An Inexact Balancing Preconditioner for Large-Scale Structural Analysis

Kanayama, Hiroshi
Faculty of Engineering, Kyushu University

Ogino, Masao
Faculty of Engineering, Kyushu University

Shioya, Ryuji
Faculty of Engineering, Kyushu University

https://hdl.handle.net/2324/10760

出版情報：Journal of Computational Science and Technology. 2 (1), pp.150-161, 2008-05-02.
Japan Society of Mechanical Engineers
バージョン：
権利関係：

Journal of Computational Science and Technology

Vol.2, No.1, 2008

An Inexact Balancing Preconditioner for Large-Scale Structural Analysis*

Masao OGINO**, Ryuji SHIOYA** and Hiroshi KANAYAMA**

** Faculty of Engineering, Kyushu University, 744 Motooka, Nishi-ku, Fukuoka, 819-0395, Japan

ogino@mech.kyushu-u.ac.jp

Abstract

The balancing domain decomposition (BDD) method is a well-known preconditioner due to its excellent convergence rate. The BDD method includes the Neumann-Neumann preconditioner and a coarse grid correction. Several studies have considered applications of the BDD method to various phenomena and improvement of its convergence rate. However, in applying the BDD method to large-scale problems, it is difficult to solve the coarse problem of a coarse grid correction since the size of the coarse problem increases in proportion to the number of subdomains (i.e., the size of the original problem). Other preconditioners with a coarse grid correction have the same problem. To overcome this problem, use of a new preconditioner, namely, incomplete balancing domain decomposition with a diagonal-scaling (IBDD-DIAG) method is proposed in this study. The method is based on the BDD method, and constructed by an incomplete balancing preconditioner and a simplified diagonal-scaling preconditioner. Moreover, it is parallelized by the hierarchical domain decomposition method. To evaluate this new approach, some computational examples of large-scale problems are demonstrated.

Key words : Balancing Domain Decomposition, Coarse Grid Correction, Non Overlapping Domain Decomposition, Parallel Computing, Elastic Problems

1. Introduction

The purpose of this paper is to present a new preconditioner of the domain decomposition method (DDM) for application to parallel finite element analyses. The DDM is well known as an effective parallel finite element method. In particular, the hierarchical domain decomposition method (HDDM)(1) is one of the most effective methods for large-scale problems due to its excellent parallel performance and suitability for various kinds of parallel computers. The HDDM was successfully applied to problems with 100 million degrees of freedom (DOFs) by Shioya and Yagawa(2). However, as large-scale problems become ill-conditioned, it is necessary to adopt an effective preconditioner in order to attain high performance in the convergence rate and parallel efficiency.

Mandel(4) proposed the balancing domain decomposition (BDD) method in which a coarse grid correction is added to the Neumann-Neumann (N-N) preconditioner of DeRoeck and LeTallec(3). The BDD method is based on non-overlapping domain decomposition with a preconditioned iterative solver, such as the conjugate gradient (CG) method. The convergence theory of the BDD method was later studied by Mandel and Brezina(5). The method was further extended to problems with jumps in coefficients by Mandel and Brezina(6), to plate problems by LeTallec et al.(7), to Stokes problems by Goldfeld(8), to Maxwell's problems by Toselli(9), and to structural problems with linear multipoint constraints by Miyamura(10). However, many studies are still considering medium-scale problems, such as sub-millions to millions of DOFs, and their domain shapes are rather simple. As the DOFs of the coarse problem for a coarse grid correction in preconditioning procedures are directly related to the number of subdomains (i.e., the DOFs of the original problem), in practice it is necessary to consider parallel computational implementations of the coarse problem.

*Received 3 Mar., 2008 (No. 08-0135)
[DOI: 10.1299/jcst.2.150]

In order to address this issue, improvements to the coarse grid correction have been investigated. The balancing domain decomposition based on constraints (BDDC) method proposed by LeTallec et al.(11) and Dohrmann(12) improves the sparsity pattern of the coarse matrix of the coarse problem, and guarantees that the coarse matrix is positive definite. As an improvement of the finite element tearing and interconnecting (FETI) method by Farhat and Roux(16), the dual-primal FETI (FETI-DP) method was proposed by Farhat et al.(14)(15). The FETI-DP method is similar to the BDDC approach, but, in contrast to the latter primal method, the FETI-DP method is a dual method. The main constraints of these methods are on the corner between subdomains, and so the selection of constraints has been studied. In the case of corner constraints only, a method similar as the BDDC method was introduced by Cross(17), while a primal version of the FETI-DP method was introduced by Fragakis and Papadrakakis(18). The BDDC and FETI-DP methods are useful improvements using the coarse grid correction approach, however, in these methods it is not easy to select constraints for arbitrary, complex-shaped models so as to attain a high performance.

In this paper, in order to solve efficiently coarse problems in the BDD preconditioner algorithm, an incomplete balancing domain decomposition with a diagonal-scaling (IBDD-DIAG) method is proposed. The formulation of the IBDD-DIAG method is based on the original BDD method by Mandel(4), but approximates the coarse problem in the coarse grid correction by an incomplete parallel Cholesky factorization of the coarse matrix, and employs a diagonal-scaling preconditioner for subdomain correction instead of the Neumann-Neumann preconditioner. It should be noted that, using the IBDD-DIAG method, Ogino et al.(19) succeeded in solving problems with over 100 million DOFs on the Earth Simulator(20).

In §2, the original BDD method is described, while in §3, the IBDD-DIAG method is proposed. In §4, a parallel implementation of the method is described. In §5, the convergence rate of the method is evaluated, and the method is applied to some examples of large-scale problems. Finally, concluding remarks are presented in §6.

2. Balancing domain decomposition method

2.1. Domain decomposition method

The BDD algorithm is based on the DDM with a preconditioned iterative solver. Consider a system of linear algebraic equations,

$$Ku = f, \tag{1}$$

arising from a finite element discretization of a linear elliptic and self-adjoint boundary value problem in the domain Ω, where K is the global stiffness matrix, u is an unknown vector, and f is a given vector. In elastostatic problems, the stiffness matrix K is symmetric positive definite.

The domain Ω is decomposed into non-overlapping N subdomains $\Omega_i, ..., \Omega_N$, for which the union of all subdomains boundaries is $\Gamma = \cup_{i=1}^{N} \partial\Omega_i$. Here, let R_i denote the 0-1 matrix that restricts the global to local DOFs mapping for Ω_i. Let u_i be the unknown vector in Ω_i, and then $u_i = R_i u$. The stiffness matrix is then described by assembling local stiffness matrices K_i $(1 \leq i \leq N)$,

$$K = \sum_{i=1}^{N} R_i^T K_i R_i. \tag{2}$$

Each u_i in Ω_i is classified as either u_{Bi}, if corresponding to the interface of Ω_i with other subdomains, or u_{Ii}, if having only interior DOFs in Ω_i. With this classification, the local stiffness matrix, the local vectors and the restriction matrix are then represented as follows:

$$u_i = \begin{bmatrix} u_{Ii} \\ u_{Bi} \end{bmatrix}, \quad K_i = \begin{bmatrix} K_{IIi} & K_{IBi} \\ K_{IB_i}^T & K_{BBi} \end{bmatrix}, \quad f_i = \begin{bmatrix} f_{Ii} \\ f_{Bi} \end{bmatrix}, \tag{3}$$

and

$$R_i = \begin{bmatrix} R_{Ii} & 0 \\ 0 & R_{Bi} \end{bmatrix}, \tag{4}$$

where subscripts I and B correspond to the interior DOFs and the interface DOFs, respectively. Furthermore, the system to be solved is described by

$$\begin{bmatrix} K_{II1} & 0 & \cdots & 0 & K_{IB1}R_{B1} \\ 0 & \ddots & & \vdots & \vdots \\ \vdots & & \ddots & \vdots & \vdots \\ 0 & \cdots & \cdots & K_{IIN} & K_{IBN}R_{BN} \\ R_{B_1}^T K_{IB_1}^T & \cdots & \cdots & R_{B_N}^T K_{IB_N}^T & K_{BB} \end{bmatrix} \begin{bmatrix} u_{I1} \\ \vdots \\ \vdots \\ u_{IN} \\ u_B \end{bmatrix} = \begin{bmatrix} f_{I1} \\ \vdots \\ \vdots \\ f_{IN} \\ f_B \end{bmatrix}, \tag{5}$$

$$K_{BB} = \sum_{i=1}^{N} R_{B_i}^T K_{BBi} R_{Bi}, \quad \text{and} \quad f_B = \sum_{i=1}^{N} R_{B_i}^T f_{Bi}. \tag{6}$$

After eliminating interior DOFs of local subdomain matrices, problem Eq.(5) is reduced to the equation for interface DOFs of subdomains,

$$S u_B = g, \tag{7}$$

where S is the Schur complement of K, u_B is the unknown vector of the interface DOFs, and g is the corresponding given vector. Then,

$$S = \sum_{i=1}^{N} R_{B_i}^T S_i R_{Bi}, \quad S_i = K_{BBi} - K_{IB_i}^T K_{II_i}^{-1} K_{IBi}, \tag{8}$$

and

$$g = \sum_{i=1}^{N} R_{B_i}^T (f_{Bi} - K_{IB_i}^T K_{II_i}^{-1} f_{Ii}), \tag{9}$$

where S_i is the local Schur complement of K_i, assumed to be positive semi-definite.

The reduced problem Eq.(7) is called the interface problem, and it is to be solved by the preconditioned CG method. At each iterative step, the preconditioned CG method requires the solution of the following auxiliary problem

$$Mz = r, \tag{10}$$

where M is a symmetric positive definite matrix, called the preconditioner, z is a preconditioned vector, and r is a residual vector in each iterative step.

2.2. Balancing domain decomposition method

The BDD method is formulated in terms of two main methods: a local subdomain correction in local spaces and a coarse grid correction in a coarse space. In the original BDD formulation(4), the Neumann-Neumann preconditioner is employed as the local subdomain correction. The Neumann-Neumann algorithm preconditions by a two-level weighted sum of the inverses of matrices, and its preconditioner is described by

$$T_{N-N} = \sum_{i=1}^{N} R_{B_i}^T D_i^T S_i^{\dagger} D_i R_{Bi}, \tag{11}$$

where $S_i^{\dagger}$ is a generalized inverse of S_i, and D_i is the weight matrix for Ω_i that is formed by decomposition of unity and satisfies

$$\sum_{i=1}^{N} R_{B_i}^{T} D_i R_{Bi} = I, \tag{12}$$

where I is the unit matrix of the order of the global interface DOFs.

Next, the coarse grid correction for structural problems is the Galerkin projection on a coarse space derived from rigid body modes. Here, the coarse space is assumed to construct the local coarse space Z_i that contains all potential local singularities. Now, let P be the S-orthogonal projection onto the coarse space defined by

$$P = QS, \tag{13}$$

and

$$Q = R_0^T S_0^{-1} R_0, \tag{14}$$

where R_0 is the weighted restriction from the global to coarse DOFs, defined by

$$R_0^T = [R_1^T D_1^T Z_1, ..., R_N^T D_N^T Z_N], \tag{15}$$

and S_0 is a coarse matrix, which is typically positive definite, defined by

$$S_0 = R_0 S R_0^T. \tag{16}$$

Consequently, the preconditioner of the BDD method is described by

$$M_{BDD}^{-1} = Q + (I - P)T_{N-N}(I - P)^T. \tag{17}$$

The operation $I - P$ is a so-called coarse grid correction.

Here, the input vector r to the preconditioner is the residual associated with an error vector e by $e = S^{-1}r$. If r is balanced, which means a residual vector has no component of the coarse space, an error vector e is also balanced. Thus, $Pe = 0$. Therefore, $P^T r = SPe = 0$ and $Qr = Pe = 0$, then Eq.(17) can be simplified as

$$M_{BDD}^{-1} = (I - P)T_{N-N}. \tag{18}$$

2.3. Discussion of the implementation of the BDD method for large-scale problems

In solving large-scale problems, we have to consider parallel efficiency, convergence rate, and the efficient use of memory in order to facilitate a good overall computational performance. The BDD method includes three procedures: one based on the DDM method, the Neumann-Neumann preconditioner of the local subdomain correction, and the solution of the coarse problem of the coarse grid correction.

The main procedure based on the DDM is the computation of a product of the Schur complement matrix with a vector on the interface. As the Schur complement assembles local Schur complements in Eq.(8), it can be parallelized subdomain-wise. For large-scale problems, the HDDM, which employs two-level domain decomposition, exhibits an effective performance. In the HDDM, at the first level, an analysis domain is distributed to parallel processors, called parts, then, at the second level, each part is decomposed into so-called subdomains. The first decomposition ensures a load balancing in the parallel processors, while the second one is determined by the experimental optimum number of total subdomains using the system. In the case of the BDD method, when the parallel procedures of a preconditioner are constructed subdomain-wise, the same parallel performance is obtained as for the HDDM technique.

The second procedure is based on the local subdomain correction. In using the Neumann-Neumann preconditioner of Eq.(11), singular problems with positive semi-definite matrices

S_i have to be solved. To solve such singular problems, several methods have been proposed: an efficient computation of the generalized inverse by Farhat and Gradin(21), replacing S_i by a positive definite matrix by DeRoeck and LeTallec(3), and the approach of adding a scaled mass matrix by Dryja and Widlund(22). As a simpler method to construct a non-singular matrix, a positive constant value is added to the diagonal elements of S_i. In this paper, since no local Schur complement matrices are made explicitly, and to save on memory and computational costs, the K_{BBi} matrix corresponding to DOFs on the interface is used instead of the local Schur complement as follows:

$$S_i^\dagger \approx (K_{BBi} + \alpha I)^{-1}, \tag{19}$$

where α is a constant value. In this paper, α is taken to be 10^{-3} multiplied by the maximum absolute value of the diagonal elements of K_{BBi}. However, if the Neumann-Neumann preconditioner is parallelized subdomain-wise, it requires the same size of memory as $K_{II_i}^{-1}$ in the Schur complement matrix in Eq.(8). Here, the BDDC method also requires the solution of local subdomain problems of the Neumann-Neumann type preconditioner, so this method faces the same problems of memory requirements.

The third procedure is based on the coarse grid correction to solve a system of linear equations whose coefficient matrix is a coarse matrix. As the coarse grid correction is implemented at each iteration with its own right-hand side vector, a Cholesky factorized coarse matrix obtained in the first iteration can be retained, and then only the forward and backward substitutions of the coarse problem are implemented in successive iterations, thus saving computation time. However, since the DOFs of a coarse space are related to the number of subdomains, it becomes more difficult to solve for larger scale problems. Moreover, as the coarse problem is constructed to the whole problem by Eq.(7), it can not be parallelized subdomain-wise. To solve such a coarse problem of large-scale problems, it is necessary to apply a parallel Cholesky factorization. However, there is at present no such efficient parallel implementation of Cholesky factorization on the massively parallel computer system. If a method, e.g., a BDDC method with an efficient constraint, improves the sparsity pattern of the coarse matrix or reduces the DOFs of the coarse space, it is certain to face the same issue for large enough DOFs.

3. An incomplete BDD with a diagonal-scaling method

3.1. A BDD with diagonal-scaling

The Neumann-Neumann preconditioner of large-scale problems requires solving local subdomain problems with the retention of generalized inverses of local subdomain matrices. To overcome the problems of memory shortages and computation time requirements, as a simpler solution, we choose a simplified diagonal-scaling preconditioner as a local subdomain correction instead of the Neumann-Neumann type. Hence, this can be formulated as

$$T_{DIAG}^{-1} = \sum_{i=1}^{N} R_{B_i}^T diag(K_{BBi}) R_{Bi}, \tag{20}$$

where $diag()$ is a diagonal matrix, whose entries are diagonal elements of K_{BBi} as a simplification of S_i. A parallel implementation of the preconditioner can clearly be expected to be performed with high parallel efficiency. With the diagonal-scaling for the local subdomain correction, the new BDD type preconditioner can be defined as

$$M_{BDD-DIAG}^{-1} = (I - P) T_{DIAG}, \tag{21}$$

which is named the BDD-DIAG method.

3.2. An incomplete balancing technique

Next, to implement the coarse grid correction with high parallel efficiency, *inexact balancing* based on an incomplete parallel Cholesky factorization is employed. In general, such an incomplete factorized operator is typically used together with some iterative computations to compensate the incompleteness. In this paper, however, the coarse problem is approximated by the incomplete factorized operator without iterations. This *inexact balancing* process decreases computation costs for each iteration and improves parallel efficiency, but may reduce the convergence rate compared with exact balancing. However, total computation time is expected to be reduced. Remarkably, with the original exact BDD preconditioner, a coarse grid correction is performed after a local subdomain correction in each iteration as in Eq.(18). However, with the present inexact BDD preconditioner, a coarse grid correction is also implemented to the CG residual vector before a local subdomain correction due to the incomplete deletion of components of the coarse space. The incomplete balancing technique is applied to two methods, the BDD and the BDD-DIAG, respectively. Consequently, two methods, named the IBDD and the IBDD-DIAG, are formulated similarly to Eq.(17), then each preconditioner are described as

$$M_{IBDD}^{-1} = \tilde{Q} + (I - \tilde{P})T_{N-N}(I - \tilde{P})^T, \tag{22}$$

$$M_{IBDD-DIAG}^{-1} = \tilde{Q} + (I - \tilde{P})T_{DIAG}(I - \tilde{P})^T, \tag{23}$$

where $\tilde{P}$ and $\tilde{Q}$ are incomplete projection operators, defined by

$$\tilde{P} = \tilde{Q}S, \tag{24}$$

and

$$\tilde{Q} = R_0^T \tilde{S}_0^{-1} R_0, \tag{25}$$

where $\tilde{S}_0^{-1}$ is an incomplete Cholesky factorized coarse matrix. In this paper, data parallel strategy of the coarse matrix is based on the block distribution scheme, and the incompleteness of Cholesky factorization is automatically determined by nonzero entries pattern of the matrix and number of processors, e.g. the first fill-ins of each column in a distributed matrix are not allowed.

4. The implementation of the IBDD-DIAG method

The implementation of the present IBDD-DIAG method is summarized as follows. First, we need to approximate a coarse problem:

- *Initial step 1: calculate a residual vector r_0*

$$r_0 = S\overline{u}_{B0} - g. \tag{26}$$

where $\overline{u}_{B0}$ is an initial guess for the solution of Eq.(7). This step is parallelized subdomain-wise.

- *Initial step 2: approximate a coarse problem of the residual vector*

$$\tilde{d}_0 = R_0^T \tilde{S}_0^{-1} R_0 r_0. \tag{27}$$

This step is operated using a parallel incomplete Cholesky factorized matrix.

- *Initial step 3: define a new initial guess*

$$\overline{u}_{B1} = \overline{u}_{B0} - \tilde{d}_0. \tag{28}$$

This step is the so-called parallel operation of distributed vectors.

- *Initial step 4: calculate an initial residual vector*

$$r_1 = S\overline{u}_{B1} - g = r_0 - S\tilde{d}_0. \tag{29}$$

This step is parallelized subdomain-wise.

Next, in each CG step, we implement the IBDD-DIAG preconditioning procedures to solve system (10):

- *Preconditioning step 1: approximate the coarse problem of the residual vector*

$$\lambda = R_0^T \tilde{S}_0^{-1} R_0 r. \tag{30}$$

This step is operated using a parallel incomplete Cholesky factorized matrix.

- *Preconditioning step 2: do a coarse grid correction for the residual vector*

$$\tilde{r} = r - S\lambda. \tag{31}$$

This step is parallelized subdomain-wise.

- *Preconditioning step 3: do the diagonal-scaling preconditioning to the corrected residual vector*

$$\tilde{u} = T_{DIAG}^{-1}\tilde{r}. \tag{32}$$

This step is parallelized subdomain-wise.

- *Preconditioning step 4: define a new residual vector*

$$\hat{r} = r - S\tilde{u}. \tag{33}$$

This step is parallelized subdomain-wise.

- *Preconditioning step 5: approximate the coarse problem of a vector*

$$\mu = R_0^T \tilde{S}_0^{-1} R_0 \hat{r}. \tag{34}$$

This step is operated using a parallel incomplete Cholesky factorized matrix.

- *Preconditioning step 6: set a preconditioned vector*

$$z = \tilde{u} + \mu. \tag{35}$$

This step is a parallel operation of distributed vectors.

5. Numerical examples

5.1. Convergence rate of BDD type preconditioners

To evaluate the convergence rate of the system, we performed an elastostatic stress analysis of a cubic-shaped model. The model is discretized by quadratic tetrahedral finite elements, and then divided into different numbers of DOFs, i.e. from 1,029 to 3,090,902. The model with 20,577 DOFs is shown in Fig.1. Boundary conditions are given by fixing the bottom face of the model while applying forced displacements to the upper face. For the purposes of comparing the convergence rate, as a method of determining the number of subdomains, all models use the same number of elements per subdomain. All the computations in this paper other than those of §5.3 were performed on a PC cluster consisting of 32 processors, where each processor is a Pentium 4 3.0 GHz (x86-64) with 2.0 GB DDR2 SD-RAM. The convergence criteria of the CG method is 1.0×10^{-6}. The five different preconditioners, namely the diagonal-scaling preconditioner (denoted as DDM), BDD that employs exact balancing and the Neumann-Neumann preconditioner, BDD-DIAG that employs exact balancing and the diagonal-scaling preconditioner, IBDD that employs inexact balancing and the Neumann-Neumann preconditioner, and IBDD-DIAG that employs inexact balancing and the diagonal-scaling preconditioner, are applied to each model. The number of iterations required by the different preconditioning approaches are summarized in Table 1. Considering IBDD and IBDD-DIAG, the incompleteness of balancing, which depends on the number of allowed fill-ins in the incomplete Cholesky factorize procedure, tends to deteriorate with increases in the number of processors. As shown in Table 1, all BDD type preconditioners reduce the number of iterations significantly compared with DDM. BDD and BDD-DIAG, which have exact balancing, exhibit an almost constant convergence rate. On the other hand, the number of iterations required by IBDD varies depending on the DOFs. However, IBDD-DIAG also exhibits a near constant convergence rate. This is perhaps due to the fact that IBDD-DIAG does not include the Neumann-Neumann preconditioner, whose preconditioned vector contains a coarse space component, and so may limit an increase in the number of required iterations.

5.2. Parallel efficiency of BDD type preconditioners

In this section, large-scale problems are considered in order to allow a comparison of the parallel efficiency of the BDD type preconditioners. The test analysis model is illustrated in Fig.2. The model is discretized by quadratic tetrahedral finite elements. The number of elements is 1,123,556, the number of nodes is 1,641,872, and the total number of DOFs is 4,904,694. Boundary conditions are given by fixing the bottom face of the model, while a

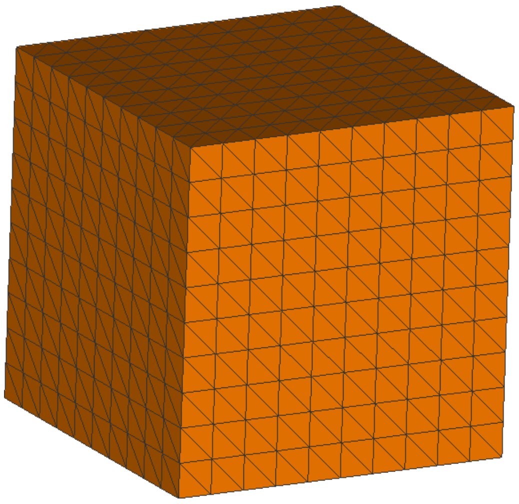
Fig. 1 A benchmark model to test the convergence rates of various BDD-type preconditioning approaches.

Table 1 Comparison of the convergence rates of the various BDD-type preconditioners.

DOFs (Sdm.†)	Procs.‡	DDM	BDD	BDD-DIAG	IBDD	IBDD-DIAG
10,125 (50)	2	252	42	69	42	69
14,729 (73)	2	283	47	70	46	70
20,577 (102)	2	323	45	70	45	70
27,783 (138)	2	359	46	72	48	72
89,373 (446)	2	425	54	74	57	74
206,763 (1,033)	4	574	52	75	70	75
397,953 (1,989)	4	684	51	77	75	77
680,943 (3,404)	4	813	56	73	92	73
1,073,733 (5,368)	8	938	57	79	100	78
1,594,323 (7,971)	8	1,076	57	75	104	75
3,090,903 (15,454)	8	1,332	59	76	130	75

† the number of subdomains.
‡ the number of processors.

surface traction is applied to the upper face. The number of subdomains is about 12,000. The BDD type preconditioners were performed, and the number of iterations, the computation time, the speed-up ratio to the case of 8 processors, and the memory requirements are summarized in Table 2. As shown in Table 2, the convergence rate exhibits the same trends as found in §5.1.

Regarding the memory requirements of the various preconditioning approaches, BDD-DIAG and IBDD-DIAG, which employ a diagonal-scaling preconditioner as a local subdomain correction, reduce memory requirements by around 40 % compared with Neumann-Neumann types.

The computation time required by BDD and BDD-DIAG unfortunately increases with the number of processors because the parallel Cholesky factorization in this paper is not able to return an efficient performance for this case. On the other hand, IBDD and IBDD-DIAG with inexact balancing show a good performance in computation time, and succeed in reducing the time required in comparison with exact balancing.

From the viewpoint of programming technique, the parallel incomplete Cholesky factorization in this paper is constructed by almost the same program as for the complete Cholesky factorization, with the difference between these programs amounting to at most a few tens of lines of code. Consequently, with a simple improvement of the BDD preconditioner, the issues of memory shortage and parallel efficiency in large scale problems can be solved.

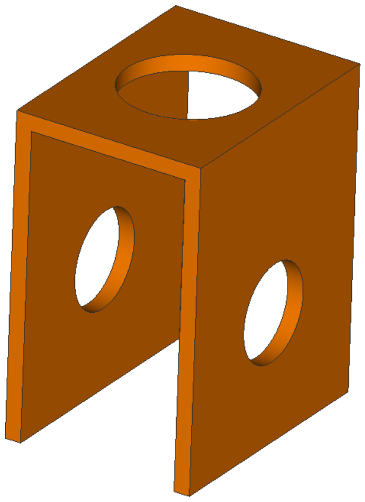

Fig. 2 A benchmark model to examine parallel efficiency.

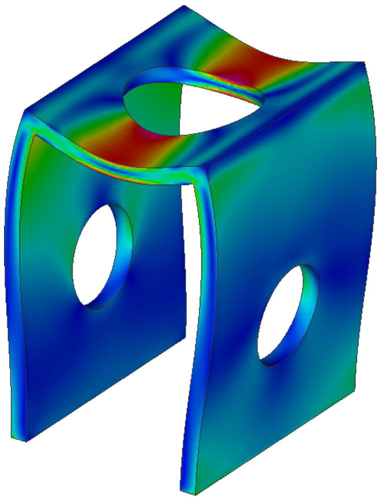

Fig. 3 Deformed configuration and equivalent stress distribution of the benchmark model.

5.3. A Pantheon model with 140 million DOFs

As a final test application, to a huge scale problem, an historical Pantheon model with 140 million DOFs is solved by the IBDD-DIAG method. Fig.4 illustrates a tetrahedral mesh of the Pantheon model with 18 million DOFs. Furthermore, in the analysis model, each finite element of the model in Fig.4 is subdivided into 64 elements. As the result, the Pantheon is modeled with 140 million DOFs. Boundary conditions are given by fixing the bottom plane, while the body's own weight is taken as a body force. Fig.5 shows the result of an equivalent stress distribution. This huge scale problem was performed on the Earth Simulator (ES), a vector-parallel type supercomputer. Table 3 shows the computational performance on 2,048 processors of the ES. The IBDD-DIAG method is successfully used exhibiting a good performance in FLOPS, with results obtained at 24.26 % of peak performance. By the way, the other methods such as DDM, BDD, BDD-DIAG, and IBDD have not obtained converged solutions. As the result, it has thus been demonstrated that the IBDD-DIAG method yields an effective solution for such huge scale analysis problems involving over 100 million DOFs.

6. Conclusion

We have proposed an incomplete balancing domain decomposition with a diagonal-scaling, which employs an incomplete parallel Cholesky factorized coarse matrix to approximate a coarse problem. In this paper, the IBDD-DIAG method exhibits an excellent perfor-

Table 2 Comparison of the parallel efficiencies of the various BDD-type preconditioners.

Num. of procs.		BDD	BDD-DIAG	IBDD	IBDD-DIAG
8	Iterations	41	63	248	125
	Time (sec.)	1,130	1,139	784	470
	Speed-up ratio	1.00	1.00	1.00	1.00
	Memory (GB)	13.5	8.3	12.3	7.2
12	Iterations	41	60	268	138
	Time (sec.)	1,254	1,264	596	358
	Speed-up ratio	0.90	0.90	1.32	1.31
	Memory (GB)	13.8	8.6	12.3	7.1
16	Iterations	41	61	284	144
	Time (sec.)	1,244	1,257	454	261
	Speed-up ratio	0.91	0.91	1.73	1.80
	Memory (GB)	14.0	8.8	12.2	7.0
20	Iterations	41	62	333	162
	Time (sec.)	1,096	1,111	428	220
	Speed-up ratio	1.0	1.03	1.83	2.12
	Memory (GB)	14.1	8.9	12.1	6.9
24	Iterations	41	62	366	181
	Time (sec.)	1,585	1,598	441	224
	Speed-up ratio	0.71	0.71	1.78	2.10
	Memory (GB)	15.1	9.9	12.1	6.9
28	Iterations	40	64	389	192
	Time (sec.)	1,469	1,481	365	186
	Speed-up ratio	0.77	0.77	2.15	2.53
	Memory (GB)	15.2	10.0	12.0	6.8

Table 3 Computational performance of the Pantheon model with 140 million DOFs.

Procs.	Iterations	Time (sec.)	Memory (TB)	TFLOPS (to peak %)
2,048	373	577	3.05	3.88 (24.26)

mance in terms of memory requirements, convergence rate, and computation time. Furthermore, this system has successfully been implemented on the Earth Simulator consisting of 2,048 processors, and succeeded in solving an elastostatic problem of a Pantheon model with 140 million DOFs.

Acknowledgment

The authors would like to thank the Earth Simulation Center for giving us the opportunity to use the Earth Simulator.

References

(1) Yagawa, G. and Shioya, R., Parallel Finite Elements on a Massively Parallel Computer with Domain Decomposition, *Computing Systems in Engineering*, Vol.4 (1994), pp. 495-503.

(2) Shioya, R. and Yagawa, G., Parallel Finite Element Analysis of 100 Million DOFs Based on the Hierarchical Domain Decomposition Method, *Transactions of JSCES*, Vol.3 (2001), pp. 201-206. (in Japanese)

(3) DeRoeck, Y.H. and LeTallec, P., Analysis and Test of a Local Domain Decomposition Preconditioner, *Proceedings of the Fourth International Symposium on Domain Decomposition Methods*, (1991), pp. 112-128.

(4) Mandel, J., Balancing Domain Decomposition, *Communications on Numerical Methods in Engineering*, Vol.9 (1993), pp. 223-241.

(5) Mandel, J. and Brezina, M., Balancing Domain Decomposition: Theory and Performance in Two and Three Dimensions, *UCD/CCM Report*, 2 (1993).

(6) Mandel, J. and Brezina, M., Balancing Domain Decomposition for Problems with Large Jumps in Coefficients, *Mathematics of Computation*, Vol.65 (1996), pp. 1387-1401.

(7) LeTallec, P., Mandel, J. and Vidrascu, M., Balancing Domain Decomposition for Plates,

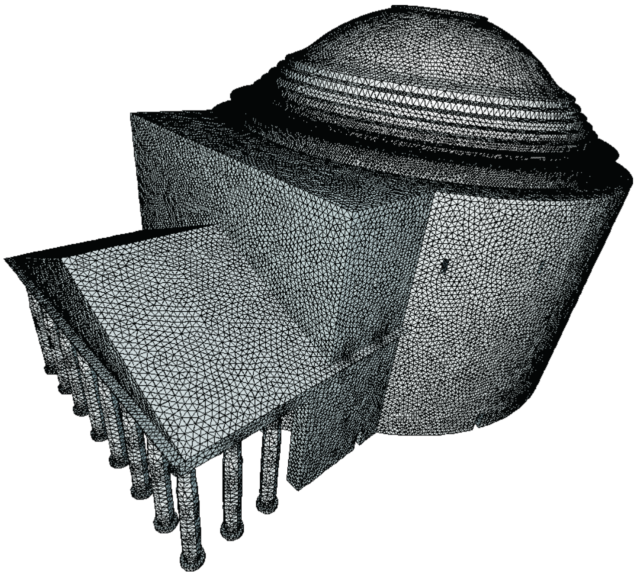

Fig. 4 A Pantheon model, which, when each element is further subdivided into 64 elements, formulates a problem with 140 million DOFs.

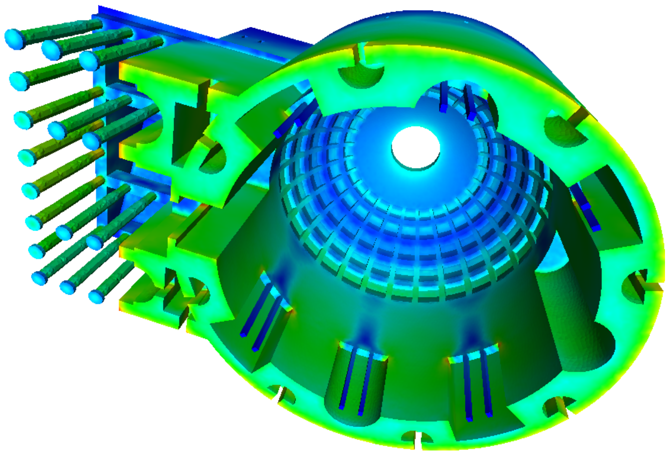

Fig. 5 Deformed configuration and equivalent stress distribution of the Pantheon model.

Proceedings of Seventh International Symposium on Domain Decomposition Methods, (1993), pp. 515-524.

(8) Goldfeld, P., Balancing Neumann-Neumann for (In)Compressible Linear Elasticity and (Generalized) Stokes - Parallel Implementation, *14th International Conference on Domain Decomposition Methods*, (2002), pp. 209-216.

(9) Toselli, A., Neumann-Neumann Methods for Vector Field Problems, *ETNA*, Vol.11 (2000), pp. 1-24.

(10) Miyamura, T., Incorporation of Multipoint Constraints into the Balancing Domain Decomposition Method and Its Parallel Implementation, *IJNME*, Vol.69, No.2 (2006), pp. 326-346.

(11) LeTallec, P., Mandel, J. and Vidrascu, M., A Neumann-Neumann Domain Decomposition Algorithm for Solving Plate and Shell Problems, *SIAM Journal on Numerical Analysis*, Vol.35 (1998), pp. 836-867.

(12) Dohrmann, C.R., A Preconditioner for Substructuring based on Constrained Energy Minimization, *SIAM Journal on Scientific Computing*, Vol.25 (2003), pp. 246-258.

(13) Mandel, J. and Dohrman, C.R., Convergence of a Balancing Domain Decomposition

by Constraints and Energy Minimization, *Numerical Linear Algebra with Applications*, Vol.10 (2003), pp. 639-659.

(14) Farhat, C., Lesoinne, M. and Pierson, P.K., A Scalable Dual-Primal Domain Decomposition Method, *Numerical Linear Algebra with Applications*, Vol.7 (2000), pp. 687-714.

(15) Farhat, C. et al., FETI-DP: A Dual Primal Unified FETI Method. I. A Faster Alternative to the Two-level FETI Method, *International Journal for Numerical Methods in Engineering*, Vol.50 (2000), pp.1523-1544.

(16) Farhat, C. and Roux, R.X., A Method of Finite Element Tearing and Interconnecting and Its Parallel Solution Algorithm, *International Journal for Numerical Methods in Engineering*, Vol.32 (1991), pp. 1205-1227.

(17) Cross, J.M., A Preconditioner for the Schur Complement Domain Decomposition Method, *14th International Conference on Domain Decomposition Methods*, (2002), pp. 383-380.

(18) Fragakis, Y. and Papadrakakis, M., The Mosaic of High Performance Domain Decomposition Methods for Structural Mechanics: Formulation, Interrelation and Numerical Efficiency of Primal and Dual Methods, *Computer Methods in Applied Mechanics and Engineering*, Vol.192, No.35 (2003), pp. 3799-3830.

(19) Ogino, M. et al., Seismic Response Analysis of Nuclear Pressure Vessel Model with ADVENTURE System on the Earth Simulator, *Journal of the Earth Simulator*, Vol.2 (2005), pp. 41-54.

(20) The Earth Simulator Center, ⟨http://www.es.jamstec.go.jp/⟩, (accessed 2008-04-07).

(21) Farhat, C. and Gradin, M., On the General Solution by a Direct Method of a Large-Scale Singular System of Linear Equations: Application to the Analysis of Floating Structures, *International Journal for Numerical Methods in Engineering*, Vol.41 (1998), pp. 675-696.

(22) Dryja, M. and Widlund, O.B., Schwarz Methods of Neumann-Neumann type for Three-Dimensional Elliptic Finite Element Problems, *Communications n Pure and Applied Mathematics*, Vol.48, No.2 (1995), pp. 121-155.