CHAPTER 22

A Saddle-Point Principle Domain Decomposition Method for the Solution of Solid Mechanics Problems

Charbel Farhat*

Abstract. A stationary variational principle is used to derive a finite element domain decomposition method for the solution of structural mechanics problems. A given mesh is partitioned into disconnected submeshes where a local solution for the displacement field is first evaluated. Intersubdomain field continuity is enforced via discrete, polynomial, and/or piece-wise polynomial Lagrange multipliers. In the static case, each floating subdomain induces a local singularity and triggers rigid body (zero energy) mode effects . The interface problem associated with this domain decomposition method is in general indefinite and of variable size. A dedicated preconditioned conjugate gradient algorithm is developed for solving the latter problem when it is not feasible to explicitly assemble the interface operator. At each iteration, the zero energy modes are filtered out via a suitable projector which somehow limits the scalability of the methodology. To alleviate this problem, the rigid body modes are preprocessed with a QR factorization which results in a simpler expression for the projector. The proposed methodology is intrinsically parallel. It offers attractive features for local memory multiprocessors, and is also suitable for shared memory parallel/vector computers. Numerical and performance results are reported. They demonstrate the potential of the methodology for real-life solid mechanics problems.

1. Introduction. In this paper, a finite element domain decomposition method is derived from a saddle-point variational principle. In many cases, the proposed method performs fewer operations than other global and subdomain based solution techniques, which makes it interesting even as a serial algorithm. It is also characterized by a small amount of communication requirements, which makes it particularly attractive for local memory parallel processors. Preliminary versions of this method were introduced by Destuynder and Roux [2], then Farhat and Roux

* Center for Space Structures and Controls, University of Colorado, Boulder, CO 80309-0429.

[8]. In a companion paper, Roux [12] establishes an interesting duality between the proposed methodology and the domain decomposition algorithm of Bjordstad and Widlund [1], and points out some important consequences.

Here, the basic method introduced by Farhat and Roux in [8] is refined in view of: (a) augmenting it with a model reduction algorithm, and (b) improving its scalability to a large number of subdomains, and therefore to a large number of processors. For this purpose, the remainder of this paper is organized as follows. In Sections 2 and 3, a subdomain-by-subdomain finite element method is derived from a stationary variational principle where an inter-subdomain continuity constraint is removed via the introduction of discrete, polynomial, and piecewise polynomial Lagrange multipliers; its essential computational and parallel features are highlighted. Singularity issues associated with the methodology are raised and resolved in Section 4. A preconditioned conjugate *projected* gradient algorithm (PCPG) is developed in Section 5 for the solution of the corresponding indefinite interface problem. Section 6 deals with scalability issues that are raised by the PCPG's projector. Section 7 illustrates the method with the parallel solution of real-life structural problems on an iPSC/2 hypercube. Finally, Section 8 offers some conclusions and outlines future work.

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

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

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 value 1 to 3, $v_{(i,j)} = (v_{i,j} + v_{j,i})/2$ and $v_{i,j}$ denotes the partial derivative of the $i-th$ component of v with respect to the $j-th$ 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.

If Ω is subdivided into N_s regions $\{\Omega_s\}_{s=1}^{s=N_s}$, (Fig. 1), solving the above elastostatic problem is equivalent to finding the displacement functions $\{u_s\}_{s=1}^{s=N_s}$ which are stationary points of the energy functionals:

$$J_s(v_s) = \frac{1}{2}a(v_s, v_s)_{\Omega_s} - (v_s, f)_{\Omega_s} - (v_s, h)_{\Gamma_s}$$

where

$$\begin{aligned} a(v_s, w_s)_{\Omega_s} &= \int_{\Omega_s} v_{s(i,j)} c_{ijkl} w_{s(k,l)}\, d\Omega \\ (v_s, f)_{\Omega_s} &= \int_{\Omega_s} v_{si} f_i\, d\Omega \\ (v_s, h)_{\Gamma_s} &= \int_{\Gamma_{h_s}} v_{si} h_i\, d\Gamma \\ s &= 1, 2, ..., N_s \end{aligned} \tag{2}$$

and which satisfy on the interface boundary Γ_I the continuity constraints:

$$u_s = u_q \qquad \text{on } \Gamma_I = \bigcup_{s=1,q=1}^{s=N_s,q=N_s} \{\Omega_s \bigcap \Omega_q\} \tag{3}$$

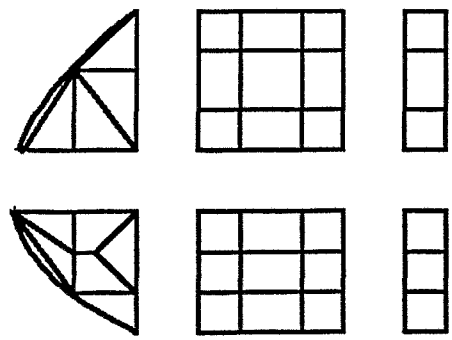

FIG. 1 *Tearing in N_s subdomains*

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

$$J^*(v_1, v_2, ..., v_{N_s}, \mu_1, \mu_2, ..., \mu_t) = \sum_{s=1}^{s=N_s} J_s(v_s) + \sum_{s=1,q=1}^{s=N_s,q=N_s} (v_s - v_q, \mu_{sq})_{\Gamma_I} \tag{4}$$

where

$$(v_s - v_q, \mu_{sq})_{\Gamma_I} = \int_{\Gamma_I} \mu_{sq}(v_s - v_q)\, d\Gamma$$

— that is, finding the displacement fields $\{u_s\}_{s=1}^{s=N_s}$ and the Lagrange multipliers λ_{sq} which satisfy:

$$\begin{aligned} J^*(u_1, u_2, ..., u_{N_s}, \mu_1, \mu_2, ..., \mu_t) &\leq J^*(u_1, u_2, ..., u_{N_s}, \lambda_1, \lambda_2, ..., \lambda_t) \\ &\leq J^*(v_1, v_2, ..., v_{N_s}, \lambda_1, \lambda_2, ..., \lambda_t) \end{aligned} \tag{5}$$

for any admissible $\{v_s\}_{s=1}^{s=N_s}$ and $\{\mu_k\}_{k=1}^{k=t}$. Among all admissible sets $\{v_s\}_{s=1}^{s=N_s}$ which satisfy the continuity conditions (3), the set $\{u_s\}_{s=1}^{s=N_s}$ minimizes the sum of the energy functionals J_s defined respectively on Ω_s. Therefore, $\{u_s\}_{s=1}^{s=N_s}$ are the restriction of the solution u of the non-partitioned problem (1) to $\{\Omega_s\}_{s=1}^{s=N_s}$. Indeed, Eqs. (4) and (5) correspond to a hybrid variational principle where the inter-subdomain continuity constraints (3) are removed by the introduction of Lagrange multipliers. The utility of Lagrange multipliers specifically for domain decomposition has also been previously recognized by other investigators (see, for example, Dihn, Glowinski and Periaux [3]).

3. Finite element adaptive approximation and parallel computing.

3.1. Galerkin solution and discrete formulation. The displacement fields $\{u_s\}_{s=1}^{s=N_s}$ are expressed by suitable shape functions as:

$$u_s = \mathbf{N}\mathbf{u}_s \quad s = 1, 2, ..., N_s \tag{6}$$

If the discrete subdomains are conforming and the continuity equations (3) are enforced for the *discrete* problem at each degree of freedom on Γ_I, a standard Galerkin procedure transforms the hybrid variational principle (4-5) in the following algebraic system:

$$\begin{aligned} \mathbf{K}_s\mathbf{u}_s &= \mathbf{f}_s - \mathbf{B}_s^T\boldsymbol{\lambda} \\ \sum_{s=1}^{s=N_s} \mathbf{B}_s\mathbf{u}_s &= 0 \\ s &= 1, 2, ..., N_s \end{aligned} \tag{7}$$

where $\mathbf{K}_s$, $\mathbf{u}_s$, and $\mathbf{f}_s$ are respectively the stiffness matrix, the displacement vector, and the prescribed force vector associated with the finite element discretization of Ω_s. $\mathbf{B}_s$ is a signed connectivity boolean matrix which describes which degrees of freedom in Ω_s lye on Γ_I. In particular when operating on a matrix or vector quantity, $\mathbf{B}_s$ does not involve any floating point operation; it simply extracts and signs the interface components of that matrix or vector quantity. The vector of Lagrange multipliers $\boldsymbol{\lambda}$ represents the interaction forces between the subdomains $\{\Omega_s\}_{s=1}^{s=N_s}$ along their interface Γ_I (Fig. 2). If every subdomain has enough boundary conditions to eliminate its rigid body motions, the above equations of equilibrium can be transformed into:

$$\begin{aligned} \mathbf{F}_I\boldsymbol{\lambda} &= \sum_{s=1}^{s=N_s} \mathbf{B}_s\mathbf{K}_s^{-1}\mathbf{f}_s \\ \mathbf{u}_s &= \mathbf{K}_s^{-1}(\mathbf{f}_s - \mathbf{B}_s^T\boldsymbol{\lambda}) \qquad s = 1, 2, ..., N_s \end{aligned} \tag{8}$$

where

$$\mathbf{F}_I = \sum_{s=1}^{s=N_s} \mathbf{B}_s\mathbf{K}_s^{-1}\mathbf{B}_s^T$$

For large problems and fine mesh decompositions, it is not feasible to explicitly assemble $\mathbf{F}_I$ whose size, n_I, is given by the total number of degrees of freedom on Γ_I. This implies that a direct method cannot be used to solve the above interface problem. The only efficient method of solving (8) in the general sparse case is that of conjugate gradients (CG), because once $\{\mathbf{K}_s\}_{s=1}^{s=N_s}$ have been factored, matrix-vector products of the form $\mathbf{F}_I\mathbf{v}$, where $\mathbf{v}$ denotes a generic n_I long vector, can be performed very efficiently using only forward and backward substitutions.

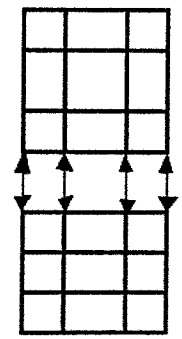

FIG. 2 *Interconnecting with discrete Lagrange multipliers*

3.2. Polynomial approximation. Alternatively, the continuity conditions on the interface (3) can be weakened in order to reduce the size of the interface problem and/or handle the case of non conforming subdomains. When accuracy can be preserved, this improves the computational performance of the subdomain-by-subdomain method. For example, polynomial expressions may be considered for approximating μ_{sq} in Eqs. (4) (see Dorr [4], for a theoretical justification). For topologically one-dimensional interfaces, this is equivalent to "gluing" the subdomain interfaces with polynomial approximations of their interaction forces (Fig. 3):

$$\lambda^1(\xi) = \sum_{k=0}^{k=r} \lambda_k^1 \xi^k; \quad \lambda^2(\xi) = \sum_{k=0}^{k=r} \lambda_k^2 \xi^k; \quad \cdots; \quad \lambda^d(\xi) = \sum_{k=0}^{k=r} \lambda_k^d \xi^k \tag{9}$$

where ξ denotes a curvilinear abscissae on Γ_I, d denotes the number of degrees of freedom at a node, r denotes the polynomial degree, and $r \times d$ is much smaller than the number of interface degrees of freedom, n_I. The superscript of λ denotes the directional freedom (x, y, or z displacement/rotation) of the corresponding generalized traction component.

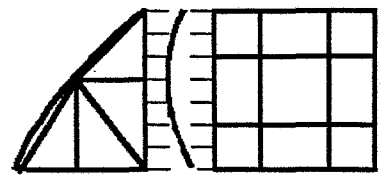

FIG. 3 *Interconnecting with polynomial Lagrange multipliers*

The constraint matrices $\{\mathbf{B}_s^r\}_{s=1}^{s=N_s}$ are in this case non-boolean finite element matrices which are assembled from their element level correspondents $\{\mathbf{B}_s^{r(e)}\}_{s=1}^{s=N_s}$ in the usual manner:

$$\mathbf{B}_s^r = \sum_e \mathbf{B}_s^{r(e)} \tag{10}$$

where e spans only the set of elements that are connected to the interface boundary Γ_I. For a finite element e with q nodes lying on Γ_I, $\mathbf{B}_s^{r(e)}$ is given by:

$$\mathbf{B}_s^{r(e)} = \begin{bmatrix} {}^1\mathbf{B}_s^{r(e)} \\ {}^2\mathbf{B}_s^{r(e)} \\ \cdot \\ \cdot \\ \cdot \\ {}^q\mathbf{B}_s^{r(e)} \end{bmatrix} \tag{11}$$

where ${}^l\mathbf{B}_s^{r(e)}$, $l = 1, 2, ..., q$, is the submatrix associated with the $l-th$ node of element e and has the following form:

$${}^l\mathbf{B}_s^{r(e)} = [{}_0^l\mathcal{B}_s^{r(e)} \quad {}_1^l\mathcal{B}_s^{r(e)} \quad {}_2^l\mathcal{B}_s^{r(e)} \quad \cdots \quad {}_s^l\mathcal{B}_s^{r(e)}] \tag{12}$$

Each of the submatrices ${}^{l}_{k}\mathcal{B}^{r(e)}_{s}$, is a $d \times d$ diagonal matrix that represents the contribution of the $k-th$ monomial ξ^k and is expressed as:

$$ {}^{l}_{k}\mathcal{B}^{r(e)}_{s} = (\int_{\Gamma_I{}^{(e)}} \mathbf{N}_l \xi^k \delta\Gamma)\, \mathbf{I}_d \tag{13} $$

where $\mathbf{N}_l$ is the shape function associated with the $l-th$ node of element e and $\mathbf{I}_d$ is the $d \times d$ identity matrix. An elegant and practical result is that increasing the degree of the polynomial approximation of λ^k involves only adding a few columns to the existing element level matrices ${}^{l}\mathbf{B}^{r(e)}_{s}$, as it is suggested by Eqs. (12-13).

As for the case of discrete interface tractions, the decomposed problem can be re-arranged as:

$$ \begin{aligned} \mathbf{F}^r_I \boldsymbol{\lambda} &= \sum_{s=1}^{s=N_s} \mathbf{B}^r_s \mathbf{K}^{-1}_s \mathbf{f}_s \\ \mathbf{u}_s &= \mathbf{K}^{-1}_s (\mathbf{f}_s - \mathbf{B}^{rT}_s \boldsymbol{\lambda}) \qquad s = 1, 2, ..., N_s \end{aligned} \tag{14} $$

where

$$ \mathbf{F}^r_I = \sum_{s=1}^{s=N_s} \mathbf{B}^r_s \mathbf{K}^{-1}_s \mathbf{B}^{rT}_s $$

$\mathbf{F}^r_I$ is the interface operator resulting from the approximation of the interface tractions with $r-th$ order polynomials. With this approach, the size of $\mathbf{F}^r_I$ is only $(r+1)d << n_I$. It is shown in Farhat and Geradin [7] that for sample structural problems and one-way mesh decompositions, very accurate results are possible with $r \times d = 10\% \times n_I$. For such problems, $\mathbf{F}^r_I$ is easily assembled and factored and the interface problem is best solved with a direct method. However for more challenging structural problems (i.e. with high stress gradients in the neighborhood of the interface), very high order polynomials may be needed to approximate the interface tractions which results in a larger interface problem that must be treated with an iterative solver. Unfortunately for large values of r, $\mathbf{F}^r_I$ becomes highly ill-conditioned and causes the overall performance of the solution algorithm to degrade. An alternative approach for reducing the size of the interface problem when a rather large number of Lagrange multipliers, N_λ, is needed to enforce the intersubdomain displacement continuity is presented below.

3.3 Piece-wise polynomial approximation. Let Γ^k_I, $k = 0, ..., \frac{N_\lambda}{d} - 2$ denote a partition $\mathcal{P}$ of the topologically one-dimensional interface boundary Γ_I defined as:

$$ \Gamma^k_I = [\xi_k,\ \xi_{k+1}] \qquad k = 0, ..., \frac{N_\lambda}{d} - 2 \tag{15} $$

where ξ_k, $k = 0, ..., \frac{N_\lambda}{d} - 1$, are the curvilinear abscissae of $\frac{N_\lambda}{d}$ specified points on Γ_I where the discrete surface tractions λ^j_k are introduced. Within each subinterval Γ^k_I, d cubic polynomial expressions are defined for the Lagrange multiplier approximations as:

$$\begin{aligned}
\tilde{\lambda}_k^1(\xi) &= c_{1\,k}^1 + c_{2\,k}^1(\xi-\xi_k) + c_{3\,k}^1(\xi-\xi_k)^2 + c_{4\,k}^1(\xi-\xi_k)^3 \\
\tilde{\lambda}_k^2(\xi) &= c_{1\,k}^2 + c_{2\,k}^2(\xi-\xi_k) + c_{3\,k}^2(\xi-\xi_k)^2 + c_{4\,k}^2(\xi-\xi_k)^3 \\
&\cdot \\
&\cdot \\
&\cdot \\
\tilde{\lambda}_k^d(\xi) &= c_{1\,k}^d + c_{2\,k}^d(\xi-\xi_k) + c_{3\,k}^d(\xi-\xi_k)^2 + c_{4\,k}^d(\xi-\xi_k)^3
\end{aligned} \tag{16}$$

where $c_{i\,k}^j$, $i = 1, ..., 4$, $j = 1, ..., d$, are determined by imposing:

$$\begin{aligned}
\tilde{\lambda}_k^j(\xi_k) &= \lambda_k^j \quad ; \quad \tilde{\lambda}_k^j(\xi_{k+1}) = \lambda_{k+1}^j \\
\frac{d\tilde{\lambda}_k^j}{d\xi}(\xi_k) &= \frac{d\lambda^j}{d\xi}(\xi_k) \; ; \; \frac{d\tilde{\lambda}_k^j}{d\xi}(\xi_{k+1}) = \frac{d\lambda^j}{d\xi}(\xi_{k+1}) \\
k &= 0, ..., \frac{N_\lambda}{d} - 2 \\
j &= 1, ..., d
\end{aligned} \tag{17}$$

The first set of equations (17) imply that $\tilde{\lambda}_k^j(\xi_{k+1}) = \tilde{\lambda}_{k+1}^j(\xi_{k+1})$, so that all λ^j are guaranteed to be continuously approximated on Γ_I. The second set of equations (17) involve the derivatives of the Lagrange multiplier functions which are neither available nor part of the weak form of the static equations of equilibrium. These derivatives are approximated by:

$$\frac{d\tilde{\lambda}_k^j}{d\xi}(\xi_k) = \frac{\frac{\Delta\xi_{k-1}}{\Delta\xi_k}(\lambda_{k+1} - \lambda_k) + \frac{\Delta\xi_k}{\Delta\xi_{k-1}}(\lambda_k - \lambda_{k-1})}{\Delta_2\xi_k} \tag{18}$$

where $\Delta\xi_k$ and $\Delta_2\xi_k$ are defined as:

$$\begin{aligned}
\Delta\xi_k &= \xi_{k+1} - \xi_k \\
\Delta_2\xi_k &= \xi_{k+1} - \xi_{k-1}
\end{aligned} \tag{19}$$

Note that (19) requires the two additional points ξ_{-1} and $\xi_{\frac{N_\lambda}{d}}$ which are chosen as:

$$\begin{aligned}
\xi_{-1} &= \xi_2 \\
\xi_{N_\lambda/d} &= \xi_{\frac{N_\lambda}{d}-3}
\end{aligned} \tag{20}$$

Substituting (16) and (18) into (17) determines the constants $c_{i\,k}^j$ as functions of the discrete Lagrange multipliers:

$$\begin{aligned}
c_{1\,k}^j &= \lambda_k^j \\
c_{2\,k}^j &= \alpha_{2k}\lambda_{k+1}^j + \beta_{2k}\lambda_k^j + \gamma_{2k}\lambda_{k-1}^j \\
c_{3\,k}^j &= \alpha_{3k}\lambda_{k+2}^j + \beta_{3k}\lambda_{k+1}^j + \gamma_{3k}\lambda_k^j + \delta_{3k}\lambda_{k-1}^j \\
c_{4\,k}^j &= \alpha_{4k}\lambda_{k+2}^j + \beta_{4k}\lambda_{k+1}^j + \gamma_{4k}\lambda_k^j + \delta_{4k}\lambda_{k-1}^j
\end{aligned} \tag{21}$$

where α_{2k}, α_{3k}, α_{4k}, β_{2k}, β_{3k}, β_{4k}, γ_{2k}, γ_{3k}, γ_{4k}, δ_{3k}, and δ_{4k} are constants that depend only the curvilinear abscissae ξ_{k-1}, ξ_k, ξ_{k+1} and ξ_{k+2}.

As previously, equations (21) are substituted into equations (16) and (16) into (4) to obtain:

$$\begin{aligned} \mathbf{K}_s\mathbf{u}_s &= \mathbf{f}_s - \mathbf{B}_s^{\mathcal{P}T}\boldsymbol{\lambda}_{\mathcal{P}} \\ \sum_{s=1}^{s=N_s} \mathbf{B}_s^{\mathcal{P}}\mathbf{u}_s &= 0 \end{aligned} \tag{22}$$

where $\mathbf{B}_s^{\mathcal{P}}$ is a sparse finite element matrix. The subscript/superscript $\mathcal{P}$ emphasizes the dependence of these quantities on the partition $\mathcal{P}$ of the interface boundary Γ_I. This matrix is assembled from its element level correspondents ${}_{k^e}\mathbf{B}_s^{\mathcal{P}(e)}$ in the usual manner:

$$\mathbf{B}_s^{\mathcal{P}} = \sum_e {}_{k^e}\mathbf{B}_s^{\mathcal{P}(e)} \quad s = 1, 2, ..., N_s \tag{23}$$

where e spans only the set of elements that are connected to Γ_I. The left subscript k^e emphasizes the dependence of ${}_{k^e}\mathbf{B}_s^{\mathcal{P}(e)}$ on the subinterval $\Gamma_I^k = [\xi_k, \xi_{k+1}]$ where one edge of element e falls. For a finite element e with q nodes lying on Γ_I, the $qd \times N_\lambda$ element level matrices ${}_{k^e}\mathbf{B}_s^{\mathcal{P}(e)}$, $s = 1, 2, ..., N_s$, are given by:

$${}_{k^e}\mathbf{B}_s^{\mathcal{P}(e)} = \begin{bmatrix} {}^1_{k^e}\mathbf{B}_s^{\mathcal{P}(e)} \\ {}^2_{k^e}\mathbf{B}_s^{\mathcal{P}(e)} \\ \cdot \\ \cdot \\ \cdot \\ {}^q_{k^e}\mathbf{B}_s^{\mathcal{P}(e)} \end{bmatrix} \tag{24}$$

where ${}^l_{k^e}\mathbf{B}_s^{\mathcal{P}(e)}$, $l = 1, 2, ..., q$, is a $d \times N_\lambda$ matrix associated with the $l-th$ node of element e and has the following form:

$${}^l_{k^e}\mathbf{B}_s^{\mathcal{P}(e)} = \left[\mathcal{O}_L^{k^e} \quad {}^l_{k^e-1}\mathcal{B}_s^{\mathcal{P}(e)} \quad {}^l_{k^e}\mathcal{B}_s^{\mathcal{P}(e)} \quad {}^l_{k^e+1}\mathcal{B}_s^{\mathcal{P}(e)} \quad {}^l_{k^e+2}\mathcal{B}_s^{\mathcal{P}(e)} \quad \mathcal{O}_R^{k^e}\right] \tag{25}$$

where $\mathcal{O}_L^{k^e}$ and $\mathcal{O}_R^{k^e}$ are respectively left and right $d\times(k^e-1)d$ and $d\times(N_\lambda-(k^e+3)d)$ zero matrices, and ${}^l_{k^e}\mathcal{B}_s^{\mathcal{P}(e)}$ is expressed as:

$${}^l_{k^e}\mathcal{B}_s^{\mathcal{P}(e)} = \eta_{k^e}\mathbf{I}_d \tag{26}$$

where $\mathbf{I}_d$ is the $d \times d$ identity matrix and η_{k^e-1}, η_{k^e}, η_{k^e+1} and η_{k^e+2} are function only of the partition $\mathcal{P}$ of Γ_I and are given by the following integration:

$$\int_{\Gamma_I{}^{(e)}} \tilde{\lambda}^j_{k^e}\mathbf{N}_l\delta\Gamma = \eta_{k^e-1}\lambda^j_{k^e-1} + \eta_{k^e}\lambda^j_{k^e} + \eta_{k^e+1}\lambda^j_{k^e+1} + \eta_{k^e+2}\lambda^j_{k^e+2} \tag{27}$$

It should be noted that while the symbolic derivation of equations (21-27) appears to be somehow complicated, their computer implementation is straightforward and their processing is inexpensive.

Approximating λ with polynomials does not require the location of the corresponding physical surface tractions to be specified. On the other hand, using piecewise low order polynomials such as the splines presented above requires the explicit definition of a partitioning $\mathcal{P}$ of Γ_I. This means that the locations of the physical surface tractions along Γ_I have to be specified. Therefore from a practical viewpoint, the polynomial approach seems more attractive, even though it is very limited. However, specifying where a surface traction is to be introduced can be transformed into an algorithmic advantage if one treats this issue as an additional degree of freedom in the problem. For example, if the stress field along Γ_I could be first predicted or estimated, the partition $\mathcal{P}$ would be dense in the areas of stress oscillation or high stress concentration, and coarse elsewhere. An iterative refinement procedure for positioning the ξ_ks of the subintervals $\Gamma_I^k = [\xi_k,\ \xi_{k+1}]$ is outlined below.

3.4. Iterative refinement procedure. It is assumed that a reasonable initial value $N_\lambda^{(0)}$ is given. The following convergence criterion is selected:

$$\begin{aligned} ||\mathbf{u}^{1(m+1)} - \mathbf{u}^{1(m)}||_\infty &< \epsilon||\mathbf{u}^{1(m)}||_\infty \\ ||\mathbf{u}^{2(m+1)} - \mathbf{u}^{2(m)}||_\infty &< \epsilon||\mathbf{u}^{2(m)}||_\infty \\ &\cdots \\ ||\mathbf{u}^{d(m+1)} - \mathbf{u}^{d(m)}||_\infty &< \epsilon||\mathbf{u}^{d(m)}||_\infty \end{aligned} \tag{28}$$

where the superscripts d and m refer respectively to the $d-th$ component of the solution at each node and to the $m-th$ iteration, and ϵ is a specified tolerance. As indicated by equations (28), the convergence of each of the d components of the displacement field is independently monitored. This is in order to avoid that potential important relative errors in a component of the solution whose magnitude is relatively small — for example, the x displacement of a cantilever beam with a load parallel to the y direction, are masked by an otherwise perfect convergence for a component of the solution whose magnitude is relatively large — for example, the y displacement. Such masked errors would ruin the accuracy of the stress field (which is the derivative of the displacement field).

Let $\Gamma_I^{k(m)}$, $k = 0, ..., \frac{N_\lambda^{(m)}}{d} - 2$ denote the partition $\mathcal{P}^{(m)}$ of the interface boundary Γ_I at iteration m:

$$\Gamma_I^{k(m)} = [\xi_k^{(m)},\ \xi_{k+1}^{(m)}] \qquad k = 0, ..., \frac{N_\lambda^{(m)}}{d} - 2 \tag{29}$$

If at iteration $m+1$ an additional discrete Lagrange multiplier is introduced, say in the subinterval $\Gamma_I^{k^*(m)}$, the resulting partition $\mathcal{P}^{(m+1)}$ becomes:

$$\Gamma_I^{k(m+1)} = [\xi_k^{(m+1)}, \xi_{k+1}^{(m+1)}] \qquad k = 0, ..., \frac{N_\lambda^{(m)}}{d} - 1 \tag{30}$$

where

$$\begin{aligned} \xi_k^{(m+1)} &= \xi_k^{(m)} \quad k \leq k^* \\ \xi_k^{(m+1)} &= \xi_{k-1}^{(m)} \quad k > k^* + 1 \end{aligned} \tag{31}$$

It can be easily shown that the regeneration at iteration $m+1$ of the constraint matrices $\mathbf{B}_s^{\mathcal{P}}$, $s = 1, 2, ..., N_s$ involves basically recomputing the coefficients c_{ik}^j, $i = 1, ..., 4$, $j = 1, ..., d$, of the polynomial expressions (16), only for those interface elements which intersect Γ_I inside $\Gamma_I^{k^*-1(m+1)}$, or $\Gamma_I^{k^*(m+1)}$, or $\Gamma_I^{k^*+1(m+1)}$, or $\Gamma_I^{k^*+2(m+1)}$. However, a refinement procedure for this case should also specify the location where an additional discrete Lagrange multiplier is to be introduced during the following iteration, — that is to define $\xi_{k^*}^{(m+1)}$. In the present work, this location is determined by the point of Γ_I where $||\mathbf{u}^{(m+1)} - \mathbf{u}^{(m)}||_\infty / ||\mathbf{u}^{(m)}||_\infty$ and/or the violation of static equilibrium prior to the averaging and improvement procedures (see below) are the largest. The introduction at iteration $m+1$ of more than one discrete Lagrange multiplier is handled exactly in the same manner. Paragraphs 3.3 and 3.4 are summarized in figure 4.

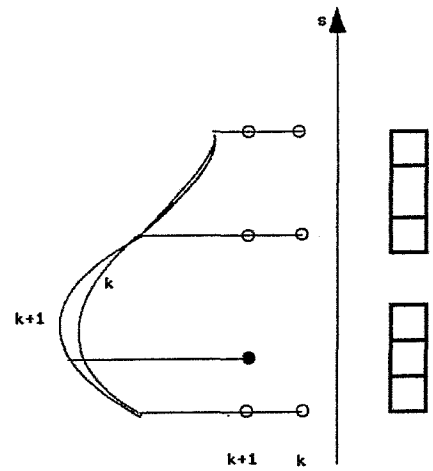

FIG. 4 *Interconnecting with adaptive splines*

3.5. Intersubdomain smoothing. Clearly, the finite element approximations of the subdomain displacement fields, $\mathbf{u}_s$, as computed by the solution of Eqs. (22), are in general discontinuous along the interface boundary Γ_I. If the discrete subdomains

are conforming, it is usually desirable to uniquely predict the displacement field at the common interface nodes. In this case, an additional smoothing procedure is required along Γ_I. For example, the interface boundary displacement field can be post-processed in the following manner:

$$\mathbf{u}^* = \mathbf{u}|_{\Gamma_I} = [\sum_{s=1}^{s=N_s} \overline{\mathbf{B}}_s \overline{\mathbf{B}}_s^T]^{-1} \sum_{s=1}^{s=N_s} \overline{\mathbf{B}}_s \mathbf{u}_s \tag{32}$$

where $\overline{\mathbf{B}}_s$ is the unsigned counterpart of $\mathbf{B}_s$. Equation (32) above is simply an averaging procedure. The quantity $\sum_{s=1}^{s=N_s} \overline{\mathbf{B}}_s \overline{\mathbf{B}}_s^T$ is a diagonal matrix which reports for each interface degree of freedom the number of subdomains that are attached to it. Therefore, it does not need neither to be explicitly computed, nor to be inverted.

Physical arguments can be used to postulate that the above averaged interface solution $\mathbf{u}^*$ is more accurate than each of the traces of the subdomain solutions, $\mathbf{u}_s|_{\Gamma_I}$. Therefore, the enhancing effect of the averaging procedure (32) should be back-propagated to the interiors of subdomains Ω_s by solving the following N_s independent displacement-driven subdomain problems:

$$\mathbf{u}_s = \mathbf{K}_{ss}{}^{-1}(\mathbf{f}_s - \mathbf{K}_{sI}\mathbf{u}^*) \qquad s = 1, 2, ..., N_s \tag{33}$$

where the additional subscripts s and I identify respectively interior and interface degrees of freedom. The above improvement of the subdomain solutions $\mathbf{u}_s$ requires only one sparse matrix-vector multiply and one pair of sparse forward/backward substitutions per subdomain. The triangular factors of $\mathbf{K}_{ss}$ are embedded in those of $\mathbf{K}_s$ which are readily available.

3.6. Parallel computational features. Clearly, the methodology is intrinsically parallel as it involves mostly subdomain-by-subdomain independent computations. Specific guidelines for carrying out the practical mesh decomposition are outlined in Farhat and Roux [8]. It is worthwhile mentioning that in this formulation, interprocessor communication is exclusively induced by the weak form of the continuity constraint:

$$(v_s - v_q, \mu_{sq})_{\Gamma_{I_{sq}}} = \int_{\Gamma_{I_{sq}}} \mu_{sq}(v_s - v_q)\, d\Gamma$$

Therefore, if $\Gamma_{I_{sq}}$ has a zero measure, then $(v_s - v_q, \mu_{sq})_{\Gamma_{I_{sq}}}$ is identically zero and no exchange of information is needed between subdomains Ω_s and Ω_q. Hence, 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 Schur-based domain decomposition methods where any subdomain interconnection induces interprocessor communication.

The computational method presented here differs also from Schur-based methods in the formulation and numerical properties of the interface problem. Briefly,

the eigenvalue spectrum of the Schur complement operator has an accumulation point towards infinity. From a physical viewpoint, this corresponds to the fact that the high frequencies of a stiffness-based operator are mainly mesh frequencies. On the other hand, $\mathbf{F}_I$ involves flexibility-like matrices; its eigenvalue spectrum has an accumulation point towards zero, which is clearly advantageous for a CG algorithm. Indeed, it can be shown (see Farhat and Roux [8], and Roux [12]) that under some assumptions, $\mathbf{F}_I$ and the Schur complement are dual operators. This particular topic is discussed in details in the companion paper by Roux [12] where the numerical advantages of the proposed method are demonstrated.

REMARK: Throughout the remainder of this paper, the notation used for the case of discrete Lagrange multipliers is adopted. However, unless otherwise indicated, the approaches, formulations and results are valid also for the cases of polynomial and piece-wise polynomial approximations of the Lagrange multipliers.

4. Singularity issues. Here we focus on two singularity issues that may arise in either a subdomain local problem, or in the interface global problem.

4.1 Local singularities. It is most likely that the mesh partitioning process will result in one or several subdomains that do not have enough prescribed displacement boundary conditions to eliminate local rigid body modes. For each of these floating subdomains, the stiffness matrix is singular and special care is required for the solution of its local equilibrium problem. For example, if Ω_f is a floating subdomain, $\mathbf{u}_f$ is given by:

$$\mathbf{u}_f = \mathbf{K}_f^+(\mathbf{f}_f - \mathbf{B}_f^T\boldsymbol{\lambda}) + \mathbf{R}_f\boldsymbol{\alpha}_f \tag{34}$$

where $\mathbf{K}_f^+$ is a generalized inverse of $\mathbf{K}_f$ that is not computed explicitly (see Farhat and Roux [8] for computing $\mathbf{K}_f^+$ and $\mathbf{R}_f$). $\mathbf{R}_f$ stores the rigid body modes of Ω_f (null space of $\mathbf{K}_f$) and $\boldsymbol{\alpha}_f$ specifies a linear combination of these. The fact that the rigid modes do not produce any internal energy can be expressed as:

$$\mathbf{R}_f^T\mathbf{K}_f\mathbf{u}_f = \mathbf{R}_f^T(\mathbf{f}_f - \mathbf{B}_f^T\boldsymbol{\lambda}) = 0 \tag{35}$$

Substituting (34) into (8) for each of the N_f floating subdomains leads after some algebraic manipulations to the following *indefinite* interface problem:

$$\begin{bmatrix} \mathbf{F}_I & \mathbf{G}_I \\ \mathbf{G}_I{}^T & \mathbf{O} \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} \mathbf{d} \\ -\mathbf{e} \end{bmatrix}$$

where

$$\begin{aligned} \mathbf{F}_I &= \sum_{s=1}^{s=N_s} \mathbf{B}_s \mathbf{K}_s^* \mathbf{B}_s^T \\ \mathbf{G}_I &= [\mathbf{R}_1^I \quad \dots \quad \mathbf{R}_{N_f}^I] \\ \mathbf{R}_s^I &= \mathbf{B}_s \mathbf{R}_s \quad s = 1, \dots, N_f \\ \mathbf{d} &= \sum_{s=1}^{s=N_s} \mathbf{B}_s \mathbf{K}_s^* \mathbf{f}_s \\ \mathbf{e}_s &= \mathbf{R}_s^T \mathbf{f}_s \quad s = 1, \dots, N_f \\ \boldsymbol{\alpha} &= [\alpha_1 \quad \dots \quad \alpha_{N_f}]^T \\ \mathbf{K}_s^* &= \mathbf{K}_s^{-1} \quad \text{if } \Omega_s \text{ is not a floating subdomain} \\ \mathbf{K}_s^* &= \mathbf{K}_s^+ \quad \text{if } \Omega_s \text{ is a floating subdomain} \end{aligned} \tag{36}$$

If the global domain Ω is not a floating domain, two floating subdomains Ω_i and Ω_j will never interconnect with the global interface Γ_I at exactly the same set of interface nodes. Hence, their corresponding rigid body modes $\mathbf{R}_i$ and $\mathbf{R}_j$ are always linearly independent. Therefore, for any mesh partitioning, $\mathbf{G}_I$ has full column rank. However, $\mathbf{F}_I$ may be singular as discussed below.

4.2. Global singularity. Consider the triangular domain shown in figure 5 below. It is torn in three subdomains Ω_1, Ω_2 and Ω_3. For the purpose of the following analysis, the subdomain interfaces — excluding their intersection point d — are labeled a, b and c.

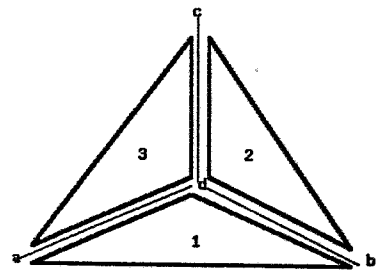

FIG. 5 *Three-subdomain partitioning of a triangular region*

Using the unsigned connectivity matrices $\overline{\mathbf{B}}_s$, the continuity constraints along $\Gamma_I = \{a, b, c, d\}$ can be written as:

$$\begin{aligned} \overline{\mathbf{B}}_1\mathbf{u}_1 &= \overline{\mathbf{B}}_2\mathbf{u}_2 \\ \overline{\mathbf{B}}_2\mathbf{u}_2 &= \overline{\mathbf{B}}_3\mathbf{u}_3 \\ \overline{\mathbf{B}}_3\mathbf{u}_3 &= \overline{\mathbf{B}}_1\mathbf{u}_1 \end{aligned} \tag{37}$$

More specifically, if $\overline{\mathbf{B}}_s^l$ denotes the unsigned connectivity matrix which glues Ω_s along the interface component l, the above equations can be expanded as:

$$\begin{aligned} \overline{\mathbf{B}}_1^b\mathbf{u}_1 &= \overline{\mathbf{B}}_2^b\mathbf{u}_2 \\ \overline{\mathbf{B}}_1^d\mathbf{u}_1 &= \overline{\mathbf{B}}_2^d\mathbf{u}_2 \\ \overline{\mathbf{B}}_2^c\mathbf{u}_2 &= \overline{\mathbf{B}}_3^c\mathbf{u}_3 \\ \overline{\mathbf{B}}_2^d\mathbf{u}_2 &= \overline{\mathbf{B}}_3^d\mathbf{u}_3 \\ \overline{\mathbf{B}}_3^a\mathbf{u}_3 &= \overline{\mathbf{B}}_1^a\mathbf{u}_1 \\ \overline{\mathbf{B}}_3^d\mathbf{u}_3 &= \overline{\mathbf{B}}_1^d\mathbf{u}_1 \end{aligned} \tag{38}$$

In particular, at the intersection point d, these equations are:

$$\begin{aligned} \overline{\mathbf{B}}_1^d\mathbf{u}_1 &= \overline{\mathbf{B}}_2^d\mathbf{u}_2 \\ \overline{\mathbf{B}}_2^d\mathbf{u}_2 &= \overline{\mathbf{B}}_3^d\mathbf{u}_3 \\ \overline{\mathbf{B}}_3^d\mathbf{u}_3 &= \overline{\mathbf{B}}_1^d\mathbf{u}_1 \end{aligned} \tag{39}$$

Clearly, the third of equations (39) is implicitly implied by the first two equations. Therefore the continuity equations (37) are redundant and the interface operator $\mathbf{F}_I = \sum_{s=1}^{s=N_s} \mathbf{B}_s\mathbf{K}_s^*\mathbf{B}_s^T$ is singular at the point d. This demonstrates that wherever more than two subdomains which effectively communicate (see Paragraph 3.6) intersect at a node, a redundant interface equation is generated for each degree of freedom at that node. Therefore for arbitrary mesh partitions with more than two subdomains, the interface operator $\mathbf{F}_I$ is in general symmetric positive semi-definite, and the Lagrangian matrix:

$$\mathbf{L} = \begin{bmatrix} \mathbf{F}_I & \mathbf{G}_I \\ \mathbf{G}_I{}^T & \mathbf{O} \end{bmatrix} \tag{40}$$

is singular. Of course, one can simply omit the last of equations (39) every time a crosspoint-like interface node is encountered. This is equivalent to introducing a cut between two — and only two — arbitrary subdomains at the crosspoint interface node. While doing so removes the singularity of the interface system, it un-necessarily complicates the implementation of the method. Indeed, even though $\mathbf{L}$ is singular, it is consistent with its right hand side (see (35)), and therefore the system of equations (36) admits more than one solution $(\boldsymbol{\lambda}, \boldsymbol{\alpha})$. However, $\boldsymbol{\lambda}$ and $\boldsymbol{\alpha}$

are not interesting by themselves. Any combination of these variables which satisfies equilibrium is acceptable. Moreover, even if $\mathbf{F}_I$ is singular, as long as it is positive any CG algorithm is guaranteed to find a solution $(\boldsymbol{\lambda}, \boldsymbol{\alpha})$ that satisfies the interface equations (36). This solution uniquely determines the subdomain displacement fields and their corresponding stress and strain fields.

5. Numerical solution of the interface problem. The Lagrangian matrix $\mathbf{L}$ is indefinite, and therefore a straightforward conjugate gradient algorithm cannot be directly applied to solve the system of equations (36). However since $\mathbf{F}_I$ is positive, the conjugate *projected* gradient (CPG) iteration (see, for example, Gill and Murray [9]) can be used to obtain the sought-after solution. Basically, at each iteration of the (CG) algorithm, the search direction is projected onto the null space of $\mathbf{G}_I^T$ so that for each floating subdomain Ω_f, the constraint $(\mathbf{B}_f\mathbf{R}_f)^T\boldsymbol{\lambda} = \mathbf{R}_f^T\mathbf{f}_f$ is enforced. Using the standard notations for a CG algorithm, the resulting CPG algorithm can be written as:

Iterate k = 1,2, ... until convergence

$$
\begin{aligned}
\zeta^{(k)} &= \mathbf{r}^{(k-1)^T}\mathbf{r}^{(k-1)}/\mathbf{r}^{(k-2)^T}\mathbf{r}^{(k-2)} \quad (\zeta^{(1)} = 0) \\
\mathbf{p}^{(k)} &= \mathbf{r}^{(k-1)} + \zeta^{(k)}\mathbf{p}^{(k-1)} \quad (\mathbf{p}^{(1)} = \mathbf{r}^{(0)}) \\
\bar{\mathbf{p}}^{(k)} &= [\mathbf{I} - \mathbf{G}_I(\mathbf{G}_I{}^T\mathbf{G}_I)^{-1}\mathbf{G}_I{}^T]\mathbf{p}^{(k)} \\
\nu^{(k)} &= \mathbf{r}^{(k-1)^T}\mathbf{r}^{(k-1)}/\bar{\mathbf{p}}^{(k)^T}\mathbf{F}_I\bar{\mathbf{p}}^{(k)} \\
\boldsymbol{\lambda}^{(k)} &= \boldsymbol{\lambda}^{(k-1)} + \nu^{(k)}\bar{\mathbf{p}}^{(k)} \\
\mathbf{r}^{(k)} &= \mathbf{r}^{(k-1)} - \nu^{(k)}\mathbf{F}_I\bar{\mathbf{p}}^{(k)}
\end{aligned}
\tag{41}
$$

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. However, the entire Lagrangian matrix, $\mathbf{L}$, does not need to be preconditioned, even in the presence of floating subdomains; only $\mathbf{F}_I$ needs to be treated. In matrix form, $\mathbf{F}_I$ can be written as:

$$
\mathbf{F}_I = [\mathbf{B}_1 \quad \mathbf{B}_2 \quad \ldots \quad \mathbf{B}_{N_s}]
\begin{bmatrix}
\mathbf{K}_1^* & \mathbf{O} & \mathbf{O} & \mathbf{O} \\
\mathbf{O} & \mathbf{K}_2^* & \mathbf{O} & \mathbf{O} \\
\mathbf{O} & \mathbf{O} & \ldots & \mathbf{O} \\
\mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{K}_{N_s}^*
\end{bmatrix}
\begin{bmatrix}
\mathbf{B}_1^T \\ \mathbf{B}_2^T \\ \ldots \\ \mathbf{B}_{N_s}^T
\end{bmatrix}
\tag{42}
$$

which suggests as an approximate inverse:

$$
\begin{aligned}
\widetilde{\mathbf{F}_I^{-1}} &= [\mathbf{B}_1 \quad \mathbf{B}_2 \quad \ldots \quad \mathbf{B}_{N_s}]
\begin{bmatrix}
\mathbf{K}_1 & \mathbf{O} & \mathbf{O} & \mathbf{O} \\
\mathbf{O} & \mathbf{K}_2 & \mathbf{O} & \mathbf{O} \\
\mathbf{O} & \mathbf{O} & \ldots & \mathbf{O} \\
\mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{K}_{N_s}
\end{bmatrix}
\begin{bmatrix}
\mathbf{B}_1^T \\ \mathbf{B}_2^T \\ \ldots \\ \mathbf{B}_{N_s}^T
\end{bmatrix} \\
&= \sum_{s=1}^{s=N_s} \mathbf{B}_s\mathbf{K}_s\mathbf{B}_s^T
\end{aligned}
\tag{43}
$$

The efficiency of $\widetilde{\mathbf{F}_I}$ as a preconditioner is reported in Farhat and Roux [8]; its numerical properties are discussed in the companion paper by Roux [12]. Here it is briefly noted that the preconditioner $\widetilde{\mathbf{F}_I}$ leads to auxiliary systems that are easy to solve — actually these involve only matrix-vector products — and perfectly parallelizable on both local and shared memory parallel architectures.

If two floating subdomains Ω_i and Ω_j are not geometrically interconnected, the products $\mathbf{R}_i^{I^T}\mathbf{R}_j^I$ and $\mathbf{R}_j^{I^T}\mathbf{R}_i^I$ are identically zero. Therefore, $\mathbf{G}_I^T\mathbf{G}_I$ is a sparse symmetric matrix whose structure is determined exclusively by the subdomain interconnectivity. Since there are at most 3 rigid body modes per floating subdomain in two-dimensional problems and 6 in three-dimensional ones, the size of $\mathbf{G}_I^T\mathbf{G}_I$ is at most $2N_f \times 2N_f$ for two-dimensional problems and $6N_f \times 6N_f$ for three-dimensional ones. Unfortunately, $\mathbf{G}_I^T\mathbf{G}_I$ is the only quantity in the presented computational method that cannot be evaluated using only subdomain-by-subdomain computations. However, as long as the number of floating subdomains is kept relatively small, say less than 128, the method is still very efficient on both serial and parallel processors. Next, a preprocessing technique is examined in order to improve the scalability of this domain decomposition method in the presence of a very large number of floating subdomains.

6. Preprocessing the rigid body modes with a QR factorization. At each iteration of the PCPG algorithm, the projector:

$$\mathbf{P} = \mathbf{I} - \mathbf{G}_I(\mathbf{G}_I{}^T\mathbf{G}_I)^{-1}\mathbf{G}_I{}^T \tag{44}$$

filters out the rigid body mode components of the solution. The matrix product $\mathbf{G}_I{}^T\mathbf{G}_I$ is sparse symmetric positive definite and its Choleski triangular factors:

$$\mathbf{G}_I{}^T\mathbf{G}_I \;=\; \mathbf{C}\mathbf{C}^T \tag{45}$$

are computed once, before any iteration begins. Subsequently, each matrix-vector product $\tilde{\mathbf{p}}^{(k)} = \mathbf{P}\mathbf{p}^{(k)}$ is evaluated as following:

$$\begin{aligned}
\mathbf{x}_s^{(k)} &= \mathbf{R}_s^{I^T}\mathbf{p}_s^{(k)} \quad s = 1, ..., N_f \\
\textit{Lower Solve} \quad \mathbf{C}[\mathbf{y}_1^{(k)} ... \mathbf{y}_{N_f}^{(k)}]^T &= [\mathbf{x}_1^{(k)} ... \mathbf{x}_{N_f}^{(k)}]^T \\
\textit{Upper Solve} \quad \mathbf{C}^T[\mathbf{z}_1^{(k)} ... \mathbf{z}_{N_f}^{(k)}]^T &= [\mathbf{y}_1^{(k)} ... \mathbf{y}_{N_f}^{(k)}]^T \\
\tilde{\mathbf{p}}^{(k)} &= \mathbf{p}^{(k)} - \sum_{s=1}^{s=N_f} \mathbf{R}_s^I \mathbf{z}_s^{(k)}
\end{aligned} \tag{46}$$

where $\mathbf{p}_s^{(k)}$ is the localization of $\mathbf{p}^{(k)}$ to Ω_s. The first and last steps of the above procedure require only subdomain-by-subdomain sparse computations. The lower and upper triangular solve induce long-range communication.

Here, it is assumed that the problems of interest and the mesh partitions are such that the size of any subdomain interface is much smaller than the size of the

subdomain interior. Implementations on highly parallel processors are considered as long as they do not result in an element-by-element formulation of the proposed computational method. Therefore, the additional cost incurred by the Choleski factorization of $\mathbf{G}_I{}^T\mathbf{G}_I$ is negligible when compared to the cost of factoring all the subdomain stiffnesses $\mathbf{K}_s$. This factorization is parallelizable, and since it is performed only once, its parallel efficiency is not an issue. However, the lower and triangular solve in (46) are performed at every iteration and are known to be very inefficient on both distributed memory parallel processors and shared memory parallel/vector supercomputers. (see, for example, Farhat and Wilson [5], Storaasli, Nguyen and Agarwal [13]). In practice, for a number of subdomains smaller than 128, the triangular systems may be solved in a serial fashion without affecting the overall speedup. However, for finer mesh partitions, the serial solution of these auxiliary systems may ruin the sought-after speedup.

Alternatively, one can note that any full rank matrix $\overline{\mathbf{G}}_I$ that spans the same proper subspace as $\mathbf{G}_I$ can be equally used to construct a suitable projector $\overline{\mathbf{P}}$. In particular, if $\overline{\mathbf{G}}_I$ is chosen such that:

$$\begin{aligned} span(\overline{\mathbf{G}}_I) &= span(\mathbf{G}_I) \\ \overline{\mathbf{G}}_I^T\overline{\mathbf{G}}_I &= 0 \end{aligned} \tag{47}$$

then $\overline{\mathbf{P}}$ simplifies to:

$$\overline{\mathbf{P}} = \mathbf{I} - \overline{\mathbf{G}}_I\overline{\mathbf{G}}_I^T \tag{48}$$

which presents two attractive computational features:

1. $\overline{\mathbf{P}}$ does not involve any matrix factorization.
2. each product $\bar{\mathbf{p}}^{(k)} = \overline{\mathbf{P}}\mathbf{p}^{(k)}$ can be carried out using only subdomain-by-subdomain computations:

$$\bar{\mathbf{p}}^{(k)} = \overline{\mathbf{P}}\mathbf{p}^{(k)} = \sum_{s=1}^{s=N_I} \overline{\mathbf{R}}_s^I\overline{\mathbf{R}_s^I}^T\mathbf{p}^{(k)} \tag{49}$$

(Note that $\mathbf{p}^{(k)}$ is used in the above summation and not $\mathbf{p}_s^{(k)}$.)

A $\overline{\mathbf{G}}_I$ matrix that satisfies relations (47) is obtained by orthonormalizing all the columns of $\mathbf{G}_I$ — that is, all the traces of the rigid body modes on Γ_I. This can be achieved, for example, via a QR factorization, which has been shown to be feasible on massively parallel processors (Kratzer [11]):

$$\mathbf{G}_I = \overline{\mathbf{G}}_I\mathcal{R} \tag{50}$$

The cost of this QR factorization is of the same order as the combined costs of forming and factoring $\mathbf{G}_I^T\mathbf{G}_I$. Unfortunately, while $\mathbf{G}_I$ is sparse, $\overline{\mathbf{G}}_I$ is in general full. However, only one matrix $\overline{\mathbf{R}}_s^I$ which contains of at most six augmented rigid body modes (six columns) needs to be stored within one subdomain (processor) Ω_s. More importantly, the unscalable upper and lower parallel solve are avoided.

7. Numerical examples. In this section, the saddle point domain decomposition (SPDD) method is illustrated with the analysis of a submarine structure subjected to a pressure shock wave. The submarine skin is modeled with thin shell elements (thickness = 0.05 × radius of curvature) which produce ill-conditioned finite element stiffness matrices (fig. 6).

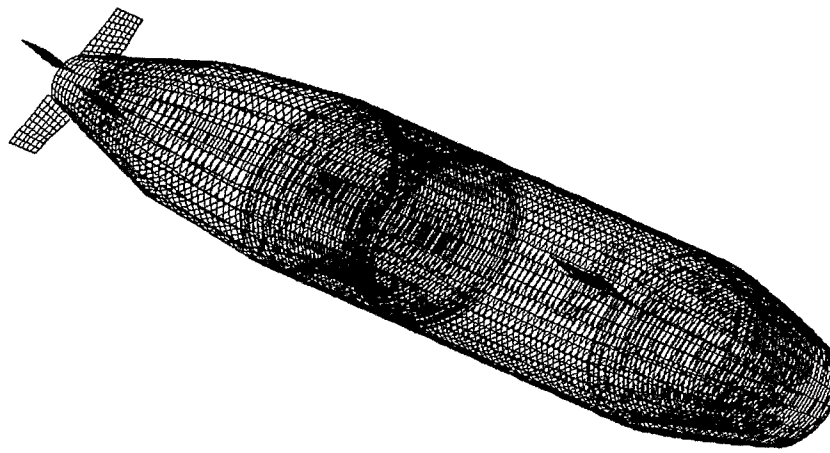

FIG. 6 *Finite element discretization of a submarine structure*

All computations are performed on an iPSC/2 with 4 Mbytes memory boards and a maximum of 32 processors. Different meshes with different resolutions and different element aspect ratios are designed for each subcube configuration. The mesh decomposer described in [6] is used to create the mesh partitions. Table I below summarizes the characteristics of the resulting problems ("size" refers to the number of equations).

TABLE I *Problem definition*

Problem	Problem size	Partitioning	Interface size	Element aspect ratio
P1	4275	4 sub.	225	1
P2	16875	16 sub.	1125	0.25
P3	33675	32 sub.	2325	0.125

Since direct solvers are not in general very efficient on the iPSC, the measured performance results are compared with those of a Jacobi preconditioned conjugate

gradient algorithm (JPCG) for the same problems and the same multiprocessor. This is not to say that JPCG is a particularly performing algorithm for structural problems; however, it is often used in the computational mechanics community as a reference point for iterative methods (see, for example, Hughes, Ferencz, and Hallquist [10]). Each of the two algorithms uses optimal data structures and the following stopping criterion:

$$||\mathbf{r}^{(k)}||_2 \leq 10^{-6} \times ||\mathbf{r}^{(0)}||_2 \tag{51}$$

where $\mathbf{r}^{(k)}$ is the *primary* residual vector at step k. For all cases, only one subdomain is assigned to one processor. In order to overcome the practical difficulties associated with running a large problem on a single node of the iPSC-2, the parallel speed-up SP, defined here as the ratio between the total time using one processor and the total time using N_p processors, is computed in the following manner:

$$SP = \frac{N_p}{1 + N_p \times \frac{T_{cmn}}{T_{cmp}}} \tag{52}$$

where N_p denotes the number of processors, and T_{cmn} and T_{cmp} denote respectively the total communication time and total computation time associated with the algorithm.

First, the SPDD method is applied with discrete Lagrange multipliers. The measured performance results are reported in Table II below.

TABLE II *Performance results on the iPSC-2*

SPDD (discrete Lagrange multipliers)

Problem	N_p	JPCG iterations	JPCG CPU	JPCG SP	SPDD iterations	SPDD CPU	SPDD SP
P1	4	601	499 secs	3.95	30	160 secs	3.96
P2	16	1965	1645 secs	15.50	96	373 secs	15.80
P3	32	3974	3340 secs	28.80	219	759 secs	29.00

Clearly, very high speedup are achieved and the SPDD method is shown to be about four times as fast as the JPCG algorithm. The number of iterations seems to grow linearly with the size of the interface. However, the reader should recall that the average element aspect ratio decreases from 1.0 in mesh P1 to 0.125 in mesh P2, which injects an additional source of ill-conditioning when refining the problem.

Next, the SPDD method is applied with piece-wise polynomial approximations of the Lagrange multipliers (Table III).

TABLE III *Performance results on the iPSC-2*

SPDD (piece-wise polynomial Lagrange multipliers)

Problem	N_p	Problem size	Interface size	N_λ	CPU
P1	4	4275	225	48	116
P2	16	16875	1125	380	280
P3	32	33675	2325	862	492

The above examples seem to indicate that a number of Lagrange multipliers that is roughly equal to 35 % of the total number of interface degrees of freedom is sufficient to assemble the subdomain solutions. The improvement in CPU consumption is demonstrated.

8. Closure. A domain decomposition method based on a hybrid variational principle is presented for the parallel finite element solution of self-adjoint elliptic partial differential equations. First, the spatial domain is partitioned into a set of totally disconnected subdomains and a local finite element solution is computed in each of these subdomains. Next, a set of discrete, polynomial or piece-wise polynomial Lagrange multipliers are introduced in order to enforce compatibility constraints between the subdomain finite element approximations. The interface problem associated with this domain decomposition method is in general indefinite and of variable size. Its numerical properties are discussed in a companion paper by Roux [12]. A dedicated preconditioned conjugate gradient algorithm is developed for solving the latter problem when it is not feasible to explicitly assemble the interface operator. At each iteration, the zero energy modes are filtered out via a suitable projector which somehow limits the scalability of the methodology. To alleviate this problem, the rigid body modes are preprocessed with a QR factorization which improves the scalability of the overall algorithm. Numerical and performance results on an iPSC-2 hypercube are reported. They demonstrate the potential of the methodology for real-life solid mechanics problems. Future work will focus on the generalization of the piece-wise polynomial approximation of the Lagrange multipliers to topologically arbitrary two-dimensional interfaces (for three-dimensional problems). Also, penalty formulations will be investigated to transform the interface problem into an unconstrained minimization problem and therefore to further improve its scalability.

Acknowledgments

The author would like to acknowledge partial support by the National Science Foundation under Grant ASC-8717773, and partial support by NASA Langley under Grant NAG-1536427.

References

1. BJORDSTAD, P. E. and WIDLUND, O. B., *Iterative methods for solving elliptic problems on regions partitioned into substructures*, SIAM J. of Numerical Analysis, 23, No. 6 (1986).
2. DESTUYNDER, P. and ROUX, F. X., *A parallel solver for the linear elasticity equations on a composite beam*, in Domain Decomposition Methods, T. F. Chan, R. Glowinski, J. Periaux and O. Widlund, eds., SIAM, 1989, pp. 314-320.
3. DIHN, Q. V., GLOWINSKI, R. and PERIAUX, J., *Solving elliptic problems by domain decomposition methods with applications*, in Elliptic Problem Solvers II, A. Schoenstadt, ed., Academic Press, 1984.
4. DORR, M., *On the discretization of interdomain coupling in elliptic boundary-value problems*, in Domain Decomposition Methods, T. F. Chan, R. Glowinski, J. Periaux and O. Widlund, eds., SIAM, 1989, pp. 17-37.
5. FARHAT, C. and WILSON, E., *A parallel active column equation solver*, Computers & Structures, 28, No. 4, (1988), pp. 289-304.
6. FARHAT, C., *A simple and efficient automatic FEM domain decomposer*, Computers & Structures, 28, No. 5, (1988), pp. 579-602.
7. FARHAT, C. and GERADIN, M., *Using a reduced number of Lagrange multipliers for assembling parallel incomplete field finite element approximations*, CU-CSSC-90-23, University of Colorado at Boulder (1990).
8. FARHAT, C. and ROUX, F. X., *A method of finite element tearing and interconnecting and its parallel solution algorithm*, Int. J. Num. Meth. Eng., 32 (1991).
9. GILL, P. E. and MURRAY, W., *Numerical methods for constrained optimization*, P. E. Gill and W. Murray, eds., Academic Press, London, 1974, pp. 132-135.
10. HUGHES, T. J. R., FERENCZ, R. M. and HALLQUIST, J. O., *Large-scale vectorized implicit calculations in solid mechanics on a Cray X-MP/48 utilizing EBE preconditioned conjugate gradients*, Comp. Meth. Appl. Mech. Eng., 61, No. 2, (1987), pp. 215-248.
11. KRATZER, S. G., *Unstructured sparse QR factorization on SIMD computers*, in Unstructured Scientific Computation on Scalable Multiprocessors, P. Mehrotra, J. Saltz and R. Voigt, eds., The MIT Press 1990.
12. ROUX, F. X., *Dual and spectral properties of Schur and saddle point domain decomposition methods*, Proc. SIAM Conference on Domain Decomposition Methods for Partial Differential Equations, (1991)
13. STORAASLI, O. O., NGUYEN, D. T. and AGARWAL, T. K., *Parallel-vector solution of large-scale structural analysis problems on supercomputers*, AIAA / ASME / ASCE / AHS / ASC Structures, Structural Dynamics and Materials Conference, Mobile, Alabama, 30, (1989), pp. 859-867.