A Scalable Balancing Domain Decomposition Based Preconditioner for Large Scale Heat Transfer Problems

Mukaddes, A.M.M.
Department of Intelligent Machinery and Systems, Faculty of Engineering, Kyushu University

Mukaddes, Abul Mukid Mohammad

Ogino, Masao
Department of Intelligent Machinery and Systems, Faculty of Engineering, Kyushu University

Kanayama, Hiroshi
Department of Intelligent Machinery and Systems, Faculty of Engineering, Kyushu University

他

https://hdl.handle.net/2324/5654

出版情報：JSME International Journal Series B. 49 (2), pp.533-540, 2006-05. The Japan Society of Mechanical Engineers
バージョン：
権利関係：

A Scalable Balancing Domain Decomposition Based Preconditioner for Large Scale Heat Transfer Problems

A.M.M.MUKADDES*, Masao OGINO**, Hiroshi KANAYAMA**
and Ryuji SHIOYA**

* Department of Intelligent Machinery and Systems, Graduate School of Engineering, Kyushu University, 744, Motooka, Nishi-ku, Fukuoka 819-0395, Japan
Email: mukaddes@cm.mech.kyushu-u.ac.jp
** Department of Intelligent Machinery and Systems, Faculty of Engineering, Kyushu University, 744, Motooka, Nishi-ku, Fukuoka 819-0395, Japan
Email: ogino@cm.mech.kyushu-u.ac.jp,
kanayama@mech.kyushu-u.ac.jp,
shioya@mech.kyushu-u.ac.jp

***Key Words**:* Heat Conduction, Balancing Domain Decomposition, Incomplete Cholesky Factorization, Parallel Processing

An efficient and scalable Balancing Domain Decomposition (BDD) type preconditioner for large scale linear systems arising from 3-dimensional heat transfer problems is presented. The new method improves parallel scalability of BDD by employing an incomplete balancing technique to approximate a coarse space problem and a diagonal scaling to precondition the local fine space problems instead of the Neumann-Neumann preconditioner. It may increase the number of iterations but reduces the computation costs of the precondition process for each iteration. Consequently, total computation time and required memory are expected to be reduced. The convergence estimates may also be independent of the number of subdomains. We have implemented this algorithm on the parallel processors and have succeeded in solving some ill-conditioned large scale heat transfer problems.

1. Introduction

With the growth of parallel computers, domain decomposition method has become increasingly popular for the solution of large scale linear systems arising from 3-dimensional heat transfer problems. Indeed, domain decomposition method is simpler to implement on most parallel computational platforms. In general, subdomain (or substructure) equations are solved using a direct method, while an iterative scheme is applied to the solution of the interface problem(1). When the interface problem is solved iteratively, usually via a preconditioned Conjugate Gradient (CG) algorithm, the overall domain decomposition method becomes a genuine iterative solver whose success hinges on two important properties: *numerical scalability* and *parallel scalability*. A domain decomposition method based on an iterative scheme is said to be numerically scalable if after preconditioning the number of iterations does not grow with the number of subdomains(2). It is well known that in order to achieve the numerical scalability, the domain decomposition method must involve a coarse problem with a few degrees of freedom per subdomain, which must be solved at each iteration to accelerate the convergence. In practice, numerical scalability is most interesting when parallel scalability can also be achieved. Parallel scalability characterizes the ability of an algorithm to deliver larger speed-ups for large scale problems using a larger number of processors (2).

As an efficient domain decomposition preconditioner for the solution of algebraic systems from the approximation of a partial differential equation problem, Balancing Domain Decomposition (BDD) has received much attention in the last few years. This method is introduced by Mandel(3) by adding a coarse problem to an earlier method of De Roeck and Le Tallec(4) known as the Neumann-

Neumann method and then is extended for problems with large jumps in coefficients(5). Mandel's BDD algorithm is now exclusively employed for the solution of huge structural problems(6), Stokes problems(7), semiconductor simulation(8) and plate and shell problems(9). While this method shows good performances to solve large scale elastic problems(10), it still suffers from the high computation costs and the large required memory to solve the 3-dimensional heat conductive problems(11). Recently the BDD preconditioner with an incomplete coarse operator is proposed in the thermal-solid coupling analysis(12, 13). In this paper the improved BDD preconditioner(12, 13) called Incomplete Balancing Domain Decomposition with DIAGonal Scaling (IBDD-DIAG) is thoroughly presented and investigated in 3-dimensional heat transfer problems. In addition, we demonstrate the effective scalability of the domain decomposition method with the new preconditioner.

The original BDD algorithm uses at each CG iteration the solutions of local Neumann-Neumann problems on the subdomains coupled with a coarse problem in a coarse space. In the Neumann-Neumann problems, generalized inverse matrices are required such as the Moore-Penrose pseudoinverse or the regularized inverse, since they are typically singular. However the Moore-Penrose pseudo-inverse takes high computation costs, while the regularized inverse is less accurate. To overcome the issues of memory shortage and computation time, we choose a simplified diagonal scaling as a local subdomain correction. Again the degrees of freedom of coarse problems are related to the number of subdomains, and then it is difficult to solve them for large scale problems. The improved method employs a new coarse problem solver based on an incomplete parallel Cholesky factorization. This IBDD-DIAG preconditioning technique reduces the computation time and improves parallel efficiency, but may increase the number of iterations. Consequently, total computation time and required memory are expected to be reduced.

All the algorithms discussed have been implemented in parallel codes that have been successfully tested on large scale problems in massively parallel computers. Numerical results show that the convergence of the improved method is almost independent of the number of subdomains and show an excellent efficiency in large scale problems.

This paper is organized as follows. In section 2, we briefly describe the class of problems considered. The substructuring process is explained in section 3. In section 4 the BDD algorithm is recalled. An improved system of the BDD algorithm is presented in section 5. Finally, computational results illustrating the performances of the improved method are discussed in section 6.

2. Problems and Discretizations

We consider a heat conductive problem in a domain Ω. Defining $\overline{f}$ as internal heat generation, $\overline{u}$ temperature applied on the boundary Γ_u, $\overline{Q}$ the heat flux applied on the boundary Γ_Q and $\alpha_c(u-u_c)$ convection heat transfer on the boundary Γ_c, the fundamental equations of this heat conductive problem are given by:

$$\begin{cases} q=-k\,grad\,u & in\ \Omega \\ divq=\overline{f} & in\ \Omega \\ q\cdot n=\overline{Q} & on\ \Gamma_Q \\ q\cdot n=\alpha_c(u-u_c) & on\ \Gamma_c \\ u=\overline{u} & on\ \Gamma_u \end{cases} \qquad (1)$$

where u is temperature, q the heat flux, k the thermal conductivity, α_c the convective heat transfer coefficient, u_c the external temperature and n an outer normal unit vector, respectively. The finite element (quadratic tetrahedral) discretization of (1) yields a linear system of the form

$$Ku = f \tag{2}$$

where K is the global stiffness matrix, u is an unknown vector of temperature and f is a known vector.

3. Substructuring (Reduction to the Interface Problem)

The domain Ω is decomposed into N non-overlapping subdomains, $\{\Omega_i\}_{i=1,\dots,N}$. As usual the global stiffness matrix K can be generated by subassembling:

$$K = \sum\nolimits_{i=1}^{N} R^{(i)} K^{(i)} R^{(i)T} \tag{3}$$

where $R^{(i)T}$ is the 0-1 matrix which translates the global indices of the nodes into local numbering. With superscript ^{T} we denote the transpose of a matrix (or vector). Let $u^{(i)}$ be the vector corresponding to the elements in $\Omega^{(i)}$ and it can be expressed as $u^{(i)} = R^{(i)T} u$. Each $u^{(i)}$ is split into degrees of freedom $u_B^{(i)}$, which correspond to $\partial\Omega^{(i)}$, called interface degrees of freedom and the remaining interior degrees of freedom $u_I^{(i)}$. The subdomain matrix $K^{(i)}$, the vector $u^{(i)}$ and the 0-1 matrices are then split accordingly:

$$K^{(i)} = \begin{pmatrix} K_{II}^{(i)} & K_{IB}^{(i)} \\ K_{IB}^{(i)T} & K_{BB}^{(i)} \end{pmatrix}, \tag{4}$$

$$u^{(i)} = \begin{pmatrix} u_I^{(i)} \\ u_B^{(i)} \end{pmatrix}, \tag{5}$$

and $R^{(i)T} = \begin{pmatrix} R_I^{(i)T} & 0 \\ 0 & R_B^{(i)T} \end{pmatrix}$. (6)

After eliminating the interior degrees of freedom, the system (2) reduces to a system on the interface:

$$Su_B = g \tag{7}$$

where $S = \sum_{i=1}^{N} R_B^{(i)} S^{(i)} R_B^{(i)T}$ is symmetric positive definite since K is symmetric positive definite, u_B is the vector of the unknown variables on the interface, g is a known vector and $S^{(i)}$ are the local Schur complements of subdomain $i = 1,...,N$, assumed to be positive semi-definite. The problem (7) can be solved by a preconditioned CG method which requires to solve the following auxiliary problem:

$$z = M^{-1}r \tag{8}$$

where r is the residual of (7) and M is a preconditioner. An efficient solution of the large scale problem depends on how we choose an efficient and scalable preconditioner. This issue is dealt with in the next two sections.

4. Balancing Domain Decomposition

The BDD preconditioner proposed by Mandel(3) is constructed by solutions of the local Neumann-Neumann problems on the subdomains coupled with a coarse problem in a coarse space. The BDD preconditioner is of the form:

$$M_{BDD}^{-1} = Q_c + (I - Q_c S) Q_l (I - S Q_c) \qquad (9)$$

where Q_l is the local level part and Q_c is the coarse level part of the preconditioner. The action of the preconditioner (9) in the solution of (8) at each CG iteration is presented in subsection 4.3.

4.1 Local Level

The local level part of the preconditioner basically involves the solution of the local problems. The original BDD employs the Neumann-Neumann preconditioner, Q_l is then expressed by

$$Q_l = \sum_{i=1}^{N} R_B^{(i)} D^{(i)} S^{(i)+} D^{(i)T} R_B^{(i)T} \ . \qquad (10)$$

The dagger (+) indicates the generalized inverse, since $S^{(i)}$ are singular for floating subdomains. The BDD method uses a collection of matrices $D^{(i)}$ that determine the partition of unity on the interface(3, 11),

$$\sum_{i=1}^{N} R_B^{(i)} D^{(i)} R_B^{(i)T} = I \ . \qquad (11)$$

The simplest choice for $D^{(i)}$ is the diagonal matrix with diagonal elements equal to the reciprocal of the number of subdomains with which the degree of freedom is associated.

4.2 Coarse Level

The application of the coarse term $Q_c = R_0 (R_0^T S R_0)^{-1} R_0^T$ amounts to the solution of a coarse problem whose coefficient matrix is $S_0 = R_0^T S R_0$. The operator R_0 translates the coarse degrees of freedom to the corresponding global degrees of freedom and is defined by

$$R_0 = \left[R_B^{(1)} D^{(1)} Z^{(1)}, \ldots, R_B^{(N)} D^{(N)} Z^{(N)}\right] \ . \tag{12}$$

For the scalar heat conductive problem, $Z^{(i)}$ is a column constant vector(3, 11) and can be defined by

$$Z^{(i)} = (1 \ldots 1)^T \tag{13}$$

where the number of element "1" is for each interface point in subdomain i. The operator R_0 is a n by N matrix, where n is the dimension of S.

4.3 Construction of the BDD Preconditioner

The implementation of the BDD preconditioner (9) goes as follows(3, 6):

Step 1: Balance the original residual by solving the coarse problem for an unknown vector $\lambda \in \Re^N$ *:*

$$S_0 \lambda = R_0^T r \, . \tag{14}$$

Step 2: Set

$$s = r - S R_0 \lambda \, . \tag{15}$$

Step 3: Solve Neumann-Neumann problems and average these results

$$\bar{u} = \sum_{i=1}^{N} R_B^{(i)} D^{(i)} S^{(i)+} D^{(i)T} R_B^{(i)T} s \, . \tag{16}$$

Step 4: Compute

$$\bar{s} = r - S\bar{u}\,. \tag{17}$$

Step 5: Solve the coarse problem again for an unknown vector $\mu \in \Re^N$

$$S_0 \mu = R_0^T \bar{s}\,. \tag{18}$$

Step 6: Find the preconditioned vector

$$z = \bar{u} + R_0 \mu\,. \tag{19}$$

The equation (9) can also be expressed as:

$$M_{BDD}^{-1} = \big(P + (I-P)Q_l S(I-P)\big)S^{-1} \tag{20}$$

where $P = Q_c S$ is the $S-$orthogonal projection onto the coarse space.

In an implementation of the BDD preconditioner(3, 9), it is advantageous to use an initial approximation as $u_{B0} = R_0 S_0^{-1} R_0^T g$, then the Step 1 and Step 2 in every iteration can be omitted since after using the residual of the initial approximation in the Step 1, the unknown vector λ becomes the zero vector. As a result, the BDD preconditioner (20) can be written as:

$$M_{BDD}^{-1} = (I - Q_c S)Q_l\,. \tag{21}$$

4.4 Parallel Implementation of BDD

In practice, numerical scalability is most interesting when parallel scalability can also be achieved. The latter property is not related to the convergence rate of an iterative domain decomposition method, but to its implementational features.

It characterizes the ability of an algorithm to deliver larger speed-up using a larger number of processors.

Aiming at parallel scalability, we construct the BDD preconditioner based on the Hierarchical Domain Decomposition Method (HDDM)(14) system. In this system the entire type model is decomposed into two levels. As the first hierarchical level, the whole domain is decomposed into some “parts”. Next as the second hierarchical level, each part is then decomposed into so-called subdomains. On the parallel processors, one processor treats one part, and then solves local subdomain problems independently. Some computational steps of the BDD preconditioner can be trivially carried out on a subdomain by subdomain basis, and therefore are easily amenable to parallel processing and the HDDM system. For example the construction of the coarse matrix and the local solving (16) are done in parallel.

Given that the coarse problem appears at each CG iteration (see subsection 4.3), we propose to precompute the coarse matrix S_0. The construction of the coarse matrix and its Cholesky factorization are done once. It involves the local subdomain computation and requires interprocessor communication only between neighbouring subdomains. The factorized coarse matrix has been kept and the forward and backward substitutions of a coarse system are only employed at each iteration for saving the computation time. Hence special attention is needed only for the construction of the coarse matrix and the solution of the coarse problem $S_0\lambda = R_0^T r$.

5. Incomplete Balancing Domain Decomposition with DIAGonal Scaling (IBDD-DIAG)

The original BDD employs the Neumann-Neumann type method as a local subdomain correction in Step 3 of the subsection 4.3. The Neumann-Neumann method preconditions by the two level weighted sum of inverses of $S^{(i)}$ matrices.

Since $S^{(i)}$ matrices are typically singular, generalized inverse of these matrices is required, e.g. the Moore-Penrose pseudoinverse or some regularized inverse. The regularized inverse takes high computation costs and requires more memory because of the loss of sparsity pattern. To overcome the issues of memory shortage and computation costs, we choose a simplified diagonal scaling preconditioner for $S^{(i)}$ instead of the Neumann-Neumann preconditioner. As the operation of the local subdomain correction is in subdomains, its parallel algorithm is basically compatible to the HDDM system. Hence we term this preconditioner as BDD-DIAG and define as

$$M^{-1}_{BDD-DIAG} = (I - Q_c S) Q_{DIAG} \tag{22}$$

where

$$Q_{DIAG} = \sum_{i=1}^{N} R_B^{(i)} \left(diag\left(K_{BB}^{(i)} \right) \right)^{-1} R_B^{(i)T} \tag{23}$$

is a diagonal matrix.

Now, the degrees of freedom of a coarse problem are related to the number of subdomains, i.e. the dimension of the coarse matrix will increase with the number of subdomains, and then it is difficult to solve them for large-scale problems. As a solution, the developed system employs a new coarse problem solver based on an incomplete balancing technique. In an incomplete balancing procedure, the coarse matrix is factorized incompletely by performing an incomplete Cholesky factorization, and the coarse problem is approximated by the forward and backward substitutions with this incompletely factorized operator. In general, such incomplete Cholesky factorization is performed with an iterative scheme but in our approach the coarse problem is approximated by the incompletely factorized operator without iterations. The process of avoiding *fill-in* during factorization considered in this paper is described next.

In the heat conductive problem, the coarse matrix is symmetric and sparse, with zero elements corresponding to subdomains that have no common neighbouring subdomain. The dimension of the coarse matrix is equal to the number of subdomains. Due to the symmetric property, only the lower triangular elements are stored. Fig. 1 shows an arbitrary example of the sparse coarse matrix where "x" represents the nonzero element and "0" represents the zero element. The elements of the coarse matrix are distributed to the processors as shown in Fig. 1. During the update of one row of the coarse matrix, each processor is to wait until the final updated elements of the same row belonging to the previous processor have been completed. It is clear that in the row-oriented factorization of such sparse coarse matrix, *fill-in* occurs in the position of some zero elements due to the presence of any previous nonzero element in the same row. As a result the factors of the coarse matrix become less sparse than the original coarse matrix and these make the solution expensive. To overcome such a problem, we neglect *fill-in* in some restricted positions. Since elements of each row are distributed to the processors, the distributed portion of the row in some processors may start with zero elements where *fill-in* occurs due to the presence of nonzero elements of the same row belonging to the previous processors. We neglect the *fill-in* in the positions of those zero elements. Fig. 1 shows the positions of *fill-in* which are neglected. This process reduces the waiting time for each processor during the factorization of the coarse matrix and thus it improves the parallel efficiency.

In the original BDD method, the first two steps of the subsection 4.3 can be omitted by using a special initial approximation but here the first two steps can not be omitted since the solution vector of the coarse problem are not the zero vector due to the incomplete balancing.

Placement of Fig. 1

Hence we call such preconditioner as IBDD and define as:

$$M_{IBDD}^{-1} = \widetilde{Q}_c + \left(I - \widetilde{Q}_c S\right) Q_I \left(I - S\widetilde{Q}_c\right) \tag{24}$$

where $\widetilde{Q}_c$ is constructed from the incomplete factorized coarse operator. The IBDD preconditioner (24) becomes the IBDD-DIAG when we use the simplified diagonal scaling instead of the Neumann-Neumann preconditioner. Then the IBDD-DIAG preconditioner can be expressed as:

$$M_{IBDD-DIAG}^{-1} = \widetilde{Q}_c + \left(I - \widetilde{Q}_c S\right) Q_{DIAG} \left(I - S\widetilde{Q}_c\right) \tag{25}$$

The implementation of the IBDD-DIAG preconditioner (25) goes as follows:

Step 1: Balance the original residual by approximating the coarse problem using the incomplete coarse operator for an unknown vector $\widetilde{\lambda} \in \Re^N$*:*

$$\widetilde{S}_0 \widetilde{\lambda} = R_0^T r \,. \tag{26}$$

Step 2: Set

$$\widetilde{s} = r - S R_0 \widetilde{\lambda} \,. \tag{27}$$

Step 3: Perform the diagonal scaling and average these results

$$\widetilde{u} = \sum_{i=1}^{N} R_B^{(i)} \left(diag\left(K_{BB}^{(i)}\right)\right)^{-1} R_B^{(i)T} \widetilde{s} \,. \tag{28}$$

Step 4: Compute

$$\hat{s} = r - S\widetilde{u} \,. \tag{29}$$

Step 5: Approximate the coarse problem again for an unknown vector $\widetilde{\mu} \in \Re^N$

$$\widetilde{S}_0 \widetilde{\mu} = R_0^T \hat{s} \,. \tag{30}$$

Step 6: Find the preconditioned vector

$$z = \widetilde{u} + R_0 \widetilde{\mu} . \tag{31}$$

The $\widetilde{S}_0$ means the corresponding term of the coarse matrix of $R_0^T S R_0$, which is factorized incompletely. Hence, we say that the residual is incompletely balanced in (27). The implementation of incomplete balancing reduces the computation costs for factorization of the coarse matrix and for forward and backward substitution of the problem (26) and (30) and consequently the amount of work of each iteration is reduced. For this reason although IBDD-DIAG preconditioner may increase the number of iterations, a speed-up is achieved for large scale problems in the massively parallel computer.

All the BDD type preconditioners presented in this section are evaluated by solving large scale heat conductive problems in parallel platforms.

6. Numerical Results and Discussion

Usually, the presence or absence of a coarse problem affects only the scalability of a domain decomposition method with respect to the number of subdomains and subdomain sizes. Here we show that the domain decomposition method with the improved BDD type preconditioner is numerically scalable with respect to the number of subdomains. We do not offer any theoretical proof. We try to compare the behavior of the new method with that of the original BDD with respect to the convergence criteria, computation costs and size of required memory. The computational platforms used in different analyses are summarized in Table 1.

We begin with the heat conduction on a test model (a cross section of a cylinder model) (Fig. 2), heat conductivity 8.6475e-2$[W/(mm \cdot K)]$, inner radius 125 mm and outer radius is 250 mm.

Placement of Fig. 2

Placement of Table 1

Placement of Table 2

Fluid with high temperature $310[K]$ flows through the inner surface and fluid with low temperature $290[K]$ flows through the outer surface. The convective heat transfer coefficient of fluid is 2.83723e-3 $[W/(mm^2 \cdot K)]$. Other surfaces are considered with natural boundary conditions.

We perform this analysis in the computational environment I (Table 1) consisting of 6 processing units. We construct several different size of finite element uniform discretizations, which are partitioned into different number of subdomains keeping in mind that the ratio of subdomain size over mesh size is constant approximately. For each discretization, Table 2 reports the number of iterations for convergence with increasing the degrees of freedom. The convergence criterion is that the norm of the relative residual is reduced to 1.0e-6, and then all of the following problems employ the same convergence criterion. Besides with a simplified diagonal scaling (23) (denoted as DDM) the IBDD-DIAG is compared with the other three BDD type preconditioners: 1) the original BDD(denoted as BDD), 2) BDD with a simplified diagonal scaling for the local problem instead of the Neumann-Neumann method (denoted as BDD-DIAG), 3) BDD with incomplete Cholesky factorization of the coarse problem and the Neumann-Neumann method for the local problem (denoted as IBDD). The convergence rate of the simplified diagonal scaling is shown to deteriorate with increasing degrees of freedom. On the other hand, the results depicted in Table 2 show that the convergence rate of the improved BDD type preconditioner, IBDD-DIAG is almost independent of the number of subdomains and can solve larger problems with about the same number of iterations as smaller ones, simply by increasing the number of subdomains.

Another popular engineering benchmark for assessing numerical scalability consists in freezing the mesh size, and refining the domain decomposition by decreasing the subdomain size and therefore increasing the number of subdomains. For a fixed size problem, increasing the number of subdomains results in smaller substructures and a larger interface. Here we consider again the thermal analysis of the test problem described above and focus on the finite element discretization with 413,357 elements and 574,354 nodes. We generate several different domain decompositions with a number of subdomains varying between 100 and 3,200. Fig. 3 shows the number of subdomains vs the number of iterations for this analysis. It can be evaluated from Fig. 3 that the number of iterations of IBDD and IBDD-DIAG show almost the same performance compared with those of the original BDD and BDD-DIAG, respectively. Clearly, these results demonstrate that the domain decomposition method with the improved BDD type preconditioner is numerically scalable as with the original BDD preconditioner.

Next we consider a High Temperature Test Reactor (HTTR) model(15, 16) (Fig. 4) on the computational environment II. The model contains 1,167,268 elements and 1,893,340 nodes. Here we concentrate on the computation time by changing the number of subdomains and also by changing the number of processors. The results in Fig. 5 again confirm the numerical scalability properties of the BDD type preconditioners. The performance results summarized in Fig. 6 report that with the increase of the number of subdomains the IBDD-DIAG algorithm shows better performance compared with BDD and BDD-DIAG. This is because as the number of subdomains increases the degrees of freedom of a coarse problem increases and then BDD and BDD-DIAG require most of the computation time for complete Cholesky factorization of the coarse matrix. The performance results summarized in Table 3 show that the BDD type preconiditioners can compute faster solutions of a fixed mesh problem when the number of processors is increased. These results confirm the parallel scalability properties of the new BDD type preconditioner.

Placement of Fig. 3

Placement of Fig. 4

Placement of Fig. 5

Placement of Fig. 6

Placement of Table 3

Placement of Fig. 7

Placement of Table 4

Finally, as a large scale and real shape model problem having a bad convergence, the present method is applied to a 12 million node unstructured mesh for a precise model of Advanced Boiling Water Reactor (ABWR)(11, 17) as shown in Fig. 7. The model is expressed with 30 parts, 3,000 subdomains, 7,486,792 elements, 11,794,506 nodes in the HDDM system. This problem is solved in the computational environment III. Table 4 shows the computational performances of the thermal analysis of the ABWR model. The positive effects of the improved BDD type preconditioner on the convergence rate, computation costs and required memory are clearly demonstrated. BDD reduces the number of iterations to about 2% and the computation time to about 40% compared with DDM while the IBDD-DIAG reduces the number of iterations to about 4% and the computation time to about 22% compared with DDM. Though a faster convergence rate is observed for the original BDD method, a speed-up is achieved for the IBDD-DIAG method for large scale problems. Especially, almost with the same memory size as DDM, IBDD-DIAG shows the best performance in the computation time. Therefore, the IBDD-DIAG is an effective method with respect to the computation time and the required memory to analyze large scale heat transfer problems.

7. Conclusion

In this paper, we have presented an efficient and scalable balancing domain decomposition type method for solving large scale heat transfer problems. The method is derived by employing a diagonal scaling to precondition the local problems and using an incomplete parallel Cholesky factorized coarse operator. This method has been successfully implemented on a PC cluster for solving a large scale thermal problem. We have reported computational results that confirm that the domain decomposition method with the new preconditioner is numerically scalable. We have also shown that for sufficiently large problems, the improved method outperforms the original BDD method with respect to the computation time and the required memory. The method can be applied to any 3-dimensional large scale heat conductive problems.

Acknowledgement

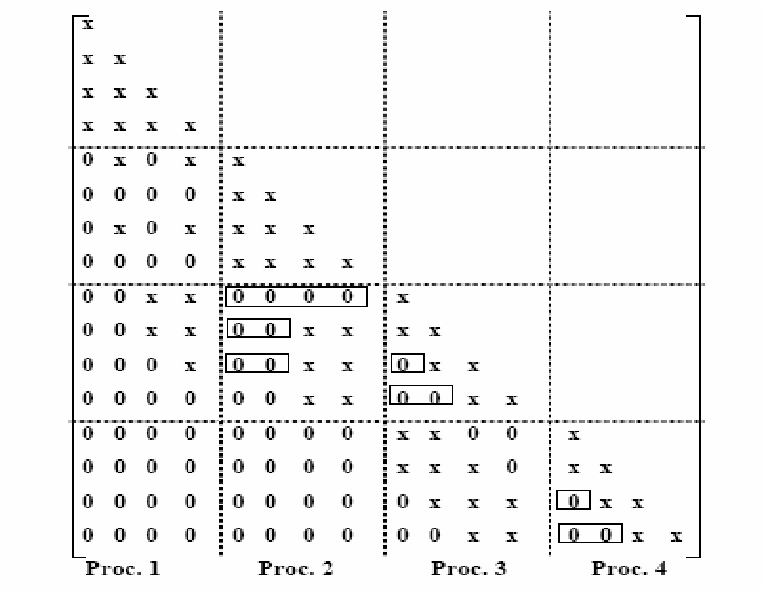
The authors would like to thank Toyo University, Japan and University of Tokyo, Japan, for giving us the opportunity to use the PC clusters. The author also would like to thank Professor Yoshimura, the University of Tokyo, for providing us the ABWR model.

References

(1) Dryja, M. and Widlund O.B., Towards a Unified Theory of Domain Decomposition Algorithms for Elliptic Problems, Third International Symposium on Domain Decomposition Methods for Partial Differential Equations, Held in Houston, Texas, March (1989).
(2) Farhat, C., Chen, P.S. and Mandel, J., A Scalable Lagrange Multiplier Based Domain Decomposition Method for Time-Dependent Problems, International Journal for Numerical Methods in Engineering, Vol. 38 (1995), p. 3831-3853.
(3) Mandel, J., Balancing Domain Decomposition, Comm. on Num. Meth. in Eng., Vol. 9 (1993), p. 223-241.
(4) De Roeck, Y.-H. and Le Tallec, P., Analysis and Test of a Local Domain Decomposition Preconditioner, Fourth International Symposium on Domain Decomposition Methods for Partial Differential Equations, SIAM, Philadelphia, PA (1991).
(5) Mandel, J. and Brezina, M., Balancing Domain Decomposition for Problems with Large Jumps in Coefficients, Mathematics of Computations, Vol. 65, No. 256 (1996), p.1387-1401.
(6) Shioya, R., Ogino, M., Kanayama, H. and Tagami, D., Large Scale Finite Element Analysis with a Balancing Domain Decomposition Method, Key Eng. Mate., 243-244 (2003), p.21-26.
(7) Goldfeld, P., Balancing Neumann-Neumann for (In)Compressible Linear Elasticity and Stokes- Parallel Implementation, Fourteenth International Conference on Domain Decomposition Methods for Partial Differential Equations (2003).
(8) Giraud, L., Kostar, J., Marrocco, A. and Rioual, J.-C., Domain Decomposition Method in Semiconductor Device Modeling, Technical Report (2001), TR/PA/01/51.

(9) Le Tallec, P., Mandel, J. and Vidrascu, M., A Neumann-Neumann Domain Decomposition Algorithm for Solving Plate and Shell Problems, SIAM J. Numer. Aanal., Vol. 35, No. 2 (1998), p.836-867.
(10) Ogino, M., Shioya, R., Kawai, H. and Yoshimura, S., Seismic Response Analysis of Nuclear Pressure Vessel Model with ADVENTURE System on the Earth Simulator, Journal of the Earth Simulator, Vol. 2 (2005), p.41-54.
(11) Shioya, R., Kanayama, H., A.M.M.Mukaddes and Ogino, M., Heat Conductive Analysis with Balancing Domain Decomposition, Theoritical and Applied Mechanics, Vol. 52(2003), p.43-53.
(12) Ogino, M., A.M.M.Mukaddes, Shioya, R. and Kanayama, H., Large Scale Thermal-Solid Coupling Finite Element Analysis with ADVENTURE System on Massively Parallel Computer, European Congress on Computational Methods in Applied Sciences and Engineering (2004).
(13) Ogino, M., Shioya, R., Kanayama, H. and A.M.M.Mukaddes, Incomplete Balancing Domain Decomposition for Large Scale Thermal-Solid Coupling Problems, WCCM VI in Conjunction with APCOM (2004).
(14) Yagawa, G. and Shioya, R., Parallel Finite Elements on a Massively Parallel Computer with Domain Decomposition, Computing Systems in Engineering 4:4-6 (1993), p.495-503.
(15) Shioya, R., Kanayama, H., Tagami, D. and Imamura, E., A Domain Decomposition Approach for Non-steady Heat Conductive Analysis, Advances in Computational Engineering and Science 189.pdf (2001), p.1-6.
(16) A.M.M. Mukaddes, Kanayama, H., Shioya, R. and Ogino, M., Analysis of Large Scale Thermal-Solid Coupling Problems with the ADVENTURE System, KAIST-Kyushu University Joint Workshop on Mechanical and Aerospace Engineering (2004).

(17) Yoshimura, S., Shioya, R., Noguchi, H. and Miyamura, T., Advanced General-Purpose Computational Mechanics System for Large Scale Analysis and Design, Journal of Computational and Applied Mathematics 149 (2002), p.279-296.



$$\left[\begin{array}{cccc:cccc:cccc:cccc}
x &&&&&&&&&&&&&&& \\
x & x &&&&&&&&&&&&&& \\
x & x & x &&&&&&&&&&&&& \\
x & x & x & x &&&&&&&&&&&& \\
\hdashline
0 & x & 0 & x & x &&&&&&&&&&& \\
0 & 0 & 0 & 0 & x & x &&&&&&&&&& \\
0 & x & 0 & x & x & x & x &&&&&&&&& \\
0 & 0 & 0 & 0 & x & x & x & x &&&&&&&& \\
\hdashline
0 & 0 & x & x & \multicolumn{4}{c:}{\boxed{0 \quad 0 \quad 0 \quad 0}} & x &&&&&&& \\
0 & 0 & x & x & \multicolumn{2}{c}{\boxed{0 \quad 0}} & x & x & x & x &&&&&& \\
0 & 0 & 0 & x & \multicolumn{2}{c}{\boxed{0 \quad 0}} & x & x & \boxed{0} & x & x &&&&& \\
0 & 0 & 0 & 0 & 0 & 0 & x & x & \multicolumn{2}{c}{\boxed{0 \quad 0}} & x & x &&&& \\
\hdashline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x & x & 0 & 0 & x &&& \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x & x & x & 0 & x & x && \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x & x & x & \boxed{0} & x & x & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x & x & \multicolumn{2}{c}{\boxed{0 \quad 0}} & x & x
\end{array}\right]$$

$$\begin{array}{cccc}
\text{Proc. 1} & \text{Proc. 2} & \text{Proc. 3} & \text{Proc. 4}
\end{array}$$

$\boxed{\quad\quad}$ shows elements for which *fill-in* is not permitted

Fig. 1 An example of the coarse matrix

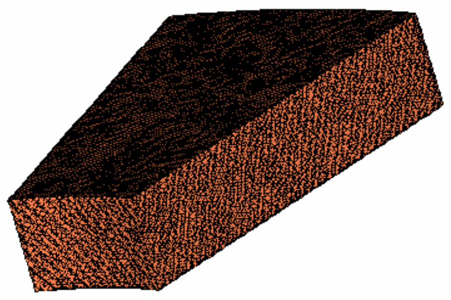


Fig. 2 Mesh of the cross section of a cylinder model

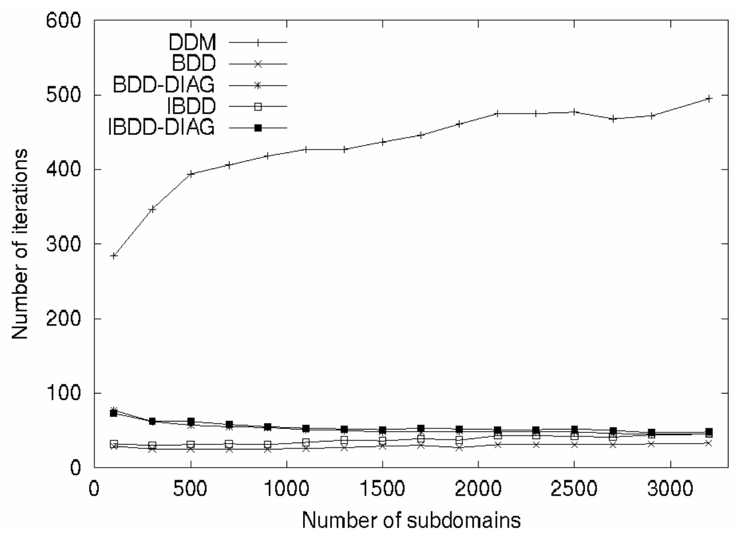


Fig. 3 Number of subdomains vs number of iterations
(Cross section of a cylinder model)

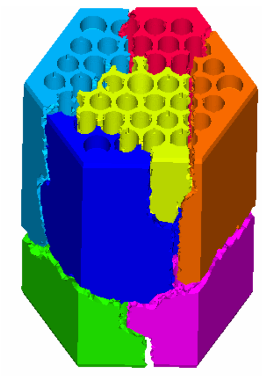


Fig. 4 Part decomposition of a HTTR model

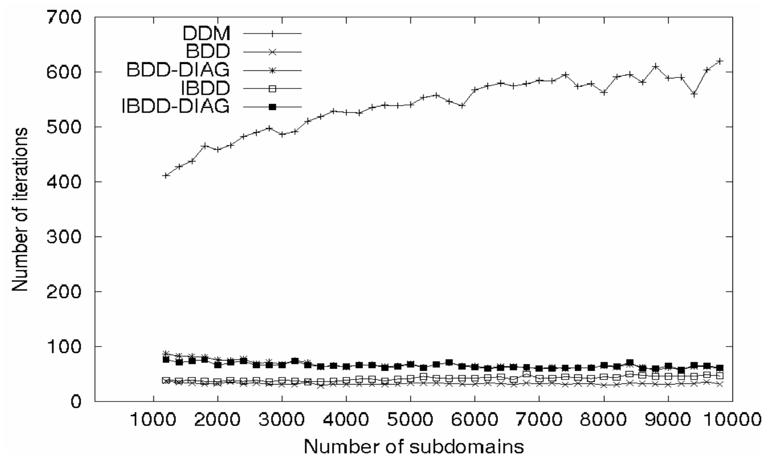


Fig. 5 Number of subdomains vs number of iterations (HTTR)

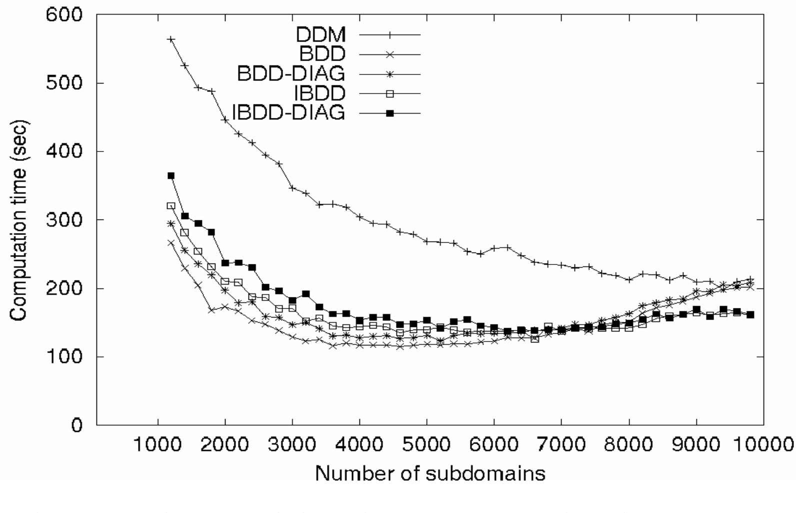


Fig. 6 Number of subdomains vs computation time (HTTR)

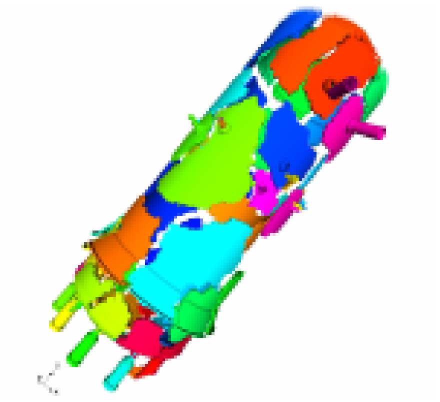


Fig. 7 Part decomposition of a ABWR model

Table 1 Computational environments

Computational environments	Processors	Memory/Proc. (GB)
I	*Pentium 4 2.0 GHz x 6	1
II	** Hitachi SR8000 250 MHz	1
III	***Pentium 4 2.0 GHz x 30	1

* Computational mechanics laboratory, Kyushu University, Japan

** Computer center, University of Tokyo, Japan

***Department of computational science and engineering, Toyo University, Japan

Table 2 Degrees of freedom vs number of iterations
(Cross section of a cylinder model)

*DOF	** N	DDM	BDD	BDD-DIAG	IBDD	IBDD-DIAG
72,251	176	214	26	55	29	49
88,064	224	237	22	50	25	49
122,359	304	265	25	57	29	53
353,532	872	384	26	59	30	55
574,354	1,416	457	29	60	36	60
1,009,832	2,496	517	28	62	37	62

*DOF: Total degrees of freedom ** N : Number of subdomains

Table 3 Total computation time (sec) for different number of processors (HTTR)

Num. of processors	DDM	BDD	BDD-DIAG	IBDD	IBDD- DIAG
8	1,712.46	560.97	802.16	825.61	980.51
16	816.10	258.29	348.48	354.71	393.35
24	485.19	178.51	219.70	231.28	258.43
32	378.04	205.46	226.57	218.86	242.02
56	218.35	175.62	200.74	152.31	147.44

Table 4 Computational performances of thermal analysis (ABWR)

Preconditioner type	Iterations	Time (sec)	#ParLU (sec)	Memory (GB)
DDM	3,201	1,259	---	13.0
BDD	59	522	371	20.1
BDD-DIAG	98	538	371	13.6
IBDD	114	288	30	19.8
IBDD-DIAG	124	264	30	13.4

#ParLU: Time for parallel complete/incomplete Cholesky factorization of the coarse matrix