This is the peer reviewed version of the following article: *Int J Numer Methods Eng*. 2022;123(17):3974-4000, which has been published in final form at https://doi.org/10.1002/nme.6995. This article may be used for non-commercial purposes in accordance with Wiley Terms and Conditions for Use of Self-Archived Versions. This article may not be enhanced, enriched or otherwise transformed into a derivative work, without express permission from Wiley or by statutory rights under applicable legislation. Copyright notices must not be removed, obscured or modified. The article must be linked to Wiley's version of record on Wiley Online Library and any embedding, framing or otherwise making available the article or pages thereof by third parties from platforms, services and websites other than Wiley Online Library must be prohibited.

RESEARCH ARTICLE

Implementation of balancing domain decomposition method for parallel finite element analysis involving inactive elements

Yasunori Yusa[1] | Hiroaki Kobayashi[2] | Yuma Murakami[2] | Hiroshi Okada[2]

[1]Department of Mechanical and Intelligent Systems Engineering, Graduate School of Informatics and Engineering, The University of Electro-Communications, Tokyo, Japan
[2]Department of Mechanical Engineering, Faculty/Graduate School of Science and Technology, Tokyo University of Science, Chiba, Japan

Correspondence
Yasunori Yusa, 1-5-1 Chofugaoka, Chofu, Tokyo 182-8585, Japan. Email: y.yusa@uec.ac.jp

Abstract

The present study developed a numerical method to implement domain decomposition (DD) solvers with the diagonal-scaling preconditioner, the balancing domain decomposition (BDD) preconditioner and the BDD with diagonal scaling (BDD-DIAG) preconditioner, for inactive elements. The inactive element, which is a finite element having zero stiffness, is used in several fields such as multi-pass welding analysis, additive manufacturing analysis, damage analysis and topology optimization. For this sort of analysis, we adopted the one-time decomposition approach, in which the DD process is performed once at the beginning of the analysis. Based on this approach, we formulated the matrix–vector multiplication, the preconditioning and the vector operations in the algorithm of the conjugate gradient method, along with the inactive elements and floating degrees of freedom caused by the inactive elements. Consideration of the inactive elements is enabled by the slight modifications of matrices and vectors in the algorithm. Numerical examples confirmed the scalability of the BDD and BDD-DIAG preconditioners with the present implementation method. Moreover, the capability of the present method for damage analysis, topology computation and thermal elastic–plastic analysis of metal additive manufacturing problems was demonstrated.

KEYWORDS:
parallel finite element method, domain decomposition method, inactive element method, damage analysis, topology optimization, additive manufacturing analysis

1 | INTRODUCTION

The domain decomposition (DD) method[1,2] is a powerful parallel computing technique for finite element analysis of solid mechanics problems at large scales. The present study focuses on the non-overlapping iterative type of DD method, which is also known as the iterative substructuring method. In this method, a coarse-grid-correction-based preconditioner, such as balancing domain decomposition (BDD)[3], BDD by constraints (BDDC)[4], finite element tearing and interconnect (FETI)[5] or dual–primal FETI (FETI-DP)[6], is necessary for the analysis of realistic problems to achieve an acceptable number of iteration steps until convergence. Such preconditioners are usually formulated in the framework of Krylov-subspace-based linear iterative solvers, such as the conjugate gradient (CG) method, although they are sometimes combined with other iterative solvers, such as quasi-Newton methods[7]. DD methods with these preconditioners have been applied to various solid mechanics problems. The ADVENTURE System[8,9] uses the BDD preconditioner as well as the BDD with diagonal scaling (BDD-DIAG) preconditioner[10]. The BDD-DIAG preconditioner is an inexact version of the BDD preconditioner. The ADVENTURE System has been used to analyze seismic response problems in nuclear power plants on supercomputers[11,12]. Salinas[13,14] is a FETI-based solver, and it has been applied to the analyses of static and dynamic problems in weapon systems and lithographic systems on supercomputers. Furthermore, the aforementioned preconditioners were implemented in scientific computation frameworks such as PETSc[15,16,17,18] and Trilinos[19,20], and in finite element frameworks such as FEMPAR[21]. These frameworks have contributed to the application of the DD methods to a wide range of solid mechanics problems.

In the DD methods, the finite element analysis model should be decomposed into subdomains before the linear system of equations is solved. In many problems, the analysis model does not change throughout the analysis with time steps, leading to the DD process being performed only once at the beginning of the analysis. In contrast, regeneration of the analysis model is necessary in some problems, such as large-deformation problems with severely deformed finite elements, crack propagation problems and machining problems. In these problems, regeneration of the domain-decomposed analysis model is required at every time step or every few time steps.

Furthermore, one can find another type of problem that is the focus of the present study. In this type, the analysis model involves finite elements that have zero stiffness. For example, multi-pass welding analysis uses inactive or quiet elements to model the region to be welded[22,23,24]. An inactive element has exactly zero stiffness, whereas a quiet element has very low nonzero stiffness to approximately represent zero stiffness. Although quiet elements are frequently used in engineering situations, the iterative linear solvers do not prefer such very-low-stiffness elements due to the possibility of degrading the convergence performance. In the literature on DD methods, this sort of issue has been addressed with jump coefficients[25,26,27]. Thus, only inactive elements are considered in the present study. The inactive element for multi-pass welding analysis has been applied to metal additive manufacturing (AM) analysis[28,29,30]. DD-based metal AM analysis was also performed[31]. Note that there are approaches other than the DD methods or other parallel finite element methods to reduce the computational time for welding or metal AM analysis, for example, with the use of adaptive mesh refinement[31,32,33] and an explicit solver accelerated by a graphics processing unit[34,35]. Furthermore, inactive elements can be found in damage analysis, specifically when the damage variable reaches unity[36]. Also, a similar computation is seen in topology optimization based on the density approach with an unchangeable grid mesh[37,27,38]. Elements that do not contribute significantly to structural stiffness are inactivated in the topology optimization algorithm. For these problems, although regeneration of the domain-decomposed analysis model by actually removing inactive elements at every time step is possible, this approach requires effort to frequently regenerate the data structure of the domain-decomposed analysis model during the analysis. Therefore, we adopted the one-time decomposition approach, in which the DD process is performed once at the beginning of the analysis. To adopt the one-time decomposition approach, we need to formulate the implementation of the DD solver for the inactive elements. This is because the inactive elements cause floating degrees of freedom (DOFs) that should be constrained in an appropriate manner. How to constrain the floating DOFs depends on the adopted linear solution method.

The present study proposes an implementation method for DD solvers with diagonal-scaling, BDD and BDD-DIAG preconditioners with the inactive elements. Although the present implementation method is based on the CG method, it should be able to be applied directly to other preconditioners, such as the BDDC, FETI and FETI-DP preconditioners, as well as other Krylov subspace solvers, such as the biconjugate gradient and minimal residual solvers. Studies on such implementation methods of DD solvers for the inactive elements could not be found in the literature. We use our developed implementation method to present numerical examples of a weak scaling test, a damage analysis, a topology computation and a thermal elastic–plastic analysis of a metal AM problem. The capability of the present method for these problems is demonstrated.

2 | BALANCING DOMAIN DECOMPOSITION METHOD WITH INACTIVE ELEMENTS

First, this section briefly reviews the DD solvers with diagonal-scaling, BDD and BDD-DIAG preconditioners without the inactive elements and their implementation as used in the present study[3,11,10,39,40]. This implementation is almost the same as that of the ADVENTURE System[8,9]. Then, an implementation method for the DD solvers with the three preconditioners for the inactive elements is formulated.

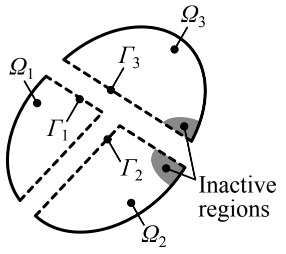


FIGURE 1 Decomposed domain with inactive regions.

2.1 | Implementation without inactive elements

In the DD method, the analysis domain, Ω, is decomposed into N subdomains, Ω_i ($i = 1, 2, \ldots, N$), as shown in Fig. 1, before a linear system of equations is solved. The dashed lines in Fig. 1 are subdomain interfaces, which are represented by Γ_i ($i = 1, 2, \ldots, N$). Moreover, let Γ be $\Gamma_1 \cup \Gamma_2 \cup \cdots \cup \Gamma_N$.

For the i-th subdomain, a linear system of equations generated from the discretized weak form of the equilibrium equation is represented by

$$\begin{bmatrix} \boldsymbol{K}_{\Omega_i\Omega_i} & \boldsymbol{K}_{\Omega_i\Gamma_i} \\ \boldsymbol{K}^{\mathrm{T}}_{\Omega_i\Gamma_i} & \boldsymbol{K}_{\Gamma_i\Gamma_i} \end{bmatrix} \begin{Bmatrix} \boldsymbol{u}_{\Omega_i} \\ \boldsymbol{u}_{\Gamma_i} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{f}_{\Omega_i} \\ \boldsymbol{f}_{\Gamma_i} \end{Bmatrix}, \tag{1}$$

where $\boldsymbol{K}$, $\boldsymbol{u}$ and $\boldsymbol{f}$ are the coefficient matrix, the solution vector and the right-hand side vector, respectively. The subscripts of Ω_i and Γ_i are the internal and interface DOFs of the i-th subdomain, respectively. Superposing Eq. (1) for all the subdomains yields

$$\begin{bmatrix} \boldsymbol{K}_{\Omega_1\Omega_1} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{K}_{\Omega_1\Gamma_1}\boldsymbol{R}_{\Gamma_1} \\ \boldsymbol{0} & \boldsymbol{K}_{\Omega_2\Omega_2} & \ddots & \vdots & \boldsymbol{K}_{\Omega_2\Gamma_2}\boldsymbol{R}_{\Gamma_2} \\ \vdots & \ddots & \ddots & \boldsymbol{0} & \vdots \\ \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{K}_{\Omega_N\Omega_N} & \boldsymbol{K}_{\Omega_N\Gamma_N}\boldsymbol{R}_{\Gamma_N} \\ \boldsymbol{R}^{\mathrm{T}}_{\Gamma_1}\boldsymbol{K}^{\mathrm{T}}_{\Omega_1\Gamma_1} & \boldsymbol{R}^{\mathrm{T}}_{\Gamma_2}\boldsymbol{K}^{\mathrm{T}}_{\Omega_2\Gamma_2} & \cdots & \boldsymbol{R}^{\mathrm{T}}_{\Gamma_N}\boldsymbol{K}^{\mathrm{T}}_{\Omega_N\Gamma} & \sum_{i=1}^{N}\boldsymbol{R}^{\mathrm{T}}_{\Gamma_i}\boldsymbol{K}_{\Gamma_i\Gamma_i}\boldsymbol{R}_{\Gamma_i} \end{bmatrix} \begin{Bmatrix} \boldsymbol{u}_{\Omega_1} \\ \boldsymbol{u}_{\Omega_2} \\ \vdots \\ \boldsymbol{u}_{\Omega_N} \\ \boldsymbol{u}_{\Gamma} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{f}_{\Omega_1} \\ \boldsymbol{f}_{\Omega_2} \\ \vdots \\ \boldsymbol{f}_{\Omega_N} \\ \sum_{i=1}^{N}\boldsymbol{R}^{\mathrm{T}}_{\Gamma_i}\boldsymbol{f}_{\Gamma_i} \end{Bmatrix}, \tag{2}$$

where $\boldsymbol{R}_{\Gamma_i}$ is a restriction operator from the subdomain interfaces, Γ, to the i-th subdomain interface, Γ_i. Instead of solving Eq. (2), the DD method applies static condensation to Eq. (2) as

$$\boldsymbol{S}\boldsymbol{u}_{\Gamma} = \boldsymbol{g}, \tag{3}$$

where

$$\boldsymbol{S} = \sum_{i=1}^{N} \boldsymbol{R}^{\mathrm{T}}_{\Gamma_i}\boldsymbol{S}_{\Gamma_i}\boldsymbol{R}_{\Gamma_i}, \quad \boldsymbol{S}_{\Gamma_i} = \boldsymbol{K}_{\Gamma_i\Gamma_i} - \boldsymbol{K}^{\mathrm{T}}_{\Omega_i\Gamma_i}\boldsymbol{K}^{-1}_{\Omega_i\Omega_i}\boldsymbol{K}_{\Omega_i\Gamma_i}, \tag{4}$$

$$\boldsymbol{g} = \sum_{i=1}^{N} \boldsymbol{R}^{\mathrm{T}}_{\Gamma_i}\boldsymbol{g}_{\Gamma_i}, \quad \boldsymbol{g}_{\Gamma_i} = \boldsymbol{f}_{\Gamma_i} - \boldsymbol{K}^{\mathrm{T}}_{\Omega_i\Gamma_i}\boldsymbol{K}^{-1}_{\Omega_i\Omega_i}\boldsymbol{f}_{\Omega_i}. \tag{5}$$

Equation (3) is solved by the CG method with a preconditioner. The algorithm of the CG method with a preconditioner for solving Eq. (3) is listed in Algorithm 1. In this algorithm, $\boldsymbol{r}$, $\boldsymbol{z}$, $\boldsymbol{p}$ and $\boldsymbol{q}$ are vectors; α and β are scalars; k and τ denote the iteration step and the tolerance to break the iteration, respectively; and $\boldsymbol{M}$ is a preconditioning matrix. In the matrix–vector multiplication process, $\boldsymbol{S}\boldsymbol{p}$, the summation term in Eq. (4) is computed in parallel. To compute the multiplication of $\boldsymbol{K}^{-1}_{\Omega_i\Omega_i}$ and a vector, we adopted the lower–diagonal–lower (LDL) factorization method with the skyline matrix format.

For the preconditioning process, $\boldsymbol{M}^{-1}\boldsymbol{r}$, in Algorithm 1, the present study used the diagonal-scaling, BDD or BDD-DIAG preconditioner. The diagonal-scaling preconditioning matrix used in the present study is

$$\boldsymbol{M}_{\mathrm{DIAG}} = \mathrm{diag}\left(\boldsymbol{K}_{\Gamma\Gamma}\right) = \mathrm{diag}\left(\sum_{i=1}^{N}\boldsymbol{R}^{\mathrm{T}}_{\Gamma_i}\boldsymbol{K}_{\Gamma_i\Gamma_i}\boldsymbol{R}_{\Gamma_i}\right), \tag{6}$$

in which $\mathrm{diag}\,(\boldsymbol{S}) \approx \mathrm{diag}\left(\boldsymbol{K}_{\Gamma\Gamma}\right)$ is assumed. In the implementation, only the diagonal entries of $\boldsymbol{M}_{\mathrm{DIAG}}$ are stored in memory to reduce memory consumption.

Then, the inverse of the BDD preconditioning matrix is

$$\boldsymbol{M}^{-1}_{\mathrm{BDD}} = \left(\boldsymbol{I} - \boldsymbol{R}^{\mathrm{T}}_{0}\boldsymbol{S}^{-1}_{0}\boldsymbol{R}_{0}\boldsymbol{S}\right)\boldsymbol{M}^{-1}_{\mathrm{NN}}, \tag{7}$$

where $\boldsymbol{M}_{\mathrm{NN}}$ is the preconditioning matrix of the Neumann–Neumann preconditioner, which will be explained later. Moreover, $\boldsymbol{S}_0$ is a coarse matrix, which is

$$\boldsymbol{S}_0 = \boldsymbol{R}_0 \boldsymbol{S} \boldsymbol{R}_0^{\mathrm{T}}. \tag{8}$$

To compute the multiplication of $\boldsymbol{S}_0^{-1}$ and a vector, we adopted the LDL factorization method with a sparse direct solver library, namely, MUMPS[41,42,43]. Finally, $\boldsymbol{R}_0$ is a restriction operator from the subdomain interfaces, Γ, to a coarse space. $\boldsymbol{R}_0$ is expressed by

$$\boldsymbol{R}_0 = \begin{bmatrix} \boldsymbol{Z}_{\Gamma_1}^{\mathrm{T}} \boldsymbol{D}_{\Gamma_1} \boldsymbol{R}_{\Gamma_1} \\ \boldsymbol{Z}_{\Gamma_2}^{\mathrm{T}} \boldsymbol{D}_{\Gamma_2} \boldsymbol{R}_{\Gamma_2} \\ \vdots \\ \boldsymbol{Z}_{\Gamma_N}^{\mathrm{T}} \boldsymbol{D}_{\Gamma_N} \boldsymbol{R}_{\Gamma_N} \end{bmatrix}. \tag{9}$$

In this equation, $\boldsymbol{D}_{\Gamma_i}$ is a weight matrix that satisfies $\sum_{i=1}^{N} \boldsymbol{R}_{\Gamma_i}^{\mathrm{T}} \boldsymbol{D}_{\Gamma_i} \boldsymbol{R}_{\Gamma_i} = \boldsymbol{I}$, where $\boldsymbol{I}$ is the identity matrix. As $\boldsymbol{D}_{\Gamma_i}$, we adopted a diagonal matrix in which each diagonal entry is the inverse of the number of subdomains adjacent to the DOF. Then, $\boldsymbol{Z}_{\Gamma_i}$ is a matrix that satisfies

$$\text{Range } \boldsymbol{Z}_{\Gamma_i} \supseteq \text{Null } \boldsymbol{S}_{\Gamma_i}. \tag{10}$$

Based on the rigid-body motion, we use

$$\boldsymbol{Z}_{\Gamma_i} = \begin{bmatrix} 1 & 0 & 0 & 0 & z_1 & -y_1 \\ 0 & 1 & 0 & -z_1 & 0 & x_1 \\ 0 & 0 & 1 & y_1 & -x_1 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & 0 & 0 & z_{n_{\Gamma_i}} & -y_{n_{\Gamma_i}} \\ 0 & 1 & 0 & -z_{n_{\Gamma_i}} & 0 & x_{n_{\Gamma_i}} \\ 0 & 0 & 1 & y_{n_{\Gamma_i}} & -x_{n_{\Gamma_i}} & 0 \end{bmatrix}, \tag{11}$$

where x_i, y_i and z_i ($i = 1,\ 2,\ \ldots,\ n_{\Gamma_i}$) denote nodal coordinates on the i-th subdomain interface, Γ_i. n_{Γ_i} is the number of nodes on Γ_i.

The inverse of the Neumann–Neumann preconditioning matrix, which appeared in the previous paragraph, is expressed by

$$\boldsymbol{M}_{\mathrm{NN}}^{-1} = \sum_{i=1}^{N} \boldsymbol{R}_{\Gamma_i}^{\mathrm{T}} \boldsymbol{D}_{\Gamma_i}^{\mathrm{T}} \boldsymbol{S}_{\Gamma_i}^{\dagger} \boldsymbol{D}_{\Gamma_i} \boldsymbol{R}_{\Gamma_i}. \tag{12}$$

In this equation, $\boldsymbol{S}_{\Gamma_i}^{\dagger}$ denotes a generalized inverse of $\boldsymbol{S}_{\Gamma_i}$. Note that the exact inverse of $\boldsymbol{S}_{\Gamma_i}$ cannot be computed because $\boldsymbol{S}_{\Gamma_i}$ is a singular matrix. The present study adopted the Scaled-BDD regularization technique[39], in which

$$\boldsymbol{S}_{\Gamma_i}^{\dagger} = \left[\boldsymbol{S}_{\Gamma_i} + a \boldsymbol{R}_{\Gamma_i} \boldsymbol{M}_{\mathrm{DIAG}} \boldsymbol{R}_{\Gamma_i}^{\mathrm{T}} \right]^{-1}. \tag{13}$$

ALGORITHM 1 Algorithm of the CG method with a preconditioner for solving Eq. (3).

```
$\boldsymbol{r}^{(0)} = \boldsymbol{g} - \boldsymbol{S}\boldsymbol{u}_{\Gamma}^{(0)}$    ▷ Matrix–vector multiplication and vector operation
$\boldsymbol{z}^{(0)} = \boldsymbol{M}^{-1}\boldsymbol{r}^{(0)}$    ▷ Preconditioning
$\boldsymbol{p}^{(0)} = \boldsymbol{z}^{(0)}$
$k = 0$
repeat
    $\boldsymbol{q}^{(k)} = \boldsymbol{S}\boldsymbol{p}^{(k)}$    ▷ Matrix–vector multiplication
    $\alpha^{(k)} = \frac{\boldsymbol{r}^{(k)\mathrm{T}}\boldsymbol{z}^{(k)}}{\boldsymbol{p}^{(k)\mathrm{T}}\boldsymbol{q}^{(k)}}$    ▷ Vector operations
    $\boldsymbol{u}_{\Gamma}^{(k+1)} = \boldsymbol{u}_{\Gamma}^{(k)} + \alpha^{(k)}\boldsymbol{p}^{(k)}$    ▷ Vector operations
    $\boldsymbol{r}^{(k+1)} = \boldsymbol{r}^{(k)} - \alpha^{(k)}\boldsymbol{q}^{(k)}$    ▷ Vector operations
    $\boldsymbol{z}^{(k+1)} = \boldsymbol{M}^{-1}\boldsymbol{r}^{(k+1)}$    ▷ Preconditioning
    $\beta^{(k)} = \frac{\boldsymbol{r}^{(k+1)\mathrm{T}}\boldsymbol{z}^{(k+1)}}{\boldsymbol{r}^{(k)\mathrm{T}}\boldsymbol{z}^{(k)}}$    ▷ Vector operations
    $\boldsymbol{p}^{(k+1)} = \boldsymbol{z}^{(k+1)} + \beta^{(k)}\boldsymbol{p}^{(k)}$    ▷ Vector operations
    $k = k + 1$
until $\|\boldsymbol{r}^{(k)}\| / \|\boldsymbol{g}\| \leq \tau$
```

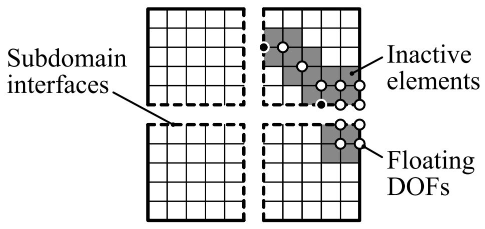


FIGURE 2 Inactive elements and floating DOFs in subdomains.

In this equation, a denotes a positive scalar parameter for regularization. We adopted $a = 10^{-2}$ for every analysis in the present study. In the implementation, we solved a linear system of equations for each subdomain:

$$\begin{bmatrix} \boldsymbol{K}_{\Omega_i\Omega_i} & \boldsymbol{K}_{\Omega_i\Gamma_i} \\ \boldsymbol{K}^{\mathrm{T}}_{\Omega_i\Gamma_i} & \boldsymbol{K}_{\Gamma_i\Gamma_i} + a\boldsymbol{R}_{\Gamma_i}\boldsymbol{M}_{\mathrm{DIAG}}\boldsymbol{R}^{\mathrm{T}}_{\Gamma_i} \end{bmatrix} \begin{Bmatrix} \boldsymbol{z}'_{\Omega_i} \\ \boldsymbol{z}'_{\Gamma_i} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{0} \\ \boldsymbol{r}_{\Gamma_i} \end{Bmatrix}, \tag{14}$$

where $\boldsymbol{r}$ and $\boldsymbol{z}'$ are vectors. To solve this linear system, we adopted the LDL factorization method with the skyline matrix format. The solution of this linear system is equivalent to $\boldsymbol{z}'_{\Gamma_i} = \left[\boldsymbol{S}_{\Gamma_i} + a\boldsymbol{R}_{\Gamma_i}\boldsymbol{M}_{\mathrm{DIAG}}\boldsymbol{R}^{\mathrm{T}}_{\Gamma_i}\right]^{-1}\boldsymbol{r}_{\Gamma_i}$.

Finally, the inverse of the BDD-DIAG preconditioning matrix is

$$\boldsymbol{M}^{-1}_{\text{BDD-DIAG}} = \left(\boldsymbol{I} - \boldsymbol{R}^{\mathrm{T}}_0\boldsymbol{S}^{-1}_0\boldsymbol{R}_0\boldsymbol{S}\right)\boldsymbol{M}^{-1}_{\mathrm{DIAG}}. \tag{15}$$

Its implementation is similar to those of the diagonal-scaling and BDD-DIAG preconditioners.

2.2 | Implementation of the matrix–vector multiplication with inactive elements

Even with the inactive elements, the analysis domain is similarly decomposed into subdomains, as shown in Fig. 1. In addition, we assume that some of the subdomains contain inactive regions that have zero stiffness. The inactive regions may extend over multiple subdomains. Figure 2 illustrates the analysis model after finite element discretization and domain decomposition. The inactive elements are gray, whereas the floating DOFs caused by the inactive elements are marked with white circles. When an inactive element is located at the subdomain interface, whether the DOF on the subdomain interface becomes floating depends on the inactive element in the neighbor subdomain. Non-floating DOFs that are on the subdomain interfaces and that neighbor the inactive elements are marked with black circles. To detect the floating DOFs on the subdomain interfaces, we use $\boldsymbol{M}_{\mathrm{DIAG}}$. If a diagonal entry of $\boldsymbol{M}_{\mathrm{DIAG}}$ is zero, the DOF is floating.

The implementation of the matrix–vector multiplication, $\boldsymbol{q} = \boldsymbol{S}\boldsymbol{p}$, in Algorithm 1 is explained. Considering the floating DOFs, we represent the matrix–vector multiplication by

$$\begin{Bmatrix} \tilde{\boldsymbol{q}} \\ \tilde{\tilde{\boldsymbol{q}}} \end{Bmatrix} = \boldsymbol{S}\begin{Bmatrix} \tilde{\boldsymbol{p}} \\ \tilde{\tilde{\boldsymbol{p}}} \end{Bmatrix}. \tag{16}$$

Variables with a single tilde and a double tilde denote non-floating and floating DOFs, respectively. In the present study, we formulate a method in which components of the vectors at the floating DOFs are always kept at zero. Thus, $\tilde{\tilde{\boldsymbol{p}}}$ is assumed to be zero, specifically

$$\begin{Bmatrix} \tilde{\boldsymbol{q}} \\ \tilde{\tilde{\boldsymbol{q}}} \end{Bmatrix} = \boldsymbol{S}\begin{Bmatrix} \tilde{\boldsymbol{p}} \\ \boldsymbol{0} \end{Bmatrix}. \tag{17}$$

Substituting Eq. (4) into this equation leads to

$$\begin{aligned}
\begin{Bmatrix} \tilde{\boldsymbol{q}} \\ \tilde{\tilde{\boldsymbol{q}}} \end{Bmatrix} &= \left(\sum_{i=1}^{N} \boldsymbol{R}^{\mathrm{T}}_{\Gamma_i}\boldsymbol{S}_{\Gamma_i}\boldsymbol{R}_{\Gamma_i}\right)\begin{Bmatrix} \tilde{\boldsymbol{p}} \\ \boldsymbol{0} \end{Bmatrix} \\
&= \left(\sum_{i=1}^{N} \boldsymbol{R}^{\mathrm{T}}_{\Gamma_i}\left(\boldsymbol{K}_{\Gamma_i\Gamma_i} - \boldsymbol{K}^{\mathrm{T}}_{\Omega_i\Gamma_i}\boldsymbol{K}^{-1}_{\Omega_i\Omega_i}\boldsymbol{K}_{\Omega_i\Gamma_i}\right)\boldsymbol{R}_{\Gamma_i}\right)\begin{Bmatrix} \tilde{\boldsymbol{p}} \\ \boldsymbol{0} \end{Bmatrix} \\
&= \left(\sum_{i=1}^{N} \begin{bmatrix} \tilde{\boldsymbol{R}}_{\Gamma_i} & \boldsymbol{0} \\ \boldsymbol{0} & \tilde{\tilde{\boldsymbol{R}}}_{\Gamma_i} \end{bmatrix}^{\mathrm{T}} \left(\begin{bmatrix} \tilde{\boldsymbol{K}}_{\Gamma_i\Gamma_i} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} - \begin{bmatrix} \tilde{\boldsymbol{K}}_{\Omega_i\Gamma_i} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \tilde{\boldsymbol{K}}_{\Omega_i\Omega_i} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}^{-1} \begin{bmatrix} \tilde{\boldsymbol{K}}_{\Omega_i\Gamma_i} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}\right) \begin{bmatrix} \tilde{\boldsymbol{R}}_{\Gamma_i} & \boldsymbol{0} \\ \boldsymbol{0} & \tilde{\tilde{\boldsymbol{R}}}_{\Gamma_i} \end{bmatrix}\right)\begin{Bmatrix} \tilde{\boldsymbol{p}} \\ \boldsymbol{0} \end{Bmatrix}.
\end{aligned} \tag{18}$$

In this equation, the entries of $\boldsymbol{K}_{\Omega_i\Omega_i}$, $\boldsymbol{K}_{\Gamma_i\Gamma_i}$ and $\boldsymbol{K}_{\Omega_i\Gamma_i}$ at the floating DOFs are zero. Moreover, we assume that the submatrix at the non-floating DOFs, $\tilde{\boldsymbol{K}}_{\Omega_i\Omega_i}$, is invertible. This equation cannot be computed because it includes the inverse of a singular matrix. In the present study, we replace this equation with

$$\begin{Bmatrix} \tilde{\boldsymbol{q}} \\ \tilde{\tilde{\boldsymbol{q}}} \end{Bmatrix} = \left(\sum_{i=1}^{N} \begin{bmatrix} \tilde{\boldsymbol{R}}_{\Gamma_i} & \boldsymbol{0} \\ \boldsymbol{0} & \tilde{\tilde{\boldsymbol{R}}}_{\Gamma_i} \end{bmatrix}^{\mathrm{T}} \left(\begin{bmatrix} \tilde{\boldsymbol{K}}_{\Gamma_i\Gamma_i} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} - \begin{bmatrix} \tilde{\boldsymbol{K}}_{\Omega_i\Gamma_i} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \tilde{\boldsymbol{K}}_{\Omega_i\Omega_i} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}^{-1} \begin{bmatrix} \tilde{\boldsymbol{K}}_{\Omega_i\Gamma_i} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \right) \begin{bmatrix} \tilde{\boldsymbol{R}}_{\Gamma_i} & \boldsymbol{0} \\ \boldsymbol{0} & \tilde{\tilde{\boldsymbol{R}}}_{\Gamma_i} \end{bmatrix} \right) \begin{Bmatrix} \tilde{\boldsymbol{p}} \\ \boldsymbol{0} \end{Bmatrix}. \tag{19}$$

It can be confirmed by some simple algebraic manipulations that both $\tilde{\boldsymbol{q}} = \left(\sum_{i=1}^{N} \tilde{\boldsymbol{R}}_{\Gamma_i}^{\mathrm{T}} \left(\tilde{\boldsymbol{K}}_{\Gamma_i\Gamma_i} - \tilde{\boldsymbol{K}}_{\Omega_i\Gamma_i}^{\mathrm{T}} \tilde{\boldsymbol{K}}_{\Omega_i\Omega_i}^{-1} \tilde{\boldsymbol{K}}_{\Omega_i\Gamma_i} \right) \tilde{\boldsymbol{R}}_{\Gamma_i} \right) \tilde{\boldsymbol{p}}$ and $\tilde{\tilde{\boldsymbol{q}}} = \boldsymbol{0}$ hold. Hence, this replacement is equivalent to actually removing inactive elements. In the implementation, we simply assign one to every diagonal entry with a value of zero, before LDL factorization. This implementation method does not require regeneration of the matrix data structure at every time step, even if an element is activated or inactivated at the time step.

2.3 | Implementation of the diagonal-scaling preconditioning with inactive elements

This subsection presents the implementation method for the diagonal-scaling preconditioning, $\boldsymbol{z} = \boldsymbol{M}_{\mathrm{DIAG}}^{-1}\boldsymbol{r}$. Similar to the approach in § 2.2, the multiplication of the inverse of the preconditioning matrix and a vector can be represented by

$$\begin{Bmatrix} \tilde{\boldsymbol{z}} \\ \tilde{\tilde{\boldsymbol{z}}} \end{Bmatrix} = \boldsymbol{M}_{\mathrm{DIAG}}^{-1} \begin{Bmatrix} \tilde{\boldsymbol{r}} \\ \boldsymbol{0} \end{Bmatrix}. \tag{20}$$

From Eq. (6), we have

$$\begin{Bmatrix} \tilde{\boldsymbol{z}} \\ \tilde{\tilde{\boldsymbol{z}}} \end{Bmatrix} = \begin{bmatrix} \mathrm{diag}\left(\tilde{\boldsymbol{K}}_{\Gamma\Gamma}\right) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}^{-1} \begin{Bmatrix} \tilde{\boldsymbol{r}} \\ \boldsymbol{0} \end{Bmatrix}. \tag{21}$$

We assume that the diagonal entries of the submatrix at the non-floating DOFs, diag $\left(\tilde{\boldsymbol{K}}_{\Gamma\Gamma}\right)$, are not zero. Because it is impossible to invert a singular matrix, we replace this equation with

$$\begin{Bmatrix} \tilde{\boldsymbol{z}} \\ \tilde{\tilde{\boldsymbol{z}}} \end{Bmatrix} = \begin{bmatrix} \mathrm{diag}\left(\tilde{\boldsymbol{K}}_{\Gamma\Gamma}\right) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}^{-1} \begin{Bmatrix} \tilde{\boldsymbol{r}} \\ \boldsymbol{0} \end{Bmatrix}. \tag{22}$$

In the implementation, we assign one to every diagonal entry with a value of zero. This implementation is equivalent to $\tilde{\boldsymbol{z}} = \left[\mathrm{diag}\left(\tilde{\boldsymbol{K}}_{\Gamma\Gamma}\right)\right]^{-1}\tilde{\boldsymbol{r}}$ and $\tilde{\tilde{\boldsymbol{z}}} = \boldsymbol{0}$.

2.4 | Implementation of the BDD preconditioning with inactive elements

Similar to the diagonal-scaling preconditioning, the multiplication of the inverse of the BDD preconditioning matrix and a vector can be represented by

$$\begin{Bmatrix} \tilde{\boldsymbol{z}} \\ \tilde{\tilde{\boldsymbol{z}}} \end{Bmatrix} = \boldsymbol{M}_{\mathrm{BDD}}^{-1} \begin{Bmatrix} \tilde{\boldsymbol{r}} \\ \boldsymbol{0} \end{Bmatrix}. \tag{23}$$

Equation (7) is substituted into this equation, resulting in

$$\begin{Bmatrix} \tilde{\boldsymbol{z}} \\ \tilde{\tilde{\boldsymbol{z}}} \end{Bmatrix} = \left(\boldsymbol{I} - \boldsymbol{R}_0^{\mathrm{T}}\boldsymbol{S}_0^{-1}\boldsymbol{R}_0\boldsymbol{S}\right)\boldsymbol{M}_{\mathrm{NN}}^{-1} \begin{Bmatrix} \tilde{\boldsymbol{r}} \\ \boldsymbol{0} \end{Bmatrix}. \tag{24}$$

In this equation, we consider the Neumann–Neumann preconditioning part:

$$\begin{Bmatrix} \tilde{\boldsymbol{z}}' \\ \tilde{\tilde{\boldsymbol{z}}}' \end{Bmatrix} = \boldsymbol{M}_{\mathrm{NN}}^{-1} \begin{Bmatrix} \tilde{\boldsymbol{r}} \\ \boldsymbol{0} \end{Bmatrix}, \tag{25}$$

where $\boldsymbol{z}'$ is a vector that is different from $\boldsymbol{z}$. We first remove the contribution of inactive elements from the weight matrix, $\boldsymbol{D}_\Gamma$. In other words, $\boldsymbol{D}_\Gamma$ becomes a diagonal matrix in which each diagonal entry is the inverse of the number of subdomains connected to the DOF by an active element. Secondly, as for Eq. (14), Eq. (25) is equivalent to

$$
\begin{Bmatrix} \tilde{\boldsymbol{z}}_{\Omega_i} \\ \tilde{\tilde{\boldsymbol{z}}}_{\Omega_i} \\ \tilde{\boldsymbol{z}}_{\Gamma_i} \\ \tilde{\tilde{\boldsymbol{z}}}_{\Gamma_i} \end{Bmatrix}
= \begin{bmatrix}
\begin{bmatrix} \tilde{\boldsymbol{K}}_{\Omega_i\Omega_i} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} & \begin{bmatrix} \tilde{\boldsymbol{K}}_{\Omega_i\Gamma_i} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \\
\begin{bmatrix} \tilde{\boldsymbol{K}}^{\mathrm{T}}_{\Omega_i\Gamma_i} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} & \begin{bmatrix} \tilde{\boldsymbol{K}}_{\Gamma_i\Gamma_i} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} + a \begin{bmatrix} \tilde{\boldsymbol{R}}_{\Gamma_i} & \boldsymbol{0} \\ \boldsymbol{0} & \tilde{\tilde{\boldsymbol{R}}}_{\Gamma_i} \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{M}}_{\mathrm{DIAG}} & \boldsymbol{0} \\ \boldsymbol{0} & \tilde{\tilde{\boldsymbol{M}}}_{\mathrm{DIAG}} \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{R}}_{\Gamma_i} & \boldsymbol{0} \\ \boldsymbol{0} & \tilde{\tilde{\boldsymbol{R}}}_{\Gamma_i} \end{bmatrix}^{\mathrm{T}}
\end{bmatrix}^{-1}
\begin{Bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \\ \tilde{\boldsymbol{r}}_{\Gamma_i} \\ \tilde{\tilde{\boldsymbol{r}}}_{\Gamma_i} \end{Bmatrix}
$$

$$
= \begin{bmatrix}
\tilde{\boldsymbol{K}}_{\Omega_i\Omega_i} & \boldsymbol{0} & \tilde{\boldsymbol{K}}_{\Omega_i\Gamma_i} & \boldsymbol{0} \\
\boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\
\tilde{\boldsymbol{K}}^{\mathrm{T}}_{\Omega_i\Gamma_i} & \boldsymbol{0} & \tilde{\boldsymbol{K}}_{\Gamma_i\Gamma_i} + a\tilde{\boldsymbol{R}}_{\Gamma_i}\tilde{\boldsymbol{M}}_{\mathrm{DIAG}}\tilde{\boldsymbol{R}}^{\mathrm{T}}_{\Gamma_i} & \boldsymbol{0} \\
\boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & a\tilde{\tilde{\boldsymbol{R}}}_{\Gamma_i}\tilde{\tilde{\boldsymbol{M}}}_{\mathrm{DIAG}}\tilde{\tilde{\boldsymbol{R}}}^{\mathrm{T}}_{\Gamma_i}
\end{bmatrix}^{-1}
\begin{Bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \\ \tilde{\boldsymbol{r}}_{\Gamma_i} \\ \tilde{\tilde{\boldsymbol{r}}}_{\Gamma_i} \end{Bmatrix}. \tag{26}
$$

Note that, although $\tilde{\tilde{\boldsymbol{K}}}_{\Omega_i\Omega_i}$ and $\tilde{\tilde{\boldsymbol{K}}}_{\Gamma_i\Gamma_i}$ are always zero, $\tilde{\tilde{\boldsymbol{M}}}_{\mathrm{DIAG}}$ and $\tilde{\tilde{\boldsymbol{r}}}_{\Gamma_i}$ can be nonzero due to the contribution of the neighboring subdomains. Such DOFs are marked with black circles in Fig. 2. However, this contribution should be deleted in the Neumann–Neumann preconditioning, to obtain equivalence to the case that inactive elements are actually removed. In addition, Eq. (26) includes the inverse of a singular matrix, which cannot be computed. Note that

$$
\begin{bmatrix}
\tilde{\boldsymbol{K}}_{\Omega_i\Omega_i} & \tilde{\boldsymbol{K}}_{\Omega_i\Gamma_i} \\
\tilde{\boldsymbol{K}}^{\mathrm{T}}_{\Omega_i\Gamma_i} & \tilde{\boldsymbol{K}}_{\Gamma_i\Gamma_i} + a\tilde{\boldsymbol{R}}_{\Gamma_i}\tilde{\boldsymbol{M}}_{\mathrm{DIAG}}\tilde{\boldsymbol{R}}^{\mathrm{T}}_{\Gamma_i}
\end{bmatrix}
$$

is assumed to be invertible. Therefore, we replace Eq. (26) with

$$
\begin{Bmatrix} \tilde{\boldsymbol{z}}_{\Omega_i} \\ \tilde{\tilde{\boldsymbol{z}}}_{\Omega_i} \\ \tilde{\boldsymbol{z}}_{\Gamma_i} \\ \tilde{\tilde{\boldsymbol{z}}}_{\Gamma_i} \end{Bmatrix}
= \begin{bmatrix}
\tilde{\boldsymbol{K}}_{\Omega_i\Omega_i} & \boldsymbol{0} & \tilde{\boldsymbol{K}}_{\Omega_i\Gamma_i} & \boldsymbol{0} \\
\boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\
\tilde{\boldsymbol{K}}^{\mathrm{T}}_{\Omega_i\Gamma_i} & \boldsymbol{0} & \tilde{\boldsymbol{K}}_{\Gamma_i\Gamma_i} + a\tilde{\boldsymbol{R}}_{\Gamma_i}\tilde{\boldsymbol{M}}_{\mathrm{DIAG}}\tilde{\boldsymbol{R}}^{\mathrm{T}}_{\Gamma_i} & \boldsymbol{0} \\
\boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I}
\end{bmatrix}^{-1}
\begin{bmatrix}
\boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\
\boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\
\boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} \\
\boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0}
\end{bmatrix}
\begin{Bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \\ \tilde{\boldsymbol{r}}_{\Gamma_i} \\ \tilde{\tilde{\boldsymbol{r}}}_{\Gamma_i} \end{Bmatrix}. \tag{27}
$$

The equivalence to actually removing inactive elements can be proved by confirming $\tilde{\boldsymbol{z}}_{\Gamma_i} = \left[\tilde{\boldsymbol{S}}_{\Gamma_i} + a\tilde{\boldsymbol{R}}_{\Gamma_i}\tilde{\boldsymbol{M}}_{\mathrm{DIAG}}\tilde{\boldsymbol{R}}^{\mathrm{T}}_{\Gamma_i}\right]^{-1}\tilde{\boldsymbol{r}}_{\Gamma_i}$ and $\tilde{\tilde{\boldsymbol{z}}}_{\Gamma_i} = \boldsymbol{0}$. In the implementation, zero is assigned to every component of $\tilde{\tilde{\boldsymbol{r}}}_{\Gamma_i}$. Moreover, we assign one to every diagonal entry of the floating DOFs, before LDL factorization.

After the Neumann–Neumann preconditioning [Eq. (25)] and the matrix–vector multiplication [Eq. (17)], the remaining part of Eq. (24) is

$$
\begin{Bmatrix} \tilde{\boldsymbol{z}}'' \\ \tilde{\tilde{\boldsymbol{z}}}'' \end{Bmatrix} = \boldsymbol{R}_0^{\mathrm{T}}\boldsymbol{S}_0^{-1}\boldsymbol{R}_0 \begin{Bmatrix} \tilde{\boldsymbol{r}}' \\ \boldsymbol{0} \end{Bmatrix}, \tag{28}
$$

where $\boldsymbol{r}'$ and $\boldsymbol{z}''$ are vectors. From Eq. (9), $\boldsymbol{R}_0$ consists of $\boldsymbol{Z}^{\mathrm{T}}_{\Gamma_i}$, $\boldsymbol{D}_{\Gamma_i}$ and $\boldsymbol{R}_{\Gamma_i}$ $(i = 1, 2, \ldots, N)$. Because the diagonal entries of $\boldsymbol{D}_{\Gamma_i}$ at the floating DOFs are zero, the multiplication of $\boldsymbol{D}_{\Gamma_i}$ and a vector always results in a vector having zero components at the floating DOFs. Then, although the multiplication of $\boldsymbol{Z}^{\mathrm{T}}_{\Gamma_i}$ and a vector includes a reduction operation, the contribution of the floating DOFs becomes zero in this reduction operation. Furthermore, in the prolongation operation by $\boldsymbol{R}_0^{\mathrm{T}}$, the process of the multiplication of $\boldsymbol{D}_{\Gamma_i}$ and a vector produces zero components at the floating DOFs. These behaviors of the operators also work in the process of generating the coarse matrix, $\boldsymbol{S}_0$, by Eq. (8). From the above, the remaining part of Eq. (28) is

$$
\begin{Bmatrix} \tilde{\boldsymbol{z}}'_0 \\ \tilde{\tilde{\boldsymbol{z}}}'_0 \end{Bmatrix} = \begin{bmatrix} \tilde{\boldsymbol{S}}_0 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}^{-1} \begin{Bmatrix} \tilde{\boldsymbol{r}}'_0 \\ \boldsymbol{0} \end{Bmatrix}, \tag{29}
$$

where $\boldsymbol{r}'_0$ and $\boldsymbol{z}'_0$ are vectors in the coarse space. Note that $\tilde{\tilde{\boldsymbol{r}}}'_0$ and $\tilde{\tilde{\boldsymbol{S}}}_0$ are equal to zero only when all the finite elements in a subdomain are inactive. Furthermore, $\tilde{\boldsymbol{S}}_0$ is assumed to be invertible. Because a singular matrix cannot be inverted, we replace this equation with

$$
\begin{Bmatrix} \tilde{\boldsymbol{z}}'_0 \\ \tilde{\tilde{\boldsymbol{z}}}'_0 \end{Bmatrix} = \begin{bmatrix} \tilde{\boldsymbol{S}}_0 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}^{-1} \begin{Bmatrix} \tilde{\boldsymbol{r}}'_0 \\ \boldsymbol{0} \end{Bmatrix}. \tag{30}
$$

In the implementation, we assign one to every diagonal entry with a value of zero, before LDL factorization. This implementation is equivalent to $\tilde{\boldsymbol{z}}'_0 = \tilde{\boldsymbol{S}}_0^{-1}\tilde{\boldsymbol{r}}'_0$ and $\tilde{\tilde{\boldsymbol{z}}}'_0 = \boldsymbol{0}$.

2.5 | Implementation of the BDD-DIAG preconditioning with inactive elements

Similar to the BDD preconditioning, the multiplication of the inverse of the BDD-DIAG preconditioning matrix [Eq. (15)] and a vector can be expressed by

$$\begin{Bmatrix} \tilde{\boldsymbol{z}} \\ \tilde{\tilde{\boldsymbol{z}}} \end{Bmatrix} = \left(\boldsymbol{I} - \boldsymbol{R}_0^{\mathrm{T}} \boldsymbol{S}_0^{-1} \boldsymbol{R}_0 \boldsymbol{S}\right) \boldsymbol{M}_{\mathrm{DIAG}}^{-1} \begin{Bmatrix} \tilde{\boldsymbol{r}} \\ \boldsymbol{0} \end{Bmatrix}. \tag{31}$$

The first part is the multiplication of the inverse of the diagonal-scaling preconditioning matrix and a vector. This is the same as Eq. (22). The remaining part is similar to the BDD preconditioning described in § 2.4. After the computation of Eq. (31), $\tilde{\tilde{\boldsymbol{z}}}$ becomes zero.

2.6 | Implementation of the vector operations with inactive elements

In accordance with the implementation in the previous subsections, $\tilde{\tilde{\boldsymbol{u}}}_\Gamma$, $\tilde{\tilde{\boldsymbol{p}}}$, $\tilde{\tilde{\boldsymbol{q}}}$, $\tilde{\tilde{\boldsymbol{r}}}$ and $\tilde{\tilde{\boldsymbol{z}}}$ are always kept at zero. Thus, the vector operations in Algorithm 1 become

$$\begin{Bmatrix} \tilde{\boldsymbol{p}} \\ \tilde{\tilde{\boldsymbol{p}}} \end{Bmatrix}^{\mathrm{T}} \begin{Bmatrix} \tilde{\boldsymbol{q}} \\ \tilde{\tilde{\boldsymbol{q}}} \end{Bmatrix} = \tilde{\boldsymbol{p}}^{\mathrm{T}} \tilde{\boldsymbol{q}}, \quad \begin{Bmatrix} \tilde{\boldsymbol{r}} \\ \tilde{\tilde{\boldsymbol{r}}} \end{Bmatrix}^{\mathrm{T}} \begin{Bmatrix} \tilde{\boldsymbol{z}} \\ \tilde{\tilde{\boldsymbol{z}}} \end{Bmatrix} = \tilde{\boldsymbol{r}}^{\mathrm{T}} \tilde{\boldsymbol{z}}, \tag{32}$$

$$\begin{Bmatrix} \tilde{\boldsymbol{u}}_\Gamma \\ \tilde{\tilde{\boldsymbol{u}}}_\Gamma \end{Bmatrix} + \alpha \begin{Bmatrix} \tilde{\boldsymbol{p}} \\ \tilde{\tilde{\boldsymbol{p}}} \end{Bmatrix} = \begin{Bmatrix} \tilde{\boldsymbol{u}}_\Gamma + \alpha \tilde{\boldsymbol{p}} \\ \boldsymbol{0} \end{Bmatrix}, \quad \begin{Bmatrix} \tilde{\boldsymbol{r}} \\ \tilde{\tilde{\boldsymbol{r}}} \end{Bmatrix} - \alpha \begin{Bmatrix} \tilde{\boldsymbol{q}} \\ \tilde{\tilde{\boldsymbol{q}}} \end{Bmatrix} = \begin{Bmatrix} \tilde{\boldsymbol{r}} - \alpha \tilde{\boldsymbol{q}} \\ \boldsymbol{0} \end{Bmatrix}, \quad \begin{Bmatrix} \tilde{\boldsymbol{z}} \\ \tilde{\tilde{\boldsymbol{z}}} \end{Bmatrix} + \beta \begin{Bmatrix} \tilde{\boldsymbol{p}} \\ \tilde{\tilde{\boldsymbol{p}}} \end{Bmatrix} = \begin{Bmatrix} \tilde{\boldsymbol{z}} + \beta \tilde{\boldsymbol{p}} \\ \boldsymbol{0} \end{Bmatrix}. \tag{33}$$

The vector operations do not require additional implementation on the inactive elements.

3 | NUMERICAL EXAMPLES

To demonstrate the capability of the developed implementation method, four numerical examples are presented. The first example is a weak scaling test of a simple elastic problem for confirming the scalability of the method. The second and third examples are a damage analysis and a topology computation, respectively. In both examples, no inactive elements exist at the beginning of the analysis, but as the time step advances, inactive elements are generated gradually. The final example is a thermal elastic–plastic analysis of a metal AM problem. Before the analysis, inactive elements are placed at the region to be solidified. In these four examples, we tested the three iterative solvers, namely, the DD solvers with the diagonal-scaling, BDD and BDD-DIAG preconditioners. The convergence performance of each iterative solver was evaluated and compared.

3.1 | Weak scaling test

A weak scaling test was performed to confirm the scalability of the present method. The problem was a plate with a square hole subjected to tension. The square hole was modeled by the inactive elements. The dimensions and boundary conditions are illustrated in Fig. 3. Uniformly distributed loads of 10 MPa were applied to the top and the bottom of the plate. Moreover, rigid-body motion was constrained. As the material model, an infinitesimal elastic body was assumed. Young's modulus and Poisson's ratio were set to 200 GPa and 0.3, respectively. Figure 4 depicts the smallest-scale finite element analysis model in this weak scaling test. Hexahedral finite elements based on the B-bar method[44] were used. The inactive elements are red. This finite element analysis model was decomposed by ADVENTURE_Metis[8] into parts and subdomains, as depicted in Fig. 5. ADVENTURE_Metis is a two-level domain decomposer that is based on the METIS and ParMETIS graph partitioning libraries[45,46]. The two-level hierarchical domain decomposition approach[47] was used for parallel implementation in the present study. In Fig. 5, the parts are visualized in a separate manner, whereas the subdomains are colored. Figure 6 visualizes the distribution of the von Mises equivalent stress computed from Figs. 4 and 5. The deformation is magnified by 1,000. The inactive elements were removed from the visualization. Stress concentration in the vicinity of the square hole was observed. To perform a weak scaling test, we prepared six finite element analysis models. Their element divisions, as well as the number of elements, nodes, parts and subdomains, are summarized in Table 1. The numbers of nodes of the six analysis models were approximately in the ratios of 1:2:4:8:16:32. The numbers of subdomains were also in the same ratios. Although the standard weak scaling test uses a computer whose number of processing elements depends on the problem scale, we used a single computer that is independent from the problem scale. This is because we intended to remove the effect of network communications on computational time. The two-level hierarchical domain decomposition approach enables us to increase subdomains without increasing message passing

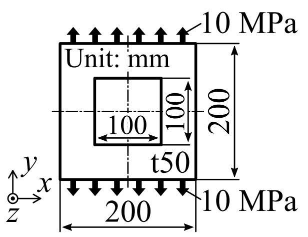


FIGURE 3 Dimensions and boundary conditions for the plate with a square hole for the weak scaling test.

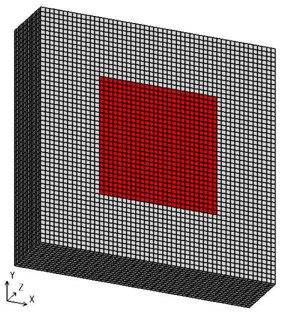


FIGURE 4 Finite element analysis model of the plate with a square hole.

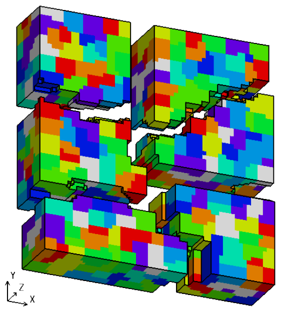


FIGURE 5 Two-level domain decomposition of Fig. 4.

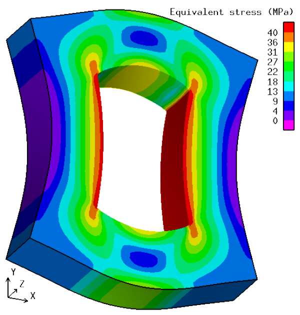


FIGURE 6 Distribution of the equivalent stress computed from Figs. 4 and 5.

interface (MPI) processes. The used computer had an Intel Core i7-8700 CPU with six cores and 64 GB of DDR4-2666 memory. For every analysis in the weak scaling test, six MPI processes were launched on this six-core CPU.

Figure 7 shows the CG residuals of the DD solvers with the diagonal-scaling, BDD and BDD-DIAG preconditioners in the smallest-scale analysis. The horizontal and vertical axes represent the CG iteration step, k, and the CG residual, $\|\boldsymbol{r}^{(k)}\| / \|\boldsymbol{g}\|$, respectively. The tolerance for the CG residual, τ, was set to 10^{-3}. The BDD and BDD-DIAG preconditioners achieved much smaller numbers of CG iteration steps until convergence than that for the diagonal-scaling preconditioner. The number of CG iteration steps until convergence is plotted in Fig. 8. The horizontal and vertical axes represent the number of subdomains of the analysis model and the number of CG iteration steps until convergence, respectively. The number of CG iteration steps for the diagonal-scaling preconditioner depended on the number of subdomains, whereas those of the BDD and BDD-DIAG preconditioners were almost constant. The results confirmed the scalability of the BDD and BDD-DIAG preconditioners with the inactive elements on the number of CG iteration steps. In this example, the difference between the results of the BDD and BDD-DIAG preconditioners was very small.

Figure 9 shows the measured computational times for the three preconditioners. The horizontal and vertical axes represent the number of subdomains and the measured computational time, respectively. The BDD-DIAG preconditioner achieved the shortest computational time among the three preconditioners. The longest computational time was exhibited by the diagonal-scaling preconditioner. Because a single computer was used in this weak scaling test, the ideal computational times depend on the number of subdomains. Thus, the ratio of the measured computational times to the ideal ones is plotted in Fig. 10. In the present study, the ideal computational time was the measured computational time with 480 subdomains multiplied by the number of subdomains per 480. If this ratio stays at unity, the solver is exactly scalable about computational time. This figure indicates that the BDD and BDD-DIAG preconditioners achieved more scalable computational times than the diagonal-scaling preconditioner. The change in the ratio of the diagonal-scaling preconditioner can be explained by the number of CG iteration steps. The solid line in Fig. 10 is very similar in shape to the solid line in Fig. 8, indicating that they are almost proportional. In contrast, even though the number of CG iteration steps stayed almost the same, the ratios of the BDD and BDD-DIAG preconditioners changed. This is likely due to the multiplication process of $\boldsymbol{S}_0^{-1}$ and a vector. Because a sparse direct solver was used for this process in the present study, the ratios increased gradually depending on the number of subdomains.

TABLE 1 Number of analysis model components in the weak scaling test.

Element divisions	# of elements	# of nodes	# of parts (MPI processes)	# of subdomains
52×52×14	37,856	42,135	6	480
68×68×17	78,608	85,698	6	960
84×84×22	155,232	166,175	6	1,920
108×108×27	314,928	332,668	6	3,840
136×136×34	628,864	656,915	6	7,680
172×172×43	1,272,112	1,316,876	6	15,360

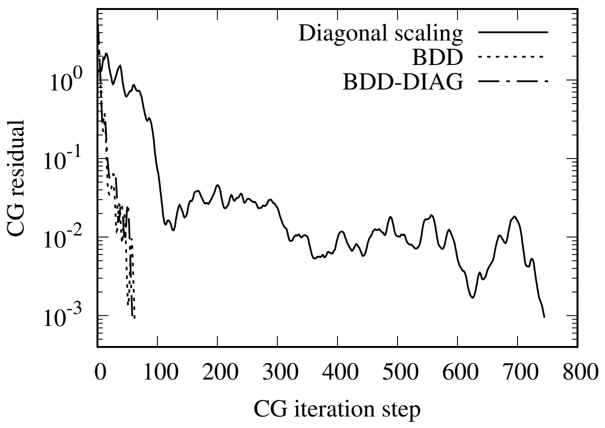


FIGURE 7 CG residuals in the smallest-scale analysis of the weak scaling test.

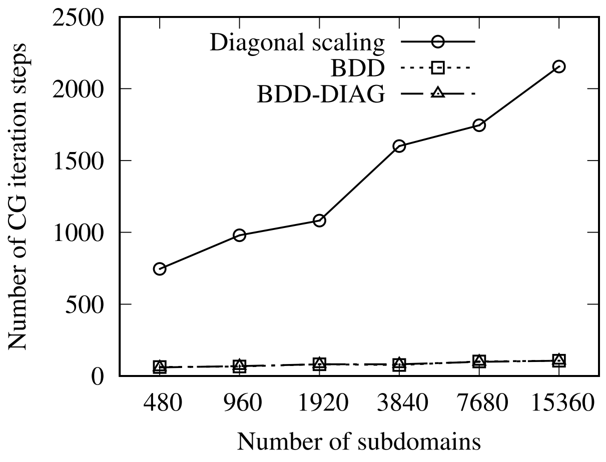


FIGURE 8 Change in the number of CG iteration steps until convergence in the weak scaling test.

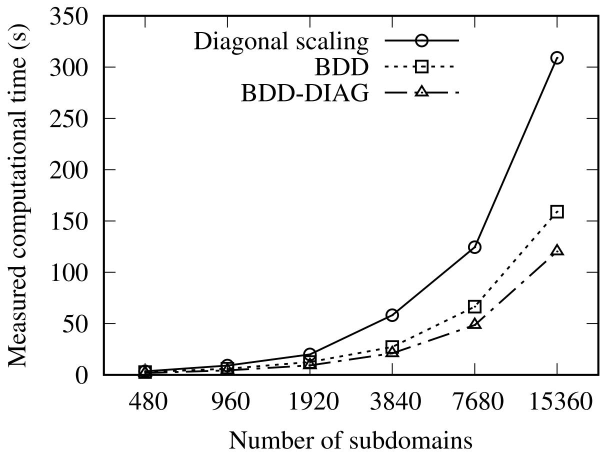


FIGURE 9 Change in the measured computational time in the weak scaling test.

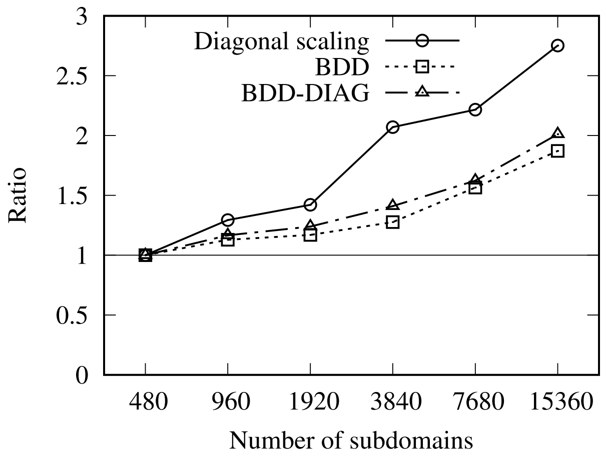


FIGURE 10 Ratios of the measured computational times to the ideal ones in the weak scaling test.

3.2 | Damage analysis

Although various damage models have been proposed to date[36], we used a very primitive damage model to demonstrate the capability of the present implementation method for inactive elements. This model is based on studies of the quasi-brittle damage and failure of concrete[36]. The damage variable evolution for the peak stress and decreased stiffness can be expressed mainly by the maximum principal strain or stress, because concrete has much lower tensile strength than compressive strength. In the present numerical example, the decrease in stiffness was ignored for simplicity. Thus, the peak stress as effective failure is represented only by the maximum principal strain or stress. In the algorithm, the damage variable, D, is initially zero, and it becomes unity if $\sigma_1 \geq \sigma_{1c}$ is satisfied. σ_1 and σ_{1c} denote the maximum principal stress and its critical value, respectively. At each incremental step, a linear system of equations for an infinitesimal elastic body is solved once, followed by the update of D. Every element having $D = 1$ at any integration point is treated as an inactive element.

The problem was a cantilever beam subjected to a load at the free end. Its dimensions and boundary conditions are illustrated in Fig. 11. A uniformly distributed load was applied incrementally on the free end face in 76 increments of 0.001 MPa (1.6 N). Hence, the applied load at the final incremental step was 0.076 MPa (121.6 N). In addition to the illustrated constraints, translational rigid-body motion in the z direction was constrained. Young's modulus, Poisson's ratio and the critical maximum principal stress, σ_{1c}, were set to 20 GPa, 0.2 and 2 MPa, respectively. Figure 12 depicts the finite element analysis model of the beam. Because the whole view is filled with black mesh lines, only an enlarged view is presented. The numbers of elements

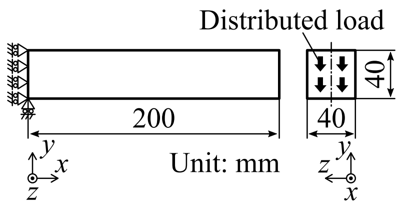


FIGURE 11 Dimensions and boundary conditions of the cantilever beam for the damage analysis.

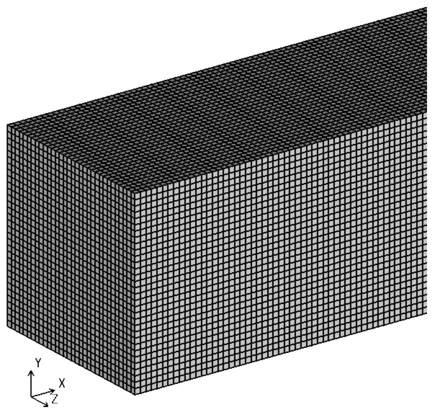


FIGURE 12 Finite element analysis model of the beam.

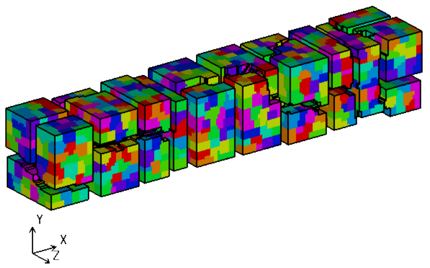


FIGURE 13 Two-level domain decomposition of Fig. 12.

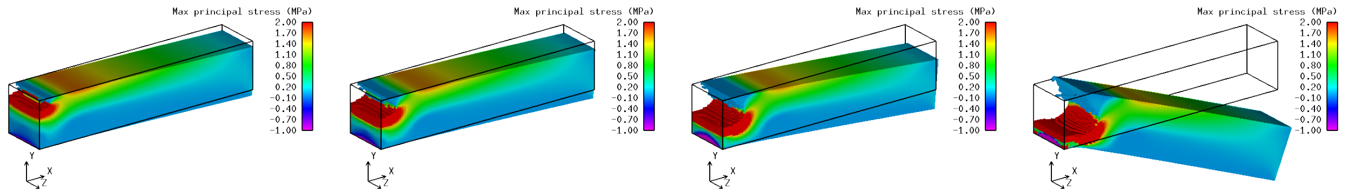
and nodes were 163,840 and 175,329, respectively. Hexahedral finite elements based on the B-bar method[44] were used. Each element was a cube whose edge length was 1.25 mm. At the beginning of the analysis, no inactive elements existed. This finite element analysis model was decomposed by ADVENTURE_Metis[8] into 24 parts and 1,200 subdomains, as depicted in Fig. 13. The number of MPI processes was the same as the number of parts (24). A computer cluster having four computing nodes connected by a 10 Gigabit Ethernet network was used. Each computing node had an Intel Core i7-8700 CPU with six cores and 64 GB of DDR4-2666 memory.

Figure 14 visualizes the distributions of the maximum principal stress at the final four incremental steps computed by the DD solver with the BDD-DIAG preconditioner. The deformation is magnified by 50. The inactive elements were removed from the visualization. Figure 15 depicts the inactive elements in red. As inactive elements at the fixed end increased due to stress concentration, the deformation increased steeply. At the final incremental step, most of the upper part at the fixed end became inactive. Note that floating regions surrounded by inactive elements were generated in this example. However, the inverse of a singular matrix did not appear because the floating regions were connected to the subdomain interfaces. If there is a floating region that is totally inside a subdomain, the region should be constrained by an appropriate manner. Figure 16 shows the load–deflection relation. The vertical and horizontal axes represent the incrementally applied load and the computed deflection at the free end, respectively. In addition to the numerical solution with damage, the solution of the beam theory without damage is plotted. The numerical solution agreed well with the solution of the beam theory in most of the initial steps. After that, the discrepancy increased gradually as the applied load increased, due to damage propagation.

Figure 17 shows the CG residuals of the DD solvers with the diagonal-scaling, BDD and BDD-DIAG preconditioners at the final incremental step. The horizontal and vertical axes represent the CG iteration step, k, and the CG residual, $\|\boldsymbol{r}^{(k)}\| \,/\, \|\boldsymbol{g}\|$, respectively. The tolerance for the CG residual, τ, was set to 10^{-3}. The number of CG iteration steps until convergence for each preconditioner and incremental step is plotted in Fig. 18. The horizontal and vertical axes represent the incremental step and the number of CG iteration steps until convergence, respectively. These two figures show that the diagonal-scaling preconditioner required many more iteration steps until convergence than did the BDD and BDD-DIAG preconditioners. The difference between the results of the BDD and BDD-DIAG preconditioners was very small. As the incremental step advanced, inactive elements increased gradually from zero, indicating that the present implementation achieved good convergence performance with or without inactive elements. Finally, the measured computational times for the diagonal-scaling, BDD and BDD-DIAG preconditioners for all incremental steps were 339 s, 259 s and 191 s, respectively.

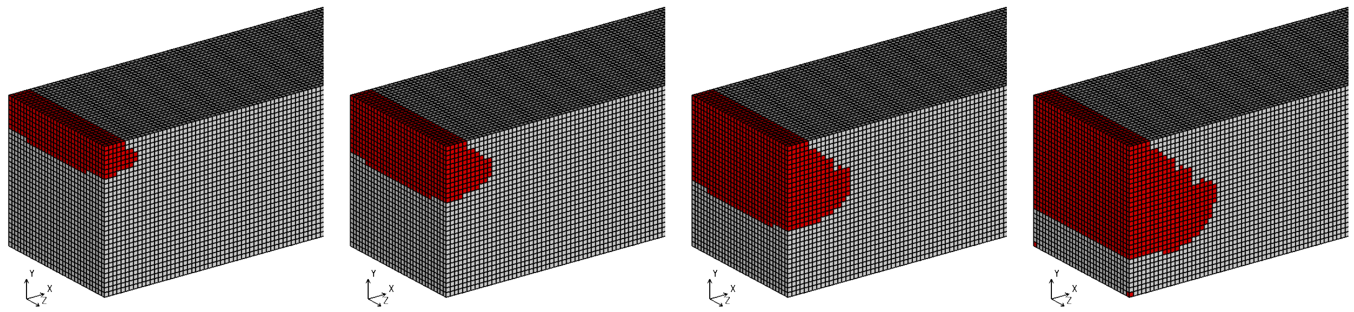
3.3 | Topology computation

Although sophisticated formulations and algorithms for topology computation have been developed for many years[37], we adopted a very primitive algorithm to demonstrate the capability of the present implementation method for inactive elements. This algorithm is based on the density approach and is similar to that of Mrzygłód[38]. It involves computation steps to gradually reduce the number of active elements for minimizing the volume. Let the initial value of the normalized density, $\rho^{(1)}$, be unity.



(A) At the 73rd incremental step. (B) At the 74th incremental step. (C) At the 75th incremental step. (D) At the 76th incremental step.

FIGURE 14 Distributions of the maximum principal stress at the final four incremental steps in the damage analysis.



(A) At the 73rd incremental step. (B) At the 74th incremental step. (C) At the 75th incremental step. (D) At the 76th incremental step.

FIGURE 15 Inactive elements (red) at the final four incremental steps in the damage analysis, before solving each linear system of equations.

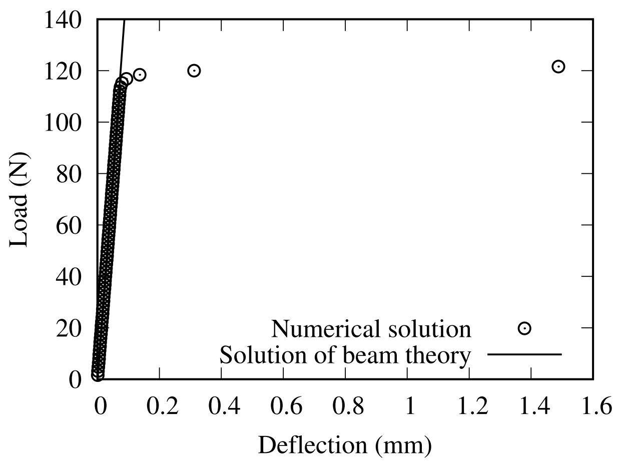


FIGURE 16 Relation between the applied load and deflection in the damage analysis.

At the κ-th computation step, a linear system of equations for an infinitesimal elastic body is solved once, followed by the update of $\rho^{(\kappa+1)}$. If $\bar{\sigma}^{(\kappa)} \leq \bar{\sigma}_c^{(\kappa)}$ is satisfied, $\rho^{(\kappa+1)}$ becomes zero. $\bar{\sigma}$ and $\bar{\sigma}_c$ denote the von Mises equivalent stress and its critical value, respectively. $\bar{\sigma}_c$ is also updated at each computation step by $\bar{\sigma}_c^{(\kappa+1)} = \bar{\sigma}_c^{(\kappa)} + \Delta\bar{\sigma}_c$, where $\Delta\bar{\sigma}_c$ denotes the critical equivalent stress increment. At each computation step, every element having $\rho^{(\kappa)} = 0$ at any integration point is inactivated.

The problem was a cantilever beam subjected to a concentrated load at the free end. Its dimensions and boundary conditions are illustrated in Fig. 19. The dimensions are the same as those for the damage analysis in § 3.2. A concentrated load of 1 kN was applied constantly for 40 computation steps at the center of the free end face. Young's modulus and Poisson's ratio for $\rho = 1$

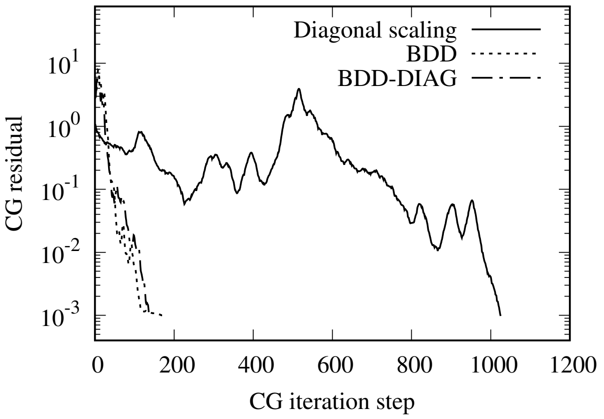


FIGURE 17 CG residuals in the damage analysis at the final incremental step.

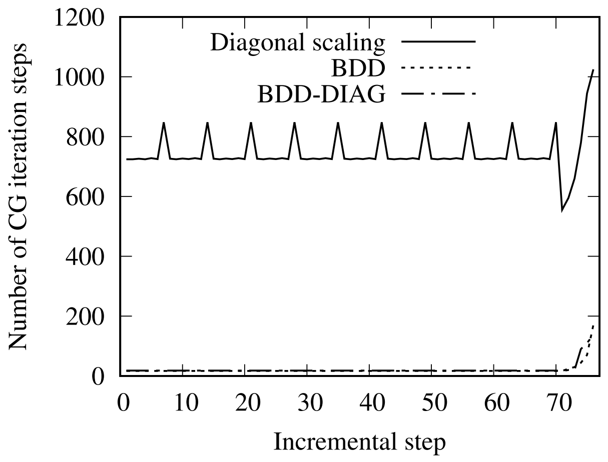


FIGURE 18 Change in the number of CG iteration steps until convergence in the damage analysis.

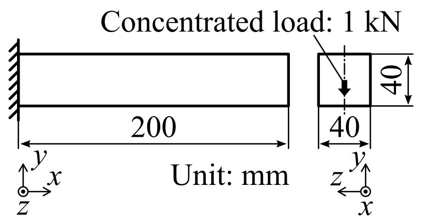
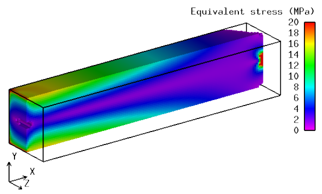


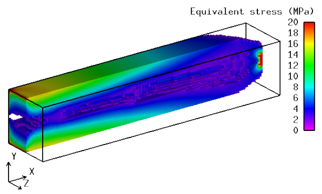
FIGURE 19 Dimensions and boundary conditions for the beam for the topology computation.

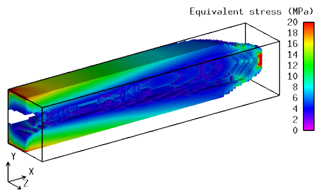
were 70 GPa and 0.33, respectively. The initial critical equivalent stress, $\bar{\sigma}_c^{(1)}$, and the critical equivalent stress increment, $\Delta\bar{\sigma}_c$, were set to 0.1 MPa and 0.1 MPa, respectively. The finite element analysis model and its domain decomposition were the same as those in § 3.2, and are visualized in Figs. 12 and 13, respectively. The computer cluster described in § 3.2 was used again.

Figure 20 visualizes sections of the distributions of the von Mises equivalent stress at the 10th, 20th, 30th and 40th steps computed by the DD solver with the BDD-DIAG preconditioner. The inactive elements were removed from the visualization. Figure 21 depicts sections of the distributions of the inactive elements, which are indicated by red. As the computation step advanced, elements in the mid-height region and in the outer region of the free end became inactive. Note that, although floating regions surrounded by inactive elements were generated, they were connected to the subdomain interfaces. The inverse of a singular matrix did not appear in this example. Figures 22 and 23 show the change in the volume and the deflection of the beam. In both figures, the horizontal axes represent the computation step, κ. The vertical axes in Figs. 22 and 23 represent the volume and the deflection, respectively. In addition to the numerical solution, the initial volume and the solution of the beam theory are plotted. As inactive elements increased, the volume decreased notably, whereas the deflection increased gradually.

Figure 24 shows the CG residuals of the DD solvers with the diagonal-scaling, BDD and BDD-DIAG preconditioners at the final incremental step. The horizontal and vertical axes represent the CG iteration step, k, and the CG residual, $\|\boldsymbol{r}^{(k)}\| \,/\, \|\boldsymbol{g}\|$, respectively. The tolerance for the CG residual, τ, was set to 10^{-3}. The number of CG iteration steps until convergence for each computation step is plotted in Fig. 25. The horizontal and vertical axes represent the computation step, κ, and the number of CG iteration steps until convergence, respectively. These two figures show that the diagonal-scaling preconditioner required a much larger number of iteration steps until convergence than did the BDD and BDD-DIAG preconditioners. The difference between the results of the BDD and BDD-DIAG preconditioners was very small. Moreover, the number for the diagonal-scaling preconditioner oscillated depending on the computation step. In contrast, the BDD and BDD-DIAG preconditioners showed relatively constant but gradually increasing numbers of CG iteration steps, as the bulky structure approached a thin-walled one. One of the reasons why the numbers were not constant may be the dimensions of the coarse space. A subdomain can be divided into multiple regions by inactive elements, leading to the increase of the coarse space dimensions. Finally, the measured


(A) At the 10th computation step.


(B) At the 20th computation step.


(C) At the 30th computation step.

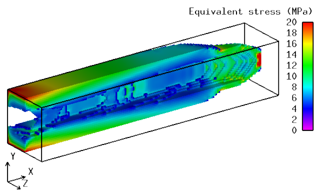
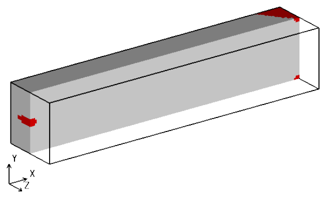
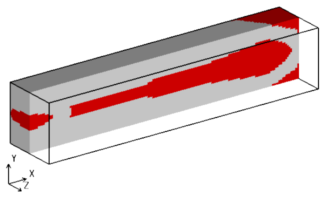
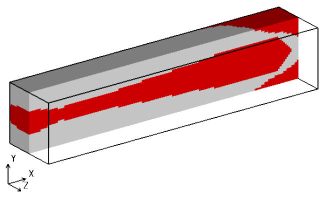

(D) At the 40th computation step.

FIGURE 20 Distributions of the equivalent stress at the 10th, 20th, 30th and 40th steps in the topology computation.


(A) At the 10th incremental step.


(B) At the 20th incremental step.


(C) At the 30th incremental step.

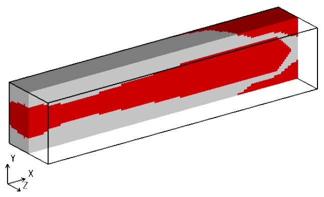

(D) At the 40th incremental step.

FIGURE 21 Inactive elements (red) at the 10th, 20th, 30th and 40th steps in the topology computation, before solving each linear system of equations.

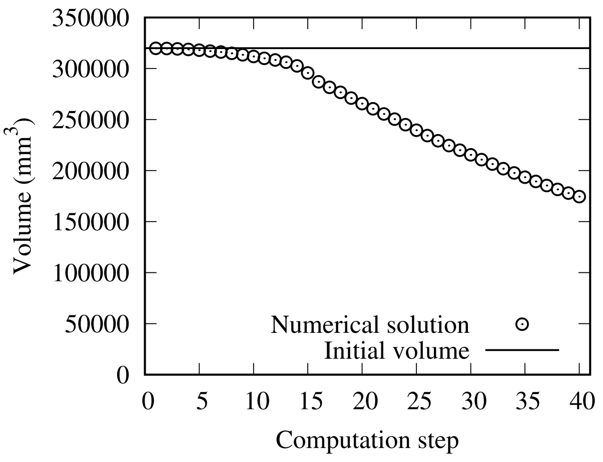


FIGURE 22 Change in volume in the topology computation.

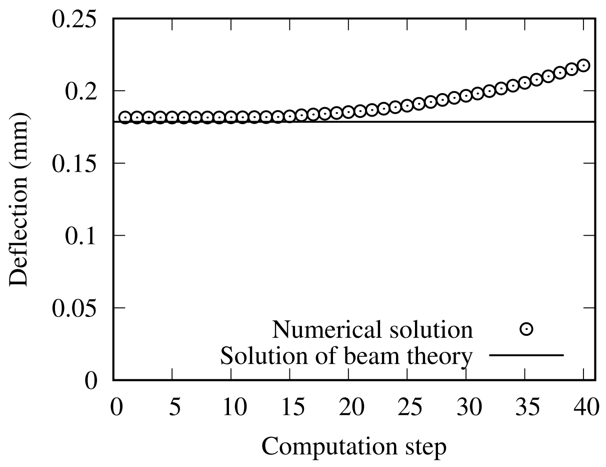


FIGURE 23 Change in deflection in the topology computation.

computational times for the diagonal-scaling, BDD and BDD-DIAG preconditioners for all the computation steps were 124 s, 207 s and 161 s, respectively.

3.4 | Thermal elastic–plastic analysis of a metal additive manufacturing problem

Next, a thermal elastic–plastic analysis of a metal AM problem is presented. This analysis was performed by the authors' DD-based thermal elastic–plastic welding solver[40] after program modification. The problem was a plate additively manufactured on a substrate, in which only the first layer of the manufacturing process was analyzed. Manufacturing conditions and parameters in the analysis were set by referring to Chen et al.[48]. Figure 26 illustrates the dimensions of the plate and the substrate. We adopted a substrate length and width that were smaller than the actual ones, although the thickness was the same as the actual thickness.

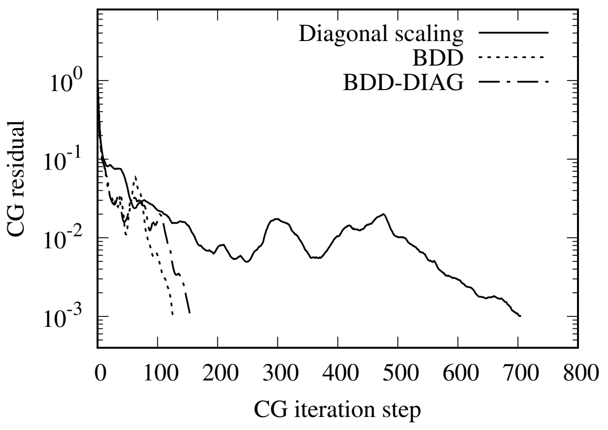


FIGURE 24 CG residuals in the topology computation at the final computation step.

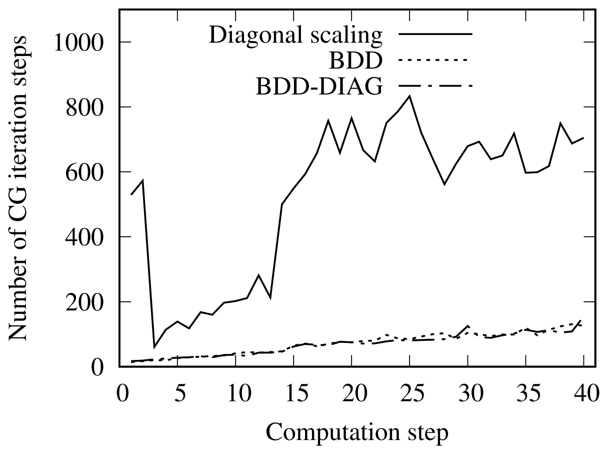


FIGURE 25 Change in the number of CG iteration steps until convergence in the topology computation.

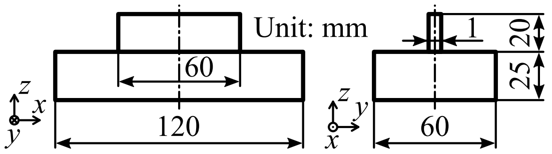


FIGURE 26 Dimensions of the plate manufactured on the substrate for the metal AM analysis.

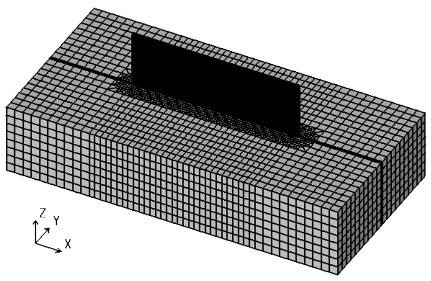


FIGURE 27 Finite element analysis model of the plate manufactured on the substrate.

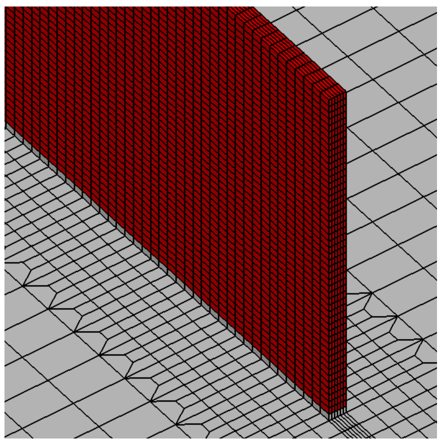


FIGURE 28 Inactive elements (red) in the metal AM analysis.

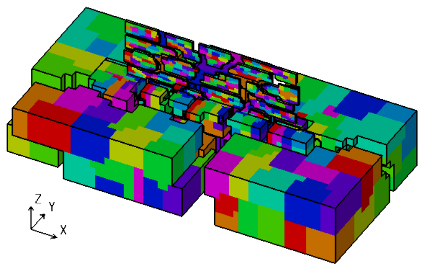


FIGURE 29 Two-level domain decomposition of Fig. 27.

Figure 27 shows the finite element analysis model of the plate and the substrate. Most of the plate was modeled by inactive elements, which are red in Fig. 28. The numbers of elements and nodes were 44,100 and 51,478, respectively. Hexahedral finite elements based on the B-bar method[44] were used. The length, thickness and height of each element at the plate were 0.667 mm, 0.167 mm and 0.4 mm, respectively. Note that, although the actual height of each layer of the manufacturing process is 0.04 mm, we adopted 0.4 mm for simplicity. The finite element analysis model was decomposed by ADVENTURE_Metis[8] into parts and subdomains, as depicted in Fig. 29. The number of parts (MPI processes) and subdomains were 24 and 600, respectively. The aforementioned computer cluster was used.

A heat conduction analysis was performed, and the results were then used in a thermal elastic–plastic analysis. In the heat conduction analysis, the heat conduction equation was discretized for the temporal direction by the backward Euler method with

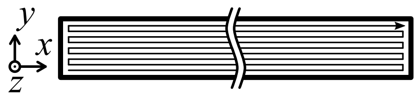


FIGURE 30 Zigzag manner in which the heat source moved.

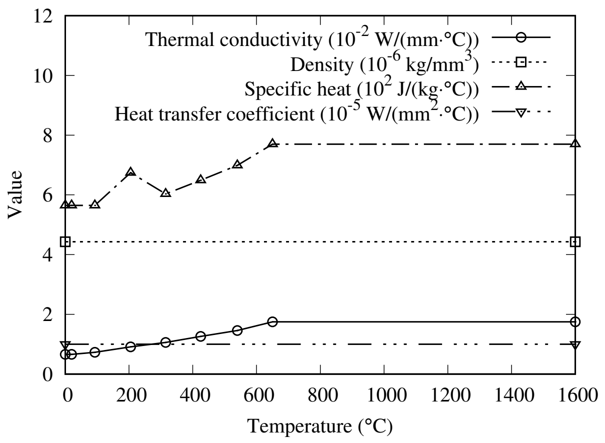


FIGURE 31 Thermal material parameters of Ti-6Al-4V[30].

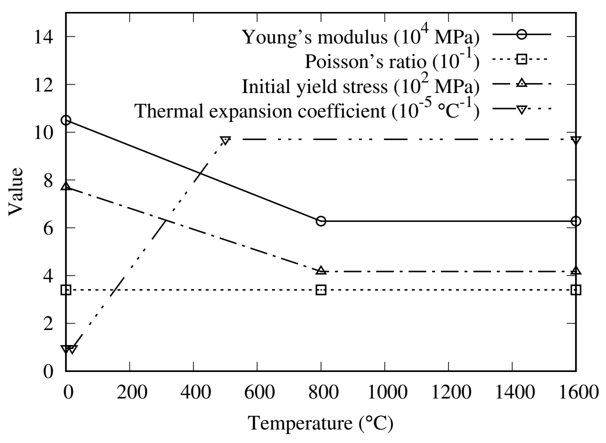


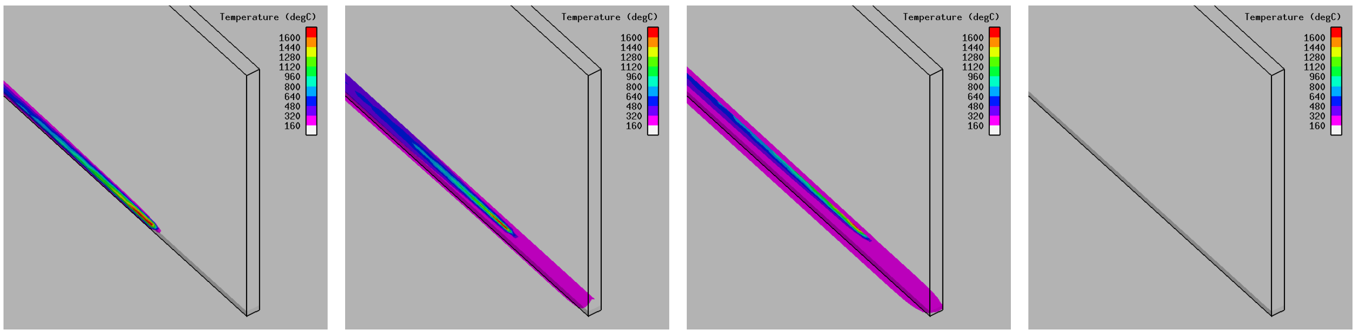
FIGURE 32 Mechanical material parameters of Ti-6Al-4V[30].

the lumped heat capacity matrix[24], and then solved. The initial and room temperatures were set to 25°C. A heat transfer boundary condition was applied to every surface of the analysis model, including the interfaces between the active and inactive elements. As the heat source model, a hemisphere uniformly heated at 300 W was used. The radius of the hemisphere was set to 0.15 mm, which is equal to the laser spot radius. This heat source moved at 1,000 mm/s along the length direction in a zigzag manner, as illustrated in Fig. 30. The hatch spacing was 0.1 mm. For the quadrature related to the moving heat source, an octree-based element subdivision technique[49] was used. After the elements were subdivided three times, one-point Gaussian quadrature was applied to each subdivision. The material was assumed to be Ti-6Al-4V. Values of the heat conductivity, density, specific heat and heat transfer coefficient[30] are plotted in Fig. 31. The horizontal and vertical axes represent the temperature and the value of the temperature-dependent material parameters, respectively. The time increment in the heating process was set to 0.000 5 s. In the cooling process, the time increment changed depending on a nodal temperature that was maximum for the spatial direction. If the change in the spatially maximum nodal temperature was lower than 5°C, the time increment was doubled. If it was higher than 20°C, the time increment was halved. The cooling process terminated when every nodal temperature fell below the room temperature plus 0.001°C.

The nodal temperatures computed by the heat conduction analysis were modified before the thermal elastic–plastic analysis. When a nodal temperature that was maximum for the spatial direction changed by more than 100°C in one time increment, the time step was subdivided by linear interpolation, for stable computation of the thermal elastic–plastic analysis. The subdivided time steps are referred to as temperature steps in the present study. In addition, nodal temperatures that were higher than the melting temperature of 1,600°C were reduced to 1,600°C.

In the thermal elastic–plastic analysis, the equilibrium equation was solved. Only the rigid-body motion was constrained by the boundary conditions. An elastic–plastic body with the von Mises yield criterion and the linear isotropic hardening law were assumed as the material model. The values of Young's modulus, Poisson's ratio, the initial yield stress and the thermal expansion coefficient[30] are plotted in Fig. 32. The horizontal and vertical axes are similar to those in Fig. 31. The linear hardening coefficient was assumed to be one-hundredth of Young's modulus.

Even when most of the plate consisted of inactive elements, converged solutions for all the time and temperature steps were obtained successfully. The numbers of time and temperature steps were 1,206 and 1,617, respectively. Figure 33 visualizes the distributions of the temperature at times 0.05 s, 0.29 s, 0.53 s and 33,557 s. In the heating process, the temperature at and behind the moving heat source was high. After the heating process, the temperature decreased until 33,557 s. Figure 34 shows the distributions of the stress x at the same times as shown in Fig. 33. Stress behind the moving heat source was tensile, whereas that in its vicinity was compressive. After the cooling process, residual stress at the plate was completely tensile.



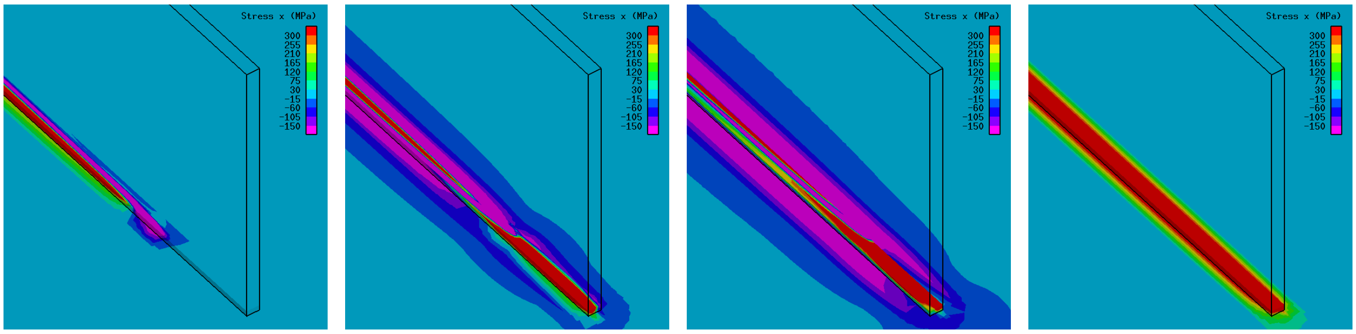
(A) At time 0.05 s (100th time step).

(B) At time 0.29 s (580th time step).

(C) At time 0.53 s (1,060th time step).

(D) At time 33,557 s (1,206th time step).

FIGURE 33 Distributions of the temperature at times 0.05 s, 0.29 s, 0.53 s and 33,557 s in the metal AM analysis.



(A) At time 0.05 s (118th temperature step).

(B) At time 0.29 s (781st temperature step).

(C) At time 0.53 s (1,467th temperature step).

(D) At time 33,557 s (1,617th temperature step).

FIGURE 34 Distributions of the stress x at times 0.05 s, 0.29 s, 0.53 s and 33,557 s in the metal AM analysis.

Newton–Raphson and CG residuals of the DD solvers with the diagonal-scaling, BDD and BDD-DIAG preconditioners at time 0.05 s are plotted in Figs. 35, 36 and 37, respectively. In the three figures, the horizontal axis represents the CG iteration step, whereas the vertical axis represents the Newton–Raphson or CG residual, respectively. The dashed lines denote the Newton–Raphson residuals, whereas the solid lines denote the CG residuals. The tolerances for the Newton–Raphson and CG residuals were set to 10^{-4} and 10^{-3}, respectively. We used a relative tolerance control technique[50], in which the CG tolerance is larger than the Newton–Raphson tolerance for reducing the number of total CG iteration steps until Newton–Raphson convergence. Note that this sort of tolerance control technique degrades the quadratic convergence of the Newton–Raphson method in general. All three preconditioners required four repetitions of the linear system solution for Newton–Raphson convergence. For each linear system solution, the numbers of required CG iteration steps of the diagonal-scaling, BDD and BDD-DIAG preconditioners were 58–577, 13–18 and 26–28, respectively. Figure 38 shows the number of Newton–Raphson iteration steps until convergence. The horizontal and vertical axes represent the temperature step and the number of Newton–Raphson iteration steps until convergence, respectively. The differences among the three preconditioners were very small. The accumulated numbers of Newton–Raphson iteration steps for the diagonal-scaling, BDD and BDD-DIAG preconditioners were 5,239, 5,227 and 5,218, respectively. At each Newton–Raphson iteration step, a linear system of equations was solved. The numbers of CG iteration steps for the linear system solutions are summarized in Figs. 39, 40 and 41. In the three figures, the horizontal and vertical axes represent the number of CG iteration steps until convergence and its frequency, respectively. The histogram bin width in Fig. 39 is 10, whereas that in Figs. 40 and 41 is 1. The mean numbers of CG iteration steps were 232, 15.6 and 27.5, respectively. Moreover, the number for the diagonal-scaling preconditioner varied widely, whereas those for the BDD and BDD-DIAG preconditioners did not. This indicates that the convergence performance of the diagonal-scaling preconditioner was greatly affected by the change in element stiffness matrices due to plasticity, whereas those of the BDD and BDD-DIAG preconditioners were not. Furthermore, the numbers of CG iteration steps for the BDD preconditioner were obviously smaller than those for

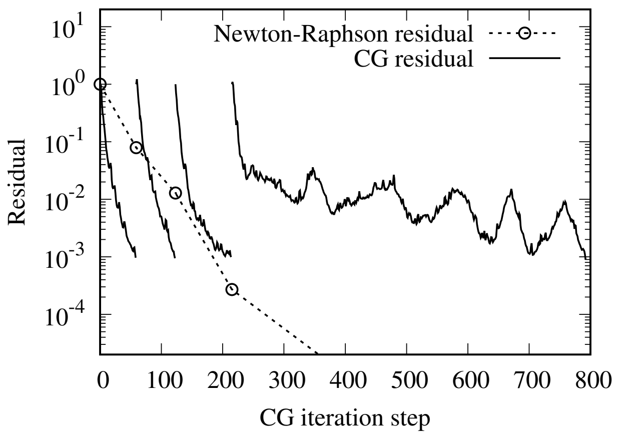


FIGURE 35 Newton–Raphson and CG residuals in the metal AM analysis by the DD solver with the diagonal-scaling preconditioner.

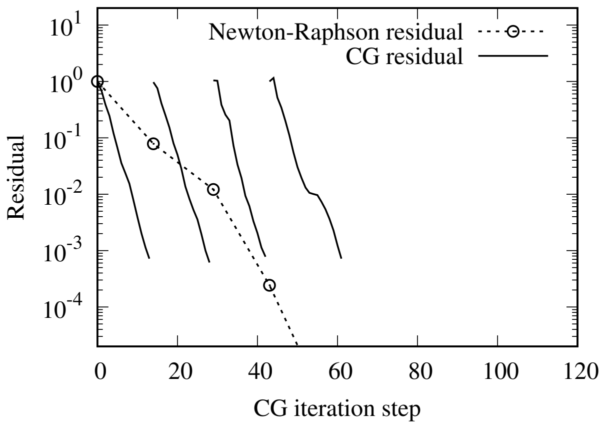


FIGURE 36 Newton–Raphson and CG residuals in the metal AM analysis by the DD solver with the BDD preconditioner.

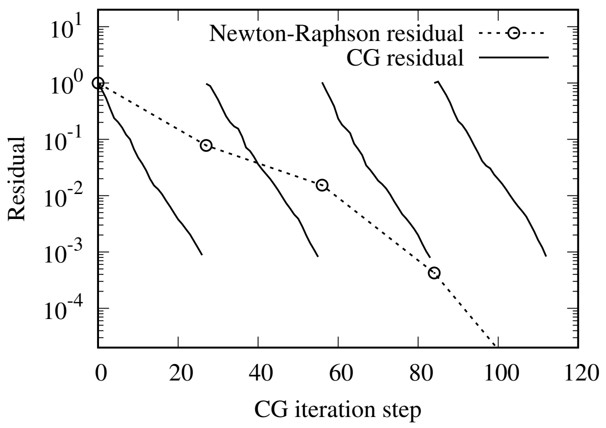


FIGURE 37 Newton–Raphson and CG residuals in the metal AM analysis by the DD solver with the BDD-DIAG preconditioner.

the BDD-DIAG preconditioner. This tendency is different from § 3.1, § 3.2 and § 3.3. In this example, the Neumann–Neumann preconditioning part would contribute to the convergence performance more significantly than the diagonal-scaling preconditioning part. Finally, the measured computational times for the diagonal-scaling, BDD and BDD-DIAG preconditioners for all the temperature steps were 1,900 s, 2,443 s and 2,653 s, respectively.

4 | CONCLUSION

An implementation method for DD solvers with diagonal-scaling, BDD and BDD-DIAG preconditioners with inactive elements was proposed. We formulated the matrix–vector multiplication, the preconditioning and the vector operations in the CG algorithm, along with the inactive elements and floating DOFs caused by the inactive elements. Consideration of the inactive elements is enabled by the slight modifications of matrices and vectors in the algorithm. The present implementation method does not require regeneration of the domain-decomposed analysis model during the analysis. In the numerical examples, the present implementation method for the BDD and BDD-DIAG preconditioners with the inactive elements exhibited scalability for the number of CG iteration steps until convergence. Moreover, the capability of the method for damage analysis, topology computation and thermal elastic–plastic analysis of metal additive manufacturing problems was demonstrated. In all of the numerical examples, the developed implementation method achieved good convergence performance.

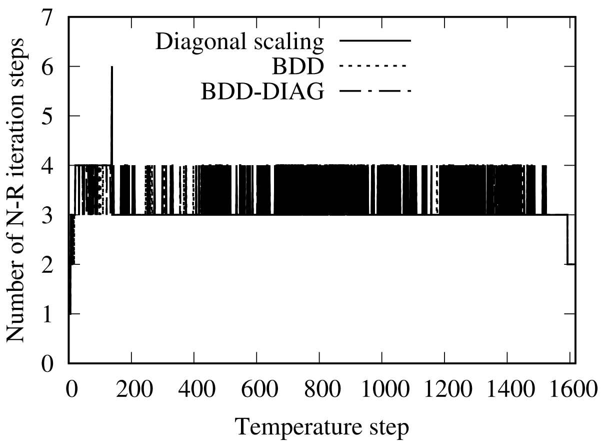


FIGURE 38 Change in the number of Newton–Raphson iteration steps until convergence in the metal AM analysis.

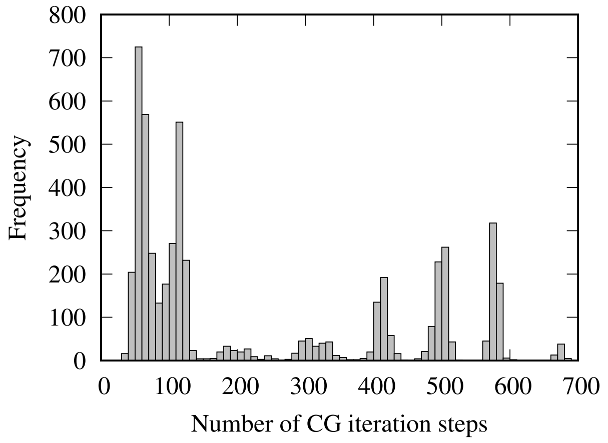


FIGURE 39 Number of CG iteration steps until convergence in the metal AM analysis with the diagonal-scaling preconditioner.

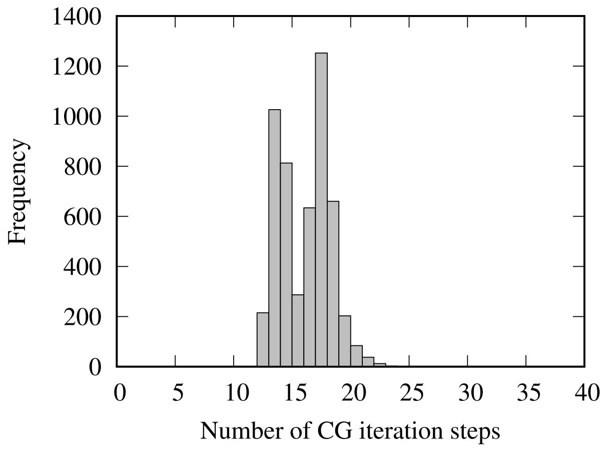


FIGURE 40 Number of CG iteration steps until convergence in the metal AM analysis with the BDD preconditioner.

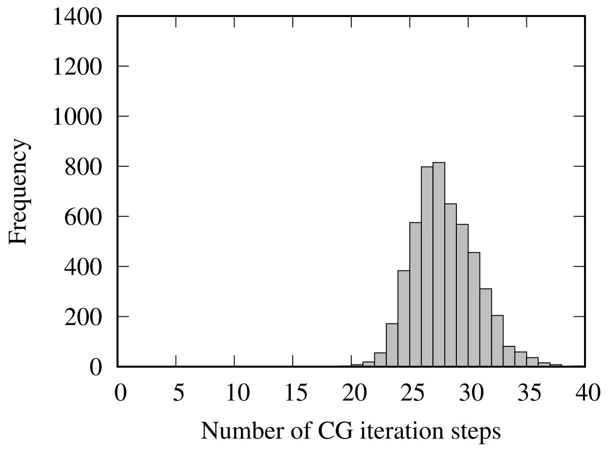


FIGURE 41 Number of CG iteration steps until convergence in the metal AM analysis with the BDD-DIAG preconditioner.

Using the present implementation method, we are planning to analyze the fatigue damage problem of offshore wind turbine blades in a research project. Each blade consists of laminated composites, which require a large number of elements to capture the fatigue damage propagation. Furthermore, analysis of metal AM of realistic mechanical parts is planned. This problem also needs many elements to simulate the manufacturing process involving a moving heat source.

ACKNOWLEDGMENT

The present study was supported by the Grants-in-Aid for Scientific Research of the Japan Society for the Promotion of Science (Grant Number JP19K14871).

References

1. Toselli A, Widlund O. *Domain Decomposition Methods: Algorithms and Theory*. Berlin Heidelberg: Springer . 2005. https://doi.org/10.1007/b137868

2. Mathew TPA. *Domain Decomposition Methods for the Numerical Solution of Partial Differential Equations*. Berlin Heidelberg: Springer . 2008. https://doi.org/10.1007/978-3-540-77209-5

3. Mandel J. Balancing domain decomposition. *Commun Numer Methods Eng* 1993; 9(3): 233–241. https://doi.org/10.1002/cnm.1640090307

4. Dohrmann C. A preconditioner for substructuring based on constrained energy minimization. *SIAM J Sci Comput* 2003; 25(1): 246–258. https://doi.org/10.1137/S1064827502412887

5. Farhat C, Roux FX. A method of finite element tearing and interconnecting and its parallel solution algorithm. *Int J Numer Methods Eng* 1991; 32(6): 1205–1227. https://doi.org/10.1002/nme.1620320604

6. Farhat C, Lesoinne M, LeTallec P, Pierson K, Rixen D. FETI-DP: A dual–primal unified FETI method—part I: A faster alternative to the two-level FETI method. *Int J Numer Methods Eng* 2001; 50(7): 1523–1544. https://doi.org/10.1002/nme.76

7. Yusa Y, Okada H, Yamada T, Yoshimura S. Scalable parallel elastic–plastic finite element analysis using a quasi-Newton method with a balancing domain decomposition preconditioner. *Comput Mech* 2018; 62(6): 1563–1581. https://doi.org/10.1007/s00466-018-1579-4

8. ADVENTURE Project. https://adventure.sys.t.u-tokyo.ac.jp/; Accessed 30 September 2021.

9. Yoshimura S, Shioya R, Noguchi H, Miyamura T. Advanced general-purpose computational mechanics system for large-scale analysis and design. *J Comput Appl Math* 2002; 149(1): 279–296. https://doi.org/10.1016/S0377-0427(02)00536-8

10. Ogino M, Shioya R, Kanayama H. An inexact balancing preconditioner for large-scale structural analysis. *J Comput Sci Technol* 2008; 2(1): 150–161. https://doi.org/10.1299/jcst.2.150

11. Ogino M, Shioya R, Kawai H, Yoshimura S. Seismic response analysis of nuclear pressure vessel model with ADVENTURE System on the Earth Simulator. *J Earth Simul* 2005; 2: 41–54.

12. Miyamura T, Yoshimura S, Yamada T. Feasibility study of full-scale elastic-plastic seismic response analysis of nuclear power plant. *Mech Eng J* 2019; 6(6): 19-00281. https://doi.org/10.1299/mej.19-00281

13. Bhardwaj M, Day D, Farhat C, Lesoinne M, Pierson K, Rixen D. Application of the FETI method to ASCI problems—scalability results on 1000 processors and discussion of highly heterogeneous problems. *Int J Numer Methods Eng* 2000; 47(1–3): 513–535. https://doi.org/10.1002/(SICI)1097-0207(20000110/30)47:1/3<513::AID-NME782>3.0.CO;2-V

14. Bhardwaj M, Pierson K, Reese G, et al. Salinas: A scalable software for high-performance structural and solid mechanics simulations. *Proceedings of the 2002 ACM/IEEE Conference on Supercomputing* 2002: 35. https://doi.org/10.1109/SC.2002.10028

15. PETSc. https://petsc.org/; Accessed 5 October 2021.

16. Jolivet P, Hecht F, Nataf F, Prud'homme C. Scalable domain decomposition preconditioners for heterogeneous elliptic problems. *Proceedings of the 2013 International Conference on High Performance Computing, Networking, Storage and Analysis* 2013: 80. https://doi.org/10.1145/2503210.2503212

17. Zampini S. PCBDDC: A class of robust dual-primal methods in PETSc. *SIAM J Sci Comput* 2016; 38(5): S282–S306. https://doi.org/10.1137/15M1025785

18. Klawonn A, Lanser M, Rheinbach O, Weber J. Preconditioning the coarse problem of BDDC methods—three-level, algebraic multigrid, and vertex-based preconditioners. *Electron Trans Numer Anal* 2019; 51: 432–450. https://doi.org/10.1553/etna_vol51s432

19. Trilinos Project. https://trilinos.github.io/; Accessed 5 October 2021.

20. Heinlein A, Klawonn A, Rajamanickam S, Rheinbach O. FROSch: A fast and robust overlapping Schwarz domain decomposition preconditioner based on Xpetra in Trilinos. *Domain Decomposition Methods in Science and Engineering XXV* 2020: 176–184. https://doi.org/10.1007/978-3-030-56750-7_19

21. Badia S, Martín AF, Principe J. FEMPAR: An object-oriented parallel finite element framework. *Arch Comput Methods Eng* 2018; 25(2): 195–271. https://doi.org/10.1007/s11831-017-9244-1

22. Lindgren LE, Runnemalm H, Näsström MO. Simulation of multipass welding of a thick plate. *Int J Numer Methods Eng* 1999; 44(9): 1301–1316. https://doi.org/10.1002/(SICI)1097-0207(19990330)44:9<1301::AID-NME479>3.0.CO;2-K

23. Lindgren LE, Hedblom E. Modelling of addition of filler material in large deformation analysis of multipass welding. *Commun Numer Methods Eng* 2001; 17(9): 647–657. https://doi.org/10.1002/cnm.414

24. Lindgren LE. *Computational Welding Mechanics: Thermomechanical and Microstructural Simulations*. Cambridge: Woodhead Publishing . 2007.

25. Mandel J, Brezina M. Balancing domain decomposition for problems with large jumps in coefficients. *Math Comput* 1996; 65(216): 1387–1401. https://doi.org/10.1090/S0025-5718-96-00757-0

26. Klawonn A, Widlund OB, Dryja M. Dual-primal FETI methods for three-dimensional elliptic problems with heterogeneous coefficients. *SIAM J Numer Anal* 2002; 40(1): 159–179. https://doi.org/10.1137/S0036142901388081

27. Evgrafov A, Rupp CJ, Maute K, Dunn ML. Large-scale parallel topology optimization using a dual-primal substructuring solver. *Struct Multidiscip Optim* 2008; 36(4): 329–345. https://doi.org/10.1007/s00158-007-0190-7

28. Denlinger ER, Irwin J, Michaleris P. Thermomechanical modeling of additive manufacturing large parts. *J Manuf Sci Eng* 2014; 136(6): 061007. https://doi.org/10.1115/1.4028669

29. Michaleris P. Modeling metal deposition in heat transfer analyses of additive manufacturing processes. *Finite Elem Anal Des* 2014; 86: 51–60. https://doi.org/10.1016/j.finel.2014.04.003

30. Gouge M, Michaleris P. *Thermo-Mechanical Modeling of Additive Manufacturing*. Oxford: Butterworth–Heinemann . 2018. https://doi.org/10.1016/C2016-0-00317-0

31. Neiva E, Badia S, Martín AF, Chiumenti M. A scalable parallel finite element framework for growing geometries. Application to metal additive manufacturing. *Int J Numer Methods Eng* 2019; 119(11): 1098–1125. https://doi.org/10.1002/nme.6085

32. Duranton P, Devaux J, Robin V, Gilles P, Bergheau J. 3D modelling of multipass welding of a 316L stainless steel pipe. *J Mater Process Technol* 2004; 153–154: 457–463. https://doi.org/10.1016/j.jmatprotec.2004.04.128

33. Ganeriwala RK, Hodge NE, Solberg JM. Towards improved speed and accuracy of laser powder bed fusion simulations via multiscale spatial representations. *Comput Mater Sci* 2021; 187: 110112. https://doi.org/10.1016/j.commatsci.2020.110112

34. Ikushima K, Shibahara M. Prediction of residual stresses in multi-pass welded joint using idealized explicit FEM accelerated by a GPU. *Comput Mater Sci* 2014; 93: 62–67. https://doi.org/10.1016/j.commatsci.2014.06.024

35. Mozaffar M, Ndip-Agbor E, Lin S, Wagner GJ, Ehmann K, Cao J. Acceleration strategies for explicit finite element analysis of metal powder-based additive manufacturing processes using graphical processing units. *Comput Mech* 2019; 64(3): 879–894. https://doi.org/10.1007/s00466-019-01685-4

36. Murakami S. *Continuum Damage Mechanics: A Continuum Mechanics Approach to the Analysis of Damage and Fracture (in Japanese)*. Tokyo: Morikita Publishing . 2008.

37. Bendsoe MP, Sigmund O. *Topology Optimization: Theory, Methods, and Applications*. Berlin Heidelberg: Springer . 2004. https://doi.org/10.1007/978-3-662-05086-6

38. Mrzygłód M. Two-stage optimization method with fatigue constraints for thin-walled structures. *J Theor Appl Mech* 2010; 48(3): 567–578.

39. Ogino M. Finite element analysis of multi-material models using a balancing domain decomposition method combined with the diagonal scaling preconditioner. *Trans JSME (in Japanese)* 2016; 82(833): 15-00325. https://doi.org/10.1299/transjsme.15-00325

40. Yusa Y, Murakami Y, Okada H. Large-scale parallel thermal elastic-plastic welding simulation using balancing domain decomposition method. *Proceedings of the ASME 2019 Pressure Vessels and Piping Conference* 2019: PVP2019-93237. https://doi.org/10.1115/PVP2019-93237

41. MUMPS: A parallel sparse direct solver. http://mumps.enseeiht.fr/; Accessed 30 September 2021.

42. Amestoy P, Duff I, L'Excellent J, Koster J. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM J Matrix Anal Appl* 2001; 23(1): 15–41. https://doi.org/10.1137/S0895479899358194

43. Amestoy PR, Buttari A, L'Excellent JY, Mary T. Performance and scalability of the block low-rank multifrontal factorization on multicore architectures. *ACM Trans Math Softw* 2019; 45(1): 2. https://doi.org/10.1145/3242094

44. Hughes TJR. *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. New York: Dover Publications . 2000.

45. Family of graph and hypergraph partitioning software. http://glaros.dtc.umn.edu/gkhome/views/metis; Accessed 30 September 2021.

46. Karypis G, Kumar V. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J Sci Comput* 1998; 20(1): 359–392. https://doi.org/10.1137/S1064827595287997

47. Yagawa G, Shioya R. Parallel finite elements on a massively parallel computer with domain decomposition. *Comput Syst Eng* 1993; 4(4–6): 495–503. https://doi.org/10.1016/0956-0521(93)90017-Q

48. Chen C, Xiao Z, Zhu H, Zeng X. Deformation and control method of thin-walled part during laser powder bed fusion of Ti-6Al-4V alloy. *Int J Adv Manuf Technol* 2020; 110(11): 3467–3478. https://doi.org/10.1007/s00170-020-06104-0

49. Okada H, Liu CT, Ninomiya T, Fukui Y, Kumazawa N. Analysis of particulate composite materials using an element overlay technique. *Comput Model Eng Sci* 2004; 6(4): 333–348. https://doi.org/10.3970/cmes.2004.006.333

50. Miyamura T, Noguchi H, Shioya R, Yoshimura S, Yagawa G. Elastic–plastic analysis of nuclear structures with millions of DOFs using the hierarchical domain decomposition method. *Nucl Eng Des* 2002; 212(1–3): 335–355. https://doi.org/10.1016/S0029-5493(01)00497-6

How to cite the version of record: Yusa Y, Kobayashi H, Murakami Y, Okada H. Implementation of balancing domain decomposition method for parallel finite element analysis involving inactive elements. *Int J Numer Methods Eng*. 2022; 123(17):3974-4000. doi: 10.1002/nme.6995