Parallel Two Level ILU Preconditioning Techniques for Large Sparse Linear Systems *

Chi Shen†

Laboratory for High Performance Scientific Computing and Computer Simulation,
Department of Computer Science, University of Kentucky,
773 Anderson Hall, Lexington, KY 40506-0046, USA

Abstract

We discuss issues related to domain decomposition and multilevel preconditioning techniques in parallel computations. We implement a parallel preconditioner for solving general sparse linear systems based on a two level block ILU factorization strategy. We give some new data structures and strategies to construct a local coefficient matrix and a local Schur complement matrix on each processor. The preconditioner constructed is fast and robust for solving certain large sparse matrices. Numerical experiments show that our domain based two level block ILU preconditioners are more robust and more efficient than some published ILU preconditioners based on Schur complement techniques for parallel sparse matrix solutions.

Key words: Parallel preconditioning, sparse matrices, Schur complement techniques.

1 Introduction

High performance parallel computing techniques have undergone a gradual maturation process in recent years. Developing efficient numerical linear algebra algorithms specifically aiming at high performance parallel computers becomes a challenging issue.

A well known approach in parallel preconditioning techniques for general sparse linear systems is to use ideas from domain decomposition concepts in which a processor is assigned a certain number of rows of the linear system to be solved. For discussions related to this point of view and comparisons of different domain decomposition strategies, see [1, 5] and the references therein. Some techniques in this class include various distributed Schur complement methods for solving general sparse linear systems developed in [8].

Recently, a class of high accuracy preconditioners (BILUM and BILUTM) that combine the inherent parallelism of domain decomposition, the robustness of ILU factorization, and the scalability potential of multigrid method have been developed in [10, 11]. They have been tested to show promising convergence rate and scalability for solving certain problems. The construction

*This research was supported in part by the U.S. National Science Foundation under grants CCR-9902022, CCR-9988165, and CCR-0092532.

†E-mail: cshen@cs.uky.edu.

of these preconditioners are based on block independent set ordering and recursive block ILU factorization with Schur complements. Although this class of preconditioners contain obvious parallelism within each level, their parallel implementations have not yet been reported.

In this study, the BILUTM preconditioner of Saad and Zhang [11] is modified to be implemented as a two level block ILU preconditioner on distributed memory parallel architectures (PBILU2). We used Saad's PSPARSLIB library [1] with MPI as basic communication routines. Our PBILU2 preconditioner is compared with one of the most favorable Schur complement based preconditioners of [8] in a few numerical experiments.

This article is organized as follows. In Section 2, we outline the distributed representations of general sparse linear systems. In Section 3, we discuss the construction of a preconditioner (PBILU2) based on two level block ILU factorization. Numerical experiments with a comparison of two Schur complement based preconditioners for solving various distributed linear systems are presented in Section 4 to demonstrate the merits of our two level block ILU preconditioner. Concluding remarks are given in Section 5.

2 Distributed Sparse Linear System and SLU Preconditioner

Let us consider a general sparse linear system of the form

$$Ax = b, \tag{1}$$

where A is an unstructured real-valued matrix of order n.

A distributed sparse linear system is a collection of sets of equations that is assigned to different processors. A type of distributed matrix data structure based on subdomain decomposition concepts has been proposed in [7, 9]. Based on these concepts, the matrix A is assigned to each processor, the unknowns in each processor are divided into three types: (1) interior unknowns that are coupled only with local equations; (2) local interface unknowns that are coupled with both nonlocal (external) and local equations; and (3) external interface unknowns that belong to other subdomains and are coupled with local equations. The submatrix assigned to a certain processor, say, processor i, is split into two parts: the local matrix A_i, which acts on the local variables, and an interface matrix X_i, which acts on the external variables. Accordingly, the local equations on a given processor can be written as

$$A_i x_i + X_i y_{i,ext} = b_i.$$

The local matrix is reordered in such a way that the interface points are listed last after the interior points. Then we have a local system written in a block format

$$\begin{pmatrix} B_i & F_i \\ E_i & C_i \end{pmatrix} \begin{pmatrix} u_i \\ y_i \end{pmatrix} + \begin{pmatrix} 0 \\ \sum_{j \in N_i} E_{ij} y_j \end{pmatrix} = \begin{pmatrix} f_i \\ g_i \end{pmatrix}, \tag{2}$$

where N_i is the indices of subdomains that are neighbors to the reference subdomain i. N_i is exactly the index set of processors that the reference processor needs to communicate with to receive information. The term $E_{ij} y_j$ is a part of the product $X_i y_{i,ext}$ which reflects the contribution to

[1]The PSPARSLIB library is available online from `http://www.cs.umn.edu/Research/arpa/p_sparslib/psp-abs.html`.

the local equation from the neighboring subdomain j. The preconditioners which are built upon this distributed data structure will not form an approximation to the global Schur complement explicitly. Such domain decomposition based preconditioners are exploited in [8]. One of the best among these Schur complement based preconditioners is SLU which is the distributed approximate Schur LU preconditioner [8]. Numerical results reported in [8] show that this Schur (I)LU preconditioner demonstrates superior scalability performance over block Jacobi preconditioner and is more efficient than the latter in terms of parallel run time.

3 A Two Level Block ILU Preconditioner

PBILU2 is a two level block ILU preconditioner based on the BILUTM techniques described in [11]. As we noted before, BILUTM offers a good parallelism and robustness due to its large size block independent set. The graph partitioner in BILUTM is the greedy algorithm for finding a block independent set [10, 11].

3.1 Distributed matrix based on block independent set

Assume that a block independent set with a uniform block size k has been found, and the coefficient matrix A is permuted into a block form:

$$\tilde{A} = PAP^T = \begin{pmatrix} B & F \\ E & C \end{pmatrix}, \tag{3}$$

where P is a permutation matrix. At the same time, the global vector of unknowns x is split into two subvectors $(u,y)^T$,where $u = (u_1, \ldots, u_m)^T$, $y = (y_1, \ldots, y_m)^T$. The right hand side vector b is also conformally split into subvectors f and g. Such a reordering leads to a block system

$$\left(\begin{array}{cccc|c} B_1 & & & & F_1 \\ & B_2 & & & F_2 \\ & & \ddots & & \vdots \\ & & & B_m & F_m \\ \hline & & E_1 & & C_1 \\ & & E_2 & & C_2 \\ & & \vdots & & \vdots \\ & & E_m & & C_m \end{array}\right) \left(\begin{array}{c} u_1 \\ u_2 \\ \vdots \\ u_m \\ \hline y_1 \\ y_2 \\ \vdots \\ y_m \end{array}\right) = \left(\begin{array}{c} f_1 \\ f_2 \\ \vdots \\ f_m \\ \hline g_1 \\ g_2 \\ \vdots \\ g_m \end{array}\right), \tag{4}$$

in which m is the number of processors used in the computation. Each block diagonal submatrix $B_i, i = 1, \ldots, m$, contains several independent blocks.

Those submatrices B_i, F_i, E_i, C_i, f_i and g_i are assigned to the same processor i. u_i and y_i are the local part of the unknown vectors, respectively. When this processor-data assignment is done, each processor holds several rows of the equations. The local system of equations in processor i can be written as:

$$\left\{ \begin{array}{rcl} B_i u_i + F_i y & = & f_i, \\ E_i u + C_i y & = & g_i, \end{array} \right. \tag{5}$$

An obvious difference between the partitionings (2) and (5) is that in (5), the action of F_i is not completely local, while it is local in (2). This viewpoint of local matrix facilitates the construction of Schur complement matrix efficiently using parallel restricted Gaussian elimination in Section 3.2.

3.2 Derivation of Schur complement techniques in parallel

A key idea in domain decomposition techniques is to develop preconditioners for the global system (1) by exploiting methods that approximately solve the Schur complement system in parallel. For deriving the global Schur complement, other parts of the submatrix of coefficient matrix E need to be partitioned and sent to a certain processor. There are two ways to partition the submatrix E: one is to partition E by rows and the other is by columns. That is

$$E = \begin{pmatrix} M_1 & M_2 & \cdots & M_m \end{pmatrix} = \begin{pmatrix} E_1 \\ E_2 \\ \vdots \\ E_m \end{pmatrix}.$$

Those submatrices $M_i, i = 1, \ldots, m$, will also be assigned to the processor i.

We now consider a block LU factorization of (3) in the form of

$$\begin{pmatrix} B & F \\ E & C \end{pmatrix} = \begin{pmatrix} I & 0 \\ EB^{-1} & I \end{pmatrix} \begin{pmatrix} B & F \\ 0 & S \end{pmatrix}, \tag{6}$$

where S is the global Schur complement:

$$S = C - EB^{-1}F. \tag{7}$$

Suppose that we can invert B by some means, we may rewrite Equation (7) as:

$$S = \begin{pmatrix} C_1 \\ \vdots \\ C_m \end{pmatrix} - \sum_{i=1}^{m} M_i B_i^{-1} F_i. \tag{8}$$

On the ith processor, a local submatrix A_i is formed based on those submatrices assigned to this processor and an ILU factorization on this local matrix will be performed. [2] This local matrix on processor i looks like

$$A_i = \begin{pmatrix} B_i & F_i \\ M_i & \bar{C}_i \end{pmatrix} = \left(\begin{array}{c|c} B_i & F_i \\ \hline & 0 \\ & \vdots \\ M_i & C_i \\ & \vdots \\ & 0 \end{array}\right). \tag{9}$$

We perform a restricted Gaussian elimination on the local matrix (9). First we perform an (I)LU factorization (Gaussian elimination) to the upper part of the local matrix, i.e., to the submatrix (B_i, F_i). We then continue the Gaussian elimination to the lower part $(M_i, \bar{C}_i)$, but the elimination is only performed with respect to nonzero (and the accepted fill-in) entries of the submatrix M_i. The entries in $\bar{C}_i$ are modified accordingly. When performing these operations on the lower part, the upper part of the matrix is only accessed, but not modified, see Figure 1.

[2] In PBILU2, the "local" matrix A_i means it is stored on the local processor i. It does not necessarily mean that A_i acts only on interior unknowns.

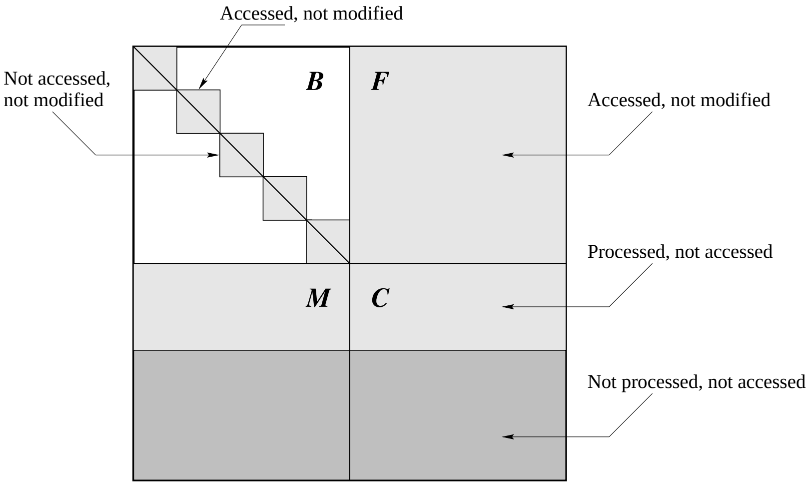

Figure 1: Illustration of the restricted IKJ version of Gaussian elimination of the lower part. Here the submatrices B, F, M, C represent the local submatrices B_i, F_i, M_i and $\bar{C}_i$, respectively, as in Equation (9).

After this is done, we obtain a new reduced submatrix $\tilde{C}_i$ and it forms a piece of the global Schur complement matrix. If the Gaussian elimination is an exact factorization, the global Schur complement matrix can be formed by summing all these submatrices $\tilde{C}_i$ together. That is

$$S = \tilde{C}_1 + \tilde{C}_2 + \cdots + \tilde{C}_m. \tag{10}$$

Each submatrix $\tilde{C}_i, i = 1, \ldots, m$, is partitioned into m parts (using the same partitionings as the original submatrix C), and the corresponding parts are scattered to relevant processors. After receiving and summing all of those parts of submatrices which have been scattered from different processors, the "local" Schur complement matrix S_i is formed. Here "local" means some rows of the global Schur complement that are held in a given processor. Note that $S_i \neq \tilde{C}_i$.

3.3 Induced global preconditioner

It is possible to develop preconditioners for the global system (1) by exploiting methods that approximately solve the reduced system (7). Considering the block LU factorization in the Equation (6), this block factored matrix can be preconditioned by replacing S in (5) by $\tilde{S}$, where $\tilde{S}$ is an approximation to the global Schur complement matrix S, formed in (10). Therefore, an approximate reduced system is $\tilde{S}y = \tilde{g}$, i.e.,

$$\begin{pmatrix} \tilde{S}_1 & X_{12} & \cdots & X_{1m} \\ X_{21} & \tilde{S}_2 & \cdots & X_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ X_{m1} & X_{m2} & \cdots & \tilde{S}_m \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} = \begin{pmatrix} \tilde{g}_1 \\ \tilde{g}_2 \\ \vdots \\ \tilde{g}_m \end{pmatrix}, \tag{11}$$

where the submatrix X_{ij} is a boundary matrix which acts on external variables $y_j, j \neq i$. $\tilde{g}_i$ is the local (Schur complement system) right hand side which can be computed as

$$\tilde{g}_i = g_i - E_i \begin{pmatrix} B_1^{-1} f_1 \\ \vdots \\ B_m^{-1} f_m \end{pmatrix}.$$

There are numerous ways to solve this reduced system. One option considered in [8] and adopted in our implementation starts with replacing

$$\tilde{S}_i y_i + \sum_{j \in N_i} X_{ij} y_j = \tilde{g}_i$$

by an approximate system of the form

$$y_i + \bar{S}_i^{-1} \sum_{j \in N_i} X_{ij} y_j = \bar{S}_i^{-1} \tilde{g}_i,$$

in which $\bar{S}_i$ is a local approximation to the local Schur complement matrix $\tilde{S}_i$. This formulation can be viewed as a block Jacobi preconditioned version of the Schur complement system (11). The above system is then solved by an iterative accelerator such as GMRES which requires a solution with $\bar{S}_i$ at each step. In our current implementation, an ILUT factorization of $\tilde{S}_i$ is performed for the purpose of block Jacobi preconditioning.

4 Numerical Experiments

In numerical experiments, we compared the performance of the PBILU2 preconditioner we just described and the distributed Schur complement LU (SLU) preconditioner of [8] for solving a few sparse matrices from discretized two dimensional convection diffusion problems, and from application problems in computational fluid dynamics.

The computations were carried out on a 32 processor (200 MHz) subcomplex of a (64 processor) HP Exemplar 2200 (X-Class) supercomputer at the University of Kentucky. Many of the communication subroutines and the SLU preconditioner code were taken from the PSPARSLIB library [6].

In all tables containing numerical results, "n" denotes the dimension of the matrix; "nnz" represents the number of nonzeros in the sparse matrix; "np" is the number of processors used; "iter" is the number of preconditioned FGMRES iterations (outer iterations); "F-time" is the CPU time in seconds for the preconditioned solution process with FGMRES; "P-time" is the total CPU time in seconds for solving the given sparse matrix, starting from the initial distribution of matrix data to each processor from the master processor (processor 0). "S-ratio" stands for the sparsity ratio, which is the ratio between the number of nonzeros in the preconditioner and the number of nonzeros in the original matrix A. k is the block size used in PBILU2, "p" is the number of nonzeros allowed in each of the L and U factors of the ILU factorizations, τ is the drop tolerance. p and τ have the same meaning as those used in Saad's ILUT [7].

4.1 5-POINT and 9-POINT matrices

We first compared the parallel performance of different preconditioners for solving some 5-POINT and 9-POINT matrices. Those matrices were generated by discretizing the following convection

Table 1: 5-POINT matrix: Re $= 0$, $n = 511^2$, $nnz = 1303561$, $k = 200$, $p = 30, \tau = 10^{-4}$. One level overlapping (nonoverlapping for results in brackets) for SLU.

Preconditioner	np	iter	F-time	P-time	S-ratio
PBILU2	4	38	86.56	114.25	5.00
SLU		36 (40)	114.14	144.29	10.50 (10.55)
PBILU2	8	37	41.45	56.07	5.00
SLU		39 (42)	64.11	82.34	10.20 (10.26)
PBILU2	16	35	20.34	28.77	4.97
SLU		42 (46)	36.11	51.78	9.50 (9.65)
PBILU2	24	32	14.83	21.16	4.99
SLU		44 (50)	31.47	51.59	9.15 (9.25)
PBILU2	32	31	13.10	18.03	5.03
SLU		46 (56)	21.49	59.79	8.90 (9.02)

Table 2: 9-POINT matrix: Re $= 10^3$, $n = 600^2$, $nnz = 3232804$. $k = 200$, $p = 30, \tau = 10^{-4}$. One level overlapping for SLU.

Preconditioner	np	iter	F-time	P-time	S-ratio
PBILU2	4	14	36.78	62.88	2.30
SLU		13	43.87	124.80	4.50
PBILU2	8	14	20.74	35.34	3.17
SLU		15	34.36	91.80	4.52
PBILU2	16	13	12.81	21.77	2.30
SLU		15	17.97	45.61	4.51
PBILU2	24	14	10.17	17.32	2.30
SLU		19	16.76	46.66	4.50
PBILU2	32	14	10.76	17.52	2.33
SLU		19	24.31	151.24	4.51

diffusion equation using the standard 5-point central difference scheme or the fourth order 9-point difference scheme [3]

$$u_{xx} + u_{yy} + \mathrm{Re}\left[\exp(xy-1)\,u_x - \exp(-xy)\,u_y\right] = f(x,y),$$

on a two dimensional unit square. Here Re is the so-called Reynolds number. The right hand side function was not used since we generated artificial right hand sides for the sparse linear systems as stated above.

We first tested 5-POINT matrix for two preconditioners PBILU2 and SLU. For SLU, we tested two cases: one level overlapping of subdomains and nonoverlapping of subdomains. The test results are listed in Table 1.

Another set of tests were run for solving the 9-POINT matrix listed in Table 2 and Figure 2. Since the overlapping version of SLU converged faster than the nonoverlapping version, we only report results with overlapping version of SLU in the remaining numerical tests.

In our experiments, we found that our PBILU2 preconditioner is faster and more efficient than the SLU preconditioner of [8] for solving those problems. Figure 2 also indicates that the convergence rate of PBILU2 behaved better than that of SLU as the number of processors increased.

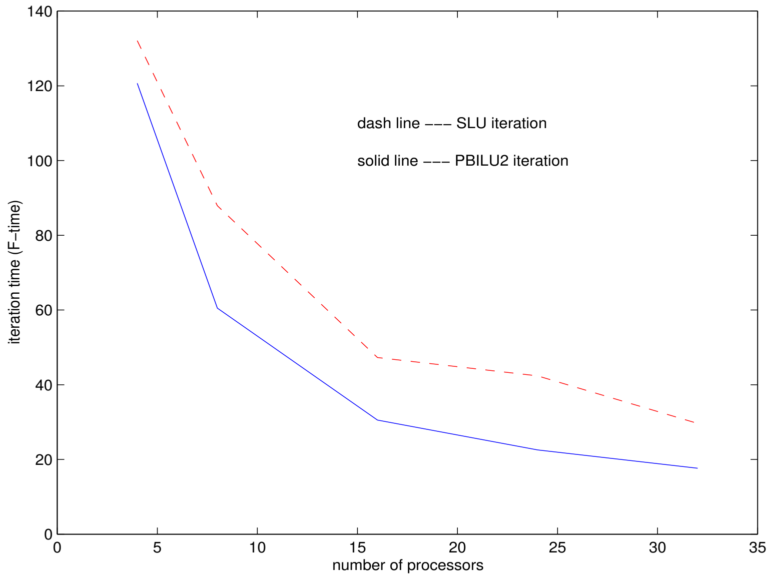

Figure 2: Comparison of parallel iteration time (F-time) for the PBILU2 and SLU preconditioners for solving a 9-POINT matrix with Re = 0 and $n = 511^2$. Parameters used are $p = 30, \tau = 10^{-4}$, and $k = 200$.

4.2 FIDAP matrices and Navier-Stokes matrices

This set of test matrices were extracted from the test problems provided in the FIDAP package [2]. [3] As many of these matrices have small or zero diagonals, they are difficult to solve with standard ILU preconditioners. We tested more than 31 FIDAP matrices for both preconditioners. We found that PBILU2 can solve more than twice as many FIDAP matrices as SLU does. In our tests, PBILU2 solved 20 FIDAP matrices and SLU solved 9. These tests show that our parallel two level block ILU preconditioner is more robust than the SLU preconditioner.

Table 3: FIDAPM29 matrix, $n = 13668$, $k = 400$, $nnz = 186294$

Preconditioner	np	p	τ	iter	F-time	P-time	S-ratio
PBILU2	2	150	10^{-10}	11	6.39	20.61	4.35
SLU		300	10^{-10}	25	22.54	47.45	7.8
PBILU2	4	150	10^{-10}	12	3.21	10.17	4.28
SLU		300	10^{-10}	–			
PBILU2	8	150	10^{-10}	23	2.64	6.49	4.11
SLU		300	10^{-10}	211	81.49	92.80	7.84
PBILU2	16	150	10^{-10}	164	10.19	12.44	3.81
SLU		300	10^{-10}	138	26.69	29.99	7.01
PBILU2	24	150	10^{-10}	292	15.15	16.86	3.81
SLU		300	10^{-10}	–			

Note that "–" in Table 3 means that the preconditioned iterative method did not converge

[3] These matrices are available online from the MatrixMarket of the National Institute of Standards and Technology at `http://math.nist.gov/MatrixMarket`.

Table 4: S30a matrix, $n = 35411$, $nnz = 5558521$. $p = 500, \tau = 10^{-10}$.

Preconditioner	np	k	iter	S-ratio
PBILU2	8	800	6	2.56
	16	800	13	2.39
	24	400	8	2.05
	32	400	9	1.87

or the number of iterations is greater than 500. We varied the parameters of fill-in (p) and drop tolerance (τ) in Table 3 for both preconditioners and adjusted the size of block independent set for the PBILU2 approach. PBILU2 is clearly shown to be more robust than SLU to solve this FIDAP matrix.

The Navier-Stokes matrices are from fully coupled mixed finite element discretization of three dimensional Navier-Stokes equations [12]. S30a means that the matrix is from the first Newton step of the nonlinear iterations, with 30 elements in each of the x and y coordinate directions, and 1 element in the z coordinate direction. There is only one element in the z coordinate direction because of the limitation on the computer memory used to generate these matrices. The same explanation holds for the S10a matrix. We tested S10a and S30a matrices. PBILU2 was able to solve these CFD matrices with small τ values. SLU did not converge for these test problems. The test result for matrix S30a is listed in Table 4.

5 Concluding Remarks

We have implemented a parallel two level block ILU preconditioner based on a Schur complement preconditioning. We discussed the details on the distribution of "small" independent blocks to form a subdomain in each processor. We gave a computational procedure for constructing a distributed Schur complement matrix in parallel. We compared our parallel preconditioner, PBILU2, with a scalable parallel two level Schur LU preconditioner (SLU) published recently. Numerical experiments show that PBILU2 demonstrates good scalability in solving large sparse linear systems on parallel computers. We also found that PBILU2 is faster and computationally more efficient than SLU in most of our test cases. PBILU2 is also efficient in terms of memory consumption, since it uses less memory space than SLU to achieve better convergence rate.

We plan to extend our parallel two level block ILU preconditioner to truly parallel multilevel block ILU preconditioners. The key problem in this future is to develop parallel graph partitioning algorithms to find block independent set efficiently. The ideas used in parallel algorithms to find maximum independent set will be examined [4].

Acknowledgments: The author would like to thank Professor Jun Zhang for his guidance and encouragement during the course of this research work and the U.S. National Science Foundation for supporting this research project.

References

[1] X.-C. Cai, W. D. Gropp, and D. E. Keyes. A comparison of some domain decomposition and ILU preconditioned iterative methods for nonsymmetric elliptic problems. *Numer. Linear Algebra Appl.*, 1(5):477–504, 1994.

[2] M. Engelman. FIDAP: Examples Manual, Revision 6.0. Technical report, Fluid Dynamics International, Evanston, IL, 1991.

[3] M. M. Gupta, R. P. Manohar, and J. W. Stephenson. A single cell high order scheme for the convection-diffusion equation with variable coefficients. *Int. J. Numer. Methods Fluids*, 4:641–651, 1984.

[4] G. Karypis and V. Kumar. Parallel multilevel k-way partitioning scheme for irregular graphs. *SIAM Rev.*, 41(2):278–300, 1999.

[5] D. E. Keyes and W. D. Gropp. A comparison of domain decomposition techniques for elliptic partial differential equations and their parallel implementation. *SIAM J. Sci. Statist. Comput*, 8(2):S166–S202, 1987.

[6] Y. Saad. Parallel sparse matrix library (P_SPARSLIB): The iterative solvers module. In *Advances in Numerical Methods for Large Sparse Sets of Linear Equations*, volume Number 10, Matrix Analysis and Parallel Computing, PCG 94, pages 263–276, Yokohama, Japan, 1994. Keio University.

[7] Y. Saad. *Iterative Methods for Sparse Linear Systems.* PWS Publishing, New York, NY, 1996.

[8] Y. Saad and M. Sosonkina. Distributed Schur complement techniques for general sparse linear systems. *SIAM J. Sci. Comput.*, 21(4):1337–1356, 1999.

[9] Y. Saad and K. Wu. Design of an iterative solution module for a parallel sparse matrix library (P_SPARSLIB). In W. Schonauer, editor, *Proceedings of IMACS Conference, 1994*, Georgia, 1995.

[10] Y. Saad and J. Zhang. BILUM: block versions of multielimination and multilevel ILU preconditioner for general sparse linear systems. *SIAM J. Sci. Comput.*, 20(6):2103–2121, 1999.

[11] Y. Saad and J. Zhang. BILUTM: a domain-based multilevel block ILUT preconditioner for general sparse matrices. *SIAM J. Matrix Anal. Appl.*, 21(1):279–299, 1999.

[12] J. Zhang, G. F. Carey, R. McLay, and B. Barth. Performance of ILU preconditioners for stationary 3D Navier-Stokes simulation. Technical Report in preparation, Department of Computer Science, University of Kentucky, Lexington, KY, 2001.