Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 05-002

Augmented FETI-DP Method Based on Polynomial Preconditioning

Yu Liang, Ramdev Kanapady, and Kumar Tamma

January 20, 2005

Augmented FETI-DP Method Based on Polynomial Preconditioning

Y. Liang[1], R. Kanapady[1], and K. K. Tamma[2]

Army High-performance Computing Research Center,
University of Minnesota, USA
{yliang@cs, ramdev@me, ktamma@tc}.umn.edu

Abstract. To foster the absolute scalability (A-scalability) in solving large-scale non-linear dynamic structural problem on large-processor-count high-performance computing (HPC) systems, this paper examines a scalable implementation of an augmented finite element tearing and interconnecting of dual-primal version (AFETI-DP) method based on generalized lieast-squares polynomial preconditioning. Compared to previous work, the AFETI-DP concerned is featured by utilizing finite element based iterative method to solve the coarse problem and a well load-balance general purpose (applicable to 2D and 3D problems) virtual corner selection strategy to obtain optimal computational performance. From the point of view of scalability, the corresponding experimental results for IBM-SP2 and Cray-X1 are presented and critically assessed.
KEY WORDS: polynomial preconditioner, linear equations, domain decomposition, augmented FETI-DP, Lagrange multipliers, non-linear structural dynamics.

1 Introduction

A parallel formulation employing generalized integration operator (GInO) method [10], an implicit time integration based finite element strategy, is implemented in solving large-scale non-linear structural dynamics and evaluated from the point of view of A-scalability [12]. The A-scalability of a parallel formulation is characterized by *numerical*, *parallel* and *memory utilization* scalability. In addition to the need for algorithmically scalable optimal time discretized operators, scalable spatial domain decomposition method and parallel graph partitioning technique (none of them is the major concern of this paper) [12] for non-linear structural dynamics, implicit computations need highly scalable solver for the resulting linear equations at every Newton-Raphson iteration and at every time step of analysis. In this paper AFETI-DP [1–8, 13, 14, 21, 22, 24] is employed as the underlying solver. Besides its intrinsic algorithmic scalability which has been proved [20], it is necessary to exploit a highly scalable solver at different levels of computation, i.e., local remainder DOF, interface and coarse problem. Our work is featured by utilizing element-based polynomial preconditioned conjugate gradient (PCG) method [19] to solve the coarse problem. Because the global assembly over the corners are always circumvented, the scalability inherited from

the domain decomposition strategy is fully preserved. In addition, as one of the fundamental operation in AFETI-DP, a well load-balance and general-purpose (applicable to 2D and 3D problem) virtual-corner selection algorithm is presented. The computation is implemented on an IBM-SP2 and Cray-X1. The associated experimental results are given and critically assessed.

The paper is organized as follows. Section 2 discusses the parallel implementation of finite element based domain-decomposition preconditioned conjugate gradient method. Section 3 outlines the mechanism of polynomial preconditioner with particular focus on generalized least-squares (GLS) [18, 17, 19] method. A detailed description about augmented FETI-DP is given in Section 4. As a dominant kernel of AFETI-DP, coarse problem is implemented using GLS polynomial preconditioned CG in Section 5. Section 6 discusses the corner-selection algorithm used in AFETI-DP. By applying AFETI-DP based on polynomial preconditioning into the implementation of GInO method for structural dynamics which is described in Section 7, Section 8 demonstrates the benefit caused by polynomial preconditioner.

2 Distributed Preconditioned Conjugate Gradient

This section first discusses the storage format for non-overlapping domain decomposition problem and associated computational kernels (vector update, inner product and matrix-vector product) . Based on it, preconditioned conjugate gradient is implemented.

Let Ω be mesh partitioned into $\mathcal{P}$ non-overlapping sub-domains. Two types of distributed format are used in our implementation. A global vector $\mathbf{u}$ or matrix $\mathbf{A}$ is in *local distributed format* [19] if each of its sub-block (denoted by $\mathbf{u}^{(s)}$ or $\mathbf{A}^{(s)}$) only contains the local data. A global vector $\mathbf{u}$ or a matrix $\mathbf{A}$ is in *global distributed format* [19] if each of its sub-block (denoted as $\hat{\mathbf{u}}^{(s)}$ or $\hat{\mathbf{A}}^{(s)}$) contains the data with consistent interface components.

Define $\mathbf{B}_s \in \Re^{N_s \times N}$ as a unsigned boolean matrix which maps the global vector into a sub-domain s. It follows that

$$\hat{\mathbf{u}}^{(s)} \equiv \mathbf{B}_s \mathbf{u}; \quad \text{and} \quad \mathbf{u} \equiv \sum_{s=1}^{\mathcal{P}} \mathbf{B}_s^{\mathsf{T}} \mathbf{u}^{(s)} \tag{1}$$

A local distributed vector $\mathbf{u}^{(s)}$ can be converted into a global distributed vector $\hat{\mathbf{u}}^{(s)}$ through

$$\hat{\mathbf{u}}^{(s)} = \bowtie \sum^{\partial\Omega_s} \mathbf{u}^{(s)}, \tag{2}$$

where $\bowtie \sum^{\partial\Omega_s}$ is called the *nearest neighbor communication* [19] within $\partial\Omega_s \backslash \partial\Omega$. Analogous to vectors, a global stiffness matrix $\mathbf{A}$ also consist of a series of global distributed sub-matrix $\hat{\mathbf{A}}^{(s)}$ or a series of local distributed sub-matrix $\mathbf{A}^{(s)}$. It is clear that, in the general case,

$$\hat{\mathbf{A}}^{(s)} = \mathbf{B}_s \mathbf{A} \mathbf{B}_s^{\mathsf{T}}; \quad \text{and} \quad \mathbf{A} = \sum_{s=1}^{\mathcal{P}} \mathbf{B}_s^{\mathsf{T}} \mathbf{A}^{(s)} \mathbf{B}_s \tag{3}$$

Here $\mathbf{A}$ is the global stiffness matrix for domain Ω with homogeneous Neumann boundary conditions on the inner boundaries $\partial\Omega$.

In the implementation of iterative methods for the solution of linear equations, four basic computational kernels, namely, vector update, inner-product, matrix-by-vector product and preconditioning, are needed. Vector update does not require any inter-process communication. Inner product $\langle \mathbf{x}, \mathbf{y} \rangle (\mathbf{x}, \mathbf{y} \in \Re^N)$ is equal to $\sum_{s=1}^{\mathcal{P}} \langle \mathbf{y}^{(s)}, \hat{\mathbf{x}}^{(s)} \rangle$ (denoted as $\forall \sum^{\Omega} \langle \mathbf{y}^{(s)}, \hat{\mathbf{x}}^{(s)} \rangle$ in this paper), or equally, $\sum_{s=1}^{\mathcal{P}} \langle \mathbf{x}^{(s)}, \hat{\mathbf{y}}^{(s)} \rangle$. The matrix-vector operation given by $\mathbf{Ax}$ ($\mathbf{x} \in \Re^N, \mathbf{A} \in \Re^{N \times N}$) can be represented as $\sum_{s=1}^{\mathcal{P}} \mathbf{B}_s^{\mathsf{T}} \mathbf{A}^{(s)} \hat{\mathbf{x}}^{(s)}$. The implementation of preconditioning is analogous to that of matrix-vector product [19].

Based on the previous discussion, an extended preconditioned conjugate gradient with respect to finite element-based linear equations

$$\bigcup_{s=1}^{\Omega\ \mathcal{P}} \left\{ \left[\mathbf{A}^{(s)} \right] \left[\hat{\mathbf{x}}^{(s)} \right] \right\} = \bigcup_{s=1}^{\Omega\ \mathcal{P}} \left\{ \mathbf{b}^{(s)} \right\}, \tag{4}$$

where "$\bigcup^{\Omega}$" denotes global assembly in domain Ω, is given by Algorithm 1.

Algorithm 1 *EDD-PCG for (4). An initial guess* $\hat{\mathbf{x}}_0$ *about the solution is given and* $\mathbf{C}$ *is the given preconditioner.*

(1) $\mathbf{r}_0^{(s)} = \mathbf{b}^{(s)} - \mathbf{A}^{(s)} \hat{\mathbf{x}}_0^{(s)}$; $\hat{\mathbf{r}}_0^{(s)} = \bowtie \sum^{\partial\Omega_s} \mathbf{r}_0^{(s)}$; $\mathbf{z}_0^{(s)} = \mathbf{C}^{(s)} \hat{\mathbf{r}}_0^{(s)}$;

(2) $\hat{\mathbf{z}}_0^{(s)} = \bowtie \sum^{\partial\Omega_s} \mathbf{z}_0^{(s)}$; $\hat{\mathbf{p}}_0^{(s)} = \hat{\mathbf{z}}_0^{(s)}$; $\sigma_0 = \forall \sum^{\Omega} \langle \hat{\mathbf{z}}_0^{(s)}, \mathbf{r}_0^{(s)} \rangle$;

(3) FOR $k = 0, 1, \cdots$ till Convergence DO /* Krylov Iteration */

(4) $\quad \mathbf{q}_k^{(s)} = \mathbf{A}^{(s)} \hat{\mathbf{p}}_k^{(s)}$; /* matrix-vector product */

(5) $\quad \delta_k = \forall \sum^{\Omega} \langle \hat{\mathbf{p}}_k^{(s)}, \mathbf{q}_k^{(s)} \rangle$; $\alpha_k = \frac{\sigma_k}{\delta_k}$; /* compute step-length */

(6) $\quad \hat{\mathbf{x}}_{k+1}^{(s)} = \hat{\mathbf{x}}_k^{(s)} + \alpha_k \hat{\mathbf{p}}_k^{(s)}$; $\mathbf{r}_{k+1}^{(s)} = \mathbf{r}_k^{(s)} - \alpha_k \mathbf{q}_k^{(s)}$; /* update x and r */

(7) $\quad \hat{\mathbf{r}}_{k+1}^{(s)} = \bowtie \sum^{\partial\Omega_s} \mathbf{r}_{k+1}^{(s)}$; $\mathbf{z}_{k+1}^{(s)} = \mathbf{C}^{(s)} \hat{\mathbf{r}}_{k+1}^{(s)}$; /* preconditioning */

(8) $\quad \hat{\mathbf{z}}_{k+1}^{(s)} = \bowtie \sum^{\partial\Omega_s} \mathbf{z}_{k+1}^{(s)}$; $\sigma_{k+1} = \forall \sum^{\Omega} \langle \hat{\mathbf{z}}_{k+1}^{(s)}, \mathbf{r}_{k+1}^{(s)} \rangle$;

(9) $\quad \beta_k = \frac{\sigma_{k+1}}{\sigma_k}$; $\hat{\mathbf{p}}_{k+1}^{(s)} = \hat{\mathbf{z}}_{k+1}^{(s)} + \beta_k \hat{\mathbf{p}}_k^{(s)}$; /* update search direction */

(10) END FOR

3 Polynomial Preconditioner

3.1 Introduction

The interest in polynomial preconditioners [17] is motivated by the need for simple, yet efficient and effective methods for efficient parallel iterative solvers [23] for high performance computers (HPC) such as vector and parallel processors. In contrast to ILU, SPAI, etc. polynomial preconditioners are constructed only based on the the matrix spectrum (i.e., eigvalues of the coefficient matrix), which is denoted as $\sigma(\mathbf{K})$. Even though $\sigma(\mathbf{K})$ is generally difficult to compute, Θ, an approximate estimation to it can be easily obtained. When $\mathbf{K}$ is a symmetric

matrix, then $\Theta \subset \Re$; otherwise $\Theta \subset \mathcal{C}$. The accuracy of Θ determines the rate of convergence of the preconditioned systems. Since polynomial preconditioning will always keep the symmetry of linear systems [17], in this paper we employ left-preconditioning [19, 23] to transform the original linear systems, for example, $\mathbf{ku} = \mathbf{f}$, to yield

$$P_m(\mathbf{K})\mathbf{K}u = P_m(\mathbf{K})\mathbf{f} \tag{5}$$

where $P_m(\mathbf{K})$ is a polynomial in $\mathbf{K}$ with degree $\leq m$. It has been shown in [17] that, if $P_m(\lambda)$ is constructed such that

$$\min_{P_m \in \wp_m[\Theta]} \|1 - \lambda P_m(\lambda)\| \quad (\lambda \in \Theta) \tag{6}$$

where $\wp_m[\Theta]$ is the set of m-degree polynomials which is valid over Θ, and $\|.\|$ represents a specific norm, namely, the uniform norm or quadratic norm, then a bound on the condition number [23] of the preconditioned systems, $\kappa(P_m(\mathbf{K})\mathbf{K})$, is minimized.

In this paper the coefficient matrix $\mathbf{A}$ is assumed to be symmetric. The difficulty in applying the polynomial preconditioner lies in the estimation of the spectrum of coefficient matrix. This can be easily achieved by a simple preprocessing technique that maps the spectrum of the coefficient into a predetermined interval. This is discussed in the next section.

3.2 Generalized Least-squares Polynomial Preconditioner

Generalized least-squares (GLS) polynomial preconditioning is preferable to other polynomial preconditioning methods (i.e., Neumann series, least-squares, Chebyshev etc.) due to its high application flexibility and low construction cost.

For GLS polynomial preconditioning method, Θ is defined as a union of arbitrary number of disjoint intervals of spectrum, i.e.,

$$\Theta = \bigcup_{k=1}^{N_I} (\ell_k, \hbar_k),\ 0 \notin \Theta \ and\ \ell_1 < \hbar_1 \leq \ell_2 < \hbar_2 \leq \cdots \leq \ell_{N_I} < \hbar_{N_I}, \tag{7}$$

where N_I is the number of intervals in which the eigenvalue of coefficient matrix $\mathbf{K}$ lies, therefore GLS method can be a general method of solving symmetric linear systems including both symmetric indefinite and symmetric positive definite systems.

Once the Θ is determined, GLS polynomial preconditioner can be constructed such that

$$\min_{P_m \in \wp_m[\Theta]} \|1 - \lambda P_m(\lambda)\|_w, \quad \lambda \in \Theta \tag{8}$$

where $\|.\|_w$ represents the weighted quadratic norm induced by the inner product $\langle f, g\rangle_w = \int_\Theta f(\lambda)g(\lambda)w(\lambda)d\lambda$ and $w(\lambda)$ is a non-negative weight function over the interval Θ. The key to the above least-squares problem Eq. 8 is the construction of a sequence of orthogonal polynomials. Given a series of orthogonal polynomial

sequence $\{\lambda\phi_i(\lambda)\}_{i=0}^m$ with respect to inner-product $\langle , \rangle_w$ [17] as a basis, then the least-squares problem Eq. 8 can be readily constructed via

$$P_m(\lambda) = \sum_{i=0}^{m} \mu_i \phi_i(\lambda), \tag{9}$$

where $\mu_i = \langle 1, \lambda\phi_i(\lambda)\rangle_w$. As a result, polynomial preconditioning can be computed iteratively as

$$P_m(\mathbf{A})\mathbf{v} = \sum_{i=0}^{m} \mu_i \phi_i(\mathbf{A})\mathbf{v}. \tag{10}$$

In this work the orthogonal polynomial sequence $\{\lambda\phi_i(\lambda)\}_{i=0}^m$ is efficiently constructed using Chebyshev polynomial [17] embedded in the Stieltje procedure [17] via the three-term recurrence relationship. The storage and time cost for this operation is negligible compared to ILU, SPAI, etc. For more detailed information about the sequential implementation of polynomial preconditioner, reader is referred to [17].

4 Element-based Domain Decomposition AFETI-DP

The linear/non-linear static/dynamic structural problems can be always cast into the solution of the following linearized equations:

$$\mathbf{Ku} = \mathbf{f}, \tag{11}$$

where $\mathbf{K} \in \Re^{N\times N}$ is the effective stiffness matrix and $\mathbf{f} \in \Re^N$ is the vector of effective external applied forces. When N is extremely large, *domain decomposition* (DD) strategy is needed and the *Finite Element Tearing and Interconnecting*(FETI) method appears promising for non-overlapping domain decomposition algorithms.

Figure 1 shows the historical progress of FETI family. The *one-level FETI method* (FETI-I) [1] was originated from *primal-Schur-Complement method* by applying the Lagrange multiplier to enforce the continuity of *degrees of freedom* (DOF) along sub-domain interfaces. The resulted dual system for the Lagrange multipliers is solved using *preconditioned conjugate gradients* (PCG). The *two-level FETI* (FETI-II) [7] method was developed for fourth-order problems. FETI-II is featured in applying an extra set of Lagrange multipliers to enforce the continuity at the sub-domain corners. Based on FETI-I, *Dual-Primal FETI* (FETI-DP) [7, 16, 21] directly enforces the continuity of the corners so as to achieve stable convergence. With the similar idea as FETI-II, *augmented FETI-DP method* (AFETI-DP) [21] is an extension of FETI-DP methods by employing a set of additional Lagrange multipliers to enforce the continuity along the chosen interface nodes (excluding the corners), which are called *virtual corners*. For three-dimensional problems, AFETI-DP is necessary to obtain more satisfactory convergence rates than FETI-DP [20, 21]. Our work will focus on the implementation of AFETI-DP.

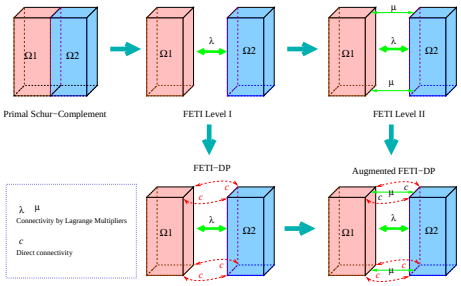

Fig. 1. Definition of FETI Methods

4.1 Derivation of Augmented FETI-DP

The FETI family methods are always based on the decoupling of DOF. Figure 2 shows the different kinds of decoupling strategies of the DOF within sub-domain s, all of which will be referred in this paper. For notational simpleness, it is assumed that $\mathbf{u}_{bc}^{(s)} \equiv \mathbf{u}_{c}^{(s)}$ and $\mathbf{u}_{vc}^{(s)} \equiv \mathbf{u}_{q}^{s}$. With respect to various of decoupling strategies, the boolean transformation matrices are defined as follow:

$$\begin{cases} \mathbf{B}_r^s \mathbf{u}_r^{(s)} = \pm \mathbf{u}_{br}^{(s)}, \ \mathbf{Q}_{br}^s \mathbf{u}_{b_r}^{(s)} = \mathbf{u}_{vc}^{(s)} \equiv \mathbf{u}_q^{(s)}, \\ \mathbf{B}_c^s \mathbf{u}_c = \hat{\mathbf{u}}_{bc}^{(s)} \equiv \hat{\mathbf{u}}_c^{(s)}, \ \mathbf{T}_\lambda^s \mathbf{u}_{br} = \hat{\mathbf{u}}_{br}^{(s)}, \ \mathbf{T}_\mu^s \mathbf{u}_q = \hat{\mathbf{u}}_q^{(s)}, \end{cases} \tag{12}$$

where $\mathbf{B}_c^s$ is determined by the corner selection strategy and $\mathbf{Q}_{br}^s$ is determined by the virtual-corner selection.

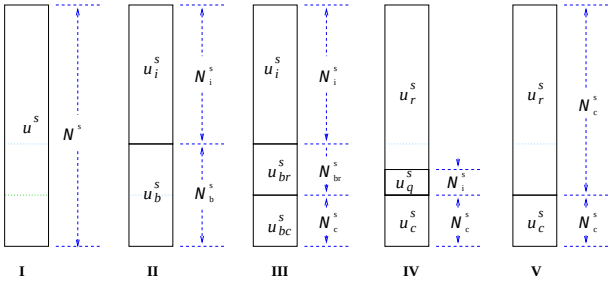

Fig. 2. De-coupling of DOF in Ω^s (subscript i, b, c, r, br, bc and vc(or q) denote interior, boundary, corner, remainder, boundary-remainder, boundary-corner and virtual-corner DOF respectively.)

After the "tearing" process, the given linear system $\mathbf{Ku} = \mathbf{f}$, is transformed into

$$\bigcup_{s=1}^{\Omega\ \mathcal{P}} \left\{ \mathbf{K}^{(s)} \mathbf{u}^{(s)} \right\} = \bigcup_{s=1}^{\Omega\ \mathcal{P}} \left\{ \mathbf{f}^{(s)} \right\} \tag{13}$$

with constraints $\bigcup_{s=1}^{\Omega_{br}\ \mathcal{P}}\{\mathbf{B}_r^s\mathbf{u}_r^{(s)}\} = 0$ (Ω_{br} denotes the context of boundary-remainder DOF). As an extension of FETI-DP, which ties disconnected sub-domains together using Lagrange multiplier $\lambda \in \Re^{N_{br}}$ and corners, the augmented FETI-DP (AFETI-DP) explores redundant Lagrange multipliers $\mu \in \Re^{N_q}$ which comes from the augmented constraints $\bigcup_{s=1}^{\Omega_{vc}\ \mathcal{P}}\{\mathbf{Q}_{br}^{s\ \mathsf{T}}\mathbf{B}_r^s\mathbf{u}_r^{(s)}\} = 0$ (Ω_{vc} denotes the context of virtual-corner DOF) to fasten the bind between sub-domains so as to achieve better convergence. By employing corner continuity, Lagrange multipliers and re-order strategy, the distributed linear equations (13) is reformulated as

$$\begin{bmatrix} \mathbf{K}_{rr}^{(1)} & \cdots & 0 & \mathbf{K}_{rc}^{(1)}\mathbf{B}_c^1 & \mathbf{B}_r^{1\mathsf{T}}\mathbf{Q}_{br}^1\mathbf{T}_q^1 & \mathbf{B}_r^{1\mathsf{T}}\mathbf{T}_\lambda^1 \\ \vdots & \ddots & \vdots & \cdots & \vdots & \vdots \\ 0 & \cdots & \mathbf{K}_{rr}^{(\mathcal{P})} & \mathbf{K}_{rc}^{(\mathcal{P})}\mathbf{B}_c^{\mathcal{P}} & \mathbf{B}_r^{\mathcal{P}\mathsf{T}}\mathbf{Q}_{br}^{\mathcal{P}}\mathbf{T}_q^{\mathcal{P}} & \mathbf{B}_r^{\mathcal{P}\mathsf{T}}\mathbf{T}_\lambda^{\mathcal{P}} \\ \mathbf{B}_c^{1\mathsf{T}}\mathbf{K}_{rc}^{(1)\mathsf{T}} & \cdots & \mathbf{B}_c^{\mathcal{P}\mathsf{T}}\mathbf{K}_{rc}^{(\mathcal{P})\mathsf{T}} & \mathbf{K}_{cc} & 0 & 0 \\ \mathbf{T}_q^{1\mathsf{T}}\mathbf{Q}_{br}^{1\ \mathsf{T}}\mathbf{B}_r^1 & \cdots & \mathbf{T}_q^{\mathcal{P}\mathsf{T}}\mathbf{Q}_{br}^{\mathcal{P}\ \mathsf{T}}\mathbf{B}_r^{\mathcal{P}} & 0 & 0 & 0 \\ \mathbf{T}_\lambda^{1\mathsf{T}}\mathbf{B}_r^1 & \cdots & \mathbf{T}_\lambda^{\mathcal{P}\mathsf{T}}\mathbf{B}_r^{\mathcal{P}} & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}_r^{(1)} \\ \vdots \\ \mathbf{u}_r^{(\mathcal{P})} \\ \mathbf{u}_c \\ \mu \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{f}_r^{(1)} \\ \vdots \\ \mathbf{f}_r^{(\mathcal{P})} \\ \mathbf{f}_c \\ 0 \\ 0 \end{bmatrix} \tag{14}$$

From (14) it directly follows that

$$\mathbf{K}_{rr}^{(s)}\mathbf{u}_r^s + \mathbf{K}_{rc}^{(s)}\mathbf{B}_c^s\mathbf{u}_c + \mathbf{B}_r^{s\mathsf{T}}\mathbf{Q}_{br}^s\mathbf{T}_q^s\mu + \mathbf{B}_r^{s\mathsf{T}}\mathbf{T}_\lambda^s\lambda = \mathbf{f}_r^{(s)},\ s = 1, \cdots, \mathcal{P}, \tag{15}$$

$$\sum_{s=1}^{\mathcal{P}} \mathbf{T}_\lambda^{s\mathsf{T}}\mathbf{B}_r^s\mathbf{u}_r^{(s)} = 0, \tag{16}$$

$$\sum_{s=1}^{\mathcal{P}} \mathbf{B}_c^{s\mathsf{T}}\mathbf{K}_{rc}^{(s)\mathsf{T}}\mathbf{u}_r^{(s)} + \sum_{s=1}^{\mathcal{P}} \mathbf{B}_c^{s\mathsf{T}}\mathbf{K}_{cc}^{(s)}\mathbf{B}_c^s\mathbf{u}_c = \mathbf{f}_c, \tag{17}$$

and

$$\sum_{s=1}^{\mathcal{P}} \mathbf{T}_q^{s\mathsf{T}}\mathbf{Q}_{br}^{s\ \mathsf{T}}\mathbf{B}_r^s\mathbf{u}_r^{(s)} = 0. \tag{18}$$

Trivially (15) leads to

$$\mathbf{u}_r^{(s)} = \mathbf{K}_{rr}^{(s)\ -1}(\mathbf{f}_r^{(s)} - \mathbf{B}_r^{s\mathsf{T}}\mathbf{T}_\lambda^s\lambda - \mathbf{K}_{rc}^{(s)}\mathbf{B}_c^s u_c - \mathbf{B}_r^{s\mathsf{T}}\mathbf{Q}_{br}^s\mathbf{T}_q^s\mu). \tag{19}$$

Substitute (19) into (16),(17) and (18) respectively, it follows that

$$\begin{array}{l} \sum_s^{\mathcal{P}} \mathbf{T}_\lambda^{s\mathsf{T}}\mathbf{B}_r^s\mathbf{K}_{rr}^{(s)\ -1}\mathbf{B}_r^{s\mathsf{T}}\mathbf{T}_\lambda^s\lambda + \sum_s^{\mathcal{P}} \mathbf{T}_\lambda^{s\mathsf{T}}\mathbf{B}_r^s\mathbf{K}_{rr}^{(s)\ -1}\mathbf{K}_{rc}^{(s)}\mathbf{B}_c^s\mathbf{u}_c + \\ \sum_s^{\mathcal{P}} \mathbf{T}_\lambda^{s\mathsf{T}}\mathbf{B}_r^s\mathbf{K}_{rr}^{(s)\ -1}\mathbf{B}_r^{s\mathsf{T}}\mathbf{Q}_{br}^s\mathbf{T}_q^s\mu = \sum_s^{\mathcal{P}} \mathbf{T}_\lambda^{s\mathsf{T}}\mathbf{B}_r^s\mathbf{K}_{rr}^{(s)\ -1}\mathbf{f}_r^s, \end{array} \tag{20}$$

$$\begin{array}{l} \sum_s^{\mathcal{P}} \mathbf{B}_c^{s\mathsf{T}}\mathbf{K}_{rc}^{(s)\mathsf{T}}\mathbf{K}_{rr}^{(s)\ -1}\mathbf{B}_r^{s\mathsf{T}}\mathbf{T}_\lambda^s\lambda - \sum_s^{\mathcal{P}}(\mathbf{B}_c^{s\mathsf{T}}\mathbf{K}_{cc}^{(s)}\mathbf{B}_c^s - \\ \mathbf{B}_c^{s\mathsf{T}}\mathbf{K}_{rc}^{(s)\mathsf{T}}\mathbf{K}_{rr}^{(s)\ -1}\mathbf{K}_{rc}^{(s)}\mathbf{B}_c^s)\mathbf{u}_c + \sum_s^{\mathcal{P}} \mathbf{B}_c^{s\mathsf{T}}\mathbf{K}_{rc}^{(s)\mathsf{T}}\mathbf{K}_{rr}^{(s)\ -1}\mathbf{B}_r^{s\mathsf{T}}\mathbf{Q}_{br}^s\mathbf{T}_q^s\mu = \\ -\mathbf{f}_c + \sum_s^{\mathcal{P}} \mathbf{B}_c^{s\mathsf{T}}\mathbf{K}_{rc}^{(s)\mathsf{T}}\mathbf{K}_{rr}^{(s)\ -1}\mathbf{f}_r^{(s)}, \end{array} \tag{21}$$

and

$$\begin{array}{l}\sum_s^{\mathcal{P}} \mathbf{T}_q^{s\mathsf{T}} \mathbf{Q}_{br}^{s\ \mathsf{T}} \mathbf{B}_r^s \mathbf{K}_{rr}^{(s)^{-1}} \mathbf{B}_r^{s\mathsf{T}} \mathbf{T}_\lambda^s \lambda + \sum_s^{\mathcal{P}} \mathbf{T}_q^{s\mathsf{T}} \mathbf{Q}_{br}^{s\ \mathsf{T}} \mathbf{B}_r^s \mathbf{K}_{rr}^{(s)^{-1}} \mathbf{K}_{rc}^{(s)} \mathbf{B}_c^s \mathbf{u}_c + \\ \sum_s^{\mathcal{P}} \mathbf{T}_q^{s\mathsf{T}} \mathbf{Q}_{br}^{s\ \mathsf{T}} \mathbf{B}_r^s \mathbf{K}_{rr}^{(s)^{-1}} \mathbf{B}_r^{s\mathsf{T}} \mathbf{Q}_{br}^s \mathbf{T}_q^s \mu = \sum_s^{\mathcal{P}} \mathbf{T}_q^{s\mathsf{T}} \mathbf{Q}_{br}^{s\ \mathsf{T}} \mathbf{B}_r^s \mathbf{K}_{rr}^{(s)^{-1}} \mathbf{f}_r^{(s)}.\end{array} \tag{22}$$

Using (12), (20), (21) and (22) can be expressed in the following matrix form:

$$\bigcup_s \left\{ \begin{bmatrix} \mathbf{F}_{rr}^{(s)} & \mathbf{F}_{rc}^{(s)} & \mathbf{F}_{rq}^{(s)} \\ \mathbf{F}_{rc}^{(s)^{\mathsf{T}}} & -\mathbf{K}_{cc}^{*(s)} & \mathbf{F}_{cq}^{(s)} \\ \mathbf{F}_{rq}^{(s)^{\mathsf{T}}} & \mathbf{F}_{cq}^{(s)^{\mathsf{T}}} & \mathbf{F}_{qq}^{(s)} \end{bmatrix} \begin{bmatrix} \hat{\lambda}^{(s)} \\ \hat{\mathbf{u}}_c^{(s)} \\ \hat{\mu}^{(s)} \end{bmatrix} \right\} = \bigcup_s \begin{bmatrix} \mathbf{d}_{br}^{(s)} \\ -\mathbf{f}^{(s)^*}{}_c \\ \mathbf{f}_q^{(s)} \end{bmatrix}, \tag{23}$$

where

$$\begin{cases} \mathbf{F}_{rr}^{(s)} = \mathbf{B}_r^s \mathbf{K}_{rr}^{(s)^{-1}} \mathbf{B}_r^{s\mathsf{T}}, & \mathbf{F}_{rc}^{(s)} = \mathbf{B}_r^s \mathbf{K}_{rr}^{(s)^{-1}} \mathbf{K}_{rc}^{(s)}, \\ \mathbf{F}_{rq}^{(s)} = \mathbf{B}_r^s \mathbf{K}_{rr}^{(s)^{-1}} \mathbf{B}_r^{s\mathsf{T}} \mathbf{Q}_{br}^s, & \mathbf{K}_{cc}^{*\ (s)} = \mathbf{K}_{cc}^{(s)} - \mathbf{K}_{rc}^{(s)^{\mathsf{T}}} \mathbf{K}_{rr}^{(s)^{-1}} \mathbf{K}_{rc}^{(s)}, \\ \mathbf{F}_{cq}^{(s)} = \mathbf{K}_{rc}^{(s)^{\mathsf{T}}} \mathbf{K}_{rr}^{(s)^{-1}} \mathbf{B}_r^{s\mathsf{T}} \mathbf{Q}_{br}^s, & \mathbf{F}_{qq}^{(s)} = \mathbf{Q}_{br}^{s\ \mathsf{T}} \mathbf{B}_r^s \mathbf{K}_{rr}^{(s)^{-1}} \mathbf{B}_r^{s\mathsf{T}} \mathbf{Q}_{br}^s, \\ \mathbf{d}_{br}^{(s)} = \mathbf{B}_r^s \mathbf{K}_{rr}^{(s)^{-1}} \mathbf{f}_r^{(s)}, & \mathbf{f}_c^{*(s)} = \mathbf{f}_c^{(s)} - \mathbf{K}_{rc}^{(s)^{\mathsf{T}}} \mathbf{K}_{rr}^{(s)^{-1}} \mathbf{f}_r^{(s)}, \\ \mathbf{f}_q^{(s)} = \mathbf{Q}_{br}^s \mathbf{d}_{br}^{(s)}. \end{cases} \tag{24}$$

In (24), $\mathbf{F}_{rr}^{(s)}$, $\mathbf{F}_{rc}^{(s)}$, $\mathbf{F}_{rq}^{(s)}$, $\mathbf{K}_{cc}^{*\ (s)}$, $\mathbf{F}_{cq}^{(s)}$, $\mathbf{F}_{qq}^{(s)}$, $\mathbf{d}_{br}^{(s)}$ and $\mathbf{f}_c^{*(s)}$ are local with respect to sub-domain s and their formulation requires no inter-processor communication.

In order to make the AFETI-DP method be in similar expression form as the FETI-DP method discussed in [14, 16], let

$$\tilde{\mathbf{u}}_c^{(s)} = \begin{bmatrix} \mathbf{u}_c^{(s)} \\ \mu^{(s)} \end{bmatrix}, \tilde{\mathbf{f}}_c^{*(s)} = \begin{bmatrix} \mathbf{f}_c^{*(s)} \\ -\mathbf{f}_q^{(s)} \end{bmatrix}, \tilde{\mathbf{K}}_{cc}^{*(s)} = \begin{bmatrix} \mathbf{K}_{cc}^{*(s)} & -\mathbf{F}_{cq}^{(s)} \\ -\mathbf{F}_{cq}^{(s)^{\mathsf{T}}} & -\mathbf{F}_{qq}^{(s)} \end{bmatrix}, \tilde{\mathbf{F}}_{rc}^{(s)} = \begin{bmatrix} \mathbf{F}_{rc}^{(s)} & \mathbf{F}_{rq}^{(s)} \end{bmatrix}. \tag{25}$$

Then after a series of algebraic computation, it follows that

$$\bigcup_{br}^{\Omega_{br}} \left\{ \left[\mathbf{F}_{rr}^{(s)} + \tilde{\mathbf{F}}_{rc}^{(s)} (\tilde{\mathbf{K}}_{cc}^{*(s)})^{-1} (\tilde{\mathbf{F}}_{rc}^{(s)})^{\mathsf{T}} \right] \left[\hat{\lambda}^{(s)} \right] \right\} = \bigcup_{br}^{\Omega_{br}} \left\{ \mathbf{d}_{br}^{(s)} - \tilde{\mathbf{F}}_{rc}^{(s)} (\tilde{\mathbf{K}}_{cc}^{*(s)})^{-1} \tilde{\mathbf{f}}_c^{*(s)} \right\}; \tag{26}$$

which is called the *interface problem*. After $\lambda^{(s)}$ in the linear system (26) is solved, $\tilde{\mathbf{u}}_c = [\mathbf{u}_c, \mu]^{\mathsf{T}}$ is easily obtained.

Based on the previous discussion, the AFETI-DP method is implemented via the following sequence of operations.

1. Factorization: $\mathbf{K}_{rr}^{(s)} = \mathbf{L}_{rr}^{(s)} \mathbf{L}_{rr}^{(s)^{\mathsf{T}}}$;
2. Solution of interface Problem (26);
3. Recovery of $\hat{\mathbf{u}}_c^{(s)}$: $\bigcup^{\Omega_c} \left\{ \left[\tilde{\mathbf{K}}_{cc}^{*(s)} \right] \begin{bmatrix} \hat{\mathbf{u}}_c^{(s)} \\ \hat{\mu}^{(s)} \end{bmatrix} \right\} = \bigcup^{\Omega_c} \left\{ \tilde{\mathbf{f}}_c^{(s),*} + (\tilde{\mathbf{F}}_{rc}^{(s)})^{\mathsf{T}} \lambda \right\}$;
4. Recovery of $\hat{\mathbf{u}}_r^{(s)}$: $\mathbf{L}_{rr}^s \mathbf{L}_{rr}^{(s)^{\mathsf{T}}} \hat{\mathbf{u}}_r^{(s)} = \hat{\mathbf{f}}_r^{(s)} - \mathbf{B}_r^{s\mathsf{T}} \hat{\lambda}^{(s)} - \mathbf{K}_{rc}^{(s)} \hat{\mathbf{u}}_c^{(s)} - \mathbf{Q}_{br}^s \hat{\mu}^{(s)}$.

4.2 Solution of Interface Problem

The key to implement AFETI-DP is the solution of interface problem (26). Because $\bigcup^{\Omega_{br}} \left\{ \mathbf{F}_{rr} + \tilde{\mathbf{F}}_{rc}(\tilde{\mathbf{K}}^{*}_{cc})^{-1}\tilde{\mathbf{F}}^{\mathsf{T}}_{rc} \right\}$ is sparse, symmetric and positive definite, the *Preconditioned Conjugate Gradient* (PCG) algorithm is employed in our work and described by Algorithm 2. Of two commonly used preconditioners for the interface problem — the *Dirichlet preconditioner*[1] and the *lumped preconditioner*[1], the latter is used here due to its high scalability.

Algorithm 2 *EDD-PCG for AFETI-DP Interface Problem (26) where s denotes native sub-domain.*

(1) **Initial**:

(2) $\hat{\lambda}^{(s)}_0 = 0$; `/* Initialize solution vector */`

(3) $\bigcup_s \left\{ \left[\tilde{\mathbf{K}}^{*(s)}_{cc}\right] \quad \begin{bmatrix} \hat{\mathbf{w}}^{(s)}_{c,0} \\ \hat{\mathbf{w}}^{(s)}_{q,0} \end{bmatrix} \right\} = \bigcup_s \begin{bmatrix} \mathbf{f}^{*(s)}_{c} \\ -\mathbf{Q}^{s\ \mathsf{T}}_{br}\mathbf{d}^{(s)}_{br} \end{bmatrix}$;

`/* coarse problem to be solved using Algorithm 3 */`

(4) $\mathbf{r}^{(s)}_0 = \mathbf{d}^{(s)}_{br} - \mathbf{F}^{(s)}_{rc}\hat{\mathbf{w}}^{(s)}_{c,0} - \mathbf{F}^{(s)}_{rq}\hat{\mathbf{w}}^{(s)}_{q,0}$; $\hat{\mathbf{r}}^{(s)}_0 = \bowtie \sum^{\partial\Omega^{s}_{br}} \mathbf{r}^{(s)}_0$;

(5) $\mathbf{z}^{(s)}_0 = \mathbf{C}^{(s)}\hat{\mathbf{r}}^{(s)}_0$; $\hat{\mathbf{z}}^{(s)}_0 = \bowtie \sum^{\partial\Omega^{s}_{br}} \mathbf{z}^{(s)}_k$; $\hat{\mathbf{p}}^{(s)}_0 = \hat{\mathbf{z}}^{(s)}_0$;

(6) $\sigma_0 = \forall \sum^{\Omega_{br}} \langle \mathbf{z}^{(s)}_0, \hat{\mathbf{r}}^{(s)}_0 \rangle$;

(7) **Iterate**:

(8) `FOR` ($k = 0, 1, \cdots$, `till convergence occurs`)

`/* SECTION 1: Matrix-vector product` $\mathbf{w}_k \leftarrow A\hat{\mathbf{p}}^{(k)}$ `*/`

(9) $\begin{bmatrix} \mathbf{y}^{(s)}_{c,k} \\ \mathbf{y}^{(s)}_{q,k} \end{bmatrix} = \begin{bmatrix} \mathbf{F}^{(s)\mathsf{T}}_{rc}\hat{\mathbf{p}}^{(s)}_k \\ \mathbf{F}^{(s)\mathsf{T}}_{rq}\hat{\mathbf{p}}^{(s)}_k \end{bmatrix}$;

(9) $\bigcup_s \left\{ \left[\tilde{\mathbf{K}}^{*(s)}_{cc}\right] \begin{bmatrix} \hat{\mathbf{w}}^{(s)}_{c,k} \\ \hat{\mathbf{w}}^{(s)}_{q,k} \end{bmatrix} \right\} = \bigcup_s \begin{bmatrix} \mathbf{y}^{(s)}_{c,k} \\ \mathbf{y}^{(s)}_{q,k} \end{bmatrix}$;

(10) $\mathbf{q}_k{}^{(s)} = \mathbf{B}^{s}_{r}\mathbf{K}^{(s)\ -1}_{rr}(\hat{\mathbf{p}}^{(s)}_k + \mathbf{K}^{(s)}_{rc}\hat{\mathbf{w}}^{(s)}_{c,k} + \mathbf{B}^{s\mathsf{T}}_{r}\mathbf{Q}^{s}_{br}\hat{\mathbf{w}}^{(s)}_{q,k})$;

`/* SECTION 2: compute step-length: */`

(11) $\delta_k = \forall \sum^{\Omega_{br}} \langle \mathbf{q}^{(s)}_k, \hat{\mathbf{p}}^{(s)}_k \rangle$; $\alpha_k = \frac{\sigma_k}{\delta_k}$;

`/* SECTION 3: update solution and residual */`

(12) $\hat{\lambda}^{(s)}_{k+1} = \hat{\lambda}^{(s)}_k + \alpha_k\hat{\mathbf{p}}^{(s)}_k$;

(12) $\mathbf{r}^{(s)}_{k+1} = \mathbf{r}^{(s)}_k - \alpha_k\mathbf{q}^{(s)}_k$;

`/* SECTION 4: gradient preconditioning */`

(13) $\hat{\mathbf{r}}^{(s)}_{k+1} = \bowtie \sum^{\partial\Omega^{s}_{br}} \mathbf{r}^{(s)}_{k+1}$;

(13) $\mathbf{z}^{(s)}_{k+1} = \mathbf{C}^{(s)}\hat{\mathbf{r}}^{(s)}_{k+1}$;

(13) $\hat{\mathbf{z}}^{(s)}_{k+1} = \bowtie \sum^{\partial\Omega^{s}_{br}} \mathbf{z}^{(s)}_{k+1}$;

`/* SECTION 5: update search direction` p_k `*/`

(14) $\sigma_{k+1} = \forall \sum^{\Omega_{br}} \langle \mathbf{r}^{(s)}_{k+1}, \hat{\mathbf{z}}^{(s)}_{k+1} \rangle$;

(14) $\beta_k = \frac{\sigma_{k+1}}{\sigma_k}$;

(14) $\hat{\mathbf{p}}_{k+1}^{(s)} = \hat{\mathbf{z}}_{k+1}^{(s)} + \beta_k \hat{\mathbf{p}}_k^{(s)}$;
(15) ENDFOR

4.3 Solution of Coarse Problem

Scalability is a key issue in the implementation of AFETI-DP. The *coarse problem* (Statement 3 and 9 In Algorithm 2) is an especially time-consuming component, therefore its scalability will greatly influence the performance of the whole work. For previous AFETI-DP that used direct solver (i.e., LU factorization) for the coarse problem, the parallel scalability is severely limited. As a remedy, our work uses iterative EDD-PCG for the coarse problem. Because matrix-vector product is the only matrix-related operation and the global assembly is circumvented, the scalability for the tearing process is full preserved. Algorithm 3 describes the EDD-PCG for the coarse problem.

Algorithm 3 *EDD-PCG for the coarse Problem derived from EDD-AFETI-DP where s denotes native sub-domain.*

$$\bigcup_s \left\{ \begin{bmatrix} \mathbf{K}_{cc}^{*(s)} & -\mathbf{F}_{cq}^{(s)} \\ -\mathbf{F}_{cq}^{(s)\mathsf{T}} & -\mathbf{F}_{qq}^{(s)} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{x}}_c^{(s)} \\ \hat{\mathbf{x}}_q^{(s)} \end{bmatrix} \right\} = \bigcup_s \begin{bmatrix} \mathbf{b}_c^{(s)} \\ \mathbf{b}_q^{(s)} \end{bmatrix} \tag{27}$$

$\mathbf{C}$ *is preconditioner.*

(1) **Initial**:
(2) $\begin{bmatrix} \hat{\mathbf{x}}_{c,0}^{(s)} \\ \hat{\mathbf{x}}_{q,0}^{(s)} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$; $\begin{bmatrix} \mathbf{r}_{c,0}^{(s)} \\ \mathbf{r}_{q,0}^{(s)} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_{c,0}^{(s)} \\ \mathbf{b}_{q,0}^{(s)} \end{bmatrix}$;
(2) $\begin{bmatrix} \hat{\mathbf{r}}_{c,0}^{(s)} \\ \hat{\mathbf{r}}_{q,0}^{(s)} \end{bmatrix} = \begin{bmatrix} \bowtie \sum^{\partial\Omega_c^s} \mathbf{r}_{c,0}^{(s)} \\ \bowtie \sum^{\partial\Omega_q^s} \mathbf{r}_{q,0}^{(s)} \end{bmatrix}$;
(3) $\begin{bmatrix} \mathbf{z}_{c,0}^{(s)} \\ \mathbf{z}_{q,0}^{(s)} \end{bmatrix} = \hat{\mathbf{C}}^{(s)} \begin{bmatrix} \mathbf{r}_{c,0}^{(s)} \\ \mathbf{r}_{q,0}^{(s)} \end{bmatrix}$;
(4) $\begin{bmatrix} \hat{\mathbf{z}}_{c,0}^{(s)} \\ \hat{\mathbf{z}}_{q,0}^{(s)} \end{bmatrix} = \begin{bmatrix} \bowtie \sum^{\partial\Omega_c^s} \mathbf{z}_{c,0}^{(s)} \\ \bowtie \sum^{\partial\Omega_q^s} \mathbf{z}_{q,0}^{(s)} \end{bmatrix}$; $\begin{bmatrix} \hat{\mathbf{p}}_{c,0}^{(s)} \\ \hat{\mathbf{p}}_{q,0}^{(s)} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{z}}_{c,0}^{(s)} \\ \hat{\mathbf{z}}_{q,0}^{(s)} \end{bmatrix}$;
(5) $\sigma_0 = \forall \sum^{\Omega_c} \langle \hat{\mathbf{z}}_{c,0}^{(s)}, \mathbf{r}_{c,0}^{(s)} \rangle + \forall \sum^{\Omega_q} \langle \hat{\mathbf{z}}_{q,0}^{(s)}, \mathbf{r}_{q,0}^{(s)} \rangle$;
(6) **Iterate**:
(7) FOR ($k = 0, 1, \cdots,$) till convergence
/* SECTION 1: matrix-vector product: $q_k = Ap_k$ */
(8) $\begin{bmatrix} \mathbf{q}_{c,k}^{(s)} \\ \mathbf{q}_{q,k}^{(s)} \end{bmatrix} = \begin{bmatrix} \mathbf{K}_{cc}^{*(s)} \hat{\mathbf{p}}_{c,k}^{(s)} - \mathbf{F}_{cq}^{(s)} \hat{\mathbf{p}}_{q,k}^{(s)} \\ -\mathbf{F}_{cq}^{(s)\mathsf{T}} \hat{\mathbf{p}}_{c,k}^{(s)} - \mathbf{F}_{qq}^{(s)} \hat{\mathbf{p}}_{q,k}^{(s)}) \end{bmatrix}$;
/* SECTION 2: compute step-length α_k */
(9) $\delta_k = \forall \sum^{\Omega_c} \langle \mathbf{q}_{c,k}^{(s)}, \hat{\mathbf{p}}_{c,k}^{(s)} \rangle + \forall \sum^{\Omega_q} \langle \mathbf{q}_{q,k}^{(s)}, \hat{\mathbf{p}}_{q,k}^{(s)} \rangle$;
(9) $\alpha_k = \frac{\sigma_k}{\delta_k}$;
/* SECTION 3: update solution and residual */

(10) $$\begin{bmatrix} \hat{\mathbf{x}}_{c,k}^{(s)} \\ \hat{\mathbf{x}}_{q,k}^{(s)} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{x}}_{c,k}^{(s)} \\ \hat{\mathbf{x}}_{q,k}^{(s)} \end{bmatrix} + \alpha_k \begin{bmatrix} \hat{\mathbf{p}}_{c,k}^{(s)} \\ \hat{\mathbf{p}}_{q,k}^{(s)} \end{bmatrix};$$

(10) $$\begin{bmatrix} \mathbf{r}_{c,k+1}^{(s)} \\ \mathbf{r}_{q,k+1}^{(s)} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_{c,k}^{(s)} \\ \mathbf{r}_{q,k}^{(s)} \end{bmatrix} - \alpha_k \begin{bmatrix} \mathbf{q}_{c,k}^{(s)} \\ \mathbf{q}_{q,k}^{(s)} \end{bmatrix};$$

/* SECTION 4: preconditioning gradient */

(11) $$\begin{bmatrix} \hat{\mathbf{r}}_{c,k}^{(s)} \\ \hat{\mathbf{r}}_{q,k}^{(s)} \end{bmatrix} = \begin{bmatrix} \bowtie \sum^{\partial\Omega_c^s} \mathbf{r}_{c,k}^{(s)} \\ \bowtie \sum^{\partial\Omega_q^s} \mathbf{r}_{q,k}^{(s)} \end{bmatrix};$$

(11) $$\begin{bmatrix} \mathbf{z}_{c,k}^{(s)} \\ \mathbf{z}_{q,k}^{(s)} \end{bmatrix} = \hat{\mathbf{C}}^{(s)} \begin{bmatrix} \mathbf{r}_{c,k}^{(s)} \\ \mathbf{r}_{q,k}^{(s)} \end{bmatrix};$$

(12) $$\begin{bmatrix} \hat{\mathbf{z}}_{c,k}^{(s)} \\ \hat{\mathbf{z}}_{q,k}^{(s)} \end{bmatrix} = \begin{bmatrix} \bowtie \sum^{\partial\Omega_c^s} \mathbf{z}_{c,k}^{(s)} \\ \bowtie \sum^{\partial\Omega_q^s} \mathbf{z}_{q,k}^{(s)} \end{bmatrix};$$

/* SECTION 5: update search direction p_k */

(13) $$\sigma_{k+1} = \forall \sum^{\Omega_c} \langle \mathbf{r}_{c,k+1}^{(s)}, \hat{\mathbf{z}}_{c,k+1}^{(s)} \rangle + \forall \sum^{\Omega_q} \langle \mathbf{r}_{q,k+1}^{(s)}, \hat{\mathbf{z}}_{q,k+1}^{(s)} \rangle; \quad \beta_k = \frac{\sigma_{k+1}}{\sigma_k};$$

(14) $$\begin{bmatrix} \hat{\mathbf{p}}_{c,k+1}^{(s)} \\ \hat{\mathbf{p}}_{q,k+1}^{(s)} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{z}}_{c,k+1}^{(s)} \\ \hat{\mathbf{z}}_{q,k+1}^{(s)} \end{bmatrix} + \beta_k \begin{bmatrix} \hat{\mathbf{p}}_{c,k}^{(s)} \\ \hat{\mathbf{p}}_{q,k}^{(s)} \end{bmatrix};$$

(15) ENDFOR

The preconditioner for the coarse problem is discussed in next section.

5 Polynomial Preconditioner for Coarse Problem

5.1 Diagonal Preconditioner

The iterative solver for the coarse problem employs a diagonal preconditioner to accelerate the converge performance, namely,

$$\begin{cases} \hat{\mathbf{C}}^{(s)} = diag(\frac{1}{\hat{d}_{c,1}^{(s)}}, \cdots, \frac{1}{\hat{d}_{c,N_c^s}^{(s)}}, \frac{1}{\hat{d}_{q,1}^{(s)}}, \cdots, \frac{1}{\hat{d}_{q,N_q^s}^{(s)}}), \\ [\hat{d}_{c,1}^{(s)}, \cdots, \hat{d}_{c,N_c^s}^{(s)}]^{\mathsf{T}} = \bowtie \sum^{\partial\Omega_c^s} [d_{c,1}^{(s)}, \cdots, d_{c,N_c^s}^{(s)}]^{\mathsf{T}}, d_{c,j}^{(s)} = \mathbf{K}_{cc}^{*(s)}[j,j], \\ [\hat{d}_{q,1}^{(s)}, \cdots, \hat{d}_{q,N_q^s}^{(s)}]^{\mathsf{T}} = \bowtie \sum^{\partial\Omega_q^s} [d_{q,1}^{(s)}, \cdots, d_{q,N_q^s}^{(s)}]^{\mathsf{T}}, d_{q,k}^{(s)} = -\mathbf{F}_{qq}^{(s)}[k,k]. \end{cases} \tag{28}$$

For small scale coarse problem, where the number of corners and virtual corner is relatively small, the diagonal preconditioner is very effective. In the case of large coarse problem, polynomial preconditioner, a highly sclable effective preconditioner will be employed.

5.2 Generalized Least-squares Preconditioner

Let $\tilde{\mathbf{K}}^{*(s)} = [\tilde{\mathbf{k}}_1^{*(s)}, \tilde{\mathbf{k}}_2^{*(s)}, \cdots, \tilde{\mathbf{k}}_{N_c^s+N_q^s}^{*(s)}]^\mathsf{T}$, then the norm-1 diagonal scaling matrix $\mathbf{D}$ can be formulated in the global distributed format, i.e., $\mathbf{D} = \bigcup_{s=1}^{\mathcal{P}} \hat{\mathbf{D}}^{(s)}$. Then

$$\hat{\mathbf{D}}^{(s)} = \begin{bmatrix} \hat{\mathbf{D}}_c^{(s)} & 0 \\ 0 & \hat{\mathbf{D}}_q^{(s)} \end{bmatrix}, \tag{29}$$

$$\hat{\mathbf{D}}_c^{(s)} = \mathrm{diag}(\frac{1}{\sqrt{\hat{d}_{c,1}^{(s)}}}, \ldots, \frac{1}{\sqrt{\hat{d}_{q,N_c^s}^{(s)}}}), \tag{30}$$

$$\hat{\mathbf{D}}_q^{(s)} = \mathrm{diag}(\frac{1}{\sqrt{\hat{d}_{q,1}^{(s)}}}, \ldots, \frac{1}{\sqrt{\hat{d}_{q,N_q^s}^{(s)}}}), \tag{31}$$

where

$$[\hat{d}_{c,1}^{(s)}, \cdots, \hat{d}_{c,N_c^s}^{(s)}]^\mathsf{T} = \bowtie \sum^{\partial\Omega_c^s} [d_{c,1}^{(s)}, \cdots, d_{c,N_c^s}^{(s)}]^\mathsf{T}, \tag{32}$$

$$d_{c,i}^{(s)} = \sum_{j=1}^{N_c^s} \mid \tilde{\mathbf{K}}_{cc}^{*(s)}[i,j] \mid + \sum_{j=1}^{N_q^s} \mid \mathbf{F}_{cq}^{*(s)}[i,j] \mid \tag{33}$$

$$[\hat{d}_{q,1}^{(s)}, \cdots, \hat{d}_{q,N_c^s}^{(s)}]^\mathsf{T} = \bowtie \sum^{\partial\Omega_q^s} [d_{q,1}^{(s)}, \cdots, d_{q,N_c^s}^{(s)}]^\mathsf{T}, \tag{34}$$

$$d_{q,i}^{(s)} = \sum_{j=1}^{N_c^s} \mid \mathbf{F}_{cq}^{*(s)}[j,i] \mid + \sum_{j=1}^{N_q^s} \mid \mathbf{F}_{qq}^{*(s)}[i,j] \mid . \tag{35}$$

Employing norm-1 diagonal scaling matrix $\mathbf{D}$, the original coarse problem is transformed into

$$\bigcup_s \left\{ \mathbf{A}^{(s)} \begin{bmatrix} \hat{\mathbf{y}}_c^{(s)} \\ \hat{\mathbf{y}}_q^{(s)} \end{bmatrix} \right\} = \bigcup_s \begin{bmatrix} \mathbf{g}_q^{(s)} \\ \mathbf{g}_c^{(s)} \end{bmatrix} \tag{36}$$

where

$$\mathbf{A}^{(s)} = \hat{\mathbf{D}}^{(s)} \tilde{\mathbf{K}}_{cc}^{*(s)} \hat{\mathbf{D}}^{(s)} = \begin{bmatrix} \hat{\mathbf{D}}_c^{(s)} \tilde{\mathbf{K}}_{cc}^{*(s)} \hat{\mathbf{D}}_c^{(s)} & -\hat{\mathbf{D}}_c^{(s)} \mathbf{F}_{cq}^{*(s)} \hat{\mathbf{D}}_q^{(s)} \\ -\hat{\mathbf{D}}_q^{(s)} \mathbf{F}_{cq}^{*(s)^\mathsf{T}} \hat{\mathbf{D}}_c^{(s)} & -\hat{\mathbf{D}}_q^{(s)} \mathbf{F}_{qq}^{*(s)} \hat{\mathbf{D}}_q^{(s)} \end{bmatrix}$$

and

$$\begin{bmatrix} \hat{\mathbf{y}}_c^{(s)} \\ \hat{\mathbf{y}}_q^{(s)} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{D}}_c^{(s)} & 0 \\ 0 & \hat{\mathbf{D}}_q^{(s)} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{x}}_c^{(s)} \\ \hat{\mathbf{x}}_q^{(s)} \end{bmatrix}, \quad \begin{bmatrix} \hat{\mathbf{g}}_c^{(s)} \\ \hat{\mathbf{g}}_q^{(s)} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{D}}_c^{(s)} & 0 \\ 0 & \hat{\mathbf{D}}_q^{(s)} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{b}}_c^{(s)} \\ \hat{\mathbf{b}}_q^{(s)} \end{bmatrix}.$$

Assuming that $\Theta = (0, 1)$, based on it generalized least-squares (GLS) polynomial preconditioner can be utilized. In summary, a complete implementation of polynomial preconditioned PCG method for coarse problem (27) is given below:

Algorithm 4 *Polynomial preconditioned CG for coarse problem:*

(1) $\begin{bmatrix} \tilde{\mathbf{K}}_{cc}^{*(s)} & -\mathbf{F}_{cq}^{*(s)} \\ -\mathbf{F}_{cq}^{*(s)\mathsf{T}} & -\mathbf{F}_{qq}^{*(s)} \end{bmatrix} = \hat{\mathbf{D}}^{(s)} \begin{bmatrix} \tilde{\mathbf{K}}_{cc}^{*(s)} & -\mathbf{F}_{cq}^{*(s)} \\ -\mathbf{F}_{cq}^{*(s)\mathsf{T}} & -\mathbf{F}_{qq}^{*(s)} \end{bmatrix} \hat{\mathbf{D}}^{(s)}$, $\begin{bmatrix} \mathbf{b}_c^{(s)} \\ \mathbf{b}_q^{(s)} \end{bmatrix} = \hat{\mathbf{D}}^{(s)} \begin{bmatrix} \mathbf{b}_c^{(s)} \\ \mathbf{b}_q^{(s)} \end{bmatrix}$;

(2) construct polynomial preconditioner P_m based on $\Theta = (0, 1)$;

(3) solve $\begin{bmatrix} \tilde{\mathbf{K}}_{cc}^{*(s)} & -\mathbf{F}_{cq}^{*(s)} \\ -\mathbf{F}_{cq}^{*(s)\mathsf{T}} & -\mathbf{F}_{qq}^{*(s)} \end{bmatrix} \begin{bmatrix} \mathbf{y}_c^{(s)} \\ \mathbf{y}_q^{(s)} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_c^{(s)} \\ \mathbf{b}_q^{(s)} \end{bmatrix}$ using Algorithm 3 with $\mathbf{C}^{(s)} = P_m(\tilde{\mathbf{K}}_{cc}^{(s)})$;

(4) $\begin{bmatrix} \mathbf{x}_c^{(s)} \\ \mathbf{x}_q^{(s)} \end{bmatrix} = \hat{\mathbf{D}}^{(s)} \begin{bmatrix} \mathbf{y}_c^{(s)} \\ \mathbf{y}_q^{(s)} \end{bmatrix}$;

6 Selection of Corner and Virtual-corner

6.1 Selection of Corners

Corner selection [16] is an important issue for the AFETI-DP. Large number of corner points picked may lead to faster convergence behavior while a higher computational cost is also incurred. Thus for large problems, the number of corners should be limited by satisfying the following two conditions: (1) Each remainder sub-domain stiffness matrix, $\mathbf{K}rr^{(s)}$, should be non-singular. (2) The augmented coarse problem matrix, $\tilde{\mathbf{K}}_{cc}^{*(s)}$, should be non-singular. Non-singularity of $\mathbf{K}_{rr}^{(s)}$ can be guaranteed by making each sub-domain have 3 non-collinear corner nodes in three-dimensions or 2 non-coincidental corner nodes in two-dimension mesh. The non-singularity of $\tilde{\mathbf{K}}_{cc}^{*(s)}$ is discussed in [16].

Definition 1. *The* degree of connectivity *(DOC) of a node is defined as the number of sub-domains which share it. Of all the interface nodes except the corner nodes, the maximum degree of connectivity is denoted as maxDOC.*

Based on the above definition, in our implementation a node is selected as corner if either of the following conditions is satisfied:

1. A node k is defined as corner node if $DOC(k) = maxDOC$.
2. A node k is defined as corner if it is located on the begin or the end of the edge between sub-domains.

Figure 3(a) shows the corners picked for a $2 \times 2 \times 2$ domain decomposition problem (discretized by 3-D brick element).

6.2 Selection of Virtual Corners

For simplicity of the selection of virtual corner, i.e., load-balance control and consistency between adjacent sub-domain, an adjacency graph (a kind of directed graph) is used to represent the relationship among the sub-domains. Figures 3(b) and 4 (a)(b) show the adjacency graph of 2-D, $4 \times 1 \times 1$ and $2 \times 2 \times 2$ mesh problems. For each adjacent couple, the sub-domain with smaller sub-domain number is defined as the *father adjacent sub-domain* (FAS) and the other is defined as

the *son adjacent sub-domain* (SAS). It should be noted that, in our implementation only FAS is authorized to select virtual corners from specified edge, thus the balance-control of virtual corner (i.e., each edge contributes approximately equal number of virtual corners) and consistency between adjacent sub-domains become much easy. Further, with regard to the effective convergence of coarse problem, the number of virtual corner should be limited. Let us define the maximum DOC of non-corner nodes as $maxDOC\_NC$. In this work a node is likely to be picked only when its degree of connectivity is equal to $maxDOC\_NC$. With Figure 3(b) as example, since $maxDOC\_NC = 2$, the virtual corners can be picked along the border between adjacent sub-domains. $maxDOC\_NC$ for Figure 4(a) is also equal to 2. The virtual corners are picked from the overlapping area between the adjacent 3-D sub-domains. Similarly, since $maxDOC\_NC = 4$ in Figure 4(b), the virtual corners are only picked along the edges of a 3-D sub-domain. Selection of virtual corner is described by Algorithm 5. Once $VcList^{(s)}$ and $nVc^{(s)}$ is obtained, Q^s_{br} will be constructed such that ${Q^s_{br}}^{\mathsf{T}} B^s_r u^s_r = u^s_q$, or equally, ${Q^s_{br}}^{\mathsf{T}} u^s_{br} = u^s_q$.

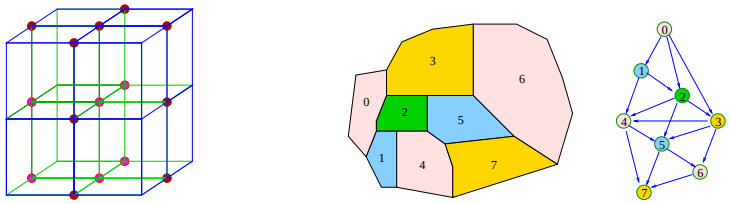

Fig. 3. (a) Corners chosen for $2 \times 2 \times 2$ mesh; (b) Adjancy graph for 2-D mesh.

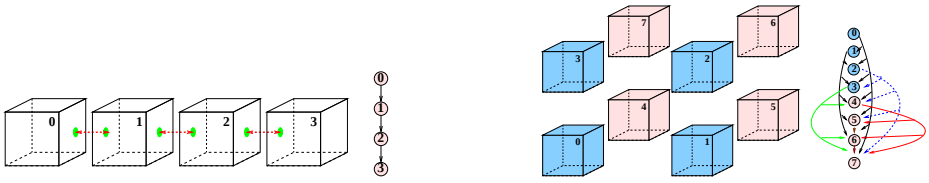

Fig. 4. Adjacency graph of: (a) $4 \times 1 \times 1$ mesh; (b) $2 \times 2 \times 2$ mesh.

Algorithm 5 *Selection of virtual corners. nVc is the number of local virtual corners and $VcList$ stores the local virtual corners. $STEP$ is a given constant. s is the ID of the native sub-domain (i.e., calling process), fs is the ID of one FAS and ss is the ID of one SAS.*

(1) `Construct the tree-structure of the mesh;`
(2) `Compute` $maxDOC\_NC$;
(3) $nVc^{(s)} = 0$; $VcList^{(s)} = \{\}$; $count = 0$;

```
(4)   FOR (inode ∈ ∂Ω_s \ ∂Ω) DO
(5)       IF ((inode is not included in FAS(s) and
(6)           (inode is not chosen as corner) and
(7)           (inode has not yet been chosen as virtual corners) and
(8)           (DOC(inode) = maxDOC_NC)) THEN
(9)           If (count = STEP) Then
(10)              VcList^(s) = VcList^(s) ∪ {inode}; nVc^(s) = nVc^(s) + 1;
(11)              count = 0;
(12)          EndIf
(13)          count = count + 1;
(14)      ENDIF
(15)  ENDFOR
(16)  FOR (ss ∈ SAS(s)) DO
(17)      send native virtual corner information (nVc^(s),VcList^(s)) to ss;
(18)  ENDFOR
(19)  FOR (fs ∈ FAS(s)) DO
(20)      receive virtual corner information (nVc^(fs),VcList^(fs)) from fs;
(21)      nVc^(s) = nVc^(s) + nVc^(fs); VcList^(s) = VcList^(s) ∪ VcList^(fs);
(22)  ENDFOR
```

7 GInO Method for Structural Dynamics

Non-linear dynamic problems, which are generally formulated as a single-field, second-order simultaneous ordinary differential equation system, yield the following after the finite element discretization in space (semi-discrete):

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{P}(\dot{\mathbf{u}}, \mathbf{u}) = \mathbf{f}; \mathbf{u}(0) = \mathbf{u}_0, \ddot{\mathbf{u}}(0) = \ddot{\mathbf{u}}_0. \tag{37}$$

Here $\mathbf{M} \in \Re^{N \times N}$ is the mass matrix, $\mathbf{P}$ is the vector of internal nodal forces, $\ddot{\mathbf{u}}$, $\dot{\mathbf{u}}$ and $\mathbf{u}$ are the vectors of nodal accelerations, velocities and displacement respectively, $\mathbf{f}$ is the vector of external applied forces. Considering the elasto-plastic material behavior [10] enables us to approximate (37) as

$$\mathbf{M}\ddot{\mathbf{u}}(t) + \mathbf{C}_t\dot{\mathbf{u}}(t) + \mathbf{K}_t(\mathbf{u})\mathbf{u}(t) = \mathbf{f}(t) \tag{38}$$

where $\mathbf{C}_t$ is independent of t. The family of generalized integration operators (GInO) [10, 12] for a single field form of representation arises by defining the weighted time field as a degenerated scalar polynomial function:

$$W(\tau) = 1 + w_1\frac{\tau}{\Delta t} + w_2(\frac{\tau}{\Delta t})^2 + w_3(\frac{\tau}{\Delta t})^3; \tag{39}$$

where $\tau + t_n = t$, $\Delta t = t_{n+1} - t_n$ and $\tau \in [0, \Delta t]$. Using the GInO formulation, (38) can be re-formulated as the time weighted residual representation (weak form):

$$\int_0^{\Delta t} W[\mathbf{M}\ddot{\mathbf{u}}(\tau+t_n)+\mathbf{C}_t\dot{\mathbf{u}}(\tau+t_n)+\mathbf{K}_t(\bar{\mathbf{u}})\mathbf{u}(\tau+t_n)-\mathbf{f}(\tau+t_n)]d\tau = \mathbf{0},\ \forall\tau \in [0, \Delta t], \tag{40}$$

where $\bar{\mathbf{u}}$ is a time weighted average of the displacement field [12]. Further, using an asymptotic series expansion

$$\mathbf{u}(\tau+t_n) \approx \mathbf{u}_n + \Lambda_1 \dot{\mathbf{u}}_n \tau + \Lambda_2 \ddot{\mathbf{u}}_n \tau^2 + \Lambda_3 \frac{\ddot{\mathbf{u}}_{n+1} - \ddot{\mathbf{u}}_n}{\Delta t} \tau^3 \tag{41}$$

$$\dot{\mathbf{u}}(\tau+t_n) \approx \mathbf{u}_n + \Lambda_4 \ddot{\mathbf{u}}_n + \Lambda_5 \frac{\ddot{\mathbf{u}}_{n+1} - \ddot{\mathbf{u}}_n}{\Delta t} \tau^2, \tag{42}$$

$$\ddot{\mathbf{u}}(\tau+t_n) \approx \ddot{\mathbf{u}}_n + \Lambda_6 \frac{\ddot{\mathbf{u}}_{n+1} - \ddot{\mathbf{u}}_n}{\Delta t} \tau, \ \textit{and} \tag{43}$$

$$\mathbf{f}(\tau+t_n) \approx \mathbf{f}_n + \frac{\mathbf{f}_{n+1} - \mathbf{f}_n}{\Delta t} \tau, \tag{44}$$

it follows from (40) that

$$\begin{aligned} &\mathbf{M}(\ddot{\mathbf{u}}_n + \Lambda_6 W_1 \Delta \ddot{\mathbf{u}}_{n+1}) + \mathbf{C}_t(\dot{\mathbf{u}}_n + \Lambda_4 W_1 \ddot{\mathbf{u}}_n \Delta t + \Lambda_5 W_2 \Delta \ddot{\mathbf{u}}_{n+1} \Delta t) + \\ &\mathbf{K}_t(\bar{\mathbf{u}})(\mathbf{u}_n + \Lambda_1 W_1 \dot{\mathbf{u}}_n + \Lambda_2 W_1 \ddot{\mathbf{u}}_n \Delta t + \Lambda_3 W_3 \Delta \ddot{\mathbf{u}}_{n+1} \Delta t^2) \\ &= (1 - W_1)\mathbf{f}_n + W_1 \mathbf{f}_{n+1} \end{aligned} \tag{45}$$

where $W_i = \frac{\sum_{j=0}^{3} \frac{w_j}{1+i+j}}{\sum_{j=0}^{3} \frac{w_j}{1+j}} (i = 1, 2, 3)$ and $\Delta \ddot{\mathbf{u}}_{n+1} = \ddot{\mathbf{u}}_{n+1} - \ddot{\mathbf{u}}_n$. Using (41) and (42) to eliminate $\mathbf{u}_{n+1}$ and $\dot{\mathbf{u}}_{n+1}$ in (45), it follows that

$$\begin{aligned} &(\Lambda_6 W_1 \mathbf{M} + \Lambda_5 W_2 \mathbf{C} \Delta t + \Lambda_3 W_3 \mathbf{K}_t \Delta t^2) \Delta \ddot{\mathbf{u}}_{n+1} = -\mathbf{M}\ddot{\mathbf{u}}_n - \mathbf{C}_t(\ddot{\mathbf{u}}_n + \Lambda_4 W_1 \ddot{\mathbf{u}}_n \Delta t) \\ &- \mathbf{K}_t(\bar{\mathbf{u}})(\mathbf{u}_n + \Lambda_1 W_1 \dot{\mathbf{u}} \Delta t + \Lambda_2 W_2 \ddot{\mathbf{u}}_n \Delta t^2) + (1 - W_1)\mathbf{f}_n + W_1 \mathbf{f}_{n+1} \end{aligned} \tag{46}$$

Due to the existence of non-linear term $\mathbf{K}_t(\bar{\mathbf{u}})$, a Newton-Raphson strategy is used to solve (46) where the approximation to acceleration, displacement and velocity are corrected using (41), (42) and (43). Once the approximate $\ddot{\mathbf{u}}_{n+1}$ is obtained, $\ddot{\mathbf{u}}_{n+1}$, $\dot{\mathbf{u}}_{n+1}$ and $\mathbf{u}_{n+1}$ can be directly computed using

$$\begin{cases} \mathbf{u}_{n+1} = \mathbf{u}_n + \lambda_1 \dot{\mathbf{u}}_n \Delta t + \lambda_2 \ddot{\mathbf{u}}_n \Delta t^2 + \lambda_3 \Delta \ddot{\mathbf{u}}_{n+1} \Delta t^2, \\ \dot{\mathbf{u}}_{n+1} = \mathbf{u}_n + \lambda_4 \ddot{\mathbf{u}}_n + \lambda_5 \Delta \ddot{\mathbf{u}}_{n+1} \Delta t, \\ \ddot{\mathbf{u}}_{n+1} = \ddot{\mathbf{u}}_n + \lambda_6 \Delta \ddot{\mathbf{u}}_{n+1}. \end{cases} \tag{47}$$

The implementation of the whole computation is given in Algorithm 6. Here it should be remarked that in Algorithm 6, twelve sets of free scalar parameters, $\Lambda_6 W_1$, $\Lambda_5 W_2$, $\Lambda_3 W_3$, $\Lambda_4 W_1$, $\Lambda_1 W_1$, $\Lambda_2 W_2$, W_1, λ_1, λ_2, λ_3, λ_4 and λ_5, are defined based on the following issues: dissipation and dispersion properties, overshooting behavior, second-order accuracy and unconditional stability, etc. An appropriate choice of these parameters will make computation A-stable. Using them as the algorithmic markers, different methods arise.

Algorithm 6 *Unified predictor multi-corrector incremental single step LMS method.* $\mathbf{u}_n$, $\dot{\mathbf{u}}_n$, $\ddot{\mathbf{u}}_n$, Δt *and free scalar parameters are given.*

(1) FOR $n = 0, 1, \cdots$ DO /* Time-step iteration */
(2) /* PART 2: predict state vectors using predictors */
(3) $\tilde{\mathbf{u}}^0_{n+1} = \mathbf{u}_n + \Lambda_1 W_1 \dot{\mathbf{u}}_n \Delta t + \Lambda_2 W_2 \ddot{\mathbf{u}}_n \Delta t^2$;
(4) $\dot{\tilde{\mathbf{u}}}^0_{n+1} = \dot{\mathbf{u}}_n + \Lambda_4 W_1 \ddot{\mathbf{u}}_n \Delta t$;
(5) $\ddot{\tilde{\mathbf{u}}}^0_{n+1} = \ddot{\mathbf{u}}_n$;
(6) /* PART 3: multi-correction (Newton-Raphson) iteration */
(7) For $j = 0, 1, \cdots$ DO
(8) /* Section 3.1: linear equations */
(9) $\bar{\mathbf{M}}_t = \Lambda_6 W_1 \mathbf{M} + \Lambda_5 W_2 \Delta t \mathbf{C}_t + \Lambda_3 W_3 \Delta t^3 \mathbf{K}_t$;
(10) $\bar{\mathbf{F}} = \bar{\tilde{\mathbf{F}}}_{n+1} - \mathbf{M}\ddot{\tilde{\mathbf{u}}}^j_{n+1} - \mathbf{P}(\dot{\tilde{\mathbf{u}}}^j_{n+1}, \tilde{\mathbf{u}}^j_{n+1})$;
(11) $\bar{\mathbf{M}}_t \Delta \ddot{\mathbf{u}}^{j+1}_{n+1} = \bar{\mathbf{F}}$;
(12) /* Section 3.2: correctors*/
(13) $\tilde{\mathbf{u}}^{j+1}_{n+1} = \tilde{\mathbf{u}}^j_{n+1} + \Lambda_3 W_3 \Delta \ddot{\mathbf{u}}^{j+1}_{n+1} \Delta t^2$;
(14) $\dot{\tilde{\mathbf{u}}}^{j+1}_{n+1} = \dot{\tilde{\mathbf{u}}}^j_{n+1} + \Lambda_5 W_2 \Delta \ddot{\mathbf{u}}^{j+1}_{n+1} \Delta t$;
(15) $\ddot{\tilde{\mathbf{u}}}^{j+1}_{n+1} = \ddot{\tilde{\mathbf{u}}}^j_{n+1} + \Lambda_6 W_1 \Delta \ddot{\mathbf{u}}^{j+1}_{n+1}$;
(16) /* Section 3.3: recompute $\mathbf{K}$ */
(17) compute $\mathbf{K}(\tilde{\mathbf{u}})^j_{n+1}$;
(18) IF ($\|\bar{\mathbf{F}}_{n+1}\| \leq \epsilon \|\bar{\mathbf{f}}_{n+1}\|$) convergence occurs;
(19) End For
(20) /* PART 4: Design updates */
(21) $\ddot{\mathbf{u}}_{n+1} = \ddot{\mathbf{u}}_n + \lambda_6(\ddot{\mathbf{u}}^{j+1}_{n+1} - \ddot{\mathbf{u}}_n)$;
(22) $\dot{\mathbf{u}}_{n+1} = \dot{\mathbf{u}}_n + \lambda_4 \dot{\mathbf{u}} \Delta t + \lambda_5(\ddot{\mathbf{u}}_{n+1} - \ddot{\mathbf{u}}_n)\Delta t$;
(23) $\mathbf{u}_{n+1} = \mathbf{u}_n + \lambda_1 \dot{\mathbf{u}} \Delta t + \lambda_2 \ddot{\mathbf{u}} \Delta t^2 + \lambda_3(\ddot{\mathbf{u}}_{n+1} - \ddot{\mathbf{u}}_n)\Delta t^2$;
(24) END FOR

Solution of linear system $\bar{\mathbf{M}}_t \Delta \ddot{\mathbf{u}}^{j+1}_{n+1} = \bar{\mathbf{F}}$ (Statement 11) is the key to Algorithm 6. AFETI-DP addressed in the previous sections will be applied here to solve it.

8 Experimental Results

With non-linear dynamic simulation of 2-D and 3-D cantilever-beam as examples, our experiments mainly focus on evaluating the effectness of GLS polynomial preconditioning in solving the coarse problem and the *A-stability* [10] of the AFETI-DP.

Figures 5-6 demonstrate the convergence rate of GLS preconditioned conjugate gradient method for coarse problem, which is the kernel operation of AFETI-DP. It is observed that, for the coarse problems derived from both 80×80 and 100×100 cantilever-beam dynamic problems,

$$PCGGLS(10) \succ PCGGLS(5) \succ PCG, \tag{48}$$

where "$\succ$" indicates "convergs faster". (48) implies that a higher degree of GLS polynomial preconditioning may bring more reduction of iteration number for convergence. However, due to the factor of stability of polynomial preconditioning, a high degree polynomial preconditioner is not recommended.

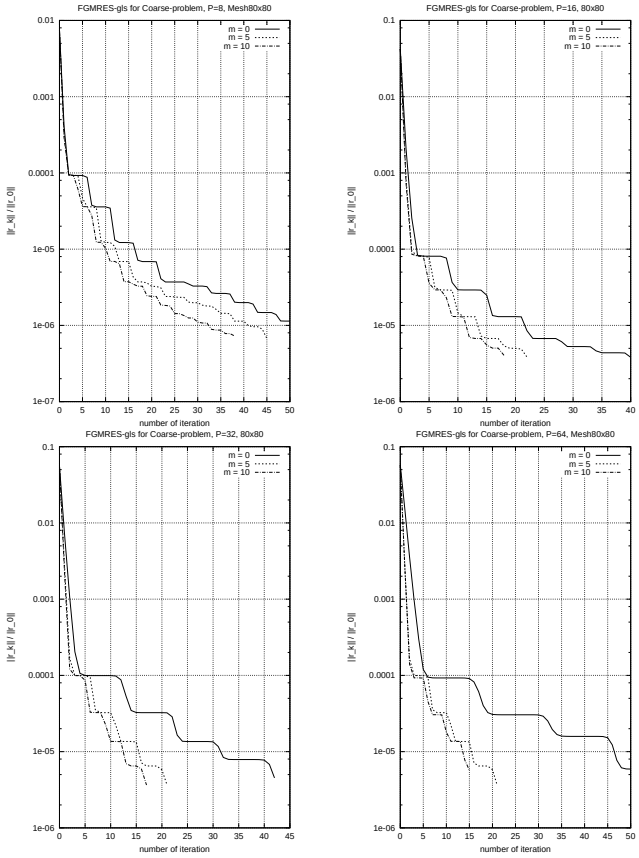

Fig. 5. IBM-SP2:PCG-gls for Coarse Problem of Mesh80x80 Problem: (a) $\mathcal{P} = 8$; (b) $\mathcal{P} = 16$; (c) $\mathcal{P} = 32$;(d) $\mathcal{P} = 64$.

Figures 7-8 demonstrate the numerical and parallel scalability of AFETI-DP with respect to 80×80 2-D cantilever-beam problem which are discretized by four-node isoperimetrical quadrilateral elements. According to Algorithm 5, the number of virtual corners per edge is controllable. Figure 7 shows that, as $\mathcal{P}$ increases (mesh size fixed), the number of convergence iteration for interface problem does not vary greatly. From 8 it is observed that AFETI-DP behaviors super-linear parallel speedup on the IBM-SP2 in the case of small-processor-count.

Finally, it should be remarked that the experiment is not fully completed. The numerical test for the situation of 3-D problems and large-processor-count HPC system (i.e., $\mathcal{P} \geq 1000$) will be provided in the final version of manuscript to demonstrate the A-scalability in the AFETI-DP formulation proposed here.

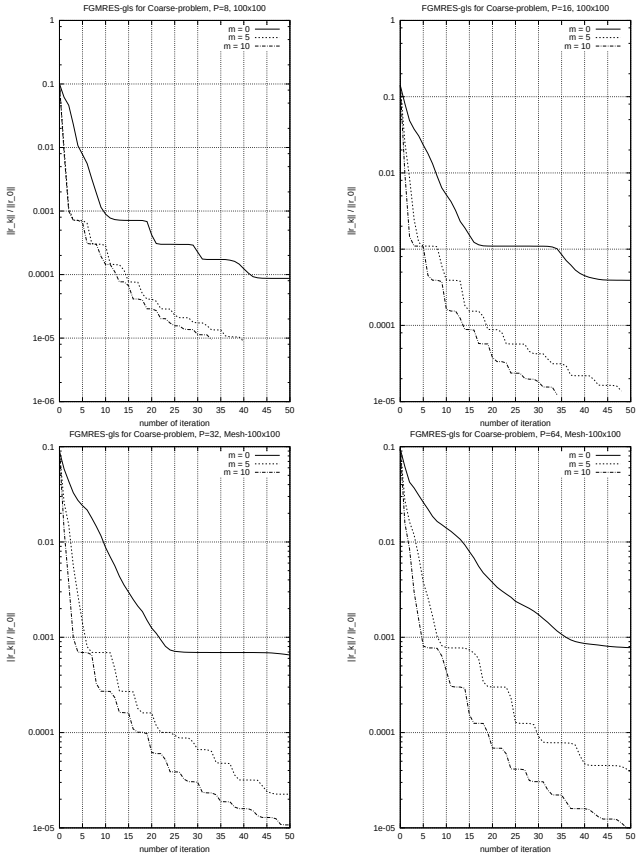

Fig. 6. IBM-SP2:PCG-gls for Coarse Problem of Mesh80x80 Problem: (a) $\mathcal{P} = 8$; (b) $\mathcal{P} = 16$; (c) $\mathcal{P} = 32$; (d) $\mathcal{P} = 64$.

9 Conclusions

An efficient augmented FETI-DP (AFETI-DP) based on element-based polynomial preconditioned [19] conjugate gradient algirithm [23] (for coarse problem) is presented and then applied in the implementation of generalized integration operator (GInO) method for large-scale non-linear structural dynamics problems (i.e., cantilever beam), Illustrated by the experimental results on IBM-SP2, the application of polynomial preconditioning technique reduces the timing-cost by accelerating the convergence rate of iterative solver, particularly, it preserves the absolute scalability (including numerical, parallel and memory utilization scalability) inherited from the domain decomposition strategy.

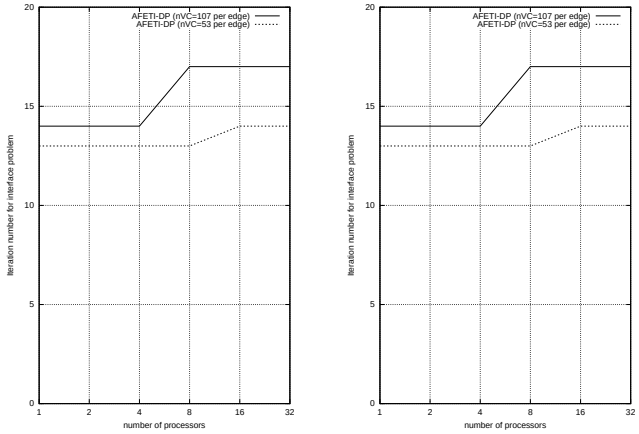

Fig. 7. IBM-SP2:Numerical scalability about AFETI-DP (tolerance $\epsilon = 10^{-6}$) for: (a)Mesh80x80;(b)Mesh100x100.

References

1. Farhat C. and Roux F.-X (1991), A Method of Finite ELement Tearing and Interconnecting and Its Parallel Solution Algorithm, Int. J. Numer. Mech. Eng., 32:1205-1227.
2. Farhat C.(1991), *Lagrange Multiplier Based Divided and Conquer Finite Element Algorithm*, J. Comput. Sys. Eng., 2:149-156.
3. Farhat C., P. S. Chen, and J. Mandel (1995), *A Scalable Lagrange Multiplier Based Domain Decomposition Method for Time-dependent Problems*, Int. J. Numer. Meth. Eng., 38:3831-3853.
4. Farhat C. and Maman N.(1995), *The two-level FETI method for static and dynamic plate problems - Part I: An optimal iterative solver for biharmonic systems*. Technical Report UCB/CAS Report CU-CAS-95-23, Center for Aerospace Structures, University of Colorado at Boulder.
5. Farhat C. and Mandel J. (1995), *FETI method - Part II: Extension to shell problems, parallel implementation and performance results*. Comp. Meth. Appl. Mech. Engrg.
6. Charbel Farhat, et. al. (1998), *A unified framework for accelerating the convergence of iterative substructuring methods with Lagrange multipliers*, International Journal for Numerical Methods in Engineering Volume 42, Issue 2, 257-288.
7. Farhat C. et al. (1999), *FETI-DP: A Dual Primal Unified FETI Method. Part I: A Faster Alternative To the Two-Level FETI Method*, Tech. Rep. CU-CAS-99-15, Center for Aerospace Structures, University of Colorado At Boulder, August.
8. Farhat C., P. S. Chen, and J. Mandel (2000), *A Scalable Dual-Primal Domain Decomposition Method*, Numerical Linear Algebra with Applications, 7:687-714.
9. Ramdev Kanapady (2001), A Unified Mathematical Framework of Continuous and Discontinuous Time Integrators and Domain Decomposition Techniques for Scalable High Performance Structural Dynamics Simulations, Ph.D thesis, University of Minnesota.

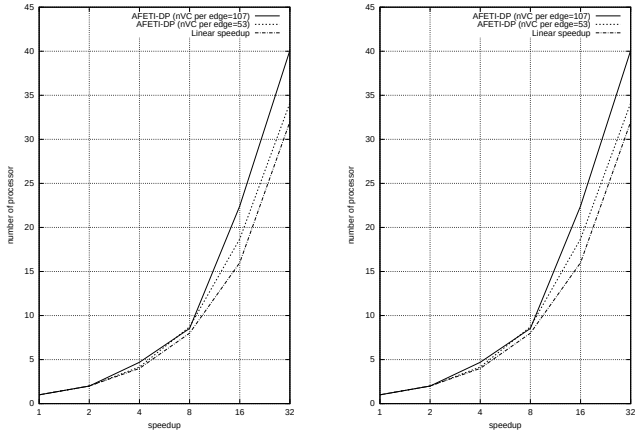

Fig. 8. IBM-SP2:Parallel scalability about AFETI-DP (tolerance $\epsilon = 10^{-6}$) for: (a)Mesh80x80; (b)Mesh100x100.

10. R. Kanapady and K. K. Tamma (2000), *A Unified Family of Generalized Integration Operators [GInO] for Non-linear Structural Dynamics: Implementation Aspects.* Advances in Engineering Software, 31:639-647.
11. R. Kanapady, K. K. Tamma and A. Mark (1999), Highly Scalable Parallel Computational Models for Large-scale RTM Process Modeling Simulations, PART 2: Parallel Formulation Theory and Implementation Numerical Heat Transfer Part B: Fundamentals, Vol.36 Numb.3, 287-308.
12. R. Kanapady and K. K. Tamma (2003), *A*-Scalability of an Integrated Computational Technology and Framework for Non-linear Structural Dynamics - Part I Theoretical Developments and Parallel Formulations, Int. J. Numer. Methods Engrg (in press).
13. Axel Klawonn, Olof B. Widlund (2000), *A Domain Decomposition Method with Lagrange Multipliers and Inexact Solvers for Linear Elasticity*, SIAM J. Scientific Computing, Vol.22, No.4, 1199-1219.
14. A. Klawonn, O.B. Widlund, and M. Dryja (2002), *Dual-Primal FETI Methods for Three-Dimensional Elliptic Problems with Heterogeneous Coefficients*, SIAM J. Numer. Anal., Vol. 40, No. 1, 159-179.
15. K. H. Law (1988), *A Parallel Finite ELement Solution Method*, Computer and Structure, Vol. 23, No. 6, 845-869.
16. M. Lesoinne (2002), *A FETI-DP Corner Selection ALgorithm for Three-dimensional Problems*, proceedings of "Fourteenth International Converence on Domain Decomposition Methods", Ed. Ismael Herrera, et al.
17. Y. Liang, J. Weston and M. Szularz (2002), *Adaptive Generalised Least-squares Polynomial Preconditioner for Symmetric Indefinite Linear Equations*, Parallel Computing, Vol.28, Issue 2, 323-341.
18. Y. Liang, J. Weston and M. Szularz, *Stability of Polynomial Preconditioning*, in Proceedings of ALGORITMY 2000, 15th Conference on Scientific Computing, A. Handlovicova, M.Komornikova, K.Mikula, and D.Sevcovic(Eds.), 264-273.
19. Y. Liang, R. Kanapady, K. K, Tamma (2004), *An Efficient Parallel Finite-element-based Domain Decomposition Iterative Solvers With Polynomial Preconditioning*, Journal of Computer Modelling in Engineering & Science. (in review).

20. J. Mandel and R. Tezaur (2001), *On the Convergence of a Dual-primal Substructuring Method*. Numer. Math., 88:543-558.
21. K. H. Pierson (2000), A Family of Domain Decomposition Methods for the Massively Parallel SOlution of Computational Mechanics Problem, Ph.D thesis, University of Colorado.
22. Daniel Rixen and Charbel Farhat (1998), *Preconditioning the FETI Method for Problems with Intra- and Inter-Subdomain Coefficient Jumps*, Proceedings of Ninth International Concerence on Domain Decomposition Methods, Editor Peter E. Bjorstad, et. al., 472-479.
23. Y. Saad (1996), *Iterative Methods for Sparse Linear Systems*, PWS Publishing.
24. O. B. Widlund, A. Klawonn and M. Dryja (2002), *Dual-Primal FETI Methods for Three-dimensional Elliptic Problems with Heterogeneous Coefficients*, SIAM J. Numer. Anal., 40:159-179.
25. D. Sha and X. Zhou and K.K. Tamma (2003), *Time discretized operators. Part 2: towards the theoretical design of a new generation of a generalized family of unconditionally stable implicit and explicit representations of arbitrary order for computational dynamics*, Computer Method in Applied Mechanics and Engineering, 291-329.