preconditioner can be written as [16]

$$\mathcal{B}_{[\delta,\delta]} = \sum_{i=1}^{p} R_{i,\delta}^T A_{i,\delta}^{-1} R_{i,\delta}. \tag{4}$$

In the case when there is no overlap, i.e., when $\delta = 0$, we obtain the special case of the *block-Jacobi* preconditioner:

$$\mathcal{B}_{[0,0]} = \sum_{i=1}^{p} R_{i,0}^T A_{i,0}^{-1} R_{i,0}. \tag{5}$$

Finally, using the same notation, the restricted additive Schwarz proposed by Cai and Sarkis [3] can be written as

$$\mathcal{B}_{[0,\delta]} = \sum_{i=1}^{p} R_{i,0}^T A_{i,\delta}^{-1} R_{i,\delta}. \tag{6}$$

The solve with the matrix A_i is rarely performed exactly. Instead, one can obtain an Incomplete LU factorization and solve the systems with an preconditioned Krylov method. Examining the incomplete factorization carefully will also yield an approach based on Schur complements.

2.2 Schur Complement – based preconditioners

Eliminating u_i from the top part of (3) leads to the relation

$$u_i = B_i^{-1}[f_i - E_i v_i] \tag{7}$$

which upon substitution into the bottom part, results in the following equation for v_i,

$$S_i v_i + \sum_{j \in N_i} E_{ij} v_j = g_i - F_i B_i^{-1} E_i \equiv g_i', i = 1, \cdots, p \tag{8}$$

where S_i is the "local" Schur complement matrix:

$$S_i = C_i - F_i B_i^{-1} E_i. \tag{9}$$

Note that this elimination of the u_i's is local, in that it can be carried out on each processor in parallel since interior unknowns of each subdomain are not coupled with variables of other subdomains.

Equations (8) for all subdomains constitute the global Schur complement system which involves only all interface variables $v_i, i = 1, \cdots, p$. The global Schur complement matrix has a natural block structure:

$$\begin{pmatrix} S_1 & E_{12} & E_{13} & \cdots & E_{1p} \\ E_{21} & S_2 & E_{23} & \cdots & E_{2p} \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ E_{p1} & E_{p2} & E_{p3} & \cdots & S_p \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ \vdots \\ v_p \end{pmatrix} = \begin{pmatrix} g_1' \\ g_2' \\ \vdots \\ \vdots \\ g_p' \end{pmatrix} . \tag{10}$$

The above system can be written as

$$Sv = g',$$

where v consists of all interface variables $v_1, v_2, \ldots, v_p$ stacked into a long vector The matrix S is the global Schur complement matrix.

If the global schur complement system (10) is solved exactly, the u variables can be obtained from (7), and the exact solution of the whole system will then be obtained. Preconditioning techniques can be obtained by solving the Schur complement system approximately.

3 RAS versions of Schur complement preconditioners

From the local linear system (3), interior variables u_i can be expressed as

$$u_i = B^{-1}(f_i - E_i v_i) . \tag{11}$$

Once interface variables v_i are available, all interior variables u_i can be computed via (11). This process can be carried out in parallel. If we simply drop the term E_{ij} in (10), v_i can be computed via $S_i^{-1} g_i'$ without any communication among processors. This amounts to a block Jacobi preconditioner. In passing, we note that the block Jacobi preconditioner for the original linear system can be induced by the block Jacobi preconditioner for the global Schur complement system[12, 13]. Due the local nature of the block Jacobi preconditioner, the resulting iteration will usually converge slowly, since one may expect that changes in a given domain will take quite a few steps before propagating to all domains.

One way to cure this is to attack the global Schur complement system directly, using Krylov subspace methods with block Jacobi or distributed ILU(0) as preconditioners[13, 9]. Solving the global Schur complement system more accurately, will result in a preconditioner that is usually superior to the standard block-Jacobi preconditioner for the original system. Part of this superiority comes from the fact that Schur complement-based preconditioners tend to require fewer iterations than ASM to converge. However, note that because the inner steps for solving the global Schur complement systems are more expensive, the overall performance may be poorer in some cases[9].

Part of the problem with the Schur-complement based methods developed in [9, 13] lies in the ineffectiveness of the block Jacobi preconditioner which is used to solve the global Schur complement system. Since RAS shows better performance than the block Jacobi for the original linear system one can think of solving the global schur complement system using an RAS-like procedure for the Schur complement. We will refer to this approach as SchurRAS. Because the condition number of the Schur complement system is better than that of the original linear system[16] and RAS is a better preconditioner than the block Jacobi preconditioner, we may expect to obtain a more "exact" solution for the global Schur complement system. Therefore, SchurRAS should outperform the block jacobi preconditioner. This will be borne out by the experiments.

3.1 The SchurRAS preconditioner

The local matrix $A_{i,\delta}$ can be written in the following block form

$$A_{i,\delta} = \begin{pmatrix} B_{i,\delta} & E_{i,\delta} \\ F_{i,\delta} & C_{i,\delta} \end{pmatrix} \tag{12}$$

where $B_{i,\delta}$ is the matrix associated with interior nodes of the subdomain Ω_i^δ. Recall that interior nodes are those nodes which are coupled with unknowns from other domains. Assume that the LU or ILU factorization of $B_{i,\delta}$ is available:

$$B_{i,\delta} \approx L_{i,\delta} U_{i,\delta}$$

and consider the following block-ILU factorization of $A_{i,\delta}$,

$$\begin{aligned} A_{i,\delta} &\approx \begin{pmatrix} L_{i,\delta} & 0 \\ F_{i,\delta}U_{i,\delta}^{-1} & I \end{pmatrix} \begin{pmatrix} U_{i,\delta} & L_{i,\delta}^{-1}E_{i,\delta} \\ 0 & S_{i,\delta} \end{pmatrix} \\ &= \begin{pmatrix} L_{i,\delta} & 0 \\ F_{i,\delta}U_{i,\delta}^{-1} & I \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & S_{i,\delta} \end{pmatrix} \begin{pmatrix} U_{i,\delta} & L_{i,\delta}^{-1}E_{i,\delta} \\ 0 & I \end{pmatrix} \\ &\equiv \mathcal{L}_{i,\delta}\mathcal{D}_{i,\delta}\mathcal{U}^{i,\delta} \ . \end{aligned} \tag{13}$$

Then, the preconditioner (4) using this approximation for $A_{i,\delta}$ can be expressed as

$$\mathcal{B}_{[\delta,\delta]} = \sum_{i=1}^{p} R_{i,\delta}^T \mathcal{U}_{i,\delta}^{-1}\ \mathcal{D}_{i,\delta}^{-1}\ \mathcal{L}_{i,\delta}^{-1} R_{i,\delta}^T \ . \tag{14}$$

We can treat $\mathcal{U}_{i,\delta}, \mathcal{D}_{i,\delta}, \mathcal{L}_{i,\delta}$ as three separate operators, in which case it helps to expand each of them as an operator on the whole space. Then Equation (14) can be recast as

$$\mathcal{B}_{[\delta,\delta]} \ = \sum_{i=1}^{p} (R_{i,\delta}^T \mathcal{U}_{i,\delta}^{-1} R_{i,\delta})\ (R_{i,\delta}^T \mathcal{D}_{i,\delta}^{-1} R_{i,\delta})\ (R_{i,\delta}^T \mathcal{L}_{i,\delta}^{-1} R_{i,\delta}) \ . \tag{15}$$

Similarly, in the non-overlapping case the block-Jacobi preconditioner (5) becomes

$$\mathcal{B}_{[0,0]} = \sum_{i=1}^{p} (R_{i,0}^T \mathcal{U}_{i,0}^{-1} R_{i,0})\ (R_{i,0}^T \mathcal{D}_{i,0}^{-1} R_{i,0})\ (R_{i,0}^T \mathcal{L}_{i,0}^{-1} R_{i,0}) \tag{16}$$

while, the RAS preconditioner takes the form

$$\mathcal{B}_{[0,\delta]} \ = \sum_{i=1}^{p} (R_{i,0}^T \mathcal{U}_{i,\delta}^{-1} R_{i,\delta})\ (R_{i,\delta}^T \mathcal{D}_{i,\delta}^{-1} R_{i,\delta})\ (R_{i,\delta}^T \mathcal{L}_{i,\delta}^{-1} R_{i,\delta}) \ . \tag{17}$$

Note that in the above equations, the solves with $\mathcal{D}_{i,\delta}$, correspond to solving Schur complement systems. The operations $(R_{i,\delta}^T \mathcal{L}_{i,\delta}^{-1} R_{i,\delta})$ and $(R_{i,\delta}^T \mathcal{U}_{i,\delta}^{-1} R_{i,\delta})$ are akin to a restriction and a prolongation operator respectively. If we assume for a moment that the factorization $B_{i,\delta} = L_{i,\delta} U_{i,\delta}$ is exact, then one can expect that all that is required is an exact solution to the Schur complement system, to obtain an exact solution of the global problem. Keeping in mind the interpretation of $(R_{i,0}^T \mathcal{L}_{i,0}^{-1} R_{i,0})$ and $(R_{i,0}^T \mathcal{U}_{i,0}^{-1} R_{i,0})$, as "prolongation" and "restriction" operators, a careful look at (16) reveals that only the term $R_{i,0}^T \mathcal{D}_{i,0}^{-1} R_{i,0}$ needs to be improved to lead to an improvement to the global preconditioner. Recall that the assumption here is that the factorization of $B_{i,0}$ is exact. We can therefore focus on this operator and obtain a more exact solve by possibly using a preconditioned GMRES iteration [13, 9]. What is known about RAS can now be exploited and we arrive naturally at the following definition of SchurRAS preconditioner:

$$\mathcal{B} = \sum_{i=1}^{p} (R_{i,0}^T \mathcal{U}_{i,0}^{-1} R_{i,0})\ (R_{i,0}^T \mathcal{D}_{i,\delta}^{-1} R_{i,\delta})\ (R_{i,0}^T \mathcal{L}_{i,0}^{-1} R_{i,0}) \tag{18}$$

One may ask why we do not use the overlapping versions of the prolongation and restriction operators $(R_{i,\delta}^T \mathcal{L}_{i,\delta}^{-1} R_{i,\delta})$ and $(R_{i,\delta}^T \mathcal{U}_{i,\delta}^{-1} R_{i,\delta})$ in the above expressions. The main reason is our assumption of an accurate solve for the Schur complement system.

Comparing (17) and (18), we can see that SchurRAS and RAS have almost the same computational costs. Only the operator $R_{i,\delta}$ involves communications between neighboring processors. In particular, this means that SchurRAS and RAS involve the same communication when applying the preconditioner to a vector. According to (3), the term E_{ij} remains unchanged after the (block) Gaussian elimination process. Accordingly, the couplings of the unknowns among neighboring subdomains for the global Schur complement system are the same as those of the original linear system. In particular, when $\delta = 1$, SchurRAS and RAS involve exactly the same volume of communication.

4 Parallel Implementation

This section discusses some implementation details of the preprocessing phase in which the preconditioner is computed and the iteration phase in which the preconditioner is applied at each step.

4.1 Constructing the Preconditioning Matrix

Consider the local matrix A_i defined by (3). To compute the preconditioner, we make use of a relationship that was exploited in [13], between the Schur complement of A_i with respect to C_i, i.e., $S_i = C_i - F_i B_i^{-1} E_i$, and the block LU factorization of A_i. Specifically, we have

$$A_i = L_i U_i \equiv \begin{pmatrix} L_{i,B} & 0 \\ F_i U_{i,B}^{-1} & I \end{pmatrix} \begin{pmatrix} U_{i,B} & L_{i,B}^{-1} E_i \\ 0 & S_i \end{pmatrix} . \tag{19}$$

What is interesting is that the above factorization can be obtained via a simple variation of Gaussian Elimination referred to as the "Restricted" or "Partial" Gaussian elimination [15, 14]. The algorithm proceeds as any form of Gaussian elimination, but does not pursue the elimination in the (2,2) block of the matrix A_i. It is described next.

Algorithm **4.1** *Restricted IKJ version of Gaussian elimination*

```
1.   For i = 2, n, Do
2.       For k = 1, min(i-1,m), Do
3.           $a_{i,k} := a_{i,k}/a_{k,k}$
4.           For j = k+1, n, Do
```

```
5.               $a_{i,j} := a_{i,j} - a_{i,k} * a_{k,j}$
6.            End Do
7.         End Do
8.      End Do
```

Clearly, dropping will be used in this approximate factorization, resulting in an approximation $\tilde{S}_i$ to S_i. For clarity, we use the same symbol S_i for this approximation.

Once S_i is computed we then extend it by including rows associated with all external variables v_j that are adjacent to subdomain i. The extended matrix S'_i for the Schur complement matrix S_i has the following form

$$S'_i = \begin{pmatrix} S_i & E_{i,ext} \\ G_i & H_i, \end{pmatrix} . \tag{20}$$

In the above definition, $E_{i,ext}$ denotes simply the inter-domain coupling matrix. The matrices G_i and H_i are matrices from neighboring subdomains which act on overlapping nodes. The matrix H_i is a Schur complement corresponding to overlapping variables with respect to internal variables from other processors. Algorithm 4.1 describes the procedure for constructing the preconditioning matrix of SchurRAS.

Algorithm **4.2** *Construction of SchurRAS*

```
1.    Construct approximate local Schur complement matrix $S_i$.
2.    Extend $S_i$ to $S'_i$ by including rows corresponding to
          external variables received from neighboring processors.
3.    Compute the ILU factorization of $S'_i$.
```

It is also interesting to note that the SchurRAS procedure for two subdomains can be thought of as a variant of the global ILU factorization performed in parallel on each subdomain with different global orderings. First, S_1 and S_2 can be constructed in parallel on each processor as described in Section 2.2. This corresponds to eliminating u_1 and u_2 from the global point of view. Then, we can obtain the extended Schur complement matrices S'_1 and S'_2, which have the following forms

$$S'_1 = \begin{pmatrix} S_1 & E_{12} \\ E_{21} & S_2 \end{pmatrix}, \qquad S'_2 = \begin{pmatrix} S_2 & E_{21} \\ E_{12} & S_1 \end{pmatrix}. \tag{21}$$

Last, the ILU factorizations for S'_1 and S'_2 are performed in parallel on each processor. It is easy to understand that the whole process corresponds to a global ILU factorization performed on processor 1 with the order u_1, u_2, v_1, v_2 and with the order of u_2, u_1, v_2, v_1 on processor 2.

In contrast, the additive Schwarz preconditioner (with overlap) processes the ILU factorization for the rows associated with the overlapping variables differently: it does not take into account contributions from interior nodes into the external overlapping variables. In other words, the S_2 block in the left-side of (21) would be replaced by a matrix similar to the the C_2 matrix on processor 2 (albeit with a different ordering). Similarly for the right-side matrix S'_2.

This argument should make it clear that we expect the Schur-based preconditioner SchurRAS to outperform the additive Schwarz at least on two processors.

4.2 Applying the Preconditioner

Once the Schur complement system (8) has been computed, we could also extend the right hand side into the vector $(g'_i, g'_{i,ext})$. Here, $g'_{i,ext}$ is obtained from neighboring subdomains. The extended local equation on subdomain i for the global Schur complement system $SV = g'$ can be written as follows:

$$\begin{pmatrix} S_i & E_{i,ext} \\ G_i & H_i \end{pmatrix} \begin{pmatrix} v_i \\ v_{i,ext} \end{pmatrix} = \begin{pmatrix} g'_i \\ g'_{i,ext} \end{pmatrix}, \tag{22}$$

where $v_{i,ext}$ corresponds to the components of external variables on subdomain i, $g'_{i,ext}$ denotes components of associated right hand side vectors from neighboring subdomains.

After equation (22) is approximately solved (e.g., using an ILU factorization), we obtain v_i by simply dropping the overlapping term $v_{i,ext}$, see [3]. Finally, u_i can be computed by backsubstitution, from (11). The whole process is described in the next Algorithm.

```
ALGORITHM 4.3 Applying SchurRAS to a vector $z$
1.    Compute $g'_i = L_i^{-1} z$.
2.    Exchange interface variables in $g'_i$.
3.    Compute $v_i$ by solving equation (22)
4.    Compute $u_i$ using (11).
```

5 Numerical Experiments

This section presents a few experiments with the SchurRAS preconditioning technique. The first test is with a model test problem on a regular rectangular grid. The second considers a more realistic linear system related to a magnetohydrodynamics (MHD) problem. The experiments were performed on an SGI Origin 3800 with 64 CPUs at the Institute of Computational Mathematics and Scientific/Engineering Computing, Chinese Academy of Sciences, China.

5.1 Scaled problem: two–dimensional case

Consider the elliptic partial differential equation

$$-\Delta u + 100\frac{\partial}{\partial x}\left(e^{xy}u\right) + 100\frac{\partial}{\partial y}\left(e^{-xy}u\right) - 10u = f \tag{23}$$

on a square region with Dirichlet boundary conditions, discretized with a five-point centered finite-difference scheme on a $n_x \times n_y$ grid, excluding boundary points. The mesh is mapped to a virtual $p_x \times p_y$ grid of processors, such that a subrectangle of $r_x = n_x/p_x$ points in the x direction and $r_y = n_y/p_y$ points in the y direction is mapped to a processor. In the following experiments, the mesh-size in each processor is kept constant at $r_x = 100, r_y = 100$. As the number of processors increases, the problem size increases proportionally. For example, on 4 processors the mesh is $(2r_x, 2r_y)$ leading to a problem size of 40,000 and on 16 processors the mesh is $(4r_x, 4r_y)$ leading to a problem size of 160,000. The resulting problems become harder as the number of processors grows. In a perfectly scalable situation, the final execution time for solving the problem with size $n = 160,000$ on 64 PEs should be identical with the time for solving the problem of size $n = 40,000$ on 4 PEs. Note that in order to maintain the aspect ratio of the physical domain, we need to consider *square* processor grids of increasing size.

In the following tests, ILUT is used as the local preconditioner. The fill-in parameter is 20 and the drop tolerance is set to 0.001. Table 1 shows a comparison of preconditioners. Here, its indicates the number of iteration, t_{set} denotes the time for building the preconditioner, and t_{it} is the iteration time. BJ, RAS and SchurRAS stand for the block Jacobi, the restricted additive Schwarz , and the Schur additive Schwarz method, respectively. The amount of overlap is 1 for RAS and SchurRAS. FGMRES [12] with the subspace size of 60 was used as an accelerator in all tests. The iteration is stopped when the residual norm is reduced by a factor of 10^{-6}.

As shown in table 1, the construction phase for all three preconditioners exhibits good scalability: the time for constructing preconditioners is almost a constant for three preconditioners as the number of processors increases. It is relatively more expensive to construct the preconditioning for the SchurRAS due to the extra cost of constructing the approximate Schur complement matrix with Algorithm 4.1. If ARMS[14] is used as the local preconditioner instead of ILUT, then this additional cost can be reduced because the Schur complement matrix can be naturally obtained as a by-product of the ARMS preconditioning matrix. Table 1 also shows that when iteration counts and iteration times are considered, SchurRAS is the fastest of the three preconditioners. As the number of processors increases, the increase in the iteration counts for SchurRAS is more moderate that with the other 2 methods. The number of iterations taken by the SchurRAS on 64 processors is almost exactly half that required by RAS, which is itself almost exactly half that required by the block-Jacobi. The observations regarding iteration times are similar: The resulting iteration time for SchurRAS on 64 processors is about half that of the RAS, which is about 55% that of BJ. This means in particular, that the computational costs to apply the preconditioner to a vector is almost the same for SchurRAS and RAS. According to [16], the multiplicative Schwarz (MS) requires about half the iterations of the additive Schwarz. If this were also the case for this model problem (the operator is not quite a Laplacean) then one could say that SchurRAS and MS have a similar convergence rate.

Table 1: Comparison of preconditioners without inner iteration for a 2D PDE problem with fixed problem size. The times are in seconds.

	BJ			RAS			SchurRAS		
np	its	t_{set}	t_{it}	its	t_{set}	t_{it}	its	t_{set}	t_{it}
4	28	0.105	0.284	18	0.127	0.167	12	0.149	0.112
9	42	0.117	0.482	27	0.138	0.278	18	0.176	0.175
16	54	0.118	0.732	37	0.139	0.430	25	0.174	0.267
25	68	0.123	0.992	49	0.140	0.620	32	0.179	0.385
36	91	0.125	1.366	57	0.142	0.939	39	0.180	0.547
49	138	0.127	2.223	74	0.147	1.248	47	0.180	0.786
64	232	0.127	3.589	115	0.147	1.972	56	0.185	0.982

5.2 Using inner iterations

It is natural to ask whether or not the performance can be improved by employing an inner iteration for the Schur complement system. Specifically we can use GMRES acceleration to solve the Schur complement using SchurRAS as a preconditioner. The test problem is the same as the one in the previous section. The subspace size of GMRES is 5 and the tolerance for inner iteration is 10^{-6} (stop iteration when initial residual norm is reduced by a factor of 10^{-6}). Table 2 compares two different preconditioners. One is the block Jacobi combined with GMRES(5) for solving the Schur complement system. The corresponding induced preconditioner, denoted by Schur–BJ, was presented in [13]. The other preconditioner, denoted by Schur–SchurRAS, is the preconditioner induced by solving the Schur complement system with GMRES(5) with SchurRAS as a preconditioner. A comparison between Table 1 and Table 2, reveals that the number of iterations for SchurRAS and Schur–BJ are essentially identical. At the same time, and to our surprise, Table 2 also shows that Schur–SchurRAS also takes about the same number of iteration to converge as SchurRAS. The interpretation of what has happened is as follows. In all cases, the resulting Schur complement system is the same since it involves the same (interface) variables and the same approximations coming from the same ILUT. By iterating on the Schur complement system, this system is solved to an accuracy that has become better than the accuracy used to compute the Schur complement system itself (ILUT). While these iterations are helpful in the BJ case (they solve the Schur complement more accurately), they seem to be useless in the SchurRAS case. The only explanation for this is that SchurRAS solves the complement systems *very accurately, i.e., within the range of accuracy with which the Schur complement matrices are computed.* The end result is that the SchurRAS is still the fastest since it involves no inner iterations.

Table 2: Comparison of preconditioners with inner iteration for a 2D PDE problem with fixed problem size. The times are in seconds.

	Schur–BJ		Schur–SchurRAS	
np	its	t_{it}	its	t_{it}
4	12	0.127	12	0.137
9	19	0.223	17	0.243
16	25	0.308	24	0.354
25	31	0.414	30	0.474
36	39	0.625	38	0.561
49	47	0.885	45	0.967
64	56	1.076	54	1.167

5.3 Test with a matrix from an MHD simulation

The test matrix used in the following experiment originates from the simulation of Magneto-Hydrodynamics flows. The flow equations are represented as coupled Maxwell's and the Navier-Stokes equations. The conservative magnetohydrodynamic system is modeled by the Maxwell equations, written as:

$$\frac{\partial \mathbf{B}}{\partial t} - \nabla \times (\mathbf{u} \times \mathbf{B}) + \eta \nabla \times (\nabla \times \mathbf{B}) + \nabla \mathbf{q} = 0 \qquad (24)$$

$$\nabla \cdot \mathbf{B} = 0 \; , \qquad (25)$$

where η, $\mathbf{B}$, $\mathbf{u}$ and q are, respectively, the magnetic diffusivity coefficient, magnetic induction field, velocity field, and the scalar Lagrange multiplier for the magnetic-free divergence constraint. In fully coupled magnetohydrodynamics, this system is solved along with the incompressible Navier-Stokes equations. In the tests that follow, we solve problems which arise from the Maxwell equations only. A pre-set periodic induction field $\mathbf{u}$ is used in Maxwell's equation (24). The physical region is the three-dimensional unit cube $[-1, 1]^3$ and the discretization uses a Galerkin-Least-Squares discretization. The magnetic diffusivity coefficient is $\eta = 1$. The matrix is of size $n = n = 470, 96$ and has $nnz = 23, 784, 208$ nonzero entries.

The following input parameters have been selected to solve this problem: the residual is to be reduced by 10^6, the FGMRES restart value is 60. The fill-in parameter used for ILUT is 40 and the dropping tolerance is 0.001. SchurRAS always takes fewer iterations than RAS and BJ. Note for example that SchurRAS takes 76 iterations to converge on two processors, which is even less than the number of iterations 78 required by BJ on one processor. The execution times are 84.46s and 161.27s, respectively. The efficiency is 0.95. It is verified that SchurRAS on two processors is a variant of global ILU factorization. It takes the same number of iterations as BJ one processor, or even fewer. RAS and BJ converge in 86 and 117 iterations on two processors, respectively. On the SGI Origin 3800, this amounts to 128.18s and 94.43s, respectively. The corresponding efficiencies are 0.629 and 0.854, respectively. SchurRAS always takes fewer iterations than BJ and RAS. As the number of processors increases, the number of iterations required by SchurRAS exhibits moderate growth, as shown in Figure 2. Because the computational costs of SchurRAS are similar to those of RAS, the reduction in the iteration number also leads to a reduction in the execution time as shown in Figure 3.

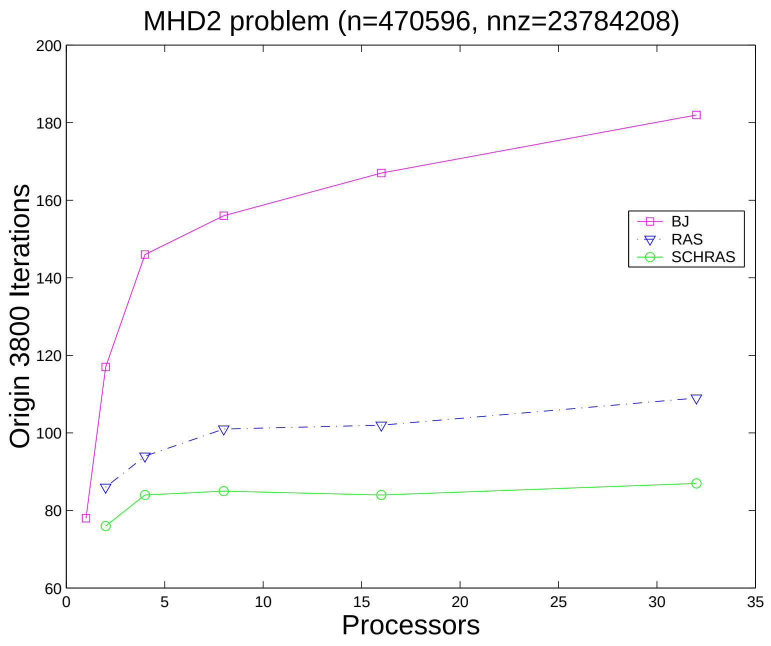

Figure 2: Iterations required by three methods for solving MHD problem

6 Conclusions

We presented a new preconditioner, called SchurRAS, which consists of solving the approximate Schur complement system with the restricted form of the additive Schwarz method. This preconditioner can be thought of as the combination of the overlapping additive Schwarz preconditioner and the non-overlapping Schur complement preconditioner. The new technique has two main advantages. First, the construction of the preconditioning matrix for SchurRAS is not only easy but also nicely scalable. This is in contrast with parallel ILU-based preconditioners or more standard Schur-complement techniques. Second, SchurRAS has the same degree of parallelism as the additive Schwarz.

The experimental tests show that SchurRAS is superior to the widely used additive Schwarz preconditioner and the Schur complement based preconditioner. For a model two dimensional scaled problem, the convergence rate of SchurRAS is similar to that of the multiplicative Schwarz preconditioner. For the same model problem and using 64 processors, we found that SchuRAS is about twice as fast as the restricted additive Schwarz,

Acknowledgements

The computing resources for this work were provided by the State Key Laboratory of Scientific and Engineering Computing (LSEC), Chinese Academy of Sciences, China. The authors would like to thank Prof. Azzeddine Soulaimani (Ecole de Technologie Superierieure, Montreal, Canada) for providing them with the MHD test example used in the experiments.

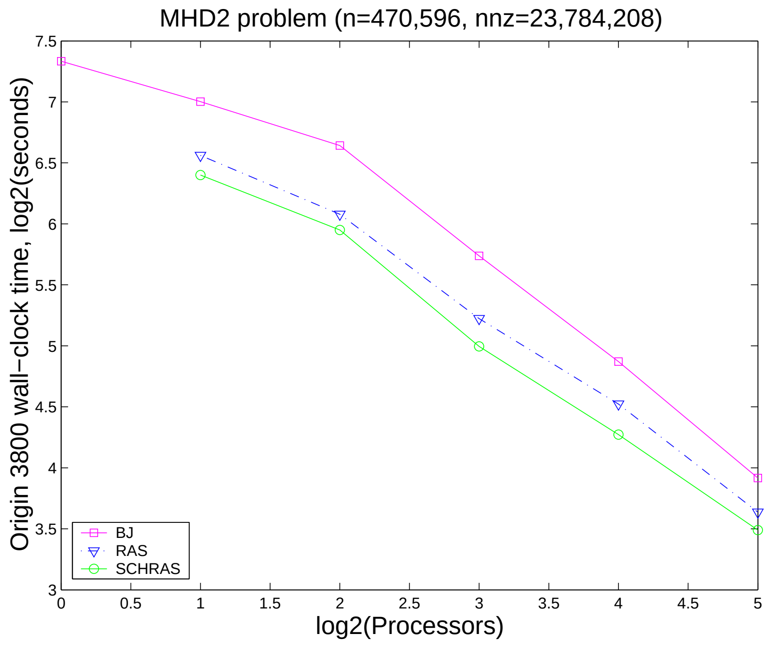

Figure 3: Origin seconds required by three methods for solving the MHD problem

References

[1] M. Benzi, W. D. Joubert, and G. Mateescu. Numerical experiments with parallel orderings for ILU preconditioners. *Electron. Trans. Numer. Anal.*, 8:88, 1999.

[2] X.-C. Cai and Y. Saad. Overlapping domain decomposition algorithms for general sparse matrices. *Numerical Linear Algebra with Applications*, 3(3):221–237, 1996.

[3] X.-C. Cai and M. Sarkis. A restricted additive Schwarz preconditioner for general sparse linear systems. *SIAM Journal on Scientific Computing*, 21(2):792–797, 1999.

[4] T. F. Chan and D. Goovaerts. A note on the efficiency of domain decomposed incomplete factorizations. *SIAM Journal on Scientific and Statistic Computing*, 11:794–803, 1990.

[5] G. Radicati di Brozolo and Y. Robert. Parallel conjugate gradient-like algorithms for solving sparse non-symmetric systems on a vector multiprocessor. *Parallel Computing*, 11:223–239, 1989.

[6] D. Hysom and A. Pothen. A scalable parallel algorithm for incomplete factor preconditioning. *SIAM Journal on Scientific Computing*, 22(6):2194–2215, 2001.

[7] G. Karyps and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.

[8] G. Karyps and V. Kumar. Parallel thereshold–based ILU factorization. Technical report, University of Minnesota, Department of Computer Science/Army HPC research Center, Minneapolis, MN 55455, 1998.

[9] Z. Li, Y. Saad, and M. Sosonkina. pARMS: a parallel version of the algebraic recursive multilevel solver. *Numerical Linear Algebra with Applications*, 10(5–6):485–509, 2003.

[10] S. Ma and Y. Saad. Distributed ILU(0) and SOR preconditioners for unstructured sparse linear systems. Technical Report ahpcrc-94-027, Army High Performance Computing Research Center, University of Minnesota, Minneapolis, MN, 1994.

[11] O. Yu. Milyukova. Parallel approximate factorization method for solving discrete elliptic equations. *Parallel Computing*, 27(10):1365–1379, 2001.

[12] Y. Saad. *Iterative Methods for Sparse Linear Systems.* Society for Industrial & Applied Mathematics, April 2003.

[13] Y. Saad and M. Sosonkina. Ddistributed Schur complement techniques for general sparse linear system. *SIAM Journal on Scientific Ccomputing*, 21(4):1337–1356, 1999.

[14] Y. Saad and B. Suchomel. ARMS: An algebraic recursive multilevel solver for genral sparse linear systems. *Numerical Linear Algebra with Applications*, 9(5):359–378, 2002.

[15] Y. Saad and J. Zhang. BILUTM: A domain–based multi–level block ILUT preconditoiner for general sparse matrixes. *SIAM Journal On Matrix Analysis and Applications*, 21(1):279–299, 2000.

[16] B. Smith, P. Bjørstad, and W. Gropp. *Domain Decomposition: Parallel Multilevel Methods for Elliptic partial Differentail Equations.* Cambridge University Press, 1996.

[17] H. A. van der Vorst. *Iterative Krylov Methods for Large Linear Systems.* Cambridge University Press, 2003.