INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING, VOL. 40, 2739–2758 (1997)

AN ALGEBRAICALLY PARTITIONED FETI METHOD FOR PARALLEL STRUCTURAL ANALYSIS: PERFORMANCE EVALUATION

MANOEL R. JUSTINO, JR., K. C. PARK AND CARLOS A. FELIPPA

Center for Aerospace Structures and Department of Aerospace Engineering Sciences, Campus Box 429, University of Colorado, Boulder, CO 80309-429, U.S.A.

SUMMARY

This paper presents the algorithmic performance of an algebraically partitioned Finite Element Tearing and Interconnection (FETI) method presented in a companion paper. A simple structural assembly topology is employed to illustrate the implementation steps in a Matlab software environment. Numerical results indicate that the method is scalable, provided the iterative solution preconditioner employs the reduced interface Dirichlet preconditioner. A limited comparison of the present method with the differentially partitioned FETI method with corner modes is also offered. Based on this comparison and a reasonable extrapolation, we conclude the present algebraically partitioned FETI method possesses a similar iteration convergence property of the differentially partitioned FETI method with corner modes. © 1997 by John Wiley & Sons, Ltd.

KEY WORDS: parallel computation; algebraic partitioning; domain decomposition; structural analysis; finite element method; iterative solution

1. INTRODUCTION

In a previous paper[1] we presented an algebraically partitioned FETI algorithm, which solves the following finite element structural equation on parallel computers:

$$\mathbf{K}_g\mathbf{u}_g = \mathbf{f}_g \tag{1}$$

where $\mathbf{K}_g$ is the global structural stiffness matrix, $\mathbf{u}_g$ is the discrete nodal displacement, and $\mathbf{f}_g$ is the applied force. By *algebraic partitioning* it is meant to exploit the way the global structural stiffness matrix $\mathbf{K}_g$ can be partitioned into the following triple product matrices:

$$\mathbf{K}_g = [\mathbf{L}^T][\mathbf{K}^{(s)}][\mathbf{L}], \quad \mathbf{K}^{(s)} = \begin{bmatrix} \mathbf{K}^{(1)} & & & \\ & \mathbf{K}^{(2)} & & \\ & & \ddots & \\ & & & \mathbf{K}^{(n_s)} \end{bmatrix} \tag{2}$$

where $\mathbf{L}$ is the finite element assembly Boolean operator, $\mathbf{K}^{(s)}$ is substructure-by-substructure stiffness matrices, and the superscript s denotes the subdomain ($s = 1, 2, 3, \ldots, n_s$) where n_s is the total number of subdomains.

The solution of the nodal displacement $\mathbf{u}_g$ is carried out in three stages:

(a) solve for $\mathbf{p}^{(s)}$ from $[\mathbf{L}^T]\mathbf{p}^{(s)} = \mathbf{f}_g$
(b) solve for $\mathbf{u}^{(s)}$ from $[\mathbf{K}^{(s)}]\mathbf{u}^{(s)} = \mathbf{p}^{(s)}$
(c) solve for $\mathbf{u}_g$ from $[\mathbf{L}]\mathbf{u}_g = \mathbf{u}^{(s)}$

CCC 0029–5981/97/152739–20$17.50

Received 9 April 1996
Revised 14 November 1996

where $\mathbf{u}^{(s)}$ is the substructural displacement at each subdomain and $\mathbf{p}^{(s)}$ is the subdomain internal resisting forces vector.

It was shown in Park *et al.*[1] that the solution of the substructural displacement $\mathbf{u}^{(s)}$ is given by

$$\mathbf{u}^{(s)} = \mathbf{L}\mathbf{u}_g = \mathbf{F}_b^{(s)}\{\mathbf{f}^{(s)} - \mathbf{I}_b^{(s)}\boldsymbol{\lambda}_b^{(s)}\} - \mathbf{R}^{(s)}\boldsymbol{\lambda}_r^{(s)}$$

$$\mathbf{F}_b^{(s)} = \mathbf{I}_b^{(s)^T}\mathbf{K}^{(s)^+}\mathbf{I}_b^{(s)} \tag{3}$$

where $\boldsymbol{\lambda}_b^{(s)}$ is the Lagrange multipliers corresponding to the subdomain interface forces; $\boldsymbol{\lambda}_r^{(s)}$ is the subdomain rigid-body mode displacement; $\mathbf{I}_b^{(s)}$ corresponds to the interface node localization Boolean matrix; $\mathbf{R}^{(s)}$ is the rigid-body modes for floating substructures; and $\mathbf{K}^{(s)^+}$ is a generalized inverse of substructural stiffness matrix, respectively.

Hence, the central concern of the present method is to compute the Lagrange multiplier $\boldsymbol{\lambda}_b^{(s)}$ and the rigid-body mode displacement $\boldsymbol{\lambda}_r^{(s)}$, which can be computed from the following coupled equations:

$$\begin{bmatrix} \mathbf{F}_b & \mathbf{R}_b & \mathbf{L}_b \\ \mathbf{R}_b^T & \mathbf{0} & \mathbf{0} \\ \mathbf{L}_b^T & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \boldsymbol{\lambda}_b \\ \boldsymbol{\lambda}_r \\ \mathbf{u}_b \end{Bmatrix} = \begin{Bmatrix} \mathbf{b}_\lambda \\ \mathbf{b}_r \\ \mathbf{0} \end{Bmatrix}$$

$$\begin{aligned} \mathbf{F}_b &= \mathbf{I}_b^{(s)^T}\mathbf{F}^{(s)}\mathbf{I}_b^{(s)} \\ \mathbf{L}_b &= \mathbf{I}_b^{(s)^T}\mathbf{L} \\ \mathbf{R}_b &= \mathbf{I}_b^{(s)^T}\mathbf{R}^{(s)} \\ \mathbf{b}_\lambda &= \mathbf{I}_b^{(s)^T}\mathbf{F}^{(s)}\mathbf{f}^{(s)} \\ \mathbf{b}_r &= \mathbf{R}^{(s)^T}\mathbf{f}^{(s)} \end{aligned} \tag{4}$$

As stated in Reference 1 two distinct features of the present method are the derivation of the localization operator $\mathbf{I}_b^{(s)}$ and the constraint operator on the localized interface force $\mathbf{L}_b$ from the finite element assembly operator $\mathbf{L}$, and the computation of the rigid-body modes $\mathbf{R}^{(s)}$ via the substructural equilibrium considerations. To this end, the present paper is organized as follows.

Section 2 describes the computation of the localization operator $\mathbf{I}_b^{(s)}$ and the constraint operator $\mathbf{L}_b$, all of which are derived from the finite element assembly Boolean matrix $\mathbf{L}$. The computations of the rigid-body modes for floating substructures are given in Section 3, which utilizes only the finite element mesh geometries. The solution of the interface force Lagrange multipliers $\boldsymbol{\lambda}_b^{(s)}$ is presented in Section 4. The construction of two projection operators that are needed for the iterative solution process are also detailed therein, along with the preconditioning strategies employed. The numerical strategies adopted are described in Section 5. Section 6 presents numerical results of the present method applied to solve beam and plate problems. The Matlab code that has been used to evaluate the present algorithm in a simulated sequential computing environment is described. Finally, discussions are offered for parallel implementation of the present method.

2. COMPUTATION OF LOCALIZATION OPERATOR $\mathbf{I}_b^{(s)}$

As discussed in Reference 1 there are several choices for constructing the localization operator. It may be that other choices may lead to favourable implementations for certain problems. However, we will defer to a later study for more systematic evaluations of different localizations. The imple-

mentation we present below corresponds to the *minimal redundancy* case discussed in Reference 1 as this yields a minimal number of localized interface forces $\boldsymbol{\lambda}_b^{(s)}$. To this end, we introduce the finite element assembly operator given by

$$\mathbf{u}^{(s)} = \mathbf{L}\mathbf{u}_g$$
$$(\text{neq}_s \times 1)\,(\text{neq}_s \times \text{neq}_g)\,(\text{neq}_g \times 1) \tag{5}$$

where neq_s is the total number of subdomain equations and neq_g is the total number of global equations.

Algorithm for constructing $\mathbf{I}_b^{(s)}$:

1. *Allocate an identity matrix* $\mathbf{I}^{(s)}(\text{neq}_s \times \text{neq}_s)$.
2. *Delete the columns of* $\mathbf{I}^{(s)}$ *that correspond to the interior degrees of freedom for all the substructures. In other words, scan* $\mathbf{L}$ *column* $\times$ *column. If there is only one unity* (*one non-zero entry in a column*) *whose corresponding row is* n_{row}, *then decimate column* n_{row} *of* $\mathbf{I}^{(s)}$.

The preceding algorithm is equivalent to the following substructure-by-substructure matrix manipulations given below. For each substructure, partition $\mathbf{I}^{(s)}$ as

$$\mathbf{I}^{(s)} = \begin{bmatrix} \mathbf{I}_b^{(s)} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_i^{(s)} \end{bmatrix} \tag{6}$$

where $\mathbf{I}_b^{(s)}$ and $\mathbf{I}_i^{(s)}$ correspond to the substructural boundary and interior degrees of freedom, respectively. Then the present minimal-redundancy localization operator $\mathbf{N}_b^{(s)}$ is obtained by

$$\mathbf{I}_b^{(s)} = \begin{bmatrix} \mathbf{I}_b^{(s)} \\ \mathbf{0} \end{bmatrix} \tag{7}$$

For illustrative purposes, consider a refined model of a beam structure used in Reference 1 as given in Figure 1.

In Figure 1 the displacement vector superscript designates the subdomain number and the subscript numbering refers to subdomain node components. Each nodal displacement vector consists of three generalized displacements, $\mathbf{u} = \langle\, u_x \;\; u_y \;\; \theta_z \,\rangle^{\mathrm{T}}$.

The assembly Boolean operator $\mathbf{L}$ for this example problem is given by

$$\mathbf{L} = \begin{bmatrix} \mathbf{I}_3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{I}_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{I}_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{I}_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{I}_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{I}_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{I}_3 & 0 \\ 0 & 0 & \mathbf{I}_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{I}_3 \\ 0 & 0 & \mathbf{I}_3 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{8}$$

Since the n_{col} list includes

$$n_{\text{col}} = \langle 1 \;\; 2 \;\; 5 \;\; 6 \;\; 7 \;\; 9 \rangle \tag{9}$$

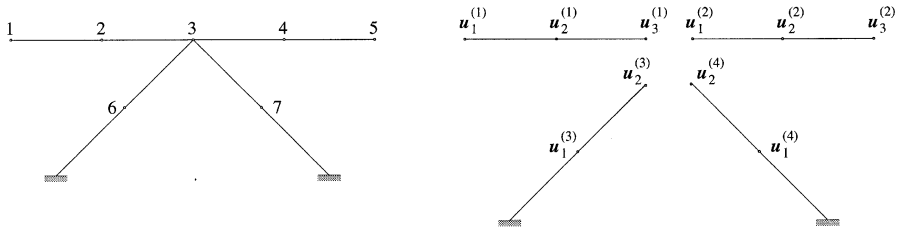

Figure 1. Refined four-subdomain geometry

the resulting $\mathbf{I}_b^{(s)}$ becomes

$$\mathbf{I}_b^{(s)} = \begin{bmatrix} \mathbf{0} & & & & \\ \mathbf{0} & & & & \\ \mathbf{I}_3 & & & & \\ & \mathbf{I}_3 & & & \\ & \mathbf{0} & & & \\ & \mathbf{0} & & & \\ & & \mathbf{0} & & \\ & & \mathbf{I}_3 & & \\ & & & \mathbf{0} & \\ & & & \mathbf{I}_3 & \end{bmatrix} = \begin{bmatrix} \mathbf{I}_b^{(1)} & & & \\ & \mathbf{I}_b^{(2)} & & \\ & & \mathbf{I}_b^{(3)} & \\ & & & \mathbf{I}_b^{(4)} \end{bmatrix} \tag{10}$$

Thus, the resulting operator is truly localized. In other words, the computations of $\mathbf{F}_b^{(s)}\boldsymbol{\lambda}_b^{(s)} = \mathbf{I}_b^{(s)^T}\mathbf{F}^{(s)}\mathbf{I}_b^{(s)}\boldsymbol{\lambda}_b^{(s)}$ can be carried out in individual processors.

3. COMPUTATION OF GLOBAL INTERFACE ASSEMBLY OPERATOR $\mathbf{L}_b$

This operator is obtained by premultiplying $\mathbf{I}_b^{(s)^T}$ on the finite element assembly equation (5):

$$\mathbf{I}_b^{(s)^T}\mathbf{u}^{(s)} = \mathbf{L}_b\mathbf{u}_g, \quad \mathbf{L}_b = \mathbf{I}_b^{(s)^T}\mathbf{L} \tag{11}$$

Carrying out the necessary computations for the example problem using (8) and (10), we obtain

$$\mathbf{L}_b = \mathbf{I}_b^{(s)^T}\mathbf{L} = [\,\mathbf{I}_3\ \mathbf{I}_3\ \mathbf{I}_3\ \mathbf{I}_3\,]^T \tag{12}$$

where six zero columns are compressed out as they contribute nothing to the constraint condition. An alternative algorithm which we recommend can be described as follows.

Algorithm for Constructing $\mathbf{L}_b$:

1. *Scan the columns of* $\mathbf{L}$.
2. *If there is only a single non-zero entry of a column, then decimate that column and row. Physically, the single entry, or unity occurring only one place in that column, corresponds to the substructural interior degrees of freedom.*

For the example case, this corresponds to retaining the third column while eliminating the zero column entries. The result is, for the example problem, is the same as given by (12).

It will be shown that the constraint operator $\mathbf{L}_b$ is used for constructing the following projection operator:

$$\mathbf{P}_\ell = \mathbf{I} - \mathbf{L}_b(\mathbf{L}_b^T\mathbf{L}_b)^{-1}\mathbf{L}_b^T \tag{13}$$

This projection can be shown to be factorized into a product form:

$$\mathbf{P}_\ell = \mathbf{I} - \mathbf{L}_b(\mathbf{L}_b^T\mathbf{L}_b)^{-1}\mathbf{L}_b^T = \mathbf{N}_\ell\mathbf{N}_\ell^T \tag{14}$$

where $\mathbf{N}_\ell$ is an orthonormal nullspace of $\mathbf{L}_b$. It turns out that $\mathbf{N}_\ell$ is nothing but the normalized matrix of the global nullspace matrix $\mathbf{N}_b$ of $\mathbf{L}$ described in Section 4 of Reference 1. Specifically, $\mathbf{N}_\ell$ for the example problem is given by the normalization (it is already orthogonal!):

$$\bar{\mathbf{N}}_b = \begin{Bmatrix} \mathbf{I}/\sqrt{2} & 0 & \mathbf{I}/2 \\ -\mathbf{I}/\sqrt{2} & 0 & \mathbf{I}/2 \\ 0 & \mathbf{I}/\sqrt{2} & -\mathbf{I}/2 \\ 0 & -\mathbf{I}/\sqrt{2} & -\mathbf{I}/2 \end{Bmatrix} \tag{15}$$

For two-, three- and five-node interfaces the corresponding matrices become

Two-node case:

$$\mathbf{N}_\ell = \begin{bmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{bmatrix}$$

Three-node case:

$$\mathbf{N}_\ell = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{6} \\ -1/\sqrt{2} & 1/\sqrt{6} \\ 0 & -2/\sqrt{6} \end{bmatrix} \tag{16}$$

Five-node case:

$$\mathbf{N}_\ell = \begin{bmatrix} 1/\sqrt{2} & 0 & 1/2 & 1/\sqrt{8} \\ -1/\sqrt{2} & 0 & 1/2 & 1/\sqrt{8} \\ 0 & 1/\sqrt{2} & -1/2 & 1/\sqrt{8} \\ 0 & -1/\sqrt{2} & -1/2 & 1/\sqrt{8} \\ 0 & 0 & 0 & -2/\sqrt{8} \end{bmatrix} \tag{17}$$

Remark 1. Optimal $\mathbf{B}^{(s)}$ in the differentially partitioned FETI method?

Observe that $\mathbf{P}_\ell = \mathbf{N}_\ell\mathbf{N}_\ell^T$ is nothing but a global interface connectivity operator. Therefore, the preceeding projection operator $\mathbf{P}_\ell$ would offer an alternative to the interface connectivity matrix

$\mathbf{B}^{(s)}$ of the differentially partitioned FETI method:[2, 3]

$$\hat{\mathbf{B}}^{(s)} = \frac{1}{2}\begin{bmatrix} \mathbf{I} & -\mathbf{I} \\ -\mathbf{I} & \mathbf{I} \end{bmatrix} \quad \text{for two-nodal interface}$$

$$\hat{\mathbf{B}}^{(s)} = \frac{1}{3}\begin{bmatrix} 2\mathbf{I} & -\mathbf{I} & -\mathbf{I} \\ -\mathbf{I} & 2\mathbf{I} & -\mathbf{I} \\ -\mathbf{I} & -\mathbf{I} & 2\mathbf{I} \end{bmatrix} \quad \text{for three-nodal interface} \tag{18}$$

$$\hat{\mathbf{B}}^{(s)} = \frac{1}{4}\begin{bmatrix} 3\mathbf{I} & -\mathbf{I} & -\mathbf{I} & -\mathbf{I} \\ -\mathbf{I} & 3\mathbf{I} & -\mathbf{I} & -\mathbf{I} \\ -\mathbf{I} & -\mathbf{I} & 3\mathbf{I} & -\mathbf{I} \\ -\mathbf{I} & -\mathbf{I} & -\mathbf{I} & 3\mathbf{I} \end{bmatrix} \quad \text{for four-nodal interface}$$

in lieu of the following choices:

$$\mathbf{B}^{(s)} = [\mathbf{I} \quad -\mathbf{I}] \quad \text{for two-nodal interface}$$

$$\mathbf{B}^{(s)} = \begin{bmatrix} \mathbf{I} & -\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & -\mathbf{I} \\ -\mathbf{I} & \mathbf{0} & \mathbf{I} \end{bmatrix} \quad \text{for three-nodal interface} \tag{19}$$

$$\mathbf{B}^{(s)} = \begin{bmatrix} \mathbf{I} & -\mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & -\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & -\mathbf{I} \\ -\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \quad \text{for four-nodal interface}$$

It would be interesting to see whether the present $\hat{\mathbf{B}}^{(s)}$ given by (18) in fact offer iterative and/or accuracy advantages compared with the differentially partitioned FETI construction (19) with additional redundancies. This remains to be verified.

4. COMPUTATION OF RIGID-BODY MODES $\mathbf{R}^{(s)}$

The rigid-body modes of a completely free–free substructure with three translations and three rotations at a node were derived in Reference 1 in the form

$$\mathbf{R}^{(s)^{\mathrm{T}}} = [\mathbf{r}_1 \ \mathbf{r}_2 \ \ldots \ \mathbf{r}_n] \tag{20}$$

$$\mathbf{r}_i = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0} \\ \boldsymbol{\chi}_{(i)} & \mathbf{I}_3 \end{bmatrix}, \quad \boldsymbol{\chi} = \begin{bmatrix} 0 & -(z_i - z_0) & (y_i - y_0) \\ (z_i - z_0) & 0 & -(x_i - x_0) \\ -(y_i - y_0) & (x_i - x_0) & 0 \end{bmatrix}$$

In the above equation the six rows in $\mathbf{r}_i$ correspond to $(u, v, w, \theta_x, \theta_y, \theta_z)$ rigid modes, where (u, v, w) and $(\theta_x, \theta_y, \theta_z)$ are three translational displacements and three rotations, respectively.

Algorithm for computing $\mathbf{R}^{(s)^T}$ *of partially constrained substructures*

1. *Partition for each substructural displacement into*

$$\mathbf{u}^{(s)^T} = [\mathbf{u}_f^{(s)^T} \ \mathbf{u}_c^{(s)^T}] \tag{21}$$

   where the subscripts f *and* c *correspond to the free and constrained nodes, respectively.*
2. *Eliminate the rows and columns of the completely free–free* $\mathbf{R}^{(s)^T}$ *that correspond to the degrees of freedom of the identified constrained nodes* $\mathbf{u}_c^{(s)}$.
3. *For each constraint of the translational degrees of freedom, check to see which of the three rotations would have to be constrained as follows.*
   (a) *Introduce virtual forces that correspond to the constrained translational degrees of freedom, i.e.,* $V_x = 1$ *if* $u = 0$ *at least at two distinct discrete boundary points.*
   (b) *Compute the following virtual moments*:

$$\begin{Bmatrix} M_x \\ M_y \\ M_z \end{Bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -\triangle x \\ 0 & \triangle x & 0 \end{bmatrix} \begin{Bmatrix} V_x \\ V_y \\ V_z \end{Bmatrix} \tag{22}$$

   where $\triangle x$ *is the distance between the two distinct point, with x-co-ordinate being collinear with the two discrete points and* (y, z) *may be chosen arbitrarily.*
   (c) *Whenever any virtual moment is zero, the corresponding rotational degree of freedom is free to rotate. In other words,* $\mathbf{R}^{(s)^T}$ *must retain that rigid mode. For example,* θ_x *is seen to rotate freely since* $M_x = 0$. *If* $V_y = 0$, *then* θ_z *is free to rotate, etc.*
   (d) *If there is an additional constraint point that does not lie on the line drawn between the two points evaluated, repeat the same process with* $\triangle x'$ *and* $(V_{x'}, V_{y'}, V_{z'})$.

Remark 2. Ideally, one should choose the three distinct points $(\triangle x, \triangle x', \triangle x'')$ to coincide with the globally distinct Cartesian co-ordinates $(\triangle x, \triangle y, \triangle z)$. There are, however, situations when this is not possible. For example, take the case of a square plate for which (u, v, w) displacements are constrained at two diagonal points. The virtual moment equation take the following form:

$$\begin{Bmatrix} M_{x'} \\ M_{y'} \\ M_{z'} \end{Bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -\ell \\ 0 & \ell & 0 \end{bmatrix} \begin{Bmatrix} 1 \\ 1 \\ 1 \end{Bmatrix} \tag{23}$$

where ℓ is the diagonal length of the plate and x'-co-ordinate is parallel to the diagonal line. Clearly, we have only $\theta_{x'}$ to rotate freely.

5. SOLUTION STRATEGIES

For convenience let us recall the coupled partitioned equation (4):

$$\begin{bmatrix} \mathbf{F}_b & \mathbf{R}_b & \mathbf{L}_b \\ \mathbf{R}_b^T & \mathbf{0} & \mathbf{0} \\ \mathbf{L}_b^T & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \boldsymbol{\lambda}_b \\ \boldsymbol{\lambda}_r \\ \mathbf{u}_b \end{Bmatrix} = \begin{Bmatrix} \mathbf{b}_\lambda \\ \mathbf{b}_r \\ \mathbf{0} \end{Bmatrix}$$

$$\begin{aligned} &\mathbf{F}_b = \mathbf{I}_b^T \mathbf{K}^{(s)^+} \mathbf{I}_b \\ &\mathbf{R}_b = \mathbf{I}_b^T \mathbf{R}, \quad \mathbf{L}_b = \mathbf{I}_b^T \mathbf{L} \\ &\mathbf{b}_\lambda = \mathbf{I}_b^T \mathbf{K}^{(s)^+} \mathbf{f} \\ &\mathbf{b}_r = \mathbf{R}^T \mathbf{f} \end{aligned} \tag{4}$$

where we dropped the subdomain designation superscript (s) for notational simplicity.

Observe that the interface force Lagrange multipliers $\boldsymbol{\lambda}_b$ constitutes the primary unknown while $(\boldsymbol{\lambda}_r, \mathbf{u}_{gb})$ may be considered as constraints. Hence, in seeking the solution of $\boldsymbol{\lambda}_b$ from (4a):

$$\mathbf{F}_b \boldsymbol{\lambda}_b + \mathbf{R}_b \boldsymbol{\lambda}_r + \mathbf{L}_b \mathbf{u}_b = \mathbf{b}_\lambda \tag{24}$$

it is preferable to project $(\boldsymbol{\lambda}_r, \mathbf{u}_b)$ out. Here, one can proceed in two ways. The first is to project out the second term then the third term. The second is the reverse of the first. We will adopt the first procedure as presented below.

5.1. *Projection of* $\mathbf{R}_b \boldsymbol{\lambda}_r$ *term*

In order to project out the second term in (24), viz., $\mathbf{R}_b \boldsymbol{\lambda}_r$, we utilize the following projector:

$$\mathbf{P}_R = \mathbf{I} - \mathbf{R}_b (\mathbf{R}_b^T \mathbf{R}_b)^{-1} \mathbf{R}_b^T \tag{25}$$

Notice that we have

$$\mathbf{P}_R \mathbf{R}_b \boldsymbol{\lambda}_r = (\mathbf{I} - \mathbf{R}_b (\mathbf{R}_b^T \mathbf{R}_b)^{-1} \mathbf{R}_b^T) \mathbf{R}_b \boldsymbol{\lambda}_r = 0 \tag{26}$$

Hence, premultiplying (24) by $\mathbf{P}_R$ yields

$$\mathbf{P}_R \mathbf{F}_b \boldsymbol{\lambda}_b + \mathbf{P}_R \mathbf{L}_b \mathbf{u}_b = \mathbf{P}_R \mathbf{b}_\lambda \tag{27}$$

Observe that $\mathbf{P}_R$ consists of localized quantities as $\mathbf{I}_b$ and $\mathbf{R}^{(s)}$ are localized operators. Hence, no interprocessor communication is needed in computing $\mathbf{P}_R$.

5.2. *Projection of* $\mathbf{P}_R \mathbf{L}_b \mathbf{u}_b$ *term*

The remaining task is to project out the second term in (27), viz., $\mathbf{P}_R \mathbf{L}_b \mathbf{u}_b$. In a similar manner the appropriate projection operator for this case is

$$\mathbf{P}_L = \mathbf{I} - \mathbf{P}_R \mathbf{L}_b (\mathbf{L}_b^T \mathbf{P}_R \mathbf{L}_b)^{-1} \mathbf{L}_b^T \mathbf{P}_R \tag{28}$$

Since $\mathbf{L}_b$ is a global operator, $\mathbf{P}_L$ becomes a global operator so that its construction requires interprocessor data communications.

Premultiplying (27) by $\mathbf{P}_L$ one obtains the desired projected residual for an iterative solution given by

$$\mathbf{r}_P = \mathbf{P}_L \mathbf{P}_R (\mathbf{b}_\lambda - \mathbf{F}_b \boldsymbol{\lambda}_b) \tag{29}$$

We are now ready to solve for $\boldsymbol{\lambda}_b$ from the above projected equation by an iterative solution strategy. This involves an effective preconditioner, which is addressed below.

5.3. *Substructural flexibility* $\mathbf{K}^{(s)^+}$ *and preconditioner* $\mathbf{F}_b^+$

An iterative solution of the interface Lagrange multipliers $\boldsymbol{\lambda}_b$ from the projected residual given by (29) requires efficient computations of two matrices: a generalized inverse of subdomain substructural stiffness $\mathbf{K}^{(s)}$ and a generalized inverse of the substructural-node flexibility $\mathbf{F}_b$. Becasue the computational procedures are distinctly different depending on whether a substructure is fully constrained or free–free, two cases are dealt with separately.

5.3.1. When a substructure is fully constrained. For this case the substructural flexibility is simply given by

$$\mathbf{F}^{(s)} = \mathbf{K}^{(s)^{-1}} \;\Rightarrow\; \mathbf{F}_b = \mathbf{I}_b^T \mathbf{F}^{(s)} \mathbf{I}_b \tag{30}$$

5.3.2. When a substructure is free-floating or partially constrained. As the substructural stiffness matrix is singular, we seek a generalized inverse that satisfies the following relation:

$$\mathbf{F}^{(s)}\mathbf{K}^{(s)} = \mathbf{I} - \mathbf{R}^{(s)}(\mathbf{R}^{(s)^T}\mathbf{R}^{(s)})^{-1}\mathbf{R}^{(s)^T} \tag{31}$$

To this end, we partition the substructural stiffness $\mathbf{K}^{(s)}$ and the rigid-body mode $\mathbf{R}^{(s)}$ in the form

$$\mathbf{K}^{(s)} = \begin{bmatrix} \mathbf{K}_{cc} & \mathbf{K}_{cf} \\ \mathbf{K}_{fc} & \mathbf{K}_{ff} \end{bmatrix}, \quad \mathbf{R}^{(s)} = \begin{bmatrix} \mathbf{R}_c \\ \mathbf{R}_f \end{bmatrix} \tag{32}$$

where the subscripts c and f refer to the total rigid-mode numbers and rank-sufficient part of the stiffness.

With the above partitioning, it can be shown that the substructural stiffness $\mathbf{K}^{(s)}$ can be expressed as

$$\mathbf{K}^{(s)} = \mathbf{T}^T \mathbf{K}_{ff} \mathbf{T}, \quad \mathbf{T} = [\, -\bar{\mathbf{R}} \quad \mathbf{I}\,], \quad \bar{\mathbf{R}} = \mathbf{R}_f \mathbf{R}_c^{-1} \tag{33}$$

Therefore, a generalized inverse of $\mathbf{K}^{(s)}$ can be obtained as

$$\begin{aligned} \mathbf{F}^{(s)} = \mathbf{K}^{(s)^+} &= \mathbf{T}^+ \mathbf{K}_{ff}^{-1} \mathbf{T}^{T+} \\ &= \mathbf{T}^T \mathbf{C} \mathbf{K}_{ff}^{-1} \mathbf{C} \mathbf{T}, \quad \mathbf{C} = \mathbf{I} - \bar{\mathbf{R}}[\mathbf{I} + \bar{\mathbf{R}}^T \bar{\mathbf{R}}]^{-1} \bar{\mathbf{R}}^T \end{aligned} \tag{34}$$

It should be noted that $\mathbf{K}_{ff}^{-1}$ is a sparse matrix and $[\mathbf{I} + \bar{\mathbf{R}}^T \bar{\mathbf{R}}]$ appearing in $\mathbf{C}$ is at most a (6×6) matrix, thus making the computation of $\mathbf{F}^{(s)}$ efficient and accurate.

Once the substructural flexibility is computed, the interface-node flexibility $\mathbf{F}_b$ is obtained by the row and column extraction formula given by (30b). The generalized inverse

$$\mathbf{F}_b^+ = \mathbf{P}_R(\mathbf{P}_R \mathbf{F}_b \mathbf{P}_R + \mathbf{R}_b(\mathbf{R}_b^T \mathbf{R}_b)^{-1}\mathbf{R}_b^T)^{-1} \tag{35}$$

where $\mathbf{P}_R$ is given by (25).

Remark 3. Farhat *et al.*[4] discuss various preconditioners and show that the so-called Dirichlet or Schur complement preconditioner is, although computationally expensive, the best choice. This preconditioner is, in fact, the well-known Guyan-reduced substructural stiffness matrix in terms of the boundary nodes. To compute this preconditioner, one first partition

$$\mathbf{K}^{(s)} = \begin{bmatrix} \mathbf{K}_{ii} & \mathbf{K}_{ib} \\ \mathbf{K}_{bi} & \mathbf{K}_{bb} \end{bmatrix} \tag{36}$$

where the subscripts b and i refer to the interior and boundary nodes, respectively. With this partition, they discuss two preconditioners:

$$\begin{aligned} &\text{Lumped:} \quad \mathbf{F}_{\mathrm{b}}^{+} \approx \mathbf{K}_{\mathrm{bb}} \\ &\text{Dirichlet:} \quad \mathbf{F}_{\mathrm{b}}^{+} = \bar{\mathbf{K}}_{\mathrm{bb}} = \mathbf{K}_{\mathrm{bb}} - \mathbf{K}_{\mathrm{bi}} \mathbf{K}_{\mathrm{ii}}^{-1} \mathbf{K}_{\mathrm{ib}} \end{aligned} \tag{37}$$

When the number of the internal nodes far exceeds those of the boundary nodes, the factorization of $\mathbf{K}_{\mathrm{ii}}$ for generating the Dirichlet preconditioner can be expensive to compute. Hence, the present formula (35) may be viewed as an efficient alternative form for (37b) since $\mathbf{F}$ is obtained easily by extracting the appropriate rows and columns of the substructural flexibility given by (34).

5.4. *Preconditioned conjugate gradient residual*

Using the generalized inverse of the interface-node flexibility (35), the preconditioned conjugate gradient residual becomes

$$\mathbf{r}_{\mathrm{PCG}} = \mathbf{P}_L \mathbf{P}_R \mathbf{F}_{\mathrm{b}}^{+} \mathbf{P}_R \mathbf{P}_L (\mathbf{b}_\lambda - \mathbf{F}_{\mathrm{b}} \boldsymbol{\lambda}_{\mathrm{b}}) \tag{38}$$

which is implemented for numerical evaluation of the algorithm in Matlab programming environment.

5.5. *Starting vector of* $\boldsymbol{\lambda}_{\mathrm{b}}$

The starting vector for iterating the projected residual (38) plays a pivotal role. This is because the starting vector must satisfy the second and third row of (4). The desirable starting vector that satisfies the two constraint conditions is found as

$$\boldsymbol{\lambda}_{\mathrm{b}}^{0} = \mathbf{P}_L \mathbf{R}_{\mathrm{b}} (\mathbf{R}_{\mathrm{b}}^{\mathrm{T}} \mathbf{P}_L \mathbf{R}_{\mathrm{b}})^{-1} \mathbf{b}_{\mathrm{r}} \tag{39}$$

6. NUMERICAL EXAMPLES

In order to assess the algorithmic performance of the present algebraically partitioned FETI method, we have implemented the computational details presented in the preceding sections into a Matlab code to be run on workstations. Since the Matlab code or its parallel counterpart has not been run on any parallel computers, we have no assessment on the interprocessor communications requirements of the present method, an important factor for parallel algorithms. We do believe, however, the basic iteration convergence rate is of primary importance of a method. This is because interprocessor communication overhead can often be estimated once parallel architecture is known.

We have chosen four sample problems: a plane frame problem of four subdomains subjected to both applied shear force and moment; a multiply-connected beam with six subdomains subjected to a combination of extension, shear and moment at one end; a square plate simply supported at four corners subjected to a concentrated load; and, a cantilevered thin plate under a uniform load.

6.1. *Plane frame model*

Figure 2 shows the plane frame model that is partitioned into four subdomains. Each member is treated as a subdomain identified by a number in the parentheses. Observe that each of the two diagonal beams, designated as subdomains (3) and (4) in Figure 2, would possess a rotational rigid-body mode. Also, node 1 where both shear force and moment are applied constitutes a four-subdomain cross point.

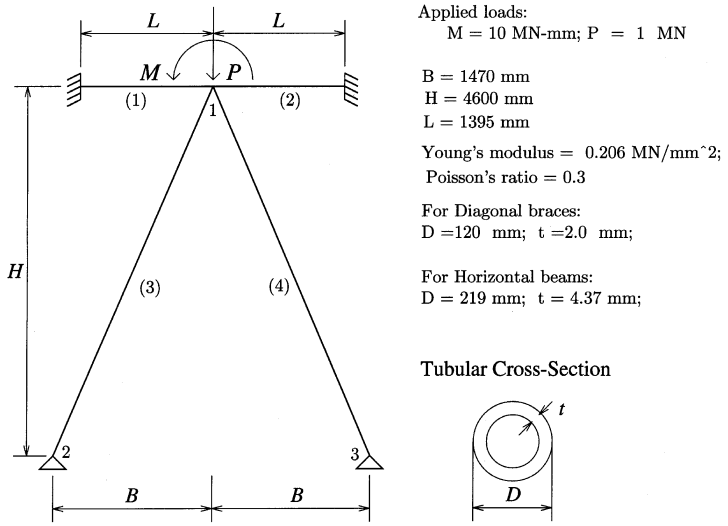

Figure 2. Plane frame model

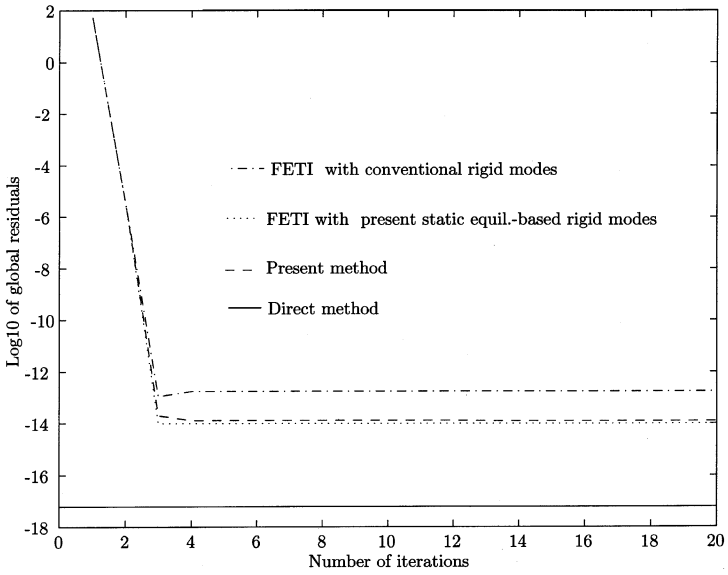

Figure 3. Global residual vs. iteration numbers for the plane frame in Figure 2

Figure 3 shows the iteration vs. convergence of the present algebraically partitioned FETI method and a *globalized** version of the differentially partitioned FETI method for the plane

* *Globalized* here implies that the interface constraint operator $\mathbf{B}^{(s)}$ is not localized. That is, the condition $\sum_{s=1}^{n_s} \mathbf{B}^{(s)}\,\mathbf{u}^{(s)} = 0$ is globally satisfied. This means that the projection operator corresponding to $\mathbf{P}_G$ (25) would be globally coupled throughout the entire interfaces

frame model described in Figure 2. The global residual is computed by the following relative ℓ_2 norm:

$$\varepsilon = \|\mathbf{f}_g - \mathbf{K}_g\mathbf{u}_g\|_2 / \|\mathbf{f}_g\|_2 \tag{40}$$

where $\mathbf{K}_g$ is the global structural stiffness matrix, $\mathbf{u}_g$ is the displacement vector of the global assembled system and $\mathbf{f}_g$ is the applied forces vector.

In performing the preconditioned conjugate gradient (PCG) iterations on the residual (38), the Dirichlet preconditioner (37b) has been used. It should be noted that the differentially partitioned FETI interface operator $\mathbf{B}^{(s)}$ for this problem introduces three redundancies at the cross point, a preferred choice for a four-cross point. Thus, a total of 18 interface Lagrange multipliers $\boldsymbol{\lambda}_b^{(s)}$ are required. In contrast, the present algebraically partitioned FETI method does not contain any redundancy, requiring a total of 9 independent interface Lagrange multipliers $\boldsymbol{\lambda}_b^{(s)}$.

In addition, in carrying out the *globalized* differentially partitioned FETI solution, two different rigid-body modes have been used: the nullspace of the subdomain stiffness matrices and the subdomain rigid-body modes computed by the subdomain static equilibrium operator (20). It can be observed from Figure 3 that, even for this simple problem, the use of rigid-body modes constructed from the subdomain static equilibrium operator (20) yields about one-digit accuracy improvement of the iterated solution. Of course, the direct solution using a LU-decomposition yields about two-digit higher accuracy than possible by the three iterated solutions.

In Figure 4 we analyse the influence of choosing the preconditioning type. In all curves the interface problem is solved via the present algebraically partitioned FETI method. The importance of using a better preconditioner (37b) is shown in terms of its fast convergence rate and improved accuracy. Clearly, the Dirichlet preconditioner outperforms the lumped preconditioner (37a). The horizontal solid curve represents the global residual obtained using the direct LU method.

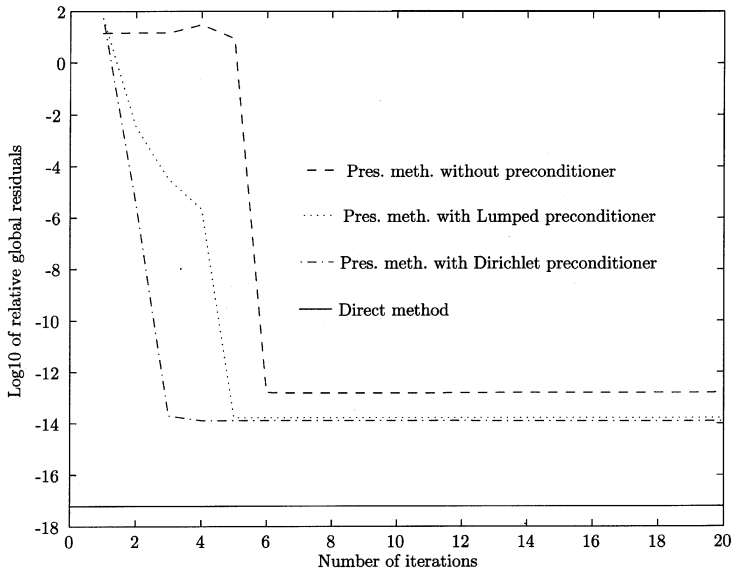

Figure 4. Influence of preconditioners for the plane frame in Figure 2

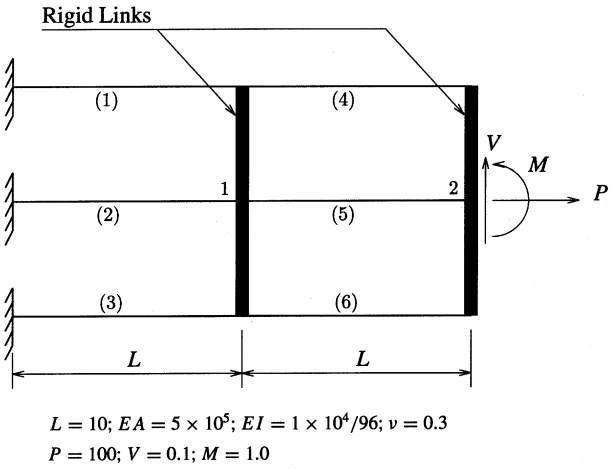

Figure 5. Multiply-connected horizontal beam

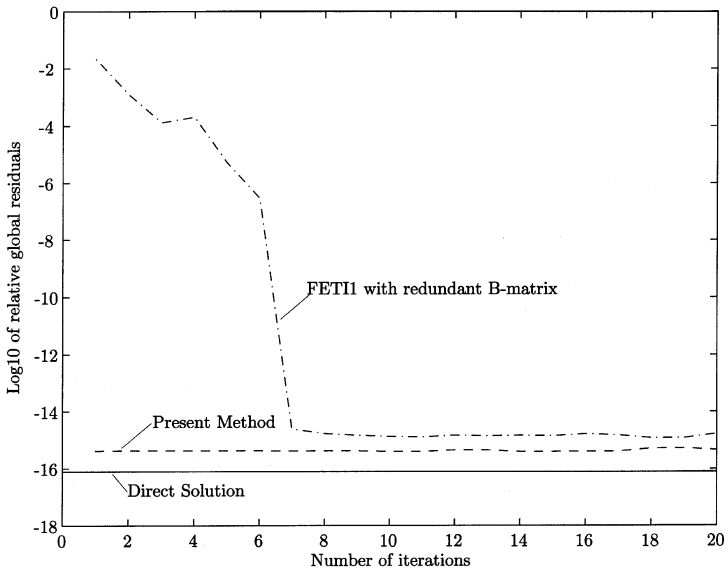

Figure 6. Global residual vs. iteration numbers for multiply-connected horizontal beam in Figure 5

6.2. *Multiply-connected horizontal beam*

A six-member multiply-connected horizontal beam is presented in Figure 5. The horizontal beams are interconnected vertically by rigid links. For this problem each element is considered as a subdomain. Because all the subdomain degrees of freedom (DOFs) become the interface boundary DOFs, the lumped and Dirichlet preconditioners are the same for this particular problem. The present algebraically partitioned FETI method introduces a total of 21 independent interface Lagrange multipliers, 15 at node 1 and 6 at node 2, respectively. For this problem, the present method converges in just one iteration as indicated in Figure 6.

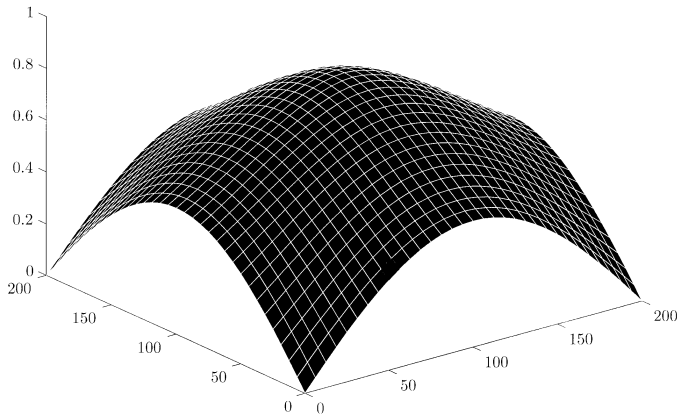

Figure 7. Deformed mesh of a 32×32 grid plate simply supported at four corners

On the other hand, the differentially partitioned FETI method, if a minimum redundancy at the cross points is chosen, would require a total of 27 Lagrange multipliers. For this problem, additional redundancies in conjunction with the latter method do not improve its convergence rate, requiring about 7 iterations.

Based on the numerical results, we conjecture that the present localized interface force operator $\mathbf{I}_b^{(s)}$ detailed in Section 2 and the constraint operator $\mathbf{L}_b$ presented in Section 3 renders a more desirable starting vector $\boldsymbol{\lambda}_b^{(s)}$.

6.3. Linear plate simply supported at four corners

The influence of the full reorthogonalization of search directions on the convergence rate of the conjugate gradient iterations is analysed for a square plate model, using the 4-node ANS plate bending elements.[5] The plate is simply supported at the four corners. A vertical concentrated load of 1 kN is applied at the center of the plate. The plate dimension is 200×200 cm, the plate thickness is 1 cm, Young's modulus is $2{\cdot}5 \times 10^4$ kN/cm^2, and Poisson's ratio 0·3. The deformed shape of the plate is presented in Figure 7 for a 32×32 element grid model. For this analysis 16 equal subdomains (each subdomain consists of a 8×8 element grid) are considered. The total number of subdomain equations is 3263 and the number of independent interface Lagrange multipliers is equal to 665. Rigid-body modes are computed via subdomain static equilibrium operator (20).

The influence of preconditioning on the convergence rate of the conjugate gradient method is shown in Figure 8. The element grid size is equal to 32×32 and the number of subdomains is equal to 16. In all curves the present method uses the subdomain flexibility matrices as pseudo-inverses, and the subdomain rigid-body modes are computed using subdomain static equilibrium operator (20). This figure highlights the importance of choicing a good preconditioner for improving the convergence rate of the conjugate gradient method for the interface method. Despite the computational cost involved, the Dirichlet preconditioner converges faster than the lumped preconditioner.

The influence of the full reorthogonalization on the convergence rate of the conjugate gradient method for this plate bending problem is shown in Figure 9. The lumped preconditioner has been used for this experiment. As can be seen in Figure 9, there is a drastic reduction on the number of iterations for reaching the minimum residual when the full reorthogonalization of search directions is used. The solid horizontal line represents the global relative residual from the direct method via LU decomposition.

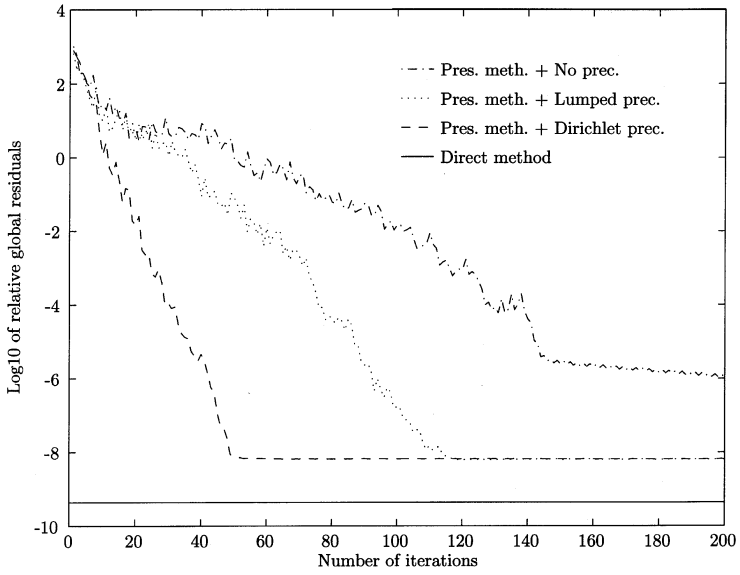

Figure 8. Preconditioning effect on the PCG convergence rate for model plate of Figure 7

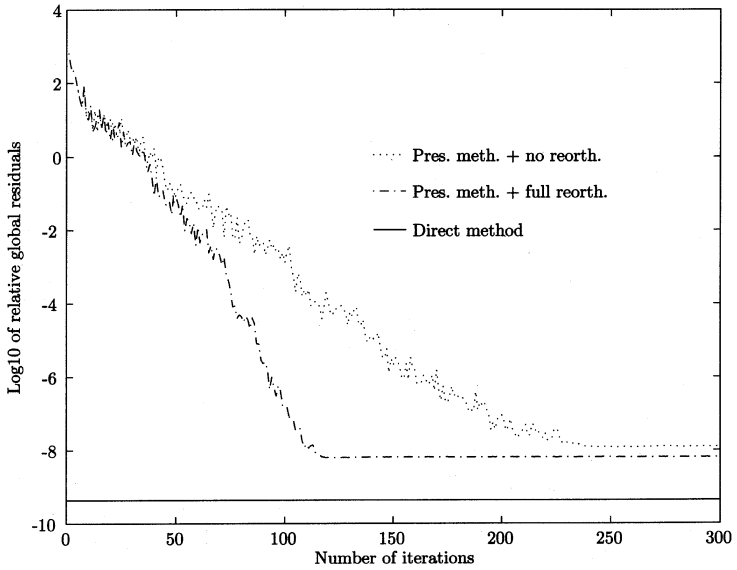

Figure 9. Influence of reorthogonalization for plate model of Figure 7 using 16 subdomains

In order to assess the algorithmic scalability of the present algebraically partitioned FETI method, we introduce Figure 10 for the definitions of subdomain and element size parameters. Table I presents the results of the present method for different grid sizes of the plate from a 4×4, 8×8, 16×16 to 32×32. For all analyses the Dirichlet preconditioner has been used in order to guarantee the scalability and optimality properties of the present algebraically partitioned FETI method. Observe from Table I that for 16 subdomains, as the three ratios H/h of 2, 4 and 8, the number of iterations remains about 30 for the relative global residual of 10^{-4} and 50

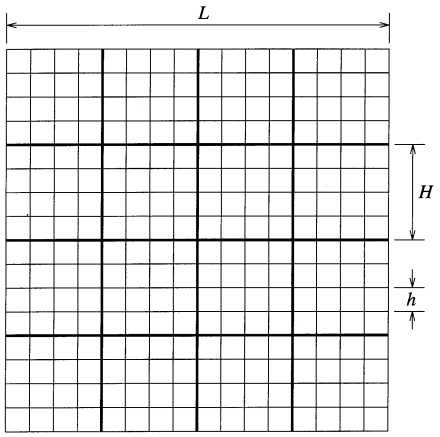

Figure 10. Definition of subdomain and element sizes for plate problems: H—subdomain size (typical), h—element size (typical), L—plate length

Table I. Scalability test of present method using plate problem (element used: four-noded C^0 shear deformable ANS element)

h/L	H/h	Number of subdomains	Number of iterations (10^{-4})	Number of iterations (10^{-8})	Number of Lagrange multipliers
1/4	2	4	2	5	41
1/4	1	16	18	21	161
1/8	4	4	5	7	65
1/8	2	16	30	39	233
1/16	8	4	9	13	113
1/16	4	16	32	48	377
1/32	8	16	31	50	665

for the case of 10^{-8} accuracy. This indicates that the present method is algorithmically scalable. This happens in spite of the increase in the number of the interface Lagrange multipliers from 233, 377 and 665, respectively. While we expect that this favourable scalability continues to hold for large-scale problems, a definite confirmation must await further numerical experiments are carried out.

6.4. Cantilever plate model

As our fourth and final example, the cantilever plate problem studied by Farhat and Mandel[6] is analysed. The plate is discretized with both the 4-ANS plate bending element and the 3-node ANDES plate bending elements of Militello and Felippa[7] and subjected to a uniform pressure load. The deformed shape of the plate for the case of a 16×16 element grid is shown in Figure 11.

Figure 12 shows the iteration convergence curves of the present method when the clamped plate is discretized by two different plate elements, namely, 4-ANS element[5] and a DKT-like element,[7] herein referred to *ANDES* element. We note that the accuracy rendered by the ANDES element is comparable to that of direct solution about $10^{-9}/3$. On the other hand, the 4-ANS element yields

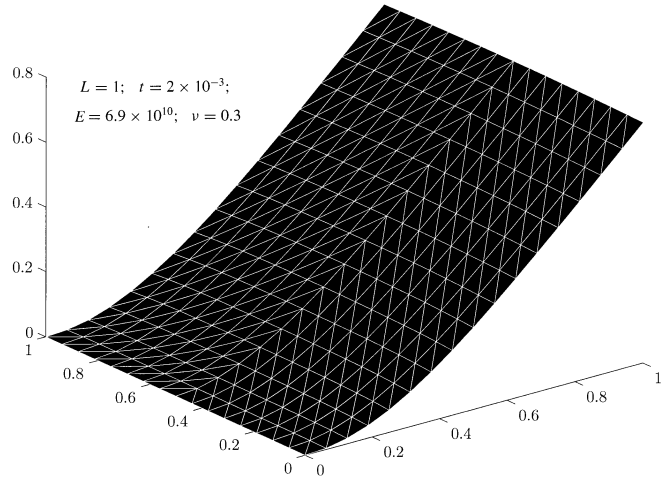

Figure 11. Deformed mesh of a 16×16 grid cantilever plate

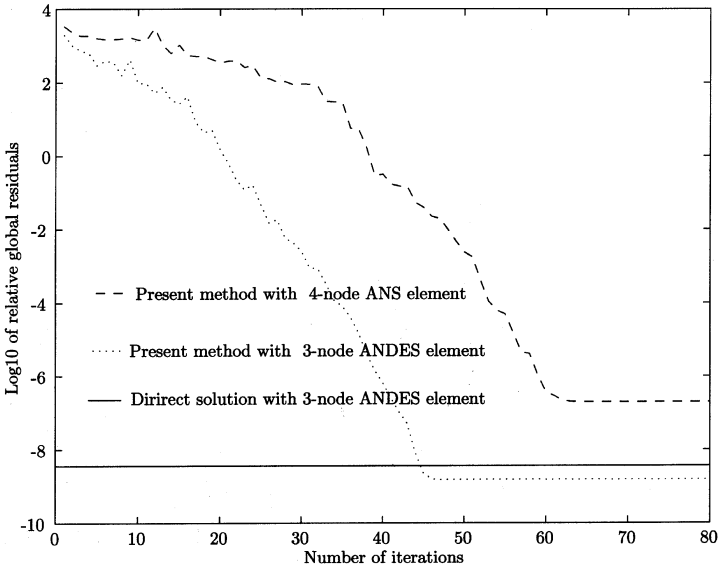

Figure 12. Convergence properties for a 32×32 grid cantilever plate

its maximum accuracy about $10^{-7}/2$. This is not surprising because the condition number of the global assembled stiffness matrix of the 4-ANS element is about 2 digit higher than that of the ANDES element.

Table II presents the performance of the present method using ANDES element. For all analyses the Dirichlet preconditioner has been used in order to assess the scalability and property of the present algebraically partitioned FETI (A-FETI) algorithm.

Comparing the performance of the present A-FETI algorithm (4) and (38) with that of FETI-1[3] and FETI-2[6] methods, it is seen that the present A-FETI algorithm performs competitively with

Table II. Scalability test of present method using cantilever plate (global relative residual ℓ_2 norm $= 10^{-6}$)

h/L	Number of subdomains	Number of iterations (FETI-1)	Number of iterations (Pres. A-FETI)	Number of iterations (FETI-2)	Lagrange multipliers (A-FETI)
1/8	4		11		102
1/8	16		19		306
1/16	4		14		198
1/16	16		29		594
1/32	16		40		1 170
1/40	16	69*	41(37*)	34*	1 396
1/80	16	82*	45	41*	2 898
1/90	36	154*	81	43*	5 430
1/120	64	238*	120	50*	10 122

* designates FETI-1 FETI-2 number of iterations when an estimated error measure is used, which is about 5–10 per cent less compared with when using a true relative error measure (40)

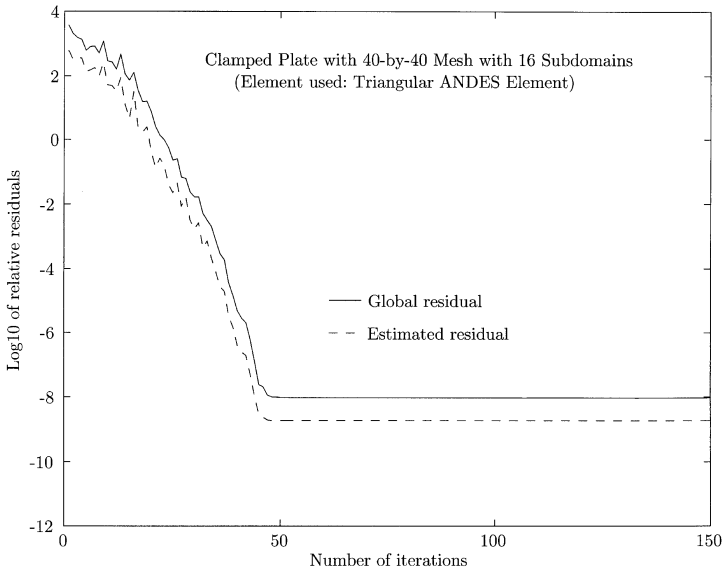

Figure 13. Convergence properties for a 40×40 grid cantilever plate

FETI-2. As the number of subdomain size increases, for example, for 64 subdomains the present algorithm takes about twice the iteration needed by FETI-2. It should be noted that FETI-2 requires to solve a coarser-grid problem iteratively at each fine-grid iteration step.

Figure 13 shows the iterations vs. the two error measures. Thus, at least for the (40×40)-grid case, the iteration numbers measured by an estimated error vs. true relative error can vary up to about 10 per cent difference. Further numerical experiments are needed for establishing the relative advantages of the three methods, especially in terms of programming ease, computational overhead and clock-time computing hours.

7. DISCUSSIONS

The scalability property of the present algebraically partitioned FETI method has been observed in moderate size beam and plate problems, which have been solved using a Matlab-based program. It should be noted that, although the subdomain sizes of the example problems are relatively small, these problems are indicative of some of the typical computational structural mechanics problems by iterative methods.

The interface operator $\mathbf{I}_b$ is constructed from the finite element assembly Boolean operator $\mathbf{L}$ by a column $\times$ column explicit procedure developed in Park *et. al.*[1] The resulting algorithm is efficient and leads to a unique $\mathbf{I}_b$. This constitutes a key first step for the present algebraically partitioned FETI method.

In most of the numerical examples analysed, the subdomain rigid-body modes obtained from the subdomain static equilibrium operator (20) improve the global residual accuracy of the solution, compared to the subdomain rigid-body modes obtained as a null-space basis of the subdomain stiffness matrices.

The present numerical experiments, limited as they are, clearly demonstrate the importance of using the Dirichlet preconditioner and full reorthogonalization of conjugate gradient search directions for fast convergence of the present method.

The solution matrix of the present method (4a) possesses its full rank. The FETI-1 method[2, 3] and the FETI-2 method[6] require redundancies in the mesh cross-points for good convergence. This means that the size of the interface Lagrange multipliers in the present method would be generally smaller than the FETI-1 and FETI-2 methods. Whether the full-rank property would be advantageous or not in terms of iteration efficiency has not been investigated neither theoretically nor numerically.

From a comparison with the FETI-1 and the FETI-2 methods, we have run the 40×40, 80×80, 90×90 and 120×120 for 16, 36 and 64 processors, respectively, using a clamped plate problem. For this particular problem, the number of iterations required by the present method is about 10 per cent more than that of the FETI-2 method for up to 16 processors. For the case of 64 processors, the iteration number of the present A-FETI method increased in proportion to $\sqrt{N_s}$ while that of the FETI-2 method according to $\log N_s$. This suggests that the use of a coarser-grid solution as the starting condition for the refined grids is advantageous in reducing the refined-grid iterations. Whether the use of coarser-grid solution offer a real reduction in computing time and benefiting the associated programming complexity or not still remains unanswered.

From a computational overhead viewpoint, the present method requires two projections, $\mathbf{P}_R$ (25) and $\mathbf{P}_L$ (28), all of which are kinematical quantities. The projection of $\mathbf{P}_R$ is strictly local and does not involve any interprocessor communication. The projection of $\mathbf{P}_L$ does require interprocessor communications, unless it is replaced by another iterations. Thus, the computational overhead of the present algebraically partitioned FETI method would be comparable to the FETI-1 method,[2, 3] although it would result in a wider band width than the FETI-1 projection matrix. This means that the present method is simpler to implement and may be computationally more efficient *per iteration* than the FETI-2 method which requires two-level iterations. Further experiments with both methods would offer the advantageous applications of each method. However, whether the present method would continue to be efficient for larger-size problems remains to be verified.

ACKNOWLEDGEMENT

This research has been partially funded by NSF/HPCC Grant ASC-9217394 and by Sandia National Laboratories under an ASCI Initiative seed contract. It is a pleasure to offer our thanks to our

colleague Prof. Charbel Farhat for his unending enthusiasm and many helpful discussions. They also thank Dr. Dave Day of Sandia National Laboratory for running the 64-processor plate problem on a Sandia workstation. The first author was supported by a post-doctoral fellowship from the National Council for Research and Development (CNPq-Brazil).

REFERENCES

1. K. C. Park, M. R. Justino, Jr. and C. A. Felippa, 'An algebraically partitioned FETI method for parallel structural analysis: algorithm description', Center for Aerospace Structures, *Report Number CU-CAS-96-06*, University of Colorado at Boulder, 1996.
2. C. Farhat and F.-X. Roux, 'A method of finite element tearing and interconnecting and its parallel solution algorithm', *Int. j. numer. methods eng.*, **32,** 1205–1227 (1991).
3. C. Farhat and F.-X. Roux, 'Implicit parallel processing in structural mechanics', *Comput. Mech. Adv.*, **2,** 1–124 (1994).
4. C. Farhat, J. Mandel and F.-X. Roux, 'Optimal convergence properties of the FETI domain decomposition method', *Comput. Methods Appl. Mech. Eng.*, **115,** 367–388 (1994).
5. K. C. Park and G. M. Stanley, 'A curved C^0 shell element based on assumed natural-coordinate strains', *J. Appl. Mech.*, **53,** 278–290 (1986).
6. C. Farhat and J. Mandel, 'The two-level FETI method for static and dynamic plate problems—Part I: an optimal iterative solver for biharmonic systems', Center for Aerospace Structures, University of Colorado, *Report Number CU-CAS-95-23*, Boulder, CO, October, 1995.
7. C. Militello and C. A. Felippa, 'The first ANDES elements: 9-dof plate bending triangles', *Comput. Methods Appl. Mech. Eng.*, **93,** 217–246 (1991).
8. J. Mandel, R. Tezaur and C. Farhat, 'An optimal Lagrange multiplier based domain decomposition method for plate bending problems', Center for Aerospace Structures, University of Colorado, *Report Number CU-CAS-95-15*, Boulder, CO, July, 1995.

