Journal of Engineering Mathematics (2024) 144:21
https://doi.org/10.1007/s10665-023-10322-2

Deflated domain decomposition method for structural problems

Hiroshi Akiba[1]

Received: 10 April 2023 / Accepted: 4 December 2023 / Published online: 8 February 2024

Abstract
The paper presents a fast and stable solver algorithm for structural problems. The point is the distance between the eigenvector of the constrained stiffness matrix and the unconstrained matrix. The coarse motions are close to the kernel of the unconstrained matrix. We use lower-frequency deformation modes to construct an iterative solver algorithm through domain decomposition expressing near-rigid-body motions, deflation algorithms, and two-level algorithms. We remove the coarse space from the solution space and hand over the iteration space to the fine space. Our solver is parallelized, and the solver thus has two sets of domain decomposition. One decomposition generates the coarse space, and the other is for parallelization. The basic framework of the solver is the parallel conjugate gradient (CG) method on the fine space. We use the CG method for the basic framework instead of the (simplest) domain decomposition method. We conducted benchmark tests using elastic static analysis for thin plate models. A comparison with the standard CG solver results shows the new solver's high-speed performance and remarkable stability.

Keywords Domain decomposition · Deflation algorithm · Distance between eigenvectors of stiffness matrix and kernel of unconstrained one · Fast stable solver · Rigid-body motion

Mathematics Subject Classification 15 Linear and multilinear algebra · Matrix theory · 65 Numerical analysis · 74 Mechanics of deformable solids

1 Introduction

The three-dimensional analysis is a standard technique for structural mechanics. However, it has yet to be used in the design of large-scale structures, safety assessment,

✉ Hiroshi Akiba
akiba@edu.k.u-tokyo.ac.jp

1 Graduate School of Frontier Sciences, The University of Tokyo, Kashiwanoha, Kashiwa, Chiba 277-8536, Japan

and maintenance. Instead, frame and lumped mass models have long been used owing to historical background, insufficient computer power, and computational techniques. However, in recent years, three-dimensional analyses have been gradually but widely used for nuclear power plants (NPPs) [1], high-rise buildings [2], and bridges [3]. For example, Japan's practice standard for the seismic probabilistic risk assessment (PRA) of NPPs requires a detailed evaluation of the damage limits of buildings and structures using three-dimensional seismic response analysis [4]. In a study of high-rise buildings [2], three-dimensional elastoplastic seismic response damage analysis has been conducted with as many as 74 million degrees-of-freedom (DOFs), which represents a level of detail and complexity of damage that is impossible to evaluate detailed damage through the frame and lumped mass analysis.

We need to shorten the calculation time and stability when performing three-dimensional analyses. The present study considers these two properties in the development of our solver.

It is widely known that small eigenvalues close to zero hamper the convergence of iterative methods, mainly because the condition number increases with a decrease in the smallest eigenvalue. Many studies have focused on avoiding eigenvalues close to zero [5–8]. The deflation algorithm mainly aims to eliminate eigenvectors corresponding to eigenvalues close to zero [7–14].

In the case of structural problems discretized by the finite element analysis (FEA), the stiffness matrices $\boldsymbol{A}$ are symmetric positive semi-definite. There are three parallel translations and three infinitesimal rotations, a total of six rigid-body motions, which implies that $\boldsymbol{A}$ duplicates the six zero eigenvalues. We must, therefore, impose at least six constraint conditions to eliminate the zero eigenvalues. Let $\bar{\boldsymbol{A}}$ be the stiffness matrix after imposing the constraint conditions as will be discussed in Sect. 3. The eigenvalues of $\boldsymbol{A}$ and $\bar{\boldsymbol{A}}$ are then different from each other. However, the eigenvalues of these two matrices have the so-called interlacing property [15]. The rigid-body mode is widely used in structural analysis for different objectives [10, 16–20, 22, 23].

We assume that the analysis models are constructed with solid elements. The rigid-body motions, which are the bases of $\mathrm{Ker}\boldsymbol{A}$ (the kernel of $\boldsymbol{A}$), relate to the lower or lowest eigenvalue(s) of $\bar{\boldsymbol{A}}$, owing to the interlacing property. We can construct the near rigid-body modes using domain decomposition in which the subdomains have six rigid-body modes. We can set a basis of a coarse space by aggregating the subdomains through a *partition of unity*. We use the above-noted six rigid-body motions to simulate low-frequency motions described in the literature [22, 24].

This paper discusses the distance between the eigenvectors of $\bar{\boldsymbol{A}}$ and $\mathrm{Ker}\boldsymbol{A}$. The distance describes how the lower-frequency modes are segregated from the higher modes.

In our analysis, we use the parallelized conjugate gradient (CG) method as our basic framework on the entire domain instead of the simplest domain decomposition method (DDM) [25], which means that we do not apply a direct method to the decomposed subdomains in our method. The performance of the DDM and CG methods are compared in the next section, showing the reason for using the CG method.

Our solver uses two sets of domain decomposition. One decomposition is for parallelizing the CG method, and the other is for generating the coarse space. The former corresponds to the subdomains of the DDM, and the latter generates the projection in the deflation algorithm.

Near-rigid-body motions generate a projection from the entire space to the coarse space. The projection space is so small that the reduced equation can be solved using direct methods. In contrast, we can apply the CG method to the large complementary space, which is much easier to solve than the entire space containing "rough" components generated by lower-frequency modes. Given the direct sum decomposition of the solution space, our method is a two-level method [13, 26, 27].

Our definition of the projection constructs by the coarse motion, which is in contrast with definitions adopted in many studies (e.g., [7–14]), which will be described in Sect. 5.2.

The deflated domain decomposition method (DDDM) algorithm, which we describe in this paper, shares similarities with the algorithm given in [7] that expands the Krylov subspace, adding approximated eigenvectors corresponding to eigenvalues close to zero using the orthogonality of the residuals, which we describe in Sect. 6.

The DDDM solver outperforms a successive symmetric overrelaxation (SSOR) preconditioned solver, which we assume is a standard solver, as will be discussed in Sect. 6.3. Also, in an article [28], an elastic seismic response analysis of an NPP building installed on the ground consisting of one million DOFs with hexahedral and plate elements involving 2700 steps for 54 s took 1.1 h using the DDDM solver. In contrast, the SSOR solver took 13.8 h. We used sixteen parallel processes in both cases. The stability of the DDDM solver compared to the SSOR solver concerning the tolerance range of the convergence was also discussed in this article.

2 Basic framework of the linear solver

2.1 Displacement-based finite element equations

We outline the FEA method used in the paper. Only a static equilibrium state is assumed. We start from the principle of virtual work. We omit the commonly used isoparametric discretization here. Refer to the detailed discussion in [21].

(a) Discretization

We only consider the solid elements. We approximate the target body as an assembly of a finite number of discrete finite elements, e.g., tetrahedral, hexahedral, or other elements. Each element m has nodal points x, y, z in a local coordinate system. Let the number of DOFs be n.

We use indicial notation and summation convention. x_i denotes the coordinate axis, where $i = 1, 2, 3$ in the three-dimensional analysis given to each nodal point and u_i denote the displacement components, where $i = 1, 2, 3$.

(b) Principle of Virtual Work

We take virtual displacements $\bar{u}_i$ and take the corresponding virtual strains $\bar{\epsilon}_{ij}$, differentiating $\bar{u}_i$. We then have the following principle of virtual displacements, which is the fundamental equation for the equilibrium of a general three-dimensional body:

$$\int_{\Omega} \tau_{ij}\bar{\epsilon}_{ij} d\Omega = \int_{\Omega} X_i \bar{u}_i d\Omega + \int_{S} Y_i \bar{u}_i^{(S)} \mathrm{d}S, \tag{1}$$

where X_i is the components of the body force $\boldsymbol{X}$, and Y_i is the components of the surface traction $\boldsymbol{Y}$ on the surface S of the body. $u_i^{(S)}$ is the displacements on the surface S. The displacement (essential) boundary conditions are given by

$$u_i = u_i^{(S)}. \tag{2}$$

The natural boundary condition is given by

$$\tau_{ij}\boldsymbol{n}_j = Y_i, \tag{3}$$

on the surface S, where $\boldsymbol{n}_j$ is the unit normal vector to the surface S. The left side of (1) corresponds to the internal virtual work, and the right side represents the external virtual work.

(c) Finite element equations

Let $\hat{\boldsymbol{u}}$ be the unknown vector of all the global displacements, where the unknown displacements $u_i, i = 1, 2, 3$ of the nodal points are globally aligned.

$$\hat{\boldsymbol{u}}^T = (\hat{u}_1\ \hat{u}_2\ \hat{u}_3\ \cdots\ \hat{u}_n). \tag{4}$$

Let $\boldsymbol{N}^{(m)}$ be a displacement interpolation matrix and $\boldsymbol{B}^{(m)}$ be the strain–displacement matrix for element m. We write

$$\begin{aligned} \boldsymbol{u}(x, y, z) &= \boldsymbol{N}^{(m)}(x, y, z)\hat{\boldsymbol{u}}, \\ \boldsymbol{\epsilon}(x, y, z) &= \boldsymbol{B}^{(m)}(x, y, z)\hat{\boldsymbol{u}}. \end{aligned} \tag{5}$$

We assume the virtual displacements and strains are

$$\begin{aligned} \bar{\boldsymbol{u}}(x, y, z) &= \boldsymbol{N}^{(m)}(x, y, z)\bar{\hat{\boldsymbol{u}}}, \\ \bar{\boldsymbol{\epsilon}}(x, y, z) &= \boldsymbol{B}^{(m)}(x, y, z)\bar{\hat{\boldsymbol{u}}}. \end{aligned} \tag{6}$$

We further need a relation between $\boldsymbol{\epsilon}$ and $\boldsymbol{\tau}$

$$\boldsymbol{\tau}^{(m)}(x, y, z) = \boldsymbol{D}^{(m)}(x, y, z)\boldsymbol{\epsilon}^{(m)}(x, y, z), \tag{7}$$

where $\boldsymbol{D}^{(m)}$ is the elasticity matrix of element m. Substituting $\boldsymbol{u}^{(m)}$, $\boldsymbol{\epsilon}^{(m)}$, $\bar{\boldsymbol{u}}^{(m)}$, $\bar{\boldsymbol{\epsilon}}^{(m)}$, and $\boldsymbol{\tau}^{(m)}$ into (1), we have a summation form of the virtual equation for the unknown displacements:

$$\begin{aligned} &\bar{\hat{\mathbf{u}}}^T \left(\sum_m \int_{\Omega^{(m)}} \boldsymbol{B}^{(m)^T} \boldsymbol{D}^{(m)} \boldsymbol{B}^{(m)} d\Omega \right) \hat{\boldsymbol{u}} \\ &= \bar{\hat{\mathbf{u}}}^T \left(\sum_m \int_{\Omega^{(m)}} \boldsymbol{N}^{(m)^T} \boldsymbol{X}^{(m)} d\Omega^{(m)} + \sum_m \int_{S^{(m)}} \boldsymbol{N}^{(m)^T} \boldsymbol{Y}^{(m)} dS^{(m)} \right). \end{aligned} \tag{8}$$

This equation leads to the finite element equation for the displacements $\hat{\boldsymbol{u}}$ of DOFs of the discretized entire body:

$$\boldsymbol{K}\hat{\boldsymbol{u}} = \boldsymbol{F}. \tag{9}$$

The second term of (8) on the right side corresponds to the constraint condition of the linear equation, which we will describe in the following sections, ensuring this equation is solvable.

As noted above, the explanation here describes the static equilibrium state. However, we can describe the dynamic or nonlinear dynamic state similarly. The corresponding dynamic equation is:

$$\boldsymbol{M}\ddot{\hat{\boldsymbol{u}}} + \boldsymbol{C}\dot{\hat{\boldsymbol{u}}} + \boldsymbol{K}\hat{\boldsymbol{u}} = \boldsymbol{F}, \tag{10}$$

where $\boldsymbol{C}$ is a damping matrix. In many cases of the FEA, the dynamic equation is solved using, e.g., Newmark's β method, which converts the dynamic equation to the static equation in the form of (9). The nonlinear equation is linearized by, e.g., Newton's method and is reduced to the linear equations.

We use the symbol $\boldsymbol{A}$ for the stiffness matrix corresponding to $\boldsymbol{K}$ in (9), and we focus on the linear equation in this paper:

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \quad \boldsymbol{A} \in \mathbf{R}^{n\times n}, \quad \boldsymbol{b} \in \mathbf{R}^{n}, \quad \boldsymbol{x} \in \mathbf{R}^{n}. \tag{11}$$

2.2 Comparison of the DDM and CG method

As noted in Sect. 1, In our DDDM algorithm, we use the CG method on almost all the solution space instead of the DDM. The DDM refers to the simplest iterative substructuring method using a domain decomposition, which overlaps only with the boundaries of the neighboring subdomains. The direct methods are applied on each subdomain inside the boundaries with the displacement boundary condition on the boundaries. It is not easy to compare the performance of the DDM with that of the CG method, even though the two methods were compared in previous work [29], wherein the advantages of the CG method were demonstrated under some conditions. However, the performances depend on the models, elements, boundary conditions, multi-point constraint (MPC)s, materials, and especially the sparsity of the stiffness matrix. We compare the two methods through simple static analysis using simple models.

Note that the results in this section do not have generality. The objective of this section is to show that our restricted but simple and standard examples have better performances than those using the DDM and the reason we take the CG method on the entire space. Significantly, the simple structure of the CG method helps the DDDM algorithm.

We used the open-source structural analysis code, Adventure [30], to compare the DDM and CG methods. Adventure has options of (a) the simplest DDM with the diagonal scaling, (b) the balancing domain decomposition (BDD) method, and (c) the parallel CG method in its simplest form with the diagonal scaling. Here, we used options (a) and (c). The CG method is parallelized, but only a single process was assumed. The CG method used the same code in the DDM, which could thus be fairly compared with the DDM. The computer used was a cluster computer with an Intel Xeon Platinum 9242, operating at 2.3 GHz, six nodes with 96 core processors per node, 384 GB of RAM (16 GB DDR4-2933 $\times$ 24), and Infiniband EDR networking capability (100 Gbps). We assume a single thread for all the following cases.

We use a plate model built by arranging hexahedral linear elements having dimensions of $1\text{mm} \times 1\text{mm} \times 1\text{mm}$ as shown in Fig. 1. We conducted cantilever-type analyses as follows. The x, y, and z elements align in the directions of the x, y, and z-axes, respectively. We refer to this plate as "$x \times y \times z$." We use standard steel with Young's modulus of 200 GPa and Poisson ratio of 0.3 as the material. The plate was rotated 90° around the x axis in the direction of $-y$. This plate's xy surface ($z = 0$) was then constrained fully, and we applied a dead weight in the direction of $-y$. y corresponds to the thickness of the plate, whereas z corresponds to the length of the plate.

The eight models had dimensions from $50 \times 5 \times 50$ up to $200 \times 10 \times 200$, as shown in Table 1.

We used up to 16 cores in the calculation. For each model, we conducted the analyses of the DDM with 2, 4, 8, and 16 processes with the same number of processor cores. The results were compared with those of the CG method without parallelization, resulting in five parallel cases. The elapsed times until convergence, measured in wall-clock time, are compared in Table 1. We set the tolerance value of the residual error for the convergence to 1.0×10^{-7}.

There are irregularities in Table 1 in the DDM analysis results. For example, the calculation time increases from two to four processes for the $50 \times 5 \times 50$ and $100 \times 5 \times 100$ models. The $100 \times 5 \times 100$ model in Fig. 2 shows this state of irregularity.

2.3 Basic framework of the linear solver

The previous section showed that the CG method can be a basic framework. The well-known solver algorithm FETI (Finite Element Tearing Interconnecting) [17, 18], BDD [19, 20] both depend on the DDM in the sense that the entire domain is decomposed into non-overlapping subdomains, a direct method is applied to each of subdomain,

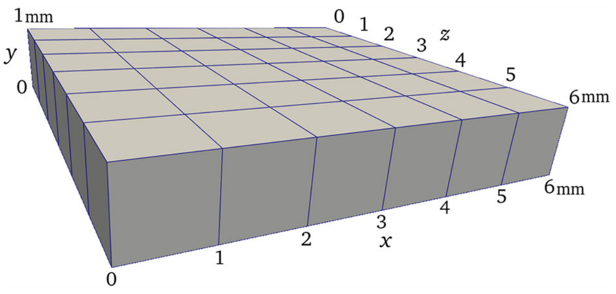

Fig. 1 Thin-plate model with the dimensions of 6mm $\times$ 1mm $\times$ 6mm. We refer to it as the "$6 \times 1 \times 6$" model, used in comparing the performance of the CG method and DDM for cantilever-type dead weight analysis. We use this model in later sections in various dimensions

Table 1 Performances of the CG method and DDM. The DDM is parallelized with 2, 4, 8, and 16 processes, whereas the CG method uses a single process. The single process CG method outperforms the DDM

Model	No of elements	DOFs	CG[s]	DDM[s][1]			
				2	4	8	16
$50 \times 5 \times 50$	12,500	46,818	1.56	2.32	2.68	0.89	0.54
$50 \times 10 \times 50$	25,000	85,833	2.81	10.16	8.04	3.51	1.48
$100 \times 5 \times 100$	50,000	183,618	18.38	30.08	33.14	18.40	8.28
$100 \times 10 \times 100$	100,000	336,633	32.24	158.60	144.83	54.60	23.39
$150 \times 5 \times 150$	112,500	410,418	84.85	154.72	150.65	85.82	40.59
$150 \times 10 \times 150$	225,000	752,433	104.71	941.36	512.29	149.66	120.00
$200 \times 5 \times 200$	200,000	727,218	231.55	1,104.87	435.22	206.99	138.00
$200 \times 10 \times 200$	400,000	1,333,233	293.98	–[2]	2,468.50	948.40	391.33

[1] The tolerance value of the residual error for the convergence is 1.0×10^{-7}
[2] Out of memory error

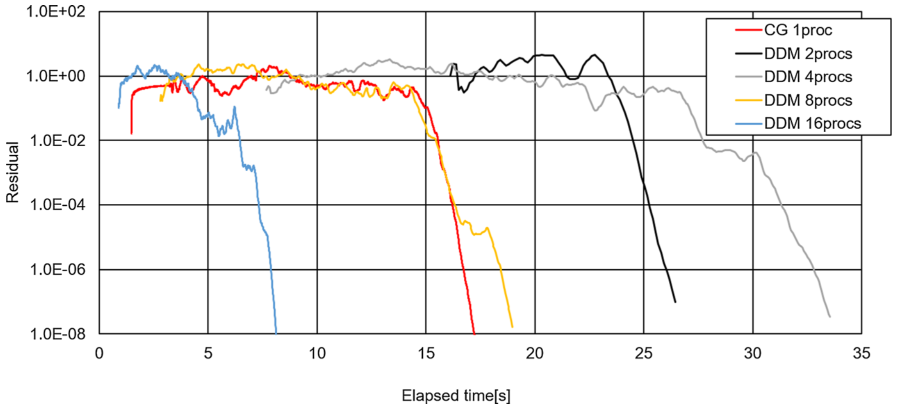

Fig. 2 Typical convergence curves of the 100 × 5 × 100 model. The single-process CG method is faster than the DDM with two, four, and eight processes

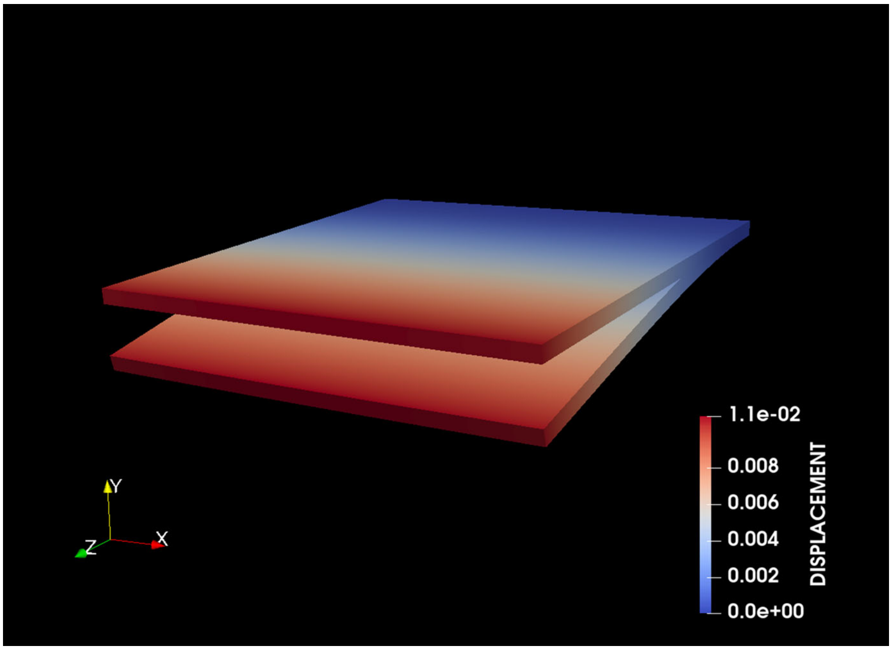

Fig. 3 Typical analysis result—displacement of the 150 × 5 × 150 model. Color represents the displacement norms. We applied the deformation factor of 1000

and the solution is divided into the rigid-body motion and the motion that includes the strain. The rigid-body motions of the subdomains are solved globally. The subdomains are taken to be structural objects. The FETI method takes the surface traction for the unknowns on the inner boundaries between the neighboring subdomains. In contrast, the BDD method takes the displacement between the boundaries of the neighboring subdomains. The CG method removes the gaps in the displacements between the boundaries in the FETI method and the gaps in the surface traction between the boundaries in the BDD method.

As noted in Sect. 1, our approach uses the CG method on the entire space in contrast to the DDM. Our solver uses two classes of domain decomposition. One parallelizes the CG method, whereas the other generates the coarse space based on the rigid-body motions. The number of subdomains for the parallel CG method corresponds to the number of parallel processes. The number of the subdomains used in the deflation algorithm should be much greater than the number used in the parallel CG method, from our experience.

In the FETI and BDD methods, the rigid-body motions are used to build the approximated motions of the subdomains in the iteration processes. Meanwhile, in our method, We use the rigid-body motions to construct lower-frequency modes.

2.4 DDDM: deflated domain decomposition method

The strategy of our algorithm is summarized as follows.

(a) We apply the parallel CG method to the entire domain as the basic framework, and we decompose the entire domain into subdomains that correspond to the parallel processes. The domain decomposition is the “nodal-point” base decomposition.
(b) Separately and independently from the domain decomposition in a), the entire domain is decomposed into some non-overlapping subdomains, which approximate lower-frequency modes (i.e., the coarse grid modes) using the rigid-body motions. The domain decomposition is the “element” base decomposition.
(c) The deflation algorithm removes the lower-frequency components from the solution based on the domain decomposition b). As will be explained in Sect. 4, there are lower-frequency eigenvectors close to the kernel of the unconstrained stiffness matrix, and we construct the lower-frequency modes using the basis of the kernel. Removal of the lower-frequency modes gives rise to high-speed performance and stability of the solver.
(d) In the parallel CG method, we distribute the coarse grid motion generation process into the parallel processes of the CG method.

3 Rewriting of the stiffness matrix

Let $\boldsymbol{A} \in \mathbf{R}^{n\times n}$ be a stiffness matrix without constraint conditions, i.e., a symmetric semi-positive definite matrix. $\boldsymbol{A}$ includes six zero eigenvalues and six corresponding eigenvectors, namely rigid-body motions. Our problem is to solve

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \tag{12}$$

by imposing necessary constraint conditions. Let $r \geq 6$ be the number of constraint conditions. By relocating the DOF numbers of the constraint conditions to the upper position and taking the first r DOFs as the constraint conditions in ascending order, $\boldsymbol{A}$ is rewritten as:

$$A' = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \bar{A} \end{pmatrix} \in \mathbf{R}^{n\times n}. \tag{13}$$

where $\bar{A}$ is a symmetric positive definite matrix. We rewrite (12) in the form:

$$A'x' = b', \tag{14}$$

where[1]

$$x' = \begin{pmatrix} \mathbf{0} \\ x_{r+1} \\ \vdots \\ x_n \end{pmatrix} \in \mathbf{R}^n, \quad b' = \begin{pmatrix} \mathbf{0} \\ b_{r+1} \\ \vdots \\ b_n \end{pmatrix} \in \mathbf{R}^n. \tag{15}$$

We further rewrite the components $x_{r+1}, \dots, x_n$ as $x_1, \dots, x_{n-r}$ and rewrite $b_{r+1}, \dots, b_n$ as $b_1, \dots, b_{n-r}$. We thus write

$$x' = \begin{pmatrix} \mathbf{0} \\ \bar{x} \end{pmatrix}, \quad \bar{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_{n-r} \end{pmatrix}; \quad b' = \begin{pmatrix} \mathbf{0} \\ \bar{b} \end{pmatrix}, \quad \bar{b} = \begin{pmatrix} b_1 \\ \vdots \\ b_{n-r} \end{pmatrix}. \tag{16}$$

The original constrained components $x_1, \dots, x_r$ are included in $\bar{b}$ in the form of a product sum of $\bar{b}$ and the first to r-th components of Ax in (12). We identify $\bar{x} \in \mathbf{R}^{n-r}$ and $x' \in \mathbf{R}^n$ given above to evaluate the distance between $\bar{x}$ and the kernel of A in the following sections. Our problem (12) is then rewritten as

$$\bar{A}\bar{x} = \bar{b}. \tag{17}$$

These procedures give a structure of $\mathbf{R}^{n-r}$ as a subspace of $\mathbf{R}^n$.

Let

$$\lambda_1 = \cdots = \lambda_6 = 0, \quad (0 <)\, \lambda_7 \le \cdots \le \lambda_n \tag{18}$$

be the eigenvalues of A, and let

$$(0 <)\, \bar{\lambda}_1 \le \cdots \le \bar{\lambda}_{n-r} \tag{19}$$

be the eigenvalues of $\bar{A}$.

[1] We can also define

$$A' = \begin{pmatrix} \bar{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}, \quad x' = \begin{pmatrix} x_1 \\ \vdots \\ x_{n-r} \\ \mathbf{0} \end{pmatrix}, \quad b' = \begin{pmatrix} b_1 \\ \vdots \\ b_{n-r} \\ \mathbf{0} \end{pmatrix}.$$

The relationship of $\lambda_i,\ (1 \le i < n)$ and $\bar{\lambda}_i\ (1 \le i < n-r)$ is known from the so-called interlacing properties [15]:

$$\begin{array}{c}
\lambda_1^{(0)} \le \lambda_1^{(1)} \le \lambda_2^{(0)} \le \lambda_2^{(1)} \le \cdots \le \lambda_{n-1}^{(0)} \le \lambda_{n-1}^{(1)} \le \lambda_n^{(0)}, \\
\lambda_1^{(1)} \le \lambda_1^{(2)} \le \lambda_2^{(1)} \le \lambda_2^{(2)} \le \cdots \le \lambda_{n-2}^{(1)} \le \lambda_{n-2}^{(2)} \le \lambda_{n-1}^{(1)}, \\
\cdots\cdots \\
\lambda_1^{(r-1)} \le \lambda_1^{(r)} \le \lambda_2^{(r-1)} \le \lambda_2^{(r)} \le \cdots \le \lambda_{n-r}^{(r-1)} \le \lambda_{n-r}^{(r)} \le \lambda_{n-r+1}^{(r-1)},
\end{array} \tag{20}$$

where $\lambda_1^{(0)}, \cdots, \lambda_n^{(0)}$ are the eigenvalues of $\boldsymbol{A}$, which are the alias names of $\lambda, \cdots, \lambda_n$ shown in (18), and $\lambda_1^{(r)}, \cdots, \lambda_{n-r}^{(r)}$ are those of $\bar{\boldsymbol{A}}$, which are the alias names shown in (19). From this relation, we obtain the following relationship of the intervals of the maximum and minimum of the eigenvalues with the increasing the number of constraint conditions:

$$[\lambda_1^{(0)}, \lambda_n^{(0)}] \supset [\lambda_1^{(1)}, \lambda_{n-1}^{(1)}] \supset \cdots \supset [\lambda_1^{(r)}, \lambda_{n-r}^{(r)}], \tag{21}$$

(21) shows that the existence ranges of the eigenvalues are monotonously decreasing, included in the range with the smaller number of constraints. Significantly, the range of the eigenvalues (19) are included in the range of the eigenvalues (18).

We let the eigenvectors corresponding to the eigenvalues (19) be

$$\bar{\boldsymbol{x}}_1, \ \ldots, \ \bar{\boldsymbol{x}}_{n-r}. \tag{22}$$

The eigenvalues of $\boldsymbol{A}$ and $\bar{\boldsymbol{A}}$ are close in the sense of (21), which gives the fundamentals of considering the distance between the eigenvectors of $\boldsymbol{A}$ and the kernel of $\bar{\boldsymbol{A}}$.

4 Distance between the eigenvector and the kernel of the unconstrained stiffness matrix

4.1 Coarse grid matrix

Let $\boldsymbol{A} \in \mathbf{R}^{n\times n}$ be the unconstrained stiffness matrix, as noted above. In this section, we discuss the distances between Ker$\boldsymbol{A}$ and the eigenvectors of $\bar{\boldsymbol{A}}$.

Let Ω be the target of the analysis model and (x_i, y_i, z_i), $1 \le i \le n$ be the nodal points of Ω. All the elements are assumed to be solid elements. According to descriptions given in the literature [22, 24], let

$$\mathbf{\Phi} = \begin{pmatrix} 1 & 0 & 0 & 0 & z_1 & -y_1 \\ 0 & 1 & 0 & -z_1 & 0 & x_1 \\ 0 & 0 & 1 & y_1 & -x_1 & 0 \\ 1 & 0 & 0 & 0 & z_2 & -y_2 \\ 0 & 1 & 0 & -z_2 & 0 & x_2 \\ 0 & 0 & 1 & y_2 & -x_2 & 0 \\ \cdots & & & & & \\ 1 & 0 & 0 & 0 & z_{n/3} & -y_{n/3} \\ 0 & 1 & 0 & -z_{n/3} & 0 & x_{n/3} \\ 0 & 0 & 1 & y_{n/3} & -x_{n/3} & 0 \end{pmatrix} \in \mathbf{R}^{n\times 6}, \tag{23}$$

where (x_i, y_i, z_i) denotes the coordinates of nodal point i. Each column of $\mathbf{\Phi}$ is normalized. We refer to $\mathbf{\Phi}$ as a *coarse-grid matrix*.

The first three columns correspond to the parallel translations of the structure, and the last three are infinitesimal rotations around each coordinate. These six columns are independent and consist of a basis of Ker$\boldsymbol{A}$, which means that the six vectors are the basis of the rigid-body motions of Ω. We write these as $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_6$:

$$\mathbf{\Phi} = (\boldsymbol{f}_1\ \boldsymbol{f}_2\ \boldsymbol{f}_3\ \boldsymbol{f}_4\ \boldsymbol{f}_5\ \boldsymbol{f}_6) \in \mathbf{R}^{n\times 6}. \tag{24}$$

We replace these vectors' first r rows with zero values, where these r rows correspond to the constraint condition. We write these as $\bar{\boldsymbol{f}}_1, \ldots, \bar{\boldsymbol{f}}_6$. According to the rule described in Sect. 3, we identify $\bar{\boldsymbol{f}}_i \in \mathbf{R}^{n-r}$ and $\boldsymbol{f}' \in \mathbf{R}^n$:

$$f'_i = \begin{pmatrix} \mathbf{0} \\ \bar{f}_i \end{pmatrix}. \tag{25}$$

$\mathbf{\Phi}$ is rewritten as $\mathbf{\Phi}' \in \mathbf{R}^{n\times 6}$ or $\bar{\mathbf{\Phi}} \in \mathbf{R}^{(n-r)\times 6}$ corresponding to (14) or (17):

$$\mathbf{\Phi}' = (f'_1\ \cdots\ f'_6) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ \cdots & & & & & \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & z_1 & -y_1 \\ 0 & 1 & 0 & -z_1 & 0 & x_1 \\ 0 & 0 & 1 & y_1 & -x_1 & 0 \\ \cdots & & & & & \\ 1 & 0 & 0 & 0 & z_{n/3-[r/3]} & -y_{n/3-[r/3]} \\ 0 & 1 & 0 & -z_{n/3-[r/3]} & 0 & x_{n/3-[r/3]} \\ 0 & 0 & 1 & y_{n/3-[r/3]} & -x_{n/3-[r/3]} & 0 \end{pmatrix}, \tag{26}$$

$$\bar{\mathbf{\Phi}} = (\bar{\boldsymbol{f}}_1\ \cdots\ \bar{\boldsymbol{f}}_6) = \begin{pmatrix} 1 & 0 & 0 & 0 & z_1 & -y_1 \\ 0 & 1 & 0 & -z_1 & 0 & x_1 \\ 0 & 0 & 1 & y_1 & -x_1 & 0 \\ \cdots & & & & & \\ 1 & 0 & 0 & 0 & z_{n/3-[r/3]} & -y_{n/3-[r/3]} \\ 0 & 1 & 0 & -z_{n/3-[r/3]} & 0 & x_{n/3-[r/3]} \\ 0 & 0 & 1 & y_{n/3-[r/3]} & -x_{n/3-[r/3]} & 0 \end{pmatrix}, \tag{27}$$

where $[r/3]$ represents the quotient $r/3$. In the expression for $\mathbf{\Phi}'$, the first r rows are zero matrix in $\mathbf{R}^{r\times 6}$, and in $\bar{\mathbf{\Phi}}$, these r rows are excluded. Although we assume that the active elements in both (26) and (27) start from the row $(1\ \ 0\ \ 0\ \ 0\ \ z_1\ \ -y_1)$ for simplicity, the expression of the first three rows depends on whether r is a multiple of 3 or not. In other words, although n is a multiple of 3, r is not necessarily a multiple of 3. We also refer to $\mathbf{\Phi}'$ and $\bar{\mathbf{\Phi}}$ as the *coarse grid matrices*, as we refer to $\mathbf{\Phi}$ defined by (23).

In the following, we assume the form of the equation for our problem to be (17), and accordingly, the form of the coarse grid matrix is taken to be (27); i.e., $\bar{\mathbf{\Phi}}$.

4.2 Distance between the eigenvector and the kernel of the unconstrained stiffness matrix

In this section, we consider the L^2 distance between the eigenvector $\bar{\boldsymbol{x}}$ of $\bar{\boldsymbol{A}}$, which we identify with $\boldsymbol{x}'$, and $\mathrm{Ker}\boldsymbol{A}$.

Let U be a subspace of $\mathbf{R}^n$. The distance between a point $\boldsymbol{x} \in \mathbf{R}^n$ and U is then the distance between $\boldsymbol{x}$ and the closest point of U. We denote this distance as

$$\mathrm{dist}\,(\boldsymbol{x}, U) = \min_{\boldsymbol{y}\in U} \|\boldsymbol{x} - \boldsymbol{y}\|_2. \tag{28}$$

Let $\boldsymbol{P}$ be an orthogonal projection from $\mathbf{R}^n$ to U. This equation is equal to

$$\mathrm{dist}\,(\boldsymbol{x}, U) = \|\boldsymbol{x} - \boldsymbol{P}\boldsymbol{x}\|_2. \tag{29}$$

In particular, if $\|\boldsymbol{x}\|_2 = 1$, then

$$\|\boldsymbol{x} - \boldsymbol{P}\boldsymbol{x}\|_2^2 = 1 - \boldsymbol{x}^T\boldsymbol{P}\boldsymbol{x}, \quad 0 \le \mathrm{dist}\,(\boldsymbol{x},\ U) \le 1. \tag{30}$$

The projection from $\mathbf{R}^n$ to $\mathrm{Ker}\boldsymbol{A}$ is given by

$$\boldsymbol{P} = \mathbf{\Phi}(\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T = \mathbf{\Phi}\mathbf{\Phi}^T \in \mathbf{R}^{n\times n}, \tag{31}$$

where Φ is defined by (23).

In the following, we evaluate the distance between the eigenvector $\bar{\boldsymbol{x}} \in \mathbf{R}^{n-r}$ of $\bar{\boldsymbol{A}}$ and $\mathrm{Ker}\boldsymbol{A}$. According to the rule described in Sect. 3, we identify $\bar{\boldsymbol{x}} \in \mathbf{R}^{n-r}$ and $\boldsymbol{x}' \in \mathbf{R}^n$. We consider the distance to be

$$\mathrm{dist}(\bar{\boldsymbol{x}},\ \mathrm{Ker}\boldsymbol{A}) = \|\boldsymbol{x}' - \boldsymbol{P}\boldsymbol{x}'\|_2. \tag{32}$$

Additionally, we discuss, in the following theorem, the angle (minimal angle) between a point $\boldsymbol{x} \in \mathbf{R}^n$ with the unit norm and a subspace U, which is defined as in [31]

$$\theta = \arccos \max_{\boldsymbol{u}\in U,\ \|\boldsymbol{u}\|_2=1} \boldsymbol{x}^T\boldsymbol{u}. \tag{33}$$

Theorem 1 (Distance between the eigenvector and the kernel of the unconstrained stiffness matrix)

Let the eigenvalues and the corresponding eigenvectors of $\bar{\boldsymbol{A}}$ be given by (19) and (22), respectively. We then have the following properties.

(1) For arbitrary $1 \leq k \leq n - r$, $\boldsymbol{x}'_k \notin \mathrm{Ker}\boldsymbol{A}$, and accordingly,

$$\mathrm{dist}\,(\boldsymbol{x}'_k,\ \mathrm{Ker}\boldsymbol{A}) > 0. \tag{34}$$

(2) If $k < n-r$ is large enough, then $\boldsymbol{x}'_k$ is nearly orthogonal to $\mathrm{Ker}\boldsymbol{A}$, and accordingly,

$$\mathrm{dist}\,(\boldsymbol{x}'_k,\ \mathrm{Ker}\ \boldsymbol{A}) \approx 1. \tag{35}$$

(3) If $k \geq 1$ is small enough, then

$$\mathrm{dist}\,(\boldsymbol{x}'_k,\ \mathrm{Ker}\boldsymbol{A}) \approx 0. \tag{36}$$

In particular, if $\boldsymbol{x}'_k$ is the *fundamental mode* $\boldsymbol{x}'_1$, then $\mathrm{dist}\,(\boldsymbol{x}'_1,\ \mathrm{Ker}\boldsymbol{A})$ takes the smallest value.

Proof As noted previously, the number of the eigenvalues and eigenvectors is assumed to be $1 \leq k \leq n - r$ imposing the r constrained conditions.

We prove 1). If $\boldsymbol{A}\boldsymbol{x}'_k = \boldsymbol{0}$, then $\boldsymbol{x}'_k$ is the rigid-body mode. The r zeros put above $\bar{\boldsymbol{x}}_k$ cannot be represented by the rigid-body motions. This proves (34).

We prove 2) and 3). It is known from *modal analysis* that the modal displacement components of a structural body are dominant for the lower frequency range, and the whole body vibrates largely, whereas, in the range of higher frequencies, vibrations become minute in proportion to the frequencies and depending on the shape of the body.

Take an arbitrary $1 \leq k \leq n - r$. Referring to (33), the angle (minimal angle) between $\boldsymbol{x}'_k$ and $\mathrm{Ker}\boldsymbol{A}$ is given by

$$\theta = \arccos \max_{1 \leq j \leq 6} {\boldsymbol{x}'_k}^T \boldsymbol{g}_j, \tag{37}$$

where $\boldsymbol{g}_1, \cdots, \boldsymbol{g}_6$ are the appropriate basis of $\mathrm{Ker}\boldsymbol{A}$.

We write the components of the i-th nodal point of $\boldsymbol{x}'_k$ as $\{x'_{ki}\ y'_{ki}\ z'_{ki}\}$. The first r components of $\boldsymbol{x}'_k$ are zeros, and $\boldsymbol{x}'_k$ can thus be written as

$$\boldsymbol{x}'_k = \frac{1}{\|\boldsymbol{x}'_k\|}\Big(0 \ \cdots \ 0 \ \{x'_{k,[r/3]+1} \ y'_{k,[r/3]+1} \ z'_{k,[r/3]+1}\} \\ \cdots \ \{x'_{k,n/3-[r/3]} \ y'_{k,n/3-[r/3]} \ z'_{k,n/3-[r/3]}\}\Big)^T . \tag{38}$$

Although $\boldsymbol{g}_i$ in (37) does not coincide with $\boldsymbol{f}_i$ in general, by representing appropriate $\boldsymbol{g}_i$ as a linear combination of $\boldsymbol{f}_i$, we can take $\boldsymbol{f}_i$ instead of $\boldsymbol{g}_i$.

The representations of $\boldsymbol{f}_1, \boldsymbol{f}_2, \boldsymbol{f}_3$ are easily obtained according to their forms. $\boldsymbol{f}_4, \boldsymbol{f}_5, \boldsymbol{f}_6$ are given as

$$\begin{aligned} \boldsymbol{f}_4 &= \frac{1}{\|\boldsymbol{f}_4\|}\big(\{0 \ -z_1 \ y_1\} \cdots \{0 \ -z_{[r/3]} \ y_{[r/3]}\} \ \{0 \ -z_{[r/3]+1} \ y_{[r/3]+1}\} \\ &\qquad \cdots \{0 \ -z_{n/3-[r/3]} \ y_{n/3-[r/3]}\}\big)^T , \\ \boldsymbol{f}_5 &= \frac{1}{\|\boldsymbol{f}_5\|}\big(\{z_1 \ 0 \ -x_1\} \cdots \{z_{[r/3]} \ 0 \ -x_{[r/3]}\} \ \{z_{[r/3]+1} \ 0 \ -x_{[r/3]+1}\} \\ &\qquad \cdots \{z_{n/3-[r/3]} \ 0 \ -x_{n/3-[r/3]}\}\big)^T , \\ \boldsymbol{f}_6 &= \frac{1}{\|\boldsymbol{f}_6\|}\big(\{-y_1 \ x_1 \ 0\} \cdots \{-y_{[r/3]} \ x_{[r/3]} \ 0\} \ \{-y_{[r/3]+1]} \ x_{[r/3]+1]} \ 0\}\} \\ &\qquad \cdots \{-y_{n/3-[r/3]} \ x_{n/3-[r/3]} \ 0\}\big)^T . \end{aligned} \tag{39}$$

Assuming that all the vectors are normalized, the inner products of $\boldsymbol{x}'_k$ and $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_6$ are

$$\begin{aligned} {\boldsymbol{x}'_k}^T \boldsymbol{f}_1 &= \sum_{i=1}^{n/3-[r/3]} x'_{ki}, \\ {\boldsymbol{x}'_k}^T \boldsymbol{f}_2 &= \sum_{i=1}^{n/3-[r/3]} y'_{ki}, \\ {\boldsymbol{x}'_k}^T \boldsymbol{f}_3 &= \sum_{i=1}^{n/3-[r/3]} z'_{ki}, \\ {\boldsymbol{x}'_k}^T \boldsymbol{f}_4 &= \sum_{i=1}^{n/3-[r/3]} (0 \ -z_{k,[r/3]+i} \ \ y_{k,[r/3]+i}) \begin{pmatrix} x'_{k,[r/3]+i} \\ y'_{k,[r/3]+i} \\ z'_{k,[r/3]+i} \end{pmatrix}, \\ {x'_k}^T \boldsymbol{f}_5 &= \sum_{i=1}^{n/3-[r/3]} (z_{k,[r/3]+i} \ 0 \ -x_{k,[r/3]+i}) \begin{pmatrix} x'_{k,[r/3]+i} \\ y'_{k,[r/3]+i} \\ z'_{k,[r/3]+i} \end{pmatrix}, \\ {\boldsymbol{x}'_k}^T \boldsymbol{f}_6 &= \sum_{i=1}^{n/3-[r/3]} (-y_{k,[r/3]+i} \ \ x_{k,[r/3]+i} \ 0) \begin{pmatrix} x'_{k,[r/3]+i} \\ y'_{k,[r/3]+i} \\ z'_{k,[r/3]+i} \end{pmatrix}. \end{aligned} \tag{40}$$

We can evaluate the variation of the six inner products ${\boldsymbol{x}'_k}^T \boldsymbol{f}_i$ using these representations depending on k. The first three equations involve only the displacements, whereas the displacements are multiplied by the constant values of the coordinates of

Table 2 Cuboid models for evaluating the distance between the eigenvectors and the kernel. See Fig. 4

Models	Elements	^1n	^1r	$n-r$
$1 \times 1 \times 6$	Linear hexa[2]	84	9	75
$2 \times 2 \times 12$	Linear hexa[2]	351	12	339
$4 \times 3 \times 12$	Linear hexa	780	18	762
$4 \times 3 \times 12$	Linear tetra[2]	780	18	762
$4 \times 3 \times 12$	Quad tetra[2]	4725	18	4707

[1] n and r indicate the number of DOFs and the number of the constraints, respectively
[2] "linear hexa", "linear tetra," and "quad tetra" indicate linear hexahedral, linear tetrahedral, and quadrilateral tetrahedral elements, respectively

the nodal points in the last three equations. If $1 \leq k \leq n-r$ is sufficiently small, then the modal shape of the target model deforms largely from the static state. In particular, the deformation is largest for the fundamental mode with $k = 1$, which corresponds to the maximum of the six values of (40), and dist $(\boldsymbol{x}'_1, \operatorname{Ker}\boldsymbol{A})$ is accordingly the smallest. This proves (36) and the last statement of property 3).

Meanwhile, if k becomes large, the vibration state of the model becomes minute, and the variations of the components of $\boldsymbol{x}'_k$ decrease. As a result, fixing $1 \leq l \leq n-r$ and taking sufficiently large $k > l$, we have

$$\max_{1\leq j\leq 6} \boldsymbol{x}_l'^{T} \boldsymbol{f}_j \geq \max_{1\leq j\leq 6} \boldsymbol{x}_k'^{T} \boldsymbol{f}_j, \tag{41}$$

and a larger k results in $\boldsymbol{x}'_k$ being closer to the direction orthogonal to $\operatorname{Ker}\boldsymbol{A}$, which means dist$(\boldsymbol{x}'_k, \operatorname{Ker}\boldsymbol{A})$ approaches 1. This proves (35).

4.3 Example of the distance between eigenvectors and the kernel of the unconstrained matrix

In this section, we present actual distance curves of the distance between eigenvectors and the kernel of the unconstrained matrix using simple examples.

We show three cuboid models in Fig. 4. On the left and at the center are $1 \times 1 \times 6$ and $4 \times 3 \times 12$ models with hexahedral linear elements, respectively. On the right is a model of the same size as the $4 \times 3 \times 12$ model but with tetrahedral elements set to be linear and quadrilateral. Additionally, we include a $2 \times 2 \times 12$ model with hexahedral linear elements in our testing. The constraint conditions are assumed to have 9, 12, and 18 DOFs for the $1 \times 1 \times 6$, $2 \times 2 \times 12$, and $4 \times 3 \times 12$ models, respectively. The cuboid models are summarized in Table 2.

Moreover, we use long and short perforated plates and a pipe model. We present these three models in Fig. 5 and Table 3. We assume linear and quadrilateral hexahedral elements for these models.

We assume the physical properties to be those of standard steel. Young's modulus and Poisson ratio are 200 GPa and 0.3, respectively.

Table 3 Other models for evaluating the distance between the eigenvectors and the kernel. See Fig. 5

Models	Elements	n	r	$n-r$
Short perforated plate	linear tetra[1]	1437	9	1428
Short perforated plate	quad tetra[1]	8802	9	8793
Long perforated plate	inear tetra	2064	9	2055
Long perforated plate	quad tetra	12462	9	12453
Pipe	linear tetra	1476	9	1467
Pipe	quad tetra	8640	9	8631

[1]"linear tetra" and "quad tetra" indicate linear tetrahedral and quadrilateral tetrahedral elements, respectively

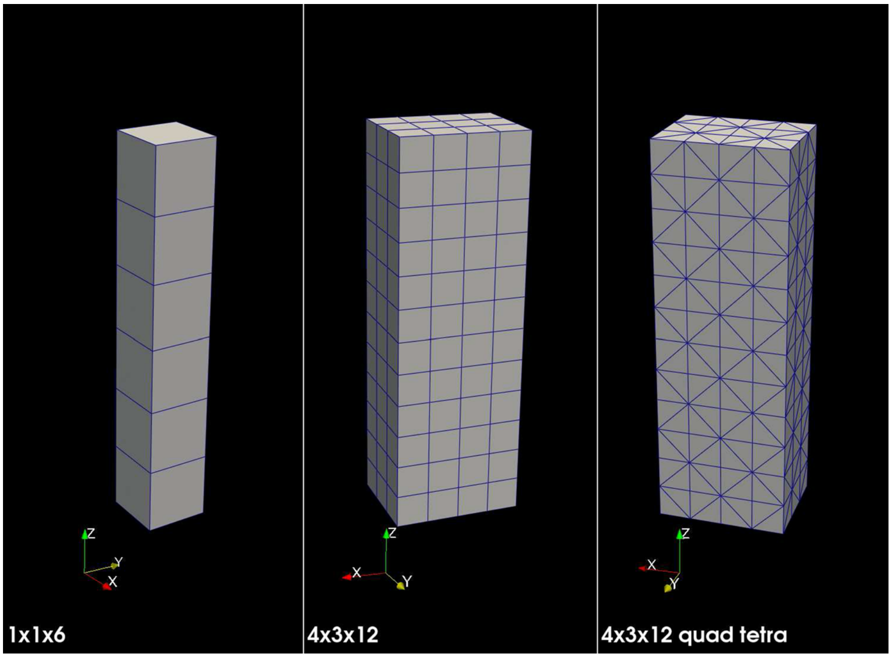

Fig. 4 Cuboid models used to evaluate the distance between the eigenvectors and the kernel

Curves of the distance between the eigenvectors and the kernel for the cuboid models, those for the short perforated plate, those for the long perforated plate, and those for the pipe model are shown in Fig. 6, 7, 8, and 9, respectively.

The eigenvalues and the distance values up to $k = 15$ are given in Table 4 as examples; (a) corresponds to the cuboid $1 \times 1 \times 6$ model with linear hexahedral elements, (b) corresponds to the cuboid $4 \times 3 \times 12$ model with quadrilateral tetrahedral element, (c) corresponds to the long plate model with quadrilateral tetrahedral elements, and (d) corresponds to the pipe model with quadrilateral tetrahedral elements.

All the results are in accordance with (1), (2), and (3) described in Theorem 1.

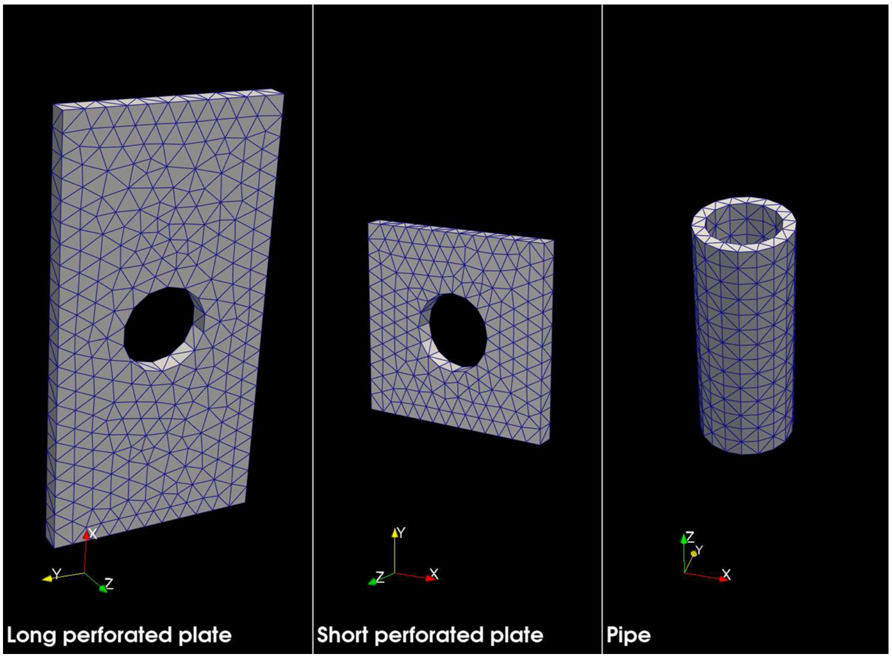

Fig. 5 Long, short perforated plates and pipe used to evaluate the distance. The long perforated plate model with the quadrilateral tetrahedral elements has the largest number of DOFs among our models, including other models (see Fig. 4 and Table 3)

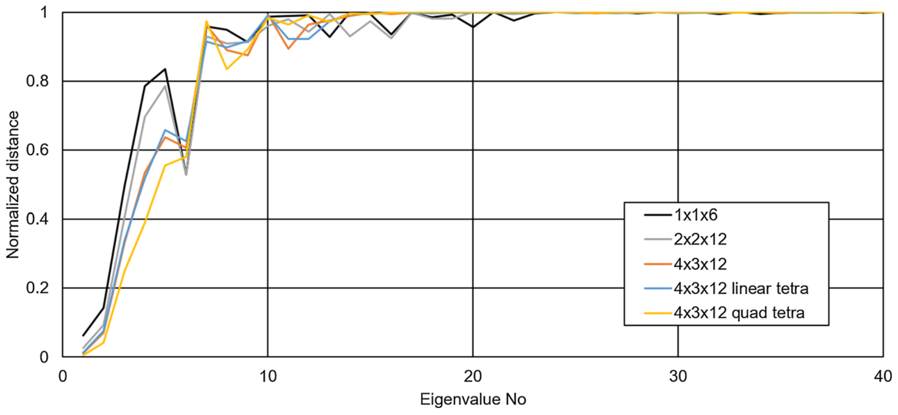

Fig. 6 Distance between the eigenvector and the kernel of the cuboid models. The distance is small at smaller eigenvalues and rapidly approaches a value of 1 at higher eigenvalues

Table 4 Example of the eigenvalues and the distances from the kernel up to $k = 15$

	k	λ_k	[1]dist	k	λ_k	[1]dist	k	λ_k	[1]dist
(a) [2]$1 \times 1 \times 6$	1	18.468	0.0632	6	2527.9	0.5285	11	13970	0.9883
	2	43.342	0.1430	7	3152.1	0.9585	12	15849	0.9910
	3	364.41	0.4931	8	6162.6	0.9490	13	17522	0.9269
	4	835.70	0.7847	9	7274.9	0.9120	14	19919	1.0000
	5	1311.1	0.8349	10	8142.8	0.9865	15	20111	0.9945
(b) [3]$4 \times 3 \times 12$	1	0.2526	0.0056	6	163.51	0.5804	11	1334.5	0.9632
	2	3.6903	0.0422	7	383.23	0.9734	12	1477.9	0.9906
	3	42.031	0.2488	8	611.22	0.8351	13	1629.1	0.9716
	4	64.709	0.3899	9	693.50	0.8914	14	2643.6	0.9963
	5	139.95	0.5542	10	1252.3	0.9838	15	2772.5	0.9945
(c) [3]Long plate	1	0.0730	0.0062	6	77.246	0.8580	11	420.91	0.9915
	2	2.0272	0.0270	7	129.81	0.5549	12	460.09	0.9955
	3	7.0486	0.1560	8	153.71	0.7520	13	542.13	0.9734
	4	7.8107	0.5793	9	207.13	0.7003	14	631.39	0.9980
	5	30.108	0.9915	10	242.80	0.8775	15	1048.2	0.9994
(d) [3]Pipe	1	0.0230	0.0004	6	355.14	0.4411	11	1153.7	0.9986
	2	1.8567	0.0088	7	673.27	0.9998	12	1183.4	0.9983
	3	6.5902	0.0548	8	691.64	0.9999	13	1327.9	0.9988
	4	129.50	0.2629	9	862.02	0.9926	14	1752.0	0.9755
	5	163.58	0.2762	10	931.42	0.9818	15	2215.4	0.9478

[1]"dist" means distance
[2] Linear hexahedral elements
[3] Quadrilateral tetrahedral elements

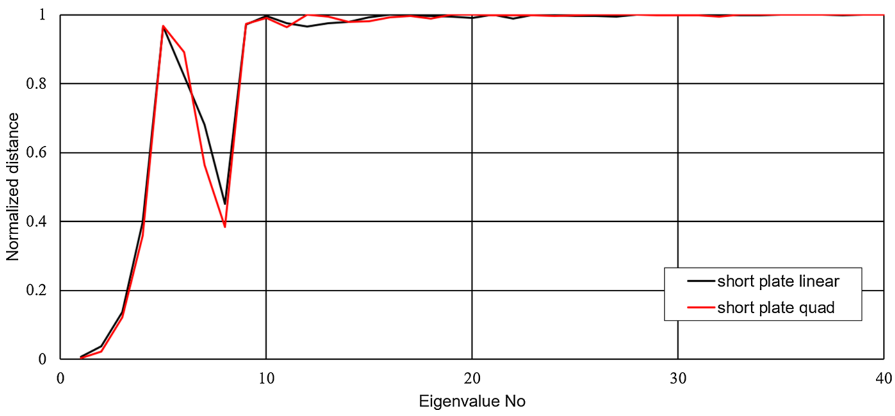

Fig. 7 Distance between the eigenvector and the kernel of the short perforated plate. The same tendency is seen in Fig. 6. A finer mesh decomposition reduces the distance between the first eigenvector and the kernel

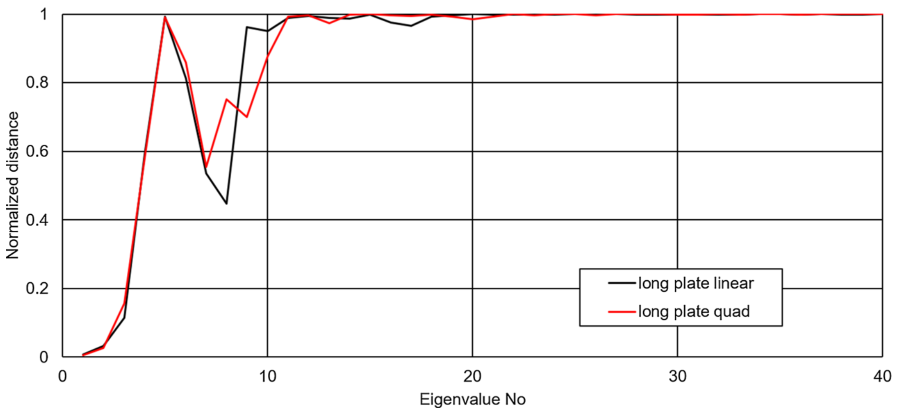

Fig. 8 Distance between the eigenvector and the kernel of the long perforated plate

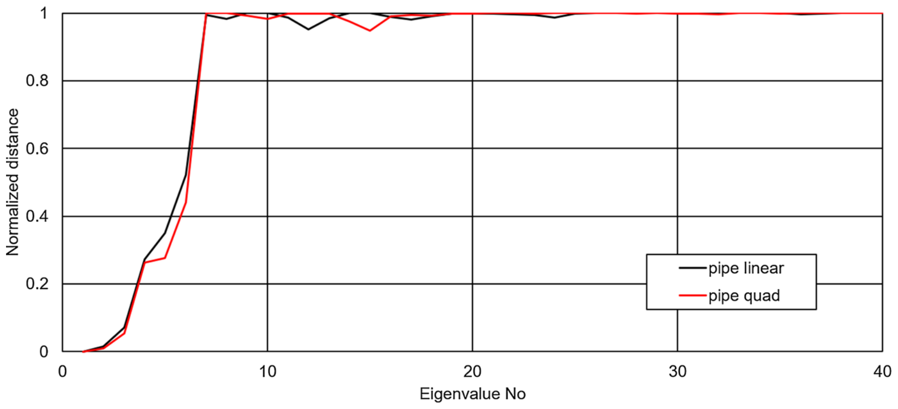

Fig. 9 Distance between the eigenvector and the kernel of the pipe model. In our examples, the pipe model with quadrilateral elements has the smallest first eigenvalue and the smallest distance between the eigenvector and the kernel

5 Introduction to the DDDM

We use solid elements for all the elements given in this section.

5.1 Expression of the coarse grid by domain decomposition

We decompose Ω into N non-overlapping subdomains Ω_J, $1 \leq J \leq N$, except that the boundaries between subdomains overlap one another. Each Ω_J is a closed domain that includes boundary surfaces. We write Ω as

$$\Omega = \bigcup_J \Omega_J. \tag{42}$$

The domain decomposition here is "element-base" decomposition as opposed to the domain decomposition used for the parallelization is "nodal-point-base" decomposition which we describe in Sect. 7.1.

The domain decomposition is represented by diagonal matrices $\bar{\boldsymbol{D}}_J \in \mathbf{R}^{(n-r)\times(n-r)}$ comprising d_{Ji}:

$$\bar{\boldsymbol{D}}_J = \mathrm{diag}\,(0,\ \ldots,\ 0,\ d_{J*},\ \ldots,\ d_{J*},\ 0,\ \ldots,\ 0), \tag{43}$$

$$d_{Ji} = \begin{cases} 0 & i \notin \Omega_J \\ 1 & i \in \Omega_J^{\circ} \\ 1/(\text{number of overlaps}), & \text{where } i \text{ is a point of the boundary} \end{cases}, \tag{44}$$

where Ω_J° is the interior of the subdomain Ω_J. Representing i that appears in the component $\bar{\boldsymbol{D}}_J$ is complex, and thus in (43), an abbreviated expression d_{J*} is used. In (43), there are cases in which non-zero components leap and are not continuously aligned.

From the construction of $\bar{\boldsymbol{D}}_J$, the domain decomposition (42) corresponds to

$$\sum_{J=1}^{N} \bar{\boldsymbol{D}}_J = I. \tag{45}$$

This means that $\{\bar{\boldsymbol{D}}_J\}$ is a *partition of unity* on the space $\mathbf{R}^{n-r}$. We then have

$$\sum_{J=1}^{N} \bar{\boldsymbol{D}}_J \bar{\boldsymbol{\Phi}} = \bar{\boldsymbol{\Phi}}. \tag{46}$$

Let

$$\bar{\boldsymbol{F}}_J = \bar{\boldsymbol{D}}_J \bar{\boldsymbol{\Phi}} \in \mathbf{R}^{(n-r)\times 6}, \quad 1 \le J \le N. \tag{47}$$

We align (47) and let

$$\bar{\boldsymbol{F}} = (\bar{\boldsymbol{F}}_1 \ \ \cdots \ \ \bar{\boldsymbol{F}}_N) \in \mathbf{R}^{(n-r)\times 6N}. \tag{48}$$

We refer to this matrix as the *extension matrix*. We write $W \equiv \mathbf{R}^{6N}$ and $V \equiv \mathbf{R}^{n-r}$. W is a *coarse space*, whereas V is the global *solution space*. $\bar{\boldsymbol{F}}$ embeds W into V. The column vectors of $\bar{\boldsymbol{F}}_1, \cdots, \bar{\boldsymbol{F}}_N$ constitute a basis of W. Although W is not a subspace of V, $\bar{\boldsymbol{F}}W$ is a subspace. Since $\bar{\boldsymbol{F}}^T$ is of full rank, $\mathrm{Im}\bar{\boldsymbol{F}}^T$ is isomorphic to $W : \bar{\boldsymbol{F}}^T V \cong W$.

5.2 Framework of the DDDM algorithm

We describe the DDDM algorithm as follows. Construct a coarse grid using the rigid-body modes $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_6$ of the original problem without a constraint condition; remove the corresponding low-frequency modes from the entire space V using the deflation algorithm; and apply the CG method to the segregated complementary space.

Let

$$\bar{\boldsymbol{A}}_{\bar{\boldsymbol{F}}} = \bar{\boldsymbol{F}}^T \bar{\boldsymbol{A}} \bar{\boldsymbol{F}} \in \mathbf{R}^{6N \times 6N}. \tag{49}$$

We refer to this matrix as a *contraction matrix*. Because the size of this matrix is small, we can obtain the inverse matrix $\bar{\boldsymbol{A}}_{\bar{\boldsymbol{F}}}^{-1}$ using a direct method or LU decomposition (lower-upper decomposition). Moreover, we extend $\bar{\boldsymbol{A}}_{\bar{\boldsymbol{F}}}^{-1}$ onto V and express $(\bar{\boldsymbol{A}}_{\boldsymbol{F}}^{-1})^*$ as

$$(\bar{\boldsymbol{A}}_{\bar{\boldsymbol{F}}}^{-1})^* = \bar{\boldsymbol{F}} \bar{\boldsymbol{A}}_{\bar{\boldsymbol{F}}}^{-1} \bar{\boldsymbol{F}}^T = \bar{\boldsymbol{F}}(\bar{\boldsymbol{F}}^T \bar{\boldsymbol{A}} \bar{\boldsymbol{F}})^{-1} \bar{\boldsymbol{F}}^T \in \mathbf{R}^{(n-r)\times(n-r)}. \tag{50}$$

We refer to this matrix as a *pullback* of $\bar{\boldsymbol{A}}$ under $\bar{\boldsymbol{F}}$. Figure 10 is a diagram of $\bar{\boldsymbol{A}}_{\bar{\boldsymbol{F}}}$ and $(\bar{\boldsymbol{A}}_{\bar{\boldsymbol{F}}}^{-1})^*$.

Furthermore, let $\bar{\boldsymbol{P}}_{\bar{A}}$ be a matrix obtained by multiplying $\bar{\boldsymbol{A}}$ by the pullback from the right side:

$$\bar{\boldsymbol{P}}_{\bar{A}} = (\bar{\boldsymbol{A}}_{\bar{\boldsymbol{F}}}^{-1})^* \bar{\boldsymbol{A}} = \bar{\boldsymbol{F}}(\bar{\boldsymbol{F}}^T \bar{\boldsymbol{A}} \bar{\boldsymbol{F}})^{-1} \bar{\boldsymbol{F}}^T \bar{\boldsymbol{A}} \in \mathbf{R}^{(n-r)\times(n-r)}. \tag{51}$$

Proposition 2 (Contraction projection)

The following properties hold:

$$\bar{\boldsymbol{P}}_{\bar{A}}^2 = \bar{\boldsymbol{P}}_{\bar{A}}, \quad \bar{\boldsymbol{A}} \bar{\boldsymbol{P}}_{\bar{A}} = \bar{\boldsymbol{P}}_{\bar{A}}^T \bar{\boldsymbol{A}}. \tag{52}$$

Therefore, $\bar{\boldsymbol{P}}_{\bar{A}}$ *is a projection.*

Proof Easily shown.

We refer to $\bar{\boldsymbol{P}}_{\bar{A}}$ as a *contraction projection obtained from the pullback* of $\bar{\boldsymbol{A}}_{\bar{\boldsymbol{F}}}^{-1}$, or simply a *contraction projection*. The image of V obtained by the contraction projection is called a *contraction projection space*, or simply a *contraction space*.

Remark 1 *(Definition of the deflation projection)*

In many articles (e.g., [7–14]), the deflation projection $\bar{\boldsymbol{P}}$ is defined as

$$\bar{\boldsymbol{P}} = \boldsymbol{I} - \bar{\boldsymbol{A}} \bar{\boldsymbol{F}}(\bar{\boldsymbol{F}}^T \bar{\boldsymbol{A}} \bar{\boldsymbol{F}})^{-1} \bar{\boldsymbol{F}}^T \tag{53}$$

if using our notation.

(a) The second term on the right side of (53) corresponds to (51), but in (51), the matrix $\bar{\boldsymbol{A}}$ is multiplied from the right side, whereas in (53), $\bar{\boldsymbol{A}}$ is multiplied from the left.

(b) Additionally, the projection $\bar{\boldsymbol{P}}$ here is defined as the complementary projection of the *contraction projection* (in our sense).

(a)

$$\begin{tikzcd}
V \arrow[r, "\bar{\boldsymbol{A}}"] \arrow[dr, phantom, "\circlearrowleft"] & V \arrow[d, "\boldsymbol{F}^T"] \\
W \arrow[u, "\boldsymbol{F}"] \arrow[r, "\bar{\boldsymbol{A}}_{\boldsymbol{F}} = \boldsymbol{F}^T \bar{\boldsymbol{A}} \boldsymbol{F}"'] & W
\end{tikzcd}$$

(b)

$$\begin{tikzcd}
V \arrow[dr, phantom, "\circlearrowright"] & V \arrow[l, "\boldsymbol{F} \bar{\boldsymbol{A}}_{\boldsymbol{F}}^{-1} \boldsymbol{F}^T = (\bar{\boldsymbol{A}}_{\boldsymbol{F}}^{-1})^*"'] \arrow[d, "\boldsymbol{F}^T"] \\
W \arrow[u, "\boldsymbol{F}"] & W \arrow[l, "\bar{\boldsymbol{A}}_{\boldsymbol{F}}^{-1}"]
\end{tikzcd}$$

Fig. 10 Restriction of $\bar{\boldsymbol{A}}$ to $\bar{\boldsymbol{A}}_{\boldsymbol{F}}$ and the extension of $\bar{\boldsymbol{A}}_{\boldsymbol{F}}^{-1}$ to its pullback

(c) Our definition (51) is needed as will be described in Remark 2 in Sect. 6.2 to reduce the calculation cost in each CG step.

Proposition 3 (Contraction space and the image of the extension matrix)
The image of the extension matrix coincides with the contraction projection space:

$$\bar{\boldsymbol{F}} W = \bar{\boldsymbol{P}}_{\bar{A}} V. \tag{54}$$

Proof Easily shown.

Because $\bar{\boldsymbol{F}} \in \mathbf{R}^{(n-r)\times 6N}$ is of full rank and one-to-one, and Proposition 3 thus shows that $\bar{\boldsymbol{F}}$ is an isomorphism from the coarse space W onto the contraction space. We therefore refer to $\bar{\boldsymbol{F}} W \equiv \bar{\boldsymbol{P}}_{\bar{A}} V$ as the *coarse space* like W.

Proposition 2 leads to the direct sum decomposition of V:

$$\begin{aligned} V &= \bar{\boldsymbol{P}}_{\bar{A}} V \oplus (\boldsymbol{I} - \bar{\boldsymbol{P}}_{\bar{A}}) V \\ &= \{\bar{\boldsymbol{x}} \in V \mid \bar{\boldsymbol{P}}_{\bar{A}} \bar{\boldsymbol{x}} = \bar{\boldsymbol{x}}\} \oplus \{\bar{\boldsymbol{x}} \in V \mid \bar{\boldsymbol{P}}_{\bar{A}} \bar{\boldsymbol{x}} = \boldsymbol{0}\}. \end{aligned} \tag{55}$$

Here, $\bar{\boldsymbol{P}}_{\bar{A}} V = \operatorname{Im} \bar{\boldsymbol{P}}_{\bar{A}}$ is a space spanned by the eigenvectors with the smaller eigenvalues including the lowest one. Meanwhile, $(\boldsymbol{I} - \bar{\boldsymbol{P}}_{\bar{A}}) V = \operatorname{Ker} \bar{\boldsymbol{P}}_{\bar{A}}$ is a space obtained by eliminating those eigenvectors with the small eigenvalues. In other words, $\bar{\boldsymbol{P}}_{\bar{A}} V$ is a coarse space and $(\boldsymbol{I} - \bar{\boldsymbol{P}}_{\bar{A}}) V$ is a fine space.

Theorem 4 (Restriction of the stiffness matrix to the contraction space)
The following three properties hold.

(i) The restriction of $\bar{\boldsymbol{A}}^{-1}$ onto $\boldsymbol{P}_{\bar{A}} V$ by the contraction projection $\bar{\boldsymbol{P}}_{\bar{A}}$ coincides with the pullback of $\bar{\boldsymbol{A}}$ under $\bar{\boldsymbol{F}}$:

$$\bar{\boldsymbol{P}}_{\bar{A}} \bar{\boldsymbol{A}}^{-1} \bar{\boldsymbol{P}}_{\bar{A}}^T = \bar{\boldsymbol{F}} (\bar{\boldsymbol{F}}^T \bar{\boldsymbol{A}} \bar{\boldsymbol{F}})^{-1} \bar{\boldsymbol{F}}^T = (\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^*. \tag{56}$$

(ii) The image of the pullback coincides with the contraction space:

$$(\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^* V = \bar{\boldsymbol{P}}_{\bar{A}} V. \tag{57}$$

$$\begin{tikzcd}[column sep=huge] \text{(a)}\ V \arrow[d, "\bar{\boldsymbol{P}}_{\bar{\boldsymbol{A}}}"'] \arrow[r, phantom, "\circlearrowleft", shift right=8ex] & V \arrow[l, "\bar{\boldsymbol{A}}^{-1}"'] \\ \bar{\boldsymbol{P}}_{\bar{A}} V & \bar{\boldsymbol{P}}_{\bar{\boldsymbol{A}}} V \arrow[l, "\bar{\boldsymbol{P}}_{\bar{\boldsymbol{A}}} \bar{\boldsymbol{A}}^{-1} \bar{\boldsymbol{P}}^{T}_{\bar{\boldsymbol{A}}} = (\bar{\boldsymbol{A}}_{\boldsymbol{F}}^{-1})^*"] \arrow[u, "\bar{\boldsymbol{P}}^{T}_{\bar{\boldsymbol{A}}}"'] \end{tikzcd}
\qquad
\begin{tikzcd}[column sep=huge] \text{(b)}\ V \arrow[r, "\bar{\boldsymbol{A}}"] \arrow[r, phantom, "\circlearrowright", shift right=8ex] & V \arrow[d, "(\bar{\boldsymbol{A}}_{\boldsymbol{F}}^{-1})^*"] \\ \bar{\boldsymbol{P}}_{\bar{\boldsymbol{A}}} V \arrow[u, "(\bar{\boldsymbol{A}}_{\boldsymbol{F}}^{-1})^*"] \arrow[r, "{(\bar{\boldsymbol{A}}_{\boldsymbol{F}}^{-1})^* \bar{\boldsymbol{A}} (\bar{\boldsymbol{A}}_{\boldsymbol{F}}^{-1})^* = (\bar{\boldsymbol{A}}_{\boldsymbol{F}}^{-1})^*}"'] & \bar{\boldsymbol{P}}_{\bar{\boldsymbol{A}}} V \end{tikzcd}$$

Fig. 11 Reciprocal relation between the restrictions of $\bar{\boldsymbol{A}}^{-1}$ and $\bar{\boldsymbol{A}}$. Panel (a) shows that the pullback $(\bar{\boldsymbol{A}}_{\bar{\boldsymbol{F}}}^{-1})^*$ on the coarse space $\bar{\boldsymbol{P}}_{\bar{\boldsymbol{A}}} V$ hides the inverse $\bar{\boldsymbol{A}}^{-1}$ on the global space V

(iii) The restriction of the stiffness matrix $\bar{\boldsymbol{A}}$ onto the contraction space $\bar{\boldsymbol{P}}_{\bar{\boldsymbol{A}}} V$ coincides with the pullback:

$$(\bar{\boldsymbol{A}}_{\bar{\boldsymbol{F}}}^{-1})^* \bar{\boldsymbol{A}} (\bar{\boldsymbol{A}}_{\bar{\boldsymbol{F}}}^{-1})^* = (\bar{\boldsymbol{A}}_{\bar{\boldsymbol{F}}}^{-1})^*. \tag{58}$$

Proof i) and ii) can be easily seen. (58) is a rewriting of (56).

(56) presents the restriction of $\bar{\boldsymbol{A}}^{-1}$ onto the contraction space by $\bar{\boldsymbol{P}}_{\bar{\boldsymbol{A}}}$, and (58) presents the restriction onto the contraction space by the pullback $(\bar{\boldsymbol{A}}_{\bar{\boldsymbol{F}}}^{-1})^*$, showing that the two restrictions give rise to the representations of the same $(\bar{\boldsymbol{A}}_{\bar{\boldsymbol{F}}}^{-1})^*$, and a reciprocal relation as seen in Fig. 11.

The condition number of the problem (17) is

$$\kappa(\bar{\boldsymbol{A}}) = \frac{\bar{\lambda}_{n-r}}{\bar{\lambda}_1}. \tag{59}$$

In the DDDM algorithm, we removed the lower, e.g., m modes, and the condition number changes to

$$\frac{\bar{\lambda}_{n-r}}{\bar{\lambda}_{m+1}} \quad \Big(\leq \kappa(\bar{\boldsymbol{A}})\Big), \tag{60}$$

which means that the DDDM algorithm reduces the condition number to

$$\frac{\lambda_1}{\lambda_{m+1}} \kappa(\bar{\boldsymbol{A}}). \tag{61}$$

5.3 Examples of deformation modes obtained using the DDDM

The target object Ω deforms in the coarse space, as represented by the coarse motion of the subdomains Ω_J, $1 \leq J \leq N$. The coarse motion of Ω is close to the rigid-body motion of Ω, though the imposed constraint conditions fix some part of the deformation of Ω. Ω_J itself has rigid-body modes, and the boundaries of some neighboring subdomains or the constrained boundaries restrict the deformation of Ω.

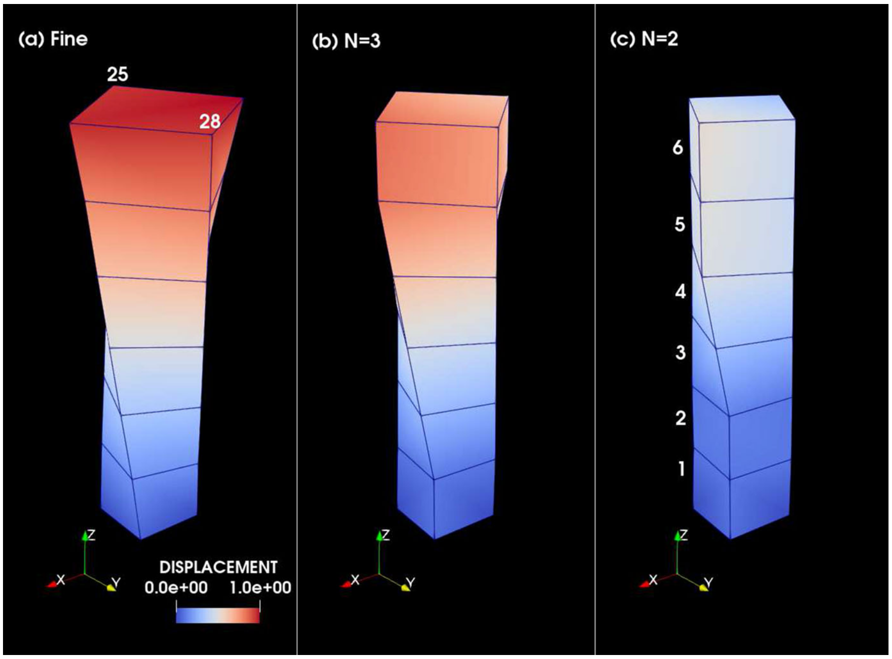

Fig. 12 Deformation of the $1 \times 1 \times 6$ model in the coarse problem setting. Case (a) shows the deformation in the fine space. There are three and two subdomains in Cases (b) and (c), respectively. In Case (c), the model body is decomposed only between elements 3 and 4, whereas in Case (b), the body is decomposed between elements 2 and 3 and between 4 and 5. In Case (c), elements 3 and 4 are dragged by the joining surface of elements 3 and 4, and both elements are distorted. The result of Case (b) is close to the fine result (a) because we give a finer decomposition than Case (c)

We present an example of how the target object deforms in the coarse problem. The stiffness matrix in the coarse space is given by (49). We use here the $1 \times 1 \times 6$ model shown in Fig. 4 in Sect. 4.3. The material is standard steel. We give the constraint conditions to nodal points 1, 2, and 3, corresponding to DOFs 1, 2, 3, 4, 5, 6, 7, 8, and 9. We eliminate these DOFs. Thus, $n = 84$, $r = 9$, and $n - r = 75$, which are the same as noted in Table 2 in Sect. 4.3.

Our problem is to solve the following equation for the displacement $\bar{\boldsymbol{x}}$ given an external force $\bar{b}$:

$$\bar{\boldsymbol{A}}_{\bar{\boldsymbol{F}}} \bar{\boldsymbol{x}} = \bar{\boldsymbol{b}}. \tag{62}$$

The results are shown in Fig. 12. In the figure, we show the nodal point numbers 25 and 28 in Case (a), and the element numbers 1, 2, 3, 4, 5, and 6 in Case (c). The number of decompositions N is two in Case (c) and three in Case (b), whereas Case (a) has no decomposition (fine analysis). In Case (b), we decomposed the model between elements 2 and 3 and between elements 4 and 5. In Case (c), we decomposed between elements 3 and 4.

We apply the external force on the first DOF (the x coordinate) of nodal points 25 and 28 in the negative and positive directions of the x axis, respectively, clockwise with twisting. The force strength is assumed to be as high as 3000 N, which creates an artificially large displacement. The swelling seen in the upper part of the figures, especially in Cases (a) and (b), is supposed to be caused by a violation of the assumption of the infinitesimal deformation in the FEA. In Case (c), elements 3 and 4 are dragged by the joining surface between elements 3 and 4, and both elements are distorted. In Case (b), elements 2, 3, 4, and 5 are distorted. In particular, Case (c) has a deformation near the rigid-body motion of the model. The result of Case (b), which has a finer decomposition than Case (c), is close to the fine result of Case (a).

6 DDDM algorithm

The DDDM algorithm shares similarities with the algorithm given in [7] that expands the Krylov subspace, adding approximated eigenvectors corresponding to eigenvalues close to zero using the orthogonality of the residuals. In our method, instead of adding approximated eigenvectors to the Krylov subspace, we choose an appropriate domain decomposition, which we defined in Sect. 5.1, to eliminate eigenvectors corresponding to eigenvalues close to zero determining the contraction projection space; see Sect. 5.2.

We describe the DDDM algorithm below. The DDDM algorithm is the same as the standard deflation algorithm. We define the algorithm within the standard preconditioned CG algorithm. The two-level algorithm described below has a much shorter calculation time owing to the form of the contraction projection matrix (51).

6.1 Preconditioned CG method

For use in Theorem 6 below, we note here the general preconditioned CG algorithm. Let $\bar{\boldsymbol{M}} \in \mathbf{R}^{(n-r)\times(n-r)}$ be an arbitrary precondition matrix, and let $\bar{\boldsymbol{x}}_0 \in \mathbf{R}^{n-r}$ be an arbitrary initial guess of the solution vector. Let the residual vector $\bar{\boldsymbol{r}}_0 \in \mathbf{R}^{n-r}$ and the gradient vector $\bar{\boldsymbol{p}}_0 \in \mathbf{R}^{n-r}$ be

$$\bar{\boldsymbol{r}}^0 = \bar{\boldsymbol{b}} - \bar{\boldsymbol{A}}\bar{\boldsymbol{x}}^0, \tag{63}$$

$$\bar{\boldsymbol{p}}^0 = \bar{\boldsymbol{M}}\bar{\boldsymbol{r}}^0. \tag{64}$$

Then, repeat for $k = 1, 2, \cdots$

$$\alpha^k = \frac{(\bar{\boldsymbol{r}}^k, \bar{\boldsymbol{M}}\bar{\boldsymbol{r}}^k)}{(\bar{\boldsymbol{p}}^k, \bar{\boldsymbol{A}}\bar{\boldsymbol{p}}^k)}, \tag{65}$$

$$\bar{\boldsymbol{x}}^{k+1} = \bar{\boldsymbol{x}}^k + \alpha^k \bar{\boldsymbol{p}}^k, \tag{66}$$

$$\bar{\boldsymbol{r}}^{k+1} = \bar{\boldsymbol{r}}^k - \alpha^k \bar{\boldsymbol{A}}\bar{\boldsymbol{p}}^k, \tag{67}$$

$$\beta^k = \frac{(\bar{\boldsymbol{r}}^{k+1}, \bar{\boldsymbol{M}}\bar{\boldsymbol{r}}^{k+1})}{(\bar{\boldsymbol{r}}^k, \bar{\boldsymbol{M}}\bar{\boldsymbol{r}}^k)}, \tag{68}$$

$$\bar{\boldsymbol{p}}^{k+1} = \bar{\boldsymbol{M}}\bar{\boldsymbol{r}}^{k+1} + \beta^k \bar{\boldsymbol{p}}^k. \tag{69}$$

(a)

$$\begin{tikzcd} V \arrow[r, "\bar{\boldsymbol{A}}^{-1}"] \arrow[dr, phantom, "\circlearrowright"] & V \arrow[d, "\bar{\boldsymbol{P}}_{\bar{\boldsymbol{A}}}"] \\ \bar{\boldsymbol{P}}_{\bar{\boldsymbol{A}}}V \arrow[u, "\bar{\boldsymbol{P}}_{\bar{\boldsymbol{A}}}^T"] \arrow[r, "\bar{\boldsymbol{P}}_{\bar{\boldsymbol{A}}}\bar{\boldsymbol{A}}^{-1}\bar{\boldsymbol{P}}_{\bar{\boldsymbol{A}}}^T = (\bar{\boldsymbol{A}}_{\boldsymbol{F}}^{-1})^*"'] & \bar{\boldsymbol{P}}_{\bar{\boldsymbol{A}}}V \end{tikzcd}$$

(b)

$$\begin{tikzcd} V \arrow[r, "\bar{\boldsymbol{R}}"] \arrow[dr, phantom, "\circlearrowright"] & V \arrow[d, "\boldsymbol{I} - \bar{\boldsymbol{P}}_{\bar{\boldsymbol{A}}}"] \\ (\boldsymbol{I} - \bar{\boldsymbol{P}}_{\bar{\boldsymbol{A}}})V \arrow[u, "(\boldsymbol{I} - \bar{\boldsymbol{P}}_{\bar{\boldsymbol{A}}})^T"] \arrow[r, "(\boldsymbol{I} - \bar{\boldsymbol{P}}_{\bar{\boldsymbol{A}}})\bar{\boldsymbol{R}}(\boldsymbol{I} - \bar{\boldsymbol{P}}_{\bar{\boldsymbol{A}}})^T"'] & (\boldsymbol{I} - \bar{\boldsymbol{P}}_{\bar{\boldsymbol{A}}})V \end{tikzcd}$$

Fig. 13 Relationship between the direct sum decomposition and the precondition matrix. $\bar{\boldsymbol{R}}$ is an arbitrary precondition matrix in the fine space V

6.2 Definition of the precondition matrix and the DDDM algorithm

Our problem is (17), which we obtained by imposing the constraint condition on (12). Let the precondition matrix be $\bar{\boldsymbol{M}}$. We multiply both sides of (17) by $\bar{\boldsymbol{M}}$ to obtain

$$\bar{\boldsymbol{M}}\bar{\boldsymbol{A}}\bar{\boldsymbol{x}} = \bar{\boldsymbol{M}}\bar{\boldsymbol{b}}. \tag{70}$$

Along with the direct sum decomposition (55), we assume $\bar{\boldsymbol{M}}$ as

$$\bar{\boldsymbol{M}} = \bar{\boldsymbol{P}}_{\bar{A}}\bar{\boldsymbol{A}}^{-1}\bar{\boldsymbol{P}}_{\bar{A}}^T + (\boldsymbol{I} - \bar{\boldsymbol{P}}_{\bar{A}})\bar{\boldsymbol{R}}(\boldsymbol{I} - \bar{\boldsymbol{P}}_{\bar{A}}^T), \tag{71}$$

where $\bar{\boldsymbol{R}} \in \mathbf{R}^{(n-r)\times(n-r)}$ is a symmetric positive definite matrix that plays the role of a precondition matrix in the fine space. The relationship between the decomposition (55) and the precondition matrix (71) is shown in Fig. 13.

The second term of (71) is a precondition on the fine space. The first term on the right side of (71), which eliminates the residuals that comprise the lower modes, coincides with the pullback $(\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^*$ from (56) in Theorem 4, and then

$$\bar{\boldsymbol{M}} = (\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^* + (\boldsymbol{I} - \bar{\boldsymbol{P}}_{\bar{A}})\bar{\boldsymbol{R}}(\boldsymbol{I} - \bar{\boldsymbol{P}}_{\bar{A}}^T). \tag{72}$$

The first term on the right side of (71) or (72), which we have already seen in (50), is

$$\bar{\boldsymbol{P}}_{\bar{A}}\bar{\boldsymbol{A}}^{-1}\bar{\boldsymbol{P}}_{\bar{A}}^T = (\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^* = \bar{\boldsymbol{F}}(\bar{\boldsymbol{F}}^T\bar{\boldsymbol{A}}\bar{\boldsymbol{F}})^{-1}\bar{\boldsymbol{F}}^T. \tag{73}$$

This expression hides $\bar{\boldsymbol{A}}^{-1}$ by the pullback; see Fig. 11. $\bar{\boldsymbol{F}}^T\bar{\boldsymbol{A}}\bar{\boldsymbol{F}} \in \mathbf{R}^{6N\times 6N}$ is a symmetric positive definite matrix on $W \equiv \mathbf{R}^{6N}$. It is so small that we can obtain $(\bar{\boldsymbol{F}}^T\bar{\boldsymbol{A}}\bar{\boldsymbol{F}})^{-1}$ can be obtained using a direct method (or LU decomposition).

Proposition 5 (Product of the projection and the precondition matrix)
It holds that

$$\bar{\boldsymbol{P}}_{\bar{A}}\bar{\boldsymbol{M}} = (\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^*. \tag{74}$$

Proof Multiplying both sides of (72) by $\bar{\boldsymbol{P}}_{\bar{A}} = (\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^*\bar{\boldsymbol{A}}$ yields

$$\bar{\boldsymbol{P}}_{\bar{A}}\bar{\boldsymbol{M}} = (\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^*\bar{\boldsymbol{A}}(\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^* + \bar{\boldsymbol{P}}_{\bar{A}}(\boldsymbol{I} - \bar{\boldsymbol{P}}_{\bar{A}})\bar{\boldsymbol{R}}(\boldsymbol{I} - \bar{\boldsymbol{P}}_{\bar{A}}^T). \tag{75}$$

From (58) in Theorem 4, the first term on the right side of this equation is $(\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^*$, and the second term becomes zero because $\bar{\boldsymbol{P}}_{\bar{A}}$ is a projection. Thus, (74) is obtained.

Theorem 6 (Range of the pullback and the gradient vector)
In the CG iteration steps, (63) to (69), let

$$\bar{\boldsymbol{x}}^0 = (\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^*\bar{\boldsymbol{b}}. \tag{76}$$

Using this vector, we define

$$\bar{\boldsymbol{r}}^0 = \bar{\boldsymbol{b}} - \bar{\boldsymbol{A}}\bar{\boldsymbol{x}}^0 \in \mathbf{R}^{n-r}, \tag{77}$$

$$\bar{\boldsymbol{p}}^0 = \bar{\boldsymbol{M}}\bar{\boldsymbol{r}}^0 \in \mathbf{R}^{n-r}. \tag{78}$$

For $k \geq 0$, the residual vector $\bar{\boldsymbol{r}}^k$ belongs to the kernel of $(\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^*$, and the gradient vector $\bar{\boldsymbol{p}}^k$ belongs to the kernel of $\bar{\boldsymbol{P}}_{\bar{A}}$, which is equal to the conjugate space $(\boldsymbol{I} - \bar{\boldsymbol{P}}_{\bar{A}})V$; see (55). Two equations thus hold for arbitrary $k \geq 0$:

$$(\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^*\bar{\boldsymbol{r}}^k = \boldsymbol{0}, \tag{79}$$

$$\bar{\boldsymbol{P}}_{\bar{A}}\bar{\boldsymbol{p}}^k = \boldsymbol{0}. \tag{80}$$

Proof We show (79) and (80) simultaneously by induction. First, we have

$$\bar{\boldsymbol{r}}^0 = \bar{\boldsymbol{b}} - \bar{\boldsymbol{A}}\bar{\boldsymbol{x}}^0 = \bar{\boldsymbol{b}} - \bar{\boldsymbol{A}}(\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^*\bar{\boldsymbol{b}}. \tag{81}$$

Multiplying both sides from the left side of this equation by $(\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^*$, and using (58) in Theorem 4, we have

$$(\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^*\bar{\boldsymbol{r}}^0 = (\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^*\bar{\boldsymbol{b}} - (\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^*\bar{\boldsymbol{A}}(\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^*\bar{\boldsymbol{b}} = (\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^*\bar{\boldsymbol{b}} - (\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^*\bar{\boldsymbol{b}} = \boldsymbol{0}. \tag{82}$$

This result shows that

$$\begin{aligned}\bar{\boldsymbol{p}}^0 &= \bar{\boldsymbol{M}}\bar{\boldsymbol{r}}^0 = ((\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^* + (\boldsymbol{I} - (\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^*\bar{\boldsymbol{A}})\bar{\boldsymbol{R}}(\boldsymbol{I} - \bar{\boldsymbol{A}}(\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^*))\bar{\boldsymbol{r}}^0 \\ &= (\boldsymbol{I} - (\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^*\bar{\boldsymbol{A}})\bar{\boldsymbol{R}}(\boldsymbol{I} - \bar{\boldsymbol{A}}(\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^*)\bar{\boldsymbol{r}}^0 = (\boldsymbol{I} - \bar{\boldsymbol{P}}_{\bar{A}})\bar{\boldsymbol{R}}\bar{\boldsymbol{r}}^0.\end{aligned} \tag{83}$$

We then have

$$\bar{\boldsymbol{P}}_{\bar{A}}\bar{\boldsymbol{p}}^0 = \bar{\boldsymbol{P}}_{\bar{A}}(\boldsymbol{I} - \bar{\boldsymbol{P}}_{\bar{A}})\bar{\boldsymbol{R}}\bar{\boldsymbol{r}}^0 = \boldsymbol{0}. \tag{84}$$

(82) and (84) show (79) and (80) for $k = 0$.

We then assume (79) and (80) for $k \geq 1$. Multiplying both sides of (67) of the CG iteration steps by $(\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^*\bar{\boldsymbol{A}}$, using (79) for k, we have

$$(\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^*\bar{\boldsymbol{r}}^{k+1} = (\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^*\bar{\boldsymbol{r}}^k - \alpha^k(\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^*\bar{\boldsymbol{A}}\bar{\boldsymbol{p}}^k = -\alpha^k(\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^*\bar{\boldsymbol{A}}\bar{\boldsymbol{p}}^k = -\alpha^k\bar{\boldsymbol{P}}_{\bar{A}}\bar{\boldsymbol{p}}^k = \boldsymbol{0}. \tag{85}$$

This shows (79) for $k + 1$. Multiplying both sides of (69) of the CG iteration steps by $\bar{\boldsymbol{P}}_{\bar{A}} = (\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^*\bar{\boldsymbol{A}}$, we obtain from (80) for k,

$$\bar{\boldsymbol{P}}_{\bar{A}}\bar{\boldsymbol{p}}^{k+1} = \bar{\boldsymbol{P}}_{\bar{A}}\bar{\boldsymbol{M}}\bar{\boldsymbol{r}}^{k+1} + \beta^k\bar{\boldsymbol{P}}_{\bar{A}}\bar{\boldsymbol{p}}^k = \bar{\boldsymbol{P}}_{\bar{A}}\bar{\boldsymbol{M}}\bar{\boldsymbol{r}}^{k+1}. \tag{86}$$

From (74) in Proposition 5, we obtain

$$\bar{\boldsymbol{P}}_{\bar{A}}\bar{\boldsymbol{p}}^{k+1} = (\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^*\bar{\boldsymbol{r}}^{k+1}. \tag{87}$$

Using (79) for $k + 1$, which we have already proved, the right side of this equation becomes zero, which means (80) for $k + 1$.

Theorem 6 shows that the residual vectors $\bar{\boldsymbol{r}}^k,\ k \geq 1$ that appear in the iteration steps stay in the kernel of $(\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^*$, and the gradient vectors $\bar{\boldsymbol{p}}^k$, $k \geq 1$, stay in the fine space $(\boldsymbol{I} - \bar{\boldsymbol{P}}_{\bar{A}})V$, which means that none of these ever steps over to the counter space.

Remark 2 *(Advantageous effect of the definition of the deflation projection. See also Remark 1 in Sect. 5.2)*

a) In the CG iteration from (65) to (69), the matrix–vector products that are necessary for updating the steps from k to $k + 1$ are $\bar{\boldsymbol{A}}\bar{\boldsymbol{p}}^k$ and $\bar{\boldsymbol{M}}\bar{\boldsymbol{r}}^k$, and $\bar{\boldsymbol{p}}_k$ is determined by $\bar{\boldsymbol{M}}\bar{\boldsymbol{r}}^k$. Theorem 6 shows that $(\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^*\bar{\boldsymbol{r}}^k = \boldsymbol{0}$ for all $k \geq 0$; therefore, we have

$$\begin{aligned}\bar{\boldsymbol{M}}\bar{\boldsymbol{r}}^k &= (\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^*\bar{\boldsymbol{r}}^k + (\boldsymbol{I} - (\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^*\bar{\boldsymbol{A}})\bar{\boldsymbol{R}}(\boldsymbol{I} - \bar{\boldsymbol{A}}(\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^*)\bar{\boldsymbol{r}}^k \\ &= (\boldsymbol{I} - (\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^*\bar{\boldsymbol{A}})\bar{\boldsymbol{R}}\bar{\boldsymbol{r}}^k.\end{aligned} \tag{88}$$

Thus, the property $(\bar{\boldsymbol{A}}_{\bar{F}}^{-1})^*\bar{\boldsymbol{r}}^k = \boldsymbol{0}$ eliminates one matrix–vector product in each CG iteration step.

b) The reduction of the calculation in (88) is given by the definition of the contraction projection (51) and the complementary projection defined in the precondition matrix (71).

6.3 Performance of the DDDM algorithm

We discuss the performance of the DDDM. Although we parallelize the DDDM, we assumed only a single process calculation in this section.

In general, depending on their thickness, convergence is difficult to achieve for thin-plate models when using iterative methods. We thus use thin plate models for our benchmarks.

Remark 3 *(Precondition matrix for the fine space in the DDDM)*

We give the precondition matrix for the fine space in the DDDM as $\bar{\boldsymbol{R}}$ in the definition for $\bar{\boldsymbol{M}}$ in (71) or (72). Although we can set $\bar{\boldsymbol{R}}$ to be any precondition matrix, we assume $\bar{\boldsymbol{R}}$ is a matrix of the successive symmetric overrelaxation (SSOR) for the following reasons.

The DDDM solver for our benchmarks is incorporated into FrontISTR [32], an open-source structural analysis code. FrontISTR has the SSOR preconditioned CG solver as a standard solver. The choice of the SSOR precondition for the DDDM enables a fair comparison between the SSOR preconditioned CG solver on the entire space and the SSOR-preconditioned-on-the-fine-space DDDM solver. In the following benchmarks, we switch between the SSOR-preconditioned CG solver and the SSOR-preconditioned DDDM solver. *SSOR below refers to the SSOR-preconditioned CG method, and DDDM below refers to the SSOR-preconditioned-on-the-fine-space of DDDM.*

In this section, the computer used was a cluster computer with Intel Xeon E5-2670, operating at 2.6 GHz, 16 cores $\times$ 13 nodes, and 128 GB of RAM per node. We do not parallelize with processes or the multi-threads. We used METIS [33] for the domain decomposition in the DDDM solver.

We built the plate model by arranging 1mm $\times$ 1mm $\times$ 1mm hexahedral linear elements in the same way as in Sect. 2.2. The x, y, and z elements were aligned in the directions of the x, y, and z-axes, respectively, except for some models noted below. We refer to the "$x \times y \times z$" plate in the same manner as in Sect. 2.2. The direction of the plate, constraint conditions, and loading conditions were likewise the same as in Sect. 2.2; see Fig. 1.

The analysis is the same cantilever-type dead weight analysis as conducted in Sect. 2.2. We ran the analysis starting with the $300 \times 10 \times 300$ model and then reducing the thickness (i.e., the number of hexahedral elements in the y-axis direction) to 9, 8, 7, 6, 5, 4, 3, 2, and 1. The DDDM converged even with $y = 1$. We then set y as thin as $y = 0.5$, 0.2, and 0.1.

We set the tolerance as 1.0×10^{-7} in all cases and set the maximum number of the iterations as 10,000 for both the SSOR and DDDM. We considered that the iteration did not converge when the number of the iteration reached the maximum number, and there was no tendency for a decrease in the relative residual error.

We summarize the benchmark result in Table 5.

Stability of the DDDM solver. Table 5 presents the stability of the DDDM solver in the sense that the SSOR converged with the $300 \times 10 \times 300$ model but failed to converge with the $300 \times 9 \times 300$ model. This tendency shows the difficulty of handling thin plates using the SSOR preconditioned CG method. Meanwhile, the DDDM converged from

Table 5 Stability of the DDDM solver. As y decreased from 10, the SSOR diverged at $y = 9$, whereas the DDDM converged to $y = 0.2$. The number of iterations for the DDDM was as low as approximately 40 to 80, whereas that for the SSOR was 4127. The thickness was so small at $y = 1$ and lower than the number of iterations increased. The number of subdomains N was determined by preliminary benchmark tests

^1y	^2n	^2r	SSOR^0		DDDM^0		
			^{3}Rep	^{3}Time	^4N	^{3}Rep	^{3}Time
10	2,989,833	3,311	4,127	2,419	1,500	48	140.46
9	2,718,030	3,010	—	—	2,000	47	128.39
8	2,446,277	2,709	—	—	1,200	52	115.49
7	2,174,424	2,408	—	—	1,500	51	102.17
6	1,902,621	2,107	—	—	1,500	53	91.60
5	1,630,818	1,806	—	—	1,500	54	78.87
4	1,359,015	1,505	—	—	1,000	83	70.23
3	1,087,212	1,204	—	—	2,000	59	54.36
2	815,409	903	—	—	2,000	84	43.50
1	543,606	602	—	—	2,000	148	33.93
0.5	543,606	602	—	—	3,500	265	58.38
0.2	543,606	602	—	—	7,000	676	165.19
0.1	543,606	602	—	—	—	—	—

^{0} The tolerance value of the residual error for the convergence is 1.0×10^{-7}
^{1} y: thickness of the plates
^{2} n and r: number of DOFs and number of the constraints, respectively
^{3} Rep: number of the iterations, Time: total calculation time[s]
^{4} N: number of of subdomains

Table 6 Performance of the DDDM solver for the $300 \times 10 \times 300$ model. The result is taken from the first row of Table 5

y	n	r	SSOR^0		$^{(1')}$Rep	DDDM^0		(2)/(2')
			$^{(1)}$Rep	$^{(2)}$Time		$^{(2')}$Time	(1)/(1')	
10	2,989,833	3,311	4,127	2,419	48	140.45	85.9	17.2

^{0} The tolerance value of the residual error for the convergence is 1.0×10^{-7}

the $300 \times 10 \times 300$ model to the $300 \times 0.2 \times 300$ model but did not converge for the $300 \times 0.1 \times 300$ model. The number of iterations and calculation time of the DDDM decreased from the $300 \times 10 \times 300$ model to the $300 \times 1 \times 300$ model, presumably because of the decrease in the number of DOFs. However, the trend turned at the $300 \times 0.5 \times 300$ model, which shows the difficulty of convergence with a reduction in the thickness even when using the DDDM.

Performance of the DDDM solver. Table 6 compares the performances of the DDDM solver and SSOR solver taken from the first row of Table 5 for $y = 10$. We can see that the DDDM solver is 85.9 times as fast as the SSOR solver in terms of the number of iterations and 17.2 times as fast as the calculation time.

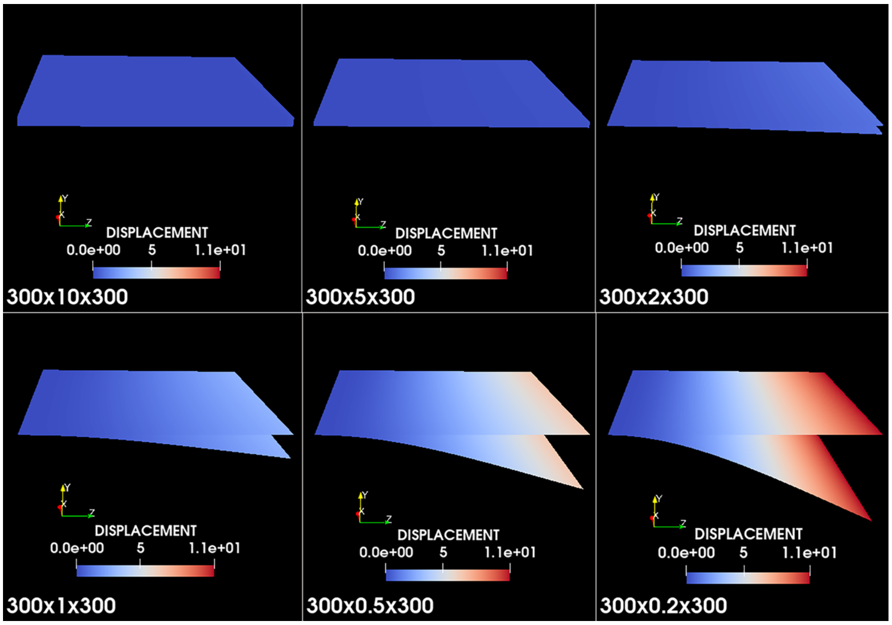

Fig. 14 Results of the thin model benchmark. Six cases from Table 5 are shown. We enlarged the deformations by a factor of 10 in all cases. The color contours show the displacement norms with the same range

We show the analysis results of the $300 \times 10 \times 300$, $300 \times 5 \times 300$, $300 \times 2 \times 300$, $300 \times 1 \times 300$, $300 \times 0.5 \times 300$, and $300 \times 0.2 \times 300$ models in Fig. 14. The color represents the displacement norms, and we enlarged the deformations by a factor of 10 in all cases.

7 Parallelization

As stated in Sect. 1 and Sect. 2.3, our solver uses two sets of domain decomposition. One is for the generation of the coarse grid, which corresponds to the structure of the DDDM, and we use the other for parallelizing the CG method in the DDDM.

7.1 Overview of the parallelization

We decompose the entire body Ω into non-overlapping subspaces:

$$\Omega = \bigcup_{I=1}^{M} \Omega_I,$$

as in Fig. 15, where M is the number of parallel processes, and each Ω_I corresponds to each of the parallel processes. The domain decomposition here is “nodal-point-

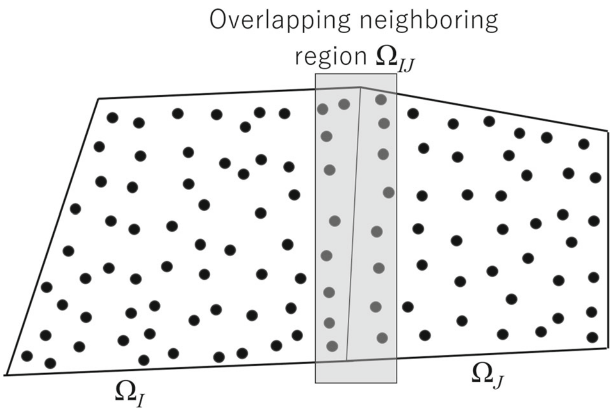

Fig. 15 Example of domain decomposition for parallelization

base" as opposed to the domain decomposition used for generation of the coarse grid is "element-base." Ω_{IJ} is a region consisting of the points nearest the neighboring both Ω_I and Ω_J. The points in the figure show the nodal points $\bar{\boldsymbol{x}}$. We fix I and J ($I \neq J$). Though both Ω_I and Ω_J have other neighboring subspaces, we omit them for an explanation. We apply the CG method to the extended subdomains $\Omega_I \cup \Omega_{IJ}$ and $\Omega_J \cup \Omega_{IJ}$ in parallel. The CG iteration refers to the nodal value in $\Omega_I \cup \Omega_{IJ}$ and also $\Omega_J \cup \Omega_{IJ}$ updating the values of the nodal points in the iteration process through the communications between the parallel processes I and J. In the CG process, the points in Ω_{IJ} have different values in each of the parallel processes I and J. After some number of the CG iterations, all the nodal values converge to some values, including the points in Ω_{IJ} within a given residual error range.

In the DDDM algorithm, we parallelize the coarse grid generation process in the parallel CG method. As noted in Sect. 5.2, we use the contraction matrix $\bar{\boldsymbol{A}}_{\bar{\boldsymbol{F}}} = \bar{\boldsymbol{F}}^T \bar{\boldsymbol{A}} \bar{\boldsymbol{F}} \in \mathbf{R}^{6N \times 6N}$ defined by Equation (49) in the whole space. The components of $\bar{\boldsymbol{A}}$ and $\bar{\boldsymbol{F}}$ are distributed to some parallel process I, we then calculate $(\bar{\boldsymbol{F}}^T \bar{\boldsymbol{A}} \bar{\boldsymbol{F}})_I$ independently in the process I. We gather those components to the parent MPI process. We apply the LU decomposition to $\bar{\boldsymbol{F}}^T \bar{\boldsymbol{A}} \bar{\boldsymbol{F}}$. We then apply $(\bar{\boldsymbol{F}}^T \bar{\boldsymbol{A}} \bar{\boldsymbol{F}})^{-1}$, which we distribute to each parallel process in the CG process in parallel for each I. The parallel process I includes not only the inner points of Ω_I but points in Ω_{IJ} outside of Ω_I.

7.2 Parallel performance

We present here the cantilever-type cuboid model under the dead weight conditions, which is the same analysis as in Sect. 6.3; however, we assumed a larger model with dimensions of $500 \times 30 \times 500$. We compared the result of the DDDM with the result of the SSOR. We chose the configuration referring to Sect. 6.3. The configuration of $500 \times 30 \times 500$ was almost the lower bound for convergence for the SSOR concerning the thickness; i.e., we cannot set the thickness to be less than 30 in this configuration,

Table 7 Parallel performance of the DDDM and comparison with the SSOR. The configuration $500 \times 30 \times 500$ is almost the lower bound concerning the thickness $y = 30$ by the lower limit of the convergence of the SSOR

Processes	1	2	4	8	16	24	32
(a)DDDM[s^0]	1274.6	647.7	399.9	281.1	201.3	184.1	172.6
(b)SSOR[s^0]	19075.6	12251.3	6458.3	3448.5	1925.9	1419.0	1249.0
(b)/(a)	14.96	18.92	16.156	12.27	9.57	7.71	7.23

^{0} The tolerance value of the residual error for the convergence to 1.0×10^{-7}

as the SSOR does not converge. The computer used was the Oakbridge-CX system [34] installed at the University of Tokyo.

The number of elements, DOFs, and constraints were 7,500,000, $n = 23,343,093$, and $r = 15,531$, respectively. We conducted the analyses using 1, 2, 4, 8, 16, 24, and 32 processes. The number of subdomains N for the DDDM (see Sect. 5.1) was set at 2500 for these processes by conducting preliminary benchmarks.

The performance results are given in Table 7. Although the superiority of the DDDM is clear, the parallel performance of the DDDM decreases with an increasing number of processes in contrast with the parallel performance of the SSOR. This tendency is because the larger number of parallel processes increases the communication traffic.

8 Conclusions

Static structural problems discretized by the FEA method without constraint conditions have singular stiffness matrices, and the kernel of the stiffness matrices has six eigenvectors with duplicated zero eigenvalues. The static equations become solvable by imposing at least six forced displacement conditions (i.e., constraint conditions).

The starting point of the DDDM algorithm is that we apply the (parallel) CG method to the entire domain as the basic framework rather than using the direct method as in the DDM. We decompose the entire domain in two ways. One decomposition parallelizes the CG method, and the other generates the coarse space based on the rigid-body modes.

We discussed the distance between the eigenvectors and the kernel of the unconstrained stiffness matrices. The basis of the kernel represents the rigid-body motions. The small distance between the eigenvectors with small eigenvalues gives rise to generating the space of the coarse motions together with the domain decomposition.

We construct the solver algorithm to remove the coarse space obtained by the coarse motion from the solution space. The iteration space is handed over to the fine space using the deflation and two-level algorithm. The two-level method contributes to high-speed performance.

We conducted benchmark tests of the elastic static analysis for thin plate models, and the results showed the high-speed performance and stability of the DDDM. We also presented a brief overview of the parallelization and its performance.

Acknowledgements The author expresses great appreciation to Prof. Hiroshi Okuda (The University of Tokyo), Dr. Masataka Sawada (Central Research Institute of Electric Power Industry), Dr. Muneo Hori (Japan Agency for Marine-Earth Science and Technology), Prof. Ichiro Hagiwara (Meiji University), and Mr. Kuniaki Koike (Advanced Simulation Technology of Mechanics R&D, Co., Ltd) for their support of this work.

Funding This work was partly supported through projects commissioned by the Agency for Natural Resources and Energy (ANRE), Ministry of Economy, Trade and Industry (METI), Japan.

Code Availability All the data used for the current study are available from the corresponding author upon reasonable request. However, the DDDM solver source code cannot be made available because it has been jointly developed with the Central Research Institute of Electric Power Industry.

Declarations

Conflict of interest The author declares that he has no conflict of interest.

References

1. Miyamura T, Yamada T (2019) Feasibility study of full-scale elastic-plastic seismic response analysis of nuclear power plant. Mech. Eng. J. 6(6):19–002811900281. https://doi.org/10.1299/mej.19-00281
2. Miyamura T, Hori M (2015) Large-scale seismic response analysis of super-high-rise steel building considering soil-structure interaction using K computer. Int J High-Rise Build 4(1):75–83
3. Wu YS, Yang YB, Yau JD (2001) Three-dimensional analysis of train-rail-bridge interaction problems. Veh Syst Dyn 36(1):1–35
4. Yamada H, Miura H, Ebisawa K (2019) Improvement of fragility evaluation on seismic PRA—an evaluation method on realistic response of component considering dynamic nonlinear characteristics of building and enhancement of sub-response factor regarding input seismic motion. Research Report of Central Research Institute of Electric Power Industry, O18010, in Japanese
5. Morgan RB (1995) A restarted GMRES method augmented with eigenvectors. SIAM J Matrix Anal Appl 16(4):1154–1171
6. Chapman A, Saad Y (1996) Deflated and augmented Krylov subspace techniques. Numer Linear Algebra Appl 4(1):43–66
7. Saad Y, Yeung M, Erhel J (2000) Guyomarc'h: a deflated version of the conjugate gradient algorithm. SIAM J Sci Comput 21(5):1909–1926
8. Nabben R, Vulk C (2006) A comparison of deflation and the balancing preconditioner. SIAM J Sci Comput 27(5):1742–1759
9. Frank J, Vuik C (2001) On the construction of deflation-based preconditioner. SIAM J Sci Comput 23(2):442–462
10. Jönsthövel TB, Gijzen MB, Vuik C (2012) Comparison of the deflated preconditioned conjugate gradient method and algebraic multigrid for composite materials. Comput Mech 50:321–333
11. Nabben R, Vuik C (2004) A comparison of deflation and coarse grid correction applied to porous media flow. SIAM J Numer Anal 42(4):1631–1647
12. Nabben R, Vuik C (2008) A comparison of abstract versions of deflation, balancing and additive coarse grid correction preconditioners. Numer Linear Algebra Appl 15:355–372

13. Tang JM, Nabben R, Vulk C, Erlangga YA (2009) Comparison of two-level preconditioners derived from deflation, domain decomposition and multigrid methods. J Sci Comput 39(3):340–370
14. Nicolaides RA (1987) Deflation of conjugate gradients with applications to boundary value problems. SIAM J Numer Anal 24(2):355–365
15. Golub GH, Van Loan CF (2013) Matrix computations, 4th edn. The Johns Hopkins University Press, Baltimore, p 443
16. Jönsthövel TB, Gijzen MB, Scarpas A (2013) On the use of rigid body modes in the deflated preconditioned conjugate gradient method. SIAM J Sci Comput 35(1):207–225
17. Farhat C, Roux FX (1991) A method of finite element tearing and interconnecting and its parallel solution algorithm. Int J Numer Methods Eng 32(6):1205–1227
18. Farhat C, Lesoinne M, LeTallec P, Pierson K, Rixen D (2001) FETI-DP: A dual-primal unified FETI method-part I: a faster alternative to the two-level FETI method. Int J Numer Methods Eng 50:1523–1544
19. Mandel J, Dohrmann CR (1993) Balancing domain decomposition, communications numerical methods in engineering. Commun Numer Methods Eng 9:233–241
20. Mandel J, Dohrmann CR (2003) Convergence of a balancing domain decomposition by constraints and energy minimization. Numer Linear Algebra Appl 10:639–659
21. Bathe KJ (2014) Finite element procedures, 2nd edn. Prentice Hall, Inc., Hoboken
22. Miyamura T, Takaya T, Yoshimura S, Hori M (2014) Improvement of balancing domain decomposition method for problem with multi-point constraints, Barcelona, Spain. In: 11th World Congress on Computational Mechanics (WCCM XI)
23. Baggio R, Franceschini A, Spiezia N, Janna C (2017) Rigid body modes deflation of the preconditioned conjugate gradient in the solution of discretized structural problems. Comput Struct 185:15–26
24. Shioya R, Kanayama R, Tagami H, Ogino M (2000) 3D large scale structural analysis using balancing domain decomposition method. Trans JSCES 2:139–144 ((**in Japanese**))
25. Smith B, Bjørstad P, Grop W (1996) Domain decomposition. University Press, Cambridge
26. Falgout RD, Vassilevski PS, Zikatanov LT (2005) On two-grid convergence estimates. Numer. Linear Algebra Appl. 12:471–494
27. Ciaramella G, Vanzan T (2022) Substructured two-grid and multi-grid domain decomposition methods. Numer. Algorithms 91:413–448
28. Motoyama H, Sawada M, Hotta W, Ohtsuka Y, Akiba H, Hori M (2021) Development of a general-purpose parallel finite element method for analyzing earthquake engineering problems. Earthq Eng Struct Dyn 50(15):4180–4189
29. Garatani K, Okuda H, Yagawa G (1999) A study on solution method for large-scaled parallel FEM (performance evaluation of two methods based on DDM). In: Transaction of JSCES, 19990009 (in Japanese)
30. Adventure Project. https://adventure.sys.t.u-tokyo.ac.jp/ (2021)
31. Meyer CD (2000) Matrix analysis and applied linear algebra. SIAM, Philadelphia
32. FrontISTR Commons. https://manual.frontistr.com/en/ (2021)
33. METIS. http://glaros.dtc.umn.edu/gkhome/metis/metis/overview/
34. Oakbridge-CX (2023) https://www.cc.u-tokyo.ac.jp/en/supercomputer/obcx/service/