The FETI Level 1 Method: Theory and Implementation

Chandrika Kamath
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
P.O. Box 808, L-561
Livermore, CA 94551
kamath2@llnl.gov

April 7, 2000

Abstract

This report summarizes our experiences in developing a prototype serial code for the implementation of the Level 1 Finite Element Tearing and Interconnecting (FETI) method. This method is a non-overlapping domain-decomposition scheme for the parallel solution of ill-conditioned systems of linear equations arising in structural mechanics problems. The FETI method has been shown to be numerically scalable for second order elasticity and fourth order plate and shell problems. In this report, we first outline the theory underlying the FETI method and discuss the approaches taken to improve the robustness and convergence of the method. We next provide implementation details, focusing on our serial prototype code. Finally, we present experimental results, followed by a summary of our observations.

UCRLID-138075 -This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

Contents

1 INTRODUCTION

The solution of sparse linear systems of equations is an important and time consuming step in computational mechanics simulations. For example, when nonlinear materials and large-deformation phenomena are modeled using Nike [25], multiple linear solves are required at each time step. As a result, solving linear systems of equations almost always represents the single largest share - at least 50% - of the overall computational cost in three dimensional implicit simulations.

Techniques for solving linear systems can be broadly divided into two - direct and iterative. Each has its strengths and drawbacks. Direct methods are robust and reliable, with a predictable CPU time. However, they require a global data structure that grows rapidly with the problem size. Further, direct solvers are not scalable to massively parallel systems with thousands of processors, though recent results using the PSPASES software [20] suggest that they scale well on moderately large systems up to 256 processors [17]. Iterative methods, on the other hand, usually scale well with increasing numbers of processors, but are not yet as reliable and robust as direct solvers. In particular, iterative solvers perform poorly on problems common in computational mechanics simulations, which include characteristics such as material softening and damage, material anisotropy, mesh refinement to capture local effects, combined bending and membrane response of shells, and the near singularities that result as buckling behavior is approached.

In this report, we consider the Finite Element Tearing and Interconnecting (FETI) method. This is one among a class of methods that are referred to as non-overlapping domain decomposition methods in the numerical analysis literature [36] and as conjugate-gradient-based substructuring methods in the structural analysis community [14]. In these methods, a large problem is broken into smaller and more tractable parts or sub-domains. For each sub-domain, we partition the unknowns into internal degrees of freedom(d.o.f.), denoted by subscript i, and interface boundary d.o.f., denotes by the subscript b. If the internal d.o.f. are numbered first, and the interface boundary d.o.f. are numbered last, then for a problem with N_s sub-domains, the static equations of equilibrium can be written in matrix form as:

$$\begin{bmatrix} K_{ii}^{(1)} & 0 & 0 & 0 & 0 & K_{ib}^{(1)} \\ 0 & . & 0 & 0 & 0 & . \\ 0 & 0 & K_{ii}^{(s)} & 0 & 0 & K_{ib}^{(s)} \\ 0 & 0 & 0 & . & 0 & . \\ 0 & 0 & 0 & 0 & K_{ii}^{(N_s)} & K_{ib}^{(N_s)} \\ K_{ib}^{(1)^T} & . & K_{ib}^{(s)^T} & . & K_{ib}^{(N_s)^T} & K_{bb} \end{bmatrix} \begin{bmatrix} u_i^{(1)} \\ . \\ u_i^{(s)} \\ . \\ u_i^{(N_s)} \\ u_b \end{bmatrix} = \begin{bmatrix} f_i^{(1)} \\ . \\ f_i^{(s)} \\ . \\ f_i^{(N_s)} \\ f_b \end{bmatrix} \tag{1.1}$$

where

$$K_{bb} = \sum_{s=1}^{N_s} K_{bb}^{(s)} \quad , \quad f_b = \sum_{s=1}^{N_s} f_b^{(s)} \tag{1.2}$$

and u_b is the vector of the interface d.o.f. The superscript on the stiffness matrices K, the force vectors f, and the displacements u, denotes the sub-domain number s. Applying static condensation to the internal d.o.f. of each substructure, Equation (1.1) can be written as the following interface problem:

$$\left(K_{bb} - \sum_{s=1}^{N_s} K_{ib}^{(s)^T} K_{ii}^{(s)^{-1}} K_{ib}^{(s)}\right) u_b = f_b - \sum_{s=1}^{N_s} K_{ib}^{(s)^T} K_{ii}^{(s)^{-1}} f_i^{(s)} \tag{1.3}$$

Using Equation (1.2), this reduces to

$$Su_b = f_b - \sum_{s=1}^{N_s} K_{ib}^{(s)^T} K_{ii}^{(s)^{-1}} f_i^{(s)} \tag{1.4}$$

where

$$S = \sum_{s=1}^{N_s} S^{(s)} \tag{1.5}$$

and

$$S^{(s)} = K_{bb}^{(s)} - K_{ib}^{(s)^T} K_{ii}^{(s)^{-1}} K_{ib}^{(s)} \tag{1.6}$$

Each substructure matrix, $S^{(s)}$, also known as the local Schur complement, corresponds to a local static condensation operator. The interface problem, Equation (1.4), is usually solved using the conjugate gradient (CG) method, as this avoids the explicit formation of the matrix S. This substructure-based CG method combines both direct and iterative solvers. It has better parallel scalability than CG applied to the global problem as the local computation on each processor is higher, though the communication is the same. Further, as the Schur complement matrix is better conditioned than the global matrix, the substructure-based CG method converges faster than the global method [16]. This method is also referred to as the "primal" substructuring method as the interface problem is formulated in terms of the displacement, or the primal variable.

The convergence of the substructure-based CG method can be improved through the use of preconditioning. A global preconditioner for S can be constructed by approximating an inverse of a sum by a sum of inverses:

$$\tilde{S}^{-1} = \sum_{s=1}^{N_s} S^{(s)^{-1}} \tag{1.7}$$

This preconditioner, referred to as the Neumann preconditioner, is amenable to parallel processing. However, it is not numerically scalable for mesh partitions that have either floating sub-structures or cross-points [16]. In these cases, the condition number of the interface problem grows as the square of the number of sub-domains, thus suggesting a mesh partition with a few sub-domains. From a mechanical point of view, this behavior can be explained by observing that the primal substructuring method restricts the direct flow of information to between neighboring substructures. In the absence of a mechanism for exchanging information among all substructures, the condition number of the problem grows with the number of substructures.

This problem of lack of numerical scalability has been addressed by both the FETI method [15] and the closely related balancing method [26] through the use of a coarse problem for exchanging information among all the sub-domains. These methods have demonstrated numerical scalability for both second order elasticity problems [13] and fourth order plate and shell problems [12], [7].

The FETI method, and its variants, have been studied extensively - see for example [14], [4] and the references therein. In this report, we focus on the simplest form of the FETI algorithm, referred to as the one-level FETI method, and discus a prototype implementation. In the next Section, we discuss the theory underlying the FETI method and the factors that determine the convergence of the method. In the following Section,

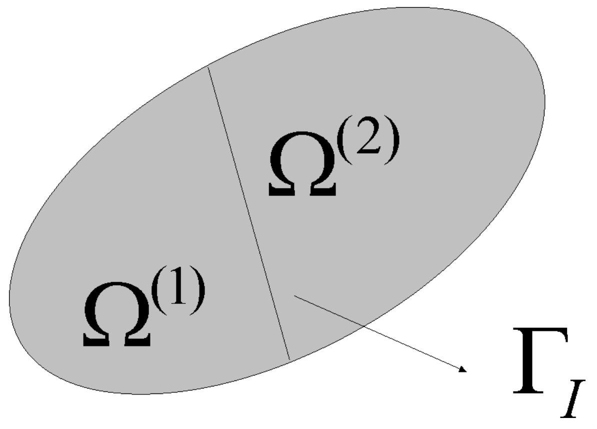

Figure 1: A domain divided into two sub-domains

we provide implementation details, focusing on the serial prototype code we have written in Fortran 77 to implement the method. We next discuss some simple experimental results, and conclude with a summary of our observations.

The main aim of report is to summarize, in one place, the ideas behind the FETI level 1 method and to describe the subtleties encountered in a practical implementation. We acknowledge that none of the ideas presented here are new; in fact, the method has been studied extensively by Charbel Farhat and his colleagues, as indicated in the Bibliography.

2 FETI LEVEL 1 METHOD - THEORY

The FETI method is a domain-decomposition-based iterative method with Lagrange multipliers. The one-level FETI method that is described in this report, can be considered as a two step preconditioned conjugate gradient (PCG) algorithm, where problems with Dirichlet boundary conditions are solved in the preconditioning step, and related problems with Neumann boundary conditions are solved in a second step. A small size auxiliary problem based on the sub-domain rigid body modes acts as a coarse problem, propagating the error globally during the PCG iteration, thus accelerating convergence.

2.1 Two Sub-domain Case

In the simplest case of two sub-domains, $\Omega^{(1)}$ and $\Omega^{(2)}$, Figure 1, the variational form of the three dimensional boundary value problem to be solved can be reduced to the following algebraic system [15]:

$$K^{(1)}u^{(1)} = f^{(1)} + {B^{(1)}}^T\lambda \tag{2.1}$$

$$K^{(2)}u^{(2)} = f^{(2)} + {B^{(2)}}^T\lambda \tag{2.2}$$

$$B^{(1)}u^{(1)} = B^{(2)}u^{(2)} \tag{2.3}$$

where $K^{(j)}$, $u^{(j)}$, and $f^{(j)}$, $j = 1, 2$ are the stiffness matrix, the displacement vector, and the prescribed force vector, respectively, associated with the finite element discretization of $\Omega^{(j)}$. The vector of Lagrange multipliers, λ, represents the interaction forces between the two sub-domains along their common boundary Γ_I. The number of Lagrange multipliers is equal to the number of interface nodal unknowns. Equation (2.3) ensures that the displacements on either side of the interface Γ_I are equal, and provides the additional equations that enable the solution of Equations (2.1) and (2.2).

The matrix $B^{(j)}$ is a signed Boolean matrix localizing a substructure quantity to the global sub-domain interface resulting from the domain decomposition. In the case of two sub-domains, this global sub-domain interface is the same as the sub-domain interface for each of the sub-domains. We denote the number of interior nodal unknowns in the two sub-domain case by $n_i^{(1)}$ and $n_i^{(2)}$ and the number of interface boundary nodal unknowns by $n_b^{(1)}$ and $n_b^{(2)}$, respectively. If the interior d.o.f. are numbered first, followed by the interface boundary d.o.f., the two Boolean matrices, $B^{(1)}$ and $B^{(2)}$, have the form:

$$B^{(j)} = \begin{bmatrix} O^{(j)} & I^{(j)} \end{bmatrix} \tag{2.4}$$

where $O^{(j)}$ is an $n_b^{(j)} \times n_i^{(j)}$ null matrix and $I^{(j)}$ is an $n_b^{(j)} \times n_b^{(j)}$ signed identity matrix.

In the simplest case, when both $K^{(1)}$ and $K^{(2)}$ are non-singular, we can rewrite Equations (2.1) and (2.2) as follows:

$$u^{(1)} = {K^{(1)}}^{-1} \left(f^{(1)} + {B^{(1)}}^T \lambda \right) \tag{2.5}$$

$$u^{(2)} = {K^{(2)}}^{-1} \left(f^{(2)} + {B^{(2)}}^T \lambda \right) \tag{2.6}$$

Substituting these in Equation (2.3), we obtain

$$\left(B^{(1)} {K^{(1)}}^{-1} {B^{(1)}}^T + B^{(2)} {K^{(2)}}^{-1} {B^{(2)}}^T \right) \lambda = B^{(1)} {K^{(1)}}^{-1} f^{(1)} + B^{(2)} {K^{(2)}}^{-1} f^{(2)} \tag{2.7}$$

which can be solved for λ. Once λ is obtained, the displacements $u^{(1)}$ and $u^{(2)}$ can be obtained by substituting for λ in Equations (2.1) and (2.2).

In most problems however, not all the stiffness matrices for the sub-domains will be non-singular, that is, the domain decomposition process will likely produe some "floating" sub-domains. In these cases, the solution process described above will break down, and a special computational strategy is required to handle the local singularities. We next describe this strategy for the simple case of two sub-domains, where one is a floating sub-domain, for example in the case of a cantilever beam, Figure 2, which has been vertically partitioned into two sub-domains.

In this case, $K^{(1)}$ is positive definite and $K^{(2)}$ is positive semi-definite as no essential boundary condition is specified over $\Omega^{(2)}$. Thus, the system in equation (2.2) is singular. If this system is consistent, a general solution can be written as

$$u^{(2)} = {K^{(2)}}^{+} \left(f^{(2)} + {B^{(2)}}^T \lambda \right) + R^{(2)} \alpha \tag{2.8}$$

where ${K^{(2)}}^{+}$ is a pseudo-inverse of $K^{(2)}$ [3], $R^{(2)}$ represents the rigid body modes of $\Omega^{(2)}$, and α indicates a linear combination of these modes. α is a vector of length $n_r^{(2)}$, where $\Omega^{(2)}$ has $n_r^{(2)}$ rigid body modes. For

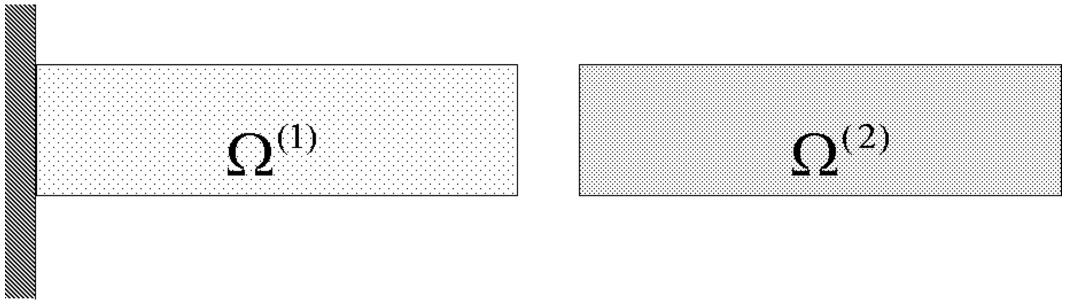

Figure 2: A cantilever beam with a floating sub-domain $\Omega^{(2)}$

well-posed three dimensional problems, the number of rigid body modes can be at most 6, and for well-posed two dimensional problems, it can be at most 3. $R^{(2)}$ is an $(n_i^{(2)} + n_b^{(2)}) \times n_r^{(2)}$ rectangular matrix whose columns form a basis of the null space of $K^{(2)}$.

The replacement of Equation (2.2) by Equation (2.8), introduces an additional set of unknowns α in the system, making the system of equations (2.1), (2.8), and (2.3) underdetermined. However, we observe that since $K^{(2)}$ is symmetric, the singular equation (2.2) admits at least one solution if and only if the right hand side, $(f^{(2)} + {B^{(2)}}^T\lambda)$ has no component in the null space of $K^{(2)}$. This can be written as

$$ {R^{(2)}}^T \left(f^{(2)} + {B^{(2)}}^T \lambda\right) = 0 \tag{2.9} $$

which gives us an additional equation, enabling the solution of λ, α, and the displacements $u^{(1)}$ and $u^{(2)}$.

Combining the equations (2.1), (2.3), (2.8), (2.9) we obtain the "interface system" as follows:

$$ \begin{bmatrix} F_I & -B^{(2)}R^{(2)} \\ -{R^{(2)}}^T {B^{(2)}}^T & 0 \end{bmatrix} \quad \begin{bmatrix} \lambda \\ \alpha \end{bmatrix} \quad \begin{bmatrix} B^{(2)}{K^{(2)}}^{+} f^{(2)} + B^{(1)}{K^{(1)}}^{-1} f^{(1)} \\ -{R^{(2)}}^T f^{(2)} \end{bmatrix} \tag{2.10} $$

where

$$ F_I = B^{(1)}{K^{(1)}}^{-1}{B^{(1)}}^T + B^{(2)}{K^{(2)}}^{+}{B^{(2)}}^T \tag{2.11} $$

This system can be solved for (λ, α), and the displacements $u^{(1)}$ and $u^{(2)}$ are obtained from Equations (2.5) and (2.8).

2.2 Multi-sub-domain Case

The two sub-domain case can be easily extended to the case when the global problem is divided into more than two sub-domains. Suppose the number of sub-domains in the problem is denoted by N_s, and the Boolean matrix that describes the interconnectivity of domain $\Omega^{(j)}$ with its neighbor $\Omega^{(k)}$ is denoted by $B_k^{(j)}$. In general, $B_k^{(j)}$ is an $n_I \times (n_i^{(j)} + n_b^{(j)})$ matrix, where n_I denotes the total number of interface d.o.f. for the problem, $n_i^{(j)}$ is the number of interior d.o.f. in domain j, and $n_b^{(j)}$ is the number of interface boundary d.o.f. in domain j. $B_k^{(j)}$ has the following structure:

$$ B_k^{(j)} = \begin{bmatrix} O_1(j,k) \\ C_k^{(j)} \\ O_2(j,k) \end{bmatrix} \tag{2.12} $$

where O_1 and O_2 are null matrices with $(n_i^{(j)} + n_b^{(j)})$ columns and $m_1(j,k)$ and $m_2(j,k)$ rows, respectively. $C_k^{(j)}$ is a connectivity matrix with $(n_i^{(j)} + n_b^{(j)})$ columns and $m_c(j,k)$ rows. The number of rows, $m_1(j,k)$, $m_2(j,k)$, and $m_c(j,k)$, are all non-negative integers that satisfy

$$m_1(j,k) + m_2(j,k) + m_c(j,k) = n_I \tag{2.13}$$

The connectivity matrix $C_k^{(j)}$ has the structure

$$C_k^{(j)} = \begin{bmatrix} O_3(j,k) & I_k^{(j)} & O_4(j,k) \end{bmatrix} \tag{2.14}$$

where $O_3(j,k)$, $I_k^{(j)}$, and $O_4(j,k)$ all have $m_c(j,k)$ rows and $m_3(j,k)$, $m_c(j,k)$, and $m_4(j,k)$ columns, respectively. The number of columns are non-negative integers that satisfy the constraint:

$$m_3(j,k) + m_4(j,k) + m_c(j,k) = n_i^{(j)} + n_b^{(j)} \tag{2.15}$$

$O_3(j,k)$ and $O_4(j,k)$ are null matrices, while $I_k^{(j)}$ is a signed identity matrix.

In the multi-sub-domain case, the finite element variational form generates the following algebraic system:

$$K^{(j)} u^{(j)} = f^{(j)} + \sum_{k=1}^{a^{(j)}} {B_k^{(j)}}^T \lambda \quad , \quad j = 1, \dots, N_s \tag{2.16}$$

$$B_k^{(j)} u^{(j)} = B_j^{(k)} u^{(k)}, \quad j = 1, \dots, N_s \text{ and } \Omega^{(j)} \text{ connected to } \Omega^{(k)} \tag{2.17}$$

where $a^{(j)}$ is the number of sub-domains connected to $\Omega^{(j)}$. When all the $K^{(j)}$ are non-singular, the solution process to calculate $u^{(j)}$ is a simple extension of the process outlined in Section 2.1. When there is one or more floating sub-domains, we can again extend the ideas developed in Section 2.1 to yield the following interface problem:

$$\begin{bmatrix} F_I & -G_I \\ -G_I^T & 0 \end{bmatrix} \begin{bmatrix} \lambda \\ \alpha \end{bmatrix} = \begin{bmatrix} d \\ -e \end{bmatrix} \tag{2.18}$$

where

$$F_I = \sum_{s=1}^{N_s} B^{(s)} {K^{(s)}}^+ {B^{(s)}}^T$$

$$G_I = \begin{bmatrix} B^{(1)} R^{(1)} & \cdots & B^{(N_f)} R^{(N_f)} \end{bmatrix}$$

$$\alpha = \begin{bmatrix} \alpha_1^T & \cdots & \alpha_{N_f}^T \end{bmatrix}^T$$

$$d = \sum_{s=1}^{N_s} B^{(s)} {K^{(s)}}^+ f^{(s)}$$

$$e^{(s)} = \begin{bmatrix} {f_{(s)}}^T R^{(s)} \end{bmatrix}^T \quad s = 1, \dots, N_f$$

$$K^{(s)^+} = \begin{cases} {K^{(s)}}^{-1} & \text{if } \Omega^{(s)} \text{ is not a floating sub-domain} \\ \text{a generalized inverse of } K^{(s)} & \text{if } \Omega^{(s)} \text{ is a floating sub-domain} \end{cases}$$

N_f is the number of floating sub-domains, and the signed Boolean matrix for each sub-domain s can be written as

$$B^{(s)} = \sum_{k=1}^{a^{(j)}} B_k^{(j)} \tag{2.19}$$

The system matrix in Equation (2.18):

$$\begin{bmatrix} F_I & -G_I \\ -G_I^T & 0 \end{bmatrix} \tag{2.20}$$

also referred to as the Lagrangian matrix, is indefinite, while the FETI interface operator, F_I, is symmetric and usually positive semi-definite.

In the case where a domain decomposed problem has no crosspoints, that is, nodes where more than two sub-domains meet, F_I is symmetric positive definite. However, for problems with crosspoints, the inclusion of all the continuity constraints (Equation (2.17)) at the cross point results in redundant equations [16], as shown in Figure 3 for a four sub-domain problem, where three constraints must be satisfied at the crosspoint for each sub-domain. This makes the matrix F_I singular at the crosspoint, though it is positive as each ${K^{(s)}}^+$ is positive. A simple way to handle this singularity is to omit the redundant continuity constraint at the crosspoint. However, this makes the implementation of the method more complicated. It is easier to include all the continuity constraints and observe that while the Lagrangian matrix is singular, the system (2.18) is consistent and therefore admits to at least one solution (λ, α). Any combination of λ and α that satisfies equilibrium is an acceptable solution; the actual values of λ and α are not interesting by themselves.

2.3 Solution of the Interface System

In order to solve Equation (2.18), we note that it is difficult to explicitly assemble the matrix F_I. This makes it infeasible to use a direct method to solve this system. However, an iterative method, such as Conjugate Gradient (CG), can be efficient as it requires only matrix vector products of the form $F_I u = v$, which can be accomplished using the factors of the sub-domain stiffness matrices. However, the indefiniteness of the Lagrangian precludes a straightforward application of CG to the solution of Equation (2.18).

To address this problem, we observe that the symmetry of F_I enables us to rewrite the FETI interface problem as the following constrained minimization problem:

$$min_\lambda \Phi(\lambda) = \frac{1}{2}\lambda^T F_I \lambda - \lambda^T (G_I \alpha + d) \tag{2.21}$$

$$\text{subject to } G_I^T \lambda = e \tag{2.22}$$

Since F_I is positive semi-definite, we can solve this problem using CG provided we ensure that the value of λ at each iteration satisfies the constraint. This can be achieved by first choosing the initial value of $\lambda = \lambda_0$,

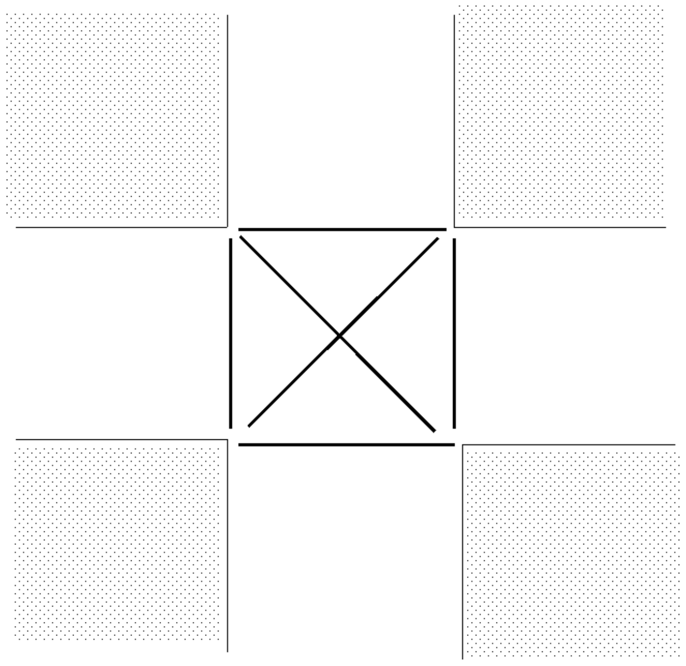

Figure 3: Redundant continuity constraints at a crosspoint

that satisfies the constraint, for example:

$$\lambda_0 = G_I^T (G_I^T G_I)^{-1} e \tag{2.23}$$

Then, we ensure that the constraint is satisfied at each iteration by removing from the improved solution λ_k, the component that does not satisfy the constraint. In other words, since λ_k is updated by a scaled version of the direction vector at each CG iteration, if we start with the value of λ_0 given in Equation (2.23), we can ensure that the constraint is satisfied at each iteration by projecting the direction vector onto the null space of G_I^T using the projector

$$P = \left(I - G_I (G_I^T G_I)^{-1} G_I^T \right) \tag{2.24}$$

The version of CG that incorporates this projection is referred to as conjugate projected gradient (CPG) method. The projector requires the formation of the matrix $G_I^T G_I$ of dimension at most $3N_f \times 3N_f$ for two-dimensional problems and $6N_f \times 6N_f$ for three dimensional problems, where N_f is the number of floating sub-domains.

The FETI method thus reduces to the solution of Equation (2.18), using the Conjugate Projected Gradient technique applied to the matrix F_I, with d as the right hand side, and λ as the unknown. As shown in Algorithm 1, the CPG method is very similar to the standard Conjugate Gradient technique with the added operation of projection to ensure that the constraint in Equation (2.22) is satisfied at each iteration.

In comparison to Equation (1.4), the FETI method can be considered to be a "dual" substructuring method as the interface problem is formulated in terms of the traction forces, which are the dual variables. Mechanically, the FETI method can be interpreted as follows: Each iteration starts with a projection step that evaluates

Algorithm 1: Conjugate Projected Gradient Method

```
Step 1.
    Set $\lambda_0 = G_I^T(G_I^T G_I)^{-1} e$
    Set $k = 0$
    Compute $r_0 = d - F_I \lambda_0$
    Compute $w_0 = [I - G_I(G_I^T G_I)^{-1} G_I^T] r_0$
    Compute $(w_0, w_0)$
    Set $p_0 = w_0$

Step 2. Compute
    $\gamma_k = (w_k, w_k)/(p_k, F_I p_k)$
    $x_{k+1} = x_k + \gamma_k p_k$
    $r_{k+1} = r_k - \gamma_k F_I p_k$
    $w_{k+1} = [I - G_I(G_I^T G_I)^{-1} G_I^T] r_{k+1}$
    $\beta_k = (w_{k+1}, w_{k+1})/(w_k, w_k)$
    $p_{k+1} = w_{k+1} + \beta_k p_k$

Step 3. If convergence criterion satisfied, terminate iterations. Else set $k = k + 1$, and go to Step 2.
```

the global position of the floating sub-domains by analyzing their rigid body modes. Next, a traction vector $p_{kb}^{(s)}$ is prescribed at the boundary d.o.f. Note that the matrix-vector multiplication $F_I p_k$, in the k-th iteration can be written as

$$F_I p_k = \sum_{s=1}^{N_s} B^{(s)} {K^{(s)}}^+ {B^{(s)}}^T p_k$$

which implies that in each substructure s, we need to evaluate

$$ {K^{(s)}}^+ {B^{(s)}}^T p_k \tag{2.25}$$

If the matrix $K^{(s)}$ is partitioned such that its interior d.o.f., denoted by subscript i, are numbered first, followed by its boundary d.o.f., denoted by the subscript b, then, we can write the matrix as

$$\begin{bmatrix} K_{ii}^{(s)} & K_{ib}^{(s)} \\ {K_{ib}^{(s)}}^T & K_{bb}^{(s)} \end{bmatrix} \tag{2.26}$$

Thus, evaluating the quantity in Equation (2.25) is equivalent to the solution of

$$\begin{bmatrix} K_{ii}^{(s)} & K_{ib}^{(s)} \\ {K_{ib}^{(s)}}^T & K_{bb}^{(s)} \end{bmatrix} \begin{bmatrix} \bar{p}_{ki}^{(s)} \\ \bar{p}_{kb}^{(s)} \end{bmatrix} = {B^{(s)}}^T p_k = \begin{bmatrix} 0 \\ p_{kb}^{(s)} \end{bmatrix} \tag{2.27}$$

where $p_{kb}^{(s)}$ is the restriction of p_k to the boundary d.o.f. of the s-th sub-domain. This gives us the displacements $(\bar{p}_{ki}^{(s)} \quad \bar{p}_{kb}^{(s)})$ for the sub-structure. Next, the sum of all the displacements at the sub-structure boundaries is computed via

$$\sum_{s=1}^{N_s} B^{(s)} \begin{bmatrix} \bar{p}_{ki}^{(s)} \\ \bar{p}_{kb}^{(s)} \end{bmatrix}$$

If this sum is zero, it implies that the displacement field is continuous across substructure interfaces, and convergence is achieved. If not, the traction vector is updated and the process repeated.

We also note that unlike the primal substructuring method, the FETI method propagates the error globally at each CPG iteration. The projection at each step can be thought of as a way of preventing a floating sub-domain from determining the contributions of its rigid body modes independently of the contributions of the rigid body modes of other floating sub-domains. The global propagation of the error is achieved through the solution of a smaller (coarser) problem represented by $(G_I^T G_I)$. For a fixed mesh, as the number of sub-domains in increased, the amount of global propagation of information increases, which suggests that the FETI method should be numerically scalable [16].

The convergence criterion for FETI is derived based on the observation that at each iteration, the λ_k and $u_k^{(s)}$ satisfy

$$u_k^{(s)} = K^{(s)^+}\left(f^{(s)} + B^{(s)^T}\lambda_k\right) + R^{(s)}\alpha^{(s)} \tag{2.28}$$

Using this fact, and Equations (2.18) and (2.24) we can write

$$\begin{aligned} \sum_{s=1}^{N_s} B^{(s)} u_k^{(s)} &= F_I \lambda_k + d + G_I \alpha \\ &= P(-F_I \lambda_k + d) \end{aligned}$$

In other words, the jump in the displacement field across the interface Γ_I is equal to the projected residual from the CPG algorithm. Thus, by monitoring the projected residual, we can determine when to stop the CPG iterations.

2.4 The Role of Reorthogonalization in FETI

It has been observed [16] that for some problems, the spectral distribution of the interface problem has a few large eigenvalues that are well separated from the smaller ones. This has important consequences in the convergence rate of the CPG algorithm. In the first few iterations, the algorithm captures mostly the eigenvectors corresponding to the largest eigenvalues of F_I. Since these correspond to the low physical modes of the structure, we quickly obtain a good approximation to the displacement field. After the high modes of F_I are captured, the effective condition number of the interface problem becomes much smaller, resulting in superconvergence behavior of the the CPG algorithm.

This distribution of eigenvalues, where a few large eigenvalues are well separated from the smaller ones, is known to cause a rapid loss of orthogonality between the direction vectors of the CPG algorithm [31], resulting in slower convergence. To offset the loss of orthogonality, we may need to reorthogonalize the direction vectors at each iteration of the CPG algorithm, as illustrated in Algorithm 2. This would require the storage of the direction vectors, as well as the product of F_I with the direction vectors. As the interface

Algorithm 2: Conjugate Projected Gradient Method with re-orthogonalization

```
Step 1.
        Set $\lambda_0 = G_I^T(G_I^T G_I)^{-1}e$
        Set $k = 0$
        Compute $r_0 = d - F_I\lambda_0$
        Compute $w_0 = [I - G_I(G_I^T G_I)^{-1}G_I^T]r_0$
        Compute $(w_0, w_0)$
        Set $p_0 = w_0$

Step 2. Compute
        $\gamma_k = (w_k, w_k)/(p_k, F_I p_k)$
        $x_{k+1} = x_k + \gamma_k p_k$
        $r_{k+1} = r_k - \gamma_k F_I p_k$
        $w_{k+1} = [I - G_I(G_I^T G_I)^{-1}G_I^T]r_{k+1}$
        $p_{k+1} = w_{k+1} - \sum_{i=0}^{k} \frac{w_{k+1}^T F_I p_i}{p_i^T F_I p_i} p_i$

Step 3. If convergence criterion satisfied, terminate iterations. Else set $k = k + 1$, and go to Step 2.
```

problem is expected to be smaller than the global problem, the extra memory needed for reorthogonalization may not be a significant percentage of the total memory required. However, if the amount of memory is limited, the authors in [16] suggest using only a few previous direction vectors for reorthogonalization. In their opinion, an optimal strategy will use the first direction vectors rather than the most recently computed ones, as the subspace generated by these vectors is closer to the subspace associated with the eigenvectors corresponding to the largest eigenvalues.

Note that this loss of orthogonality occurs in both the primal and the dual substructuring methods.

2.5 Preconditioning the Interface Problem

The convergence of the Conjugate Projected Gradient method can be improved through the use of preconditioning. The incorporation of preconditioning into the CPG method with reorthogonalization results in Algorithm 3, where we denote the preconditioner, which is an approximation to the inverse of F_I, by $\tilde{F}_I^{-1}$.

In this case, we monitor the preconditioned projected residual to determine when to stop the CPG iterations.

The choice of a good preconditioner for the FETI level one algorithm is determined by several factors. Since the CPG algorithm is applied to the interface problem, we seek a preconditioner which is a good approximation to the inverse of the interface matrix F_I. Since F_I is not explicitly assembled, we would like

Algorithm 3: Preconditioned Conjugate Projected Gradient Method with re-orthogonalization

```
Step 1.
    Set $\lambda_0 = G_I^T (G_I^T G_I)^{-1} e$
    Set $k = 0$
    Compute $r_0 = d - F_I \lambda_0$
    Compute $w_0 = [I - G_I (G_I^T G_I)^{-1} G_I^T] r_0$
    Compute $y_0 = [I - G_I (G_I^T G_I)^{-1} G_I^T] \tilde{F}_I^{-1} w_0$
    Compute $(y_0, w_0)$
    Set $p_0 = w_0$

Step 2. Compute
    $\gamma_k = (p_k, w_k)/(p_k, F_I p_k)$
    $x_{k+1} = x_k + \gamma_k p_k$
    $r_{k+1} = r_k - \gamma_k F_I p_k$
    $w_{k+1} = [I - G_I (G_I^T G_I)^{-1} G_I^T] r_{k+1}$
    $y_{k+1} = [I - G_I (G_I^T G_I)^{-1} G_I^T] \tilde{F}_I^{-1} w_{k+1}$
    $p_{k+1} = y_{k+1} - \sum_{i=0}^{k} \frac{y_{k+1}^T F_I p_i}{p_i^T F_I p_i} p_i$

Step 3. If convergence criterion satisfied, terminate iterations. Else set $k = k + 1$, and go to Step 2.
```

the preconditioner to be independent of the values of the elements of the matrix F_I. In addition, we would also like the preconditioner to be amenable to parallel computation. The following preconditioners satisfy both these requirements.

Lumped preconditioner: This preconditioner arises from the observation that since F_I is represented as the sum of matrices:

$$F_I = \sum_{s=1}^{N_s} B^{(s)} K^{(s)^+} {B^{(s)}}^T$$

$$= \begin{bmatrix} B^{(1)} & . & B^{(s)} & . & B^{(N_s)} \end{bmatrix} \begin{bmatrix} K^{(1)^+} & 0 & 0 & 0 & 0 \\ 0 & . & 0 & 0 & 0 \\ 0 & 0 & K^{(s)^+} & 0 & 0 \\ 0 & 0 & 0 & . & 0 \\ 0 & 0 & 0 & 0 & K^{(N_s)^+} \end{bmatrix} \begin{bmatrix} {B^{(1)}}^T \\ . \\ {B^{(s)}}^T \\ . \\ {B^{(N_s)}}^T \end{bmatrix}$$

a simple approximation to its inverse is given by

$$(\tilde{F}_I^L)^{-1} = \begin{bmatrix} B^{(1)} & . & B^{(s)} & . & B^{(N_s)} \end{bmatrix} \begin{bmatrix} K^{(1)} & 0 & 0 & 0 & 0 \\ 0 & . & 0 & 0 & 0 \\ 0 & 0 & K^{(s)} & 0 & 0 \\ 0 & 0 & 0 & . & 0 \\ 0 & 0 & 0 & 0 & K^{(N_s)} \end{bmatrix} \begin{bmatrix} {B^{(1)}}^T \\ . \\ {B^{(s)}}^T \\ . \\ {B^{(N_s)}}^T \end{bmatrix} \tag{2.29}$$

By exploiting the structure of $B^{(s)}$, we can use Equation (2.26) to simplify Equation (2.29) as follows:

$$(\tilde{F}_I^L)^{-1} = \sum_{s=1}^{N_s} B^{(s)} \begin{bmatrix} 0 & 0 \\ 0 & K_{bb}^{(s)} \end{bmatrix} \quad {B^{(s)}}^T$$

The lumped preconditioner is so named because, from a mechanical viewpoint, it corresponds to finding a set of "lumped" interface forces that can reproduce the displacement jumps at the substructure interfaces, when only the interface d.o.f. are allowed to displace. In [13], the authors have shown that the lumped preconditioner is not mathematically optimal. However, it is a rather economical preconditioner as it does not require any additional storage and needs only matrix-vector products of size equal to the sub-structure interfaces. It is also easily parallelizable on both shared and distributed memory systems.

Dirichlet Preconditioner: The Dirichlet preconditioner is based on the mechanical interpretation of FETI. Recall from Section 2.3 that in each iteration of the FETI method, we prescribe traction forces $p_{kb}^{(s)}$ at the interface boundaries of each substructure, and determine the corresponding jump in the displacements at the interface d.o.f. So, the inverse problem would consist of imposing the jump in the displacements at the substructure interfaces, and calculating the corresponding interface traction. This leads to a preconditioner of the form:

$$(\tilde{F}_I^D)^{-1} = \sum_{s=1}^{N_s} B^{(s)} \begin{bmatrix} 0 & 0 \\ 0 & K_{bb}^{(s)} - {K_{ib}^{(s)}}^T {K_{ii}^{(s)}}^{-1} K_{ib}^{(s)} \end{bmatrix} \quad {B^{(s)}}^T$$

where $(K_{bb}^{(s)} - {K_{ib}^{(s)}}^T {K_{ii}^{(s)}}^{-1} K_{ib}^{(s)})$ is the sub-domain Schur complement. The authors in [16] consider this preconditioner to be an optimal one as the condition number of the FETI method with Dirichlet preconditioner is bounded by $O(1 + log^2(H/h))$, where H/h is the ratio of substructure to element sizes. However, as we discuss in Section 3.3, the implementation of this preconditioner requires additional memory for the storage of the matrix elements that are needed in the computation of the sub-domain Schur complement. Further, due to the additional forward/back solves required in each application of the preconditioner, the Dirichlet preconditioner, though optimal, is an expensive one, and in some cases, may be less efficient than the Lumped preconditioner when the total time for convergence is taken into account.

2.5.1 Preconditioning in the presence of crosspoints

In [32], the authors show that in the presence of crosspoints, the Lumped and Dirichlet preconditioners, as defined in Section 2.5, are mechanically inconsistent. To solve this problem, they suggest a simple modification based on an averaging scheme. For a homogeneous problem, where all the sub-domains at a crosspoint have similar material, geometrical, and discretization properties, and therefore have similar stiffnesses, the two preconditioners can be modified as follows:

$$(\tilde{F}_I^L)^{-1} = \sum_{s=1}^{N_s} A^{(s)} B^{(s)} \begin{bmatrix} 0 & 0 \\ 0 & K_{bb}^{(s)} \end{bmatrix} \quad {B^{(s)}}^T A^{(s)} \tag{2.30}$$

$$(\tilde{F}_I^D)^{-1} = \sum_{s=1}^{N_s} A^{(s)} B^{(s)} \begin{bmatrix} 0 & 0 \\ 0 & K_{bb}^{(s)} - {K_{ib}^{(s)}}^T {K_{ii}^{(s)}}^{-1} K_{ib}^{(s)} \end{bmatrix} \quad {B^{(s)}}^T A^{(s)} \tag{2.31}$$

where the matrix $A^{(s)}$ is a diagonal matrix where each element corresponding to an interface node represents the multiplicity of the node. The multiplicity is defined as the number of sub-domains incident at an interface node. The matrix $A^{(s)}$ essentially acts as a scaling matrix, scaling each value of an interface node by its multiplicity, thus ensuring that the preconditioners are mechanically consistent. The computational cost incurred by the introduction of these scaling matrices is negligible.

2.5.2 Effects of heterogeneity on preconditioners

The scaling matrix, as defined in terms of the multiplicities of the interface d.o.f., results in a mechanically consistent preconditioner only when the sub-domains are homogeneous. However, a substructure-based finite element structural model may give rise to different sources of heterogeneity. For example, different sub-domains may have different material properties or mesh sizes. Such heterogeneities may lead to highly discontinuous stiffnesses at the interface d.o.f. For such problems, the authors in [32] show that the simple averaging scheme described in Section 2.5.1 is not mechanically sound. To address this problem, they suggest using not the multiplicity of an interface d.o.f. as the scale factor, but the ratio of the stiffnesses associated with an interface node in each sub-domain. Consider, for example, Figure 4, where four heterogeneous sub-domains are connected at a cross point. Let $k^{(i)}$ be the diagonal stiffness element at the crosspoint d.o.f. in sub-domain i. Then, the scale factors corresponding to the three Lagrange multipliers that connect sub-domain 1 to the remaining sub-domains are calculated as follows:

- collect the diagonal stiffnesses of all the d.o.f. that are incident at the cross point (in this case $k^{(1)}, k^{(2)}, k^{(3)}$ and $k^{(4)}$)
- the scale factor for the Lagrange multiplier connecting sub-domain 1 and j will be $k^{(j)}/(k^{(1)} + k^{(2)} + k^{(3)} + k^{(4)})$

Including the stiffnesses in the scale factors enables the preconditioners defined in Equations (2.30) and (2.31) to remain mechanically sound at minimal additional computational cost. Note that if the sub-domains at a cross point are homogeneous, their stiffnesses will be roughly similar, and the new scale factors will reduce to the multiplicity based scheme described in Section 2.5.1.

2.6 Optimal Decomposition of the Domain

The FETI method requires that the domain of interest be decomposed into sub-domains. Typically, when a problem is decomposed into sub-domains for implementation on a parallel processor, two factors determine the decomposition:

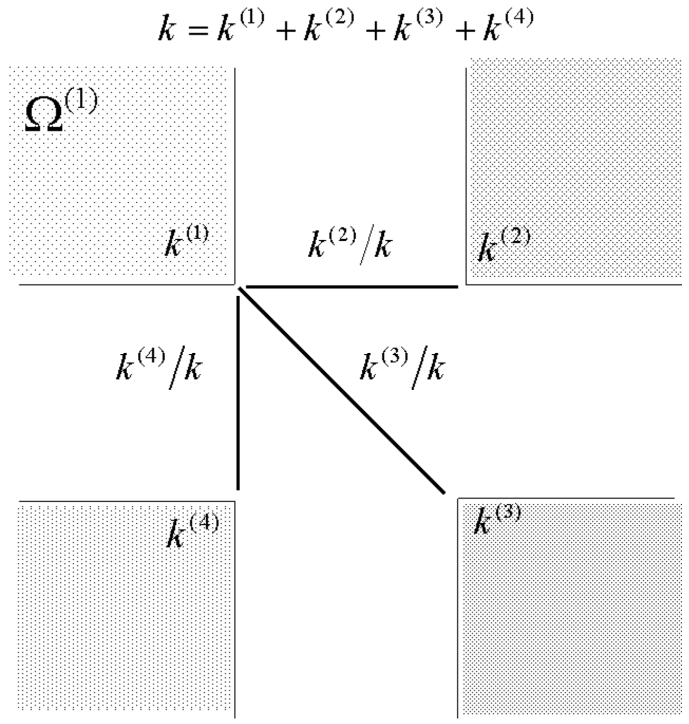

Figure 4: Handling heterogeneous sub-domains at a cross-point

- the edge-cut, which must be small to minimize interprocessor communication
- the size of the sub-domains. which must be roughly the same to minimize load imbalance across processors

Both these constraints are based on hardware and performance considerations and therefore valid for any algorithm. In the case of the FETI method, additional constraints that are based on the algorithm must be satisfied:

- **number of floating subdomains:** this must be kept small to minimize the size of the projection operator which is determined by the number of floating sub-domains.
- **sub-domain interconnectivity:** this must be kept high for faster convergence as the information in each iteration is propagated along the global interface
- **number of interface nodes:** this must be kept small as the number of interface nodes determines the size of the interface problem
- **bandwidth of the sub-domains:** if a skyline solver is used for the sub-domain solve, the work done in the factorization and solve is dependent on the bandwidth of the system matrix. It is therefore desirable that the bandwidth of the sub-domain systems be a small fraction of the bandwidth of the global problem, thus reducing the cost of sub-domain solves.
- **sub-domain aspect ratio:** in order to keep each sub-domain problem well-conditioned, it is desirable to have sub-domains with near perfect (=1) aspect ratios.

Note that several of these constraints are contradictory and may be difficult to achieve for real problems with irregular discretization and complex structure. Experimental results in [16, 11] have shown that the sub-domain aspect ratio is frequently the determining factor in numerical scalability and convergence of the FETI method for academic problems. In order to include this in the generation of the mesh decomposition, the authors in [9] describe the use of a greedy algorithm to generate a decomposition that gives good aspect ratios when the original mesh has elements with a good aspect ratio.

In the case of real problems, with irregular geometries, this domain decomposition problem is solved using a two-step approach. In the first step, an automatic mesh decomposer such as Metis [23] is used to obtain an initial partitioning of the mesh. This partitioning is then post-processed to obtain a decomposition that meets the criteria required by FETI. In [11] the authors observe the connection between the aspect ratio problem and the nearest centroid problem that is solved using clustering techniques [19] in the field of pattern recognition. They however reject this approach as being too expensive, instead focusing on non-deterministic algorithms such as Simulated Annealing, Tabu Search, and Stochastic Evolution to solve an optimization problem, where an appropriate cost function is optimized. This cost function takes into account constraints such as sub-domain aspect ratio, communication between sub-domains, load balance etc. In [37] the authors show that a two-step approach is always better than the conventional one-step approach that attempts to minimize only the edge-cut, and hence the communication.

Additional work in the area of optimizing the sub-domain aspect ratio has also been presented in [5, 6]. Further, in [14, 4], the authors discuss the removal of any mechanisms within a sub-domain in the post-processing step to ensure that the rigid body modes and the pseudo-inverses of the sub-domains can be calculated in a robust manner.

More recent work on multi-objective graph partitioning [35] leads one to believe that such methods could also be used to solve the sub-domain aspect ratio problem. In addition, instead of using techniques such as Simulated Annealing, one could also consider Genetic Algorithms [18] as an alternative approach to the optimization problem.

3 FETI LEVEL 1 METHOD - IMPLEMENTATION DETAILS

In this Section, we describe a prototype serial Fortran 77 code written to implement the FETI level 1 method. The code is very general, with a user-friendly interface that can be used to easily control various input options such as the type of preconditioning, the use of scaling in heterogeneous domains, the incorporation of re-orthogonalization etc. We next describe some of the implementation details of the main modules in the code, followed by a description of the user interface and the input files.

The implementation of the FETI level 1 method can be decomposed into several independent modules as follows:

- Sub-domain solves for each sub-domain
- Calculation of the pseudo-inverse on the floating sub-domains (that is, the matrix ${K^{(s)}}^{+}$)
- Calculation of the rigid body modes on the floating subdomains (that is, the matrices $R^{(s)}$)

- Handling the interior and interface nodes for each sub-domain to enable transfer from a sub-domain variable to the global interface variable and back (that is, the implementation of the matrices $B^{(s)}$)
- Matrix-vector products with G_I and G_I^T
- Calculation of $G_I^T G_I$ and $(G_I^T G_I)^{-1}$ for use in the projection operator (Equation (2.24))
- Conjugate projected gradient with re-orthogonalization and preconditioning (Algorithm 3)
- Calculation of the initial guess for the Lagrange multipliers (λ_0)
- Calculation of the right hand side for the CPG iterations (Equation (2.18))
- Calculation of α, given the λ obtained from the CPG iterations
- Calculation of the solution on each sub-domain, given the values of α and λ.
- Calculation and application of the Lumped and Dirichlet preconditioners, including the handling of cross-points and heterogeneous domains

Several of these modules are a simple implementation of the algorithms outlined in the previous sections, while others are more involved and described in greater detail in the following sections.

3.1 Calculation of the Rigid Body Modes and the Pseudo-inverse

In our implementation of the FETI method, the stiffness matrix $K^{(s)}$ for each sub-domain is input as a skyline matrix in profile-in storage format. This means that each column (or row) in the profile is stored starting with the first non-zero element in the column (or row) [22]. There are several pros and cons to this choice of storage - these will be discussed in greater detail at the end of this Section. In addition, we assume that the numbering of the nodes in each sub-domain is such that the interface boundary nodes are numbered at the end, after the interior nodes have been numbered as shown in Equation (2.26). This numbering enables easy implementation of the preconditioners and allows the interface d.o.f. to be extracted in a simple way from the sub-domain d.o.f.

When the matrix $K^{(s)}$ is non-singular, its Cholesky factorization is relatively straight-forward, and lends itself to easy optimization and parallelization [22]. Once $K^{(s)}$ is factored as

$$K^{(s)} = L^{(s)} {L^{(s)}}^T \tag{3.1}$$

where $L^{(s)}$ is a lower triangular matrix, an application of its inverse is simply

$$\begin{aligned} {K^{(s)}}^{-1} u &= (L^{(s)} {L^{(s)}}^T)^{-1} u \\ &= ({L^{(s)}}^T)^{-1}((L^{(s)})^{-1} u) \end{aligned}$$

However, when $K^{(s)}$ is singular, that is, $\Omega^{(s)}$ is a floating sub-domain, the factorization in Equation (3.1) runs into problems when a zero pivot is encountered. To discuss how this zero pivot can be used to efficiently calculate both the pseudo-inverse and the rigid body modes of $K^{(s)}$, we first digress to outline some theory.

Suppose the matrix $K^{(s)}$ were ordered such that it could be written as

$$\begin{bmatrix} K_{pp}^{(s)} & K_{pr}^{(s)} \\ K_{pr}^{(s)^T} & K_{rr}^{(s)} \end{bmatrix} \tag{3.2}$$

where the subscript p denotes a principal quantity and the subscript r denotes a redundant quantity. The matrix $K_{pp}^{(s)}$ has full rank equal to $n_i^{(s)} + n_b^{(s)} - n_r^{(s)}$, where $n_r^{(s)}$ is the number of rigid body modes of $K^{(s)}$ as well as the dimension of the square sub-matrix $K_{rr}^{(s)}$. Then, it can be easily shown that if we define

$$R^{(s)} = \begin{bmatrix} -(K_{pp}^{(s)})^{-1} K_{pr}^{(s)} \\ I \end{bmatrix} \tag{3.3}$$

it follows that

$$K^{(s)} R^{(s)} = 0 \tag{3.4}$$

that is, $R^{(s)}$ spans the null space of $K^{(s)}$ and thus can be used as the rigid body modes of $K^{(s)}$ [15].

The partitioning of the singular matrix $K^{(s)}$ as given in Equation (3.2) implies that

$$K_{rr}^{(s)} - K_{pr}^{(s)^T} K_{pp}^{(s)^{-1}} K_{pr}^{(s)} = 0 \tag{3.5}$$

Using this, we can easily verify that the matrix $K^{(s)^+}$ as defined as

$$K^{(s)^+} = \begin{bmatrix} (K_{pp}^{(s)})^{-1} & 0 \\ 0 & 0 \end{bmatrix} \tag{3.6}$$

is a pseudo-inverse of $K^{(s)}$, that is

$$K^{(s)} K^{(s)^+} K^{(s)} = K^{(s)} \tag{3.7}$$

So, a matrix-vector product of the form $K^{(s)^+} f^{(s)}$ can be written as follows:

$$K^{(s)^+} f^{(s)} = K^{(s)^+} \begin{bmatrix} f_p^{(s)} \\ f_r^{(s)} \end{bmatrix} \tag{3.8}$$

$$= \begin{bmatrix} (K_{pp}^{(s)})^{-1} f_p^{(s)} \\ 0 \end{bmatrix} \tag{3.9}$$

In theory, Equations (3.3) and (3.6) give us the rigid body modes and the pseudo-inverse of the matrix $K^{(s)}$. However, this does not help us in the practical implementation of these quantities as the matrix $K^{(s)}$ is seldom available in the form given in Equation (3.2). In [15], the authors propose a practical implementation for the case when the matrix $K^{(s)}$ is stored in the skyline form. They observe that a zero pivot that is encountered during the factorization of the matrix $K^{(s)}$ indicates a redundant equation that must be labeled and removed (see Figure 5).

To do this, they zero out the column in the skyline which has the zero pivot, set the pivot location to 1.0, and set the locations in the skyline matrix corresponding to that variable to be zero. The values in the column above the zero pivot are negated and copied to a separate vector prior to being zeroed out. This separate vector has 1.0 at the pivot location, and zero in the elements that lie outside the matrix profile for that column. Note that this vector essentially corresponds to a redundant column of $K^{(s)}$ which has

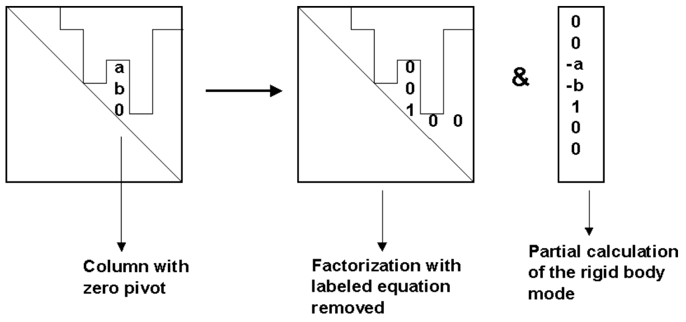
a
b
0
0
0
1
0 0
&
0
0
-a
-b
1
0
0
Column with zero pivot
Factorization with labeled equation removed
Partial calculation of the rigid body mode

Figure 5: Incorporating the calculation of pseudo-inverse and rigid body modes during the factorization of the skyline matrix. The column with zero pivot is replaced by zeros above the pivot, and the pivot is replaced by 1.0. The pivot variable is removed from the matrix by zeroing out the row corresponding to it. The pivot column is negated and a new vector created which is the partial calculation of the rigid body mode, corresponding to a forward reduction using the factors of $K_{pp}^{(s)}$ with a column of $K_{pr}^{(s)}$ as the right hand side.

undergone forward elimination in the process of the factorization. When the factorization is completed, a backward substitution on this vector yields a column of $-(K_{pp}^{(s)})^{-1}K_{pr}^{(s)}$. Doing this for all vectors that were obtained as a result of zero pivots gives us the matrix of the rigid body modes $R^{(s)}$. The factored matrix with the unlabeled equations corresponds to the Cholesky factors of $K_{pp}^{(s)}$.

In our prototype code, the locations of the zero pivots are stored and used in the calculation of ${K^{(s)}}^{+}$ times a vector to identify those variables that are labeled and therefore must be ignored. In practice, the value of a zero pivot element will rarely be exactly zero. We consider any pivot smaller than 1.0×10^{-7} to be a "zero" pivot. This value can be set by the user through the input data file (see Section 3.4). The choice of this value can be crucial in the FETI method as we need to correctly identify those small pivots that result from singularity, and those that result from ill-conditioning. In [8], the authors describe the problems that may result from the choice of the value for a small pivot, and suggest alternate ways in which the rigid body modes and the pseudo-inverse can be calculated.

Note that this approach to the calculation of the rigid body modes and the pseudo-inverse comes at very little additional cost. It can easily handle the case where the redundant quantities are anywhere in the matrix, thus allowing the use of re-ordering schemes on each sub-domain to reduce the profile of the skyline matrix. However, it does require that a column with a zero pivot be removed as soon as the pivot is encountered, a fact that fits in well with the way in which factorization is done on skyline matrices.

There are several other benefits as well as drawbacks to the use of the skyline storage scheme. The factorization of skyline matrices is relatively straight-forward to implement, optimize, and parallelize. It is also well suited to the calculation of the rigid body modes and the pseudo-inverse as described earlier. On the other hand, it can be wasteful in its use of memory as all the zeros within the profile of the skyline are stored. Since the boundary d.o.f. in each sub-domain are numbered last, this usually implies that the last few columns of the skyline matrix, corresponding to the boundary d.o.f., have a profile that is close to the size of matrix. If we were to apply a node renumbering scheme to the entire matrix in order to reduce the profile, and hence the amount of memory needed, it would likely result in the boundary d.o.f. being mixed with the interior d.o.f.. This would cause problems in the easy extraction of the boundary variables and in implementing the preconditioners. One solution to this problem would be to renumber only the interior d.o.f., keeping the boundary d.o.f. as the variables numbered last in each sub-domain. This approach would not only help the sub-domain solves, but also make it easier to implement $(K_{ii}^{(s)})^{-1}$ times a vector, which is an operation in the Dirichlet preconditioner. In our current prototype implementation, we have not renumbered the d.o.f. to reduce memory usage; however, this can be easily incorporated into the code.

3.2 Implementation of the Projection Operator

The projection operator, Equation (2.24), requires the evaluation of $(G_I^T G_I)^{-1}$ as well as the matrix-vector products using the matrices G_I and G_I^T, where G_I is as defined in Equation (2.18). The matrix-vector products are easy to implement once the rigid body modes $R^{(s)}$ have been determined for each sub-domain.

In order to form $(G_I^T G_I)^{-1}$, the matrix G_I is first formed explicitly from the rigid body modes. In our implementation, it is this matrix that is used in the matrix-vector products. Once G_I is formed, the matrix $G_I^T G_I$ is computed and its Cholesky factors obtained using the LAPACK routine DPOTRF [1]. The forward and back solves required in the projection are evaluated using the LAPACK routine DPOTRS. These routines require the matrix to be stored as a dense matrix. As the matrix is symmetric, only the upper or lower triangular part needs to be stored. Since the matrix $G_I^T G_I$ is at most of order $3N_f$ for two-dimensional problems and $6N_f$ for three-dimensional problems, where N_f is the number of floating sub-domains, it is a relatively small matrix and we incur little overhead in storing it as a dense matrix.

3.3 Implementation of the Preconditioners

The implementation of Lumped and Dirichlet preconditioners, Equations (2.30) and (2.31), is relatively straight-forward. Both preconditioners require the use of either the whole, or a part, of the stiffness matrix $K^{(s)}$ for the sub-domain. However, the stiffness matrix has already been replaced by the factors needed for sub-domain solves. This means that in order to implement the preconditioners, we need to store a copy of the original matrix as well. Further, we need the factors of the submatrix $K_{ii}^{(s)}$ for the Dirichlet preconditioner. If, as suggested, we number the interior nodes first, and apply a profile reduction to only the interior nodes, then the factors of $K_{ii}^{(s)}$ have already been computed for us during the factorization of $K^{(s)}$. Therefore, instead of storing the full stiffness matrix for use during preconditioning, we need additional memory for only the part of the matrix that corresponds to the boundary d.o.f.

3.4 Input files for the Prototype Code

Our serial F77 implementation of the FETI level 1 method is driven by two data files - a file that contains information on the problem (input_feti.data), and a file that contain information on the matrix (input_matrix.data). The stiffness matrices are input in skyline format, using profile-in storage. As mentioned earlier, for each sub-domain, the interior nodes are numbered first, followed by the interface boundary nodes. The input_matrix.data file contains the stiffness matrix and well as the right hand side for each sub-domain.

The input_feti.data file contains information on the problem itself, including the number of sub-domains, the number of interior and boundary nodes in each sub-domain, the list of interior nodes, and the list of boundary nodes, including connectivity information. For crosspoints, we include redundant information on the connectivity. This file also contains the parameters that can be used to control the method, such as the maximum number of iterations for CPG, the value of what constitutes a small pivot, and flags that determine if re-orthogonalization is done, the type of preconditioner that is used, and whether the domains are homogeneous or heterogeneous.

A sample test problem and the corresponding input data file containing the problem information is given in Appendix A.

4 EXPERIMENTAL RESULTS

Our prototype serial code F77 code for the FETI level 1 method was developed and debugged in stages. The modular design of the code enabled us to build and test the code incrementally. The code was tested using simple problems obtained from the equations of static equilibrium of a beam being pulled in the horizontal direction. We first started with the simple case of a two sub-domain problem with a single floating sub-domain (Figure 6). This problem was made more complex first by the addition of another floating sub-domain (Figure 7), and finally the addition of a cross-point (Figure 9).

Having debugged the code with these simple test problems, we next studied the behavior of the FETI level 1 method on a larger problem (Figure 8). In this case, we first started with all the sub-domains being homogeneous and then gradually introduced heterogeneity into the problem by changing the stiffness of the sub-domains. Finally, to study the effects of a different source of ill-conditioning, we considered a homogeneous problem with nearly incompressible elasticity coefficients. The results for these simple test cases are summarized in Table 1. For these, and other FETI results presented in this report, the convergence criterion for the CPG iterations was

$$||z_i||_2 \leqslant 1.0 \times 10^{-8} \tag{4.1}$$

where z_i is the projected residual, or the preconditioned projected residual, at the i-th iteration. Unless otherwise mentioned, the number of vectors used in re-orthogonalization is 15. Finally, for this simple test problem, we also compared the results from FETI with the solution of the global problem using a standard iterative technique, Conjugate Gradient with preconditioning. We used the iterative solver software from Compaq's DXML library [22], with the following convergence criterion:

$$||r_i||_2 \leqslant 1.0 \times 10^{-8} \tag{4.2}$$

where r_i is the residual of the global problem at the i-th iteration. The order of the polynomial in the polynomial preconditioning was 2, and the maximum number of iterations was 94 (equal to the size of the global problem). The results are summarized in Table 2.

5 SUMMARY AND CONCLUSIONS

In this report, we have focused mainly on the theory and prototype implementation of the FETI level 1 method. Our preliminary results indicate that the method holds promise for the solution of highly ill-conditioned systems that arise in structural mechanics problems at the Lawrence Livermore National Laboratory (LLNL). The numerical and parallel scalability of the method suggests that it would be a potential alternative to direct methods, especially as the size of the problem and the number of processors is increased. This would make it suitable for solving the problems being addressed in the Accelerated Strategic Computing Initiative (ASCI) [2].

While our prototype serial code is a first step towards understanding the FETI method, the issues involved in its implementation, and the role it can play in ASCI problems, much work remains to be done before the code can be used in a production environment. In particular, some of the issues that must be addressed include:

- Parallelization of the code
- Incorporation of the FETI level 2 method [12, 7, 28] to ensure numerical scalability for fourth order elasticity problems and plate bending problems
- Load balancing, parallelization of the coarse problem [34], and parallelization of the sub-domain solves for implementation on massively parallel systems
- Techniques for efficiently generating the decomposition of the domain that is suitable for the FETI method

The FETI method is gaining popularity as evidenced in the recent work of several authors [30, 21, 24, 33, 10, 27]. A public-domain parallel version of FETI in F90 is also available at [29]. This code was implemented as part of the PARASOL project and incorporated into industrial finite element codes. The time is thus right for serious consideration of the FETI method in problems of interest to LLNL.

6 ACKNOWLEDGEMENT

This work was done in collaboration with the Methods Development Group at LLNL. In particular, I would like to acknowledge Bob Ferencz for his insights into the computational mechanics aspects of the FETI method. He also provided the test problems used in this study. I would also like to thank Carol Hoover and Peter Raboin for their support of this effort. And finally, this study would not have been possible without the extensive work done in the area of FETI methods by Charbel Farhat and his colleagues.

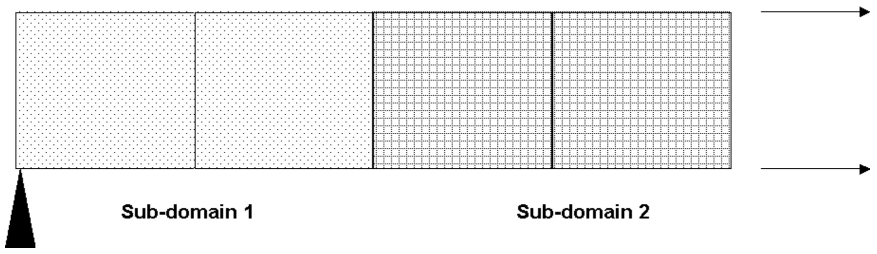

Figure 6: Simple 4 element test problem with a single floating domain

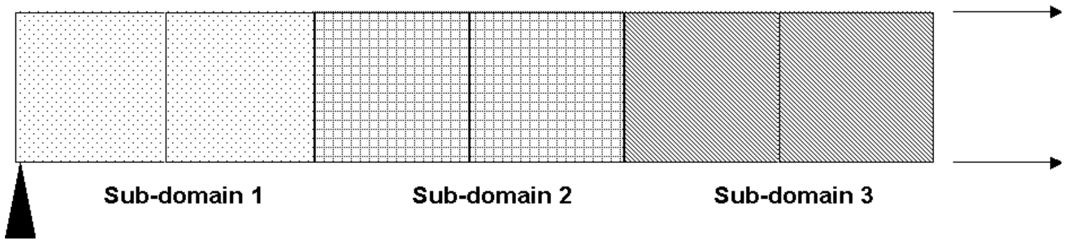

Figure 7: Simple 6 element test problem with two floating sub-domains

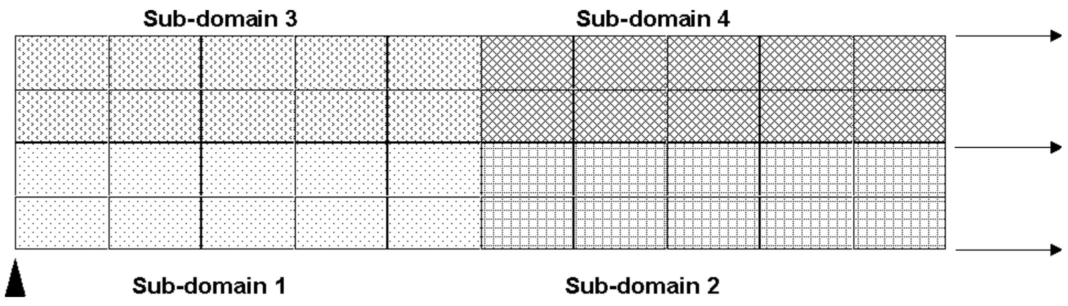

Figure 8: Simple test problem to understand the behavior of FETI

	No Preconditioning	Lumped Preconditioning	Dirichlet Preconditioning
Homogeneous Sub-domains	22	13	10
Sub-domain 4: 10x stiffness	25	13	10
Sub-domain 4: 1000x stiffness	24	13	9
Sub-domain 1: 100x stiffness, Sub-domain 2: 10x stiffness, Sub-domain 3: 1x stiffness, Sub-domain 4: 1000x stiffness	34	17	10
Homogeneous sub-domains, nearly incompressible elasticity coefficients	22	34	15
Homogeneous sub-domains, nearly incompressible elasticity coefficients with re-orthogonalization	20	27	12

Table 1: Behavior of the FETI method for the test problem in Figure 8

	No Preconditioning	Diagonal Preconditioning	Polynomial Preconditioning	ILU Preconditioning
Sub-domain 4: 10x stiffness	94 (close to the solution)	67	48	25
Sub-domain 4: 1000x stiffness	94 (not close to solution)	86	62	34
Sub-domain 1: 100x stiffness, Sub-domain 2: 10x stiffness, Sub-domain 3: 1x stiffness, Sub-domain 4: 1000x stiffness	94 (not close to solution)	75	55	28
Homogeneous sub-domains, nearly incompressible elasticity coefficients:	No convergence	No convergence	No convergence	Preconditioned matrix not positive definite

Table 2: Behavior of the CG method for the test problem in Figure 8

References

[1] Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., and Sorensen, D. *LAPACK Users' Guide, Third Edition.* SIAM, Philadelphia, 1999.

[2] ASCI: Accelerated Strategic Computing Initiative. http://www.llnl.gov/ASCI.

[3] Ben-Israel, A., and Greville, T. *Generalized Inverses: Theory and Applications.* John Wiley, 1973.

[4] Bhardwaj, M., Day, D., Farhat, C., Lesoinne, M., Pierson, K., and Rixen, D. Applications of the FETI Method to ASCI Problems: Scalability Results on One Thousand Processors and Discussion of Highly Heterogeneous Problems. Tech. Rep. SAND99-0937, Sandia National Laboratories Report, 1999.

[5] Diekmann, R., Meyer, D., and Monien, B. Parallel Decomposition of Unstructured FEM-meshes. *Concurrency: Practice and Experience 10* (1998), 53–72.

[6] Diekmann, R., Preis, R., Schlimbach, F., and Walshaw, C. Shape-Optimized Mesh Partitioning and Load Balancing for Parallel Adaptive FEM. *Parallel Computing* (2000).

[7] Farhat, C., Chen, P., Mandel, J., and Roux, F.-X. The two level FETI Method - Part II: Extension to Shell Problems Parallel Implementation, and Performance Results. *Computer Methods in Applied Mechanics and Engineering*, 155 (1998), 153–180.

[8] Farhat, C., and Geradin, M. On the General Solution by a Direct Method of a Large-scale Singular System of Linear Equations: Application to the Analysis of Floating Structures. *International Journal for Numerical Methods in Engineering 41* (1998), 675–696.

[9] Farhat, C., and Lesoinne, M. Automatic Partitioning of Unstructured Meshes for the parallel Solution of Problems in Computational Mechanics. *International Journal for Numerical Methods in Engineering 36* (1993), 745–764.

[10] Farhat, C., Lesoinne, M., LeTallec, P., Pierson, K., and Rixen, D. FETI-DP: A Dual-Primal Unified FETI Method Part I: A Faster Alternative to the Two-Level FETI Method. Tech. Rep. CU-CAS-99-15, College of Engineering, University of Colorado, 1999.

[11] Farhat, C., Maman, N., and Brown, G. Mesh Partitioning for Implicit Computations via Iterative Domain Decomposition: Impact and optimization of the Sub-domain Aspect Ratio. *International Journal for Numerical Methods in Engineering 38* (1995), 989–1000.

[12] Farhat, C., and Mandel, J. The two level FETI Method for Static and Dynamic Problems - Part I: An Optimal Iterative Solver for Biharmonic Systems. *Computer Methods in Applied Mechanics and Engineering*, 155 (1998), 153–180.

[13] Farhat, C., Mandel, J., and Roux, F.-X. Optimal Convergence Properties of the FETI Domain Decomposition Method. *Computer Methods in Applied Mechanics and Engineering*, 115 (1994), 365–385.

[14] Farhat, C., Pierson, K., and Lesoinne, M. The Second Generation of FETI Methods and their Application to the Parallel Solution of Large-Scale Linear and Geometrically Nonliner Structural Analysis Problems. Tech. Rep. CU-CAS-99-01, College of Engineering, University of Colorado, 1999.

[15] Farhat, C., and Roux, F.-X. A Method of Finite Element Tearing and Interconnecting and its Parallel Solution Algorithm. *International Journal for Numerical Methods in Engineering 32* (1991), 1205–1227.

[16] Farhat, C., and Roux, F.-X. Implicit Parallel Processing in Structural Mechanics. *Computational Mechanics Advances 2*, 1 (1994), 1–124.

[17] R. Ferencz, 1999. private communication.

[18] Goldberg, D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, 1989.

[19] Jain, A. K., and Dubes, R. C. *Algorithms for Clustering data.* Prentice Hall, 1988.

[20] Joshi, M., karypis, G., Kumar, V., Gupta, A., and Gustavson, F. Pspases: An efficient and scalable parallel sparse direct solver. In *International Workshop on Frontiers of Parallel Numerical Computations and Applications. Frontiers 99* (1999).

[21] Justino, M., Park, K., and Felippa, C. An Algebraically Partitioned FETI Method for Parallel Structural Analysis:Implementation and Numerical Performance Evaluation . *International Journal for Numerical Methods in Engineering 40* (1997), 2739–2758.

[22] Kamath, C., Ho, R., and Manley, D. DXML: A High Performance Scientific Subroutine Library. *Digital Technical Journal 6*, 3 (1994), 44–56.

[23] Karypis, G., and Kumar, V. Multi-level k-way Partitioning Scheme for Irregular Graphs. *Journal of Parallel and Distributed Computing 48* (1998), 71–95.

[24] Klawonn, A., and Widlund, O. A Domain Decomposition Method with Lagrange Multipliers for Linear Elasticity . Tech. Rep. TR 780, Department of Computer Science, Courant Institute, 1999.

[25] Maker, B., Hallquist, J., and Ferencz, R. NIKE3D: A Nonlinear, Implicit, Three-domainsional Finite Element Code for Solid and Structural Mecahnics - User's Manual. Tech. Rep. Report UCRL-MA-105268 Rev. 1, University of California, Lawrence Livermore National Laboratory, 1995.

[26] Mandel, J. Balancing Domain Decomposition. *Communications in Applied Numerical Methods and Engineering 9* (1993), 233–241.

[27] Mandel, J., and Tezaur, R. On the Convergence of a Dual-Primal Substructuring Method. Tech. Rep. UCD/CCM Report 150, Department of Mathematics, University of Colorado at Denver, 2000.

[28] Mandel, J., Tezaur, R., and Farhat, C. A Scalable Substructuring Method by Lagrange Multipliers for Plate Bending Problems. *SIAM Journal of Numerical Analysis 36* (1999), 1370–1391.

[29] PARASOL Software. http://www.pallas.com/parasol/.

[30] Park, K., Justino, M., and Felippa, C. An Algebraically Partitioned FETI Method for Parallel Structural Analysis: Algorithm Description . *International Journal for Numerical Methods in Engineering 40* (1997), 2717–2737.

[31] Parlett, B. *The Symmetric Eigenvalue Problem.* Prentice Hall, Inc., Englewood Cliffs, 1980.

[32] Rixen, D., and Farhat, C. A Simple and Efficient Extension of a Class of Substructure-based Preconditioners to Heterogeneous Structural Mechanics Problems. Tech. Rep. CU-CAS-97-18, College of Engineering, University of Colorado, 1997.

[33] Rixen, D., Farhat, C., Tezaur, R., and Mandel, J. Theoretical Comparison of the FETI and Algebraically Partitioned FETI Methods, and Performance Comparisons with a Direct Sparse Solver . *International Journal for Numerical Methods in Engineering 46* (1999), 501–534.

[34] Roux, F.-X., and Farhat, C. Parallel Implementation of Direct Solution Strategies for the Coarse Grid Solvers in 2-Level FETI Method . *Contemporary Mathematics 218* (1998), 158–173.

[35] Schloegel, K., Karypis, G., and Kumar, V. A New ALgorithm for Multi-objective Graph Partitioning. Tech. Rep. Preprint 99-063, Army High Performance Computing Research Center, 1999.

[36] Smith, B., Bjorstad, P., and Gropp, W. *Domain Decomposition: Parallel Multilevel Methods For Elliptic Partial Differential Equations.* Cambridge University Press, New York, 1996.

[37] vanderstraeten, D., Farhat, C., Chen, P., Keunings, R., and Ozone, O. A Retrofit Based Methodology for the Fast Generation and Optimization of Large-scale Mesh Partitions: Beyond the Minimum Interface Size Criterion. *Computer Methods in Applied Mechanics and Engineering*, 133 (1996), 25–45.

7 APPENDIX A: Sample test problem and input data set

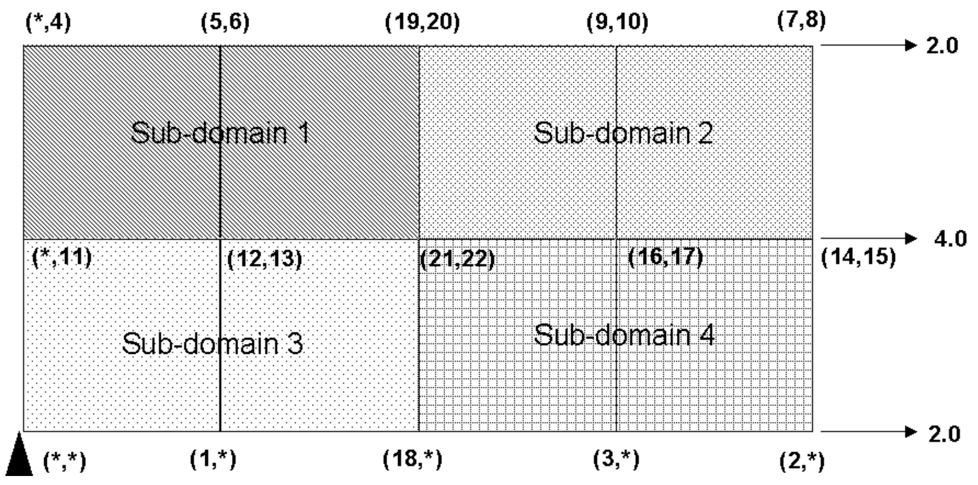

Figure 9: A simple test problem with 8 2-dimensional quadrilateral elements representing a bar being pulled in the x-direction. The numbers in parenthesis, (x,y), denote the unknowns, that is, the displacements in the x and y directions, respectively. Sub-domains 2 and 3 have one rigid body mode, while sub-domain 4 has 3 rigid body modes. Note that the interface boundary equations for each sub-domain are numbered last, after the "interior" nodes.

SAMPLE INPUT DATA FILE:

```
c input data file for the feti code: input_feti.data
c  name of output file = output_summry
c
   output_feti.data
c
c unit number for output file = iounit_summry
c
  7
c
c name of input file for matrix input = input_matrix
c
  input_matrix.data
c
c unit number for matrix input file = iounit_matrix
c
  8
c
c size of the problem = n_eqns
c
  22
c
c number of domains = n_domains
c
  4
c
c number of nodes in each domain = s_snodes, s_inodes
c     interior nodes    interface nodes
           1               10
           2               11
           3               11
           4               12
c
c list of interior nodes in a domain = idomain, s_list_inodes
c   domain number    node number
        1              1
        2              2
        2              3
        3              4
        3              5
        3              6
        4              7
        4              8
        4              9
        4              10
c
c list of interface nodes in a domain = idomain, s_list_inodes
c domain number  node number  domain shared with   multiplicity
      1                  11               3               2
      1                  12               3               2
      1                  13               3               2
      1                  18               2               2
      1                  21               2               4
      1                  21               3               4
      1                  21               4               4
      1                  22               2               4
      1                  22               3               4
      1                  22               4               4
   2        14      4       2
   2        15      4       2
   2        16      4       2
   2        17      4       2
   2        18      1       2
   2        21      1       4
   2        21      3       4
   2        21      4       4
   2        22      1       4
   2        22      3       4
   2        22      4       4
   3        11      1       2
   3        12      1       2
   3        13      1       2
   3        19      4       2
   3        20      4       2
   3        21      1       4
   3        21      2       4
   3        21      4       4
   3        22      1       4
   3        22      2       4
   3        22      4       4
   4        14      2       2
   4        15      2       2
   4        16      2       2
   4        17      2       2
   4        19      3       2
   4        20      3       2
   4        21      1       4
   4        21      2       4
   4        21      3       4
   4        22      1       4
   4        22      2       4
   4        22      3       4
c
c maximum number of iterations for cg = itmax
c
  12
c
c tolerance used in cg for convergence = errtol
c
  1.0e-7
c
c value of small pivot for identifying rigid body modes = spivot
c
  1.0e-7
c
c value of flag to determine re-orthogonalization - iflag_orth
c
  0
c
c number of vectors in re-orthogonalization - n_orth
c
  5
c
c value of flag to determine preconditioner - iflag_pcond
c  0 = none, 1 = lumped, 2 = dirichlet
  0
c
c value of flag that determines type of domains - iflag_hetero
c  0 = homogeneous, 1 = heterogeneous
  0
```