INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING, VOL. 32, 1205–1227 (1991)

A METHOD OF FINITE ELEMENT TEARING AND INTERCONNECTING AND ITS PARALLEL SOLUTION ALGORITHM

CHARBEL FARHAT

Department of Aerospace Engineering Sciences and Center for Space Structures and Controls, University of Colorado at Boulder, Boulder, CO 80309–0429, U.S.A.

FRANCOIS-XAVIER ROUX

O.N.E.R.A. Groupe Calcul Parallele, 29 Av. de la Division Leclerc, BP72 92322 Chatillon Cedex, France

SUMMARY

A novel domain decomposition approach for the parallel finite element solution of equilibrium equations is presented. The spatial domain is partitioned into a set of totally disconnected subdomains, each assigned to an individual processor. Lagrange multipliers are introduced to enforce compatibility at the interface nodes. In the static case, each floating subdomain induces a local singularity that is resolved in two phases. First, the rigid body modes are eliminated in parallel from each local problem and a direct scheme is applied concurrently to all subdomains in order to recover each partial local solution. Next, the contributions of these modes are related to the Lagrange multipliers through an orthogonality condition. A parallel conjugate *projected* gradient algorithm is developed for the solution of the coupled system of local rigid modes components and Lagrange multipliers, which completes the solution of the problem. When implemented on local memory multiprocessors, this proposed method of tearing and interconnecting requires less interprocessor communications than the classical method of substructuring. It is also suitable for parallel/vector computers with shared memory. Moreover, unlike parallel direct solvers, it exhibits a degree of parallelism that is not limited by the bandwidth of the finite element system of equations.

1. INTRODUCTION

A number of methods based on domain decomposition procedures have been proposed in recent years for the parallel solution of both static and dynamic finite element equations of equilibrium. Most of these methods are derived from the popular substructuring technique. Typically, the finite element domain is decomposed into a set of subdomains and each of these is assigned to an individual processor. The solution of the local problems is trivially parallelized and usually a direct method is preferred for this purpose. Parallel implementations of both a direct[1] and an iterative[2] solution of the resulting interface problem have been reported in the literature. A number of more original approaches have also been spurred by the advent of new parallel processors. Ortiz and Nour-Omid[3] have developed a family of unconditionally stable concurrent procedures for transient finite element analysis and Farhat[4] has designed a multigrid-like algorithm for the massively parallel finite element solution of static problems. Both of these developments relate to the divide and conquer paradigm but depart from classical substructuring.

In this paper, we present a parallel finite element computational method for the solution of static problems that is also a departure from the classical method of substructures. The unique feature about the proposed procedure is that it requires fewer interprocessor communications

0029–5981/91/141205–23$11.50

Received 28 February 1990
Revised 30 July 1990

than traditional domain decomposition algorithms, while it still offers the same amount of parallelism. Roux[5,6] has presented an early version of this work that is limited to a very special class of problems where a finite element domain can be partitioned into a set of disconnected but non-floating subdomains. Here, we generalize the method for arbitrary finite element problems and arbitrary mesh partitions. We denote the resulting computational strategy by 'finite element tearing and interconnecting' because of its resemblance with the very early work of Kron[7] on tearing methods for electric circuit models. In Section 2, we partition the finite element domain into a set of totally disconnected subdomains and derive a computational strategy from a hybrid variational principle where the inter-subdomain continuity constraint is removed by the introduction of a Lagrange multiplier. An arbitrary mesh partition typically contains a set of floating subdomains which induce local singularities. The handling of these singularities is treated in Section 3. First, the rigid body modes are eliminated in parallel from each local problem and a direct scheme is applied concurrently to all subdomains in order to recover each partial local solution. Next, the contributions of these modes are related to the Lagrange multipliers through an orthogonality condition. A parallel conjugate *projected* gradient algorithm is developed in Section 4 for the solution of the coupled system of local rigid modes components and Lagrange multipliers, which completes the solution of the problem. Section 5 deals with the preconditioning of the interface problem in order to speed up the recovery of the Lagrange multipliers. Section 6 emphasizes the parallel characteristics of the proposed method and Section 7 contrasts it with the method of substructures. Section 8 discusses some important issues related to the partitioning of a given finite element mesh. Finally, Section 9 illustrates the method with structural examples on the distributed memory hypercube iPSC (32 processors) and the shared memory parallel/vector CRAY-2 system (4 processors), and Section 10 concludes the paper.

2. FINITE ELEMENT TEARING AND INTERCONNECTING

Here we present a domain decomposition based algorithm associated with a hybrid formulation for the parallel finite element solution of the linear elastostatic problem. However, the method is equally applicable to the finite element solution of any self-adjoint elliptic partial differential equation. For the sake of clarity, we consider first the case of two subdomains, then generalize the method for an arbitrary number of subdomains.

The variational form of the three-dimensional boundary-value problem to be solved goes as follows. Given f and h, find the displacement function u which is a stationary point of the energy functional

$$J(v) = \tfrac{1}{2}a(v, v) - (v, f) - (v, h)_\Gamma$$

where

$$\begin{aligned} a(v, w) &= \int_\Omega v_{(i,j)} c_{ijkl} w_{(k,l)} \, d\Omega \\ (v, f) &= \int_\Omega v_i f_i \, d\Omega \\ (v, h)_\Gamma &= \int_{\Gamma_h} v_i h_i \, d\Gamma \end{aligned} \tag{1}$$

In the above, the indices i, j, k take the values 1 to 3, $v_{(i,j)} = (v_{i,j} + v_{j,i})/2$ and $v_{i,j}$ denotes the partial derivative of the ith component of v with respect to the jth spatial variable, c_{ijkl} are the elastic coefficients, Ω denotes the volume of the elastostatic body, Γ its piecewise smooth boundary and Γ_h the piece of Γ where the tractions h_i are prescribed.

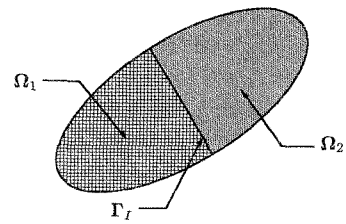

Figure 1. Decomposition in two subdomains

If Ω is subdivided into two regions Ω_1 and Ω_2 (Figure 1), solving the above elastostatic problem is equivalent to finding the two displacements functions u_1 and u_2 which are stationary points of the energy functionals

$$J_1(v_1) = \tfrac{1}{2}a(v_1, v_1)_{\Omega_1} - (v_1, f)_{\Omega_1} - (v_1, h)_{\Gamma_1}$$
$$J_2(v_2) = \tfrac{1}{2}a(v_2, v_2)_{\Omega_2} - (v_2, f)_{\Omega_2} - (v_2, h)_{\Gamma_2}$$

where

$$\begin{aligned}
a(v_1, w_1)_{\Omega_1} &= \int_{\Omega_1} v_{1(i,j)} c_{ijkl} w_{1(k,l)} \, \mathrm{d}\Omega \\
a(v_2, w_2)_{\Omega_2} &= \int_{\Omega_2} v_{2(i,j)} c_{ijkl} w_{2(k,l)} \, \mathrm{d}\Omega \\
(v_1, f)_{\Omega_1} &= \int_{\Omega_1} v_{1i} f_i \, \mathrm{d}\Omega \\
(v_2, f)_{\Omega_2} &= \int_{\Omega_2} v_{2i} f_i \, \mathrm{d}\Omega \\
(v_1, h)_{\Gamma_1} &= \int_{\Gamma_{h_1}} v_{1i} h_i \, \mathrm{d}\Gamma \\
(v_2, h)_{\Gamma_2} &= \int_{\Gamma_{h_2}} v_{2i} h_i \, \mathrm{d}\Gamma
\end{aligned} \tag{2}$$

and which satisfy on the interface boundary Γ_I the continuity constraint

$$u_1 = u_2 \quad \text{on } \Gamma_I \tag{3}$$

Solving the two above variational problems (2) with the subsidiary continuity condition (3) is equivalent to finding the saddle point of the Lagrangian:

$$J^*(v_1, v_2, \mu) = J_1(v_1) + J_2(v_1) + (v_1 - v_2, \mu)_{\Gamma_I} \tag{4}$$

where

$$(v_1 - v_2, \mu)_{\Gamma_I} = \int_{\Gamma_I} \mu(v_1 - v_2) \, \mathrm{d}\Gamma$$

—that is, finding the two displacement fields u_1 and u_2 and the Lagrange multipliers λ which satisfy

$$J^*(u_1, u_2, \mu) \leqslant J^*(u_1, u_2, \lambda) \leqslant J^*(v_1, v_2, \lambda) \tag{5}$$

for any admissible v_1, v_2 and μ. Clearly, the left inequality in (5) implies that $(u_1 - u_2, \mu)_{\Gamma_I} \leqslant (u_1 - u_2, \lambda)_{\Gamma_I}$, which imposes $(u_1 - u_2, \mu)_{\Gamma_I} = 0$ for any admissible μ and therefore $u_1 = u_2$ on Γ_I. The right inequality in (5) imposes $J_1(u_1) + J_2(u_2) \leqslant J_1(v_1) + J_2(v_2)$ for any pair of admissible functions (v_1, v_2). This implies that, among all admissible pairs (v_1, v_2) which satisfy the continuity condition (3), the pair (u_1, u_2) minimizes the sum of the energy functionals J_1 and J_2 defined respectively on Ω_1 and Ω_2. Therefore, u_1 and u_2 are the restriction of the solution u of the non-partitioned problem (1) to respectively Ω_1 and Ω_2. Indeed, equations (4) and (5) correspond to a hybrid variational principle where the inter-subdomain continuity constraint (3) is removed by the introduction of a Lagrange multiplier (see, for example, Pian[8]).

If now the displacement fields u_1 and u_2 are expressed by suitable shape functions as

$$u_1 = \mathbf{N}\mathbf{u}_1 \quad \text{and} \quad u_2 = \mathbf{N}\mathbf{u}_2$$

and the continuity equation is enforced for the discrete problem, a standard Galerkin procedure transforms the hybrid variational principle (4) in the following algebraic system:

$$\begin{aligned} \mathbf{K}_1\mathbf{u}_1 &= \mathbf{f}_1 + \mathbf{B}_1^{\mathrm{T}}\boldsymbol{\lambda} \\ \mathbf{K}_2\mathbf{u}_2 &= \mathbf{f}_2 - \mathbf{B}_2^{\mathrm{T}}\boldsymbol{\lambda} \\ \mathbf{B}_1\mathbf{u}_1 &= \mathbf{B}_2\mathbf{u}_2 \end{aligned} \tag{6}$$

where $\mathbf{K}_j$, $\mathbf{u}_j$ and $\mathbf{f}_j$, $j = 1, 2$, are respectively the stiffness matrix, the displacement vector and the prescribed force vector associated with the finite element discretization of Ω_j. The vector of Lagrange multipliers $\boldsymbol{\lambda}$ represents the interaction forces between the two subdomains Ω_1 and Ω_2 along their common boundary Γ_I. Within each subdomain Ω_j, we denote the number of interior nodal unknowns by n_j^{s} and the number of interface nodal unknowns by n_j^{I}. The total number of interface nodal unknowns is denoted by n_{I}. Note that $n_{\mathrm{I}} = n_1^{\mathrm{I}} = n_2^{\mathrm{I}}$ in the particular case of two subdomains. If the interior degrees of freedom are numbered first and the interface ones are numbered last, each of the two connectivity matrices $\mathbf{B}_1$ and $\mathbf{B}_2$ takes the form

$$\mathbf{B}_j = [\mathbf{O}_j \quad \mathbf{I}_j] \qquad j = 1, 2$$

where $\mathbf{O}_j$ is an $n_j^{\mathrm{I}} \times n_j^{\mathrm{s}}$ null matrix and $\mathbf{I}_j$ is the $n_j^{\mathrm{I}} \times n_j^{\mathrm{I}}$ identity matrix. The vector of Lagrange multipliers $\boldsymbol{\lambda}$ is n_{I} long.

If both $\mathbf{K}_1$ and $\mathbf{K}_2$ are non-singular, equations (6) can be written as

$$\begin{aligned} (\mathbf{B}_1\mathbf{K}_1^{-1}\mathbf{B}_1^{\mathrm{T}} + \mathbf{B}_2\mathbf{K}_2^{-1}\mathbf{B}_2^{\mathrm{T}})\boldsymbol{\lambda} &= \mathbf{B}_2\mathbf{K}_2^{-1}\mathbf{f}_2 - \mathbf{B}_1\mathbf{K}_1^{-1}\mathbf{f}_1 \\ \mathbf{u}_1 &= \mathbf{K}_1^{-1}(\mathbf{f}_1 + \mathbf{B}_1^{\mathrm{T}}\boldsymbol{\lambda}) \\ \mathbf{u}_2 &= \mathbf{K}_2^{-1}(\mathbf{f}_2 - \mathbf{B}_2^{\mathrm{T}}\boldsymbol{\lambda}) \end{aligned} \tag{7}$$

and the solution of (6) is obtained by solving the first of equations (7) for the Lagrange multipliers $\boldsymbol{\lambda}$, then substituting these in the second of (7) and backsolving for $\mathbf{u}_1$ and $\mathbf{u}_2$.

For an arbitrary number of subdomains Ω_j, the method goes as follows. First, the finite element mesh is 'torn' into a set of totally disconnected meshes (Figure 2).

For each mesh, the stiffness matrix $\mathbf{K}_j$ and the vector of prescribed forces $\mathbf{f}_j$ are formed. Next, for each Ω_j, a set of Boolean symbolic matrices $\mathbf{B}_j^k$ is set up to interconnect the mesh of Ω_j with those of its neighbours Ω_k. In general, $\mathbf{B}_j^k$ is $n_{\mathrm{I}} \times (n_j^{\mathrm{s}} + n_j^{\mathrm{I}})$ and has the following pattern:

$$\mathbf{B}_j^k = \begin{bmatrix} \mathbf{O}_1(j,k) \\ \mathbf{C}_j^k \\ \mathbf{O}_2(j,k) \end{bmatrix}$$

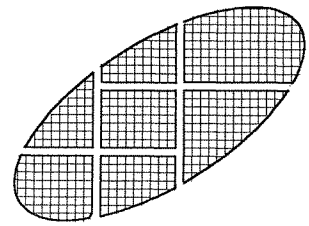

Figure 2. Finite element tearing

where $\mathbf{O}_1(j,k)$ is an $m_1(j,k)\times(n_j^s+n_j^l)$ zero matrix, $\mathbf{O}_2(j,k)$ is another $m_2(j,k)\times(n_j^s+n_j^l)$ zero matrix and $\mathbf{C}_j^k$ is an $m_c(j,k)\times(n_j^s+n_j^l)$ connectivity matrix, $m_c(j,k)$ is the number of Lagrange multipliers that interconnect Ω_j with its neighbour Ω_k, and $m_1(j,k)$ and $m_2(j,k)$ are two non-negative integers which satisfy $m_1(j,k)+m_c(j,k)+m_2(j,k)=n_l$. The connectivity matrix $\mathbf{C}_j^k$ can be written as

$$\mathbf{C}_j^k=[\mathbf{O}_3(j,k)\quad \mathbf{I}_j^k\quad \mathbf{O}_4(j,k)]$$

where $\mathbf{O}_3(j,k)$ is an $m_c(j,k)\times m_3(j,k)$ zero matrix, $\mathbf{I}_j^k$ is the $m_c(j,k)\times m_c(j,k)$ identity matrix, $\mathbf{O}_4(j,k)$ is another $m_c(j,k)\times m_4(j,k)$ zero matrix, and $m_3(j,k)$ and $m_4(j,k)$ are two non-negative integers which verify $m_3(j,k)+m_c(j,k)+m_4(j,k)=n_j^s+n_j^l$. If a_j and N_s denote respectively the number of subdomains Ω_k that are adjacent to Ω_j and the total number of subdomains, the finite element variational interpretation of the saddle-point problem (4) generates the following algebraic system:

$$\begin{aligned}&\mathbf{K}_j\mathbf{u}_j=\mathbf{f}_j+\sum_{k=1}^{k=a_j}\mathbf{B}_j^{kT}\boldsymbol{\lambda} && j=1,N_s\\ &\mathbf{B}_j^k\mathbf{u}_j=\mathbf{B}_k^j\mathbf{u}_k && j=1,N_s \text{ and } \Omega_k \text{ connected to } \Omega_j\end{aligned}\tag{8}$$

If $\mathbf{K}_j$ is non-singular for all $j=1,N_s$, the solution procedure (7) can be extended to the case of an arbitrary number of subdomains. However, the finite element tearing process described in this section may produce some 'floating' subdomains Ω_f which are characterized by a singular stiffness matrix $\mathbf{K}_f$. When this happens, the above solution algorithm (7) breaks down and a special computational strategy is required to handle the local singularities.

We refer to the computational procedure presented herein as the method of finite element tearing and interconnecting because of its resemblance with Kron's tearing method[7] for electric circuit models. We also note that the utility of Lagrange multipliers specifically for domain decomposition has also been previously recognized by other investigators.[9,10]

3. HANDLING LOCAL SINGULARITIES

Again, we focus on the two-subdomain tearing. The extrapolation to $N_s>2$ is straightforward. For example, suppose that Ω corresponds to a cantilever beam and that Ω_1 and Ω_2 are the result of a vertical partitioning (Figure 3).

In this case, $\mathbf{K}_1$ is positive definite and $\mathbf{K}_2$ is positive semi-definite since no boundary condition is specified over Ω_2. Therefore, the second of equations (6),

$$\mathbf{K}_2\mathbf{u}_2=\mathbf{f}_2-\mathbf{B}_2^T\boldsymbol{\lambda}\tag{9}$$

requires special attention. If the singular system (9) is consistent, a pseudo-inverse of $\mathbf{K}_2$ can be

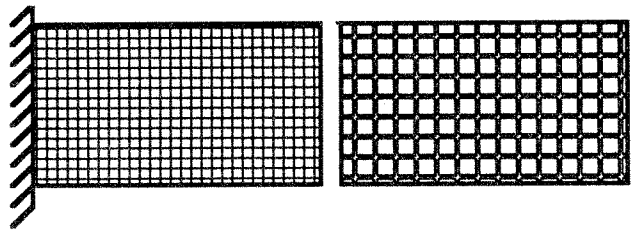

Figure 3. Decomposition resulting in a singular subdomain

found—that is a matrix $\mathbf{K}_2^+$ which verifies $\mathbf{K}_2\mathbf{K}_2^+\mathbf{K}_2 = \mathbf{K}_2$—and the general solution of (9) is given by

$$\mathbf{u}_2 = \mathbf{K}_2^+(\mathbf{f}_2 - \mathbf{B}_2^T\boldsymbol{\lambda}) + \mathbf{R}_2\boldsymbol{\alpha} \tag{10}$$

where $\mathbf{R}_2$ is an $(n_2^s + n_2^l) \times n_2^r$ rectangular matrix whose columns form a basis of the null space of $\mathbf{K}_2$, and $\boldsymbol{\alpha}$ is a vector of length n_2^r. Physically, $\mathbf{R}_2$ represents the rigid body modes of Ω_2 and $\boldsymbol{\alpha}$ specifies a linear combination of these. Consequently, we have $n_2^r \leqslant 6$ for three-dimensional problems, and $n_2^r \leqslant 3$ for two-dimensional problems. Substituting (10) into (7) leads to

$$\begin{aligned}(\mathbf{B}_1\mathbf{K}_1^{-1}\mathbf{B}_1^T + \mathbf{B}_2\mathbf{K}_2^+\mathbf{B}_2^T)\boldsymbol{\lambda} &= -\mathbf{B}_1\mathbf{K}_1^{-1}\mathbf{f}_1 + \mathbf{B}_2(\mathbf{K}_2^+\mathbf{f}_2 + \mathbf{R}_2\boldsymbol{\alpha})\\ \mathbf{u}_1 &= \mathbf{K}_1^{-1}(\mathbf{f}_1 + \mathbf{B}_1^T\boldsymbol{\lambda})\\ \mathbf{u}_2 &= \mathbf{K}_2^+(\mathbf{f}_2 - \mathbf{B}_2^T\boldsymbol{\lambda}) + \mathbf{R}_2\boldsymbol{\alpha}\end{aligned} \tag{11}$$

It should be noted that:

(i) because $\mathbf{B}_j$ is a Boolean operator, the result of its application to a matrix or vector quantity should be interpreted as an extraction process rather than a matrix–matrix or matrix–vector product. For example, $\mathbf{B}_2\mathbf{R}_2$ is the restriction of the local rigid modes $\mathbf{R}_2$ of Ω_2 to the interface unknowns. In the sequel we adopt the notation

$$\mathbf{R}_2^I = \mathbf{B}_2\mathbf{R}_2$$

(ii) the pseudo-inverse $\mathbf{K}_2^+$ does not need to be explicitly computed. For a given input vector $\mathbf{v}$, the output vector $\mathbf{K}_2^+\mathbf{v}$ and the rigid modes $\mathbf{R}_2$ can be obtained at almost the same computational cost as the response vector $\mathbf{K}_1^{-1}\mathbf{v}$, where $\mathbf{K}_1$ is non-singular (see Appendix I);

(iii) system (11) is under-determined. Both $\boldsymbol{\lambda}$ and $\boldsymbol{\alpha}$ need to be determined before $\mathbf{u}_1$ and $\mathbf{u}_2$ can be found, but only three equations are available so far.

Since $\mathbf{K}_2$ is symmetric, the singular equation (9) admits at least one solution if and only if the right hand side $(\mathbf{f}_2 - \mathbf{B}_2^T\boldsymbol{\lambda})$ has no component in the null space of $\mathbf{K}_2$. This can be expressed as

$$\mathbf{R}_2^T(\mathbf{f}_2 - \mathbf{B}_2^T\boldsymbol{\lambda}) = 0 \tag{12}$$

The above orthogonality condition provides the missing equation for the complete solution of (11). Combining (11) and (12) yields, after some algebraic manipulations,

$$\begin{aligned}\begin{bmatrix} \mathbf{F}_I & -\mathbf{R}_2^I \\ -\mathbf{R}_2^{IT} & \mathbf{O} \end{bmatrix}\begin{bmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\alpha} \end{bmatrix} &= \begin{bmatrix} \mathbf{B}_2\mathbf{K}_2^+\mathbf{f}_2 - \mathbf{B}_1\mathbf{K}_1^{-1}\mathbf{f}_1 \\ -\mathbf{R}_2^T\mathbf{f}_2 \end{bmatrix}\\ \mathbf{u}_1 &= \mathbf{K}_1^{-1}(\mathbf{f}_1 + \mathbf{B}_1^T\boldsymbol{\lambda})\\ \mathbf{u}_2 &= \mathbf{K}_2^+(\mathbf{f}_2 - \mathbf{B}_2^T\boldsymbol{\lambda}) + \mathbf{R}_2\boldsymbol{\alpha}\end{aligned} \tag{13}$$

where

$$\mathbf{F}_I = (\mathbf{B}_1\mathbf{K}_1^{-1}\mathbf{B}_1^T + \mathbf{B}_2\mathbf{K}_2^+\mathbf{B}_2^T)$$

Clearly, $\mathbf{F}_I$ is symmetric positive definite and $\mathbf{R}_2^I$ has full column rank. Therefore, the system of equations in (λ, α) is symmetric and non-singular. It admits a unique solution (λ, α) which uniquely determines $\mathbf{u}_1$ and $\mathbf{u}_2$.

It is important to note that, since $n_2^r \leqslant 6$, systems (13) and (7) have almost the same size. For an arbitrary number of subdomains N_s of which N_f are floating, the additional number of equations introduced by the handling of local singularities is bounded by $6N_f$. For large-scale problems and relatively coarse mesh partitions, this number is a very small fraction of the size of the global system. On the other hand, if a given tearing process does not result in any floating subdomain, α is zero and the systems of equations (13) and (7) are identical.

Next, we present a numerical algorithm for the solution of (13).

4. A PRECONDITIONED CONJUGATE PROJECTED GRADIENT ALGORITHM

Here we focus on the solution of the non-singular system of equations

$$\begin{bmatrix} \mathbf{F}_I & -\mathbf{R}_2^I \\ -\mathbf{R}_2^{IT} & \mathbf{O} \end{bmatrix} \begin{bmatrix} \lambda \\ \alpha \end{bmatrix} = \begin{bmatrix} \mathbf{B}_2\mathbf{K}_2^+\mathbf{f}_2 - \mathbf{B}_1\mathbf{K}_1^{-1}\mathbf{f}_1 \\ -\mathbf{R}_2^T\mathbf{f}_2 \end{bmatrix} \tag{14}$$

where

$$\mathbf{F}_I = \mathbf{B}_1\mathbf{K}_1^{-1}\mathbf{B}_1^T + \mathbf{B}_2\mathbf{K}_2^+\mathbf{B}_2^T$$

We seek an efficient solution algorithm that does not require the explicit assembly of $\mathbf{F}_I$.

The solution to the above problem can be expressed as

$$\begin{aligned} \lambda &= -\mathbf{H}(\mathbf{B}_2\mathbf{K}_2^+\mathbf{f}_2 - \mathbf{B}_1\mathbf{K}_1^{-1}\mathbf{f}_1) + \mathbf{T}\mathbf{R}_2^+\mathbf{f}_2 \\ \alpha &= \mathbf{T}^T(\mathbf{B}_1\mathbf{K}_1^{-1}\mathbf{f}_1 - \mathbf{B}_2\mathbf{K}_2^+\mathbf{f}_2) - \mathbf{U}\mathbf{R}_2^T\mathbf{f}_2 \end{aligned} \tag{15}$$

where

$$\begin{aligned} \mathbf{H} &= \mathbf{F}_I^{-1} - \mathbf{F}_I^{-1}\mathbf{R}_2^I\mathbf{U}^{-1}\mathbf{R}_2^{IT}\mathbf{F}_I^{-1} \\ \mathbf{T} &= \mathbf{F}_I^{-1}\mathbf{R}_2^I\mathbf{U}^{-1} \\ \mathbf{U} &= -\mathbf{R}_2^{IT}\mathbf{F}_I^{-1}\mathbf{R}_2^I \end{aligned}$$

As written in (15), this solution procedure is not recommended because it requires either the evaluation of the inverse of $\mathbf{F}_I$, or the nested solutions of two linear systems involving $\mathbf{F}_I$ and $\mathbf{R}_2^{IT}\mathbf{F}_I^{-1}\mathbf{R}_2^I$. It is noted by Fletcher[11] that if two matrices $\mathbf{S}$ and $\mathbf{Z}$ are computed such that

$$\begin{aligned} \mathbf{S}^T\mathbf{R}_2^I &= \mathbf{I} \\ \mathbf{Z}^T\mathbf{R}_2^I &= \mathbf{O} \end{aligned} \tag{16}$$

an alternative representation of the solution to (14) is given by

$$\begin{aligned} \lambda &= -\mathbf{H}(\mathbf{B}_2\mathbf{K}_2^+\mathbf{f}_2 - \mathbf{B}_1\mathbf{K}_1^{-1}\mathbf{f}_1) + \mathbf{T}\mathbf{R}_2^+\mathbf{f}_2 \\ \alpha &= \mathbf{T}^T(\mathbf{B}_1\mathbf{K}_1^{-1}\mathbf{f}_1 - \mathbf{B}_2\mathbf{K}_2^+\mathbf{f}_2) - \mathbf{U}\mathbf{R}_2^T\mathbf{f}_2 \end{aligned} \tag{17}$$

where

$$\mathbf{H} = \mathbf{Z}(\mathbf{Z}^T\mathbf{F}_I\mathbf{Z})^{-1}\mathbf{Z}^T$$

$$\mathbf{T} = \mathbf{S} - \mathbf{H}\mathbf{F}_I\mathbf{S}$$

$$\mathbf{U} = \mathbf{S}^T\mathbf{F}_I\mathbf{H}\mathbf{F}_I\mathbf{S} - \mathbf{S}^T\mathbf{F}_I\mathbf{S}$$

which does not require the explicit assembly of $\mathbf{F}_I$ if a suitable iterative scheme is chosen for solving all the temporary systems involving the quantity $(\mathbf{Z}^T\mathbf{F}_I\mathbf{Z})^{-1}$. Still, the above solution procedure is not feasible because it requires the computation of $\mathbf{S}$ and $\mathbf{Z}$—typically via a **QR** factorization of some matrix involving $\mathbf{R}_2^I$ [11]—and the iterative solution of too many temporary systems before λ and α can be obtained.

Clearly, the nature of $\mathbf{F}_I$ makes the solution of (14) inadequate by any technique which requires this submatrix explicitly. This implies that a direct method or an iterative method of the SOR type cannot be used. The only efficient method of solving (14) in the general sparse case is that of conjugate gradients, because once $\mathbf{K}_1$ and $\mathbf{K}_2$ have been factorized, matrix–vector products of the form $\mathbf{F}_I\mathbf{v}$ can be performed very efficiently using only forward and backward substitutions. Unfortunately, the Lagrangian matrix

$$\mathbf{L} = \begin{bmatrix} \mathbf{F}_I & -\mathbf{R}_2^I \\ -\mathbf{R}_2^{IT} & \mathbf{O} \end{bmatrix}$$

is indefinite so that a straightforward conjugate gradient algorithm cannot be directly applied to the solution of (14). However, the conjugate gradient iteration with the *projected* gradient (see, for example, Gill and Murray [12]) can be used to obtain the sought-after solution. In order to introduce the latter solution algorithm, we first note that solving (14) is equivalent to solving the equality constraint problem:

$$\begin{aligned} &\textit{minimize} \quad \Phi(\lambda) = \tfrac{1}{2}\lambda^T\mathbf{F}_I\lambda + (\mathbf{B}_1\mathbf{K}_1^{-1}\mathbf{f}_1 - \mathbf{B}_2\mathbf{K}_2^{+}\mathbf{f}_2)^T\lambda \\ &\textit{subject to} \quad \mathbf{R}_2^{IT}\lambda = \mathbf{R}_2^T\mathbf{f}_2 \end{aligned} \tag{18}$$

Since $\mathbf{F}_I$ is symmetric positive definite, a conjugate gradient algorithm is most suitable for computing the unique solution to the unconstrained problem. Therefore, this algorithm will converge to the solution to (18) if and only if it can be modified so that the constraint $\mathbf{R}_2^{IT}\lambda = \mathbf{R}_2^T\mathbf{f}_2$ is satisfied at each iteration. This can be achieved by projecting all the search directions onto the null space of $\mathbf{R}_2^I$.

The result is a conjugate gradient algorithm with the projected gradient. It is of the form

Initialize

Pick $\lambda^{(0)}$ such that $\mathbf{R}_2^{IT}\lambda^{(0)} = \mathbf{R}_2^T\mathbf{f}_2$

$$\mathbf{r}^{(0)} = (\mathbf{B}_2\mathbf{K}_2^{+}\mathbf{f}_2 - \mathbf{B}_1\mathbf{K}_1^{-1}\mathbf{f}_1) \tag{19}$$

Iterate $k = 1, 2, \ldots$ *until convergence*

$$\begin{aligned} &\beta^{(k)} = \mathbf{r}^{(k-1)T}\mathbf{r}^{(k-1)}/\mathbf{r}^{(k-2)T}\mathbf{r}^{(k-2)} \quad (\beta^{(1)} = 0) \\ &\mathbf{s}^{(k)} = \mathbf{r}^{(k-1)} + \beta^{(k)}\mathbf{s}^{(k-1)} \quad (\mathbf{s}^{(1)} = \mathbf{r}^{(0)}) \\ &\mathbf{s}^{(k)} = [\mathbf{I} - \mathbf{R}_2^I(\mathbf{R}_2^{IT}\mathbf{R}_2^I)^{-1}\mathbf{R}_2^{IT}]\mathbf{s}^{(k)} \\ &\gamma^{(k)} = \mathbf{r}^{(k-1)T}\mathbf{r}^{(k-1)}/\mathbf{s}^{(k)T}\mathbf{F}_I\mathbf{s}^{(k)} \\ &\lambda^{(k)} = \lambda^{(k-1)} + \gamma^{(k)}\mathbf{s}^{(k)} \\ &\mathbf{r}^{(k)} = \mathbf{r}^{(k-1)} - \gamma^{(k)}\mathbf{F}_I\mathbf{s}^{(k)} \end{aligned} \tag{20}$$

A fast scheme for finding a starting $\lambda^{(0)}$ which satisfies the constraint $\mathbf{R}_2^{\mathrm{I}^{\mathrm{T}}}\lambda^{(0)} = \mathbf{R}_2^{\mathrm{T}}\mathbf{f}_2$ is given in Appendix II. Clearly, $\mathbf{R}_2^{\mathrm{I}^{\mathrm{T}}}\mathbf{s}^{(k)} = 0$ for all $k \geqslant 1$. Therefore, $\mathbf{R}_2^{\mathrm{I}^{\mathrm{T}}}\lambda^{(k)} = \mathbf{R}_2^{\mathrm{I}^{\mathrm{T}}}\lambda^{(0)}$, which indicates that the approximate solution $\lambda^{(k)}$ satisfies the linear equality constraint of problem (14) at each iteration k. It is also important to note that, within each iteration, only one projection is performed. This projection is relatively inexpensive since the only implicit computations that are involved are associated with the matrix $\mathbf{R}_2^{\mathrm{I}^{\mathrm{T}}}\mathbf{R}_2^{\mathrm{I}}$ which is at most 6×6. This matrix is factored once, before the first iteration begins. Except for this small overhead, algorithm (20) above has the same computational cost as the regular conjugate gradient method.

After λ is found, the rigid body mode coefficients are computed as

$$\alpha = (\mathbf{R}_2^{\mathrm{I}^{\mathrm{T}}}\mathbf{R}_2^{\mathrm{I}})^{-1}(\mathbf{F}_{\mathrm{I}}\lambda - \mathbf{B}_2\mathbf{K}_2^{-1}\mathbf{f}_2 + \mathbf{B}_1\mathbf{K}_1^{-1}\mathbf{f}_1)$$

For an arbitrary number of subdomains N_s of which N_f are floating, the equality constraint is

$$\mathbf{R}^{\mathrm{I}^{\mathrm{T}}}\lambda = \begin{bmatrix} \mathbf{R}_1^{\mathrm{I}^{\mathrm{T}}} \\ \cdots \\ \mathbf{R}_f^{\mathrm{I}^{\mathrm{T}}} \end{bmatrix}\lambda = \mathbf{R}^{\mathrm{T}}\mathbf{f} = \begin{bmatrix} \mathbf{R}_1^{\mathrm{T}} & 0 & 0 \\ 0 & \cdots & 0 \\ 0 & 0 & \mathbf{R}_f^{\mathrm{T}} \end{bmatrix}\begin{bmatrix} \mathbf{f}_1 \\ \cdots \\ \mathbf{f}_f \end{bmatrix}$$

Only those columns of $\mathbf{R}_j^{\mathrm{T}}$ which operate on Lagrange multipliers that are associated with $\Gamma_{\mathrm{I}} \cap \Omega_j$ are non-zero. The projection matrix is $\mathbf{P} = [\mathbf{I} - \mathbf{R}^{\mathrm{I}}(\mathbf{R}^{\mathrm{I}^{\mathrm{T}}}\mathbf{R}^{\mathrm{I}})^{-1}\mathbf{R}^{\mathrm{I}^{\mathrm{T}}}]$, where $\mathbf{R}^{\mathrm{I}^{\mathrm{T}}}\mathbf{R}^{\mathrm{I}}$ is generally banded of dimension at most equal to $6N_f$. The banded structure of $\mathbf{P}$ is determined by the subdomains interconnectivity. If for practical reasons this banded structure is not exploited, the number of three-dimensional floating subdomains should be kept as small as possible, say less than thirty two, which implies that the proposed computational method would be suitable only for coarse and medium grain multiprocessors.

5. PRECONDITIONING THE INTERFACE PROBLEM

As in the case of the conjugate gradient method, the conjugate projected gradient algorithm is most effective when applied to the preconditioned system of equations. It should be noted that, even in the presence of floating subdomains, only $\mathbf{F}_{\mathrm{I}}$ needs to be preconditioned and not the global Lagrangian matrix $\mathbf{L}$. In the case of two subdomains, $\mathbf{F}_{\mathrm{I}}$ can be written in matrix form as

$$\mathbf{F}_{\mathrm{I}} = [\mathbf{B}_1 \quad \mathbf{B}_2]\begin{bmatrix} \mathbf{K}_1^{-1} & \mathbf{O} \\ \mathbf{O} & \mathbf{K}_2^{-1} \end{bmatrix}\begin{bmatrix} \mathbf{B}_1^{\mathrm{T}} \\ \mathbf{B}_2^{\mathrm{T}} \end{bmatrix} \tag{21}$$

where $\mathbf{K}_j^{-1}$, $j = 1, 2$, is replaced by $\mathbf{K}_j^{+}$ if Ω_j is a floating subdomain. The objective is to find an approximate inverse $\mathbf{P}_{\mathrm{I}}^{-1}$ of $\mathbf{F}_{\mathrm{I}}$ that: (a) does not need to be explicitly assembled (especially since $\mathbf{F}_{\mathrm{I}}$ is not explicitly assembled), and (b) that is amenable to parallel computations. The matrix $\mathbf{P}$ is then the preconditioner. Equation (21) above suggests the following choice for $\mathbf{P}_{\mathrm{I}}^{-1}$:

$$\mathbf{P}_{\mathrm{I}}^{-1} = [\mathbf{B}_1 \quad \mathbf{B}_2]\begin{bmatrix} \mathbf{K}_1 & \mathbf{O} \\ \mathbf{O} & \mathbf{K}_2 \end{bmatrix}\begin{bmatrix} \mathbf{B}_1^{\mathrm{T}} \\ \mathbf{B}_2^{\mathrm{T}} \end{bmatrix} \tag{22}$$

At each iteration k, the preconditioned conjugate projected gradient algorithm involves the solution of an auxiliary system of the form

$$\mathbf{P}_{\mathrm{I}}\mathbf{z}^{(k)} = \mathbf{r}^{(k)} \tag{23}$$

where $\mathbf{r}^{(k)}$ is the residual at the kth iteration. The particular choice of $\mathbf{P}_{\mathrm{I}}^{-1}$ given in (22) offers the advantage of solving (23) explicitly without the need for any intermediate factorization.

For computational efficiency, $\mathbf{P}_{\mathrm{I}}^{-1}$ is implemented as

$$\mathbf{P}_{\mathrm{I}}^{-1} = \mathbf{K}_1^{\mathrm{I}} + \mathbf{K}_2^{\mathrm{I}} \tag{24}$$

where $\mathbf{K}_1^I$ and $\mathbf{K}_2^I$ are the traces of $\mathbf{K}_1$ and $\mathbf{K}_2$ on Γ_I. Clearly, with this choice for the preconditioner, the auxiliary system (23) is 'cheap', easy to solve and perfectly parallelizable on both local and shared memory parallel architectures.

Since we do not have a strong mathematical justification for this choice of the preconditioner, we have conducted a set of numerical experiments to assess *a priori* its performance. A fixed–fixed cylindrical panel was discretized with an N by M regular mesh and was modelled with 4 node shell elements (Figure 4). All test cases used $N_s = 2$ and a vertical slicing.

Table I reports the condition numbers of the global stiffness matrix $\mathbf{K}$, the subdomain stiffness matrices $\mathbf{K}_1$ and $\mathbf{K}_2$, and the original and preconditioned interface flexibility-like matrices $\mathbf{F}_I$ and $\mathbf{P}_I^{-1}\mathbf{F}_I$, for various values of N.

For this test problem, the condition number of the preconditioned interface is two orders of magnitude lower than that of the global problem.

The extrapolation of (22) and (24) to $N_s > 2$ is straightforward. In order to reduce further the number of preconditioned conjugate projected gradient iterations, the selective reorthogonalization procedure developed by Roux and reported in Reference 13 is also utilized.

6. PARALLEL CHARACTERISTICS OF THE PROPOSED METHOD

Like most domain decomposition based algorithms, the proposed method of finite element tearing and interconnecting is perfectly suitable for parallel processing. If every subdomain Ω_j is assigned to an individual processor p_j, all local finite element computations can be performed in parallel. These include forming and assembling the stiffness matrix $\mathbf{K}_j$ and the forcing vector $\mathbf{f}_j$, factoring $\mathbf{K}_j$ and eventually computing the rigid modes $\mathbf{R}_j$, as well as backsolving for $\mathbf{u}_j$ after $\boldsymbol{\lambda}$ and $\boldsymbol{\alpha}$ have been determined. The conjugate projected gradient algorithm described in Section 4 is

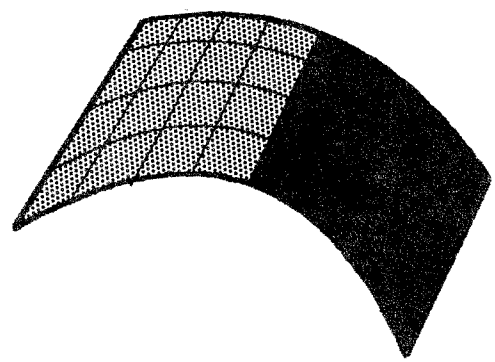

Figure 4. Cylindrical panel—$N_s = 2$

Table I. Condition numbers. Cylindrical panel—N by M mesh—shell elements—2 subdomains

N	M	$\kappa(\mathbf{K})$	$\kappa(\mathbf{K}_1)$	$\kappa(\mathbf{K}_2)$	$\kappa(\mathbf{F}_I)$	$\kappa(\mathbf{P}_I^{-1}\mathbf{F}_I)$
10	5	$2{\cdot}5\,10^4$	$5{\cdot}6\,10^3$	$5{\cdot}6\,10^3$	$1{\cdot}4\,10^4$	$4{\cdot}9\,10^2$
20	10	$3{\cdot}4\,10^5$	$2{\cdot}1\,10^4$	$2{\cdot}1\,10^4$	$2{\cdot}8\,10^4$	$3{\cdot}8\,10^3$
40	20	$5{\cdot}4\,10^6$	$9{\cdot}1\,10^4$	$9{\cdot}1\,10^4$	$1{\cdot}2\,10^5$	$3{\cdot}1\,10^4$

also amenable to parallel processing. For example, the matrix–vector product $\mathbf{F}_I\mathbf{s}^{(k)}$ can be computed in parallel by assigning to each processor p_j the task of evaluating $\mathbf{y}_j^{(k)} = \mathbf{B}_j^k\mathbf{K}_j^{-1}\mathbf{B}_j^{kT}\mathbf{s}_j^{(k)}$, and exchanging $\mathbf{y}_j^{(k)}$ with the processors assigned to neighbouring subdomains in order to assemble the global result. Interprocessor communication is required only during the solution of the interface problem (14) and takes place exclusively among neighbouring processors during the assembly of the subdomain results.

At this point, we stress that the parallel solution method developed herein requires inherently less interprocessor communication than other domain decomposition based parallel algorithms. As mentioned earlier, interprocessor communication within the proposed method occurs only during the solution of the interface problem (14). The reader should trace back this problem as well as the presence of the Lagrange multipliers to the integral quantity

$$(v_i - v_j, \lambda)_{\Gamma_{I_{ij}}} = \int_{\Gamma_{I_{ij}}} \lambda(v_i - v_j)\,d\Gamma \tag{25}$$

where $\Gamma_{I_{ij}}$ is the interface between subdomains Ω_i and Ω_j. If $\Gamma_{I_{ij}}$ has a zero measure, then $(v_i - v_j, \lambda)_{\Gamma_{I_{ij}}} = 0$ and no exchange of information is needed between Ω_i and Ω_j. Therefore the subdomains which interconnect along one edge in three-dimensional problems and those which interconnect along one vertex in both two- and three-dimensional problems do not require any interprocessor communication. This is unlike the parallel method of substructures, whether the interface problem is solved with a direct scheme[1] or with an iterative one.[2] For a three-dimensional regular mesh that is partitioned into subcubes, the proposed method of finite element tearing and interconnecting requires that each subdomain communicates with at most six neighbouring subdomains (since a cube has only six faces), while the parallel method of substructures necessitates that each subdomain communicates with up to 26 neighbours (Figure 5). This communication characteristic makes the proposed parallel solution method very attractive for a multiprocessor with a distributed memory such as a hypercube. Indeed, the advantages of the method for this family of parallel processors are twofold: (a) the number of message-passings is dramatically reduced, which reduces the overhead due to communication start-up, and (b) the complexity of the communication requirements is reduced so that an optimal mapping of the processors onto the subdomains can be reached;[14,15] therefore the elapsed time for a given message is improved. Both enhancements (a) and (b) reduce the communication overhead of the parallel solution algorithm in a synergistic manner. This algorithmic feature of the proposed method is still desirable for shared memory multiprocessors because it eases the assembly process during the interface solution and makes the latter more manageable. It is not, however, as critical for the performance as it is for local memory multiprocessors.

7. TEARING VS. SUBSTRUCTURING

Another difference between the subdomain based parallel solution method developed in this paper and the parallel method of substructures lies in the formulation of the interface problem. For the method of substructures, the interface problem corresponds to a *stiffness* formulation. For the two-subdomain decomposition it can be written as:

$$(\mathbf{K}_{II} - \mathbf{K}_{1I}^T\mathbf{K}_{11}^{-1}\mathbf{K}_{1I} - \mathbf{K}_{2I}^T\mathbf{K}_{22}^{-1}\mathbf{K}_{2I})\mathbf{u}_I = \mathbf{f}_{II} - \mathbf{K}_{1I}^T\mathbf{K}_{11}^{-1}\mathbf{f}_{11} - \mathbf{K}_{2I}^T\mathbf{K}_{22}^{-1}\mathbf{f}_{22} \tag{26}$$

where $\mathbf{K}_{II}$, $\mathbf{K}_{11}$ and $\mathbf{K}_{22}$ are the stiffness matrices associated respectively with the interface nodes and the interior nodes of subdomains Ω_1 and Ω_2, and $\mathbf{K}_{1I}$ and $\mathbf{K}_{2I}$ are the coupling stiffnesses between respectively Ω_1 and Γ_I and Ω_2 and Γ_I (see, for example Reference 1 for further details). A

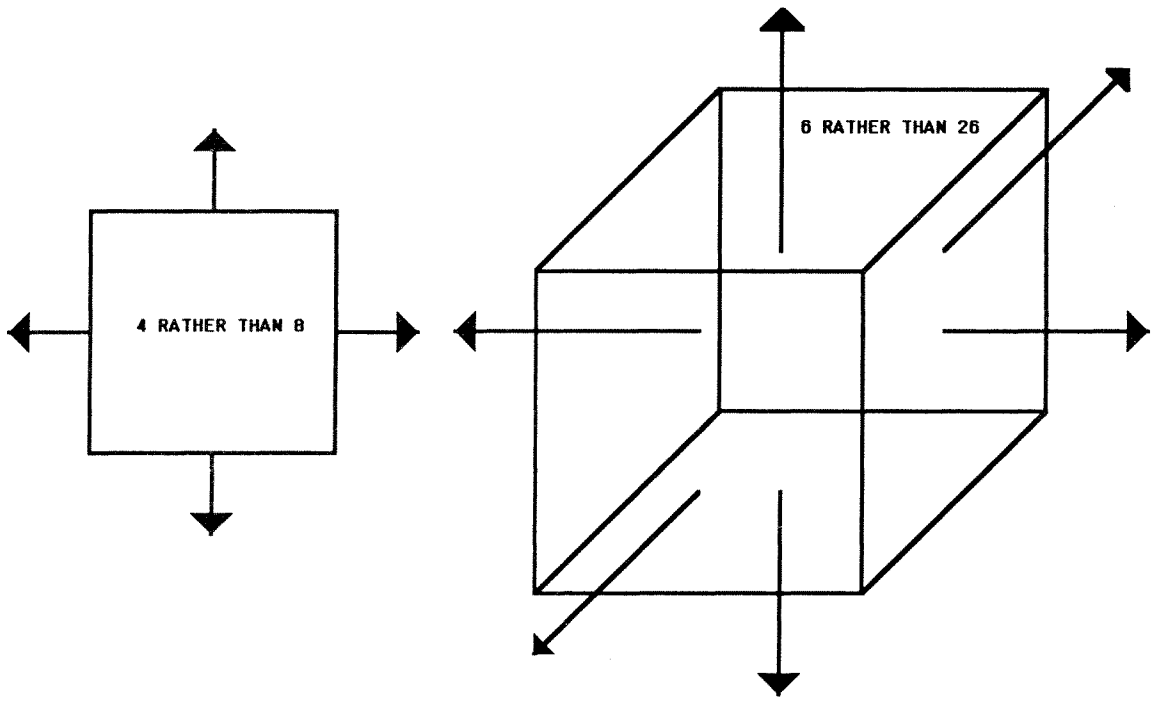

Figure 5. Reduced interprocessor communication patterns for two- and three-dimensional regular mesh partitions

standard conjugate gradient algorithm may be used for solving (26). On the other hand, the resulting interface problem for the method of finite element tearing and interconnecting corresponds to a formulation that is homogeneous to a *flexibility*, even if it is not exactly a flexibility formulation. For the two-subdomain decomposition, it can be written as in (14) and necessitates the use of a conjugate projected gradient algorithm for finding the solution λ.

If $\mathbf{K}_1$ and $\mathbf{K}_2$ are partitioned into internal and boundary (interface) components and then are injected into the first of equations (7), it can be easily shown that

$$\begin{aligned} \mathbf{B}_1\mathbf{K}_1^{-1}\mathbf{B}_1^{\mathrm{T}} &= (\mathbf{K}_{\mathrm{II}}^{(1)} - \mathbf{K}_{1\mathrm{I}}^{\mathrm{T}}\mathbf{K}_1^{-1}\mathbf{K}_{1\mathrm{I}})^{-1} \\ \mathbf{B}_2\mathbf{K}_2^{-1}\mathbf{B}_2^{\mathrm{T}} &= (\mathbf{K}_{\mathrm{II}}^{(2)} - \mathbf{K}_{2\mathrm{I}}^{\mathrm{T}}\mathbf{K}_2^{-1}\mathbf{K}_{2\mathrm{I}})^{-1} \end{aligned} \tag{27}$$

where $\mathbf{K}_{\mathrm{II}}^{(1)}$ and $\mathbf{K}_{\mathrm{II}}^{(2)}$ denote respectively the contributions of the first and second subdomains to $\mathbf{K}_{\mathrm{II}}$. Equations (27) above establish the relationship between both approaches to domain decomposition.

The computational implications of the differences between the two solution methods are as follows:

- within each iteration, the solution process of problem (14) requires an additional computational step which corresponds to the projection of the search direction onto the null space of $\mathbf{R}_2^{\mathrm{I}}$;
- within each iteration, the solution process of problem (14) requires the evaluation of the matrix–vector product $\mathbf{B}_j\mathbf{K}_j^{-1}\mathbf{B}_j^{\mathrm{T}}\mathbf{s}^{(k)}$, while the solution process of problem (26) requires the evaluation of the matrix–vector product $\mathbf{K}_{j\mathrm{I}}^{\mathrm{T}}\mathbf{K}_{jj}^{-1}\mathbf{K}_{j\mathrm{I}}\mathbf{s}^{(k)}$. Given that $\mathbf{B}_j$ is a Boolean matrix and that its application to a matrix or a vector defines a floating-point-free extraction

process, each conjugate gradient iteration applied to (14) is less computationally intensive than its counterpart that is applied to (26);

- since a conjugate gradient algorithm captures initially the high frequency mesh mode of a problem, it can be expected to perform better on a flexibility-like matrix than on a stiffness matrix because the high frequencies of the former are indeed the low frequencies of the stiffness matrix which are closer to the solution of the static problem.

In the light of the above remarks, it is reasonable to expect that for a given mesh partition:

- each conjugate projected gradient iteration that is applied to the solution of the interface problem (14) which results from the method of finite element tearing and interconnecting will not be slower—and may be even faster for large-scale problems and a small number of interface nodes—than each conjugate gradient iteration applied to the solution of the interface problem (26) which results from the method of substructures;
- the iterative solution of the interface problem associated with the tearing method will exhibit a faster rate of convergence than the iterative solution of the interface problem resulting from the conventional method of substructures.

Finally, it should be noted that domain decomposition methods in general exhibit a larger degree of parallelism than parallel direct solvers. The efficiency of the latter is governed by the bandwidth of the given finite element system of equations. If the bandwidth is not large enough, interprocessor communication and/or process synchronization can dominate the work done in parallel by each processor. This is true not only for multiprocessors with a message-passing system, but also for super-vector-multiprocessors with a shared memory such as the CRAY systems, where synchronization primitives are rather expensive. Therefore, the computational method described in this paper should be seriously considered for large-scale problems with a relatively small or medium bandwidth. These problems are typically encountered in the finite element analysis of large space structures which are often elongated and include only a few elements along one or two directions.[16] The method is also recommended for problems where the storage requirements of direct solvers cannot be met.

8. OPTIMAL MESH DECOMPOSITION

The computational method described in this paper requires that the given finite element mesh be partitioned into as many submeshes as there are available processors. In this section, we establish some guidelines for the design of an optimal mesh partition by analysing the effect of its structure on the performance of the global solution algorithm.

From the numerical point of view, the proposed solution method is hybrid in the sense it combines a direct and an iterative schemes. The direct solver is applied to each subdomain problem, the iterative one to the interface between these subdomains. If the mesh partition is such that the bandwidth of each subdomain problem is of the same order as the bandwidth of the global unpartitioned system of equations, the overall algorithm performs more operations than a direct method applied to the global unpartitioned system, independently of how fast the interface problem converges. The slicing of a parallelpiped along its largest dimension yields such a partition (Figure 6). If, on the other hand, the same parallelepiped is partitioned such that the bandwidth of each subdomain problem is much smaller than the bandwidth of the original finite element system (Figure 7), and if the convergence of the interface problem is fast enough, the method of finite element tearing and interconnecting may produce the solution with fewer computations than a global direct solver.

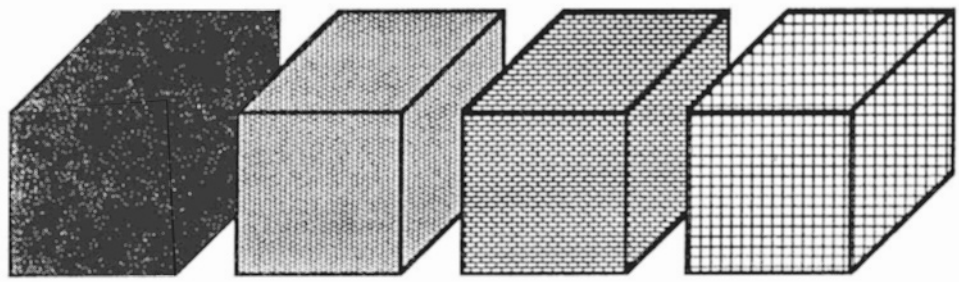

Figure 6. Stripwise partitioning of a parallelepiped

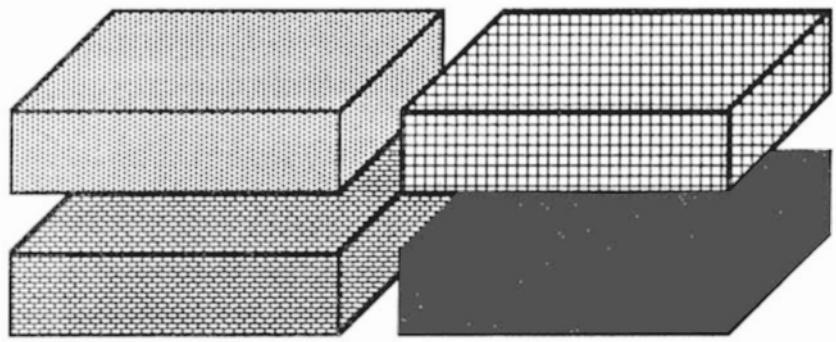

Figure 7. Boxwise partitioning of a parallelepiped

Besides conditioning, there are two other factors which affect the convergence of the interface problem (14) and which are directly related to the mesh partition: (a) the number of interface nodes, and (b) the interconnectivity of the subdomains along their interface. It can be easily checked that, within one iteration of the conjugate projected gradient algorithm, new information that is issued from a subdomain Ω_j reaches only those subdomains that interconnect with Ω_j along an edge or a plane. Therefore, the interface problem converges faster for a mesh partition that is characterized by a larger effective interconnectivity bandwidth.

The above observations suggest that an automatic finite element mesh decomposer that is suitable for the computational method described herein should meet or strike a balanced compromise between the seven following requirements:

(i) it should yield a set of subdomains where the bandwidths of each local problem is only a fraction of the bandwidth of the global system of equations;
(ii) it should keep the amount of interface nodes as small as possible in order to reduce the size of the interface problem;
(iii) it should yield a set of subdomains with a relatively high interconnectivity bandwidth so that within each iteration a new correction reaches as many subdomains as possible;
(iv) it should avoid producing subdomains with 'bad' aspect ratio (for example, elongated and flat subdomains) in order to keep the local problems as well conditioned as possible;
(v) it should deliver as few as possible floating subdomains in order to keep the cost associated with the projected gradients as low as possible;
(vi) it should yield a set of balanced subdomains in order to ensure that the overall

computational load will be as evenly distributed as possible among the processors;

(vii) it should be able to handle irregular geometry and arbitrary discretization in order to be general purpose.

For some mesh topologies, it becomes very difficult to meet simultaneously requirements (i), (ii) and (iv). In that case, priority should be given to the first two requirements. However, we have found that, for many problems, the above requirements can be met, using for example a slightly modified version of the general purpose finite element decomposer presented by Farhat in Reference 17. Several decomposition examples are described in Section 9. The most challenging problem that is yet to be resolved is the rational relationship between the mesh decomposition and the interface conditioning.

9. APPLICATIONS AND PERFORMANCE ASSESSMENT

We first illustrate the proposed parallel computational method with the static analysis on a 32 processor iPSC/2 hypercube of a three-dimensional mechanical joint subjected to internal pressure loading. We report performance results which show that the parallel method of tearing exhibits a better speed-up than the parallel method of conventional substructuring because it consumes three times less interprocessor communication. Next, we apply our algorithm to the large-scale finite element analysis on a 4 processor CRAY-2 of a three-dimensional cantilever composite beam made of more than one hundred stiff carbon fibres bound by a nearly incompressible elastomer matrix. We report and discuss in detail the measured performance results for various mesh partitioning strategies. For that problem, the proposed solution method outperforms the direct Choleski factorization by a factor greater than three, even for configurations that yield very ill-conditioned systems. In the following, NP, NE, NDF, T_{msg}, T_p and SP denote respectively the number of processors, the number of elements, the number of degrees of freedom, the time elapsed in message-passing, the total parallel time and the overall parallel speed-up.

The finite element discretization of the mechanical joint using 8 node brick elements is shown in Figure 8. Two meshes are considered. The first one contains 5002 elements, 14 932 degrees of freedom and is intended for a 16 processor cluster of the iPSC/2. The second mesh has 9912 elements, 29 654 degrees of freedom and is constructed for a 32 processor configuration of the same hypercube. The mesh decompositions into 16 and 32 subdomains are carefully designed to be topologically equivalent as much as possible to a checkerboard partitioning. Consequently, many of the resulting subdomains are floating.

The interprocessor communication time per iteration, the total parallel execution time and the overall parallel speed-up associated with the parallel method of tearing and the parallel method of substructures are reported in Table II for both meshes. For all cases, a tolerance of 10^{-3} on the global relative residuals is selected as a convergence criterion.

For both cases, the parallel tearing and parallel substructuring algorithms achieve excellent speed-up. This is generally true for all balanced algorithms that require message-passing only

Table II. Performance results on iPSC/2. Mechanical joint—brick elements—16 and 32 subdomains

NP	NE	NDF	T_{msg}/itr. subs.	T_{msg}/itr. tearing	T_p subs.	T_p tearing	SP subs.	SP tearing
16	5002	14 932	16·3 ms	5·2 ms	602 s	546 s	14·4	15·4
32	9912	29 654	17·9 ms	5·4 ms	1103 s	917 s	24·0	28·8

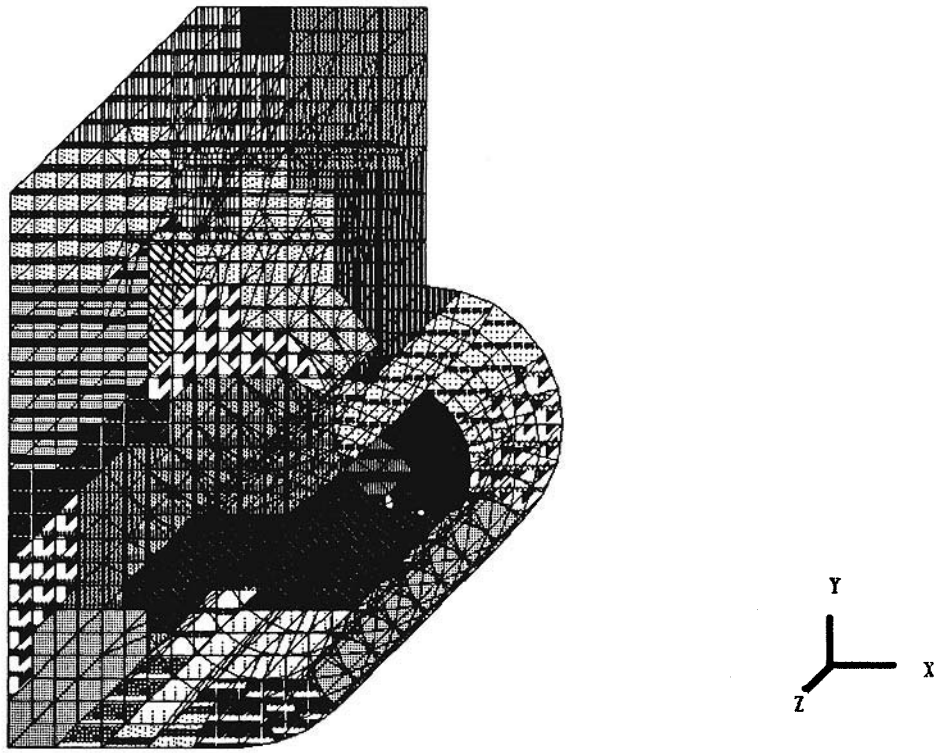

Figure 8. Finite element discretization of a mechanical joint

between neighbouring processors. However, for this problem, the tearing algorithm is faster and exhibits a 20 per cent higher speed-up than the conventional substructuring algorithm for which the time elapsed in interprocessor communication is 3·31 times higher. Again, because it avoids interprocessor communication along the edges and corners of the subdomains, the tearing algorithm requires fewer message-passing startups which, in the case of short messages, are known to account for the largest portion of the time elapsed in interprocessor communication on the iPSC/2 (see, for example, the benchmarks of Boman and Rose[18]). A performance comparison with a parallel direct solver is not provided because of the lack of memory space to store in-core the triangular factors of **K**.

Now that the parallel properties of the presented algorithm have been illustrated, we focus next on example problems that illustrate its intrinsic properties and performance. We consider the large-scale finite element static analysis of the pure bending of a set of beams made of similar jointed composite 'pencils' (Figure 9). Each composite pencil contains one carbon fibre with its elastomer matrix and is discretized in 51 vertical layers containing each 25 mesh points. The cross section of the finite element mesh corresponding to a 16 pencil beam is shown in Figure 10.

The numerical results obtained on a 4 processor CRAY-2 for a 16 pencil beam with 48 000 degrees of freedom are reported in Figures 11 and 12. These correspond to two extreme mesh decompositions, namely: a horizontal cross-slicing into 4 subdomains each containing 4 cantilever parallel pencils (D1), and (b) a vertical slicing into 4 subdomains, of which three are floating (D2). Poisson's ratio for the elastomer is 0·49.

For each decomposition case, three curves are reported which correspond to monitoring convergence with three different measures: (A) the global force relative residual, (B) the displace-

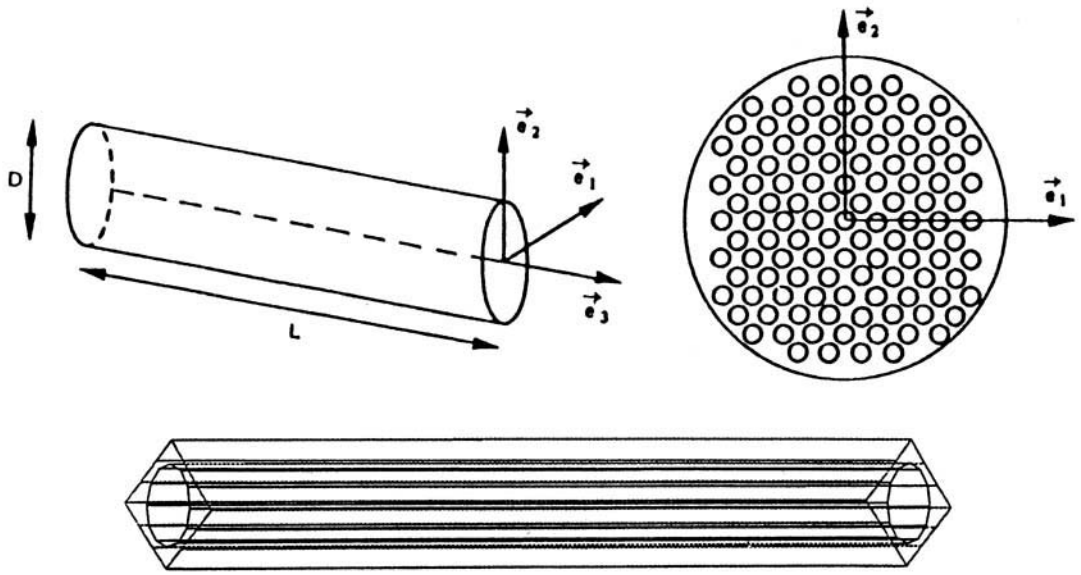

Figure 9. A composite beam and a composite pencil

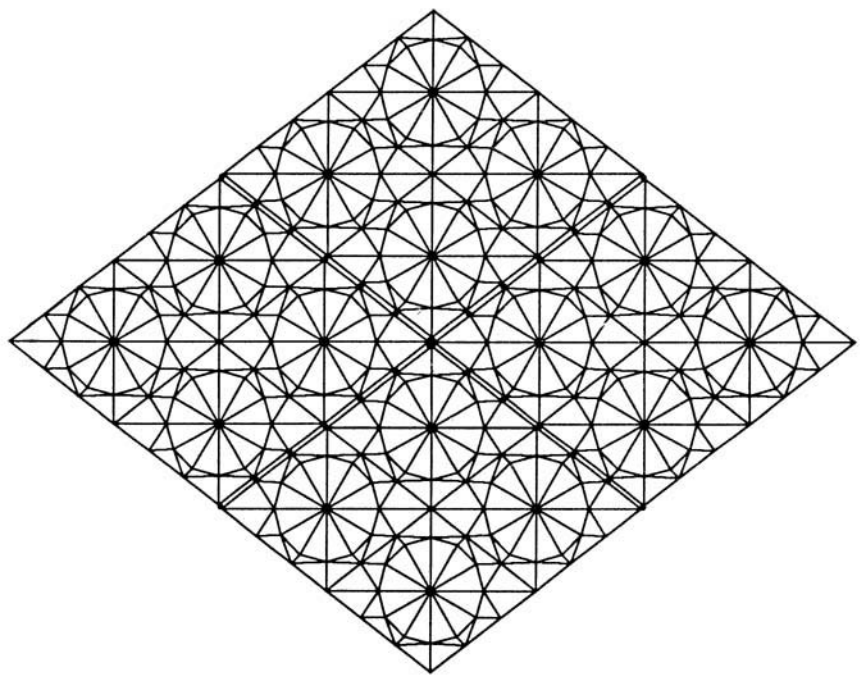

Figure 10. Cross section of the finite element mesh for a 16 pencil composite beam

ment relative variation, and (C) the interface relative residual. Clearly, decomposition D1 induces a faster convergence rate than decomposition D2. We have predicted this result since, within each iteration of the iterative solution of the interface problem, information reaches all of the subdomains in decomposition D1, while it reaches only half of these in decomposition D2.

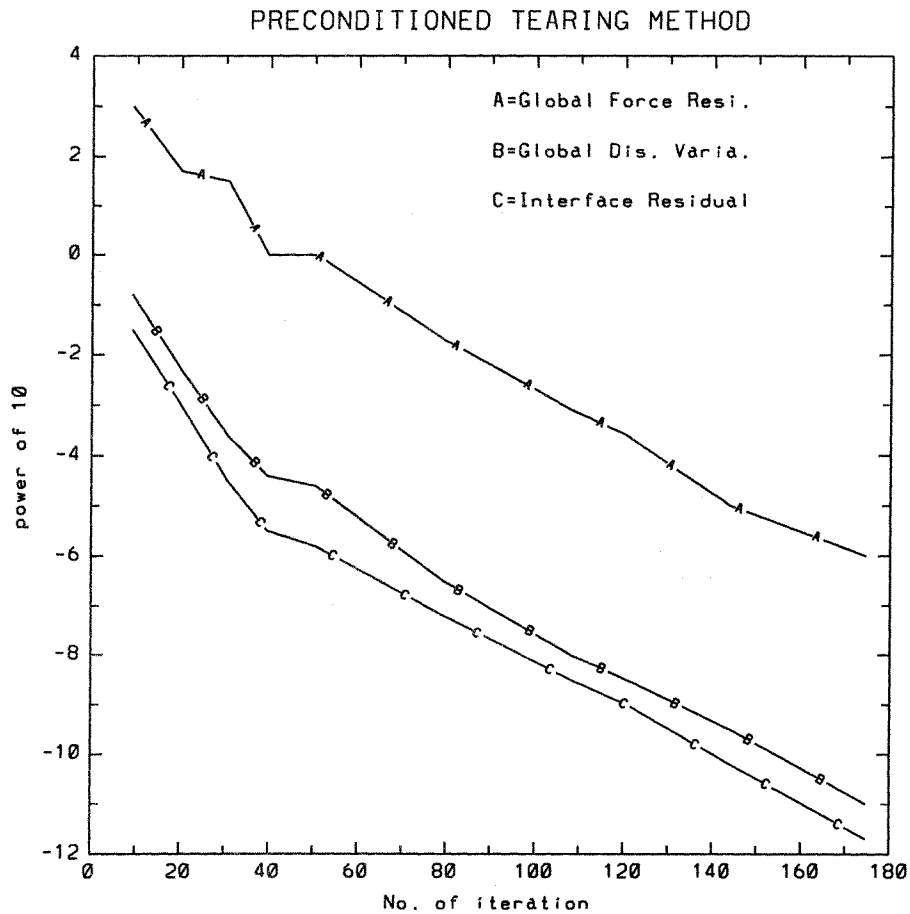

Figure 11. Numerical results for decomposition D1

Another important result relates to the relative positioning of the three curves, independently from the decomposition pattern. Note first that convergence with the global force relative residual is harder to achieve than convergence with the displacement relative variation. This is because the problem suffers from a severe ill-conditioning owing to the incompressibility of the elastomer (Poisson ratio = 0·49) and the elongated shape of the cantilever composite beam. Note also that convergence with the interface relative residual is closer to convergence with the displacement relative variation than it is to convergence with the global force relative residual. This is because the interface problem is formulated in the functional space of the stresses, so that its residuals correspond to a displacement increment.

Finally, the tearing method is compared for performance with a direct Cholesky algorithm. The factorization routine is highly optimized for CRAY-like supercomputers: it is fully vectorized and its memory traffic requirements are reduced via an unrolling procedure. In order to have a meaningful comparison, we use the same factorization routine for the Cholesky global algorithm and for factoring the subdomain stiffnesses in the tearing algorithm. The same bending problem as previously is selected for that purpose. Several different mesh configurations which correspond to different numbers of pencils are considered. Performance results on a CRAY-2 single processor

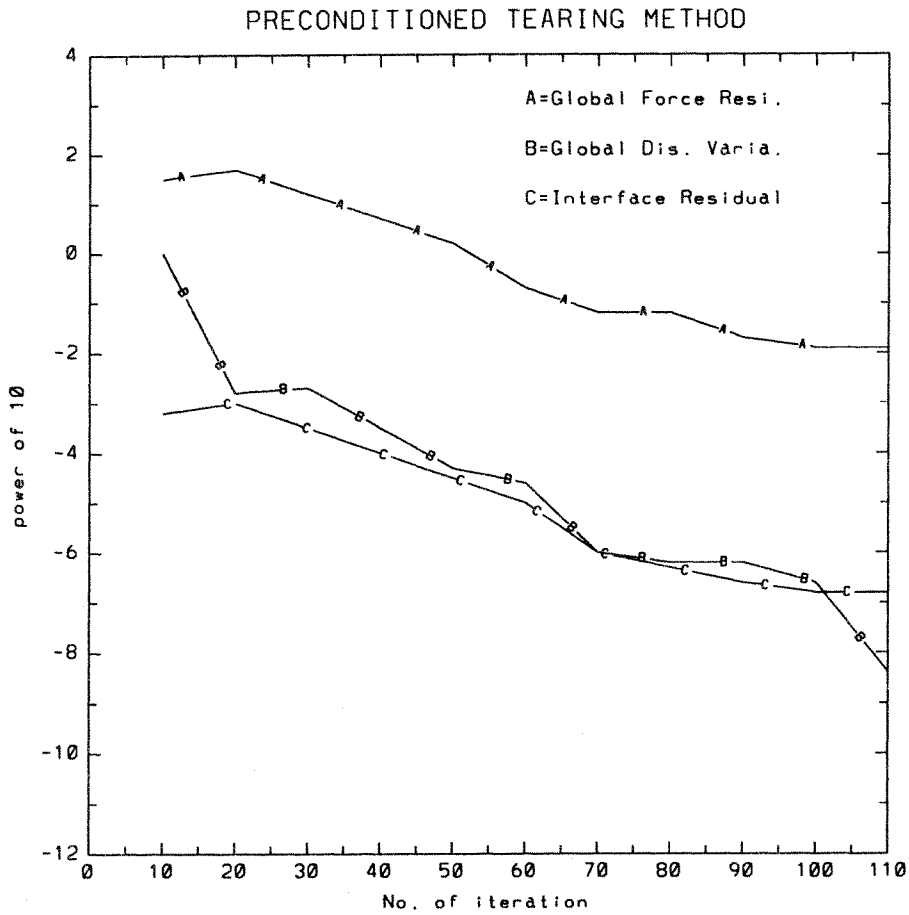

Figure 12. Numerical results for decomposition D2

are reported in Table III. A tolerance of 10^{-6} on the global relative residuals is selected as a convergence criterion.

The above results demonstrate that, for sufficiently large problems, the tearing method can outperform direct solvers. For the particular problem above, it runs up to 3·3 times faster than Cholesky factorization and requires 5·2 times less memory space.

10. CLOSURE AND OVERVIEW OF SUBSEQUENT RESEARCH

A novel domain decomposition approach for the parallel finite element solution of equilibrium equations is presented. The spatial domain is partitioned into a set of totally disconnected subdomains, each assigned to an individual processor. Lagrange multipliers are introduced to enforce compatibility at the interface nodes. In the static case, each floating subdomain induces a local singularity that is resolved in two phases. First, the rigid body modes are eliminated in parallel from each local problem and a direct scheme is applied concurrently to all subdomains in order to recover each partial local solution. Next, the contributions of these modes are related to the Lagrange multipliers through an orthogonality condition. A parallel conjugate *projected*

Table III. Performance results on CRAY-2. Composite beam—brick elements—direct vs. 4-subdomain tearing

Number of pencils	4	9	16
NDF	13 000	28 000	48 000
4-subdomain	*tearing*	*method*	
NDF interface	2450	7350	14 700
# iterations	130	210	300
CPU time	20 s	73 s	193 s
Memory size	1·6 m.w.	4·5 m.w.	9·5 m.w
Global	*Cholesky factorization*		
CPU time	15 s	130 s	650 s
Memory size	3·6 m.w.	16 m.w.	50 m.w.

gradient algorithm is developed for the solution of the coupled system of local rigid modes components and Lagrange multipliers, which completes the solution of the problem. When implemented on local memory multiprocessors, this proposed method of tearing and interconnecting requires less interprocessor communications than the classical method of substructuring. It is also suitable for parallel/vector computers with shared memory. Large-scale example applications are reported on the iPSC/1 and CRAY-2. Measured performance results illustrate the advantages of the proposed method and demonstrate its potential to outperform the classical method of substructures and parallel direct solvers.

It is our experience that domain decomposition methods are very sensitive to the mesh partition. In this paper, we have outlined some guidelines for the practical decomposition of a given finite element mesh. Subsequent research will focus on determining the relationship between a pattern of decomposition and the resulting conditioning of each of the local problems and the interface one. While several preconditioners for conventional domain decomposition methods (Schur methods) are available in the literature, further research is needed to develop a preconditioner for hybrid domain decomposition algorithms such as the tearing method developed herein.

ACKNOWLEDGEMENTS

The first author would like to thank Pr. M. Geradin at the University of Liege, Belgium, for his valuable comments. He also wishes to acknowledge partial support by the National Science Foundation under Grant ASC–8717773, and partial support by the Air Force Office of Scientific Research under Grant AFOSR–89–0422.

APPENDIX I: SOLVING A CONSISTENT SINGULAR SYSTEM $\mathbf{K}_j\mathbf{u}_j = \mathbf{f}_j$

For completeness, we include in this Appendix a derivation of the solution of a consistent singular system of equations. In this work, such a system arises in every floating subdomain Ω_j and takes the form

$$\mathbf{K}_j\mathbf{u}_j = \mathbf{f}_j \tag{28}$$

where $\mathbf{K}_j$ is the $(n_j^s + n_j^l) \times (n_j^s + n_j^l)$ stiffness matrix associated with Ω_j, and $\mathbf{u}_j$ and $\mathbf{f}_j$ are the corresponding displacement and forcing vectors. If Ω_j has n_j^r rigid body modes, $\mathbf{K}_j$ is rank n_j^r deficient. Provided that $\mathbf{f}_j$ is orthogonal to the null space of $\mathbf{K}_j$, the singular system (28) is consistent and admits a general solution of the form

$$\mathbf{u}_j = \mathbf{K}_j^+ \mathbf{f}_j + \mathbf{R}_j \boldsymbol{\alpha} \tag{29}$$

where $\mathbf{K}_j^+$ is a pseudo-inverse of $\mathbf{K}_j$—that is, $\mathbf{K}_j^+$ verifies $\mathbf{K}_j \mathbf{K}_j^+ \mathbf{K}_j = \mathbf{K}_j$, $\mathbf{R}_j$ is a basis of the null space of $\mathbf{K}_j$—that is, $\mathbf{R}_j$ stores the n_j^r rigid body modes of Ω_j, and $\boldsymbol{\alpha}$ is a vector of length n_j^r containing arbitrary real coefficients.

1. Computing the rigid body modes

Let the superscripts p and r denote respectively a principal and a redundant quantity. The singular stiffness matrix $\mathbf{K}_j$ is partitioned as

$$\mathbf{K}_j = \begin{bmatrix} \mathbf{K}_j^{pp} & \mathbf{K}_j^{pr} \\ \mathbf{K}_j^{prT} & \mathbf{K}_j^{rr} \end{bmatrix} \tag{30}$$

where $\mathbf{K}_j^{pp}$ has full rank equal to $n_j^s + n_j^l - n_j^r$. If $\mathbf{R}_j$ is defined as

$$\mathbf{R}_j = \begin{bmatrix} -\mathbf{K}_j^{pp-1} \mathbf{K}_j^{pr} \\ \mathbf{I}_{n_j^r} \end{bmatrix} \tag{31}$$

where $\mathbf{I}_{n_j^r}$ is the $n_j^r \times n_j^r$ identity matrix, then $\mathbf{R}_j$ satisfies

$$\mathbf{K}_j \mathbf{R}_j = 0$$

Moreover, $\mathbf{I}_{n_j^r}$ has full column rank and so does $\mathbf{R}_j$. Therefore, the n_j^r columns of $\mathbf{R}_j$ as defined in (29) form a basis of the null space of $\mathbf{K}_j$.

2. Computing $\mathbf{K}_j^+ \mathbf{f}_j$

The partitioning of the singular matrix $\mathbf{K}_j$ defined in (30) implies that

$$\mathbf{K}_j^{rr} = \mathbf{K}_j^{prT} \mathbf{K}_j^{pp-1} \mathbf{K}_j^{pr} \tag{32}$$

Using the above identity, it can be easily checked that the matrix $\mathbf{K}_j^+$ defined as

$$\mathbf{K}_j^+ = \begin{bmatrix} \mathbf{K}_j^{pp-1} & \mathbf{O} \\ \mathbf{O} & \mathbf{O} \end{bmatrix}$$

is a pseudo-inverse of $\mathbf{K}_j$. Therefore, a solution of the form $\mathbf{K}_j^+ \mathbf{f}_j$ can be also written as

$$\mathbf{u}_j = \mathbf{K}_j^+ \mathbf{f}_j = \begin{bmatrix} \mathbf{K}_j^{pp-1} \mathbf{f}_j^p \\ \mathbf{O} \end{bmatrix}$$

In practice, $\mathbf{K}_j$ cannot be explicitly re-arranged as in (30). Rather, the following should be implemented when $\mathbf{K}_j$ is stored in skyline form. A zero pivot that is encountered during the factorization process of $\mathbf{K}_j$ corresponds to a redundant equation which needs to be labelled and removed from the system. The zero pivot is set to one, the reduced column above it is copied into an extra right hand side—this corresponds to a forward reduction with $\mathbf{K}_j^{pr} \mathbf{u}_j^r$ as right hand side—and the coefficients in the skyline corresponding to that pivotal equation are set to zero. At the end of the factorization process, the non-labelled equations define the full rank matrix $\mathbf{K}_j^{pp}$. The

backward substitution is modified to operate also on the n_j^r extra right hand sides in order to recover $\mathbf{u}_j^p = -\mathbf{K}_j^{pp-1}\mathbf{K}_j^{pr}\mathbf{u}_j^r$.

The above procedure for solving a consistent singular system of equations has almost the same computational complexity as the solution of a non-singular one.

APPENDIX II: STARTING LAGRANGE MULTIPLIER VECTOR

In this Appendix we present a fast scheme for generating a starting vector $\lambda^{(0)}$ for the conjugate projected gradient algorithm (19)–(20). We consider the general case of an arbitrary mesh partition.

For each floating subdomain Ω_j, the corresponding component of the starting vector has to satisfy the equality constraint

$$\mathbf{R}_j^{I^T}\lambda_j^{(0)} = \mathbf{R}_j^T\mathbf{f}_j \tag{33}$$

where $\mathbf{R}_j$ is an $(n_j^s + n_j^I) \times n_j^r$ full column rank matrix which stores the rigid body modes of the floating subdomain Ω_j, $\mathbf{R}_j^I$ is the restriction of $\mathbf{R}_j$ to the intersection of Ω_j and the interface Γ_I, and $\mathbf{f}_j$ is the vector of prescribed forces in Ω_j. If $\lambda_j^{(0)}$ is written as

$$\lambda_j^{(0)} = \mathbf{R}_j^I\mu_j^{(0)} \tag{34}$$

then (33) becomes

$$(\mathbf{R}_j^{I^T}\mathbf{R}_j^I)\mu_j^{(0)} = \mathbf{R}_j^T\mathbf{f}_j \tag{35}$$

which admits as solution

$$\mu_j^{(0)} = (\mathbf{R}_j^{I^T}\mathbf{R}_j^I)^{-1}\mathbf{R}_j^T\mathbf{f}_j \tag{36}$$

Therefore, a starting vector $\lambda_j^{(0)}$ which satisfies the constraint equation (33) is given by

$$\lambda_j^{(0)} = \mathbf{R}_j^I(\mathbf{R}_j^{I^T}\mathbf{R}_j^I)^{-1}\mathbf{R}_j^T\mathbf{f}_j \tag{37}$$

The matrix product $(\mathbf{R}_j^{I^T}\mathbf{R}_j^I)$ is only $n_j^r \times n_j^r$, where n_j^r is the number of rigid body modes of the floating subdomain Ω_j. Therefore, $(\mathbf{R}_j^{I^T}\mathbf{R}_j^I)$ is at most 3×3 in two-dimensional problems and at most 6×6 in three-dimensional problems, and the evaluation of $\lambda_j^{(0)}$ according to (37) requires little computational effort.

REFERENCES

1. C. Farhat and E. Wilson, 'A new finite element concurrent computer program architecture', *Int. j. numer. methods eng.*, **24**, 1771–1792 (1987).
2. B. Nour-Omid, A Raefsky and G. Lyzenga, 'Solving finite element equations on concurrent computers', in A. K. Noor (ed.), *Parallel Computations and Their Impact on Mechanics*, ASME, New York, 1987, pp. 209–228.
3. M. Ortiz and B. Nour-Omid, 'Unconditionally stable concurrent procedures for transient finite element analysis', *Comp. Methods Appl. Mech. Eng.*, **58**, 151–174 (1986).
4. C. Farhat, 'A multigrid-like semi-iterative algorithm for the massively parallel solution of large scale finite elements systems', *Multigrid Methods: Proc. Fourth Copper Mountain Conf. on Multigrid Methods*, SIAM, Copper Mountain, Colorado, 1989, pp. 171–180.
5. F. X. Roux, 'Test on parallel machines of a domain decomposition method for a composite three dimensional structural analysis problem', *Proc. Int. Conf. on Supercomputing*, Saint Malo, France, 1988, pp. 273–283.
6. F. X. Roux, 'A parallel solver for the linear elasticity equations on a composite beam', in T. F. Chan *et al.* (eds.), *Proc. Second Int. Conf. on Domain Decomposition Methods*, SIAM, Los Angeles, California, 1989.
7. G. Kron, 'A set of principles to interconnect the solutions of physical systems', *J. Appl. Phys.*, **24**, 965–980 (1953).
8. T. H. H. Pian, 'Finite element formulation by variational principles with relaxed continuity requirements', in A. K. Aziz (ed.), *The Mathematical Foundation of the Finite Element Method with Applications to Partial Differential Equations, Part II*, Academic Press, London, 1972, pp. 671–687.

9. Q. V. Dihn, R. Glowinski and J. Periaux, 'Solving elliptic problems by domain decomposition methods with applications', in A. Schoenstadt (ed.), *Elliptic Problem Solvers II*, Academic Press, London, 1984.
10. M. R. Dorr, 'Domain decomposition via Lagrange multipliers', *Report No. UCRL-98532*, Lawrence Livermore National Laboratory, 1988.
11. R. Fletcher, *Practical Methods of Optimization, Vol. 2, Constrained Optimization*, Wiley, New York, 1981, pp. 86–87.
12. P. E. Gill and W. Murray, in P. E. Gill and W. Murray (eds.), *Numerical Methods for Constrained Optimization*, Academic Press, London, 1974, pp. 132–135.
13. F. X. Roux, 'Acceleration of the outer conjugate gradient by reorthogonalization for a domain decomposition method for structural analysis problems', *Proc. Third Int. Conf. on Supercomputing*, Crete, Greece, 1989, pp. 471–477.
14. S. H. Bokhari, 'On the mapping problem', *IEEE Trans. Comp.*, **C-30**, 207–214 (1981).
15. C. Farhat, 'On the mapping of massively parallel processors onto finite element graphs', *Comp. Struct.*, **32**, 347–354 (1989).
16. C. Farhat, 'Which parallel finite element algorithm for which architecture and which problem', in R. V. Grandhi *et al.* (eds.), *Computational Structural Mechanics and Multidisciplinary Optimization*, AD-Vol. 16, ASME, New York, pp. 35–43.
17. C. Farhat, 'A simple and efficient automatic FEM domain decomposer', *Comp. Struct.*, **28**, 579–602 (1988).
18. L. Bomans and D. Roose, 'Benchmarking the iPSC/2', *Report TW 114*, Katholieke Universiteit Leuven, Department of Computer Science, Belgium, 1988.