박 사 학 위 논 문

Ph.D. Dissertation

병렬 다중 레벨 부구조법의 부하 분산 및 비선형 모델 축소를 위한 성긴 격자 투영

Load balancing for parallel multilevel substructuring and coarse mesh projection for nonlinear model reduction

2022

현 철 규 (玄 喆 奎 Hyun, Cheolgyu)

한 국 과 학 기 술 원

Korea Advanced Institute of Science and Technology

박 사 학 위 논 문

병렬 다중 레벨 부구조법의 부하 분산 및 비선형 모델 축소를 위한 성긴 격자 투영

2022

현 철 규

한 국 과 학 기 술 원

기계항공공학부/기계공학과

병렬 다중 레벨 부구조법의 부하 분산 및 비선형 모델 축소를 위한 성긴 격자 투영

현 철 규

위 논문은 한국과학기술원 박사학위논문으로
학위논문 심사위원회의 심사를 통과하였음

2021년 12월 6일

심사위원장 이 필 승 (인)
심 사 위 원 박 용 화 (인)
심 사 위 원 부 승 환 (인)
심 사 위 원 유 승 화 (인)
심 사 위 원 정 형 조 (인)

Load balancing for parallel multilevel substructuring and coarse mesh projection for nonlinear model reduction

Cheolgyu Hyun

Advisor: Phill-Seung Lee

A dissertation submitted to the faculty of
Korea Advanced Institute of Science and Technology in
partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Mechanical Engineering

Daejeon, Korea
December 6, 2021

Approved by

Phill-Seung Lee
Professor of Mechanical Engineering

The study was conducted in accordance with Code of Research Ethics[1)].

1) Declaration of Ethical Conduct in Research**:** I, as a graduate student of Korea Advanced Institute of Science and Technology, hereby declare that I have not committed any act that may damage the credibility of my research. This includes, but is not limited to, falsification, thesis written by someone else, distortion of research findings, and plagiarism. I confirm that my dissertation contains honest conclusions based on my own careful research under the guidance of my advisor.

DME

현철규. 병렬 다중 레벨 부구조법의 부하 분산 및 비선형 모델 축소를 위한 성긴 격자 투영. 기계공학과. 2022년. 83+iv 쪽. 지도교수: 이필승. (영문 논문)

Hyun, Cheolgyu. Load balancing for parallel multilevel substructuring and coarse mesh projection for nonlinear model reduction. Department of Mechanical Engineering. 2022. 83+iv pages. Advisor: Lee, Phill-Seung. (Text in English)

초 록

본 논문에서는 효율적인 선형 및 비선형 유한 요소 구조 해석을 위한 투영기반 모델 축소 방법들을 제안한다. 선형 동적 해석에서는 저차 모드들이 구조의 응답을 지배하는 특성이 있기 때문에 모드 중첩 방법이 널리 쓰인다. 그러나 대규모 시스템의 경우 모드 중첩 방법을 위한 고유치 해석은 엄청난 계산 비용을 초래한다. 또한, 비선형 모델 축소에서는 축소 모델이 생성되더라도 원래 시스템의 차원에서 비선형항을 계산해야 한다. 이러한 계산 비용은 축소 차원이 아닌 원래 차원에 비례하기 때문에, 축소 모델의 해를 계산할 때 상당한 속도 향상은 기대할 수 없다. 이러한 문제들은 병렬 다중 레벨 부구조법과 성긴 격자 투영으로 해결된다. 먼저, 효율적인 고유치 해석을 위한 병렬 자동 다중 레벨 부구조법의 부하 분산 알고리즘을 제시한다. 스레드 간의 작업 부하 균형을 위해 제안된 알고리즘은 두 가지의 세분성으로 구성된다. 제안된 알고리즘은 격자 재분할없이 자동 다중 레벨 부구조화법의 병렬 효율성을 크게 향상시킨다. 또한, 효율적인 비선형 모델 축소를 위한 성긴 격자 투영 방법을 제시한다. 제안된 방법은 시스템 영역을 나타내는 성긴 격자에서 비선형 항을 계산하므로 계산 비용이 크게 절감된다.

핵심낱말 구조 해석, 유한요소법, 모델 축소, 고유치 해석, 병렬 컴퓨팅, 비선형 해석

Abstract

This dissertation proposes projection-based model reduction methods for efficient linear and nonlinear finite element analysis of structures. In linear dynamic analysis, the mode superposition method is widely used because the lowest modes dominate the response of a structure. However, in the mode superposition method for large-scale systems, solving the generalized eigenvalue problem incurs a huge computation time. In addition, in nonlinear model reduction, computing nonlinear terms still requires operations with the original dimension even though the reduced model is obtained. Since such computations are proportional to the original dimension, no significant speedup can be expected. These challenges are addressed by parallel multilevel substructuring and coarse mesh projection. A load balancing algorithm for the parallel automated multilevel substructuring (PAMLS) method is first presented to solve the generalized eigenvalue problem efficiently. To balance the workload among threads, the proposed algorithm consists of two types of granularity. Without repartitioning, the proposed algorithm significantly improves the efficiency of the PAMLS method. A coarse mesh projection method is also presented for efficient nonlinear model reduction. The proposed method computes the nonlinear terms on the coarse mesh representing the domain of the system, which considerably reduces the computational cost.

Keywords Structural analysis, Finite element method, Model reduction, Eigenvalue problem, Parallel computing, Nonlinear analysis

Contents

List of Tables

List of Figures

Chapter 1. Introduction

The finite element method plays an important role in static and dynamic analyses of structures [1,2]. Despite tremendous improvement in computer performance, large-scale finite element analysis in today's engineering practice requires considerable computational cost. Model reduction aims to decrease such computational cost, which is achieved by generating a reduced model that represents the original system behaviors. In particular, projection-based model reduction methods have been widely used because of the rigorous framework that preserves the underlying structure of the original model [3]. The framework projects the original model onto a suitably defined low-dimensional subspace.

The finite element analysis only approximates the lowest frequencies and the corresponding mode shapes, and the error increases for the higher frequencies. Moreover, the lowest normal modes dominate responses in linear dynamic analysis of structures. Therefore, the standard reduction approach is the mode superposition method that reduces the number of degrees of freedom (DOFs) using partial eigensolutions. However, this eigenvalue problem in the large-scale system leads to a huge computational cost, and such cost increases even more with high modal density. It is valuable to develop an effective solution method for the eigenvalue problem accordingly. Although the automated multilevel substructuring (AMLS) method [4–7] is an effective approach to address such difficulties, workload imbalances decrease the efficiency in the parallel AMLS (PAMLS) method [8].

In nonlinear model reduction, the two-phase offline-online strategy is commonly adopted. In the offline phase, snapshots of the original model are collected, and the low-dimensional basis vectors are then computed from the snapshot data. The reduced model is constructed using the low-dimensional basis vectors. Since the frequencies and corresponding mode shapes of the original model are no longer preserved, the proper orthogonal decomposition (POD) has been widely employed to generate such low-dimensional basis vectors. In the online phase, the reduced model is solved, and the approximate solutions are obtained. However, nonlinear terms are still computed in the dimension of the original system although the reduced model is generated. Sparse sampling methods [9,10] derive the approximate nonlinear terms from some sampling points. It has been observed that numerical inaccuracies of sampling points can induce unstable solutions [11,12]. In addition, the efficiency of the sparse sampling methods deteriorates in the finite element method because nonlinear terms are vector-valued functions [13].

The goal of this dissertation is to present novel methods that overcome the drawbacks mentioned above. In order to balance the workload for the PAMLS method, a load balancing algorithm is firstly proposed. For nonlinear finite element analysis, a novel nonlinear model reduction method to compute nonlinear terms in the low-dimensional subspace is proposed.

In Chapter 2, a load balancing algorithm that improves the parallel efficiency of the PAMLS method is presented. In the PAMLS method, load balancing is highly dependent on the computation time for the transformation and

back transformation procedures corresponding to substructures. To balance the workload among threads, the proposed algorithm consists of two types of granularity: coarse-grained and fine-grained parallel algorithms. According to the level of substructures, the coarse-grained parallel algorithm splits both the transformation and back transformation procedures and assigns them to threads. Through fine-grained parallelism, more threads are exploited for the transformation of each substructure compared to threads used in the original PAMLS method. Without repartitioning, the proposed algorithm significantly improves the efficiency of the PAMLS method. This work has been published in Ref. [14], and the material given in this chapter is only slightly modified.

In Chapter 3, we present a nonlinear model reduction method using a coarse mesh, named coarse mesh projection. To compute nonlinear terms efficiently, the coarse mesh model representing the domain of the original model is constructed. Then, the reduced basis vectors for the coarse mesh model are computed by finite element interpolation of the POD basis vectors for the original model. Here, the reduced basis vectors for the coarse mesh model are referred to as the coarsened POD basis vectors. The coarse mesh model is then projected onto the subspace spanned by the coarsened POD basis vectors, and corrected at each load or time step by using the approximate solution for the original model. In this way, only displacement POD basis vectors are extracted, and there is no need to store snapshots for nonlinear terms. Whereas sparse sampling methods approximate the nonlinear terms from some sampling points, the proposed method computes the nonlinear terms on the low-dimensional coarse mesh model. By leveraging the rigor of the finite element framework, the proposed method provides reliable solutions efficiently.

In Chapter 4, conclusions and future works are provided.

Chapter 2. A load balancing algorithm for the parallel automated multilevel substructuring method

Component mode synthesis (CMS) methods [15–20] have been widely used to solve frequencies and mode shapes of large and complex structures [21–30]. In conventional CMS methods, a structural finite element (FE) model is divided into small substructures with an interface among them, and each substructure is individually reduced via truncation of its eigensolutions. Since this approach is suitable for parallel implementation that carries out many calculations simultaneously, there have been various studies to parallelize CMS methods [31–34].

Among CMS methods, the automated multilevel substructuring (AMLS) [4–7] is a successful method to reduce the large interface DOFs efficiently and robustly. The AMLS method partitions an FE model into many levels of substructures by using the nested dissection algorithm [35]. Then the FE model is transformed by synthesizing substructure eigensolutions and constraint modes. The approximate solutions are computed through the back transformation of solutions obtained by the reduced model. The AMLS method can serve as an alternative to the Lanczos method [36,37] when computing a large number of eigensolutions [38,39] and has been applied to various engineering problems [40–42].

After Kaplan [5] proposed the directions in parallelism for the AMLS method, Elssel and Voss [8] developed the parallel AMLS (PAMLS) method. Each thread first performs the AMLS transformation of a subtree consisting of substructures. The rest of the substructures (i.e. partitioned interfaces) are then transformed in parallel. Yang et al. [43] applied a multilevel approach to nonlinear implicit dynamics, where the maximum allowed imbalance among substructures is set by a static load balancer [44,45]. Both methods [8,43] successfully parallelize the computation associated with substructures, including interfaces.

The objective of this chapter is to present a novel load balancing algorithm that improves the parallel efficiency of the PAMLS method. Since the transformation and back transformation procedures spend most of the computation time in the AMLS method, we focus on improving the parallel efficiency of those procedures. In general, a load imbalance among the transformations of substructures would occur even if the number of substructures assigned to each thread is the same and the number of their degrees of freedom (DOFs) is well balanced. As a result, there is an idle time at the synchronization point before the transformation of a parent substructure. To reduce this idle time, the proposed algorithm consists of two types of granularity: coarse-grained parallelism and fine-grained parallelism.

In coarse-grained parallelism, the transformation and back transformation procedures are split into tasks corresponding to the predefined subtrees and the rest of the substructures. To balance the computational load in each task, we employ the approach of Escaig et al., in which the number of assigned tasks (i.e. substructures) is set to be larger than the number of threads [46]. We determine the number of subtrees by using a given cutoff level instead of the number of threads. Moreover, for the back transformation procedure, both the explicit [5] and

implicit [47] strategies are parallelized. In fine-grained parallelism, the transformation procedure of each substructure is split into several tasks. In the parallel regions, tasks are processed by available threads. Through the approaches in the proposed method, the efficiency of the original PAMLS method is significantly improved without repartitioning.

In Section 2.1, the solution methods for large sparse eigenvalue problems are briefly reviewed and the key concepts of CMS are introduced. In Section 2.2, the AMLS method is briefly reviewed. Section 2.3 presents the proposed algorithm in detail. In Sections 2.4 and 2.5, the performance of the proposed algorithm is investigated through numerical examples. Conclusions are given in Section 2.6.

2.1. Solution methods for the generalized eigenvalue problem

We consider the generalized eigenvalue problem

$$\mathbf{K}\boldsymbol{\varphi} = \lambda\mathbf{M}\boldsymbol{\varphi} \tag{2.1}$$

where $\mathbf{K}$ and $\mathbf{M}$ are the stiffness and mass matrices of an original (i.e. non-reduced) finite element (FE) model, respectively, λ is the eigenvalue, and $\boldsymbol{\varphi}$ is the corresponding eigenvector.

In general, $\mathbf{K}$ and $\mathbf{M}$ are large and sparse matrices, and a small subset of eigenvalues and corresponding eigenvectors is required. The n_{ev} eigensolutions are written as

$$\mathbf{K}\boldsymbol{\Phi} = \mathbf{M}\boldsymbol{\Phi}\boldsymbol{\Lambda}, \tag{2.2}$$

where columns of $\boldsymbol{\Phi}$ are the eigenvectors and $\boldsymbol{\Lambda}$ is a diagonal matrix consisting of eigenvalues.

2.1.1. Subspace iteration

The subspace iteration method, originally proposed by Bathe and Wilson [48], is an efficient solution method for frequencies and mode shapes of structures. The equations in the subspace iteration method include the establishment of starting vectors, Ritz analysis, error measure, and Sturm sequence check.

For each iteration $k = 1, 2, \cdots$ the following equations are used:

$$\mathbf{K}\bar{\mathbf{X}}_{k+1} = \mathbf{M}\mathbf{X}_k, \tag{2.3}$$

$$\mathbf{K}_{k+1} = \bar{\mathbf{X}}_{k+1}^T\mathbf{K}\bar{\mathbf{X}}_{k+1}, \tag{2.4}$$

$$\mathbf{M}_{k+1} = \bar{\mathbf{X}}_{k+1}\mathbf{M}\bar{\mathbf{X}}_{k+1}, \tag{2.5}$$

$$\mathbf{K}_{k+1}\mathbf{Q}_{k+1} = \mathbf{M}_{k+1}\mathbf{Q}_{k+1}\boldsymbol{\Lambda}_{k+1}, \tag{2.6}$$

$$\mathbf{X}_{k+1} = \bar{\mathbf{X}}_{k+1}\mathbf{Q}_{k+1}, \tag{2.7}$$

where $\mathbf{X}_k$ is a mass orthonormalized vector, and $\mathbf{Q}_{k+1}$ and $\boldsymbol{\Lambda}_{k+1}$ are eigenvector and eigenvalue matrices for $\mathbf{K}_{k+1}$ and $\mathbf{M}_{k+1}$. When $\mathbf{X}_1$ is not orthogonal to $\boldsymbol{\Phi}$, we have

$$\boldsymbol{\Lambda}_{k+1} \to \boldsymbol{\Lambda} \quad \text{and} \quad \mathbf{X}_{k+1} \to \boldsymbol{\Phi} \quad \text{as} \quad k \to \infty. \tag{2.8}$$

In practice, there are three steps of the subspace iteration. First, the number of iteration vectors in $\mathbf{X}_1$ is should be larger than the number of eigenvalues required. Bathe [49] suggested the following formula

$$n_i = \max\{n_{ev} + 8, \quad 2n_{ev}\}, \tag{2.9}$$

where n_i denotes the number of iteration vectors.

The starting vectors are often established in such a way that unit vectors excite the corresponding DOFs of the

smallest ratios K_{ii}/M_{ii} and the last column is a random vector.

Second, in each iteration (k), convergence is measured as follows:

$$\left[1-\frac{(\lambda_i^{(k+1)})^2}{(\mathbf{q}_i^{(k+1)})^T\mathbf{q}_i^{(k+1)}}\right]^{1/2} \le 10^{-2s} \quad \text{for} \quad i=1,2,\cdots,n_e, \quad k\ge 2 \tag{2.10}$$

where $\lambda_i^{(k+1)}$ is the ith approximate eigenvalue in iteration (k), $\mathbf{q}_i^{(k+1)}$ is the ith column of $\mathbf{Q}_{k+1}$, and the tolerance 10^{-2s} is used for the eigenvalues to be accurate to about $2s$ digits.

Third, after convergence of the iteration, the Sturm sequence check is performed to verify that all required eigenvalues have been calculated. The larger number of iteration vectors is used when the Sturm sequence check is not passed.

Recently, Kim and Bathe [50] proposed the enriched subspace iteration method, which is also be parallelized in shared and distributed memory processing. In the subspace iteration method, the subspace spanned by iteration vectors is turned towards the n_{ev}-dimensional least dominant subspace. Using the amount of turning of the iteration vectors significantly improves the computational efficiency of the basic subspace iteration method.

2.1.2. Lanczos method

The Lanczos method [36] is one of the most important algorithms among the Krylov methods that approximates the eigensolutions on the Krylov subspace. The order-m Krylov subspace $\mathcal{K}$ is generated by a matrix $\mathbf{A}$ and a vector $\mathbf{x}$ as

$$\mathcal{K}(\mathbf{A},\mathbf{x},m)=\text{span}\{\mathbf{x},\mathbf{A}\mathbf{x},\mathbf{A}^2\mathbf{x},\ldots,\mathbf{A}^{m-1}\mathbf{x}\}\,. \tag{2.11}$$

The Lanczos method produces an orthonormal basis for Krylov subspace as

$$\mathcal{K}=\text{span}\{\mathbf{x},(\mathbf{K}^{-1}\mathbf{M})\mathbf{x},(\mathbf{K}^{-1}\mathbf{M})^2\mathbf{x},\ldots,(\mathbf{K}^{-1}\mathbf{M})^{m-1}\mathbf{x}\}\,, \tag{2.12}$$

where $\mathbf{x}$ is an arbitrary starting vector. The generalized eigenvalue problem is transformed into a low-dimensional standard eigenvalue problem with a tridiagonal coefficient matrix. For $i=1,2,\cdots,m$, the basic steps of the Lanczos method are

$$\mathbf{x}_1=\frac{\mathbf{x}}{\gamma} \quad \text{with} \quad \gamma=(\mathbf{x}^T\mathbf{M}\mathbf{x})^{1/2}, \tag{2.13}$$

$$\mathbf{K}\bar{\mathbf{x}}_i=\mathbf{M}\mathbf{x}_i, \tag{2.14}$$

$$\alpha_i=\bar{\mathbf{x}}_i^T\mathbf{M}\mathbf{x}_i, \tag{2.15}$$

$$\tilde{\mathbf{x}}_i=\bar{\mathbf{x}}_i-\alpha_i\mathbf{x}_i-\beta_{i-1}\mathbf{x}_{i-1}, \tag{2.16}$$

$$\beta_i=(\tilde{\mathbf{x}}_i^T\mathbf{M}\tilde{\mathbf{x}}_i)^{1/2}, \tag{2.17}$$

$$\mathbf{x}_{i+1} = \frac{\tilde{\mathbf{x}}_i}{\beta_i}, \tag{2.18}$$

where the vectors $\mathbf{x}_i$ are mass orthonormal and satisfy the following relationship

$$\mathbf{X}_m^T(\mathbf{M}\mathbf{K}^{-1}\mathbf{M})\mathbf{X}_m = \mathbf{T}_m \tag{2.19}$$

with

$$\mathbf{X} = [\mathbf{x}_1 \quad \mathbf{x}_2 \quad \cdots \quad \mathbf{x}_m], \tag{2.20}$$

$$\mathbf{T}_q = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \beta_2 & \ddots & \ddots & \\ & & \ddots & \alpha_{m-1} & \beta_{m-1} \\ & & & \beta_{m-1} & \alpha_m \end{bmatrix}. \tag{2.21}$$

The generalized eigenvalue problem $\mathbf{K}\boldsymbol{\varphi} = \lambda\mathbf{M}\boldsymbol{\varphi}$ can be rewritten in the form

$$(\mathbf{M}\mathbf{K}^{-1}\mathbf{M})\boldsymbol{\varphi} = \frac{1}{\lambda}\mathbf{M}\boldsymbol{\varphi}, \tag{2.22}$$

and finally, we obtain

$$\mathbf{T}_m\tilde{\boldsymbol{\varphi}} = \frac{1}{\lambda}\tilde{\boldsymbol{\varphi}} \quad \text{with} \quad \boldsymbol{\varphi} = \mathbf{X}_m\tilde{\boldsymbol{\varphi}}. \tag{2.23}$$

Note that the eigenvalues and eigenvectors of $\mathbf{K}\boldsymbol{\varphi} = \lambda\mathbf{M}\boldsymbol{\varphi}$ are the reciprocals of the eigenvalues of $\mathbf{T}_m$ and $\boldsymbol{\varphi} = \mathbf{X}_m\tilde{\boldsymbol{\varphi}}$, respectively.

Lanczos recognized the numerical instability due to loss of orthogonality and suggested the full reorthogonalization. Since the computational cost for the full reorthogonalization is expensive in practice, considerable efforts have been made to address this problem [51,52]. In addition, Lehoucq, Sorensen, and Yang developed ARPACK [53], which is an open-source package based on the Arnoldi/Lanczos process. They use the implicitly restarted Arnoldi method, which is closely related to the implicitly shifted QR algorithm.

2.1.3. Dynamic condensation

In 1965, Guyan [54] proposed a static condensation method, which is an application of Gauss elimination and referred to as Guyan reduction. The model reduction in static condensation is achieved by eliminating some DOFs assumed to be quasi-static. In other words, the inertia effect is simply ignored.

We partition the stiffness matrix and corresponding displacement and force vector into the form

$$\begin{bmatrix} \mathbf{K}_{ss} & \mathbf{K}_{sm} \\ \mathbf{K}_{ms} & \mathbf{K}_{mm} \end{bmatrix} \begin{bmatrix} \mathbf{x}_s \\ \mathbf{x}_m \end{bmatrix} = \begin{bmatrix} \mathbf{f}_s \\ \mathbf{f}_m \end{bmatrix}, \tag{2.24}$$

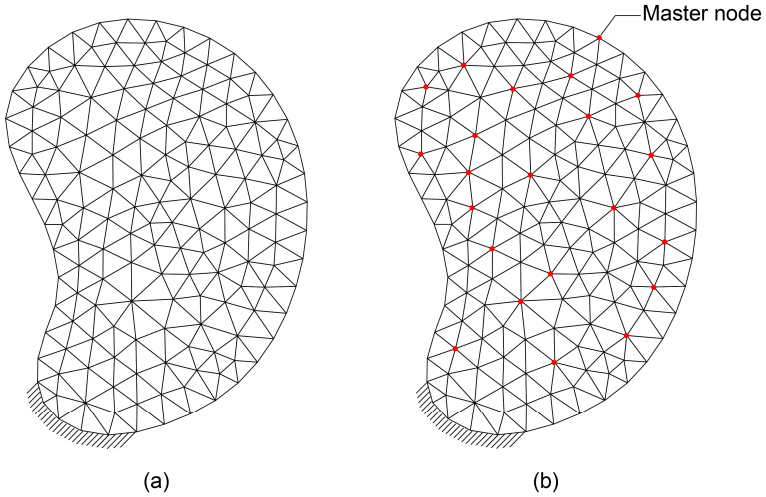


Fig. 2.1. Selection of master and slave nodes: (a) original finite element model and (b) possible selection, where the red dots are master nodes and the unselected nodes are slave nodes.

where the subscripts s and m denote the DOFs to be condensed and retained, respectively. The terms slave and master DOFs are coined by Irons [55], and a possible selection of master and slave nodes is described in Fig. 2.1. Note that the accuracy of frequency response is greatly affected by the choice of master DOFs.

Suppose the force vector $\mathbf{f}_s$ is zero, and then the displacement vector for slave DOFs $\mathbf{x}_s$ depends on $\mathbf{x}_m$. The resulting equation is

$$\begin{bmatrix} \mathbf{x}_s \\ \mathbf{x}_m \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Psi} \\ \mathbf{I} \end{bmatrix} \mathbf{x}_m \quad \text{with} \quad \boldsymbol{\Psi} = -\mathbf{K}_{sm}^{-1}\mathbf{K}_{mm}, \tag{2.25}$$

where $\mathbf{I}$ is the identity matrix.

Then, the stiffness and mass matrices can be transformed as

$$\bar{\mathbf{K}}_s = \mathbf{T}_s^T \mathbf{K} \mathbf{T}_s, \quad \bar{\mathbf{M}}_s = \mathbf{T}_s^T \mathbf{M} \mathbf{T}_s \tag{2.26}$$

with

$$\mathbf{T}_s = \begin{bmatrix} \boldsymbol{\Psi} \\ \mathbf{I} \end{bmatrix}, \tag{2.27}$$

where $\bar{\mathbf{K}}_s$ and $\bar{\mathbf{M}}_s$ are the reduced stiffness and mass matrices, respectively.

In the case of $\mathbf{f}_s = \mathbf{0}$, static condensation solves a static problem exactly in the reduced subspace. However, the solution accuracy decreases as the frequency increases. To increase the accuracy, one prefers dynamic condensation methods, which apply the static condensation to the matrix $(\mathbf{K} - \lambda\mathbf{M})$ rather than $\mathbf{K}$, viz.

$$\begin{bmatrix} \mathbf{x}_s \\ \mathbf{x}_m \end{bmatrix} = \begin{bmatrix} -(\mathbf{K}_{ss} - \lambda\mathbf{M}_{ss})^{-1}(\mathbf{K}_{sm} - \lambda\mathbf{M}_{sm}) \\ \mathbf{I} \end{bmatrix} \mathbf{x}_m. \tag{2.28}$$

Miller [56] and O'Callahan [57] employ Taylor series expansion of $(\mathbf{K}_{ss} - \lambda \mathbf{M}_{ss})^{-1}$, which includes the inertia effect. We briefly review the improved reduced system method proposed by O'Callahan [57].

Taylor series expansion of $\mathbf{x}_s$ in Eq. (2.28) is written by

$$\mathbf{x}_s = [-\mathbf{K}_{ss}^{-1}\mathbf{K}_{sm} + \lambda \mathbf{K}_{ss}^{-1}(\mathbf{M}_{sm} - \mathbf{M}_{ss}\mathbf{K}_{ss}^{-1}\mathbf{K}_{sm}) + O(\lambda^2) + O(\lambda^3) + \cdots]\mathbf{x}_m , \tag{2.29}$$

and $\mathbf{x}_s$ is approximated by neglecting the high-order terms of λ as

$$\mathbf{x}_s \approx [-\mathbf{K}_{ss}^{-1}\mathbf{K}_{sm} + \lambda \mathbf{K}_{ss}^{-1}(\mathbf{M}_{sm} - \mathbf{M}_{ss}\mathbf{K}_{ss}^{-1}\mathbf{K}_{sm})]\mathbf{x}_m . \tag{2.30}$$

Using Eq. (2.26), the unknown λ in Eq. (2.30) can be handled by

$\lambda \mathbf{x}_m = \mathbf{H}\mathbf{x}_m$ with $\mathbf{H} = \bar{\mathbf{M}}_s^{-1}\bar{\mathbf{K}}_s$,

and then

$$\mathbf{x}_s = [-\mathbf{K}_{ss}^{-1}\mathbf{K}_{sm} + \mathbf{K}_{ss}^{-1}(\mathbf{M}_{sm} - \mathbf{M}_{ss}\mathbf{K}_{ss}^{-1}\mathbf{K}_{sm})\mathbf{H}]\mathbf{x}_m . \tag{2.31}$$

The reduced stiffness and mass matrices ($\bar{\mathbf{K}}_d$ and $\bar{\mathbf{M}}_d$) are obtained as

$$\bar{\mathbf{K}}_d = \bar{\mathbf{K}}_s + \mathbf{T}_s^T\mathbf{K}\mathbf{T}_c\mathbf{H} + \mathbf{H}^T\mathbf{T}_c^T\mathbf{K}\mathbf{T}_s + \mathbf{H}^T\mathbf{T}_c^T\mathbf{K}\mathbf{T}_s\mathbf{H} , \tag{2.32}$$

$$\bar{\mathbf{M}}_d = \bar{\mathbf{M}}_s + \mathbf{T}_s^T\mathbf{M}\mathbf{T}_c\mathbf{H} + \mathbf{H}^T\mathbf{T}_c^T\mathbf{M}\mathbf{T}_s + \mathbf{H}^T\mathbf{T}_c^T\mathbf{M}\mathbf{T}_s\mathbf{H} , \tag{2.33}$$

with

$$\mathbf{T}_c = \begin{bmatrix} \mathbf{K}_{ss}^{-1}(\mathbf{M}_{sm} + \mathbf{M}_{ss}\boldsymbol{\Psi}) \\ \mathbf{0} \end{bmatrix}, \tag{2.34}$$

where $\mathbf{T}_c$ is an additional transformation matrix that includes the inertia effect. Note that the reduced matrices in Eqs (2.32) and (2.33) are fully populated. Therefore, the solution of the reduced matrices would be expensive than the original large sparse matrices.

2.1.4. Component mode synthesis

Component mode synthesis (CMS) methods are widely used to solve frequencies and mode shapes of large FE models efficiently. The solution approach, originally proposed by Hurty [15], employs a divide-and-conquer paradigm. An attractive feature of the CMS methods is that the computational cost is significantly reduced by analyzing smaller substructures instead of the large-scale original structure. The solution of the original structure is defined on its substructures (i.e. components). The frequencies and mode shapes for substructures are solved, and the dimension of each substructure is reduced by truncation of high-frequency modes. Then, the reduced model is obtained as an assemblage of the reduced substructures. CMS methods can be classified according to interface handling [58]: fixed interface, free interface, and hybrid interface methods. We briefly review the Craig-Bampton (CB) method [16], which is the most popular method among CMS methods.

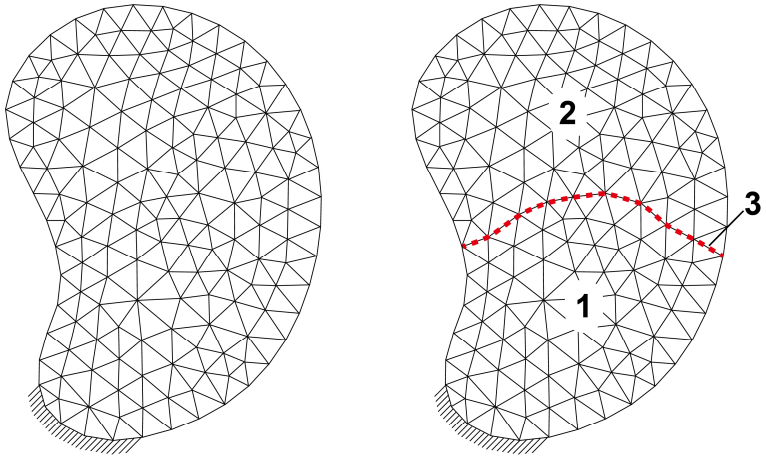


Fig. 2.2. A generic finite element model: (a) original finite element model, and (b) two substructures with the interface.

For convenience to present CB method, without loss of generality, two substructures and their interface are considered as shown in Fig. 2.2. Note that the interface is referred to as the substructure 3 in this section. We represent DOFs for substructures 1, 2, and 3 as $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{x}_3$, respectively. The stiffness and mass matrices for substructure 1 have the form

$$\mathbf{K}^{(1)} = \begin{bmatrix} \mathbf{K}_{11} & \mathbf{K}_{13} \\ \mathbf{K}_{31} & \mathbf{K}_{33}^{(1)} \end{bmatrix}, \quad \mathbf{M}^{(1)} = \begin{bmatrix} \mathbf{M}_{11} & \mathbf{M}_{13} \\ \mathbf{M}_{31} & \mathbf{M}_{33}^{(1)} \end{bmatrix}, \tag{2.35}$$

where the superscript (1) denotes the substructure 1, and $\mathbf{K}_{33}^{(1)}$ and $\mathbf{M}_{33}^{(1)}$ are the stiffness and mass matrices for substructure 3 associated with the substructure 1.

Then, $\mathbf{x}_1$ and $\mathbf{x}_3$ is expressed as

$$\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{\Phi}_1 & \mathbf{\Psi}_{13} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{q}_1 \\ \mathbf{x}_3 \end{bmatrix} = \mathbf{T}_1 \begin{bmatrix} \mathbf{q}_1 \\ \mathbf{x}_3 \end{bmatrix}, \tag{2.36}$$

where $\mathbf{T}_1$ is the transformation matrix for substructure 1, $\mathbf{I}$ is the identity matrix, and $\mathbf{q}_1$ is the generalized coordinate vector associated with the substructure eigenvector matrix $\mathbf{\Phi}_1$. The substructure eigenvector is obtained by solving the substructure eigenvalue problem

$$\mathbf{K}_{11}\mathbf{\Phi}_1 = \mathbf{M}_{11}\mathbf{\Phi}_{11}\mathbf{\Lambda}_1, \tag{2.37}$$

where $\mathbf{\Lambda}_1$ is a diagonal matrix consisting of eigenvalues. Note that the eigensolutions are truncated by a given cutoff frequency.

The constraint modes matrix $\mathbf{\Psi}_{13}$ is calculated by

$$\mathbf{\Psi}_{13} = -\mathbf{K}_{11}^{-1}\mathbf{K}_{13}, \tag{2.38}$$

where $\mathbf{\Psi}_{13}$ represents the static deformation of substructure 1 by imposing the unit displacement on substructure

3 (i.e. interface boundary).

Employing $\mathbf{T}_1$, the stiffness and mass matrices for substructure 1 are

$$\bar{\mathbf{K}}^{(1)} = \mathbf{T}_1^T \mathbf{K}^{(1)} \mathbf{T}_1 , \quad \bar{\mathbf{M}}^{(1)} = \mathbf{T}_1^T \mathbf{M}^{(1)} \mathbf{T}_1 , \tag{2.39}$$

with

$$\bar{\mathbf{K}}^{(1)} = \begin{bmatrix} \mathbf{\Lambda}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_{33}^{(1)} + \mathbf{K}_{13}^T \mathbf{\Psi}_{13} \end{bmatrix}, \tag{2.40}$$

$$\bar{\mathbf{M}}^{(1)} = \begin{bmatrix} \mathbf{I} & \mathbf{\Phi}_1^T (\mathbf{M}_{13} + \mathbf{M}_{11} \mathbf{\Psi}_{13}) \\ sym. & \mathbf{M}_{33}^{(1)} + \mathbf{\Psi}_{13}^T \mathbf{M}_{13} + \mathbf{M}_{13}^T \mathbf{\Psi}_{13} + \mathbf{\Psi}_{13}^T \mathbf{M}_{11} \mathbf{\Psi}_{13} \end{bmatrix}. \tag{2.41}$$

After the transformation of substructure 2 in a similar fashion, the reduced matrices are assembled as

$$\bar{\mathbf{K}} = \begin{bmatrix} \mathbf{\Lambda}_1 & & \\ & \mathbf{\Lambda}_2 & \\ & & \sum_{i=1}^{2} \mathbf{K}_{33}^{(i)} + \mathbf{K}_{i3}^T \mathbf{\Psi}_{i3} \end{bmatrix}, \tag{2.42}$$

$$\bar{\mathbf{M}} = \begin{bmatrix} \mathbf{I} & & \mathbf{\Phi}_1^T (\mathbf{M}_{13} + \mathbf{M}_{11} \mathbf{\Psi}_{13}) \\ & \mathbf{I} & \mathbf{\Phi}_2^T (\mathbf{M}_{23} + \mathbf{M}_{22} \mathbf{\Psi}_{23}) \\ sym. & & \sum_{i=1}^{2} \mathbf{M}_{33}^{(i)} + \mathbf{\Psi}_{i3}^T \mathbf{M}_{i3} + \mathbf{M}_{i3}^T \mathbf{\Psi}_{i3} + \mathbf{\Psi}_{i3}^T \mathbf{M}_{ii} \mathbf{\Psi}_{i3} \end{bmatrix}. \tag{2.43}$$

Then, the reduced eigenvalue problem is defined by

$$\bar{\mathbf{K}} \bar{\mathbf{x}} = \bar{\lambda} \bar{\mathbf{M}} \bar{\mathbf{x}} , \tag{2.44}$$

and the approximate eigensolutions are obtained by

$$\lambda \approx \bar{\lambda} , \quad \boldsymbol{\varphi} \approx \mathbf{T} \bar{\mathbf{x}} \tag{2.45}$$

with

$$\mathbf{T} = \begin{bmatrix} \mathbf{\Phi}_1 & & \mathbf{\Psi}_{13} \\ & \mathbf{\Phi}_2 & \mathbf{\Psi}_{23} \\ & & \mathbf{I} \end{bmatrix}. \tag{2.46}$$

2.2. AMLS method

In this section, we briefly review the formulation of the AMLS method. The detailed derivation and its implementation are well described in Refs. [5,6,59].

The mass and stiffness matrices in Eq. (2.1) are reordered and partitioned into substructures using the nested dissection algorithm [35]. The process is called substructuring and provides a hierarchy of substructures represented by a substructure tree. In the substructure tree, a substructure directly connected above (below) to the ith substructure is a parent (child) substructure of the *i*th substructure. From the parent-child relationship, an ancestor (descendant) substructure is defined by any substructure connected above (below) the substructure tree. As an example, a two-level substructuring of a generic FE model is given in Fig. 2.3. The set of the entire substructures is defined by $\Omega = \{1,2,3,4,5,6,7\}$, and the ith substructure ($\forall i \in \Omega$) has ancestor and descendant sets, A_i and D_i , respectively (e.g. $A_3 = \{7\}$ and $D_3 = \{1,2\}$). Note that, in our implementation, the sequence of the substructure numbers follows the sequence of a postorder traversal [60] of the substructure tree.

After substructuring, the stiffness matrix with n substructures is reordered as

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_{11} & & & & \\ & \ddots & & & \\ & & \mathbf{K}_{ii} & \mathbf{K}_{ij} & \\ & sym. & & \ddots & \\ & & & & \mathbf{K}_{nn} \end{bmatrix} \quad \text{for} \quad \forall i \in \Omega, \quad \forall j \in A_i \tag{2.47}$$

where the diagonal matrix $\mathbf{K}_{ii}$ denotes the stiffness matrix of the *i*th substructure, and the off-diagonal matrix $\mathbf{K}_{ij}$ denotes the stiffness matrix of the *i*th substructure coupled with the *j*th substructure. Note that the consistent mass matrix $\mathbf{M}$ is reordered in the same form as $\mathbf{K}$ in Eq. (2.47).

With a postorder traversal of the substructure tree, the AMLS method performs the following transformation for each substructure, which is composed of a block Gaussian elimination with projection onto substructure eigenspace as

$$\mathbf{M}^{(i)} = (\mathbf{T}^{(i)})^T \mathbf{M}^{(i-1)} \mathbf{T}^{(i)}, \quad \mathbf{K}^{(i)} = (\mathbf{T}^{(i)})^T \mathbf{K}^{(i-1)} \mathbf{T}^{(i)}$$

$$\text{with} \quad \mathbf{T}^{(i)} = \begin{bmatrix} \mathbf{I} & & & & \\ & \mathbf{V}_i & & \mathbf{\Psi}_{ij} & \\ & & \mathbf{I} & & \\ & & & \ddots & \\ & & & & \mathbf{I} \end{bmatrix} \quad \text{for} \quad i = 1,2,\cdots,n, \quad \forall j \in A_i, \tag{2.48}$$

where $\mathbf{M}^{(i)}$ and $\mathbf{K}^{(i)}$ denote the mass and stiffness matrices after the transformation of the *i*th substructure, respectively, $\mathbf{T}^{(i)}$ is the transformation matrix for the *i*th substructure, $\mathbf{I}$ is the identity matrix, $\mathbf{V}_i$ is the eigenvector matrix for the *i*th substructure, and $\mathbf{\Psi}_{ij}$ is the constraint mode matrix for the *i*th substructure coupled

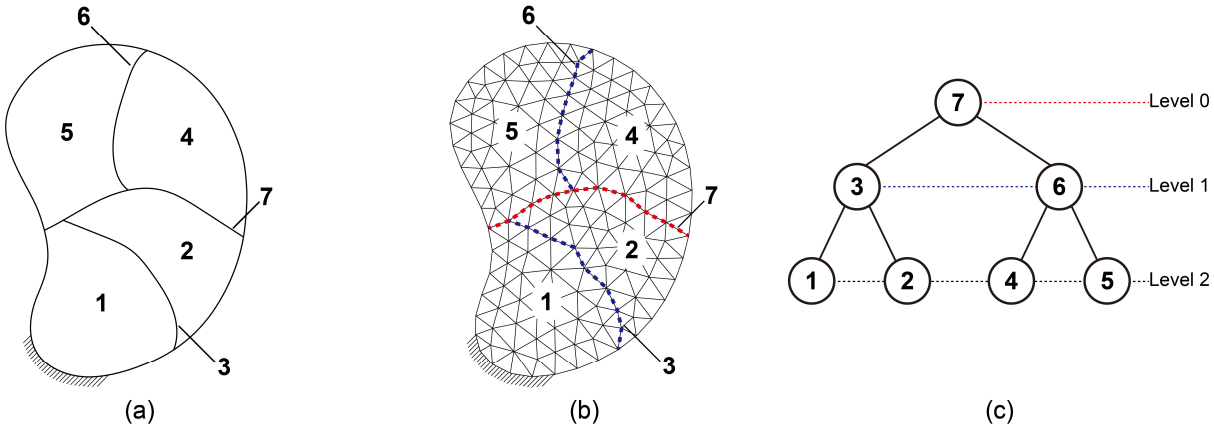


Fig. 2.3. Two-level substructuring of a generic finite element model: (a) partitioned structure, (b) partitioned finite element model, and (c) substructure tree.

with *j*th substructure. Since the sequence of the substructure numbers follows a postorder traversal sequence in our implementation, the substructure transformations are performed in ascending order; therefore, $\mathbf{M}^{(0)}$ and $\mathbf{K}^{(0)}$ represent the non-transformed $\mathbf{M}$ and $\mathbf{K}$, respectively.

Algorithm 2.1 [5,59] shows the transformation procedure of the *i*th substructure in detail. In line 2, the matrix $\mathbf{W}_i$ is a diagonal matrix composed of the eigenvalues, and the columns of the matrix $\mathbf{V}_i$ consist of the corresponding eigenvectors. Note that the substructure eigensolutions are truncated by using a given cutoff frequency ω_c. After solving the substructure eigenvalue problem and constructing the constraint mode matrices (lines 2-4), the updates of the ancestor mass and ancestor stiffness matrices (lines 5-9) and of descendant mass matrices (lines 10-13) are performed. In line 6, the set $\tilde{A}_i$ is defined as $\tilde{A}_i = (\{i\} \cup A_i)$.

After all of the substructures are transformed, the AMLS transformation matrix $\mathbf{T}$ is written as

$$\mathbf{T} = \prod_{i=1}^{n} \mathbf{T}^{(i)}, \tag{2.49}$$

and the reduced mass and stiffness matrices ($\bar{\mathbf{M}}$ and $\bar{\mathbf{K}}$) have the following forms

$$\bar{\mathbf{M}} = \mathbf{T}^T\mathbf{M}\mathbf{T} = \begin{bmatrix} \mathbf{I} & & & \\ & \ddots & & \\ & & \mathbf{I} & \bar{\mathbf{M}}_{ij} \\ & sym. & & \ddots \\ & & & & \mathbf{I} \end{bmatrix}, \quad \bar{\mathbf{K}} = \mathbf{T}^T\mathbf{K}\mathbf{T} = \begin{bmatrix} \mathbf{W}_1 & & & \\ & \ddots & & \\ & & \mathbf{W}_i & \\ & & & \ddots \\ & & & & \mathbf{W}_n \end{bmatrix}$$

for $\forall i \in \Omega$, $\forall j \in A_i$. (2.50)

Through the Rayleigh-Ritz procedure [1], the reduced eigenvalue problem is defined by

$$\bar{\mathbf{K}}\mathbf{q} = \bar{\lambda}\bar{\mathbf{M}}\mathbf{q}, \tag{2.51}$$

and the approximate solutions of Eq. (2.1) are obtained by

Algorithm 2.1. Transformation of a substructure

procedure TransformSubs (i)

input: substructure number i

1: factorize $\mathbf{K}_{ii}$

2: compute eigensolutions $\mathbf{W}_i$ and $\mathbf{V}_i$ such that $\mathbf{K}_{ii}\mathbf{V}_i = \mathbf{M}_{ii}\mathbf{V}_i\mathbf{W}_i$

3: **for** $j \in A_i$

4: $\quad \mathbf{\Psi}_{ij} \leftarrow -\mathbf{K}_{ii}^{-1}\mathbf{K}_{ij}$

5: **for** $j \in A_i$

6: $\quad$ **for** $k \in \tilde{A}_j$

7: $\quad\quad \mathbf{M}_{jk} \leftarrow \mathbf{M}_{jk} + \mathbf{\Psi}_{ij}^T\mathbf{M}_{ik} + \mathbf{M}_{ij}^T\mathbf{\Psi}_{ik} + \mathbf{\Psi}_{ij}^T\mathbf{M}_{ii}\mathbf{\Psi}_{ik}$

8: $\quad\quad \mathbf{K}_{jk} \leftarrow \mathbf{K}_{jk} + \mathbf{\Psi}_{ij}^T\mathbf{K}_{ik}$

9: $\quad \mathbf{M}_{ij} \leftarrow \mathbf{V}_i^T(\mathbf{M}_{ii}\mathbf{\Psi}_{ij} + \mathbf{M}_{ij})$

10: **for** $j \in D_i$

11: $\quad$ **for** $k \in A_i$

12: $\quad\quad \mathbf{M}_{jk} \leftarrow \mathbf{M}_{jk} + \mathbf{M}_{ji}\mathbf{\Psi}_{ik}$

13: $\quad \bar{\mathbf{M}}_{ji} \leftarrow \mathbf{M}_{ji}\mathbf{V}_i$

$$\lambda \approx \bar{\lambda}, \quad \boldsymbol{\varphi} \approx \bar{\boldsymbol{\varphi}} = \mathbf{T}\mathbf{q} \tag{2.52}$$

with

$$\bar{\boldsymbol{\varphi}} = [\bar{\boldsymbol{\varphi}}_1 \quad \bar{\boldsymbol{\varphi}}_2 \quad \cdots \quad \bar{\boldsymbol{\varphi}}_n]^T, \quad \mathbf{q} = [\mathbf{q}_1 \quad \mathbf{q}_2 \quad \cdots \quad \mathbf{q}_n]^T, \tag{2.53}$$

where $\bar{\boldsymbol{\varphi}}_i$ is the approximate eigenvector corresponding to the DOFs of the ith substructure, and $\mathbf{q}_i$ is the reduced eigenvector corresponding to the DOFs of the ith reduced substructure.

There are two ways to compute the approximate eigenvectors: the explicit [5] and implicit [47] strategies. In the explicit strategy, the AMLS transformation matrix is obtained in advance as follows:

$$\mathbf{T} = \begin{bmatrix} \mathbf{V}_1 & & & & \\ & \ddots & \mathbf{T}_{ji} & & \\ & & \mathbf{V}_i & & \\ & & & \ddots & \\ & & & & \mathbf{V}_n \end{bmatrix} \text{ with } \mathbf{T}_{ji} = \mathbf{\Psi}_{ji}\mathbf{V}_i + \sum\nolimits_k \mathbf{\Psi}_{jk}\mathbf{T}_{ki}$$

$$\text{for } i = 1, 2, \cdots, n, \quad \forall j \in D_i, \quad \forall k \in (D_i \cap A_j), \tag{2.54}$$

where the subscript j in the computation of $\mathbf{T}_{ji}$ is in descending order.

Then the back transformation in Eq. (2.52) is performed by

$$\bar{\boldsymbol{\varphi}}_i = \mathbf{V}_i\mathbf{q}_i + \sum\nolimits_j \mathbf{T}_{ij}\mathbf{q}_j \quad \text{for} \quad \forall i \in \Omega, \quad \forall j \in A_i. \tag{2.55}$$

Note that the cost of computing $\mathbf{T}$ in Eq. (2.54) is relatively large [5,47].

In order to reduce the computational cost, the implicit way is commonly adopted as follows:

$$\bar{\boldsymbol{\varphi}} = \mathbf{T}\mathbf{q} = \mathbf{T}^{(1)}\mathbf{T}^{(2)}\cdots\mathbf{T}^{(n)}\mathbf{q}\,, \tag{2.56}$$

where the order of the matrix multiplications is from right to left, and the submatrix component $\bar{\boldsymbol{\varphi}}_i$ is calculated by

$$\bar{\boldsymbol{\varphi}}_i = \mathbf{V}_i\mathbf{q}_i + \sum\nolimits_j \boldsymbol{\Psi}_{ij}\bar{\boldsymbol{\varphi}}_j \quad \text{for} \quad i = n, n-1, \cdots, 1\,, \quad \forall j \in A_i\,. \tag{2.57}$$

Ref. [47] reports that the implicit back transformation is much more efficient than the explicit regarding computation time and required memory. Thus, the implicit back transformation is set as the default option in our implementation and is used to investigate the parallel performance explained in Sections 2.4 and 2.5.

2.3. Load balancing algorithm

In this section, we propose an algorithm consisting of two types of granularity. Employing coarse-grained parallelism, both the transformation and back transformation procedures are parallelized according to the level of the substructures. Fine-grained parallelism is exploited for reducing bottlenecks in the transformation of each substructure. In the parallel regions, tasks are enqueued into the task queue. Then, on a first in, first out basis [60], each task is dequeued and executed on an available thread in the team of threads. Fig. 2.4 illustrates the serial and parallel computation strategies in our implementation.

First, an FE model is partitioned into disjoint subdomains. The disjoint subdomains are then further partitioned into substructures. In other words, a disjoint subdomain is a subtree composed of substructures. Hereinafter, the disjoint subdomains and their substructures are referred to as the substructure clusters and distributed substructures, respectively. We define interfaces between the substructure clusters as shared substructures. As an example, Fig. 2.5 shows a substructure tree consisting of the shared substructures and substructure clusters, including the distributed substructures.

In parallelism for the AMLS method, a determinacy race [60] occurs when a thread updates a substructure while another thread is accessing the substructure. For example, a substructure having descendant substructures could be concurrently processed by more than one thread (see lines 7 and 8 in Algorithm 2.1). To avoid the determinacy race, threads individually store the intermediate results of substructures to be processed concurrently. We confine such substructures to shared substructures.

The transformation is formulated in two parts corresponding to the distributed and shared substructures. Algorithm 2.2 gives the transformation procedure of distributed substructures, in which the set $\tilde{D}_i$ is defined by $\tilde{D}_i = (D_i \cup \{i\})$. To store intermediate results of the shared substructures, we initialize the local system matrices as follows:

$$\hat{\mathbf{M}}_{ij}^{(k)} = \hat{\mathbf{K}}_{ij}^{(k)} = \mathbf{0} \quad \text{for} \quad \forall i \in \Omega_s, \quad \forall j \in \tilde{A}_i, \quad \forall k \in D_i, \tag{2.58}$$

where Ω_s is the set of all shared substructures.

Unlike Algorithm 2.1, the updates of the ancestor mass and ancestor stiffness matrices are performed depending on whether the ancestor is a shared substructure or not (see lines 8-13 in Algorithm 2.2). In addition, if the transformation matrix $\mathbf{T}$ is explicitly needed, Eq. (2.54) is calculated in the transformation procedure (lines 19 and 20 in Algorithm 2.2). To calculate line 20, please refer to Algorithm 2.3. The transformation of the shared substructures will be discussed in the following section.

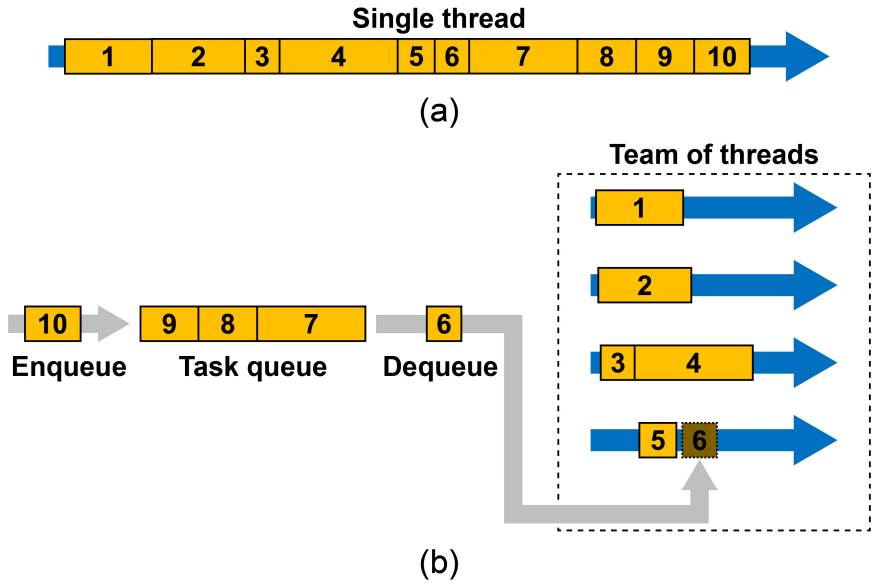


Fig. 2.4. Illustration of computation strategies: (a) serial computation and (b) parallel computation with a task queue.

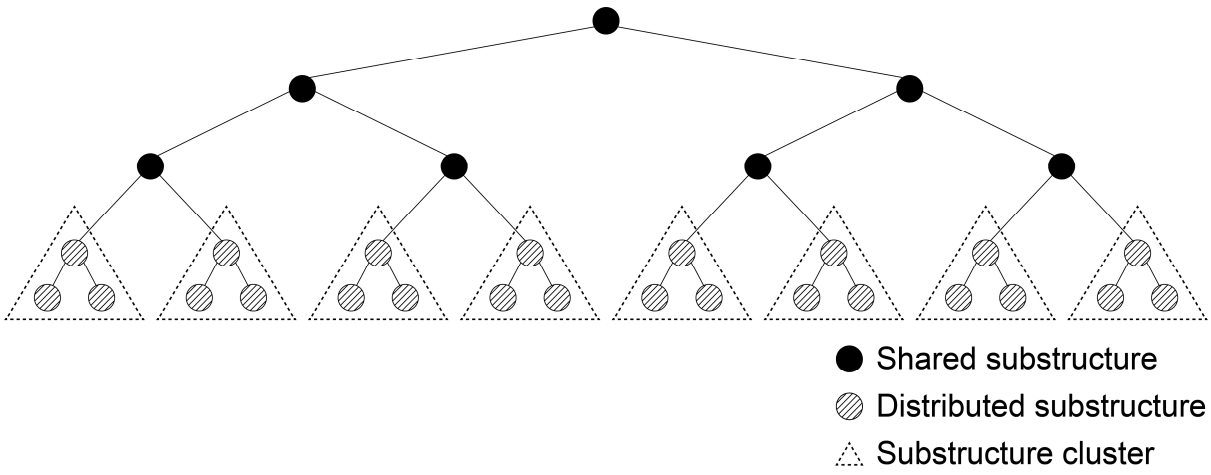


Fig. 2.5. Substructure tree consisting of shared and distributed substructures and substructure clusters.

Algorithm 2.2. Transformation of distributed substructures

procedure TransformDistSubs (p)

input: substructure number p

1: **for** $i \in \tilde{D}_p$ in ascending order

2: factorize $\mathbf{K}_{ii}$

3: compute eigensolutions $\mathbf{W}_i$ and $\mathbf{V}_i$ such that $\mathbf{K}_{ii}\mathbf{V}_i = \mathbf{M}_{ii}\mathbf{V}_i\mathbf{W}_i$

4: **for** $j \in A_i$

5: $\boldsymbol{\Psi}_{ij} \leftarrow -\mathbf{K}_{ii}^{-1}\mathbf{K}_{ij}$

6: **for** $j \in A_i$

7: **for** $k \in \tilde{A}_j$

8: **if** $j \in \Omega_s$ **then**

9: $\hat{\mathbf{M}}_{jk}^{(p)} \leftarrow \hat{\mathbf{M}}_{jk}^{(p)} + \boldsymbol{\Psi}_{ij}^T\mathbf{M}_{ik} + \mathbf{M}_{ij}^T\boldsymbol{\Psi}_{ik} + \boldsymbol{\Psi}_{ij}^T\mathbf{M}_{ii}\boldsymbol{\Psi}_{ik}$

10: $\hat{\mathbf{K}}_{jk}^{(p)} \leftarrow \hat{\mathbf{K}}_{jk}^{(p)} + \boldsymbol{\Psi}_{ij}^T\mathbf{K}_{ik}$

11: **else**

12: $\mathbf{M}_{jk} \leftarrow \mathbf{M}_{jk} + \boldsymbol{\Psi}_{ij}^T\mathbf{M}_{ik} + \mathbf{M}_{ij}^T\boldsymbol{\Psi}_{ik} + \boldsymbol{\Psi}_{ij}^T\mathbf{M}_{ii}\boldsymbol{\Psi}_{ik}$

13: $\mathbf{K}_{jk} \leftarrow \mathbf{K}_{jk} + \boldsymbol{\Psi}_{ij}^T\mathbf{K}_{ik}$

14: $\mathbf{M}_{ij} \leftarrow \mathbf{V}_i^T(\mathbf{M}_{ii}\boldsymbol{\Psi}_{ij} + \mathbf{M}_{ij})$

15: **for** $j \in D_i$

16: **for** $k \in A_i$

17: $\mathbf{M}_{jk} \leftarrow \mathbf{M}_{jk} + \mathbf{M}_{ji}\boldsymbol{\Psi}_{ik}$

18: $\bar{\mathbf{M}}_{ji} \leftarrow \mathbf{M}_{ji}\mathbf{V}_i$

19: **if** the transformation matrix $\mathbf{T}$ is explicitly needed **then**

20: ComputeTransformMat (i) Algorithm 2.3

Algorithm 2.3. Construct the AMLS transformation matrix

procedure ComputeTransformMat (i)

input: substructure number i

1: **for** $j \in D_i$ in descending order

2: $\mathbf{T}_{ji} \leftarrow \boldsymbol{\Psi}_{ji}\mathbf{V}_i + \sum_k \boldsymbol{\Psi}_{jk}\mathbf{T}_{ki}$, $\forall k \in (D_i \cap A_j)$

2.3.1. Fine-grained parallelism

In the PAMLS method, more threads become idle as more substructures are transformed. For example, the final substructure is processed by a single thread because all other substructures are already transformed [5]. Furthermore, the transformation of a shared substructure is likely to be more expensive than the transformation of a distributed substructure. Therefore, fine-grained parallelism for the transformation of shared substructures is needed for optimal performance.

The transformation procedure consists of three parts: solving the eigenvalue problem, constructing the constraint mode matrices, and block Gaussian eliminations with projection onto the substructure eigenspace. In addition, the block Gaussian eliminations with the projection can be separated into several independent tasks: the updates of the ancestor mass, ancestor stiffness, and descendant mass matrices. Note that the updates can be performed with more fine-grained parallelism. The transformation procedure consists of three parts: solving the eigenvalue problem, constructing the constraint mode matrices, and block Gaussian eliminations with projection onto the substructure eigenspace. In addition, the block Gaussian eliminations with the projection can be separated into several independent tasks: the updates of the ancestor mass, ancestor stiffness, and descendant mass matrices. Note that the updates can be performed with more fine-grained parallelism.

Fig. 2.6(a) shows the serial transformation of a substructure in the AMLS method. Fig. 2.6(b) illustrates the fine-grained parallelism for the transformation of a shared substructure, in which separable tasks are executed by the team of threads. In fine-grained parallelism, we can solve the substructure eigensolutions and construct the constraint mode matrices simultaneously after factorizing the substructure stiffness matrix. Then, the rest of the transformation is performed in parallel. If the transformation matrix $\mathbf{T}$ is explicitly needed, Eq. (2.54) is computed simultaneously during the updates of the mass and stiffness matrices.

The detailed procedure of the fine-grained parallelism for the transformation is given in Algorithm 2.4. The task is created and enqueued by the statement **spawn** [60] and executed on the team of threads. The statement **sync** [60] indicates a wait for the completion of all direct child tasks created by **spawn**. As shown in Fig. 2.6(b), solving the eigenvalue problem and constructing the constraint mode matrices are performed simultaneously after the factorization of the substructure stiffness matrix (see lines 1-4 in Algorithm 2.4). Then, the synchronization of the eigensolutions and constraint modes occurs on line 5. The updates of mass and stiffness matrices and the computation of the transformation matrix are performed in parallel (lines 6-15).

Note that the update of ancestor mass matrices is the most time-consuming task in each transformation of a shared substructure (line 6 in Algorithm 2.4). Therefore, this computation takes advantage of the more fine-grained parallelism, which is detailed in Algorithm 2.5.

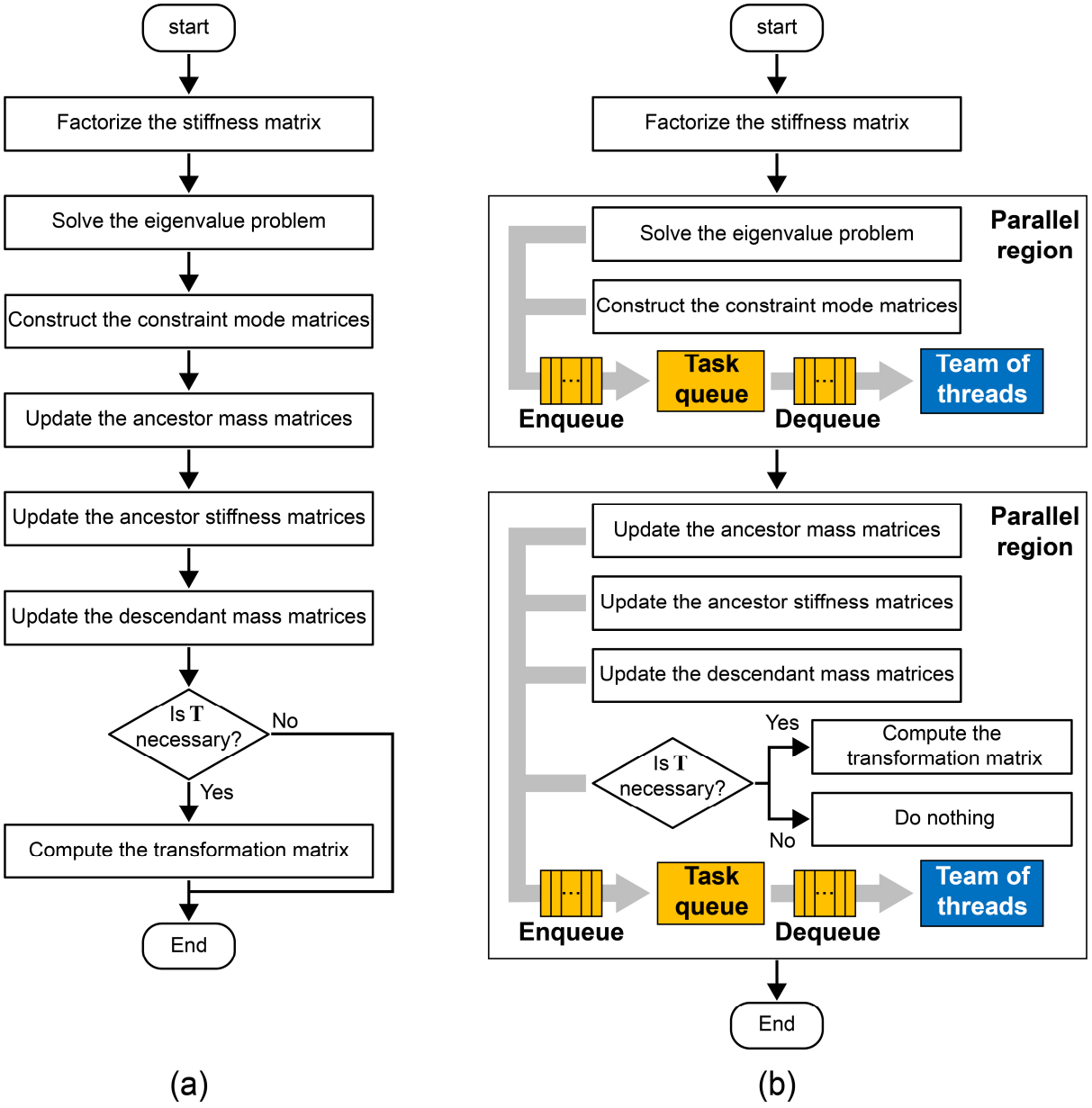


Fig. 2.6. Substructure transformation: (a) serial algorithm and (b) fine-grained parallel algorithm.

Algorithm 2.4. Transformation of a shared substructure

procedure TransformSharedSubs (i)

input: substructure number i

1: factorize $\mathbf{K}_{ii}$

2: **spawn** compute eigensolutions $\mathbf{W}_i$ and $\mathbf{V}_i$ such that $\mathbf{K}_{ii}\mathbf{V}_i = \mathbf{M}_{ii}\mathbf{V}_i\mathbf{W}_i$

3: **for** $j \in A_i$

4: **spawn** $\mathbf{\Psi}_{ij} \leftarrow -\mathbf{K}_{ii}^{-1}\mathbf{K}_{ij}$

5: **sync**

6: **spawn** update ancestor mass matrices: UpdateAncMass (i) Algorithm 2.5

7: **for** $j \in A_i$

8: **for** $k \in \tilde{A}_j$

9: **spawn** $\hat{\mathbf{K}}_{jk}^{(i)} \leftarrow \hat{\mathbf{K}}_{jk}^{(i)} + \mathbf{\Psi}_{ij}^T\mathbf{K}_{ik}$

10: **for** $j \in D_i$

11: **for** $k \in A_i$

12: **spawn** $\mathbf{M}_{jk} \leftarrow \mathbf{M}_{jk} + \mathbf{M}_{ji}\mathbf{\Psi}_{ik}$

13: **spawn** $\bar{\mathbf{M}}_{ji} \leftarrow \mathbf{M}_{ji}\mathbf{V}_i$

14: **if** the transformation matrix $\mathbf{T}$ is explicitly needed **then**

15: **spawn** ComputeTransformMat (i) Algorithm 2.3

16: **sync**

Algorithm 2.5. Update of ancestor mass matrices for a shared substructure

procedure UpdateAncMass (i)

input: substructure number i

1: **for** $j \in A_i$

2: **for** $k \in \tilde{A}_j$

3: **spawn** $\mathbf{A}_{jk} \leftarrow \mathbf{M}_{ij}^T\mathbf{\Psi}_{ik}$

4: **spawn** $\mathbf{B}_{jk} \leftarrow \mathbf{\Psi}_{ij}^T\mathbf{M}_{ik}$

5: **spawn** $\mathbf{C}_{ij} \leftarrow \mathbf{M}_{ii}\mathbf{\Psi}_{ij}$

6: **sync**

7: **for** $j \in A_i$

8: **for** $k \in \tilde{A}_j$

9: **spawn** $\hat{\mathbf{M}}_{jk}^{(i)} \leftarrow \hat{\mathbf{M}}_{jk}^{(i)} + \mathbf{A}_{jk} + \mathbf{B}_{jk} + \mathbf{\Psi}_{ij}^T\mathbf{C}_{ik}$

10: **spawn** $\mathbf{M}_{ij} \leftarrow \mathbf{V}_i^T(\mathbf{C}_{ij} + \mathbf{M}_{ij})$

2.3.2. Coarse-grained parallelism

In coarse-grained parallelism, each thread transforms the assigned distributed substructures and shared substructure based on the postorder traversal. Fig. 2.7 shows a scheduling example of coarse-grained parallelism for the transformation of substructures, including fine-grained parallelism. Note that each shared substructure can be transformed by at least one thread in consequence of the fine-grained parallelism, while each substructure cluster is transformed by a single thread.

Algorithm 2.6 presents a detailed procedure with recursion. The statement **parallel for** [60] indicates that a loop runs in parallel. Note that, at the end of the loop, there is an implicit barrier that forces threads in the **parallel for** region to wait until all threads encounter the barrier. On line 8 in Algorithm 2.6, **sync** denotes the synchronization point between threads associated with the left and right children of a shared substructure. After this, the synchronization of the mass and stiffness matrices for the shared substructure occurs (lines 9-15), where C_i is the set of direct child substructures for the ith substructure. However, the synchronization could give rise to bottlenecks because of the transformation time imbalance among substructures. To reduce these bottlenecks, we adopt the approach of Escaig et al. [46], which was proposed to improve the parallel efficiency of the static condensation method [54,55]. The number of assigned tasks (i.e. substructures) is set to be larger than the number of threads. This balances the workload among threads, although the size of the interface grows.

Instead of using the number of threads, a given cutoff level is used to determine the number of substructure clusters. In other words, the level of distributed substructures is larger than the cutoff level, and the rest of the substructures are defined as shared substructures, as seen in Fig. 2.7. In our implementation, the following cutoff level L_c has been found to be effective:

$$L_c = \min\{L_t + 5, L_s\} \quad \text{with} \quad L_t = \min\{k \in \mathbb{Z} : \log_2 N_t \le k\}, \tag{2.59}$$

where N_t and L_s are the number of threads used and the maximum level of substructures, respectively, and $\mathbb{Z}$ is the set of all integers. The effect of the cutoff level on the performance is discussed further in Appendix A.

Finally, the whole procedure of the proposed algorithm is given in Algorithm 2.7. After the transformation of the entire substructures is completed, the reduced solutions are obtained in line 3. Then, the approximate eigenvectors are computed in parallel using the implicit (line 8) or explicit (lines 5 and 6) strategies.

For the parallelization of the implicit back transformation, the detailed procedure based on the preorder traversal [60] is given in Algorithm 2.8. Similar to the coarse-grained parallelism for the transformation procedure, parallel tasks are created depending on whether the processed substructure is a shared substructure or not (lines 1 and 4 in Algorithm 2.8). Then, the approximate responses corresponding to the substructures are computed on lines 3 and 5 as Eq. (2.57). Note that there is no explicit synchronization point in the parallel implicit back transformation procedure.

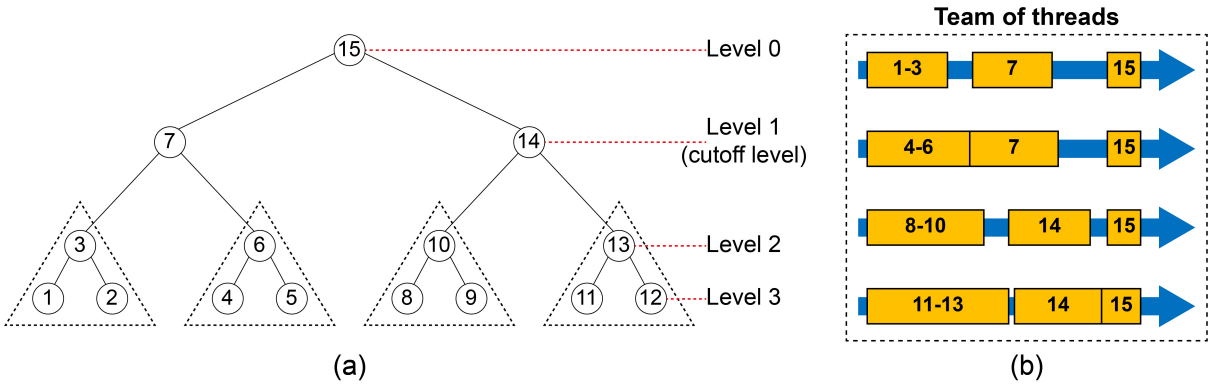


Fig. 2.7. Scheduling example for parallelization of the transformation of substructures: (a) substructure tree with 15 substructures and (b) scheduling with 4 threads, where the number indicates the substructure number.

Algorithm 2.6. Parallel transformation

---

procedure Postorder (i)

input: substructure number i

1: **if** $i \notin \Omega_s$ **then**

2: TransformDistSubs(i) Algorithm 2.2

3: **else**

4: $l \leftarrow$ left child of the ith substructure

5: $r \leftarrow$ right child of the ith substructure

6: **spawn** Postorder (l)

7: **spawn** Postorder (r)

8: **sync**

9: **parallel for** $j \in \tilde{A}_i$

10: $\mathbf{M}_{ij} \leftarrow \mathbf{M}_{ij} + \sum_p \hat{\mathbf{M}}_{ij}^{(p)}, \ \forall p \in C_i$

11: $\mathbf{K}_{ij} \leftarrow \mathbf{K}_{ij} + \sum_p \hat{\mathbf{K}}_{ij}^{(p)}, \ \forall p \in C_i$

12: **if** $j \neq i$ **then**

13: **parallel for** $k \in \tilde{A}_j$

14: $\hat{\mathbf{M}}_{jk}^{(i)} \leftarrow \sum_p \hat{\mathbf{M}}_{jk}^{(p)}, \ \forall p \in C_i$

15: $\hat{\mathbf{K}}_{jk}^{(i)} \leftarrow \sum_p \hat{\mathbf{K}}_{jk}^{(p)}, \ \forall p \in C_i$

16: TransformSharedSubs(i) Algorithm 2.4

Algorithm 2.7. Proposed parallel AMLS method

procedure PAMLS

1: partition the finite element model into n substructures as Eq. (2.47)

2: Postorder (n) Algorithm 2.6

3: solve the reduced eigenvalue problem $\bar{\mathbf{K}}\mathbf{q} = \bar{\lambda}\bar{\mathbf{M}}\mathbf{q}$

4: **if** there exists the pre-computed $\mathbf{T}$ **then**

5: **parallel for** $i \in \Omega$

6: $\bar{\mathbf{\Phi}}_i \leftarrow \mathbf{V}_i\mathbf{q}_i + \sum_j \mathbf{T}_{ij}\mathbf{q}_j$

7: **else**

8: Preorder (n) Algorithm 2.8

Algorithm 2.8. Parallel algorithm for the implicit back transformation

procedure Preorder (p)

input: substructure number p

1: **if** $p \notin \Omega_s$ **then**

2: **for** $i \in \tilde{D}_p$ in descending order $\forall j \in A_i$

3: $\bar{\boldsymbol{\varphi}}_i \leftarrow \mathbf{V}_i\mathbf{q}_i + \sum_j \mathbf{\Psi}_{ij}\bar{\boldsymbol{\varphi}}_j$, $\forall j \in A_i$

4: **else**

5: $\bar{\boldsymbol{\varphi}}_p \leftarrow \mathbf{V}_p\mathbf{q}_p + \sum_q \mathbf{\Psi}_{pq}\bar{\boldsymbol{\varphi}}_q$, $\forall q \in A_p$

6: $l \leftarrow$ left child of the pth substructure

7: $r \leftarrow$ right child of the pth substructure

8: **spawn** Preorder (l)

9: **spawn** Preorder (r)

If the explicit strategy is used, the AMLS transformation matrix is calculated during the transformation of substructures (lines 19 and 20 in Algorithm 2.2 and lines 14 and 15 in Algorithm 2.4). Then, approximate eigenvectors are computed in parallel, as seen in lines 5 and 6 in Algorithm 2.7 [5]. Here, there is no dependency between the computations of $\bar{\boldsymbol{\varphi}}_i$ and $\bar{\boldsymbol{\varphi}}_j$ for $i \neq j$.

2.4. Numerical examples

To investigate the performance, the generalized eigenvalue problems for three structural FE models are solved on 2, 4, 8, 16, and 32 threads: a rectangular plate, an automotive wheel, and an airplane. Table 2.1 lists the numbers of DOFs and nonzero entries for the FE models. The structures are modeled by shell finite elements [1,61–66] and/or tetrahedral solid finite elements [1,61,67]. In each FE model, we set the cutoff frequency $\omega_c = 8.4\omega_h$ [5] as the default option for every substructure, where ω_h is the highest excitation frequency.

The performance is evaluated by wall clock time required for the transformation and back transformation procedures, which spend a majority of the elapsed time in the AMLS method. To investigate the parallel efficiency of the transformation procedure, the proposed algorithm is compared to the original parallel AMLS (PAMLS) method proposed by Elssel and Voss [8]. The original PAMLS method is implemented in such a way that only coarse-grained parallelism is applied without fine-grained parallelism, where the number of substructure clusters is set to be the number of threads. Without the fine-grained parallelism, Algorithms 2.4 and 2.5 are serialized by deleting the parallel statements **spawn** and **sync**.

We also demonstrate the parallel performance of the implicit back transformation procedure. To the best of our knowledge, the parallelization of the implicit back transformation has not yet been reported. Hence, for comparison with the original PAMLS method, we implemented the parallel implicit back transformation procedure based on the parallelization concept of the transformation procedure. Only coarse-grained parallelism is applied without fine-grained parallelism, where the number of substructure clusters is set to be the number of threads.

The serial and parallel algorithms in this chapter are implemented in Fortran, where Intel Fortran Compiler 19.0.4 with OpenMP is used. The mesh partitioning is achieved by METIS [68], an open-source package for unstructured graph partitioning, and the eigenvalue problems are solved using ARPACK [53], an open-source package based on the Arnoldi/Lanczos process. All numerical examples are tested on CentOS 7.4 with two 20-core Intel Xeon Gold 6148 processors (2.4 GHz) and with 192 GB of memory.

Table 2.1. Finite element models for numerical examples.

Example	DOFs	Number of nonzero entries in the upper triangular part	
		Mass matrix	Stiffness matrix
Rectangular plate	196614	976170	2094701
	786438	3918378	8442289
	3145734	15701034	33885811
Automotive wheel	3688653	31712934	91545737
Airplane	8462700	76458759	220916587

2.4.1. Rectangular plate

We consider a simply supported rectangular plate, as shown in Fig. 2.8. The length L, width W, and thickness t of the plate are 2 m, 1 m, and 5 mm, respectively. The plate is modeled using four-node shell elements with three different meshes of $2N \times N$, where $2N$ elements are assigned along the plate length. The FE models of $N = 128$, 256, and 512 are partitioned into 511 substructures on 8 levels, 2047 substructures on 10 levels, and 8191 substructures on 12 levels, respectively. The reduced models with 1999, 3406, and 14706 DOFs are obtained from the original FE models with 196614, 786438, and 3145734 DOFs, respectively. The number of eigensolutions sought in this example is 150.

Fig. 2.9 presents the speed-up factors for the three FE models. The normalized wall clock times required for the 1024×512 mesh (3145734 DOFs) are shown in Fig. 2.10, and their details are listed in Table 2.2. The results show that the proposed algorithm significantly improves the parallel efficiency of the original PAMLS method. The speed-up factor of the proposed algorithm is almost three times that of the original PAMLS method when adopting 32 threads.

In the PAMLS method, the parallelism for the implicit back transformation procedure performs well compared to that for the transformation procedure. This is because the parallel implicit back transformation procedure has no explicit synchronization point; thus, the number of substructure clusters has less effect on the parallel performance. Nonetheless, the parallel performance of the implicit back transformation procedure is also enhanced through the proposed algorithm. Furthermore, as shown in Fig. 2.9 and Table 2.2, it is observed that the computation time for the transformation procedure accounts for most of the elapsed time provided that the parallelism for the implicit back transformation performs well.

To demonstrate that the proposed algorithm gives the same accuracy as the serial AMLS method, the relative eigenvalue error and relative residual [59] are respectively measured as

$$e_v = \frac{\left|\bar{\lambda} - \lambda\right|}{\lambda}, \tag{2.60}$$

$$e_r = \frac{\left\|\mathbf{K}\bar{\boldsymbol{\varphi}} - \bar{\lambda}\mathbf{M}\bar{\boldsymbol{\varphi}}\right\|_2}{\bar{\lambda}}, \tag{2.61}$$

where the overbar $(\bar{\cdot})$ denotes the approximate quantities obtained by the reduced models.

The solution accuracy of the serial and proposed algorithms is shown in Fig. 2.11, where, for brevity, only the results for 32 threads are considered. The same solution accuracy is obtained from the serial and proposed algorithms regardless of the number of threads used.

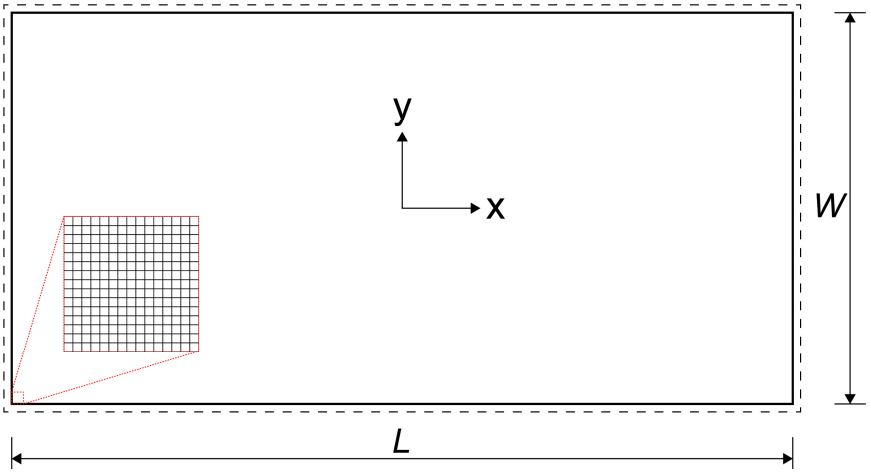


Fig. 2.8. Simply supported rectangular plate.

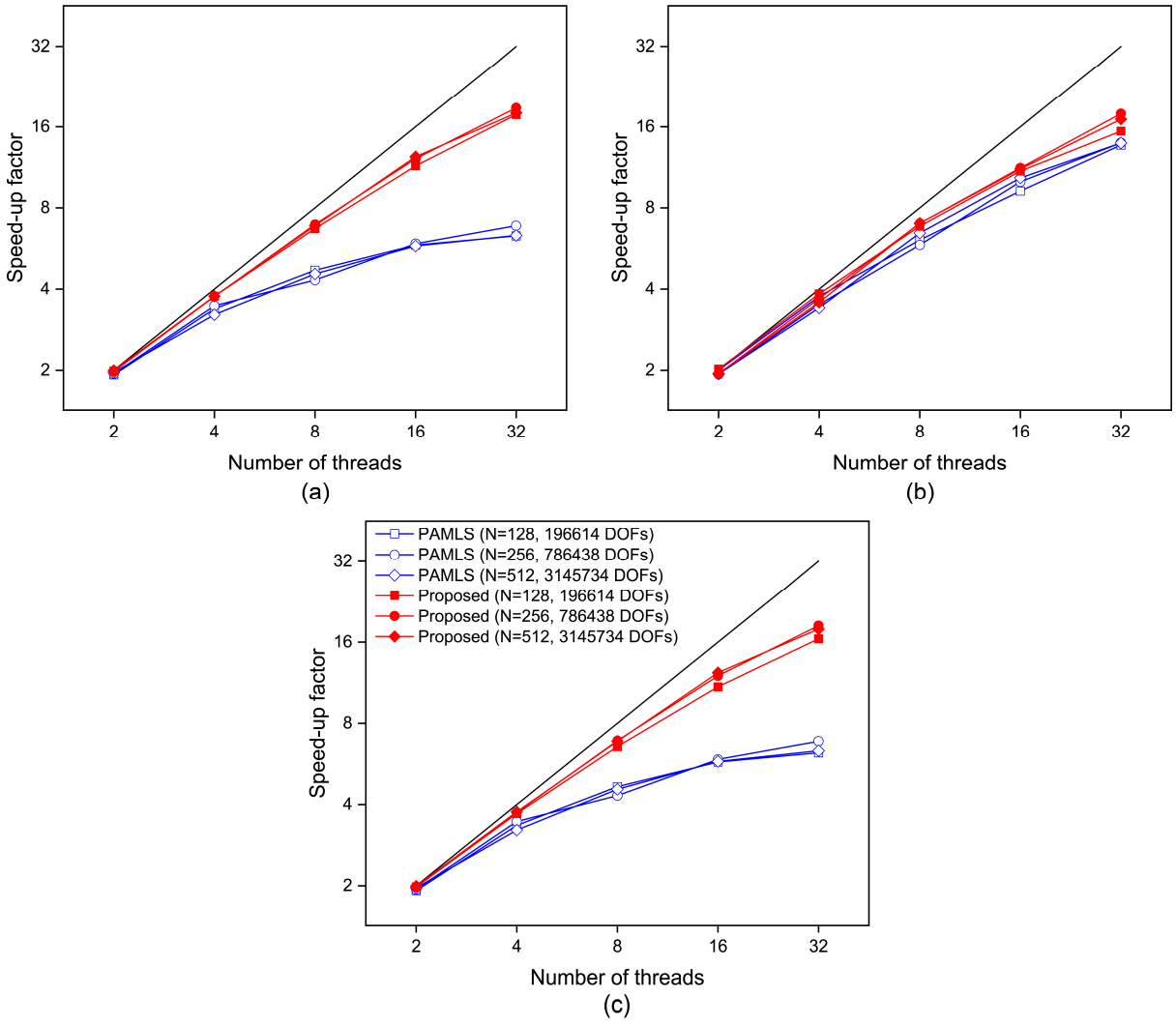


Fig. 2.9. Speed-up factors for the rectangular plate problem: (a) transformation, (b) implicit back transformation, and (c) whole procedures.

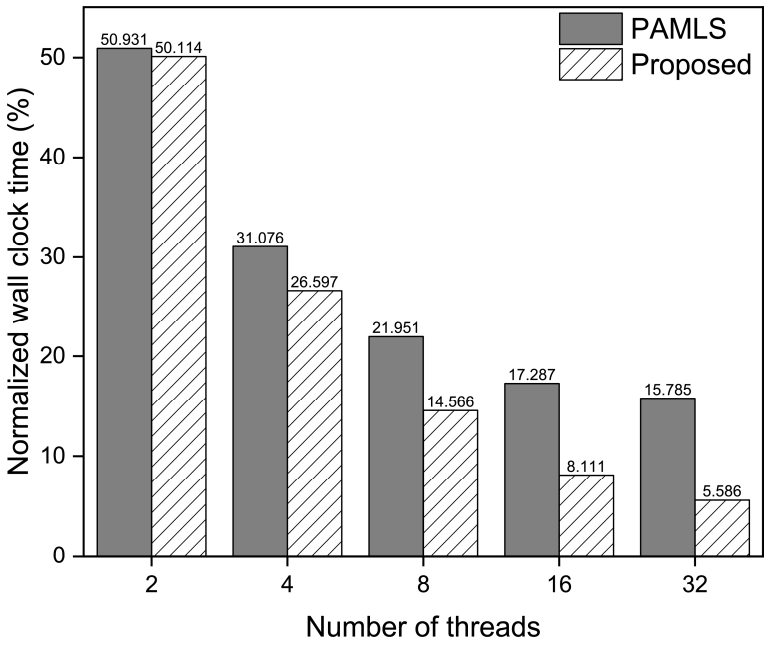


Fig. 2.10. Normalized wall clock times for the rectangular plate model of the 1024×512 mesh. The wall clock times are normalized by the total computation time required for the serial AMLS method.

Table 2.2. Normalized wall clock times for the rectangular plate model of the 1024×512 mesh, where N_t denotes the number of threads used and 'total' denotes the time elapsed for the whole procedure of the original PAMLS and proposed methods. The wall clock times are normalized by the total computation time required for the serial AMLS method.

N_t	PAMLS (%)			Proposed (%)		
	Total	Transformation	Back transformation	Total	Transformation	Back transformation
2	50.931	50.595	0.283	50.114	49.779	0.281
4	31.076	30.881	0.161	26.597	26.411	0.153
8	21.951	21.834	0.085	14.566	14.455	0.078
16	17.287	17.200	0.053	8.111	8.030	0.049
32	15.785	15.714	0.039	5.586	5.523	0.032

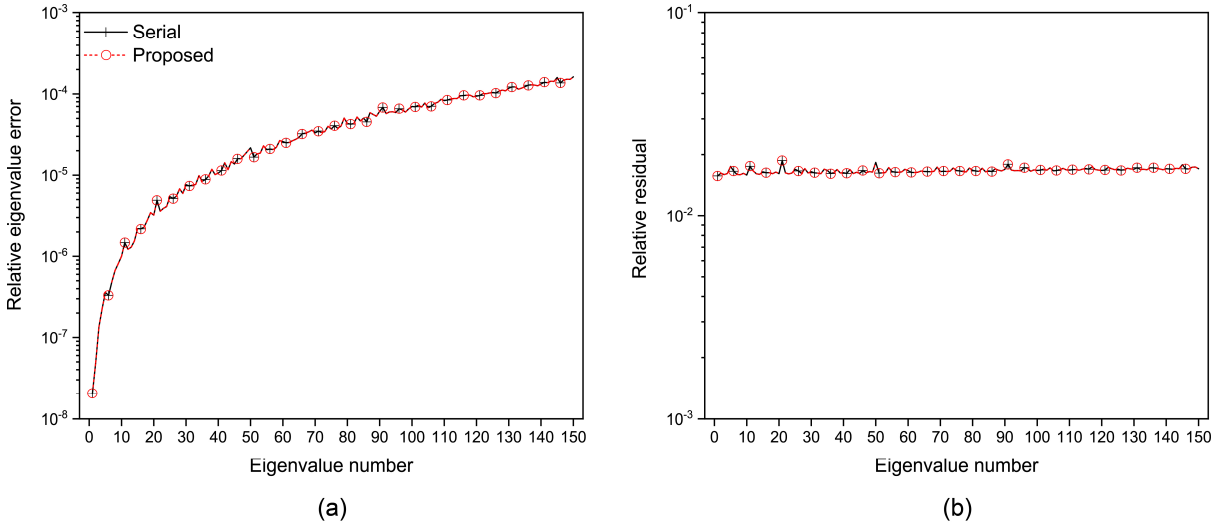


Fig. 2.11. Accuracy of the serial and proposed algorithms for the rectangular plate problem ($N = 512$, 3145734 DOFs): (a) relative eigenvalue errors and (b) relative residuals. Markers are placed at 5-eigenvalue intervals.

2.4.2. Automotive wheel

In this section, the automotive wheel described in Fig. 2.12 is considered. The wheel is modeled by 765348 three- and four-node shell elements and 266898 four-node tetrahedral elements (3688653 DOFs). Two different partitions are considered: partition A (2047 substructures on 10 levels) and partition B (15747 substructures on 13 levels). The reduced models with the 6696 and 20096 DOFs are obtained from partitions A and B, respectively, and the number of eigensolutions sought in this example is 300.

Fig. 2.13 shows the speed-up factors for the transformation, implicit back transformation, and whole procedures. The normalized wall clock times for partition B and their details are shown in Fig. 2.14 and Table 2.3, respectively. For the transformation procedure, the proposed algorithm is almost three times faster than the original PAMLS method when 32 threads are used. The parallelism for the implicit back transformation procedure also performs well regardless of the number of substructures.

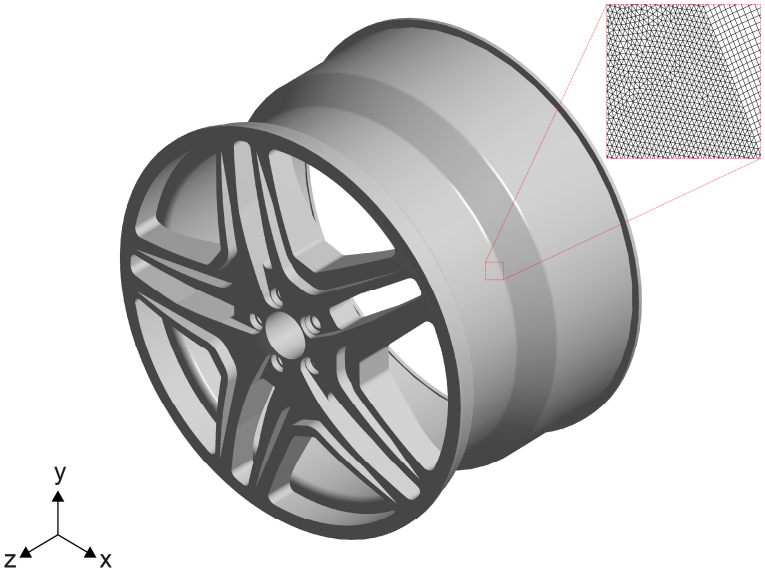


Fig. 2.12. Automotive wheel.

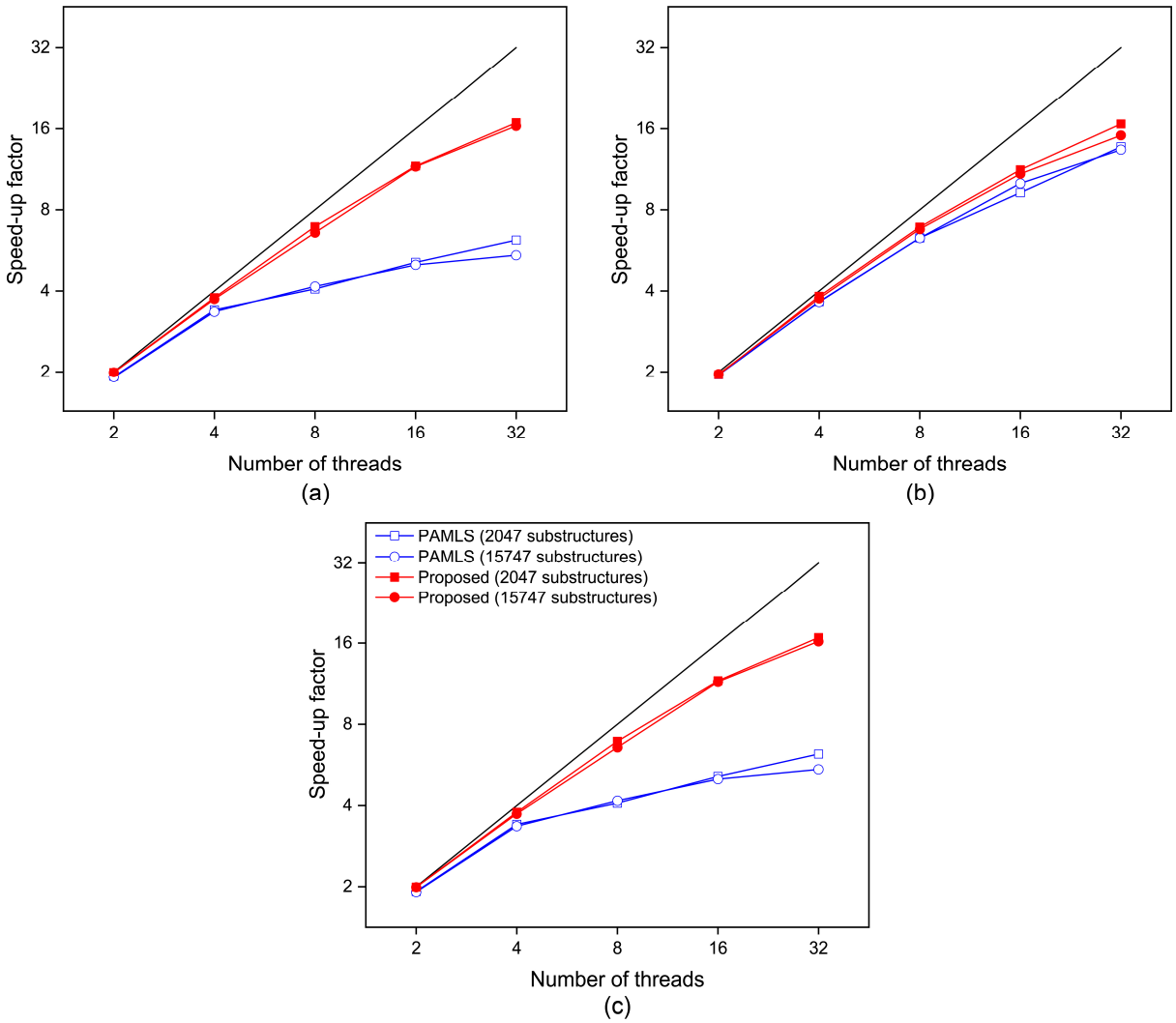


Fig. 2.13. Speed-up factors for the automotive wheel problem: (a) transformation and (b) implicit back transformation, and (c) whole procedures.

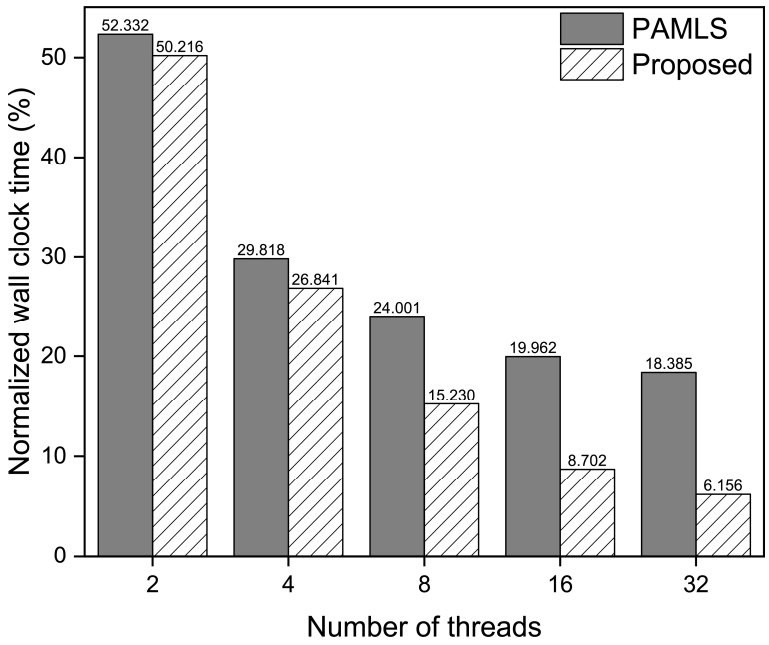


Fig. 2.14. Normalized wall clock times for the automotive wheel model with 15747 substructures. The wall clock times are normalized by the total computation time required for the serial AMLS method.

Table 2.3. Normalized wall clock times for the automotive wheel model with 15747 substructures, where N_t denotes the number of threads used and 'total' denotes the time elapsed for the whole procedure of the original PAMLS and proposed methods. The wall clock times are normalized by the total computation time required for the serial AMLS method.

N_t	PAMLS (%)			Proposed (%)		
	Total	Transformation	Back transformation	Total	Transformation	Back transformation
2	52.332	51.880	0.398	50.216	49.764	0.397
4	29.818	29.550	0.214	26.841	26.579	0.208
8	24.001	23.823	0.124	15.230	15.056	0.115
16	19.962	19.830	0.078	8.702	8.577	0.072
32	18.385	18.258	0.058	6.156	6.051	0.052

2.4.3. Airplane

Here, we consider an airplane structure as shown in Fig. 2.15. For modeling the structure, 1731466 three- and four-node shell elements are used, and the number of DOFs is 8462700. The FE model is partitioned into 32767 substructures on 14 levels. Two cutoff frequencies $\omega_c = 4.8\omega_h$ and $\omega_c = 8.4\omega_h$ are considered for finding the 600 smallest eigensolutions. The reduced models with 38489 and 45449 DOFs are obtained from the cutoff frequencies $\omega_c = 4.8\omega_h$ and $\omega_c = 8.4\omega_h$, respectively.

Fig. 2.16 shows the speed-up factors for the transformation, implicit back transformation, and whole procedures with respect to the cutoff frequencies. Fig. 2.17 and Table 2.4 show the normalized wall clock times for $\omega_c = 8.4\omega_h$ and their details, respectively. The results show that the proposed algorithm achieves better load balancing than the PAMLS method. In the original PAMLS method using 32 threads, 32 substructure clusters have the same number of distributed substructures and each substructure cluster has approximately the same number of DOFs: 262086–263886. Although the relative difference between the minimum and maximum DOFs for each substructure cluster is 0.7%, an enormous transformation time imbalance occurs in the original PAMLS method. In addition, we again observe that the parallelism for the implicit back transformation also performs well regardless of the cutoff frequencies.

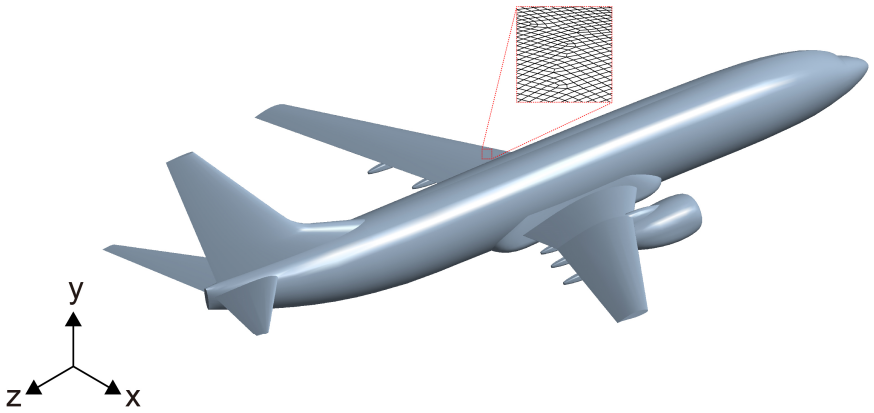


Fig. 2.15. Airplane.

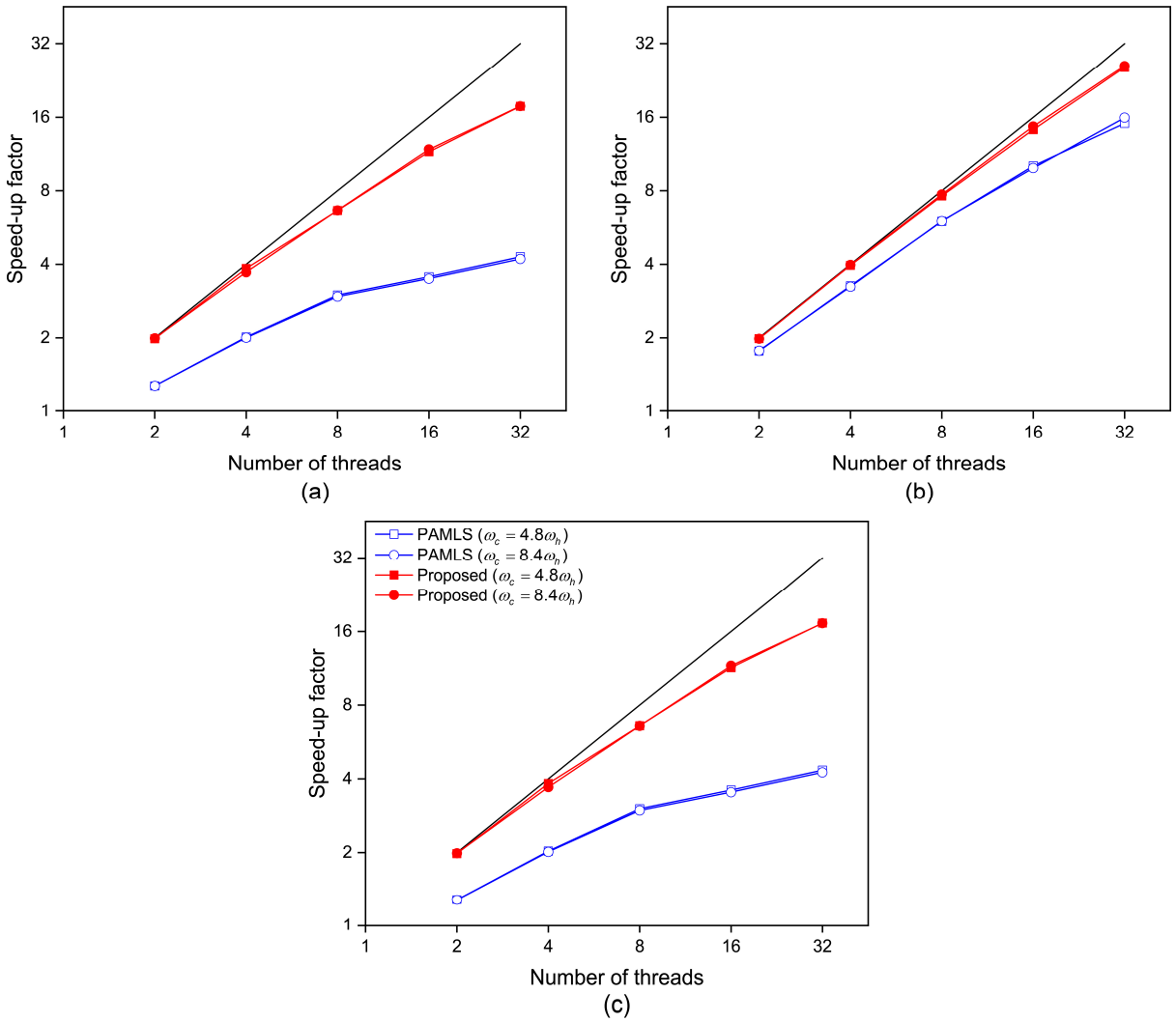


Fig. 2.16. Speed-up factors for the airplane problem: (a) transformation and (b) implicit back transformation procedures.

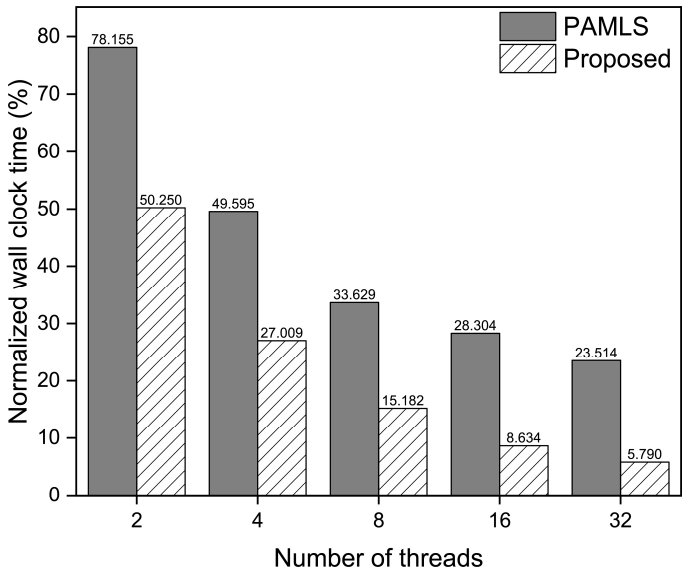


Fig. 2.17. Normalized wall clock times for the airplane model with $\omega_c = 8.4\omega_h$. The wall clock times are normalized by the total computation time required for the serial AMLS method.

Table 2.4. Wall clock times for the airplane model with $\omega_c = 8.4\omega_h$, where N_t denotes the number of threads used and 'total' denotes the time elapsed for the whole procedure of the original PAMLS and proposed methods. The wall clock times are normalized by the total computation time required for the serial AMLS method.

N_t	PAMLS (%)			Proposed (%)		
	Total	Transformation	Back transformation	Total	Transformation	Back transformation
2	78.155	76.593	1.324	50.250	48.848	1.183
4	49.595	48.651	0.724	27.009	26.202	0.590
8	33.629	33.034	0.390	15.182	14.642	0.304
16	28.304	27.848	0.237	8.634	8.245	0.160
32	23.514	23.141	0.147	5.790	5.477	0.090

2.5. Weak scaling

In this section, we investigate weak scaling performance which is typically measured by increasing the number of threads used while the workload per thread remains constant. Instead of the fixed problem size (i.e. a fixed number of DOFs for an FE model) as in Section 2.4, the problem size grows as the number of threads increases. In the AMLS method, it is difficult to assign the same workload per thread when the problem size changes. Assigning the same workload per thread is not simply achieved by assigning the same substructure sizes to each thread.

The rectangular plate problem described in Section 2.4.1 is considered again, where the number of eigensolutions sought is 150. The plate is modeled using six different meshes ($2N \times N$) of four-node shell elements, where $2N$ elements are assigned along the plate length. We generate the meshes for the plate model so that the computation time for the serial AMLS method approximately doubles as the problem size increases by one step. Information about each FE model is listed in Table 2.5.

Table 2.6 and Fig. 2.18 show the average substructure sizes at each level when partitioning the FE models by using METIS [51]. It is observed that the average substructure sizes at each level become larger as the problem size grows, except for the lowest level substructures that do not have any child substructure. As mentioned in Section 2.3.1, the large number of DOFs for a shared substructure (i.e. interface substructure) incurs more idle time at the synchronization points. In other words, for 1024×512 mesh, the parallel efficiency of both the original PAMLS method and the proposed algorithm would decrease.

The normalized wall clock times for the whole procedure are shown in Fig. 2.19 and Table 2.7, where the wall clock times are normalized by the total computation time required for 332×166 mesh in the serial AMLS method. As expected, for 1024×512 mesh, the parallel performance of both the original PAMLS method and the proposed algorithm decreases. However, with the proposed algorithm, the parallel performance is consistently improved. The effect of the mesh partitioning on the performance is discussed further in Appendix B.

Table 2.5. Finite element models for a weak scaling analysis of the rectangular plate problem, where N_t denotes the number of threads used.

N_t	Number of elements (length × width)	Number of substructures (levels)	DOFs
1	55112 (332 × 166)	1023 (9)	330678
2	88200 (420 × 210)	2047 (10)	529206
4	138338 (526 × 263)	2047 (10)	830034
8	215168 (656 × 328)	4095 (11)	1291014
16	336200 (820 × 410)	8191 (12)	2017206
32	524288 (1024 × 512)	8191 (12)	3145734

Table 2.6. Mean substructure sizes of each level for the rectangular plate models of 332 × 166 and 1024 × 512 meshes.

Level	Number of substructures	Average substructure size (DOFs)					
		332 × 166 mesh	420 × 210 mesh	526 × 263 mesh	656 × 328 mesh	820 × 410 mesh	1024 × 512 mesh
0	1	1002	1428	1626	2166	2772	3306
1	2	1233	1479	1788	2214	2571	3579
2	4	543	758	951	1097	1427	1799
3	8	534	614	899	1086	1466	1672
4	16	319	387	492	609	771	953
5	32	242	331	416	523	634	819
6	64	157	199	251	301	389	505
7	128	110	143	187	240	309	394
8	256	72	94	123	150	194	252
9	512	524	65	84	110	140	178
10	1024	-	405	667	71	92	121
11	2048	-	-	-	505	62	80
12	4096	-	-	-	-	381	624

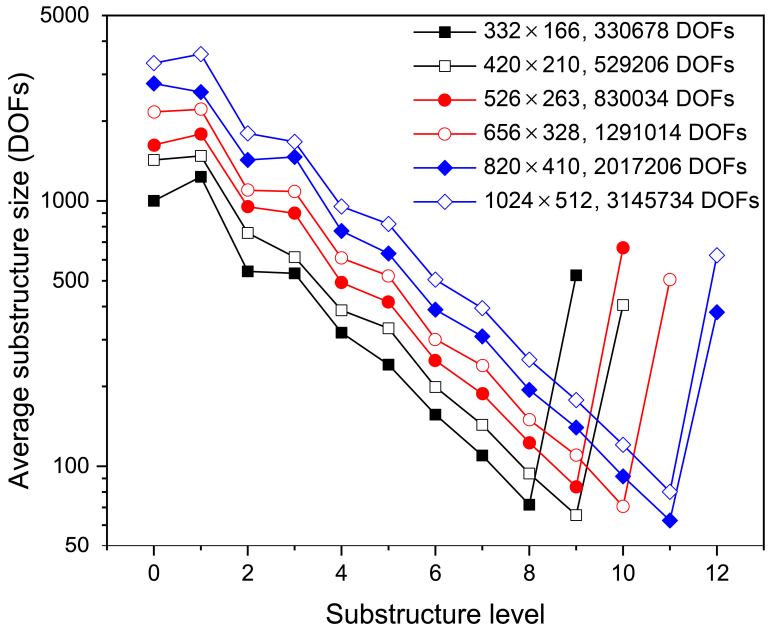


Fig. 2.18. Mean substructure sizes of each level for the rectangular plate models of 332 × 166 and 1024 × 512 meshes.

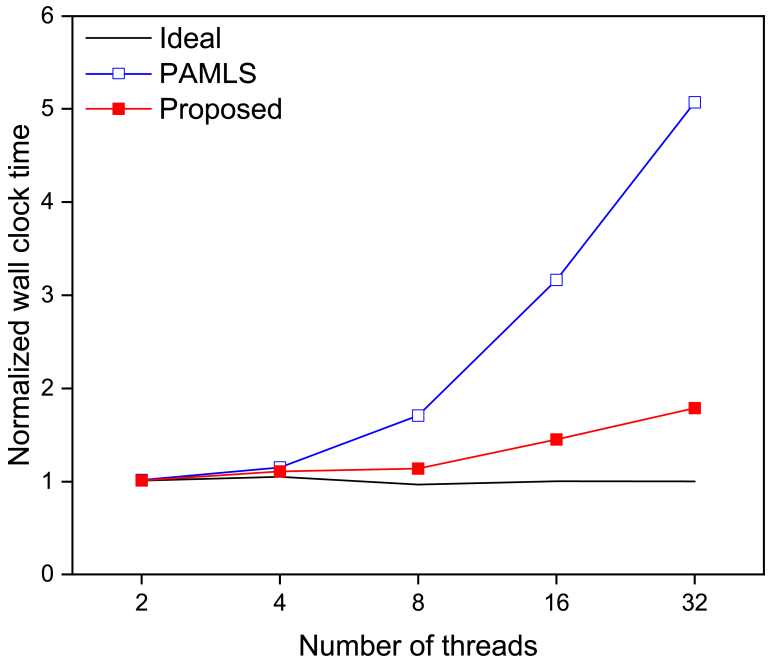


Fig. 2.19. Normalized wall clock times for a weak scaling analysis of the rectangular plate problem. The wall clock times are normalized by the total computation time required for $N = 166$ mesh in the serial AMLS method.

Table 2.7. Normalized wall clock times for a weak scaling analysis of the rectangular plate problem, where N_t denotes the number of threads used. The wall clock times are normalized by the total computation time required for $N = 166$ mesh in the serial AMLS method.

N_t	Ideal	PAMLS	Proposed
2	1.012	1.020	1.014
4	1.052	1.153	1.111
8	0.969	1.710	1.141
16	1.006	3.165	1.453
32	1.003	5.070	1.789

2.6. Concluding remarks

We proposed a load balancing algorithm for the parallel AMLS (PAMLS) method, which consisted of two types of granularity. To avoid determinacy races, we defined a shared substructure and a substructure cluster that is a subtree composed of distributed substructures. In coarse-grained parallelism, the transformation procedure was split into tasks corresponding to the substructure clusters and shared substructures, in which the number of substructure clusters was determined by using a given cutoff level instead of the number of threads. Both the explicit and implicit back transformation procedures were also split in a similar fashion. Fine-grained parallelism was used to reduce the idle time for the transformation of shared substructures. The performance of the proposed algorithm was demonstrated through numerical examples. It was observed that the proposed algorithm significantly improved the efficiency of the original PAMLS method without repartitioning.

Chapter 3. A nonlinear model reduction method using a coarse mesh model

As the demand for improved accuracy of numerical simulations increases, more complex models are required. Performing multiple simulations on such complex models induces an unmanageably large computational burden. Projection-based model reduction methods [3] reduce this computational burden by projecting the high-dimensional original model into a low-dimensional subspace. It implies that the suitably defined low-dimensional subspace well approximates the solution space.

In the field of projection-based model reduction, the offline-online strategy is commonly employed. In the offline phase, snapshot data is collected from the original model, and suitable low-dimensional basis vectors are computed from snapshots. Then, the reduced model is constructed in such a way that the original model is projected onto the low-dimensional subspace. Finally, in the online phase, approximate solutions are computed in the reduced model. The proper orthogonal decomposition (POD) [69–71] with Galerkin projection has been widely used to compute the low-dimensional basis vectors and applied in various engineering fields such as structural dynamics, fluid mechanics, signal processing, and image processing. An attractive feature of the POD is that the dominant components of the original model are captured by a low-dimensional subspace, which is established in optimal.

Although the POD successfully provides the reduced model for nonlinear problems, no significant speedup is achieved during the online phase because the nonlinear terms are computed on the high-dimensional original model. Sparse sampling methods [9,72–76], also known as hyper-reduction [77], are widely used to overcome this computational inefficiency. They compute nonlinear terms at a few sampling points and then approximate the nonlinear terms of the original model. The discrete empirical interpolation method (DEIM) [10] selects interpolation points (i.e. sampling points) through a greedy approach and provides the approximate nonlinear terms via interpolation in a low-dimensional subspace. However, some unstable results are observed in the DEIM approximations [11]. Many variants of the DEIM were developed to improve the efficiency and accuracy, such as the unassembled [13], localized [11,78], and adaptive [79] methods. Recently, the GappyPOD+E [12], a deterministic oversampling algorithm, has been proposed to avoid the stability issue.

Instead of the approximation from a few sampling points, some reduction methods [80,81] for elastoplastic problems were developed using POD with domain decomposition. In these works, POD is applied to only elastic subdomains, not plastic regions. Another approach for the efficient online phase is the use of machine learning methods. The works in [82,83] proposed non-intrusive reduced basis methods using artificial neural networks (ANN), which approximates the map between inputs and POD coefficients for the reduced model. In Ref. [84], constitutive models for coarse meshes are learned, and a coarsened mesh model is used during the online phase. At variance with this work, in Ref. [85], ANN provides the correction vector for a coarse mesh model at a given configuration. It resulted in improved solution accuracy compared to that of a standard coarse mesh model.

Despite these considerable efforts, there is still room for more effective nonlinear model reduction. In this chapter, we propose a novel nonlinear model reduction method using a coarse mesh, named coarse mesh projection. The key concept is to compute nonlinear terms on the coarse mesh model corresponding to the domain considered in the original finite element model. This is achieved by establishing reduced basis vectors for the coarse mesh model through the finite element interpolation of POD basis vectors for the original model. Herein, the reduced basis vectors for the coarse mesh model are referred to as the coarsened POD basis vectors. The coarse mesh model is then projected onto a space spanned by the coarsened POD basis vectors, and the approximate solution for the original model is computed from the reduced solution for the coarse mesh model. The approximate solution corrects the nonlinear terms at the quadrature points of the coarse mesh model. By leveraging the rigor of the finite element framework, the proposed method reliably provides the approximate solution without snapshots for nonlinear terms.

In Section 3.1, we briefly review the total Lagrangian formulation for a general three-dimensional element and the standard POD with Galerkin projection. In Section 3.2, sparse sampling methods such as the DEIM and GappyPOD+E are introduced. Then, the proposed method is presented in Section 3.3, and its performance is demonstrated through numerical examples in Section 3.4. Finally, conclusions are given in Section 3.5.

3.1. Problem statement

3.1.1. Total Lagrangian formulation

In this section, without loss of generality, the total Lagrangian formulation for the isoparametric finite element discretization is reviewed. Note that the left superscript represents the configuration in which quantities occur, and the quantities are measured in the initial configuration.

Let us consider the position interpolation of a general *q*-node three-dimensional finite element at time *t*

$$ {}^{t}\mathbf{x}(\xi_1,\xi_2,\xi_3) = \sum_{i=1}^{q} h_i(\xi_1,\xi_2,\xi_3)\,{}^{t}\mathbf{x}_i \,, \tag{3.1} $$

where $ {}^{t}\mathbf{x} $ is the position vector, $ {}^{t}\mathbf{x}_i $ is the position vector of node *i*, and $ h_i(\xi_1,\xi_2,\xi_3) $ are the standard finite element shape functions with the natural coordinates $ \xi_1 $, $ \xi_2 $, and $ \xi_3 $.

From the configuration at times $ t $ to $ t+\Delta t $, the incremental displacement vector is defined by

$$ \mathbf{u}(\xi_1,\xi_2,\xi_3) = {}^{t+\Delta t}\mathbf{x}(\xi_1,\xi_2,\xi_3) - {}^{t}\mathbf{x}(\xi_1,\xi_2,\xi_3) \,, \tag{3.2} $$

and the displacement interpolant $ \mathbf{u} $ is given by

$$ \mathbf{u}(\xi_1,\xi_2,\xi_3) = \sum_{i=1}^{q} h_i(\xi_1,\xi_2,\xi_3)\mathbf{u}_i \,, \tag{3.3} $$

where $ \mathbf{u}_i $ is the displacement vector of node *i*.

The Green-Lagrange strain tensor is written as

$$ {}^{t}\boldsymbol{\varepsilon} = {}^{t}\varepsilon_{ij}\,{}^{0}\mathbf{g}^{i} \otimes {}^{0}\mathbf{g}^{j} \quad \text{with} \quad {}^{0}\mathbf{g}^{i} = \frac{\partial \xi_i}{\partial\, {}^{0}\mathbf{x}} \tag{3.4} $$

where the covariant component $ {}^{t}_{0}\varepsilon_{ij} $ is defined by

$$ {}^{t}\varepsilon_{ij} = \frac{1}{2}\left({}^{t}\mathbf{g}_i \cdot {}^{t}\mathbf{g}_j - {}^{0}\mathbf{g}_i \cdot {}^{0}\mathbf{g}_j\right) \tag{3.5} $$

with

$$ {}^{t}\mathbf{g}_i = \frac{\partial\, {}^{t}\mathbf{x}}{\partial \xi_i} \,, \quad {}^{0}\mathbf{g}_i = \frac{\partial\, {}^{0}\mathbf{x}}{\partial \xi_i} \,. \tag{3.6} $$

Note that $ \mathbf{g}_i $ and $ \mathbf{g}^i $ denote the covariant and contravariant basis vectors, respectively.

From the configuration at times $ t $ to $ t+\Delta t $, the incremental Green-Lagrange strain tensor $ \varepsilon_{ij} $ is given by

$$ \varepsilon_{ij} = {}^{t+\Delta t}\varepsilon_{ij} - {}^{t}\varepsilon_{ij} = e_{ij} + \eta_{ij} \,, \tag{3.7} $$

where $ e_{ij} $ and $ \eta_{ij} $ are the linear and nonlinear parts of $ \varepsilon_{ij} $, respectively.

Using the strain-displacement matrices, e_{ij} and η_{ij} are obtained by

$$e_{ij} = {}^t\mathbf{B}_{ij}\mathbf{d}\ ,\quad \eta_{ij} = \frac{1}{2}\mathbf{d}^T\,{}^t\mathbf{N}_{ij}\mathbf{d} \tag{3.8}$$

with

$$\mathbf{d} = [\mathbf{u}_1^T \quad \mathbf{u}_2^T \quad \cdots \quad \mathbf{u}_q^T]^T\ , \tag{3.9}$$

where ${}^t\mathbf{B}_{ij}$ and ${}^t\mathbf{N}_{ij}$ denote the strain-displacement matrices, and $\mathbf{d}$ is the nodal displacement vector.

Then, the time-independent mass matrix $\mathbf{M} \in \mathbb{R}^{n\times n}$, tangential stiffness matrix ${}^t\mathbf{K} \in \mathbb{R}^{n\times n}$, and internal force vector ${}^t\mathbf{F} \in \mathbb{R}^n$ are given by

$$\mathbf{M} = \mathop{\mathbf{A}}_{m=1}^{n_e}(\mathbf{M}^{(m)}) \quad \text{with} \quad \mathbf{M}^{(m)} = \int_{{}^0V^{(m)}} {}^0\rho^{(m)}(\mathbf{H}^{(m)})^T\mathbf{H}^{(m)}d\,{}^0V^{(m)}\ , \tag{3.10}$$

$${}^t\mathbf{K} = \mathop{\mathbf{A}}_{m=1}^{n_e}({}^t\mathbf{K}^{(m)}) \quad \text{with} \quad {}^t\mathbf{K}^{(m)} = \int_{{}^0V^{(m)}} ({}^t\mathbf{B}_{ij}^{(m)})^T C_{ijkl}^{(m)}\,{}^t\mathbf{B}_{kl}^{(m)} + {}^t\mathbf{N}_{ij}^{(m)}\,{}^tS_{ij}^{(m)}d\,{}^0V^{(m)}\ , \tag{3.11}$$

$${}^t\mathbf{F} = \mathop{\mathbf{A}}_{m=1}^{n_e}({}^t\mathbf{F}^{(m)}) \quad \text{with} \quad {}^t\mathbf{F}^{(m)} = \int_{{}^0V^{(m)}} ({}^t\mathbf{B}_{ij}^{(m)})^T\,{}^tS_{ij}^{(m)}d\,{}^0V^{(m)}\ , \tag{3.12}$$

where $\mathbf{A}$ is an assembly operator [2], n_e is the number of elements, and the superscript (m) denotes the quantity for the element m. The matrix $\mathbf{H}^{(m)}$ is a matrix of the geometry interpolation function, $C_{ijkl}^{(m)}$ is the constitutive law tensor, and ${}^tS_{ij}^{(m)}$ is the second Piola-Kirchhoff (PK) stress tensor. Note that ${}^0V^{(m)}$ denotes the volume of the element m at the initial configuration.

The integration of finite element matrices in Eqs. (3.10)-(3.12) is numerically performed by the Gauss quadrature. As an example, ${}^t\mathbf{F}^{(m)}$ is calculated by

$${}^t\mathbf{F}^{(m)} = \int_{{}^0V^{(m)}} ({}^t\mathbf{B}_{ij}^{(m)})^T\,{}^tS_{ij}^{(m)}d\,{}^0V^{(m)} = \sum_p\sum_q\sum_r \alpha_p\alpha_q\alpha_r({}^t\mathbf{B}_{ij}^{(m)})^T\,{}^tS_{ij}^{(m)}\left|{}^t\mathbf{J}^{(m)}\right|\ , \tag{3.13}$$

where α denotes the weight factor for the one-dimensional integration and $\left|{}^t\mathbf{J}^{(m)}\right|$ is the determinant of the Jacobian matrix of the element m at time t. Note that the quantities ${}^t\mathbf{B}_{ij}^{(m)}$, ${}^tS_{ij}^{(m)}$, and ${}^t\mathbf{J}^{(m)}$ are computed on the corresponding quadrature point.

After an assemblage of all finite elements, the following equations for static analysis and explicit and implicit dynamic analyses are respectively obtained:

$${}^t\mathbf{K}\mathbf{U} = {}^{t+\Delta t}\mathbf{R} - {}^t\mathbf{F}\ , \tag{3.14}$$

$$\mathbf{M}\,{}^t\ddot{\mathbf{U}} = {}^t\mathbf{R} - {}^t\mathbf{F}\ , \tag{3.15}$$

$$\mathbf{M}\,{}^{t+\Delta t}\ddot{\mathbf{U}} + {}^t\mathbf{K}\mathbf{U} = {}^{t+\Delta t}\mathbf{R} - {}^t\mathbf{F} \tag{3.16}$$

where $\mathbf{U} \in \mathbb{R}^n$ is the incremental displacement vector, ${}^t\ddot{\mathbf{U}} \in \mathbb{R}^n$ and ${}^{t+\Delta t}\ddot{\mathbf{U}} \in \mathbb{R}^n$ are acceleration vectors at

times t and $t+\Delta t$, respectively, and ${}^{t}\mathbf{R} \in \mathbb{R}^n$ and ${}^{t+\Delta t}\mathbf{R} \in \mathbb{R}^n$ are the external load vectors at times t and $t+\Delta t$, respectively.

3.1.2. Proper orthogonal decomposition with Galerkin projection

Suppose that the displacement vector can be approximated by a linear combination of arbitrary basis vectors

$$\mathbf{\Phi} \in \mathbb{R}^{n \times n_r} \tag{3.17}$$

with

$$\mathbf{\Phi} = \begin{bmatrix} \mathbf{\Phi}_1 & \mathbf{\Phi}_2 & \cdots & \mathbf{\Phi}_{n_r} \end{bmatrix}, \tag{3.18}$$

where n_r is the number of basis vectors and much smaller than the dimension of the original model *n*.

The incremental displacement vector is then written as

$$\mathbf{U} \approx \mathbf{\Phi}\bar{\mathbf{U}}, \tag{3.19}$$

where $\bar{\mathbf{U}} \in \mathbb{R}^{n_r}$ is the coefficient vector associated with $\mathbf{\Phi}$, viz the reduced solution vector.

Enforcing the Galerkin projection with $\mathbf{\Phi}$, the reduced equations of $n_r \times n_r$ are obtained by

$${}^{t}\bar{\mathbf{K}}\bar{\mathbf{U}} = {}^{t+\Delta t}\bar{\mathbf{R}} - {}^{t}\bar{\mathbf{F}}, \tag{3.20}$$

$$\bar{\mathbf{M}}\,{}^{t}\ddot{\bar{\mathbf{U}}} = {}^{t}\bar{\mathbf{R}} - {}^{t}_{0}\bar{\mathbf{F}}, \tag{3.21}$$

$$\bar{\mathbf{M}}\,{}^{t+\Delta t}\ddot{\bar{\mathbf{U}}} + {}^{t}\mathbf{K}\bar{\mathbf{U}} = {}^{t+\Delta t}\bar{\mathbf{R}} - {}^{t}\bar{\mathbf{F}} \tag{3.22}$$

with

$${}^{t}\bar{\mathbf{K}} = \mathbf{\Phi}^T\,{}^{t}\mathbf{K}\mathbf{\Phi},\ \ \bar{\mathbf{M}} = \mathbf{\Phi}^T\mathbf{M}\mathbf{\Phi},\ \ {}^{t}\bar{\mathbf{F}} = \mathbf{\Phi}^T\,{}^{t}\mathbf{F},\ \ {}^{t}\bar{\mathbf{R}} = \mathbf{\Phi}^T\,{}^{t}\mathbf{R},\ \ {}^{t+\Delta t}\bar{\mathbf{R}} = \mathbf{\Phi}^T\,{}^{t+\Delta t}\mathbf{R}, \tag{3.23}$$

where the overbar $\bar{(\cdot)}$ denotes the reduced quantities. Since the accuracy in the approximate solutions of Eqs. (3.20)-(3.22) depends on the basis vectors $\mathbf{\Phi}$, we should establish the appropriate basis vectors.

The proper orthogonal decomposition (POD) is to construct the subspace that minimizes the mean square error of approximation of a given space. In order to compute the POD basis matrix $\mathbf{\Phi}$, we first consider the following snapshot matrix

$$\mathbf{S}_d = [\mathbf{U}_1 \quad \mathbf{U}_2 \quad \cdots \quad \mathbf{U}_{n_d}] \in \mathbb{R}^{n \times n_d}, \tag{3.24}$$

where the columns of $\mathbf{S}_d$ are the displacement vectors of the original model in the configuration at certain times, and n_d is the number of displacement snapshots. Note that the snapshots should be representative samples of solutions.

The approximation by using the POD basis matrix $\mathbf{\Phi}$ minimizes the mean square error of approximate solutions

$$\sum_{i=1}^{n_d} \left\| \mathbf{U}_i - \boldsymbol{\Phi}\boldsymbol{\Phi}^T \mathbf{U}_i \right\|_2 . \tag{3.25}$$

To find $\boldsymbol{\Phi}$ that minimizes Eq. (3.25), the low-rank approximation of $\mathbf{S}_d$ from the singular value decomposition (SVD) provides

$$\mathbf{S}_d = \boldsymbol{\Phi}_d \boldsymbol{\Sigma}_d \boldsymbol{\Omega}_d^T \tag{3.26}$$

with

$$\boldsymbol{\Phi}_d = \begin{bmatrix} \boldsymbol{\Phi}_1 & \boldsymbol{\Phi}_2 & \cdots & \boldsymbol{\Phi}_{n_d} \end{bmatrix}, \tag{3.27}$$

where columns of $\boldsymbol{\Phi}_d \in \mathbb{R}^{n \times n_d}$ and $\boldsymbol{\Omega}_d \in \mathbb{R}^{n_d \times n_d}$ consist of the left and right singular vectors, respectively, and $\boldsymbol{\Sigma}_d \in \mathbb{R}^{n_d \times n_d}$ is a diagonal matrix composed of the singular values in descending order.

The number of POD basis vectors is typically selected through the following criterion

$$\frac{\sum_{i=1}^{n_r} \sigma_i^2}{\sum_{i=1}^{n_d} \sigma_i^2} > \epsilon , \tag{3.28}$$

where σ_i is the *i*th singular value, and $\epsilon \in [0 \quad 1]$ is a given tolerance, and the mean square error becomes smaller as ϵ approaches one. This criterion is often referred to as the energy captured by n_r POD basis vectors.

3.2. Sparse sampling methods

The POD with Galerkin projection permits that the solution is approximated by solving the reduced equation whose dimension of $n_r \times n_r$. In the reduced equation, the tangential stiffness matrix ${}^t\mathbf{K}$ and/or the internal force vector ${}^t\mathbf{F}$ should be computed at each iteration in a load step or time step because of the configuration dependence. Such computation requires lifting $\bar{\mathbf{U}}$ back to $\mathbf{U}$, computing the nonlinear terms on the original dimension *n*, and projecting the nonlinear terms onto a subspace spanned by $\mathbf{\Phi}$. Although the original model is reduced, it leads to no significant improvement in efficiency and is referred to as the lifting bottleneck. In order to address this inefficiency, sparse sampling methods are commonly used. In this section, we briefly review the DEIM [10] and the GappyPOD+E sampling algorithm [12].

The internal force vector is decomposed into the linear and nonlinear parts ${}^t\mathbf{F}_l$ and ${}^t\mathbf{F}_n$ as

$${}^t\mathbf{F} = {}^t\mathbf{F}_l + {}^t\mathbf{F}_n \tag{3.29}$$

with

$${}^t\mathbf{F}_l = {}^0\mathbf{K}\,{}^t\mathbf{U}, \tag{3.30}$$

where ${}^0\mathbf{K}$ is the stiffness matrix at the first iteration of the initial configuration, and ${}^t\mathbf{U}$ is the displacement vector at time *t*.

Suppose that ${}^t\mathbf{F}_n$ can be also approximated by the POD basis vectors

$$\mathbf{\Xi} \in \mathbb{R}^{n \times n_n}, \tag{3.31}$$

where $\mathbf{\Xi}$ is computed by the SVD of nonlinear force snapshots, and n_n is the number of POD basis vectors for the nonlinear internal force vector. Note that n_n is much smaller than the dimension of the original model *n*.

Employing $\mathbf{\Xi}$, the nonlinear force vector ${}^t\mathbf{F}_n$ is then approximated as

$${}^t\mathbf{F}_n \approx \mathbf{\Xi}\,{}^t\bar{\mathbf{F}}_n, \tag{3.32}$$

where ${}^t\bar{\mathbf{F}}_n \in \mathbb{R}^{n_n}$ is the POD coefficient associated with $\mathbf{\Xi}$.

In general, Eq. (3.32) is highly overdetermined because of $n_n \ll n$. To find the POD coefficient ${}^t\bar{\mathbf{F}}_n$, a selection matrix $\mathbf{P} \in \mathbb{R}^{n \times n_s}$ is defined by

$$\mathbf{P} = \begin{bmatrix} \mathbf{e}_{\wp_1} & \mathbf{e}_{\wp_2} & \cdots & \mathbf{e}_{\wp_{n_s}} \end{bmatrix} \tag{3.33}$$

with

$$\mathbf{e}_{\wp_i} = \begin{bmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \end{bmatrix}^T, \tag{3.34}$$

where the $\wp_i$ th row of $\mathbf{e}_{\wp_i}$ is the value of unity and other entries are zero.

Using the selection matrix $\mathbf{P}$ in Eq. (3.33), ${}^t\overline{\mathbf{F}}_n$ is defined in

$$\mathbf{P}^T {}^t\mathbf{F}_n = \mathbf{P}^T \boldsymbol{\Xi} {}^t\overline{\mathbf{F}}_n , \tag{3.35}$$

and ${}^t\overline{\mathbf{F}}_n$ is then solved by

$${}^t\overline{\mathbf{F}}_n = (\mathbf{P}^T\boldsymbol{\Xi})^+ \mathbf{P}^T {}^t\mathbf{F}_n , \tag{3.36}$$

where $(\cdot)^+$ denotes a generalized inverse.

Finally, the approximation of ${}^t\mathbf{F}_n$ is obtained by

$${}^t\mathbf{F}_n = \boldsymbol{\Xi}(\mathbf{P}^T\boldsymbol{\Xi})^+ \mathbf{P}^T {}^t\mathbf{F}_n , \tag{3.37}$$

where $\mathbf{P}^T {}^t\mathbf{F}_n \in \mathbb{R}^{n_s}$ composed of n_s entries in ${}^t\mathbf{F}_n$. Therefore, the nonlinear part of the internal force vector can be computed by n_s entries. Note that $\boldsymbol{\Xi}(\mathbf{P}^T\boldsymbol{\Xi})^+$ is precomputed in the offline phase.

There are several sampling point selection algorithms to reduce the following error

$$|| {}^t\mathbf{F}_n - \boldsymbol{\Xi}(\mathbf{P}^T\boldsymbol{\Xi})^+ \mathbf{P}^T {}^t\mathbf{F}_n ||_2 , \tag{3.38}$$

and the error is bounded [86] according to

$$|| {}^t\mathbf{F}_n - \boldsymbol{\Xi}(\mathbf{P}^T\boldsymbol{\Xi})^+ \mathbf{P}^T {}^t\mathbf{F}_n ||_2 \le || (\mathbf{P}^T\boldsymbol{\Xi})^+ ||_2 || (\mathbf{I} - \boldsymbol{\Xi}\boldsymbol{\Xi}^T) {}^t\mathbf{F}_n ||_2 , \tag{3.39}$$

where $|| (\mathbf{P}^T\boldsymbol{\Xi})^+ ||_2$ and $|| (\mathbf{I} - \boldsymbol{\Xi}\boldsymbol{\Xi}^T) {}^t\mathbf{F}_n ||_2$ are the errors due to the selection of sampling points and orthogonal projection by POD, respectively.

The DEIM generates $\mathbf{P}$ by Algorithm 3.1 [10]. In this work, because of $n_n = n_s$, Eq. (3.36) becomes

$${}^t\mathbf{F}_n = (\mathbf{P}^T\boldsymbol{\Xi})^{-1} \mathbf{P}^T {}^t\mathbf{F}_n , \tag{3.40}$$

and, in a similar fashion, the derivative of ${}^t\mathbf{F}_n$ with respect to ${}^t\mathbf{U}$ is computed by

$$\frac{\partial {}^t\mathbf{F}_n}{\partial {}^t\mathbf{U}} = \boldsymbol{\Xi}(\mathbf{P}^T\boldsymbol{\Xi})^{-1} \mathbf{P}^T \frac{\partial {}^t\mathbf{F}_n}{\partial {}^t\mathbf{U}} . \tag{3.41}$$

Using Eq. (3.41), the tangential stiffness matrix is then calculated by

$${}^t\mathbf{K} = \frac{\partial {}^t\mathbf{F}_l}{\partial {}^t\mathbf{U}} + \frac{\partial {}^t\mathbf{F}_n}{\partial {}^t\mathbf{U}} = {}^0\mathbf{K} + \boldsymbol{\Xi}(\mathbf{P}^T\boldsymbol{\Xi})^{-1} \mathbf{P}^T \frac{\partial {}^t\mathbf{F}_n}{\partial {}^t\mathbf{U}} \tag{3.42}$$

with

$$\frac{\partial {}^t\mathbf{F}_n}{\partial {}^t\mathbf{U}} = {}^t\mathbf{K}_n = {}^t\mathbf{K} - {}^0\mathbf{K} , \tag{3.43}$$

where ${}^t\mathbf{K}_n$ is the nonlinear part of the tangential stiffness matrix ${}^t\mathbf{K}$.

In Ref. [12], a deterministic oversampling algorithm has been proposed to avoid the unstable behavior in the DEIM approximation. The algorithm first selects the DEIM interpolation points using the QR decomposition with

column pivoting [76]. Then, additional sampling points are selected in a greedy fashion that $\| (\mathbf{P}^T\boldsymbol{\Xi})^+ \|_2$ is minimized, which is equivalent to maximize the smallest singular value of $\mathbf{P}^T\boldsymbol{\Xi}$. The GappyPOD+E sampling algorithm is summarized in Algorithm 3.2.

In line 4 in Algorithm 3.2, the SVD of $\mathbf{P}^T\boldsymbol{\Xi}$ is performed by

$$\mathbf{P}^T\boldsymbol{\Xi} = \mathbf{V}\boldsymbol{\Sigma}_n\mathbf{W}^T \tag{3.44}$$

with

$$\boldsymbol{\Sigma}_n = \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_{n_n} \end{bmatrix}, \tag{3.45}$$

where σ_i denotes the *i*th singular value of $\mathbf{P}^T\boldsymbol{\Xi}$ in descending order, viz. $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{n_n}$.

Using the result of Eq. (3.44), a new sampling point is iteratively selected in each iteration of the GappyPOD+E sampling algorithm.

Algorithm 3.1. DEIM sampling points selection

input: POD basis vectors $\{\boldsymbol{\Xi}_i\}_{i=1}^{n_n}$

output: Selection matrix $\mathbf{P}$

find the jth row corresponding to the maximum entry of $|\boldsymbol{\Xi}_1|$

$\mathbf{P} \leftarrow \mathbf{e}_j$, $\boldsymbol{\Gamma} \leftarrow \boldsymbol{\Xi}_1$

for $i = 2$ to n_n

  solve $(\mathbf{P}^T\boldsymbol{\Gamma})\mathbf{c} = \mathbf{P}^T\boldsymbol{\Xi}_i$ for $\mathbf{c}$

  $\mathbf{r} \leftarrow \boldsymbol{\Xi}_i - \boldsymbol{\Gamma}\mathbf{c}$

  find the jth row corresponding to the maximum entry of $|\mathbf{r}|$

  $\mathbf{P} \leftarrow [\mathbf{P} \quad \mathbf{e}_j]$, $\boldsymbol{\Gamma} \leftarrow [\boldsymbol{\Gamma} \quad \boldsymbol{\Xi}_i]$

Algorithm 3.2. GappyPOD+E sampling points selection

input: POD basis vectors $\{\boldsymbol{\Xi}_i\}_{i=1}^{n_n}$, the number of additional sampling points n_a

output: Selection matrix $\mathbf{P}$

perform the QR decomposition with column pivoting $\boldsymbol{\Xi}^T\mathbf{Z} = \mathbf{QR}$, where $\mathbf{Z} = \left[\mathbf{e}_{\wp_1} \quad \mathbf{e}_{\wp_2} \quad \cdots \quad \mathbf{e}_{\wp_n}\right]$

$\mathbf{P} \leftarrow \left[\mathbf{e}_{\wp_1} \quad \mathbf{e}_{\wp_2} \quad \cdots \quad \mathbf{e}_{\wp_{n_n}}\right]$

for $i = n_n + 1$ to $n_n + n_a$

  perform the singular value decomposition $\mathbf{P}^T\boldsymbol{\Xi} = \mathbf{V}\boldsymbol{\Sigma}_n\mathbf{W}^T$

  $g \leftarrow \sigma_{n_n-1}^2 - \sigma_{n_n}^2$

  $\mathbf{X} \leftarrow \mathbf{W}^T\boldsymbol{\Xi}^T$ with $\mathbf{X} = [\mathbf{X}_1 \quad \mathbf{X}_2 \quad \cdots \quad \mathbf{X}_n]$

  compute the jth entry of $\arg\max_j \left(g + \|\mathbf{X}_j\|_2^2 - \sqrt{(g + \|\mathbf{X}_j\|_2^2)^2 - 4g(\mathbf{e}_{n_n}^T\mathbf{X}_j)^2} \right)$ such that $\mathbf{e}_j \notin \mathbf{P}$

  $\mathbf{P} \leftarrow [\mathbf{P} \quad \mathbf{e}_j]$

3.3. Coarse mesh projection method

Let us consider a coarse mesh for the body represented by the original model as shown in Fig. 3.1. From the configuration at time *t*, the time-independent mass matrix $\hat{\mathbf{M}} \in \mathbb{R}^{\hat{n}\times\hat{n}}$, tangential stiffness matrix ${}^t\hat{\mathbf{K}} \in \mathbb{R}^{\hat{n}\times\hat{n}}$, and internal force vector ${}^t\hat{\mathbf{F}} \in \mathbb{R}^{\hat{n}}$ for the coarse mesh model are given by

$$\hat{\mathbf{M}} = \mathop{\mathsf{A}}_{m=1}^{\hat{n}_e}(\hat{\mathbf{M}}^{(m)}) \quad \text{with} \quad \hat{\mathbf{M}}^{(m)} = \int_{{}^0\hat{V}^{(m)}} {}^0\hat{\rho}^{(m)}(\hat{\mathbf{H}}^{(m)})^T\hat{\mathbf{H}}^{(m)} d\,{}^0\hat{V}^{(m)} , \tag{3.46}$$

$${}^t\hat{\mathbf{K}} = \mathop{\mathsf{A}}_{m=1}^{\hat{n}_e}({}^t\hat{\mathbf{K}}^{(m)}) \quad \text{with} \quad {}^t\hat{\mathbf{K}}^{(m)} = \int_{{}^0\hat{V}^{(m)}} ({}^t\hat{\mathbf{B}}_{ij}^{(m)})^T \hat{C}_{ijkl}^{(m)}\, {}^t\hat{\mathbf{B}}_{kl}^{(m)} + {}^t\hat{\mathbf{N}}_{ij}^{(m)}\, {}^t\hat{S}_{ij}^{(m)} d\,{}^0\hat{V}^{(m)} , \tag{3.47}$$

$${}^t\hat{\mathbf{F}} = \mathop{\mathsf{A}}_{m=1}^{\hat{n}_e}({}^t\hat{\mathbf{F}}^{(m)}) \quad \text{with} \quad {}^t\hat{\mathbf{F}}^{(m)} = \int_{{}^0\hat{V}^{(m)}} ({}^t\hat{\mathbf{B}}_{ij}^{(m)})^T\, {}^t\hat{S}_{ij}^{(m)} d\,{}^0V^{(m)} , \tag{3.48}$$

where $(\hat{\cdot})$ denotes the quantity for the coarse mesh model.

The reduced basis matrix associated with the coarse mesh model is written as

$$\boldsymbol{\Psi} \in \mathbb{R}^{\hat{n}\times n_r} . \tag{3.49}$$

Note that n_r is the number of POD basis vectors for the original model.

We coarsen the POD basis vectors for the original model $\boldsymbol{\Phi}$ to obtain $\boldsymbol{\Psi}$. In other words, $\boldsymbol{\Psi}$ is not computed by the SVD of the snapshot matrix for the coarse mesh model. Since a POD basis vector can be considered as a displacement vector, it can be interpolated using nodal values for a *q*-node element.

Employing the finite element interpolation, the *i*th nodal values of $\boldsymbol{\Psi}$ are defined by

$$\boldsymbol{\psi}_i = \sum_{j=1}^{q} h_j(\xi_1,\xi_2,\xi_3)\boldsymbol{\varphi}_j , \tag{3.50}$$

where $\boldsymbol{\psi}_i$ denotes the *i*th nodal values of $\boldsymbol{\Psi}$, $h_j(\xi_1,\xi_2,\xi_3)$ are the shape functions for a *q*-node element that includes the *i*th node of the coarse mesh model, and $\boldsymbol{\varphi}_j$ are corresponding nodal values for $\boldsymbol{\Phi}$. In this way, $\boldsymbol{\Psi}$ can be calculated for all DOFs corresponding to the coarse mesh model, and hereinafter, it is referred to as the coarsened POD basis matrix. Note that, in this step, the map between the coarse mesh model and the original model is found.

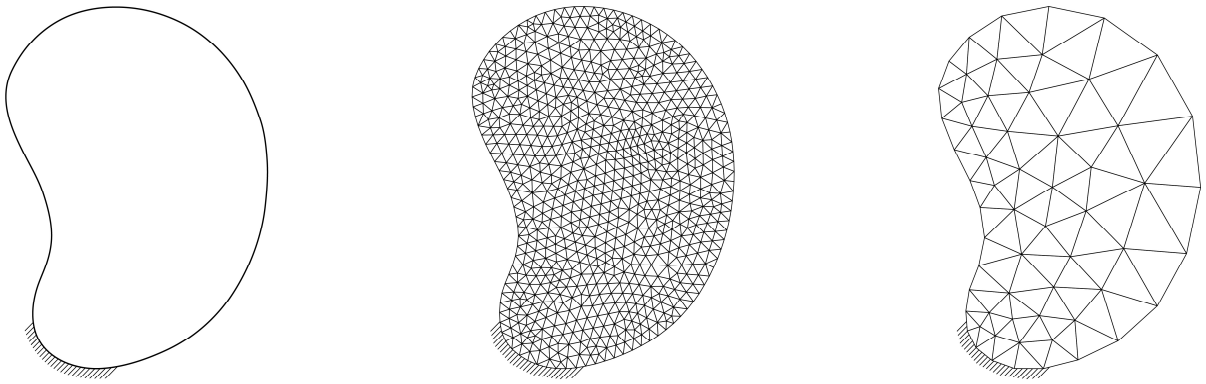
Projecting the coarse mesh model onto the subspace spanned by $\boldsymbol{\Psi}$, the reduced static equilibrium equation has the form as

$${}^t\tilde{\mathbf{K}}\tilde{\mathbf{U}} = {}^{t+\Delta t}\tilde{\mathbf{R}} - {}^t\tilde{\mathbf{F}} . \tag{3.51}$$

with

$${}^t\tilde{\mathbf{K}} = \boldsymbol{\Psi}^T\, {}^t\hat{\mathbf{K}}\boldsymbol{\Psi} , \quad {}^{t+\Delta t}\tilde{\mathbf{R}} = \boldsymbol{\Psi}^T\, {}^{t+\Delta t}\hat{\mathbf{R}} , \quad {}^t\tilde{\mathbf{F}} = \boldsymbol{\Psi}^T\, {}^t\hat{\mathbf{F}} \tag{3.52}$$

where $\tilde{\mathbf{U}} \in \mathbb{R}^{n_r}$ is the coefficient vector associated with $\boldsymbol{\Psi}$, viz. the reduced solution vector for the coarse mesh



(a) (b) (c)

Fig. 3.1. Finite element modeling of a generic structure: (a) geometry of a body, (b) original model, and (c) coarse model.

model, and the dimension of the reduced equation is the same as Eq. (3.20).

To decrease the error between the reduced solutions $\bar{\mathbf{U}}$ in Eq. (3.20) and $\tilde{\mathbf{U}}$ in Eq. (3.51), the tangential stiffness matrix and internal force vector ($^t\hat{\mathbf{K}}$ and $^t\hat{\mathbf{F}}$) should be corrected. Assuming that the coefficient vector $\tilde{\mathbf{U}}$ is close to $\bar{\mathbf{U}}$, the approximate solution for the original model is calculated by using the reduced solution for the coarse mesh model as

$$\mathbf{U} \approx \mathbf{\Phi}\tilde{\mathbf{U}} . \tag{3.53}$$

Then, $^t\hat{\mathbf{K}}^{(m)}$ and $^t\hat{\mathbf{F}}^{(m)}$ are corrected using the approximate solution for the original model as follows:

$$^t\hat{\mathbf{K}}^{(m)} = \int_{_0\hat{V}^{(m)}} (^t\hat{\mathbf{B}}_{ij}^{(m)})^T C_{ijkl}^{(m)} \, {}_0^t\hat{\mathbf{B}}_{kl}^{(m)} + {}^t\hat{\mathbf{N}}_{ij}^{(m)} \, {}^tS_{ij}^{(m)} d\,{}^0\hat{V}^{(m)} , \tag{3.54}$$

$$^t\hat{\mathbf{F}}^{(m)} = \int_{_0\hat{V}^{(m)}} (^t\hat{\mathbf{B}}_{ij}^{(m)})^T \, {}^tS_{ij}^{(m)} d\,{}^0\hat{V}^{(m)} . \tag{3.55}$$

To construct Eq. (3.51), $^t\hat{\mathbf{K}}^{(m)}$ and $^t\hat{\mathbf{F}}^{(m)}$ are obtained by Eqs. (3.54) and (3.55) instead of (3.47) and (3.48).

Note that the integration of $^t\hat{\mathbf{K}}^{(m)}$ and $^t\hat{\mathbf{F}}^{(m)}$ is numerically performed by the Gauss quadrature. Namely, the material law and second PK stress tensors $C_{ijkl}^{(m)}$ and $^tS_{ij}^{(m)}$ are computed from the original model corresponding to quadrature points in a coarse element. For example, the correction of a four-node two-dimensional solid finite element is shown in Fig. 3.2.

In dynamics, the same procedure is adopted, and the reduced equations are obtained by

$$\tilde{\mathbf{M}}\,{}^t\ddot{\tilde{\mathbf{U}}} = {}^t\tilde{\mathbf{R}} - {}^t\tilde{\mathbf{F}} , \tag{3.56}$$

$$\tilde{\mathbf{M}}\,{}^{t+\Delta t}\ddot{\tilde{\mathbf{U}}} + {}^t\tilde{\mathbf{K}}\tilde{\mathbf{U}} = {}^{t+\Delta t}\tilde{\mathbf{R}} - {}^t\tilde{\mathbf{F}} \tag{3.57}$$

with

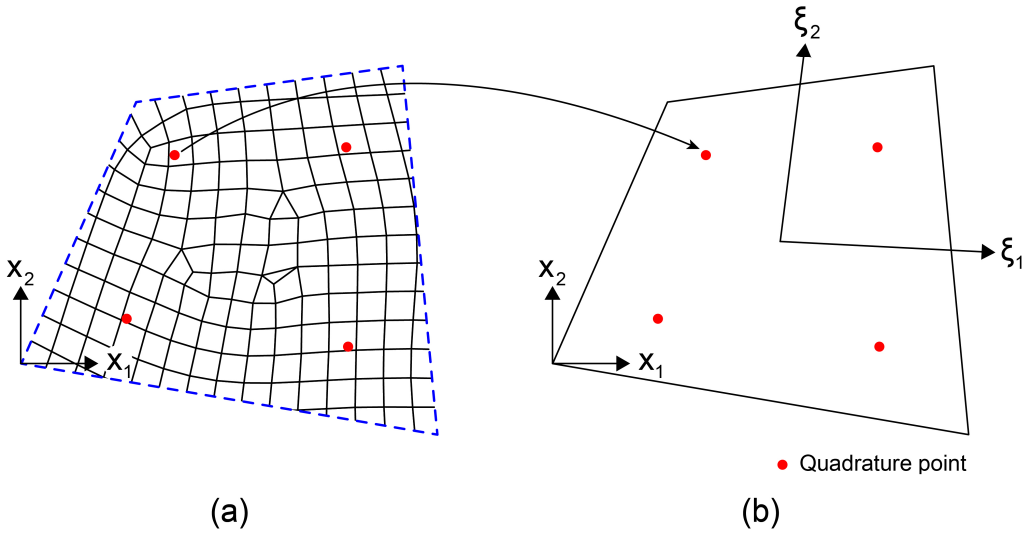


Fig. 3.2. Correction of a coarse finite element: (a) fine elements and (b) coarse element, where the red dots in the coarse element represent quadrature points.

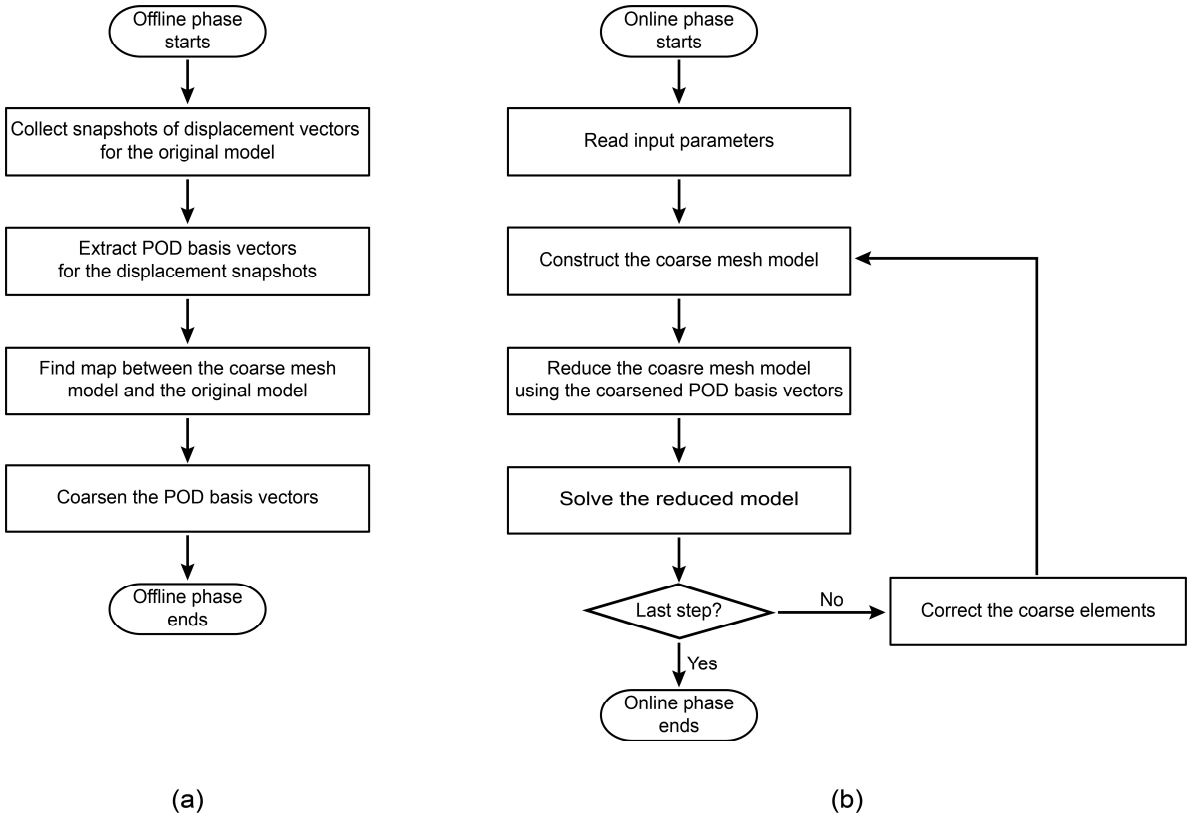


Fig. 3.3. Flowcharts of the proposed method: (a) offline and (b) online phases.

$$\tilde{\mathbf{M}} = \boldsymbol{\Psi}^T \hat{\mathbf{M}} \boldsymbol{\Psi}, \quad {}^t\tilde{\mathbf{R}} = \boldsymbol{\Psi}^T {}^t\hat{\mathbf{R}}. \tag{3.58}$$

The whole procedure of the proposed method is summarized in Fig. 3.3.

3.4. Numerical examples

We investigate the performance of the proposed method by means of several numerical examples of geometrically nonlinear and elastoplastic analyses. To solve nonlinear equations, the Newton-Raphson method is used at each load or time step. In addition, the implicit return mapping algorithm is used for the elastoplastic analysis at each quadrature point. For modeling the numerical examples, two-dimensional or three-dimensional solid finite elements are used.

The accuracy of the proposed method is compared to that of the corresponding standard coarse mesh model and the reduced models obtained by the POD-DEIM and POD-GappyPOD+E. The number of POD basis vectors is determined based on the criterion in Eq. (3.28) with $\epsilon = 99.999999\%$. In the finite element method, to compute a quantity corresponding to a nodal value, numerical integration should be performed on adjacent elements. Namely, the number of elements used in sparse sampling methods is typically greater than the number of sampling points. Therefore, we compare the proposed method to the DEIM and GappyPOD+E in the case of the approximately same number of elements used.

3.4.1. Two-dimensional column under a compressive load

A two-dimensional column subjected to a compressive load is considered as shown in Fig. 3.4. The length, width, and height of the column are $l = 1$, $w = 1$, and $h = 10$, respectively, Young's modulus and Poisson's ratio are $E = 10^6$ and $\nu = 0.3$, respectively, the plane stress condition is employed, and a compressive load $P_{\max}$ applied at point A is 4500. The column is modeled by four-node two-dimensional solid finite elements with a distorted mesh of 32×320 (10240 elements), where 32 elements are assigned along the column length.

To construct the reduced models, 84 snapshots of the displacement are collected, and the first 5 POD basis vectors are then selected. For the comparison with the DEIM, additional 84 snapshots of the internal force are collected. In this example, two cases of meshes are considered: mesh A (10 elements) and mesh B (42 elements). Fig. 3.5 shows the elements used to compute nonlinear terms in meshes A and B.

The load-displacement curves at point A for mesh A are described in Fig. 3.6. Although the proposed method uses 0.09% of the number of elements used in the original model, the deflection agrees very well with the reference solution. On the other hand, the DEIM fails to obtain the solution.

Fig. 3.7 presents the load-displacement curves at point A for mesh B. The relative errors in the displacement at each load step are listed in Table 3.1, and their minimum, maximum, and mean values are shown in Fig. 3.8. Here, the relative error in the displacement is defined by

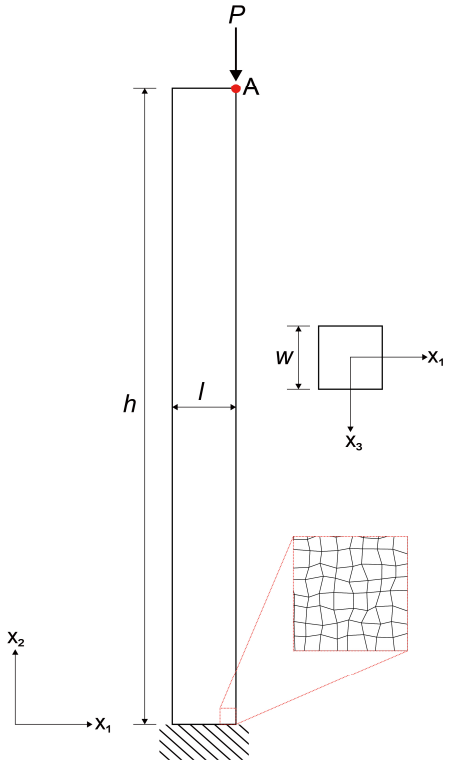


Fig. 3.4. Column subjected to a compressive load.

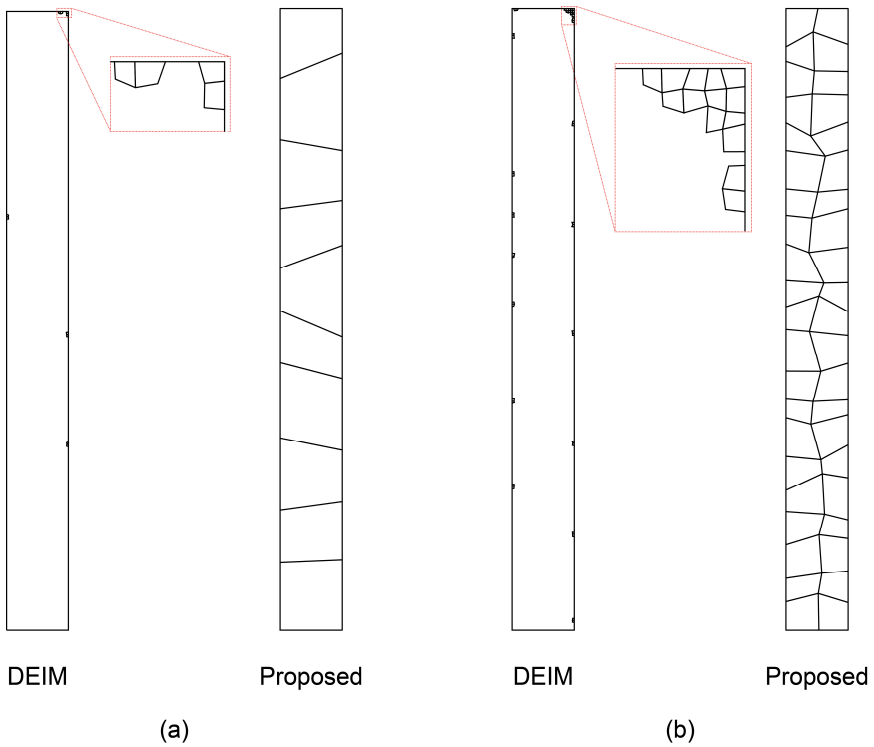


Fig. 3.5. Finite elements used to compute nonlinear terms for the two-dimensional column problem: (a) mesh A and (b) mesh B.

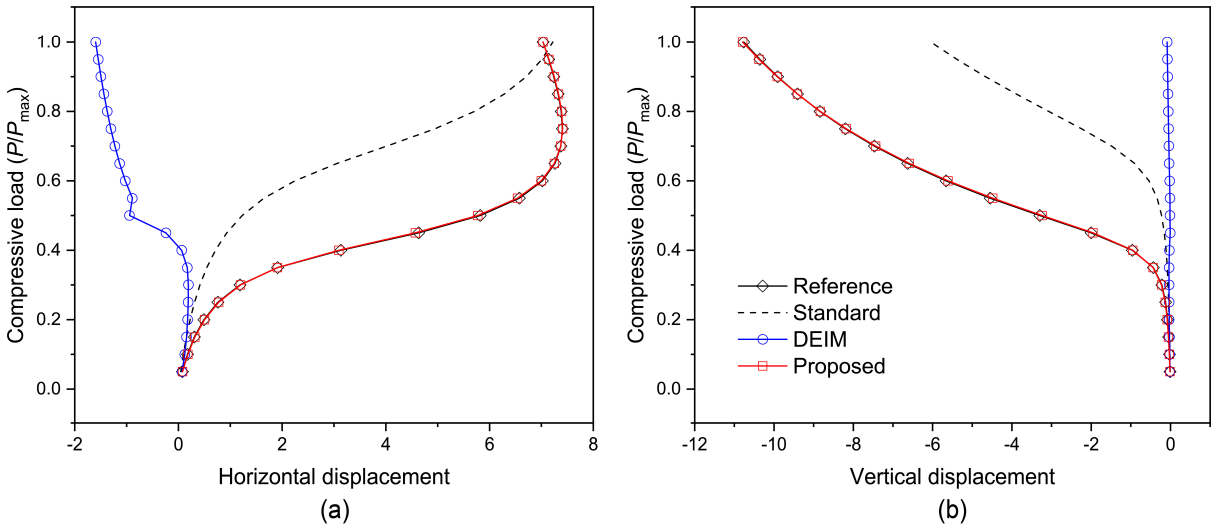


Fig. 3.6. Load-displacement curves at point A for the two-dimensional column problem with mesh A: (a) horizontal displacement and (b) vertical displacement.

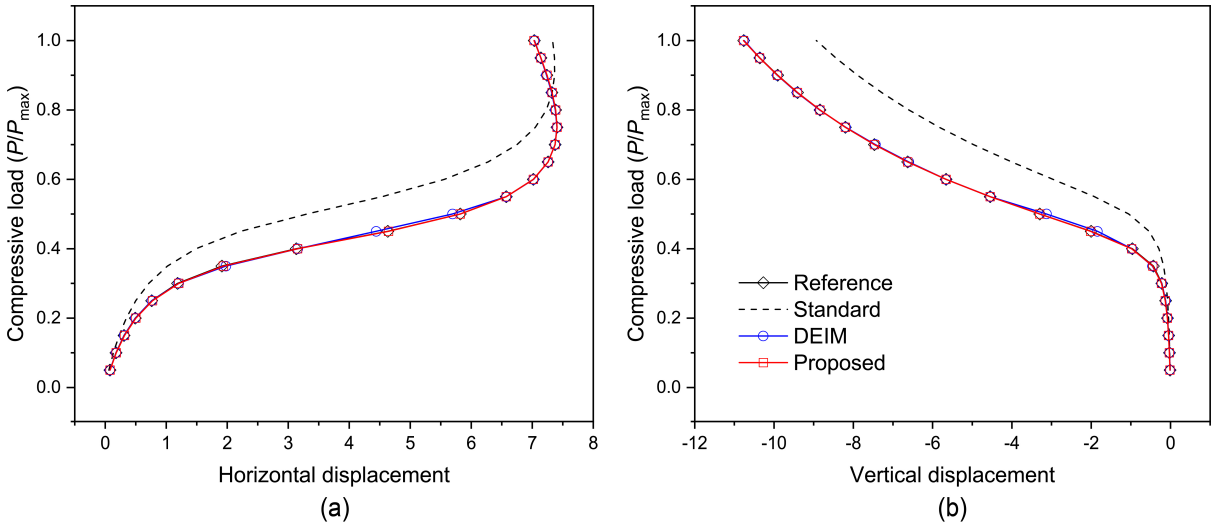


Fig. 3.7. Load-displacement curves at point A for the two-dimensional column problem with mesh B: (a) horizontal displacement and (b) vertical displacement.

$$\frac{\left\| {}^{t}\mathbf{U} - {}^{t}\mathbf{U}_{a} \right\|_2}{\left\| {}^{t}\mathbf{U} \right\|_2} \times 100, \tag{3.59}$$

where ${}^{t}\mathbf{U}_{a}$ denotes the approximate displacement at time t obtained by each method. The results also present that the proposed method outperforms the DEIM.

In addition, we compare the von Mises stress distributions at the final configuration of the proposed method to that of the standard coarse mesh models, see Fig. 3.9. The proposed method significantly improves the accuracy compared to that of the standard coarse mesh models.

Table 3.1. Relative errors in the displacement of each load step for the two-dimensional column problem with mesh B.

Load step	Relative error (%)	
	DEIM	Proposed
1	2.08	1.86
2	1.76	1.80
3	1.14	1.65
4	0.46	1.51
5	0.26	1.36
6	1.31	1.18
7	3.51	0.98
8	0.90	0.63
9	5.15	0.28
10	3.29	0.02
11	0.04	0.02
12	0.06	0.02
13	0.22	0.02
14	0.22	0.01
15	0.06	0.03
16	0.08	0.05
17	0.13	0.07
18	0.09	0.09
19	0.01	0.09
20	0.07	0.05

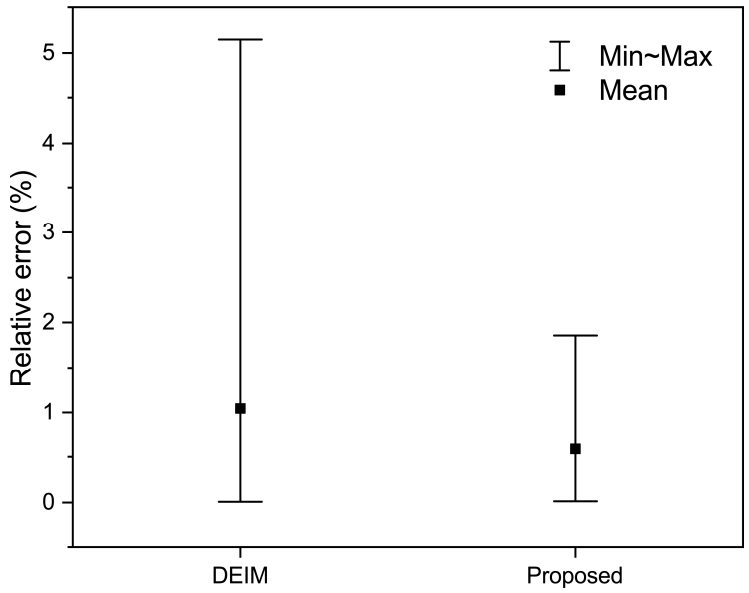


Fig. 3.8. Relative errors in the displacement for the two-dimensional column problem with mesh B.

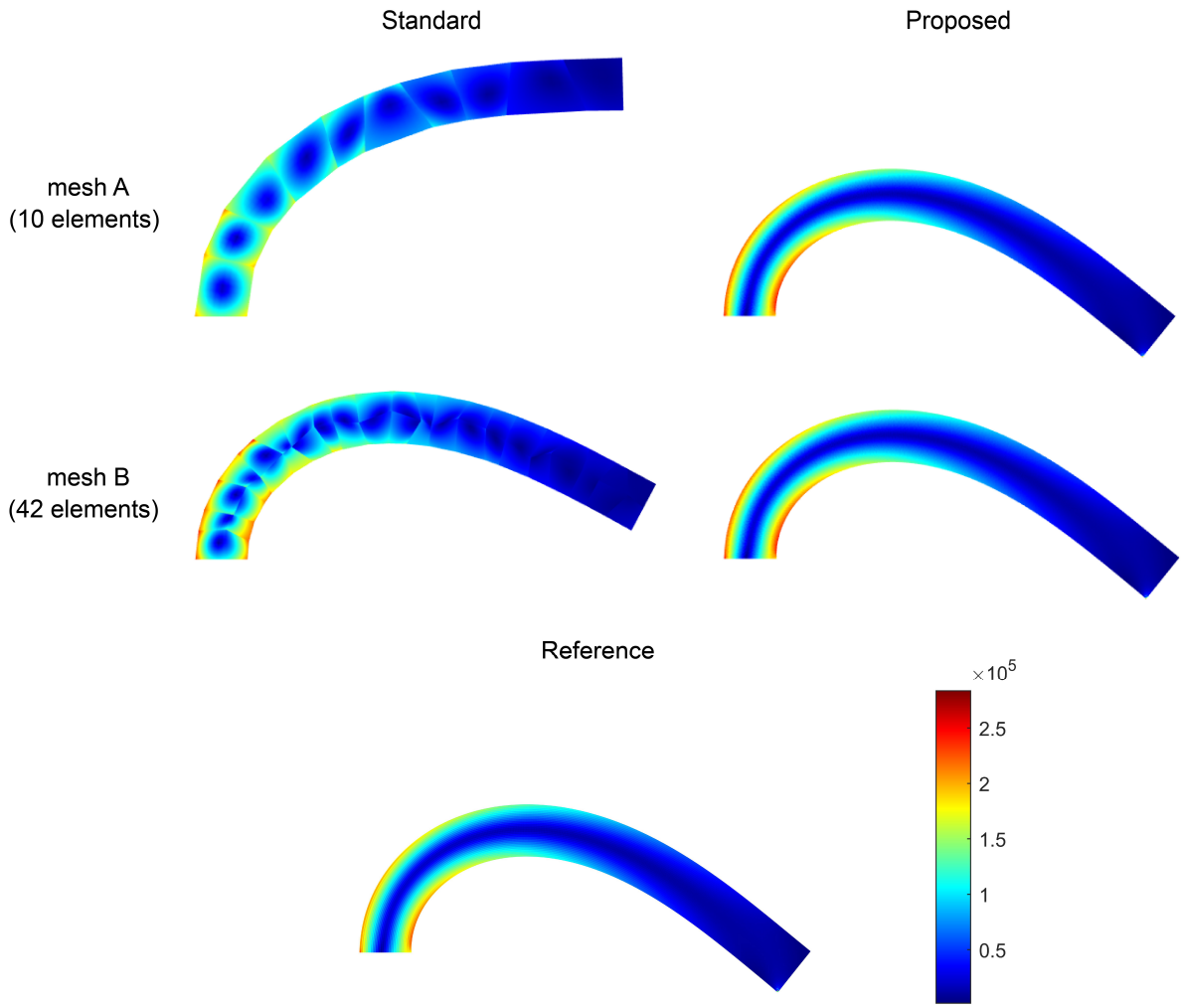


Fig. 3.9. Von Mises stress distribution of the two-dimensional column problem at the final configuration.

3.4.2. Three-dimensional column under a compressive load

We consider the column described in Section 3.4.1 again and now use three-dimensional hexahedral elements, see Fig. 3.10. For modeling the three-dimensional column, 81×99 (column cross section $\times$ column height) mesh is considered, and a compressive load $P_{\max} = 4500$ is applied at the blue edge.

We first collect the 81 snapshots of the displacement and extract the first 5 POD basis. In this example, two meshes A and B are used. The DEIM uses 16 and 48 elements for meshes A and B, respectively, and the proposed method uses 1×16 and 4×12 for meshes A and B, respectively.

The load-displacement curves at point A for mesh A are described in Fig. 3.11. As in section 3.4.1, the DEIM fails to obtain the converged solutions, while the proposed method successfully obtains the approximate solution. Fig. 3.12 presents the load-displacement curves at point A for mesh B, and Fig. 3.13 shows maximum, minimum, and mean values of the relative errors in the displacement at each load step. The results also present that the proposed method outperforms the DEIM for the three-dimensional problem.

In addition, we perform the geometrically nonlinear transient analysis using the Newmark time integration method. The design parameter of the column is the magnitude of the compressive load $P = \mu p(t)$ with $\mu \in [1125 \quad 4500]$, where $p(t)$ is shown in Fig. 3.14.

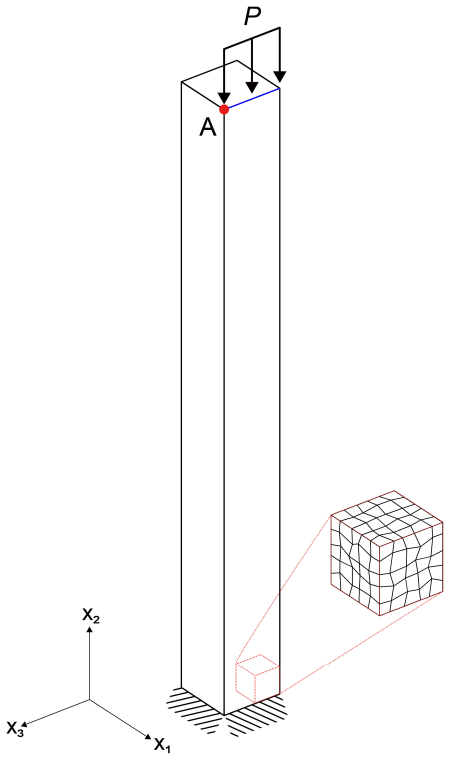


Fig. 3.10. Three-dimensional column subjected to a compressive load.

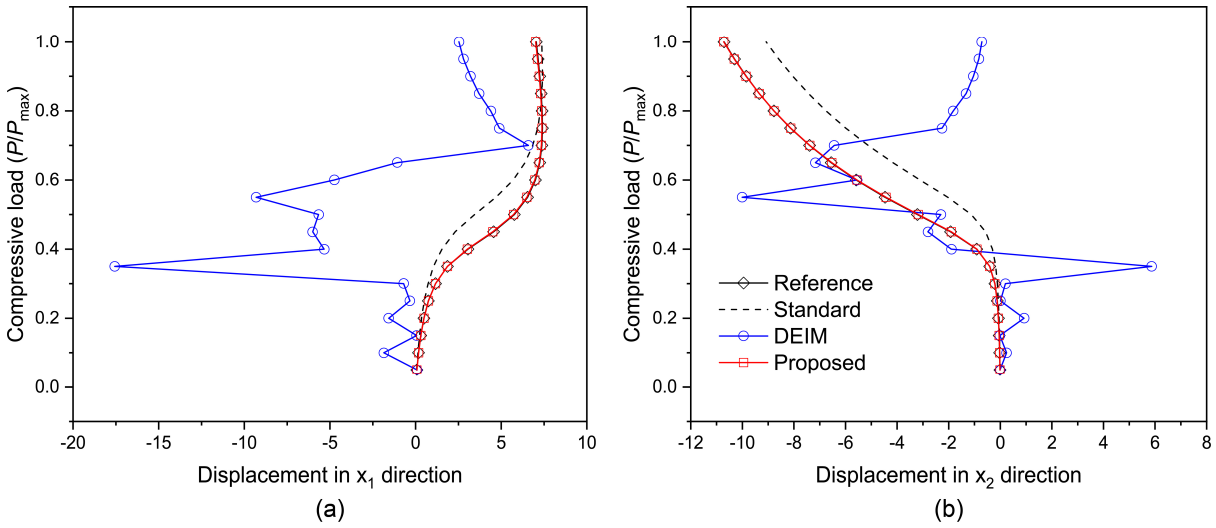


Fig. 3.11. Load-displacement curves at point A for the three-dimensional column problem with mesh A: (a) horizontal displacement and (b) vertical displacement.

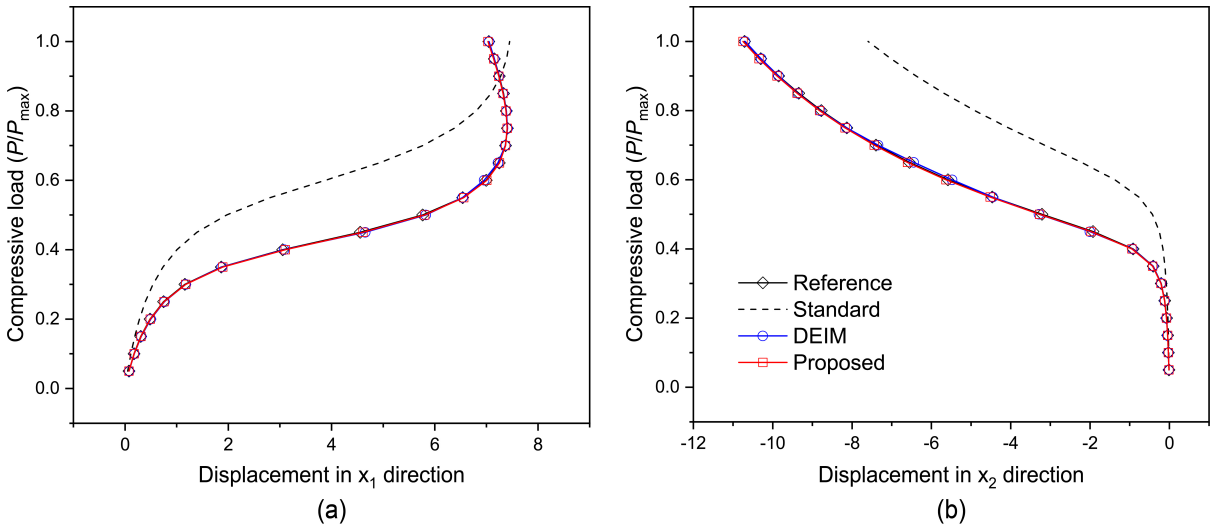


Fig. 3.12. Load-displacement curves at point A for the three-dimensional column problem with mesh B: (a) horizontal displacement and (b) vertical displacement.

We collect 710 snapshots of the displacement from two loading conditions $\mu = 0.25$ and $\mu = 1$ and extract the first 22 POD basis vectors. For evaluating nonlinear terms, the DEIM and proposed method use mesh B.

With a compressive load of $\mu = 2812.5$, the displacement in x_3 direction at point A is described in Fig. 3.15. Note that $\mu = 2812.5$ is not included in the snapshot parameters. While the DEIM fails to obtain the converged solution, the proposed method successfully provides the approximate solution.

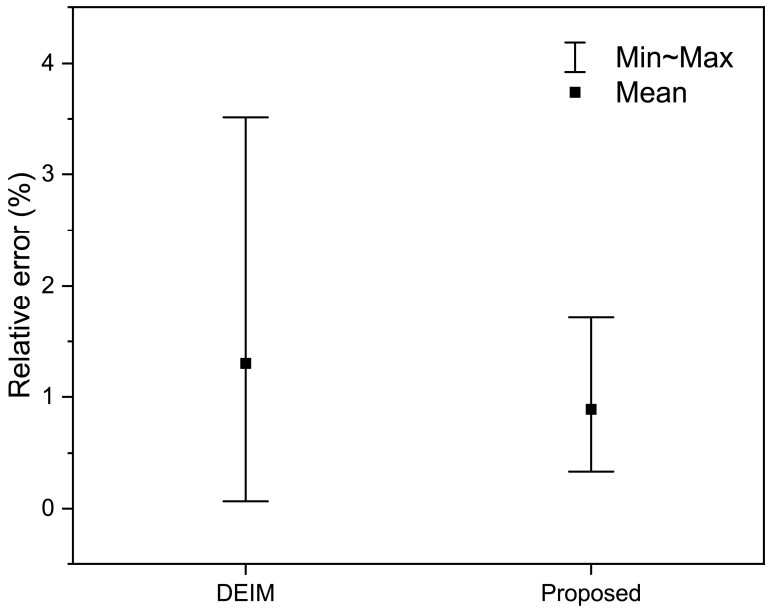


Fig. 3.13. Relative errors in the displacement for the three-dimensional column problem with mesh B.

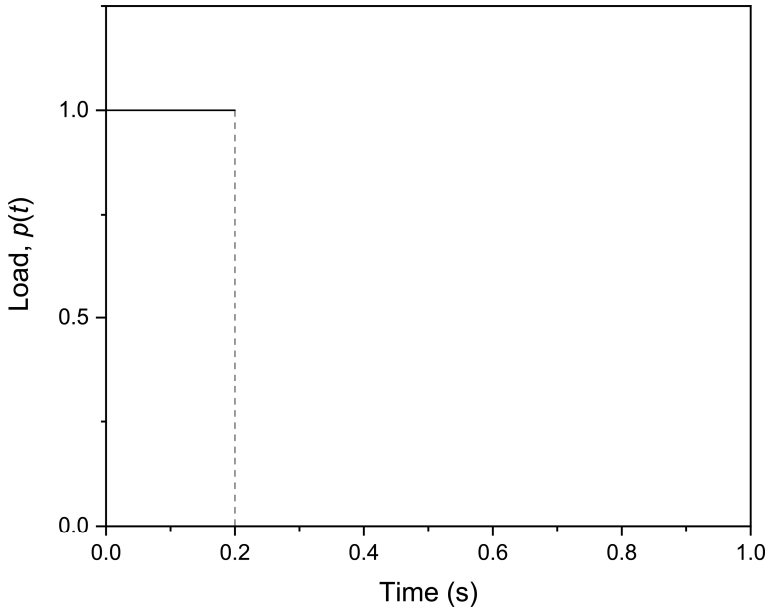


Fig. 3.14. Time history of a compressive load $p(t)$ for the three-dimensional column problem.

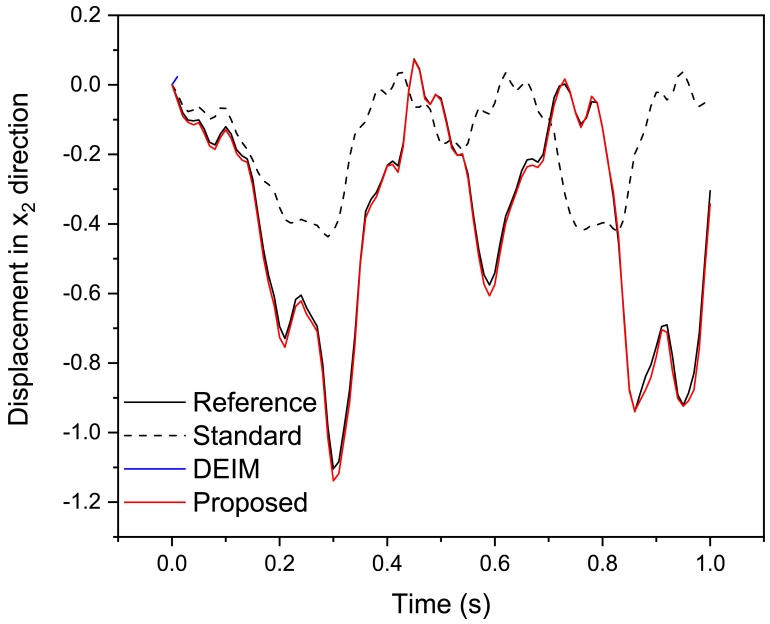


Fig. 3.15. Time history of the displacement in x_3 direction at point A for the three-dimensional column problem.

3.4.3. Rectangular plate with a hole

In this section, we investigate the performance of the proposed method in materially nonlinear analysis, especially elastic-plastic analysis. The von Mises plasticity with isotropic hardening and associated flow rule are employed. The length *l*, width *w*, radius *r*, and thickness of the plate are 10 mm, 2 mm, 1 mm, and 0.5 mm respectively, and only one-fourth of the plate is solved owing to symmetry, see Fig. 3.16. The plate is modeled by 1775 four-node two-dimensional solid finite elements. Young's modulus and Poisson's ratio of the plate are $E = 200\,GPa$, $\nu = 0.3$, respectively, and the loading profile is shown in Fig. 3.17.

We consider the initial yield stress $\sigma_{yv} = \mu_1 MPa$ with $\mu_1 \in [200 \quad 300]$ and the linear hardening modulus $H = \mu_2\, GPa$ with $\mu_2 \in [40 \quad 60]$ as input parameters. Snapshots are collected from the four cases of the upper and lower bounds of the parameters, and the first 29 POD basis vectors are then extracted by the 733 snapshots of the displacement.

The load-displacement curve at point A is described in Fig. 3.18, where two parameters are chosen as $\mu_1 = 250$ and $\mu_2 = 50$. Here, the finite elements used in this example are shown in Fig. 3.19. First, the DEIM and the proposed method respectively use 74 and 70 elements for computing nonlinear terms. The DEIM fails to obtain the converged solution, while the proposed method provides the approximate solution that agrees well with the reference solution. We use the GappyPOD+E algorithm to select additional sampling points and compare them to the proposed method with the same number of elements used. The result shows that the proposed method provides better accuracy compared to the DEIM and GappyPOD+E algorithm.

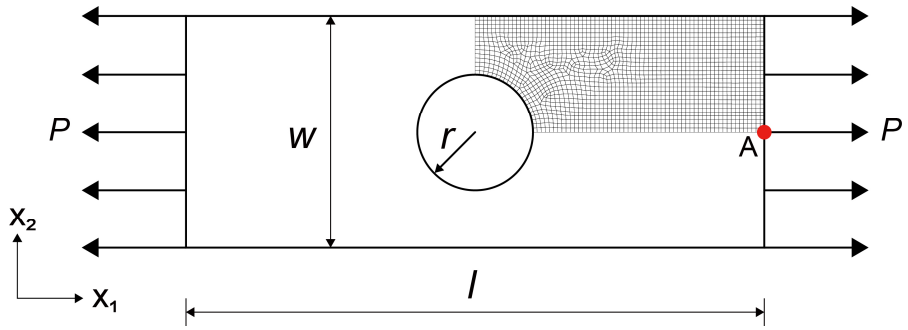


Fig. 3.16. Rectangular plate with a hole.

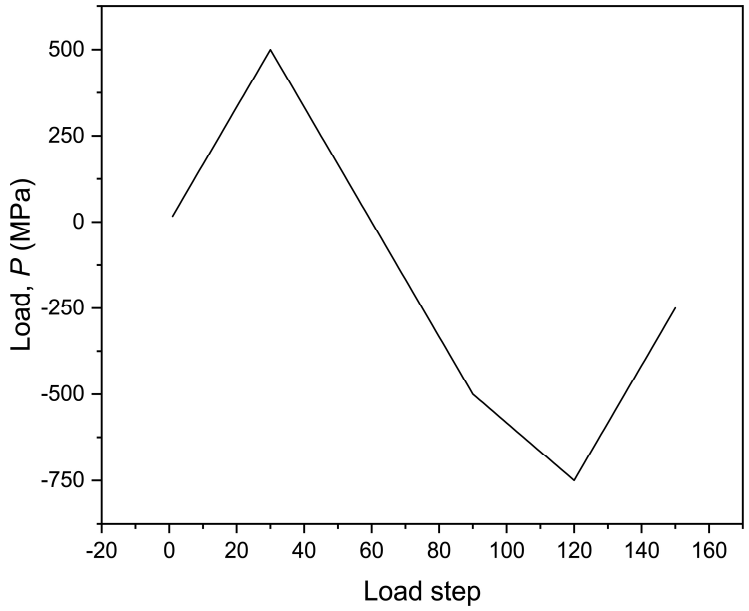


Fig. 3.17. Loading profile for the rectangular plate with a hole problem.

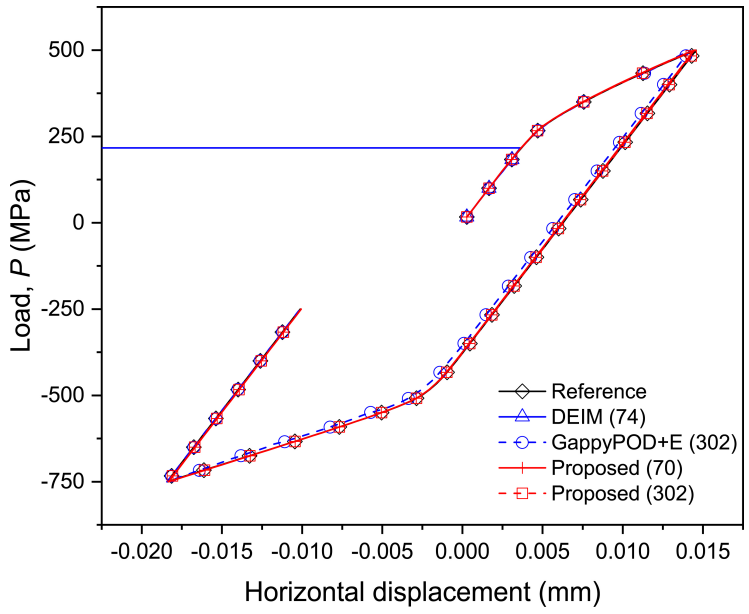


Fig. 3.18. Load-displacement curves at point A for the rectangular plate with a hole problem.

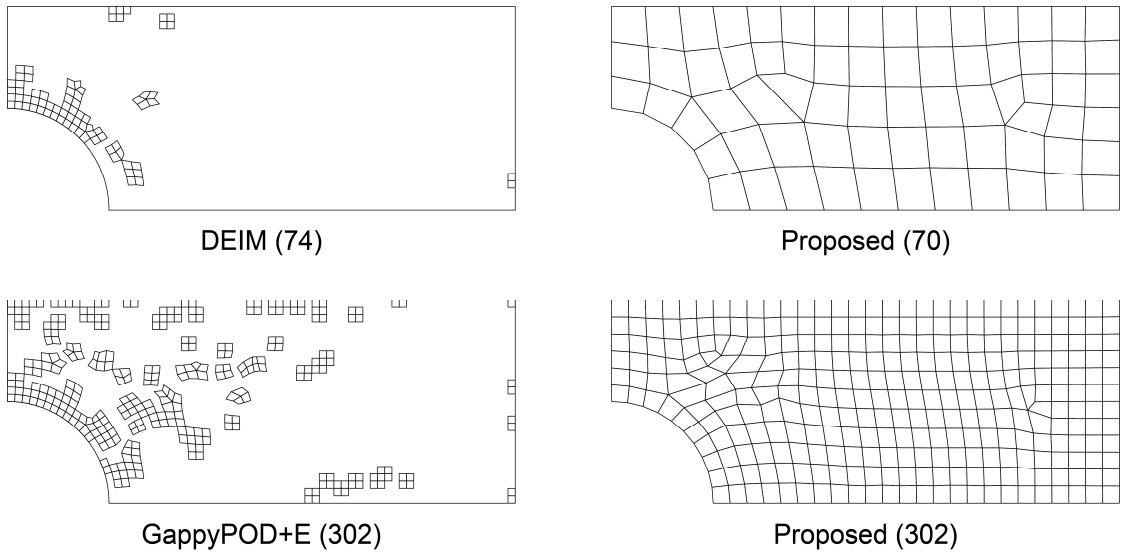


Fig. 3.19. Finite elements used to compute nonlinear terms for the rectangular plate with a hole problem. The numbers in the parenthesis denote the number of elements used.

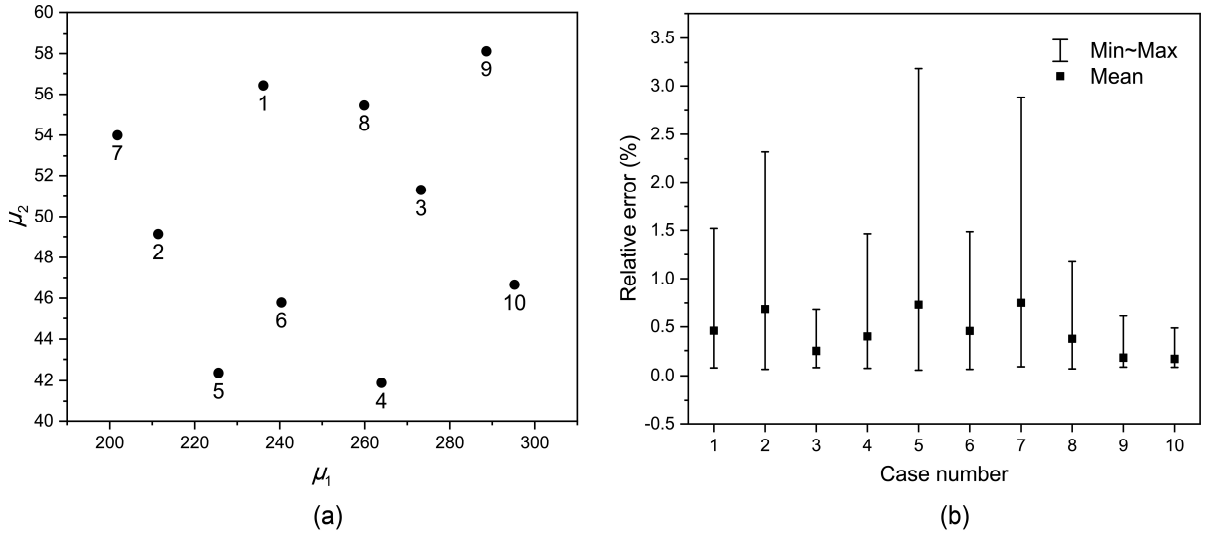


Fig. 3.20. Parametric variation: (a) chosen parameter cases and (b) relative errors in the displacement of each case.

We now generate the reduced models that provide approximate solutions over a range of parameters to conduct elastoplastic analysis. Ten cases of input parameters are used as shown in Fig. 3.20(a), where the number below a data point denotes the case number. Fig. 3.20(b) shows the relative errors in the displacement of each case. The result shows that the proposed method performs well with parametric variations.

3.4.4. Heterogeneous structure

A heterogeneous structure in Fig. 3.21 is considered and modeled by 25825 two-dimensional solid finite elements, where von Mises plasticity with isotropic hardening and associated flow rule are employed. The white circles

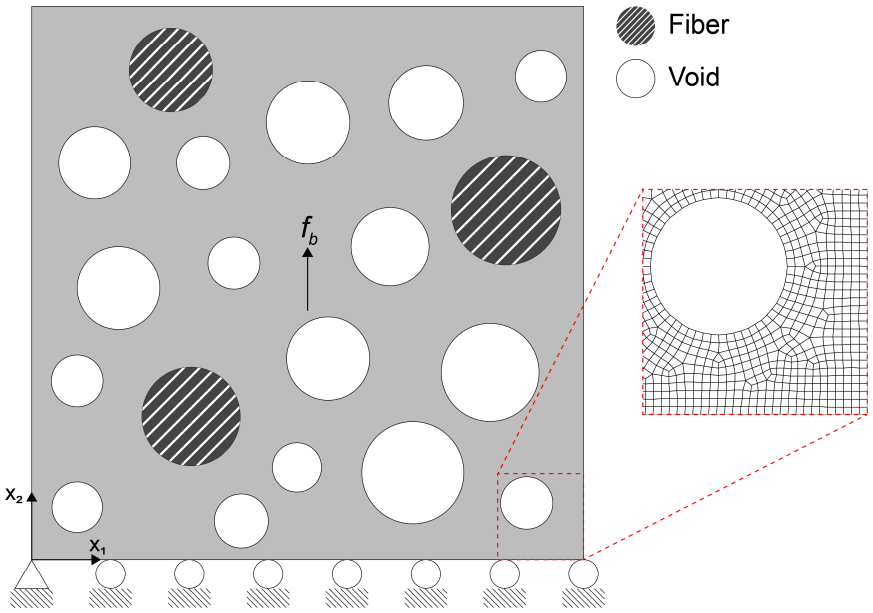


Fig. 3.21. Heterogeneous structure.

denote the void, and the hatched circles denote the stiff fiber with Young's modulus $E = 200\,\mu_1 GPa$, Poisson's ratio $\nu = 0.3$, initial yield stress $\sigma_{yv} = 200\mu_2 MPa$, and linear hardening modulus $H = 50\,\mu_3 GPa$. The rest of the structure is the soft matrix with $E = 70\mu_1\,GPa$, $\nu = 0.33$, $\sigma_{yv} = 70\mu_2 MPa$, and $H = 14\,\mu_2 GPa$. The structure is subjected to a body force $f_b = \sin^2(2\pi y)\,GPa$ in x_2 direction and simply supported along its bottom.

In this problem, only the GappyPOD+E algorithm with 826 and 1644 elements is used to compare with the proposed method because DEIM fails to obtain the converged solution. In the proposed method, 805 and 1629 elements are used, see Fig. 3.22. Three input parameters $\mu_1 \in [0.8 \quad 1.4]$, $\mu_2 \in [0.8 \quad 1.4]$, and $\mu_3 \in [0.8 \quad 1.4]$ are considered, and the first 37 POD basis vectors are obtained by the 553 snapshots of the displacement, where the snapshots are collected from the eight cases of the upper and lower bounds of the parameters.

Table 3.2 lists the twenty cases of the input parameters, and Fig. 3.23 shows the relative errors in the displacement of each case. Note that vertical scales are different in Fig. 3.23. The result shows that the proposed method performs well with parametric variations and outperforms the GappyPOD+E algorithm.

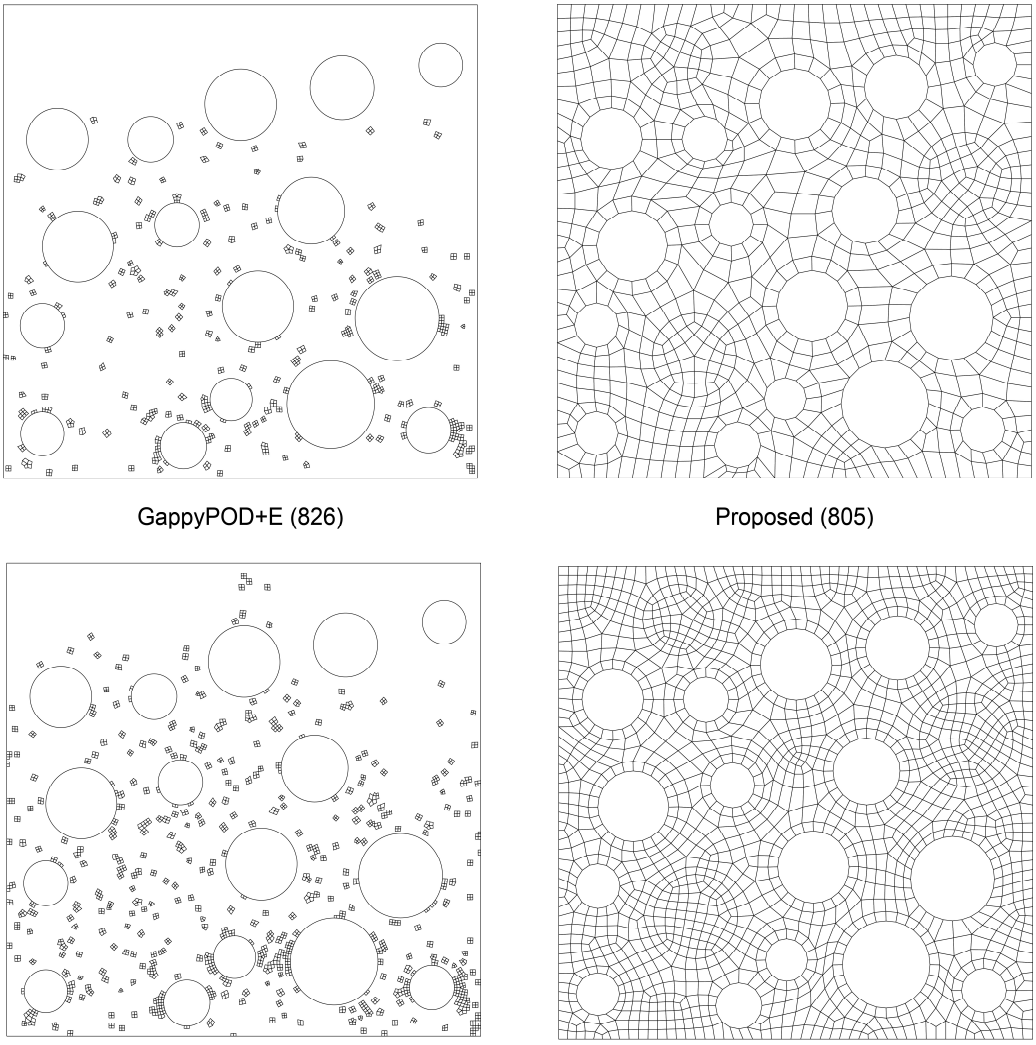


Fig. 3.22. Finite elements used to compute nonlinear terms for the heterogeneous structure problem. The numbers in the parenthesis denote the number of elements used.

Table 3.2. Parameter cases of the heterogeneous structure problem.

Parameter case	μ_1	μ_2	μ_3
1	1.24	1.03	0.91
2	1.31	0.88	1.08
3	1.09	1.19	1.14
4	1.03	0.89	0.94
5	1.36	1.06	1.01
6	0.93	1.01	1.27
7	0.97	1.12	1.11
8	0.87	1.08	0.86
9	1.32	1.27	1.02
10	1.06	1.30	1.07
11	0.92	1.24	1.17
12	1.28	1.39	0.80
13	0.84	0.94	1.21
14	0.82	1.36	1.34
15	1.13	0.85	0.85
16	1.21	0.98	1.31
17	1.13	0.81	1.37
18	1.38	1.19	1.23
19	1.16	1.32	1.30
20	0.99	1.15	0.95

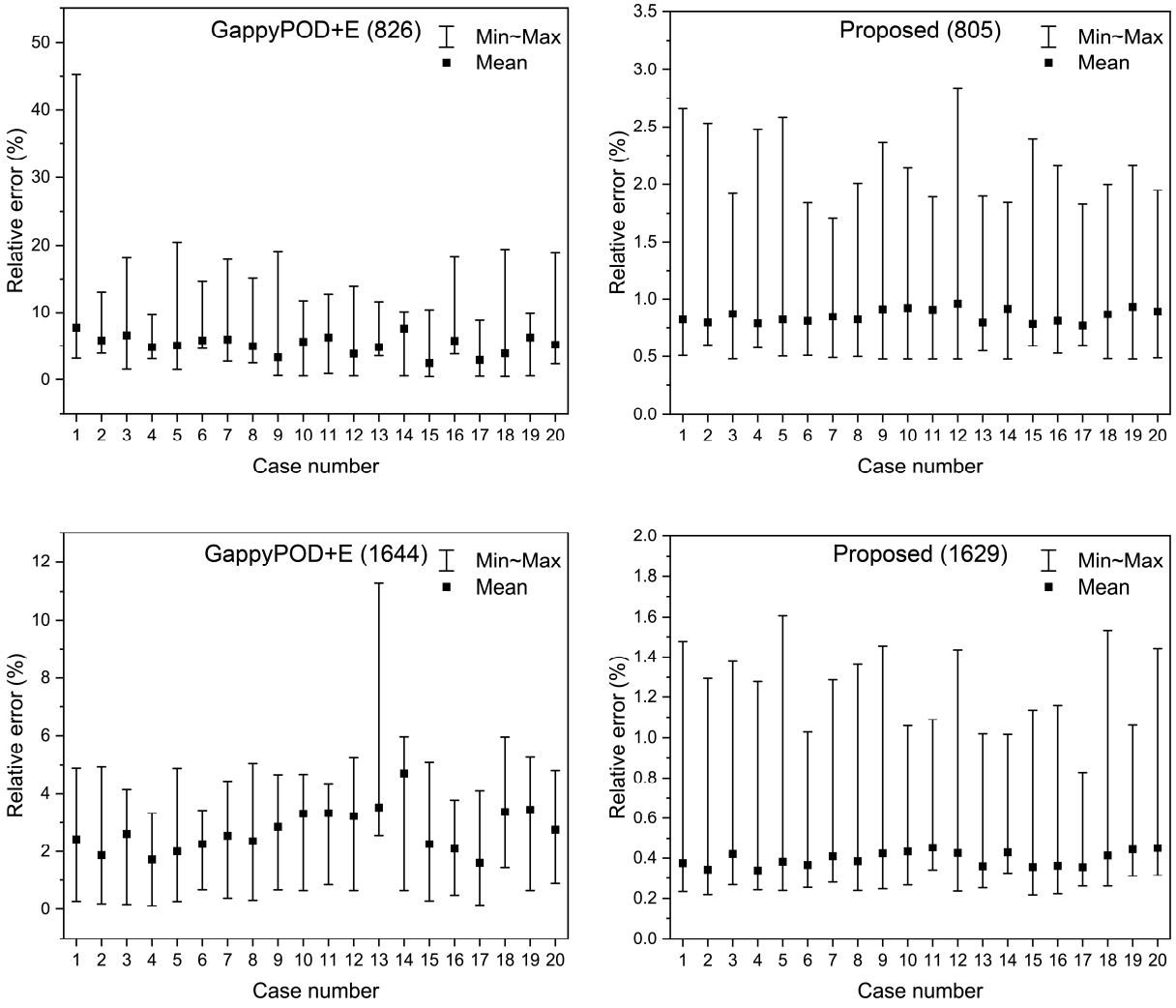


Fig. 3.23. Relative errors in the displacement of each case. The number in the parenthesis denotes the number of elements used.

3.5. Concluding remarks

We proposed the coarse mesh projection method for nonlinear model reduction. The nonlinear terms were computed on a coarse mesh model corresponding to the domain considered in the original model. Except for snapshots of nonlinear terms, only snapshots of displacement vectors were employed to extract POD basis vectors. The reduction of the coarse mesh model was performed by the Galerkin projection using the coarsened POD basis vectors. In order to evaluate the performance of the proposed method, geometrically and materially nonlinear problems were solved in comparison with the DEIM and GappyPOD+E. The proposed method outperformed these two methods when the approximately same number of elements were used for computing nonlinear terms.

Chapter 4. Conclusions

The objective of this dissertation was to present novel model reduction methods for linear and nonlinear analyses of structures. The projection-based model reductions have been widely used for solving engineering problems efficiently. They have a rigorous framework that projects the original model onto a low-dimensional subspace. However, for large-scale problems, the improvement in the efficiency of the reduction methods is still required.

In Chapter 2, a load balancing algorithm for the parallel automated multilevel substructuring (PAMLS) for efficient linear model reduction in structural dynamics has been proposed. Improving the computational efficiency of the original PAMLS has been achieved by consisting of two types of granularity: coarse- and fine-grained parallel algorithms. Without mesh repartitioning, the proposed method speeds up the original PAMLS, regardless of the DOFs of the original model, the number of substructures, and cutoff frequencies.

The concept of the proposed algorithm could easily extend to the parallelization of other multilevel CMS methods [47,87–89]. Note that, in this dissertation, the proposed algorithm was implemented on a shared memory computer. Future efforts to develop a parallel algorithm for hybrid distributed-shared memory systems would be valuable. Hybrid systems have many computation nodes which are shared memory computers; the computation nodes communicate with each other by sending and receiving messages. As with the shared memory parallel algorithm, it is desirable to maximize the local computation and minimize the communication between computation nodes because parallel performance is primarily affected by the idle time at the synchronization points. One possible approach is to use the two types of granularity in the proposed algorithm. In coarse-grained parallelism, the computational load of tasks is typically greater than in fine-grained parallelism. Therefore, the coarse-grained and fine-grained parallel algorithms would be suitable for distributed and shared memory parallel strategies, respectively.

Furthermore, research on the model reduction of nonlinear problems is still going on, although the model reduction of linear problems is quite mature. Chapter 3 was devoted to developing an efficient nonlinear model reduction method using a coarse mesh, named coarse mesh projection. The proposed method provided more accurate solutions compared to the DEIM and GappyPOD+E in geometrically nonlinear and elastoplastic analyses. The concept of the proposed method could easily extend to apply other standard finite elements. In particular, future efforts to establish a theoretical foundation of the proposed method would be valuable. For example, a priori and posterior error bounds are useful to guarantee reliable solutions.

Appendix A. Effect of a cutoff level on the parallel automated multilevel substructuring method

We investigate the effect of the following cutoff level

$$L_c = \min\{L_t + i, L_s\} \quad \text{with} \quad L_t = \min\{k \in \mathbb{Z} : \log_2 N_t \le k\} \quad \text{for} \quad i \in \mathbb{N}, \tag{A.1}$$

where N_t and L_s are the number of threads used and the maximum level of substructures, respectively, and $\mathbb{N}$ and $\mathbb{Z}$ are the sets of all natural numbers and integers, respectively.

The serial and parallel algorithms are implemented in Fortran, where Intel Fortran Compiler 19.1.0.166 with OpenMP is used. The mesh partitioning is achieved by METIS [68], an open-source package for unstructured graph partitioning, and the eigenvalue problems are solved using ARPACK [53], an open-source package based on the Arnoldi/Lanczos process. All numerical examples are tested on CentOS 7 with two 8-core Intel Xeon CPUs (E5-2667 v4 at 3.2 GHz) and with 128 GB of memory. For evaluating the performance, generalized eigenvalue problems for three finite element models are solved on 2 to 16 threads in the original PAMLS method and the proposed algorithm. Table A.1 lists the finite element models used.

The performance is evaluated by the wall clock time required for the transformation procedure, which spends a majority of the elapsed time in the AMLS method. The set of wall clock times is defined by

$$T(j) = \{i \in \mathbb{N} : t_i(j)\}, \tag{A.2}$$

where the subscript i in $t_i(j)$ denotes the additional level of $L_c = \min\{L_t + i, L_s\}$ in Eq. (A.1), and the number in the parenthesis of $t_i(\cdot)$ is the number of threads used.

The normalized wall clock time for the additional level *i* is defined by

$$\overline{T}_i = \frac{1}{N_t - 1}\sum_{j=2}^{N_t} \overline{t}_i(j) \quad \text{with} \quad \overline{t}_i(j) = \frac{\min\{T(j)\}}{t_i(j)} \in [0 \quad 1], \tag{A.3}$$

and $\tilde{T}_i$ is then defined by the mean of $\overline{T}_i$ for the finite element models in Table A.1. Note that $\tilde{T}_i$ represents the normalized parallel performance for the additional level.

The normalized parallel performance for the additional level $\tilde{T}_i$ is shown in Table. A.2 and Fig. A.1. Based on the result, we employ the following cutoff level L_c:

$$L_c = \min\{L_t + 5, L_s\}. \tag{A.4}$$

Table A.1. Finite element models used to investigate the effect of the cutoff level.

Model	DOFs	Number of nonzero entries in the upper triangular part	
		Mass matrix	Stiffness matrix
A	209118	1655714	3093933
B	1054866	8147602	23386865
C	4244496	15124689	43345801

Table A.2. Normalized parallel performance for the additional level.

Additional level	Normalized parallel performance
0	0.8234
1	0.9036
2	0.9542
3	0.9792
4	0.9718
5	0.9989
6	0.9733
7	0.9622
8	0.9826
9	0.9954
10	0.9980

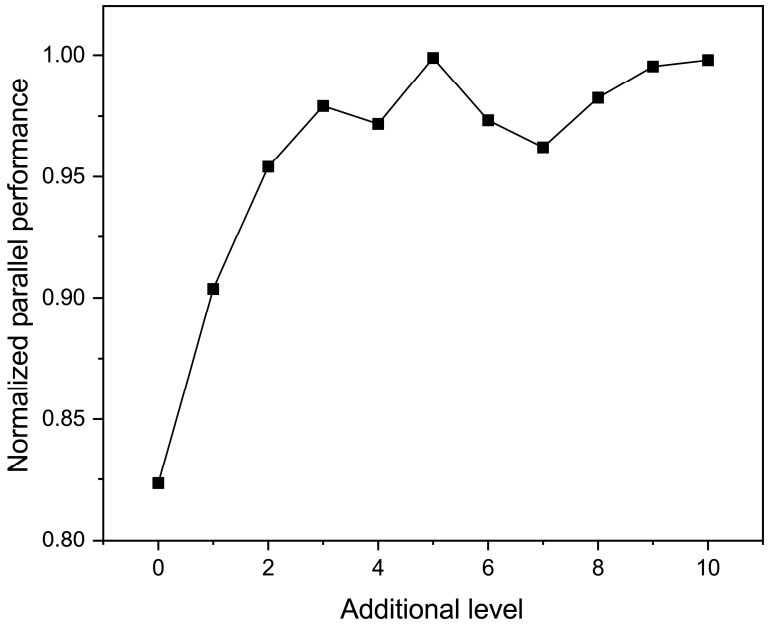


Fig. A.1. Normalized parallel performance for the additional level.

Appendix B. Effect of mesh partitioning on the parallel automated multilevel substructuring method

In the parallel AMLS method, more threads become idle as more substructures are transformed. In addition, the computational cost for the transformation of an interface substructure is typically greater than that of its descendant substructure, even if their DOFs are the same. The reason is that matrices for updated substructures generally have more nonzero entries. In other words, the computational cost for the transformation of a shared substructure is typically greater than that of a distributed substructure. Therefore, the smaller number of DOFs for interface substructures (i.e. shared substructures) provides better parallel scalability.

For example, a clamped-clamped beam described in Fig. B.1 is considered, which is taken from Ref. [49]. The beam is modeled by three-dimensional hexahedral elements with three different meshes, where the number of elements in each mesh is the same. Table B.1 lists the information about each of the meshes that are partitioned into 2047 substructures on 10 levels. Table B.2 and Fig B.2 show the mean substructure sizes of each level for the finite element models when partitioning the meshes by using METIS [68]. It is observed that the number of DOFs for interface substructures (i.e. shared substructures) is greater as the number of elements in the cross section increases.

The parallel performance of the original PAMLS method is consistently improved through the proposed algorithm although the parallel scalability is affected by the number of DOFs for shared substructures, see Fig. B.3. This partitioning issue is not addressed in this dissertation because we have focused on the load balancing for the PAMLS method in a given mesh partitioning.

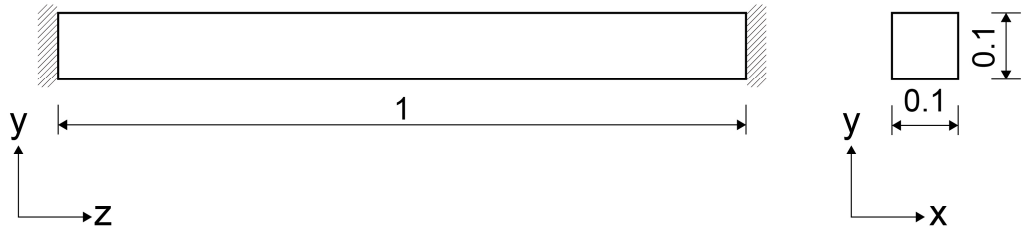


Fig. B.1. Clamped-clamped beam problem.

Table B.1. Finite element models for the beam problem.

Number of elements (cross section × length)	DOFs	Number of nonzero entries in the upper triangular part	
		Mass matrix	Stiffness matrix
163840 (16 × 16 × 640)	554013	7173879	13639251
163840 (32 × 32 × 160)	519453	6963639	13078803
163840 (64 × 64 × 40)	494325	6672615	12429795

Table B.2. Mean substructure sizes of each level for the beam problem.

level	Number of substructures	Mean substructure sizes (DOFs)		
		16 × 16 × 640 mesh	32 × 32 × 160 mesh	64 × 64 × 40 mesh
0	1	867	3267	7995
1	2	867	3267	3936
2	4	867	3341	3072
3	8	867	1940	1920
4	16	867	923	939
5	32	867	771	721
6	64	462	428	439
7	128	217	210	208
8	256	193	168	158
9	512	96	88	90
10	1024	337	298	290

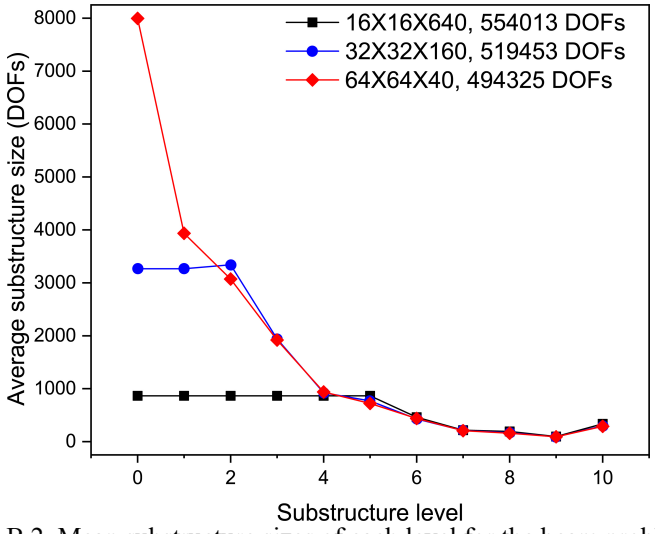


Fig. B.2. Mean substructure sizes of each level for the beam problem.

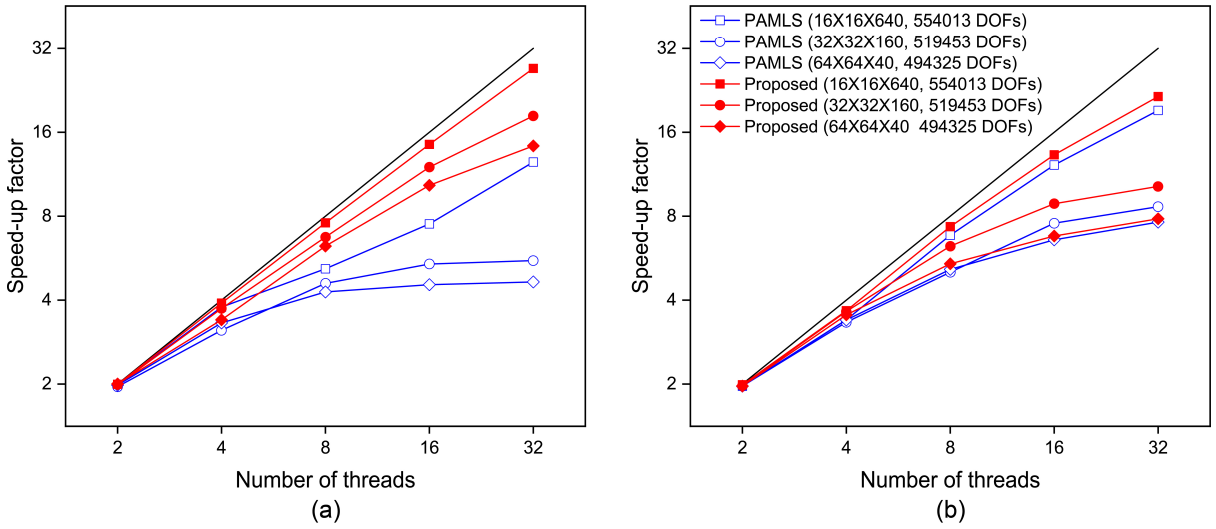


Fig. B.3. Speed-up factors for the beam problem: (a) transformation procedure and (b) implicit back transformation procedure.

Bibliography

[1] K.J. Bathe, Finite element procedures, 2nd ed., Klaus-Jürgen Bathe, Watertown (MA), 2014.

[2] T.J.R. Hughes, The finite element method: linear static and dynamic finite element analysis, Courier Corporation, 2012.

[3] P. Benner, S. Gugercin, K. Willcox, A survey of projection-based model reduction methods for parametric dynamical systems, SIAM Rev. 57 (2015) 483–531.

[4] J.K. Bennighof, C. Kim, An adaptive multi-level substructuring method for efficient modeling of complex structures, in: 33rd Struct. Struct. Dyn. Mater. Conf., American Institute of Aeronautics and Astronautics, 1992.

[5] M.F. Kaplan, Implementation of automated multilevel substructuring for frequency response analysis of structures [dissertation], Austin (TX): University of Texas at Austin, 2001.

[6] J.K. Bennighof, R.B. Lehoucq, An automated multilevel substructuring method for eigenspace computation in linear elastodynamics, SIAM J. Sci. Comput. 25 (2004) 2084–2106.

[7] M. Kim, An efficient eigensolution method and its implementation for large structural systems [dissertation], Austin (TX): University of Texas at Austin, 2004.

[8] K. Elssel, H. Voss, Multilevel extended algorithms in structural dynamics on parallel computers, Adv. Parallel Comput. 13 (2004) 243–250.

[9] R. Everson, L. Sirovich, Karhunen – Lòeve procedure for gappy data, J. Opt. Soc. Am. A. 12 (1995) 1657.

[10] S. Chaturantabut, D.C. Sorensen, Nonlinear model reduction via discrete empirical interpolation, SIAM J. Sci. Comput. 32 (2010) 2737–2764.

[11] F. Ghavamian, P. Tiso, A. Simone, POD–DEIM model order reduction for strain softening viscoplasticity, Comput. Methods Appl. Mech. Eng. 317 (2017) 458–479.

[12] B. Peherstorfer, Z. Drmač, S. Gugercin, Stability of discrete empirical interpolation and gappy proper orthogonal decomposition with randomized and deterministic sampling points, SIAM J. Sci. Comput. 42 (2020) A2837–A2864.

[13] P. Tiso, D.J. Rixen, Discrete empirical interpolation method for finite element structural dynamics, in: Topics in Nonlinear Dynamics, vol. 1, Springer, 2013: pp. 203–212.

[14] C. Hyun, P.S. Lee, A load balancing algorithm for the parallel automated multilevel substructuring method, Comput. Struct. 257 (2021) 106649.

[15] W.C. Hurty, Dynamic analysis of structural systems using component modes, AIAA J. 3 (1965) 678–685.

[16] R.R. Craig, M.C.C. Bampton, Coupling of substructures for dynamics analyses, AIAA J. 6 (1968) 1313.

[17] R.M. Hintz, Analytical methods in component modal synthesis, AIAA J. 13 (1975) 1007–1016.

[18] D.J. Rixen, A dual Craig-Bampton method for dynamic substructuring, J. Comput. Appl. Math. 168 (2004) 383–391.

[19] J.H. Kim, J. Kim, P.S. Lee, Improving the accuracy of the dual Craig-Bampton method, Comput. Struct. 191 (2017) 22–32.

[20] S.H. Boo, J.H. Kim, P.S. Lee, Towards improving the enhanced Craig-Bampton method, Comput. Struct. 196 (2018) 63–75.

[21] F.J. Herrada, J. García-Martínez, A. Fraile, L.K.H. Hermanns, F.J. Montáns, A method for performing efficient parametric dynamic analyses in large finite element models undergoing structural modifications, Eng. Struct. 131 (2017) 625–638.

[22] J. García-Martínez, F.J. Herrada, L.K.H. Hermanns, A. Fraile, F.J. Montáns, Accelerating parametric studies in computational dynamics: Selective modal re-orthogonalization versus model order reduction methods, Adv. Eng. Softw. 108 (2017) 24–36.

[23] E. Armentani, V. Giannella, R. Citarella, A. Parente, M. Pirelli, Substructuring of a petrol engine: dynamic characterization and experimental validation, Appl. Sci. 9 (2019).

[24] D. Siano, R.G. Citarella, E. Armentani, Simulation of the vibrational behaviour of a multi-cylinder engine, Int. J. Veh. Noise Vib. 1 (2018) 1.

[25] M. Gibanica, T.J.S. Abrahamsson, D.J. Rixen, Multifidelity component interface reduction and modal truncation augmentation, Int. J. Numer. Methods Eng. 120 (2019) 105–124.

[26] M. Gibanica, T.J.S. Abrahamsson, D.J. Rixen, A reduced interface component mode synthesis method

using coarse meshes, Procedia Eng. 199 (2017) 348–353.

[27] G. Battiato, C.M. Firrone, T.M. Berruti, B.I. Epureanu, Reduction and coupling of substructures via Gram–Schmidt Interface modes, Comput. Methods Appl. Mech. Eng. 336 (2018) 187–212.

[28] G. Battiato, C.M. Firrone, A modal based reduction technique for wide loose interfaces and application to a turbine stator, Mech. Syst. Signal Process. 139 (2020) 106415.

[29] W. Tian, S. Weng, Y. Xia, H. Zhu, F. Gao, Y. Sun, J. Li, An iterative reduced-order substructuring approach to the calculation of eigensolutions and eigensensitivities, Mech. Syst. Signal Process. 130 (2019) 361–377.

[30] S. Weng, H.P. Zhu, Y. Xia, L. Mao, Damage detection using the eigenparameter decomposition of substructural flexibility matrix, Mech. Syst. Signal Process. 34 (2013) 19–38.

[31] M.E.M. El-Sayed, C.K. Hsiung, Parallel finite element computation with separate substructures, Comput. Struct. 36 (1990) 261–265.

[32] K. Rothe, H. Voss, A fully parallel condensation method for generalized eigenvalue problems on distributed memory computers, Parallel Comput. 21 (1995) 907–921.

[33] W. Mackens, H. Voss, General masters in parallel condensation of eigenvalue problems, Parallel Comput. 25 (1999) 893–903.

[34] B.C.P. Heng, R.I. Mackie, Parallel modal analysis with concurrent distributed objects, Comput. Struct. 88 (2010) 1444–1458.

[35] A. George, Nested dissection of a regular finite element mesh, SIAM J. Numer. Anal. 10 (1973) 345–363.

[36] C. Lanczos, An iteration method for the solution of the eigenvalue problem of linear differential and integral operators, J. Res. Natl. Bur. Stand. 45 (1950) 255.

[37] R. Grimes, J. Lewis, H. Simon, A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems, SIAM J. Matrix Anal. Appl. 15 (1994) 228–272.

[38] C. Bekas, Y. Saad, Computation of smallest eigenvalues using spectral Schur complements, SIAM J. Sci. Comput. 27 (2005) 458–481.

[39] A. Kropp, D. Heiserer, Efficient broadband vibro-acoustic analysis of passenger car bodies using an FE-

based component mode synthesis approach, J. Comput. Acoust. 11 (2003) 139–157.

[40] W. Rachowicz, A. Zdunek, Automated multi-level substructuring (AMLS) for electromagnetics, Comput. Methods Appl. Mech. Eng. 198 (2009) 1224–1234.

[41] C. Yang, W. Gao, Z. Bai, X.S. Li, L.Q. Lee, P. Husbands, E.G. Ng, Algebraic sub-structuring for electromagnetic applications, in: Springer, 2006: pp. 364–373.

[42] Q. Song, P. Chen, S. Sun, An exact reanalysis algorithm for local non-topological high-rank structural modifications in finite element analysis, Comput. Struct. 143 (2014) 60–72.

[43] Y.S. Yang, S.H. Hsieh, T.J. Hsieh, Improving parallel substructuring efficiency by using a multilevel approach, J. Comput. Civ. Eng. 26 (2011) 457–464.

[44] K. Schloegel, G. Karypis, V. Kumar, Parallel static and dynamic multi-constraint graph partitioning, Concurr. Comput. Pract. Exp. 14 (2002) 219–240.

[45] Y.S. Yang, S.H. Hsieh, Iterative mesh partitioning optimization for parallel nonlinear dynamic finite element analysis with direct substructuring, Comput. Mech. 28 (2002) 456–468.

[46] Y. Escaig, G. Touzot, M. Vayssade, Parallelization of a multilevel domain decomposition method, Comput. Syst. Eng. 5 (1994) 253–263.

[47] J. Yin, H. Voss, P. Chen, Improving eigenpairs of automated multilevel substructuring with subspace iterations, Comput. Struct. 119 (2013) 115–124.

[48] K.J. Bathe, E.L. Wilson, Solution methods for eigenvalue problems in structural mechanics, Int. J. Numer. Methods Eng. 6 (1973) 213–226.

[49] K.J. Bathe, The subspace iteration method - Revisited, Comput. Struct. 126 (2013) 177–183.

[50] K.T. Kim, K.J. Bathe, The Bathe subspace iteration method enriched by turning vectors, Comput. Struct. 186 (2017) 11–21.

[51] B.N. Parlett, D.S. Scott, The Lanczos algorithm with selective orthogonalization, Math. Comput. 33 (1979) 217.

[52] H.G. Matthies, A subspace Lanczos method for the generalized symmetric eigenproblem, Comput. Struct. 21 (1985) 319–325.

[53] R.B. Lehoucq, D.C. Sorensen, C. Yang, ARPACK Users' Guide, Society for Industrial and Applied Mathematics, 1998.

[54] R.J. Guyan, Reduction of stiffness and mass matrices, AIAA J. 3 (1965) 380–380.

[55] B. Irons, Structural eigenvalue problems - elimination of unwanted variables, AIAA J. 3 (1965) 961–962.

[56] C.A. Miller, Dynamic reduction of structural models, J. Struct. Div. 106 (1980) 2097–2108.

[57] J.C. O'Callahan, A procedure for an improved reduced system (IRS) model, in: Proc. 7th Int. Modal Anal. Conf., 1989: pp. 17–21.

[58] S.N. Voormeeren, P.L. C. Van Der Valk, D.J. Rixen, Generalized Methodology for Assembly and Reduction of Component Models for Dynamic Substructuring, AIAA J. 49 (2011) 1010–1020.

[59] W. Gao, X.S. Li, C. Yang, Z. Bai, An implementation and evaluation of the AMLS method for sparse eigenvalue problems, ACM Trans. Math. Softw. V (2008) 1–27.

[60] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to algorithms, 3rd ed., Cambridge (MA): MIT press, 2001.

[61] Ansys® Academic Research Mechanical, 2020 R1.

[62] E.N. Dvorkin, K.J. Bathe, A continuum mechanics based four-node shell element for general non-linear analysis, Eng. Comput. 1 (1984) 77–88.

[63] Y. Lee, P.S. Lee, K.J. Bathe, The MITC3+ shell element and its performance, Comput. Struct. 138 (2014) 12–23.

[64] H. Jun, K. Yoon, P.S. Lee, K.J. Bathe, The MITC3+ shell element enriched in membrane displacements by interpolation covers, Comput. Methods Appl. Mech. Eng. 337 (2018) 458–480.

[65] Y. Ko, Y. Lee, P.S. Lee, K.J. Bathe, Performance of the MITC3+ and MITC4+ shell elements in widely-used benchmark problems, Comput. Struct. 193 (2017) 187–206.

[66] C. Lee, P.S. Lee, The strain-smoothed MITC3+ shell finite element, Comput. Struct. 223 (2019) 106096.

[67] C. Lee, P.S. Lee, A new strain smoothing method for triangular and tetrahedral finite elements, Comput.

Methods Appl. Mech. Eng. 341 (2018) 939–955.

[68] G. Karypis, V. Kumar, A fast and high quality multilevel scheme for partitioning irregular graphs, SIAM J. Sci. Comput. 20 (1998) 359–392.

[69] P. Holmes, J.L. Lumley, G. Berkooz, Turbulence, Coherent Structures, Dynamical Systems and Symmetry, Cambridge University Press, Cambridge, UK, 1996.

[70] L. Sirovich, Turbulence and the dynamics of coherent structures. I. Coherent structures, Q. Appl. Math. 45 (1987) 561–571.

[71] Y.C. Liang, H.P. Lee, S.P. Lim, W.Z. Lin, K.H. Lee, C.G. Wu, Proper orthogonal decomposition and its applications—Part I: Theory, J. Sound Vib. 252 (2002) 527–544.

[72] M. Barrault, Y. Maday, N.C. Nguyen, A.T. Patera, An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations, Comptes Rendus Math. 339 (2004) 667–672.

[73] P. Astrid, S. Weiland, K. Willcox, T. Backx, Missing point estimation in models described by proper orthogonal decomposition, IEEE Trans. Automat. Contr. 53 (2008) 2237–2251.

[74] K. Carlberg, C. Bou-Mosleh, C. Farhat, Efficient non-linear model reduction via a least-squares Petrov-Galerkin projection and compressive tensor approximations, Int. J. Numer. Methods Eng. 86 (2011) 155–181.

[75] K. Carlberg, C. Farhat, J. Cortial, D. Amsallem, The GNAT method for nonlinear model reduction: Effective implementation and application to computational fluid dynamics and turbulent flows, J. Comput. Phys. 242 (2013) 623–647.

[76] Z. Drmač, S. Gugercin, A new selection operator for the discrete empirical interpolation method—improved a priori error bound and extensions, SIAM J. Sci. Comput. 38 (2016) A631–A648.

[77] D. Ryckelynck, Hyper-reduction of mechanical models involving internal variables, Int. J. Numer. Methods Eng. 77 (2009) 75–89.

[78] B. Peherstorfer, D. Butnaru, K. Willcox, H. Bungartz, Localized discrete empirical interpolation method, SIAM J. Sci. Comput. 36 (2014) A168–A192.

[79] B. Peherstorfer, K. Willcox, Online adaptive model reduction for nonlinear systems via low-rank updates,

SIAM J. Sci. Comput. 37 (2015) A2123–A2150.

[80] A. Radermacher, S. Reese, Model reduction in elastoplasticity: Proper orthogonal decomposition combined with adaptive sub-structuring, Comput. Mech. 54 (2014) 677–687.

[81] A. Corigliano, M. Dossi, S. Mariani, Model order reduction and domain decomposition strategies for the solution of the dynamic elastic-plastic structural problem, Comput. Methods Appl. Mech. Eng. 290 (2015) 127–155.

[82] J.S. Hesthaven, S. Ubbiali, Non-intrusive reduced order modeling of nonlinear problems using neural networks, J. Comput. Phys. 363 (2018) 55–78.

[83] Q. Wang, J.S. Hesthaven, D. Ray, Non-intrusive reduced order modeling of unsteady flows using artificial neural networks with application to a combustion problem, J. Comput. Phys. 384 (2019) 289–307.

[84] D. Chen, D.I.W. Levin, S. Sueda, W. Matusik, Data-driven finite elements for geometry and material design, ACM Trans. Graph. 34 (2015) 74:1-74:10.

[85] J. Baiges, R. Codina, I. Castañar, E. Castillo, A finite element reduced-order model based on adaptive mesh refinement and artificial neural networks, Int. J. Numer. Methods Eng. 121 (2020) 588–601.

[86] R. Zimmermann, K. Willcox, An accelerated greedy missing point estimation procedure, SIAM J. Sci. Comput. 38 (2016) A2827–A2850

[87] C. Hyun, S.H. Boo, P.S. Lee, Improving the computational efficiency of the enhanced AMLS method, Comput. Struct. 228 (2020) 106158.

[88] S.H. Boo, J.G. Kim, P.S. Lee, Error estimation for the automated multi-level substructuring method, Int. J. Numer. Methods Eng. 106 (2016) 927–950.

[89] L. Wu, P. Tiso, F. van Keulen, Interface reduction with multilevel Craig–Bampton substructuring for component mode synthesis, AIAA J. 56 (2018) 1–15.