ELSEVIER

Comput. Methods Appl. Mech. Engrg. 184 (2000) 333–374

Computer methods in applied mechanics and engineering

www.elsevier.com/locate/cma

The second generation FETI methods and their application to the parallel solution of large-scale linear and geometrically non-linear structural analysis problems

Charbel Farhat *, Kendall Pierson, Michel Lesoinne

Department of Aerospace Engineering Sciences and Center for Aerospace Structures, University of Colorado at Boulder, Campus Box 429, Boulder, CO 80309-0429, USA

Abstract

The FETI algorithms are a family of numerically scalable domain decomposition methods. They have been designed in the early 1990s for solving iteratively and on parallel machines, large-scale systems of equations arising from the finite element discretization of solid mechanics, structural engineering, structural dynamics, and acoustic scattering problems, and for analyzing complex structures obtained from the assembly of substructures with incompatible discrete interfaces. In this paper, we present the second generation of these methods that operate more efficiently on large numbers of subdomains, offer greater robustness, better performance, and more flexibility for implementation on a wider variety of computational platforms. We also report on the application and performance of these methods for the solution of geometrically non-linear structural analysis problems. We discuss key aspects of their implementation on shared and distributed memory parallel processors, benchmark them against optimized direct sparse solvers, and highlight their potential with the solution of large-scale structural mechanics problems with several million degrees of freedom.

1. Introduction

Sparse direct solvers, which have been for a long time the main if not only workhorse of commercial finite element structural and thermal software, continue to play an important role in these numerical simulation codes. However, with the pressing need for higher-fidelity three-dimensional finite element structural models with millions of degrees of freedom, and the extreme demands placed by direct solvers on computer resources for such large problems, a large segment of the computational mechanics community including software development houses is increasingly investigating the adoption of iterative methods. Today, ANSYS and NISA, among others, feature powerful iterative solvers, and tomorrow, several other vendors will likely follow this trend. The significant progress achieved during the last decade in the development of fast and robust iterative algorithms for the solution of solid mechanics and plate and shell structural problems, and the advent of commercial parallel hardware, are two other key factors for this change in culture. The two latter factors are interrelated as iterative methods are more amenable to parallel processing than direct algorithms, and their development has tremendously benefited and continue to flourish under programs such as the National Science Foundation's high performance computing (HPC) and the Department of Energy's accelerated strategic computing initiative (ASCI) which continue to emphasize parallel computing.

* Corresponding author. Tel.: +1-303-492-3992; fax: +1-303-492-4990.

0045-7825/00/$ - see front matter
PII: S0045-7825(99)00234-0

Because a large class of structural and solid mechanics applications generate systems of equations that are symmetric positive definite (SPD) or semi-definite (SPSD), the conjugate gradient (CG) [1] method has always been an iterative method of choice among computational structural mechanicians. Early work on preconditioning techniques for this method has focused on incomplete Cholesky factorization procedures [2,3]. Then, attention shifted to element-by-element preconditioning techniques [4] because these utilize the structure inherent in finite element formulations and implementations, and to multigrid algorithms [5,6] because multigrid theories address the concept of numerical scalability – that is, the achievement of a convergence rate that is asymptotically independent of the size of the problem to be solved. With the advent of parallel processing, CG based iterative substructuring methods [7], also known as domain decomposition (DD) methods [8], have emerged as powerful contenders on both sequential and parallel computing platforms [9,10]. When equipped with an appropriate subdomain level preconditioner, a DD method can be numerically scalable with respect to the mesh size h (or number of elements) of the given problem. In order to be also numerically scalable with respect to the subdomain size H (or number of subdomains), it must also be equipped with a "coarse space" preconditioner [11] whose mathematical foundation is similar to that of the "coarse grids" encountered in multigrid methods. Usually, parallelism in a DD method is achieved by mapping one or several subdomains onto a processor. For this reason, scalability with respect to the number of subdomains is paramount to parallel scalability – that is, the ability to deliver a larger speedup for a larger number of processors.

The FETI [12,13] and the related Balancing [14] methods are among the first non-overlapping DD methods that have demonstrated numerical scalability with respect to both the mesh and subdomain sizes, for both second-order elasticity [15] and fourth-order plate and shell problems [16–18]. In particular, the parallel scalability of the FETI method and its ability to outperform several popular direct and iterative algorithms on both sequential and parallel computers have been extensively demonstrated (for example, see [10,19]).

The FETI method is a Lagrange multiplier based DD method. It can be viewed as a two-step preconditioned conjugated gradient (PCG) iterative procedure where subdomain problems with Dirichlet boundary conditions are solved in the preconditioning step, and related subdomain problems with Neumann boundary conditions are solved in a second step. This method is also known as the dual Schur complement method because on the outset, it constructs a dual Schur complement operator [10]. When the FETI method applied to the solution of structural problems is equipped with the so-called Dirichlet preconditioner [15] and with the optional corner Lagrange multipliers for plate [17] and shell [18] problems, the condition number of its interface problem grows asymptotically only as

$$\kappa = \mathrm{O}\left(1 + \log^2 (H/h)\right), \tag{1}$$

which proves that the FETI method is numerically scalable with respect to both the mesh size h and the subdomain size H.

The original FETI method has been successfully extended to free-vibration analysis [19], component mode synthesis [20], transient response analysis [21], systems with multiple and/or repeated right-hand sides [22], heterogeneous problems [23,24] – that is, problems with severe discontinuities in material properties – and acoustic scattering problems [25,26]. It has also inspired many variants, extensions, and applications, among which we note those described in [27–33]. All these DD methods share the so-called Dirichlet and lumped preconditioners, and the rigid body (or more generally zero energy) modes based auxiliary coarse problem, that were originally developed for the FETI method. For second-order elasticity as well as fourth-order plate and shell problems, the Dirichlet preconditioner ensures scalability with respect to the mesh size h of most FETI-like DD methods. The lumped preconditioner is a more economical version of the Dirichlet preconditioner that, for many second-order elasticity problems, delivers a superior computational performance even when it does not deliver a scalable iteration count [10].

The main objectives of this paper are to present a simplified and unified formulation of the various FETI methods that have been previously developed, and to report on recent advances in areas pertaining to the robust and efficient implementation of these methods on parallel processors. These areas include the selection of the number of subdomains and their mapping onto a given set of processors, the definition and generation of a "good" mesh partition for the FETI method, the selection of a subdomain equation solver,

the extraction of the subdomain zero energy modes that are needed for constructing FETIs basic coarse problem, the efficient solution of this coarse problem, and the application of the FETI method to geometrically non-linear problems. Progress achieved in each of these areas has led to the design of the second generation FETI solvers.

2. The one-level FETI method

2.1. From a global primal problem to a dual interface problem

The general form of a linear or linearized algebraic problem arising from the discretization of a computational structural mechanics problem defined on a domain Ω can be written as

$$\mathbf{K}\mathbf{u} = \mathbf{f}, \tag{2}$$

where $\mathbf{K}$ is an SPD or SPSD stiffness matrix, $\mathbf{u}$ a vector of generalized displacements, and $\mathbf{f}$ a vector of prescribed forces. A *non-overlapping* DD method with matching interfaces for solving efficiently the above problem on a sequential or parallel processor can be devised by partitioning Ω onto N_s non-overlapping subdomains $\Omega^{(s)}$, expressing the local equilibrium of each subdomain, and introducing displacement compatibility constraints at the subdomain interfaces. If Lagrange multipliers are employed to enforce these constriants, the original mechanical problem can be reformulated as a saddle point problem whose Euler equations are given by Farhat [12]

$$\delta_{\mathbf{v}^{(s)},\mu}\mathscr{L}(\mathbf{v}^{(s)},\mu) = 0, \tag{3}$$

where $\delta_{\mathbf{v}^{(s)},\mu}$ designates the variation with respect to the arguments $\mathbf{v}^{(s)}$ and μ, and $\mathscr{L}$ is the Lagrangian of the problem and can be written as

$$\mathscr{L}(\mathbf{v}^{(s)},\mu) = \sum_{s=1}^{N_s}\left(\frac{1}{2}\mathbf{v}^{(s)^T}\mathbf{K}^{(s)}\mathbf{v}^{(s)} - \mathbf{v}^{(s)^T}\mathbf{f}^{(s)}\right) + \mu^T\sum_{s=1}^{N_s}\mathbf{B}^{(s)}\mathbf{v}^{(s)}. \tag{4}$$

Here, the superscript s designates a subdomain quantity, the superscript T denotes the transpose of a quantity, $\mathbf{v}^{(s)}$ and μ are some admissible subdomain displacement and Lagrange multiplier fields, respectively, $\mathbf{K}^{(s)}$ is the subdomain stiffness matrix, $\mathbf{f}^{(s)}$ the subdomain vector of prescribed forces, and $\mathbf{B}^{(s)}$ the signed Boolean matrices that extracts from a subdomain vector $\mathbf{v}^{(s)}$ its signed ($\pm$) restriction to the subdomain interface boundary. The Euler equations (2) lead to the following constraint problem which is equivalent to the original problem (1)

$$\begin{aligned}&\mathbf{K}^{(s)}\mathbf{u}^{(s)} = \mathbf{f}^{(s)^\lambda}\lambda \quad s = 1,\dots N_s,\\ &\sum_{s=1}^{N_s}\mathbf{B}^{(s)}\mathbf{u}^{(s)} = 0,\end{aligned} \tag{5}$$

where $\mathbf{u}^{(s)}$ and λ are respectively the subdomain displacements and Lagrange multipliers that are the stationary points of $\mathscr{L}$. The first of Eqs. (5) expresses the local equilibrium of the subdomains $\Omega^{(s)}$, and the second of Eqs. (5) the continuity of the subdomain displacement fields across the subdomain interfaces.

Once λ is determined, the subdomain displacement fields can be recovered by solving concurrently the equilibrium equations to obtain

$$\mathbf{u}^{(s)} = \mathbf{K}^{(s)^+}\left(\mathbf{f}^{(s)} - \mathbf{B}^{(s)^T}\lambda\right) + \mathbf{R}^{(s)}\alpha^{(s)}, \tag{6}$$

where $\mathbf{K}^{(s)^+}$ denotes the inverse of $\mathbf{K}^{(s)}$ if $\Omega^{(s)}$ has sufficient Dirichlet boundary conditions to prevent $\mathbf{K}^{(s)}$ from being singular, or a generalized inverse of $\mathbf{K}^{(s)}$ if $\Omega^{(s)}$ is a floating subdomain. In the latter case, the columns of $\mathbf{R}^{(s)}$ represent the rigid body (or more generally zero energy) modes of $\Omega^{(s)}$, i.e. $\mathbf{R}^{(s)} = \ker\ \mathbf{K}^{(s)}$, and $\alpha^{(s)}$ is the set of amplitudes that specifies the contribution of the null space $\mathbf{R}^{(s)}$ to the solution $\mathbf{u}^{(s)}$.

These coefficients can be determined by requiring that each subdomain problem be mathematically solvable – that is, each floating subdomain be self-equilibrated – which can be written as

$$\mathbf{R}^{(s)^{\mathrm{T}}}\left(\mathbf{r}^{(s)} - \mathbf{B}^{(s)^{\mathrm{T}}}\lambda\right) = 0. \tag{7}$$

Substituting Eq. (5) into the compatibility equation and exploiting the solvability condition (6) transforms problem (4) into the interface problem

$$\begin{bmatrix} \mathbf{F}_{\mathrm{I}} & -\mathbf{G}_{\mathrm{I}} \\ -\mathbf{G}_{\mathrm{I}}^{\mathrm{T}} & 0 \end{bmatrix} \begin{bmatrix} \lambda \\ \alpha \end{bmatrix} = \begin{bmatrix} \mathbf{d} \\ -\mathbf{e} \end{bmatrix}, \tag{8}$$

where

$$\begin{aligned} \mathbf{F}_{\mathrm{I}} &= \sum_{s=1}^{N_s} \mathbf{B}^{(s)}\mathbf{K}^{(s)^{+}}\mathbf{B}^{(s)^{\mathrm{T}}}, \\ \mathbf{d} &= \sum_{s=1}^{N_s} \mathbf{B}^{(s)}\mathbf{K}^{(s)^{+}}\mathbf{f}^{(s)}, \\ \mathbf{G}_{\mathrm{I}} &= \left[\mathbf{B}^{(1)}\mathbf{R}^{(1)} \ \ldots \ \mathbf{B}^{(N_{\mathrm{f}})}\mathbf{R}^{(N_{\mathrm{f}})}\right], \\ \alpha &= \left[\alpha^{(1)^{\mathrm{T}}} \ \ldots \ \alpha^{(N_{\mathrm{f}})^{\mathrm{T}}}\right]^{\mathrm{T}}, \\ \mathbf{e} &= \left[\mathbf{f}^{(1)^{\mathrm{T}}}\mathbf{R}^{(1)} \ \ldots \ \mathbf{f}^{(N_s)^{\mathrm{T}}}\mathbf{R}^{(N_{\mathrm{f}})}\right]^{\mathrm{T}} \end{aligned} \tag{9}$$

and where N_{f} denotes the number of floating subdomains.

Because λ is a dual variable to the primal variables $\mathbf{u}^{(s)}$, the interface problem (7) is called a dual interface problem. It partially defines the basic FETI method that was originally introduced in [12,13]. This interface problem is best solved by an interative algorithm, the choice of which completes the description of the FETI method.

2.2. Iterative solution of the dual interface problem

In the FETI method, the interface problem is solved by a preconditioned conjugate projected gradient (PCPG) algorithm. More specifically, the indefinite interface problem (7) is transformed into a semi-definite system of equations by eliminating the self-equilibrium condition $\mathbf{G}_{\mathrm{I}}^{\mathrm{T}}\lambda = \mathbf{e}$ using the splitting

$$\lambda = \lambda^0 + \mathbf{P}(\mathbf{Q})\bar{\lambda}, \tag{10}$$

where λ^0 is a particular solution of $\mathbf{G}_{\mathrm{I}}^{\mathrm{T}}\lambda = \mathbf{e}$ of the form

$$\lambda^0 = \mathbf{Q}\mathbf{G}_{\mathrm{I}}(\mathbf{G}_{\mathrm{I}}^{\mathrm{T}}\mathbf{Q}\mathbf{G}_{\mathrm{I}})^{-1}\mathbf{e} \tag{11}$$

and $\mathbf{P}(\mathbf{Q})$ is a projection operator defined for a given matrix $\mathbf{Q}$ by

$$\mathbf{P}(\mathbf{Q}) = \mathbf{I} - \mathbf{Q}\mathbf{G}_{\mathrm{I}}(\mathbf{G}_{\mathrm{I}}^{\mathrm{T}}\mathbf{Q}\mathbf{G}_{\mathrm{I}})^{-1}\mathbf{G}_{\mathrm{I}}^{\mathrm{T}}. \tag{12}$$

Note that for any matrix $\mathbf{Q}$, the following holds

$$\mathbf{P}^2(\mathbf{Q}) = \mathbf{P}(\mathbf{Q}), \qquad \mathbf{G}_{\mathrm{I}}^{\mathrm{T}}\mathbf{P}(\mathbf{Q}) = 0. \tag{13}$$

Throughout the remainder of this paper, we denote the projector (12) simply by $\mathbf{P}$.

Applying the splitting (10) to the interface problem (8) leads to the projected interface problem

$$(\mathbf{P}^{\mathrm{T}}\mathbf{F}_{\mathrm{I}}\mathbf{P})\bar{\lambda} = \mathbf{P}^{\mathrm{T}}(\mathbf{d} - \mathbf{F}_{\mathrm{I}}\lambda^0). \tag{14}$$

For any given matrix $\mathbf{Q}$, the above projected interface problem is SPSD. Hence, it can be solved by the following PCG algorithm where $\tilde{\mathbf{F}}_{\mathrm{I}}^{-1}$ denotes a chosen preconditioner

$$\bar{\lambda}^0 = 0,$$

$$\mathbf{w}^0 = \mathbf{P}^{\mathrm{T}}(\mathbf{d} - \mathbf{F}_{\mathrm{I}}\lambda^0),$$

Iterate $k = 0, 1, \ldots$ until convergence,

$$\bar{\mathbf{y}}^k = \tilde{\mathbf{F}}_{\mathrm{I}}^{-1}\mathbf{w}^k,$$

$$\bar{\mathbf{p}}^k = \bar{\mathbf{y}}^k - \sum_{i=0}^{k-1} \frac{\bar{\mathbf{y}}^{k\mathrm{T}}(\mathbf{P}^{\mathrm{T}}\mathbf{F}_{\mathrm{I}}\mathbf{P})\bar{\mathbf{p}}^i}{\bar{\mathbf{p}}^{i\mathrm{T}}(\mathbf{P}^{\mathrm{T}}\mathbf{F}_{\mathrm{I}}\mathbf{P})\bar{\mathbf{p}}^i}\bar{\mathbf{p}}^i,$$

$$\eta^k = \frac{\bar{\mathbf{p}}^{k\mathrm{T}}\mathbf{w}^k}{\bar{\mathbf{p}}^{k\mathrm{T}}(\mathbf{P}^{\mathrm{T}}\mathbf{F}_{\mathrm{I}}\mathbf{P})\bar{\mathbf{p}}^k},$$

$$\bar{\lambda}^{k+1} = \bar{\lambda}^k + \eta^k\bar{\mathbf{p}}^k,$$

$$\mathbf{w}^{k+1} = \mathbf{w}^k - \eta^k\mathbf{P}^{\mathrm{T}}\mathbf{F}_{\mathrm{I}}\mathbf{P}\bar{\mathbf{p}}^k.$$

The PCG algorithm described above can be redesigned to iterate directly on $\lambda^k = \lambda^0 + \mathbf{P}\lambda^{-k}$. Indeed, operating directly on $\mathbf{y}^k = \mathbf{P}\bar{\mathbf{y}}^k$ and $\mathbf{p}^k = \mathbf{P}\bar{\mathbf{p}}^k$, and exploiting the first of properties (13) leads to the by now well-known FETI PCPG solver summarized in Box 1. This solver has been renamed the "one-level" FETI solver in order to distinguish it from the two-level FETI method that was more recently introduced in [17,18,34].

Box 1. The one-level FETI PCPG method.

1. Initialize

$$\lambda^0 = \mathbf{Q}\mathbf{G}_{\mathrm{I}}(\mathbf{G}_{\mathrm{I}}^{\mathrm{T}}\mathbf{Q}\mathbf{G}_{\mathrm{I}})^{-1}\mathbf{e}$$
$$\mathbf{w}^0 = \mathbf{P}^{\mathrm{T}}(\mathbf{d} - \mathbf{F}_{\mathrm{I}}\lambda^0)$$

2. Iterate $k = 0, 1, \ldots$ until convergence

$$\mathbf{y}^k = \mathbf{P}\tilde{\mathbf{F}}_{\mathrm{I}}^{-1}\mathbf{w}^k$$
$$\mathbf{p}^k = \mathbf{y}^k - \sum_{i=0}^{k-1} \frac{\mathbf{y}^{k\mathrm{T}}\mathbf{F}_{\mathrm{I}}\mathbf{p}^i}{\mathbf{p}^{i\mathrm{T}}\mathbf{F}_{\mathrm{I}}\mathbf{p}_i}\mathbf{p}^i$$
$$\eta^k = \frac{\mathbf{p}^{k\mathrm{T}}\mathbf{w}^k}{\mathbf{p}^{k\mathrm{T}}\mathbf{F}_{\mathrm{I}}\mathbf{p}^k} \tag{15}$$
$$\lambda^{k+1} = \lambda^k + \eta^k\mathbf{p}^k$$
$$\mathbf{w}^{k+1} = \mathbf{w}^k - \eta^k\mathbf{P}^{\mathrm{T}}\mathbf{F}_{\mathrm{I}}\mathbf{p}^k$$

Remark 1. *The straightforward formulation of the subdomain displacement compatibility equations generates redundant constraints at the crosspoints of a mesh partition* (*points belonging to more than two subdomains*), *and therefore* $\mathbf{B}^{\mathrm{T}}$ *does not have in general a full column rank* [35]. *It follows that* $\mathbf{F}_{\mathrm{I}}$ *and* $\tilde{\mathbf{F}}_{\mathrm{I}}^{-1}$ *are in general semi-definite, and that the solution* λ *of the interface problem* (12) *is not unique. However, the corresponding subdomain displacement solutions* $\mathbf{u}^{(s)}$ *are unique. While programming tricks can be invoked to avoid redundant compatibility constraints at the crosspoints, it is preferable to keep the full redundancy in the system because its accelerates the convergence of the FETI method. A detailed explanation of this issue as well as mechanical interpretation of this result can be found in* [24].

Remark 2. *The matrix* $\mathbf{G}_{\mathrm{I}}^{\mathrm{T}}\mathbf{Q}\mathbf{G}_{\mathrm{I}}$ *defines an auxiliary coarse problem that couples all the subdomain computations, propagates the error globally, and accelerates convergence. It is because of this coarse problem that when the Dirichlet preconditioner is used, the condition number of the FETI interface problem* (12) *can be bounded by a polylogarithmic function of* H/h (1), *where* H *and* h *denote the subdomain and mesh sizes, respectively. We remind the reader that such a bound proves the numerical scalability of the FETI method with respect to both the problem size and number of subdomains.*

Remark 3. *If the given global structure is fully restrained – that is, if* $\mathbf{K}$ *is not singular –* $\mathbf{G}_{\mathrm{I}}$ *has full column rank and* $\mathbf{G}_{\mathrm{I}}^{\mathrm{T}}\mathbf{G}_{\mathrm{I}}$ *is non-singular* [10]. *In that case, it makes sense to restrict the choice of the matrix* $\mathbf{Q}$ *by the condition that* $\mathbf{G}_{\mathrm{I}}^{\mathrm{T}}\mathbf{Q}\mathbf{G}_{\mathrm{I}}$ *be also non-singular. Furthermore, if* $\mathbf{Q}$ *is chosen among symmetric matrices,* $\mathbf{G}_{\mathrm{I}}^{\mathrm{T}}\mathbf{Q}\mathbf{G}_{\mathrm{I}}$ *becomes symmetric and thus more economical to handle. For homogeneous problems – that is, problems without severe discontinuities in material properties – the simplest choice* $\mathbf{Q}=\mathbf{I}$ *is the most computationally efficient one. This choice is adopted here for all numerical examples.*

Remark 4. *If the given global structure is not fully restrained – that is, if* $\mathbf{K}$ *is singular – the matrix* $\mathbf{G}_{\mathrm{I}}^{\mathrm{T}}\mathbf{Q}\mathbf{G}_{\mathrm{I}}$ *becomes also singular* [10]. *In that case, the FETI method is still applicable and the null space of* $\mathbf{K}$ *can be retrieved using the strategy described in* [36].

Remark 5. *As indicated in Box* 1, *the FETI algorithm is always used with an explicit full reorthogonalization procedure in order to accelerate convergence. In* [10], *it was shown that such a strategy is cost-effective for subdomain problems because reorthogonalization is applied only to the interface Lagrange multiplier unknowns.*

So far, two different preconditioners have been proposed in the literature for the FETI method: the mathematically optimal Dirichlet preconditioner $\bar{\mathbf{F}}_{\mathrm{I}}^{\mathrm{D}^{-1}}$ introduced in [15], and the computationally economical lumped preconditioner $\bar{\mathbf{F}}_{\mathrm{I}}^{\mathrm{L}^{-1}}$ proposed in earlier works [12,13]. If each subdomain stiffness matrix is partitioned as

$$\mathbf{K}^{(\mathrm{s})}=\begin{bmatrix}\mathbf{K}_{\mathrm{ii}}^{(\mathrm{s})} & \mathbf{K}_{\mathrm{ib}}^{(\mathrm{s})}\\ \mathbf{K}_{\mathrm{ib}}^{(\mathrm{s})^{\mathrm{T}}} & \mathbf{K}_{\mathrm{bb}}^{(\mathrm{s})}\end{bmatrix}, \tag{16}$$

where the subscripts i and b designate the subdomain interior and interface boundary degrees of freedom (d.o.f.), respectively, then the Dirichlet preconditioner can be written as

$$\bar{\mathbf{F}}_{\mathrm{I}}^{\mathrm{D}^{-1}}=\mathbf{W}\left(\sum_{s=1}^{N_{\mathrm{s}}}\mathbf{B}^{(\mathrm{s})}\begin{bmatrix}0 & 0\\ 0 & \mathbf{S}_{\mathrm{bb}}^{(\mathrm{s})}\end{bmatrix}\mathbf{B}^{(\mathrm{s})^{\mathrm{T}}}\right)\mathbf{W} \tag{17}$$

and the lumped preconditioner is given by

$$\bar{\mathbf{F}}_{\mathrm{I}}^{\mathrm{L}^{-1}}=\mathbf{W}\left(\sum_{s=1}^{N_{\mathrm{s}}}\mathbf{B}^{(\mathrm{s})}\begin{bmatrix}0 & 0\\ 0 & \mathbf{K}_{\mathrm{bb}}^{(\mathrm{s})}\end{bmatrix}\mathbf{B}^{(\mathrm{s})^{\mathrm{T}}}\right)\mathbf{W}. \tag{18}$$

In the above expressions of $\bar{\mathbf{F}}_{\mathrm{I}}^{\mathrm{D}^{-1}}$ and $\bar{\mathbf{F}}_{\mathrm{I}}^{\mathrm{L}^{-1}}$, $\mathbf{W}$ is a diagonal matrix storing in each of its entries the inverse of the multiplicity of an interfaced d.o.f [37] that is, the inverse of the number of subdomains to which an interface d.o.f. belongs – and $\mathbf{S}_{\mathrm{bb}}^{(\mathrm{s})}$ is the following subdomain primal Schur complement

$$\mathbf{S}_{\mathrm{bb}}^{(\mathrm{s})}=\mathbf{K}_{\mathrm{bb}}^{(\mathrm{s})}-\mathbf{K}_{\mathrm{ib}}^{(\mathrm{s})^{\mathrm{T}}}\mathbf{K}_{\mathrm{ii}}^{(\mathrm{s})^{-1}}\mathbf{K}_{\mathrm{ib}}^{(\mathrm{s})}. \tag{19}$$

Note that $\mathbf{K}_{\mathrm{ii}}^{(\mathrm{s})}$ is non-singular because it is the stiffness matrix of $\Omega^{(\mathrm{s})}$ with all interface boundaries fixed and thus $\mathbf{K}_{\mathrm{ii}}^{(\mathrm{s})^{-1}}$ exists. Both Dirichlet and lumped preconditioners have been recently extended in [23,24] for addressing more efficiently heterogeneous problems. In particular, it was shown in [24] that such extensions can be simply designed by redefining appropriately the scaling matrix $\mathbf{W}$.

Finally, we note that for heterogeneous problems, it was recommended in [10] to choose $\mathbf{Q}=\bar{\mathbf{F}}_{\mathrm{I}}^{\mathrm{D}^{-1}}$ or $\mathbf{Q}=\bar{\mathbf{F}}_{\mathrm{I}}^{\mathrm{L}^{-1}}$, depending on the selected FETI preconditioner.

3. The two-level FETI method

For second-order elasticity problems (i.e plane stress/strain and solid elements), the FETI PCPG algorithm summarized in Box 1 and equipped with the Dirichlet preconditioner $\tilde{\mathbf{F}}_{\mathrm{I}}^{-1} = \mathbf{F}_{\mathrm{I}}^{\mathrm{D}^{-1}}$ is numerically scalable with respect to both the mesh size h and subdomain size H. However, it was observed numerically that the condition number (1) does not hold for plate and shell problems, and that the one-level FETI algorithm summarized in Box 1 is not numerically scalable for such fourth-order elasticity problems. In [17,18], the authors have shown that a numerically scalable extension of the FETI method to plate and shell problems can be obtained by enforcing exactly the continuity of the transverse displacement field at the subdomain crosspoints throughout the PCPG iterations. A similar idea has also been independently developed for the Balancing method [16]. Furthermore, the authors of [17,18] have also shown that such a constraint can be enforced by solving yet another auxiliary coarse problem that contains not only the subdomain zero energy modes as in the original FETI method (see Remark 2), but also the so-called subdomain corner modes [17,18]. This enriched coarse problem, which transforms the original FETI method into a genuine two-level algorithm, has led to the development of an even more powerful FETI method known as the two-level FETI method [34].

The two-level FETI method is described in [34] as a one-level FETI PCPG algorithm where an optional admissible constriant of the form

$$\mathbf{C}^{\mathrm{T}}\mathbf{w}^k = 0 \tag{20}$$

is enforced at each iteration k. Matrix $\mathbf{C}$ is rectangular and represents some subspace to be determined, and $\mathbf{w}^k = \mathbf{P}^{\mathrm{T}}(\mathbf{d} - \mathbf{F}_{\mathrm{I}}\lambda^k)$ (see Box 1).

In this paper, we present a new derivation of the two-level FETI method that is not only more elegant than those previously described in [17,34], but also offers the following advantages

1. it explains mathematically why the two-level FETI method converges in general faster than its one-level counterpart, and
2. it leads to a new expression of the same auxiliary coarse problem that was first presented in [17], and for which a more efficient computer implementation is proposed in Section 6.

3.1. Beyond Krylov spaces

Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be a generic SPD matrix, $\mathbf{b} \in \mathbb{R}^n$ a given vector, and $\mathbf{x}.^{\mathrm{ex}} \in \mathbb{R}^n$ the exact solution of the generic problem

$$\mathbf{A}\mathbf{x} = \mathbf{b}. \tag{21}$$

Given an initial guess $\mathbf{x}^0$, a gradient method produces a sequence of approximations to $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ that are constructed as follows

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \eta^k\mathbf{p}^k, \tag{22}$$

where $\mathbf{p}^k$ is an element of the Krylov space of dimension at most $k+1$ generated by $\mathbf{w}^0 = \mathbf{b} - \mathbf{A}\mathbf{x}^0$ and $\mathbf{A}$

$$V^{k+1}(\mathbf{w}^0, \mathbf{A}) = sp\{\mathbf{w}^0, \mathbf{A}\mathbf{w}^0, \mathbf{A}^2\mathbf{w}^0, \ldots, \mathbf{A}^k\mathbf{w}^0\}; \quad \mathbf{w}^0 = \mathbf{b} - \mathbf{A}\mathbf{x}^0. \tag{23}$$

Note that the residual is denoted here by $\mathbf{w}$ rather than the usual $\mathbf{r}$ for compatibility with the notation used in Section 2.2 and previous papers discussing the FETI method.

The CG method applied to the solution of problem (21) determines η^k by minimizing the $\mathbf{A}$-norm of the error $\mathbf{x}^{\mathrm{ex}} - \mathbf{x}^{k+1}$, for $\mathbf{p}^k \in V^{k+1}(\mathbf{w}^0, \mathbf{A})$ [38].

Consider now a modified CG algorithm where the search direction $\tilde{\mathbf{p}}^k$ is not selected as an element of $V^{k+1}(\mathbf{w}^0, \mathbf{A})$, but an element of the hybrid space

$$\tilde{V}^{k+1}(\mathbf{w}^0, \mathbf{A}, \mathbf{C}) = V^{k+1}(\tilde{\mathbf{w}}^0, \tilde{\mathbf{A}}) + \mathrm{Range}(\mathbf{C}) \tag{24}$$

where $\mathbf{C} \in \mathbb{R}^{n\times m}$ is a given rectangular matrix with full column rank equal to m, and $\tilde{\mathbf{w}}^0$ and $\tilde{\mathbf{A}}$ are two quantities that are related to $\mathbf{w}^0$ and $\mathbf{A}$, respectively, and will be determined later. Hence, $\tilde{\mathbf{p}}^k$ can be written as

$$\tilde{\mathbf{p}}^k = \mathbf{p}^k + \mathbf{C}\gamma^k; \quad \mathbf{p}^k \in V^{k+1}(\tilde{\mathbf{w}}^0, \tilde{\mathbf{A}}),\ \gamma^k \in \mathbb{R}^m. \tag{25}$$

Such a modified CG algorithm generates a sequence of approximations to $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ that can be written as

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \eta^k\tilde{\mathbf{p}}^k = \mathbf{x}^k + \eta^k(\mathbf{p}^k + \mathbf{C}\gamma^k), \tag{26}$$

where η^k and γ^k are determined by minimizing the two- parameter functional

$$\Phi(\eta,\gamma) = \|\mathbf{x}^{\mathrm{ex}} - \mathbf{x}^k - \eta\tilde{\mathbf{p}}^k\|_{\mathbf{A}} = \|\mathbf{x}^{\mathrm{ex}} - \mathbf{x}^k - \eta(\mathbf{p}^k + \mathbf{C}\gamma)\|_{\mathbf{A}}, \tag{27}$$

which requires solving the two equations

$$\frac{\partial\Phi}{\partial\eta}(\eta^k,\gamma^k) = 0 \quad \frac{\partial\Phi}{\partial\gamma}(\eta^k,\gamma^k) = 0. \tag{28}$$

Substituting Eq. (27) into (28) leads to

$$\begin{array}{l}(\mathbf{p}^k + \mathbf{C}\gamma^k)^{\mathrm{T}}(\mathbf{b} - \mathbf{A}(\mathbf{x}^k + \eta^k(\mathbf{p}^k + \mathbf{C}\gamma^k))) = 0\\ \mathbf{C}^{\mathrm{T}}(\mathbf{b} - \mathbf{A}(\mathbf{x}^k + \eta^k(\mathbf{p} + \mathbf{C}\gamma^k))) = 0\end{array} \tag{29}$$

and using Eqs. (26) and (29) can be transformed into

$$\tilde{\mathbf{p}}^{k\mathrm{T}}\mathbf{w}^{k+1} = 0, \qquad \mathbf{C}^{\mathrm{T}}\mathbf{w}^{k+1} = 0. \tag{30}$$

Two observations are worth noting

1. The relationship between the search direction $\tilde{\mathbf{p}}^k$ and the residual $\mathbf{w}^{k+1}$ expressed in the first of Eqs. (30) is a mathematical property of the CG method [39],
2. The second of Eqs. (30) is identical to Eq. (20), which shows that the "optional admissible constraint" principle that was first proposed in [17] then refined in [34] for designing the two-level FETI method is a consequence of enlarging a Krylov space by Range ($\mathbf{C}$).

It remains to explicitly determine the coefficient η^k and the vector γ^k to completely define our modified CG algorithm.

Since Eqs. (30) must hold at each iteration k, it follows that

$$\mathbf{C}^{\mathrm{T}}\mathbf{w}^k = \mathbf{C}^{\mathrm{T}}(\mathbf{b} - \mathbf{A}\mathbf{x}^k) = 0 \tag{31}$$

and therefore the second of Eqs. (29) simplifies to

$$(\mathbf{C}^{\mathrm{T}}\mathbf{A}\mathbf{C})\gamma^k = -\mathbf{C}^{\mathrm{T}}\mathbf{A}\mathbf{p}^k. \tag{32}$$

Solving Eq. (32) and substituting the solution γ^k into Eq. (25) gives

$$\tilde{\mathbf{p}}^k = \mathbf{p}^k + \mathbf{C}\gamma^k = \mathbf{P}_C\mathbf{p}^k, \quad \text{where} \quad \mathbf{P}_C = \mathbf{I} - \mathbf{C}(\mathbf{C}^{\mathrm{T}}\mathbf{A}\mathbf{C})^{-1}\mathbf{C}^{\mathrm{T}}\mathbf{A}. \tag{33}$$

From the first of Eqs. (33), it follows that

$$\tilde{\mathbf{A}} = \mathbf{P}_C\mathbf{A}\mathbf{P}_C. \tag{34}$$

After γ^k is computed from the solution of problem (32), $\tilde{\mathbf{p}}^k$ can be evaluated, and η^k can be computed from the first of Eqs. (29). Interestingly, one obtains

$$\eta^k = \frac{\tilde{\mathbf{p}}^{k\mathrm{T}}(\mathbf{p} - \mathbf{A}\mathbf{x}^k)}{\tilde{\mathbf{p}}^{k\mathrm{T}}\mathbf{A}\tilde{\mathbf{p}}^k} = \frac{\tilde{\mathbf{p}}^{k\mathrm{T}}\mathbf{w}^k}{\tilde{\mathbf{p}}^{k\mathrm{T}}\mathbf{A}\tilde{\mathbf{p}}^k}, \tag{35}$$

which shows that η^k has the same expression as in any CG method that employs $\tilde{\mathbf{p}}^k$ as a search direction.

Hence, the modified CG algorithm presented herein differs from the CG method only in the selection of the starting value and the construction of the search directions. Instead of starting with $\mathbf{x}^0$, it starts with $\mathbf{x}^0 + \mathbf{C}\gamma^{\mathrm{init}}$, where γ^{init} is determined so that Eq. (31) holds for $k = 0$

$$\mathbf{C}^{\mathrm{T}}(\mathbf{b}-\mathbf{A}(\mathbf{x}^0+\mathbf{C}\gamma^{\mathrm{init}}))=0$$

$$\Rightarrow \tilde{\mathbf{x}}^0=\mathbf{x}^0+\mathbf{C}\gamma^{\mathrm{init}}=\mathbf{x}^0+\mathbf{C}(\mathbf{C}^{\mathrm{T}}\mathbf{A}\mathbf{C})^{-1}\mathbf{C}^{\mathrm{T}}(\mathbf{b}-\mathbf{A}\mathbf{x}^0)$$

$$\Rightarrow \tilde{\mathbf{w}}^0=\mathbf{b}-\mathbf{A}(\mathbf{x}^0+\mathbf{C}\gamma^{\mathrm{init}})=\mathbf{P}_C^{\mathrm{T}}(\mathbf{b}-\mathbf{A}\mathbf{x}^0)=\mathbf{P}_C^{\mathrm{T}}\mathbf{w}^0. \tag{36}$$

At each iteration k, rather than selecting a search direction $\mathbf{p}^k$ from the standard Krylov space $V^{k+1}(\mathbf{w}^0,\mathbf{A})$, it constructs a search direction $\tilde{\mathbf{p}}^k$ in the hybrid space $V^{k+1}(\mathbf{P}_C^{\mathrm{T}}\mathbf{w}^0,\mathbf{P}_C\mathbf{A}\mathbf{P}_C)+\mathrm{Range}(\mathbf{C})$, where $\mathbf{C}$ is a given matrix. Consequently, this modified CG algorithm requires solving at each iteration the auxiliary problem (32). If m is kept small, this problem becomes a "coarse" problem that is relatively inexpensive to setup and solve. Furthermore, because it is governed by the matrix $\mathbf{C}^{\mathrm{T}}\mathbf{A}\mathbf{C}$, this auxiliary coarse problem is a "second-level" problem, and therefore our modified CG algorithm is a genuine two-level method.

In Appendix A, it is proved that the search directions $\tilde{\mathbf{p}}^k$ defined in (25) are $\mathbf{A}$-orthogonal. From this result and Eqs. (28) and (30), we conclude that the two-level gradient method described in this section is a two-level CG method. It is also proved in Appendix A that this two- level CG method is equivalent to a preconditioned CG method where $\mathbf{P}_C$ is the preconditioner, and that the theory presented in [40] can be applied to establish

$$\kappa(\mathbf{P}_C^{\mathrm{T}}\mathbf{A}\mathbf{P}_C)\leqslant\kappa(\mathbf{A}) \tag{37}$$

where κ denotes the condition number of a matrix. This result proves that the two-level CG method described here can only converge faster than the standard CG algorithm.

3.2. Application to the FETI method

The two-level FETI method is obtained by substituting $\mathbf{A}=\mathbf{P}^{\mathrm{T}}\mathbf{F}_{\mathrm{I}}\mathbf{P}$ in Eq. (32), and applying the modifications to a CG algorithm described in Section 3.1 to the one-level FETI PCPG solver.

In the initialization step, $\bar{\lambda}^0=0$ is enriched by the quantity $\mathbf{C}\gamma^{\mathrm{init}}$ so that λ^0 becomes

$$\lambda^0=\mathbf{Q}\mathbf{G}_{\mathrm{I}}(\mathbf{G}_{\mathrm{I}}^{\mathrm{T}}\mathbf{Q}\mathbf{G}_{\mathrm{I}})^{-1}\mathbf{e}+\mathbf{P}\mathbf{C}\gamma^{\mathrm{init}}. \tag{38}$$

Using the second of Eqs. (13), the reader can check that the above starting value satisfies the necessary condition $\mathbf{G}_{\mathrm{I}}^{\mathrm{T}}\lambda^0=\mathbf{e}$. The value of γ^{init} is obtained by minimizing $\|\bar{\lambda}^{\mathrm{ex}}-\mathbf{C}\gamma^{\mathrm{init}}\|_{\mathbf{P}^{\mathrm{T}}\mathbf{F}_{\mathrm{I}}\mathbf{P}}$ which leads to

$$(\mathbf{C}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\mathbf{F}_{\mathrm{I}}\mathbf{P}\mathbf{C})\gamma^{\mathrm{init}}=\mathbf{C}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}(\mathbf{d}-\mathbf{F}_{\mathrm{I}}\mathbf{Q}\mathbf{G}_{\mathrm{I}}(\mathbf{G}_{\mathrm{I}}^{\mathrm{T}}\mathbf{Q}\mathbf{G}_{\mathrm{I}})^{-1}\mathbf{e}). \tag{39}$$

At each iteration k the right-hand side of Eq. (32) applied to the one-level FETI method is equal to $-\mathbf{C}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\mathbf{F}_{\mathrm{I}}\mathbf{P}\mathbf{p}k$. From Box 1, it follows that $\mathbf{p}^k$ is generated by $\mathbf{y}^k=\mathbf{P}\tilde{\mathbf{F}}_{\mathrm{I}}\mathbf{w}^k$. Hence, using the first of Eqs. (13), we conclude that

$$-\mathbf{C}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\mathbf{F}_{\mathrm{I}}\mathbf{P}\mathbf{p}^k=-\mathbf{C}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\mathbf{F}_{I}\mathbf{p}^k. \tag{40}$$

The above remarks lead to a formulation of the two-level FETI solver that is summarized in Box 2. This formulation is simpler than the previous one presented in [17], but is both mathematically and numerically equivalent to it. The underlined steps are those steps which distinguish the two-level FETI solver from its one-level counterpart. As in the case of the one-level FETI solver and for the same reasons highlighted in [10], a full orthogonalization procedure is applied to the search directions for maximum overall computational efficiency. This reorthogonalization is applied to the vectors $\mathbf{p}^k\in V^{k+1}(\mathbf{P}_C^{\mathrm{T}}\mathbf{w}^0,\mathbf{P}_C\mathbf{A}\mathbf{P}_C)$ rather that the search directions $\tilde{\mathbf{p}}^k\in\tilde{V}^{k+1}(\mathbf{w}^0,\mathbf{A},\mathbf{C})$ in order to keep the presentation of the two-level FETI solver as similar to that of its one-level counterpart as possible. However, from the expression of $\mathbf{p}^k$ given in Box 2 and Eq. (32) with $\mathbf{A}=\mathbf{P}\mathbf{F}_{\mathrm{I}}\mathbf{P}$, it follows that for all $j<k$

$$\begin{aligned}\tilde{\mathbf{p}}^{k^{\mathrm{T}}}(\mathbf{P}^{\mathrm{T}}\mathbf{F}_{\mathrm{I}}\mathbf{P})\tilde{\mathbf{p}}^{j} &= (\mathbf{p}^{k^{\mathrm{T}}}+\gamma^{k^{\mathrm{T}}}\mathbf{C}^{\mathrm{T}})(\mathbf{P}^{\mathrm{T}}\mathbf{F}_{\mathrm{I}}\mathbf{P})\tilde{\mathbf{p}}^{j}\\ &= \mathbf{p}^{k^{\mathrm{T}}}(\mathbf{P}^{\mathrm{T}}\mathbf{F}_{\mathrm{I}}\mathbf{P})\tilde{\mathbf{p}}^{j}+\gamma^{k^{\mathrm{T}}}\mathbf{C}^{\mathrm{T}}(\mathbf{P}^{\mathrm{T}}\mathbf{F}_{\mathrm{I}}\mathbf{P})\tilde{\mathbf{p}}^{j}\\ &= 0+\gamma^{k^{\mathrm{T}}}\mathbf{C}^{\mathrm{T}}(\mathbf{P}^{\mathrm{T}}\mathbf{F}_{\mathrm{I}}\mathbf{P})(\mathbf{p}^{j}+\mathbf{C}\gamma^{j})\\ &= \gamma^{k^{\mathrm{T}}}[\mathbf{C}^{\mathrm{T}}(\mathbf{P}^{\mathrm{T}}\mathbf{F}_{\mathrm{I}}\mathbf{P})\mathbf{C}\gamma^{j}+\mathbf{C}^{\mathrm{T}}(\mathbf{P}^{\mathrm{T}}\mathbf{F}_{\mathrm{I}}\mathbf{P})\mathbf{p}^{j}]\\ &= 0,\end{aligned} \tag{41}$$

which shows that an explicit full $\mathbf{PF_IP}$-orthogonalization of the vectors $\mathbf{p}^k$ implies an explicit full $\mathbf{PF_IP}$-orthogonalization of the search directions $\tilde{\mathbf{p}}^k$.

Box 2. The two-level FETI PCPG method.

1. Initialize

$$\begin{aligned}\underline{\mathrm{Solve}(\mathbf{C}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\mathbf{F}_I\mathbf{P}\mathbf{C})\gamma^{\mathrm{init}}} &= \underline{\mathbf{C}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}(\mathbf{d}-\mathbf{F}_I\mathbf{Q}\mathbf{G}_I(\mathbf{G}_I^{\mathrm{T}}\mathbf{Q}\mathbf{G}_I)^{-1}\mathbf{e})}\\ \lambda^0 &= \mathbf{Q}\mathbf{G}_I(\mathbf{G}_I^{\mathrm{T}}\mathbf{Q}\mathbf{G}_I)^{-1}\mathbf{e}+\underline{\mathbf{P}\mathbf{C}\gamma^{\mathrm{init}}}\\ \mathbf{w}^0 &= \mathbf{P}^{\mathrm{T}}(\mathbf{d}-\mathbf{F}_I\lambda^0)\end{aligned}$$

2. Iterate $k=0,1,\ldots$ until convergence

$$\begin{aligned}\mathbf{y}^k &= \mathbf{P}\tilde{\mathbf{F}}_I^{-1}\mathbf{w}^k\\ \mathbf{p}^k &= \mathbf{y}^k-\sum_{i=0}^{k-1}\frac{\mathbf{y}^{k^{\mathrm{T}}}\mathbf{F}_I\tilde{\mathbf{p}}^i}{\tilde{\mathbf{p}}^{i^{\mathrm{T}}}\mathbf{F}_I\tilde{\mathbf{p}}^i}\tilde{\mathbf{p}}^i\\ \underline{\mathrm{Solve}(\mathbf{C}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\mathbf{F}_I\mathbf{P}\mathbf{C})\gamma^k} &= \underline{-\mathbf{C}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\mathbf{F}_I\mathbf{p}^k}\\ \underline{\tilde{\mathbf{p}}^k} &= \underline{\mathbf{p}^k+\mathbf{P}\mathbf{C}\gamma^k}\\ \eta^k &= \frac{\tilde{\mathbf{p}}^{k^{\mathrm{T}}}\mathbf{w}^k}{\tilde{\mathbf{p}}^{k^{\mathrm{T}}}\mathbf{F}_I\tilde{\mathbf{p}}^k}\\ \lambda^{k+1} &= \lambda^k+\eta^k\tilde{\mathbf{p}}^k\\ \mathbf{w}^{k+1} &= \mathbf{w}^k-\eta^k\mathbf{P}^{\mathrm{T}}\mathbf{F}_I\tilde{\mathbf{p}}^k\end{aligned} \tag{42}$$

3.3. How to choose $\mathbf{C}$

The conditioning result (37) suggests that , in general, the two-level FETI solver will converge faster than its one-level counterpart. The amount of acceleration provided by the second-level auxiliary problem depends however on the choice of the matrix $\mathbf{C}$. In order to address this issue, we present first a physical interpretation of the crust of the two-level FETI method.

The enrichment of the Krylov spaces $V^{k+1}(\mathbf{P}_C^{\mathrm{T}}\mathbf{w}^0, \mathbf{P}_C\mathbf{A}\mathbf{P}_C)$ by Range ($\mathbf{C}$) leads to Eqs. (30) that govern the two-level FETI solver. As stated earlier, the first of Eqs. (30) characterizes the CG method. On the other hand, the second of Eqs. (30) states that at each iteration k, the residual $\mathbf{w}^k=\mathbf{P}^{\mathrm{T}}(\mathbf{d}-\mathbf{F}_I\lambda^k)$ is forced to be orthogonal to Range ($\mathbf{C}$). In order to understand what $\mathbf{w}^k$ physically represents, we evaluate the jump $\boldsymbol{\Delta}^k=\sum_{s=1}^{s=N_s}\mathbf{B}^{(s)}\mathbf{u}^{(s)}$ of the kth approximate solution of the focus problem (2) across the subdomain interfaces. Using Eq. (6) and the notation of Eqs. (9), we obtain

$$\boldsymbol{\Delta}^k=\sum_{s=1}^{s=N_s}\mathbf{B}^{(s)}\mathbf{u}^{(s)}=\mathbf{d}-\mathbf{F}_I\lambda^k+\mathbf{G}_I\alpha^k. \tag{43}$$

The vector α^k can be extracted from the first of Eqs. (8) as

$$\alpha^k = -(\mathbf{G}_I^{\mathrm{T}}\mathbf{Q}\mathbf{G}_I)^{-1}\mathbf{G}_I^{\mathrm{T}}\mathbf{Q}(\mathbf{d} - \mathbf{F}_I\lambda^k). \tag{44}$$

Substituting Eq. (44) into Eq. (43) gives

$$\Delta^k = \left(\mathbf{I} - \mathbf{G}_I(\mathbf{G}_I^{\mathrm{T}}\mathbf{Q}\mathbf{G}_I)^{-1}\mathbf{G}_I^{\mathrm{T}}\mathbf{Q}\right)(\mathbf{d} - \mathbf{F}_I\lambda^k), \tag{45}$$

which in view of Eq. (12) can also be written as

$$\Delta^k = \mathbf{P}^{\mathrm{T}}(\mathbf{d} - \mathbf{F}_I\lambda^k) = \mathbf{w}^k. \tag{46}$$

Hence, $\mathbf{w}^k$ represents the jump across the subdomain interfaces of the approximate subdomain solutions $\mathbf{u}^{(s)^k}$. At each FETI iteration $k, \mathbf{u}^{(s)^k}$ is evaluated by substituting in Eq. (5) the Lagrange multipliers λ^k generated by the PCPG algorithm. Hence, at each iteration k, the subdomains are in equilibrium, but the jump $\mathbf{w}^k$ of the subdomain displacement iterates across the subdomain interfaces is not necessarily zero. Only at convergence, this jump vanishes in the usual numerical sense.

It follows that the constraint $\mathbf{C}^{\mathrm{T}}\mathbf{w}^k = 0$ accelerates convergence by forcing at each iteration k some prescribed components, or prescribed linear combinations of some components, of the subdomain displacement solutions to be continuous across the subdomain interfaces.

For example, for elastodynamics problems, it was proposed in [21] to choose $\mathbf{C} = \mathbf{G}_I$, with $\mathbf{R}^{(s)}$ defined as the rigid body modes obtained for subdomain $\Omega^{(s)}$ when any prescribed Dirichlet boundary condition is ignored. Such a choice for $\mathbf{C}$ is equivalent to forcing at each iteration k the constant components of $\mathbf{u}^{(s)^k}$ to be continuous across the subdomain interfaces. It was also shown in [21] that the resulting two-level FETI method is numerically scalable for transient dynamics problems.

For plate and shell problems, it was proposed in [17,18] to enforce the continuity of the transverse displacement field at the subdomain crosspoints throughout the FETI PCPG iterations. Such an objective can be achieved by choosing $\mathbf{C}$ as the matrix of "corner" modes – that is, by setting $\mathbf{C}_{ij} = 1$ for each transverse displacement d.o.f. located at the jth crosspoint, and $\mathbf{C}_{ij} = 0$ elsewhere. In [41], it was proved that the resulting two-level FETI method is numerically scalable with respect to both the mesh size h and subdomain size H (see Eq. (2)).

Other choices for $\mathbf{C}$ have also been discussed in [34] for structural mechanics problems, and [25,26] for acoustic scattering problems. So far, intuition and physical insight into the problem to be solved have been the major sources of candidate $\mathbf{C}$ matrices. However, it has been shown in [34] that the most effective choices for $\mathbf{C}$ have been so far those for which a mathematical analysis of the condition number has been possible.

4. Which FETI method for which problem?

The two-level FETI method increases the computational complexity of the one-level FETI method by an amount that depends on the choice and size of the $\mathbf{C}$ matrix. However, depending on the problem to be solved, the computational overhead associated with the setup and repeated solutions of the auxiliary second-level problem can be offset by a dramatic improvement of the convergence rate of the FETI PCPG algorithm. Based on our experience, given a target problem, we recommend to use the FETI method that has been proved to be numerically scalable for the family of problems to which the target problem belongs. This means that for second-order elasticity problems (plane stress/strain and solid elements), we recommend the one-level FETI method. For such problems, the one-level FETI solver converges so fast that any additional improvement of convergence by means of an auxiliary second-level problem may not reduce the solution CPU time. Furthermore, for second-order elasticity problems, we also recommend equipping the one-level FETI method with the lumped preconditioner when a mesh partition with a relatively small number of subdomains N_s is generated, and the Dirichlet preconditioner when N_s is relatively large. On the other hand, for fourth-order elasticity problems such as beam, plate, and shell problems, we recommend using the two-level FETI method equipped with the Dirichlet preconditioner, and with $\mathbf{C}$ chosen as the matrix of corner modes. In [42], the authors have shown that for such problems, the two-level FETI method

is not only numerically scalable, but outperforms the one-level FETI solver CPU wise by a factor that is typically larger than two. For second-order elastodynamic problems, we recommend the two-level FETI method with $\mathbf{C} = \mathbf{G}_I$, and for the fourth-order elastodynamic problems, we recommend the two-level FETI method equipped with both the subdomain corner and rigid body modes.

5. Critical components of a FETI solver

The robustness and performance of the one-level and two-level FETI solvers depend on a set of implementational issues that are raised in this section, and addressed in Section 6. These issues determine the critical components of a FETI solver.

5.1. Choice of the local solver

A subdomain solver is required by the FETI methodology for performing two tasks at each iteration k: evaluating the subdomain displacement fields $\mathbf{u}^{(s)^k}$ associated with each approximate solution λ^k generated by the PCG algorithm (see Eq. (5)), and computing the preconditioned residual $\tilde{\mathbf{F}}_I^{-1}\mathbf{w}^k$ when the Dirichlet preconditioner is selected (see Eq. (19)). We refer to such a solver as the "local" solver.

Choosing an iterative method as a local solver can be attractive in theory, as it reduces the storage requirements of the FETI methods and opens research themes in the area of inexact subdomain solvers. However, an iterative local solver increases the complexity of the robustness issue as it introduces an additional convergence criterion in the FETI method at the subdomain level. Furthermore, the local solver is embedded in an iterative loop in the Lagrange multipliers λ^k, and therefore computational efficiency requires optimizing an iterative local solver for successive right-hand sides. While such an optimization is possible (for example, see [22]), it requires additional storage that offsets the main advantage of an iterative local solver.

On the other hand, choosing a direct method as a local solver reduces the number of robustness issues that must be dealt with when developing a FETI software, and ensures a better overall computational efficiency of a FETI solver. Indeed, since the one-level and two-level FETI methods are numerically scalable with respect to the number of subdomains N_s, the memory requirement of a local direct solver can be significantly reduced by increasing the number of subdomains in a mesh partition, because increasing N_s decreases the size of the subdomain stiffness matrices $\mathbf{K}^{(s)}$. Furthermore, when N_s is chosen sufficiently large, the size of the subdomain problems falls in a range where an iterative solver cannot compete in speed with a direct one.

Both skyline and sparse direct solvers are excellent local solver candidates. Their relative merits are discussed in Section 6.1.

5.2. Computation of the null spaces and generalized inverses

The extraction of the null spaces of the subdomain stiffness matrices can become an Achille's heel for the FETI method. If for some reason, the wrong dimension of ker $\mathbf{K}^{(s)}$ – that is, the wrong number of subdomain zero energy modes – is predicted, or a wrong representation of any subdomain zero energy mode is computed, the solvability condition (7) will be violated, the projector $\mathbf{P}$ (12) will be inadequate, and the FETI solver will fail to converge to the correct solution of a given static problem. Similar effects happen when a wrong generalized inverse $\mathbf{K}^{(s)^+}$ of a singular subdomain stiffness matrix is computed. For transient dynamic or time-dependent problems, the subdomain problems are not governed by a solvability condition, and therefore the correct extraction of the subdomain rigid body modes is less critical as it is used only for preconditioning purposes [21].

The first generation of FETI solvers has employed the factorization based procedure described in [13] for computing the null space and generalized inverse of the stiffness matrix of a floating subdomain. This procedure was motivated by the choice of a direct (skyline) method as a local solver. It exploits the fact that if l denotes the dimension of ker $\mathbf{K}^{(s)}$, then l zero pivots will appear during the factorization of $\mathbf{K}^{(s)}$. Each encountered zero pivot corresponds to a redundant equation that can be removed from the system without

interrupting the factorization process. At the end of this process, the factored columns and rows minus the removed ones correspond to the factored form of the simplest generalized inverse $\mathbf{K}^{(s)^+}$ one can construct, and a basis of ker $\mathbf{K}^{(s)}$ can be recovered by performing l forward and backward substitutions on l canonical vectors associated with the l zero pivots.

It has often been speculated and argued that the subdomain rigid body modes computed by the procedure summarized above can be sufficiently tainted by the round-off accumulated during the factorization of the subdomain stiffness matrices to ruin the convergence of the FETI method or prevent it from producing an accurate solution of the global problem. We disagree with these speculations. We firmly believe that if a subdomain stiffness matrix $\mathbf{K}^{(s)}$ is sufficiently ill-conditioned to prevent l forward and backward substitutions from being accurate, then the global stiffness matrix $\mathbf{K}$ must be even more ill-conditioned and should forbid an accurate solution of the global problem in the first place. As a matter of fact, we have designed a large number of experiments including nearly incompressible elasticity problems and have verified that the FETI method equipped with the procedure described above for extracting ker $\mathbf{K}^{(s)}$ and computing $\mathbf{K}^{(s)^+}$ could always deliver an accurate solution of the global problem and as efficiently as when it was equipped with a more sophisticated singularity handling technique that was devised later [43].

However, we recognize and have already pointed out in [43] that the factorization based technique developed in [13] and summarized above for handling singular subdomain stiffness matrices is not sufficiently robust. In practice, "small" rather than zero pivots are encountered when decomposing a singular subdomain stiffness matrix, and these small pivots are usually monitored by a criterion of the form

$$|\tilde{k}_{ii}| < \epsilon \times \max_{1 \leqslant j \leqslant n_s} \{|k_{jj}|\} \tag{47}$$

or some more sophisticated variant of the above criterion (see [43] for further details). Here, n_s denotes the size of the subdomain stiffness matrix $\mathbf{K}^{(s)} = (k_{ii})$, $\tilde{k}_{ii}$ is the ith pivot obtained after the ith factorization step has been applied to $\mathbf{K}^{(s)}$, and ϵ is an arbitrarily small constant. The ideal value of ϵ is that which allows criterion (47) to distinguish between the small pivots due to singularity, and those due to ill-conditioning. Unfortunately, for highly ill-conditioned problems, this ideal value of ε is hard if not impossible to predict without a trial and error procedure. If the selected value of the arbitrary constant ϵ turns out to be too small or too large, a smaller or larger null space of $\mathbf{K}^{(s)}$ is erroneously computed, which as stated earlier, will cause a FETI solver to fail. For this reason, alternative procedures for handling singular subdomain stiffness matrices have been proposed in [43]. Recently, we have improved one of these procedures and made it as fail-proof as possible. We report on this development in Sections 6.2 and 6.3.

5.3. Choice of the coarse problem solver

A key component of the FETI methodology is the auxiliary coarse problem that must be solved at each PCG iteration k. For the one-level FETI method, this problem is of the form

$$(\mathbf{G}_I^T \mathbf{Q} \mathbf{G}_I)\mathbf{x}^k = \mathbf{b}^k, \tag{48}$$

where $\mathbf{x}^k$ and $\mathbf{b}^k$ denote two generic left-hand side and right-hand side vectors, respectively. The two-level FETI method incorporates two coarse problems: the one described above, and a second auxiliary problem of the form

$$(\mathbf{C}^T \mathbf{P}^T \mathbf{F}_I \mathbf{P} \mathbf{C})\mathbf{x}^k = \mathbf{b}^k, \tag{49}$$

where $\mathbf{P}$ is the projector defined in Eq. (12). Both coarse problems (48) and (49) must be solved at each iteration of the two-level FETI method.

In [18,21], it was advocated to solve both coarse problems (48) and (49) by a CG algorithm tailored for the solution of problems with repeated right-hand sides, for the following reasons

1. the CG method requires only matrix–vector products, and for both coarse problems (48) and (49), these matrix–vector products incur only subdomain level computations that require the same local data structures as the remainder of the FETI computations,
2. for both coarse problem (48) and (49), these matrix–vector products can be performed at the subdomain level with short range communication occurring only between neighboring subdomains.

In other words, solving the coarse problems by the CG method simplifies software development, and ensures parallel scalability. However, unlike direct methods, the standard CG algorithm is not suitable for the solution of repeated problems with a constant left-hand side, and variable right-hand sides. For this reason, the authors of [18,21] have advocated the solution of the FETI coarse problems by a CG algorithm that was tailored in [22,44] for the solution of repeated problems, and which is further discussed in Section 7.3 of the present paper. The most important aspect of this algorithm is that it solves the first encountered coarse problem of size n_c in exactly n_c iterations, and every other coarse problem arising at each subsequent FETI PCPG iteration in a single iteration. Note that n_c is directly proportional to the number of subdomains N_s. For small mesh partitions that lead to small values of n_c, say $n_c \leqslant 600$, it was shown that such a strategy is computationally efficient and allows both one-level and two-level FETI solvers to achieve parallel scalability [18,21]. For fine mesh partitions, say $N_s \geqslant 200$, and especially for the two-level FETI method, the size of the coarse problem n_c becomes typically too large – for example, $n_c = 2000$ – to keep the cost of the n_c iterations performed during the solution of the first encountered coarse problem affordable. For this reason, and because efficient preconditioners for the FETI coarse problems have not yet been developed, we have recently switched to a direct coarse solver that requires more intricate data structures, but which is more computationally efficient when the number of subdomains is rather large. This direct solver is discussed in Section 6.4.

5.4. Influence of the number of subdomains

An important parameter of any DD method is the number of subdomains N_s. Given a problem of a certain size, into how many subdomains should it be decomposed? The answer to this question is not simple, as it depends on the numerical properties of the DD method, its computational complexity and memory requirements, and on the computational platform on which it is implemented.

In most DD work that has emphasized parallel computing, the number of sub-domains N_s has almost always been set to the number of processors N_p used for solving the given problem. This is because assigning one subdomain to one processor simplifies software development, especially when the target parallel processor has a distributed memory. Unfortunately, this strategy is not optimal because it does not take into account other considerations such as memory requirements, convergence rate, and operation count.

An important feature of the one-level and two-level FETI methods is their numerical scalability with respect to the number of subdomains. This feature is demonstrated by the optimal conditioning result (1). Hence, both FETI methods operator well with large numbers of subdomains. Whether a problem size is fixed or varied, there are multiple incentives for increasing N_s

1. it increases the degree of natural parallelism of the algorithm – that is, the number of computations that can be performed at the subdomain level and that necessitate interprocessor communication only between neighboring subdomains,
2. it reduces the memory requirements of the subdomain stiffness matrices and their factors. It also reduces the CPU time required for computing these factors and performing all subsequent local forward and backward substitutions.

However, it should also be pointed out that increasing the number of subdomains increases the sizes of the interface and auxiliary coarse problems. This in turn increases the CPU and memory cost of the FETI PCG algorithm as it inflates the CPU and memory overhead associated with reorthogonalizing the search directions and solving the coarse problems.

Hence, when using a FETI method, the number of subdomains should be chosen as to strike a balance between all the issues raised above. Achieving such a balance requires experience. Most importantly, it also requires an implementation of the one-level and two-level FETI solvers that allows assigning multiple subdomains to a processor [45]. Such an implementation can be challenging, especially for distributed memory parallel processors. However, it is worth the effort, as demonstrated by the following static analysis of a diffraction grating system using the one-level FETI method.

The finite element mesh shown in Fig. 1 is that of a diffraction grating system that is part of a satellite borne telescope spectrograph. It contains 35,328 8-noded brick elements and 120,987 d.o.f. The grating material of this structure is fused silica. When mounted, it must have face surface deflections below the micron level in 1G acceleration for accurate pre-launch alignment with the rest of the spectrograph. It must

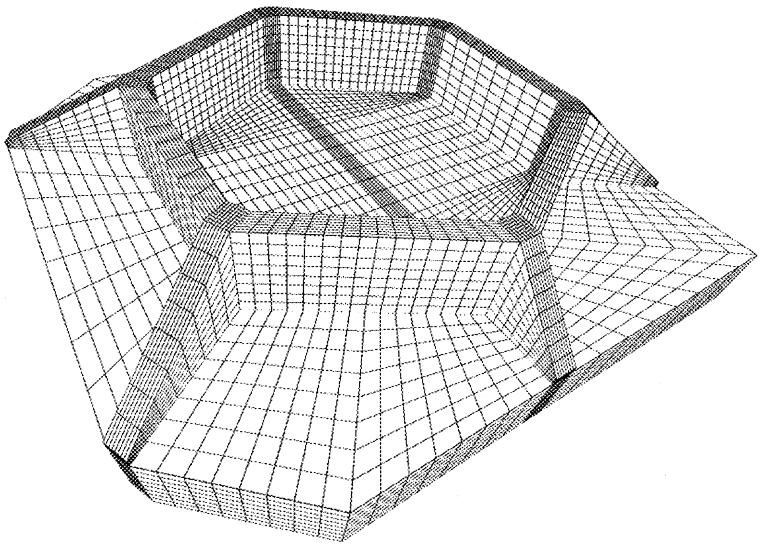

Fig. 1. Finite element discretization of a diffraction grating system.

also be able to withstand accelerations up to 15G laterally and axially during launch. Several designs for this system have been conceived at the University of Colorado [46], and analyzed by the FETI method. Here, we report on the performance results obtained on a single processor Origin 2000 Silicon graphics system, for one configuration but different numbers of subdomains. In all cases, we use the one-level FETI solver equipped with the lumped preconditioner and set the convergence criterion to

$$\|\mathbf{Ku}-\mathbf{f}\|_2 \leqslant 10^{-6} \times \|\mathbf{f}\|_2. \tag{50}$$

We report in Fig. 2 the convergence results, CPU timings associated with the different phases of the FETI method as well as the total solution CPU time, and total memory consumption obtained for this problem. For $42 \leqslant N_s \leqslant 142$, the number of iterations for convergence is shown to vary between 56 and 88, with 56 iterations for the case $N_s = 128$. When N_s is increased from 42 to 142, the CPU time consumed in the local factorizations decreases as expected from 76.7 to 25.1 s, but the CPU time elapsed in forming and solving the coarse problems increases also as expected from 8.0 to 27.3 s, and that elapsed in

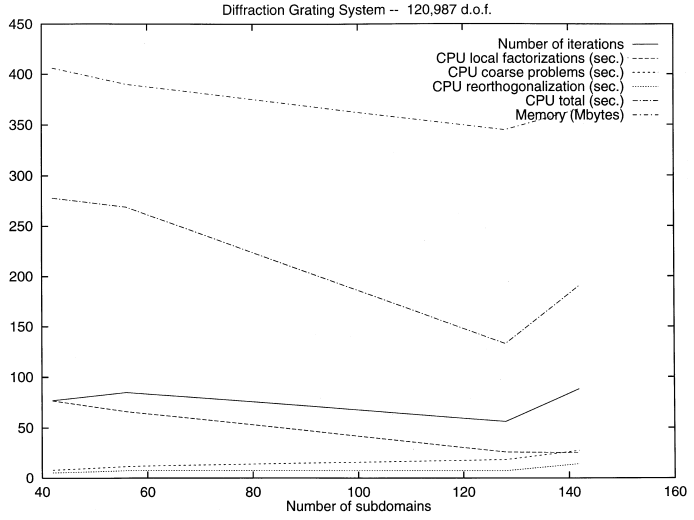

Fig. 2. Performance results on a single processor Origin 2000.

reorthogonalizing the search directions increases from 5.3 to 14.2 s. The total CPU time decreases from 278.0 to 132.9 s when N_s is increased from 42 to 128, then increases to 190.2 s when N_s is increased to 142. Hence, for this problem, the optimal number of subdomains is $N_s = 128$. The total memory consumption decreases from 406 to 345 Mb when N_s is increased from 42 to 128, then increases to 365 Mb when N_s is increased to 142. This means that for $42 \leqslant N_s \leqslant 128$, the memory requirements of the FETI solver are dominated by those of the local problems, and for $N_s \geqslant 128$ they become dominated by the memory requirements of the reorthogonalization procedure, which increase with the size of the interface problem.

In summary, the above example shows that for a given problem, the CPU time of the FETI method decreases when the number of subdomains is increased, but only until a critical value of N_s is reached; beyond that value, the CPU time of the FETI method begins to increase with N_s. More importantly, this example shows that the FETI solution CPU time can vary by as much as 200% when the number of subdomains is varied, which demonstrates the importance of choosing the optimal value of N_s independently from the number of processors N_p available on a given machine. Unfortunately, that was not possible in the early parallel implementations of the one-level and two-level FETI methods that were restricted to assigning one subdomain to one processor. Hence, most if not all previously published performance results of the FETI method – at least by this group of researchers – can be dramatically improved by a new implementation that separates the concept of a subdomain from that of a processor. Such a new implementation is overviewed in Section 6.5.

6. The second generation FETI methods

The second generation of FETI solvers addresses the issues raised in the previous section.

6.1. Local solvers

Our preference for a direct local solver has been discussed in Section 5.1. So far, we have focused in our FETI work on skyline direct local solvers because they have proved to be easier to write, optimize for a given computer, modify and maintain, than their sparse counterparts. However, sparse direct solvers have recently become more available to application engineers than ever. For example, many computer hardware manufacturers provide them in their scientific software library. For this reason, we have designed our second generation FETI solvers to work well with both skyline and sparse direct local solvers.

It is also well-known that for many large-scale structural problems, sparse solvers are faster and more memory efficient than skyline solvers. For this and the reason stated above, sparse solvers are rapidly becoming the direct method of choice for finite element structural applications. However, the subdomain problems that arise in the FETI methodology are by definition smaller size problems. Consequently, the performance of a FETI solver may or may not be very sensitive to the specific choice of a direct local solver. In order to investigate this matter, we consider here the solution by the one-level and two-level FETI methods of two different structural problems on a single processor Origin 2000

1. the diffraction grating problem described in Section 5.4 (Fig. 1) with 120,987 d.o.f. This problem is representative of a class of three-dimensional "solid" problems for which sparse solvers still produce a relatively important amount of fill-in during the factorization of a stiffness matrix. We remind the reader that for this problem, the optimal number of subdomains on an Origin 2000 processor is $N_s = 128$,
2. the stress analysis of an alloy wheel clamped at a few center points and loaded by a set of concentrated forces at its top rim. The finite element model of this wheel contains 313,856 3-noded triangular shell elements and 936,102 d.o.f. (Fig. 3). This problem is representative of a class of two-and-a-half dimensional "shell" problems for which sparse solvers are in principle more computationally efficient than skyline solvers. A preliminary investigation has indicated that $N_s = 175$ is the optimal number of subdomains for the solution of this problem by the two-level FETI method equipped with the Dirichlet preconditioner and the corner modes.

We equip our FETI methods with two different direct local solvers: the skyline solver described in [47], and the `BLK` sparse solver developed at the Oak Ridge National Laboratories [48]. The skyline solver can invoke

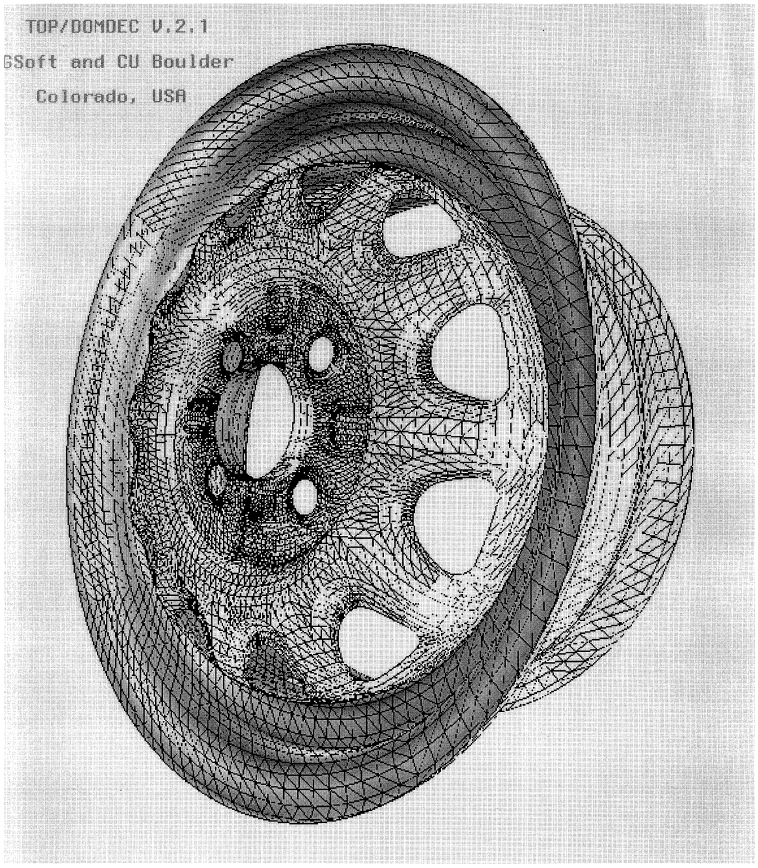

Fig. 3. Finite element discretization of an alloy wheel.

either the RCM reordering scheme [49], or the wavefront reduction algorithm [50]. The *BLK* sparse solver is bundled with its own reordering scheme.

The performance results reported in Tables 1 and 2 show that

- for the diffraction grating system, the sparse local solver is slightly slower than the skyline one at factoring the subdomain stiffness matrices. Because it also requires more memory – 4 Mb precisely – we conclude that this disappointing performance of the sparse factorizer is due to the fact that, for this problem, it produces more fill-in than the skyline counterpart. We also note that the memory requirements of the subdomain problems account for 50% of the total memory requirements of the one-level FETI method,
- for the diffraction grating problem, the local solves – that is, the subdomain forward and backward substitutions performed at each FETI iteration – are significantly slower when the sparse solver is employed. This suggests that the forward and backward substitutions of our skyline solver exploit better the caches of the Origin 2000 than their counterparts in the `BLK` sparse solver,

Table 1
CPU analysis of the solution of the diffraction grating problem on one processor Origin 2000 by the one-level FETI method equipped with the lumped preconditioner and 128 subdomains, and governed by the convergence criterion (50)

Local solver	Local factor (local memory)	Local solves	Total CPU FETI (Total memory FETI)
Skyline	25.8 s (166 Mb)	60.5 s	132.9 s (345 Mb)
Sparse	27.4 s (170 Mb)	99.5 s	179.9 s (349 Mb)

Table 2
CPU analysis of the solution of the alloy wheel problem on one processor Origin 2000 by the two-level FETI method equipped with the Dirichlet preconditioner and 175 subdomains, and governed by the convergence criterion (50)

Local solver	Local factor (local memeory)	Local solves +preconditioner	Total CPU FETI (Total memory FETI)
Skyline	269 s (1062 Mb)	2113 s	2906 s (2580 Mb)
Sparse	198 s (734 Mb)	1833 s	2646 s (1924 Mb)

- for the alloy wheel problem, the sparse local solver affects both the subdomain problems and their treatment by the Dirichlet preconditioner (17). It is 1.36 times faster than the skyline local solver at factoring the subdomain stiffness matrices, and requires 1.45 times less memory space. However, it is only 1.15 times faster at performing the subdomain local and forward backward substitutions at each FETI iteration, for the same reason as highlighted above. As a result, the two-level FETI method equipped with the sparse local solver is only 1.1 times faster at solving this problem than when equipped with the skyline local solver.

From the above sample study, it follows that a sparse direct local solver has the potential of improving the overall performance of the FETI method, memory wise and CPU wise, at least for two-and-a-half dimensional problems such as plates and shells. However, counting on this potential requires first optimizing a target sparse solver to small and medium size problems, and to cache management.

Remark 6. *Testing the* `PSLDLT` *sparse solver that is provided by Silicon Graphics for their Origin* 2000 *computers is not possible because* (*a*) *it does not handle singular systems, and* (*b*) *its source code is not available to the users and therefore it cannot be upgraded to solve singular systems.*

6.2. A fast and robust scheme for extracting $\mathbf{R}^{(s)}$ and computing $\mathbf{K}^{(s)^+}$

The subdomains generated by partitioning a given mesh into submeshes $\Omega^{(s)}$ can be either fully restrained, or partially restrained, or unrestrained. In the first case, the corresponding stiffness matrices $\mathbf{K}^{(s)}$ are SPD. In the latter two cases, the subdomains are called floating subdomains, and their corresponding stiffness matrices are singular. For each floating subdomain, a basis of $\ker \mathbf{K}^{(s)}$ is given by the so-called rigid body modes of that subdomain.

An internal mechanism is another frequent source of singularity of $\mathbf{K}^{(s)}$ that has not received much attention in the DD literature. Such a mechanism can occur when the finite element global model contains elements with less than 6 d.o.f. per node. For example, consider the discretization by solid elements and decomposition in two subdomains of the T-shaped structure shown in Fig. 4. The reader can observe that the upper subdomain can freely rotate around the *AB* axis, which illustrates how a mesh partitioning proces can generate a subdomain with an internal mechanism. Because of this mechanism, the stiffness matrix of the upper subdomain is singular, whether this subdomain has or has not a sufficient number of essential boundary conditions to prohibit any rigid body motion.

Hence in general, a subdomain stiffness matrix $\mathbf{K}^{(s)}$ can be singular either because $\Omega^{(s)}$ is a floating subdomain, and/or because $\Omega^{(s)}$ contains one or several mechanisms. It follows that in general, a basis of $\ker \mathbf{K}^{(s)}$ is given by the so-called "zero energy" modes of $\Omega^{(s)}$. These modes include the rigid body as well as the "kinematic" modes of that subdomain. Therefore, a floating subdomain can have more than 6 zero energy modes.

The factorization based general procedure described in [13] for constructing a basis of $\ker \mathbf{K}^{(s)}$ and computing a generalized inverse $\mathbf{K}^{(s)^+}$ is applicable to floating subdomains with or without internal mechanisms, as well as to subdomains with internal mechanisms only. However, as discussed in Section 5.2, this procedure depends on the arbitrary tolerance ϵ (see (47)) and lacks robustness.

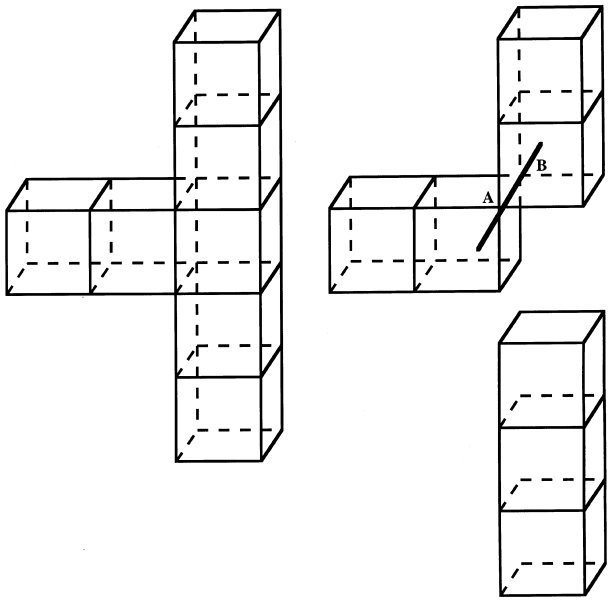

Fig. 4. A mesh partition with an internal mechanism.

Alternatively, the null space of a singular subdomain stiffness matrix can be found by applying the inverse power iteration method [39] to the solution of the shifted eigenvalue problem

$$(\mathbf{K}^{(s)} + \sigma \mathbf{I}^{(s)})\mathbf{x}^{(s)} = \zeta \mathbf{x}^{(s)}, \tag{51}$$

where $\sigma > 0$. This strategy is robust and simple to implement. However, in the context of the design of a fast iterative solver such as FETI, this strategy is also computationally expensive, especially since it does not provide simultaneously a generalized inverse $\mathbf{K}^{(s)+}$ of the subdomain stiffness matrix.

Recently, a robust and computationally efficient geometric-algebraic method for handling floating subdomains has been proposed in [43]. We have adopted this hybrid method in our second generation FETI solvers, and improved it to address a larger class of singular subdomain problems. Here, we overview this work and highlight its innovative features.

Our basic strategy for handling subdomain singularities, which is based on the methodology described in [43], consists of the following steps:

Step 1. Preprocessing a given mesh partition to remove any existing internal mechanism.

Step 2. For each subdomain $\Omega^{(s)}$, determining the exact dimension $n_R^{(s)}$ of ker $\mathbf{K}^{(s)}$ and a basis of this null space by the following two-step procedure.

Step 2a. First, construct a basis of ker $\mathbf{K}^{(s)}$ assuming that $\Omega^{(s)}$ is unrestrained.

Step 2b. Second, modify this basis to account for the restraints in $\Omega^{(s)}$.

Step 3. Using only $n_R^{(s)}$ as input, modifying the factorization procedure of the chosen direct local solver to compute the factors of $\mathbf{K}^{(s)+}$.

We perform Step 1 using the computationally efficient algorithm described in Section 6.3. After all internal mechanisms are removed, each subdomain becomes either a well-posed or a floating subdomain, and all the subdomain zero energy modes become the subdomain rigid body modes. We compute these rigid body modes by the two-step procedure (Step 2) mentioned above and described below.

First, each subdomain $\Omega^{(s)}$ is assumed to be unrestrained. In that case, it has 3 rigid body modes in two dimensions and 6 in three dimensions. For simplicity, we discuss here the three-dimensional case. The two-dimensional case is treated in a similar manner. For an unrestrained subdomain, the 6 rigid body modes can be computed by applying to $\Omega^{(s)}$ 3 rigid translations and 3 rigid rotations along and around the x, y, and z axes, Let M and O denote, respectively, a generic point in the submesh $\Omega^{(s)}$ of coordinates (x_M, y_M, z_M), and a reference point in the same submesh $\Omega^{(s)}$ of coordinates (x_O, y_O, z_O). The vector of generalized displacements at the point M induced by the 6 rigid body motions defined above can be expressed as [51]

$$\mathbf{v}_M^{(s)} = \bar{\mathbf{R}}_M^{(s)} \bar{\beta}, \quad \text{where}$$

$$\bar{\mathbf{R}}_M^{(s)} = \begin{bmatrix} 1 & 0 & 0 & 0 & (z_M - z_O) & -(y_M - y_O) \\ 0 & 1 & 0 & -(z_M - z_O) & O & (x_M - x_O) \\ 0 & 0 & 1 & (y_M - y_O) & -(x_M - x_O) & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{52}$$

$$\mathbf{v}_M^{(s)} = \begin{bmatrix} v_{M_x} \\ v_{M_y} \\ v_{M_z} \\ \Theta_{M_x} \\ \Theta_{M_y} \\ \Theta_{M_z} \end{bmatrix} \qquad \bar{\beta} = \begin{bmatrix} \bar{\beta}_1 \\ \bar{\beta}_2 \\ \bar{\beta}_3 \\ \bar{\beta}_4 \\ \bar{\beta}_5 \\ \bar{\beta}_6 \end{bmatrix}.$$

Here, Θ_{M_x}, Θ_{M_y}, and Θ_{M_z} are the three rotations around the x, y, and z axes, and $\{\bar{\beta}_i\}_{i=1}^{i=6}$ are 6 real constants. If the rigid body translational and rotational motion of all N_M mesh points in $\Omega^{(s)}$ are stored in a global vector $\mathbf{v}$, Eqs. (52) can be rearranged as

$$\mathbf{v}^{(s)} = \bar{\mathbf{R}}^{(s)} \bar{\beta}, \quad \text{where} \quad \mathbf{v}^{(s)} = \begin{bmatrix} \mathbf{v}_1^{(s)} \\ \vdots \\ \mathbf{v}_{N_M}^{(s)} \end{bmatrix} \quad \bar{\mathbf{R}}^{(s)} = \begin{bmatrix} \bar{\mathbf{R}}_1^{(s)} \\ \vdots \\ \bar{\mathbf{R}}_{N_M}^{(s)} \end{bmatrix}. \tag{53}$$

The columns of $\bar{\mathbf{R}}^{(s)}$ form a basis of the null space of ker $\mathbf{K}^{(s)}$, when $\Omega^{(s)}$ is unrestrained.

Next, attention is shifted to the treatment of the restraints. Let $n_{\mathrm{D}}^{(s)}$ denote the number of Dirichlet boundary conditions that are specified in $\Omega^{(s)}$. For each constrained d.o.f. i, a "canonical" vector $\mathbf{e}_i^{(s)}$ is constructed as follows

$$\mathbf{e}_i^{(s)} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad (1 \text{ at the } i\text{th position}). \tag{54}$$

Let $\mathbf{E}^{(s)} \in \mathbb{R}^{n^{(s)} \times n_{\mathrm{D}}^{(s)}}$, where $n^{(s)}$ denotes the size of the unconstrained subdomain stiffness matrix, be defined as

$$\mathbf{E}^{(s)} = \left[\mathbf{e}_1^{(s)} \cdots \mathbf{e}_{n_{\mathrm{D}}^{(s)}}^{(s)} \right]. \tag{55}$$

From Eq. (53), it follows that for a fully or partially restrained subdomain, the coefficients $\{\bar{\beta}_k\}_{k=1}^{k=6}$ satisfy

$$\mathbf{E}^{(s)\mathrm{T}} \bar{\mathbf{R}}^{(s)} \bar{\beta} = 0 \tag{56}$$

and therefore these coefficients are not linearly independent. If $\{\beta_k\}_{k=1}^{k=n_{\mathrm{R}}^{(s)}}$ denote the corresponding set of linearly independent coefficients, there exists a matrix $\mathbf{T}^{(s)} \in \mathbb{R}^{6 \times n_{\mathrm{R}}^{(s)}}$ such that

$$\bar{\beta} = \mathbf{T}^{(s)} \beta. \tag{57}$$

Substituting Eq. (57) into Eqs. (53) and (56) gives

$$\mathbf{v}^{(s)} = \mathbf{R}^{(s)} \beta, \quad \mathbf{E}^{(s)^{\mathrm{T}}} \mathbf{R}^{(s)} \beta = 0, \quad \text{where} \quad \mathbf{R}^{(s)} = \bar{\mathbf{R}}^{(s)} \mathbf{T}^{(s)} \tag{58}$$

The columns of $\mathbf{R}^{(s)} \in \mathbb{R}^{n^{(s)} \times n_R^{(s)}}$ describe all possible and independent rigid body motions of the subdomain $\Omega^{(s)}$ and therefore form a basis of ker $\mathbf{K}^{(s)}$. If $n_R^{(s)} = 0$, then $\Omega^{(s)}$ is fully restrained. If $n_R^{(s)} = 6$, then $\Omega^{(s)}$ is unrestrained. Finally if $0 < n_R^{(s)} < 6$, then $\Omega^{(s)}$ is partially restrained. The matrix $\mathbf{T}^{(s)}$ can be computed by constructing

$$\mathbf{Z}^{(s)} = \mathbf{E}^{(s)^T} \bar{\mathbf{R}}^{(s)} \tag{59}$$

and deducing from Eqs. (56) and (57)

$$\mathbf{T}^{(s)} = \ker \mathbf{Z}^{(s)}. \tag{60}$$

A basis of the null space of $\mathbf{Z}^{(s)}$ can be extracted by performing the singular value decomposition

$$\mathbf{Z}^{(s)} = \mathbf{X}^{(s)} \Sigma^{(s)} \mathbf{Y}^{(s)^T},$$

where $\mathbf{X}^{(s)} \in \mathbb{R}^{n_D^{(s)} \times n_D^{(s)}}$ and $\mathbf{Y}^{(s)} \in \mathbb{R}^{6 \times 6}$ are two orthogonal matrices, and $\Sigma^{(s)} = \text{diag}(\sigma_1, \ldots, \sigma_q) \in \mathbb{R}^{n_D^{(s)} \times 6}$ with $\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_q \geqslant 0$ and $q = \min(n_D^{(s)}, 6)$. Now, if l is defined by

$$\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_l > \sigma_{l+1} = \cdots = \sigma_q = 0 \tag{61}$$

then

$$\mathbf{T}^{(s)} = \ker \mathbf{Z}^{(s)} = \text{span } (\mathbf{Y}_{l+1}, \ldots, \mathbf{Y}_6) \tag{62}$$

and

$$n_R^{(s)} = 6 - l. \tag{63}$$

Strictly speaking, the determination of l via Eq. (61) implies the usage of an arbitrary tolerance as the SVD decomposition of $\mathbf{Z}^{(s)}$ can return an arbitrarily small value wherever a zero value is expected. However, the SVD algorithm is more stable than Gaussian elimination based factorization methods. Moreover, unlike $\mathbf{K}^{(s)}$ which can be ill-conditioned, $\mathbf{Z}^{(s)}$ is well-conditioned as it depends only on the geometry of the given problem. For these reasons, Eq. (61) does not raise in practice the tolerance issue raised by Eq. (47).

In summary, we compute the exact number $n_R^{(s)}$ of subdomain rigid body modes, and a matrix representation $\mathbf{R}^{(s)}$ of these quantities, by a two-step procedure. First, we assume that $\Omega^{(s)}$ is unrestrained and construct the matrix $\bar{\mathbf{R}}^{(s)}$ (53) by a fast geometric method. Next, we construct the matrix $\mathbf{Z}^{(s)}$ (59) by applying the subdomain boundary conditions to $\bar{\mathbf{R}}^{(s)}$. We compute $n_R^{(s)}$ and a basis $\mathbf{T}^{(s)}$ of ker $\mathbf{Z}^{(s)}$ by performing the SVD decomposition of $\mathbf{Z}^{(s)}$. This SVD decomposition is computationally inexpensive because $\mathbf{Z}^{(s)}$ is an $n_D^{(s)} \times 6$ matrix. Finally, we construct the matrix of subdomain rigid body modes $\mathbf{R}^{(s)}$ by multiplying $\bar{\mathbf{R}}^{(s)}$ and $\mathbf{T}^{(s)}$. For numerical examples that demonstrate the robustness and computational efficiency of this procedure, we refer the reader to [43].

It remains to discuss Step 3 of our computational procedure for handling floating subdomains, which deals with obtaining the factors of the generalized inverse $\mathbf{K}^{(s)+}$. For this purpose, we note that for each floating subdomain $\Omega^{(s)}$, there exists an equation ordering for which the zero pivots discussed in Section 5.2 occur during the factorization of the last q rows or columns of $\mathbf{K}^{(s)}$, where q is a small positive integer and $q \geqslant n_R^{(s)}$. Furthermore, we note that such an ordering may differ from the ordering that minimizes the storage requirements of the subdomain stiffness matrix only at a few nodes, and therefore does not increase significantly storage costs. For example, if the mesh contains elements with 6 d.o.f. per node, it suffices to number last the 6 d.o.f. that are attached to any node with rotational d.o.f. to obtain $q = 6$. This is because clamping all 6 d.o.f. at one node eliminates all potential rigid body motions. If the mesh does not contain any element with rotational d.o.f., it suffices to number last all translational d.o.f. that are attached to 3 non-aligned nodes – for example, any 3 nodes of any solid element – to obtain $q = 9$. This is because clamping the 9 translational d.o.f. of 3 non-aligned nodes prevents any rigid subdomain body motion. After such an equation ordering is implemented, and the exact number $n_R^{(s)}$ of subdomain rigid body is computed, the following tolerance-free version of the factorization procedure described in [13] and summarized in Section 5.2 can be used to compute the factors of the generalized inverse $\mathbf{K}^{(s)+}$. The factorization procedure of the chosen direct local solver is applied to the first $n_s - q$ rows or columns of the constrained stiffness

matrix $\mathbf{K}^{(s)}$. Then, a factorization method with *full pivoting* is applied to the subsequent q equations, and the last $n_R^{(s)}$ pivots within the last $q \times q$ block are treated as zero pivots independently of their values.

6.3. A mechanism buster

The hybrid geometric-algebraic procedure proposed in [43] and outlined in the previous section is robust when the given mesh partition is mechanism-free. One approach for satisfying this requirement is to postprocess the given mesh partition in order to eliminate any internal mechanism it may contain. Such an approach is desirable because it allows the generation of the original mesh partition by any of the readily available or published mesh partitioning algorithms.

The mechanism "buster" we have recently designed for postprocessing arbitrary mesh decompositions is based on the following observations

- a subdomain that contains n_g disconnected groups of elements can be viewed as n_g independent subdomains,
- for two-dimensional meshes that do not contain any bar element and where each node does not necessarily have rotational d.o.f., if each element of a given subdomain shares at least one edge with that subdomain, then no mechanism can exist between that element and the subdomain it belongs to,
- the partitioning of a mechanism-free two-and-a-half dimensional mesh where each node has rotational d.o.f. (i.e. beam, plate, and shell elements) cannot introduce any internal mechanism,
- for three-dimensional meshes that do not contain any bar element, it suffices that each solid element share a face with the subdomain it belongs to guarantee that no mechanism can exist between an element and the subdomain it belongs to,
- given two mechanism-free groups of elements, no mechanism can exist between them if at least one element in the three-dimensional case, or one edge in the two- and two-and-a-half dimensional cases, in one group shares a face or and edge with one elements in the other.

From the above observations, it follows that for any mesh that does not contain a bar element, the mechanism buster described below finds and removes any mechanism that is induced by mesh partitioning

Step 1. For each subdomain, build the list of faces describing each element by assigning a unique identification number to each face.

Step 2. For each subdomain, build the face wise graph of element adjacency in which two elements are adjacent if they share a common face.

Step 3. For each subdomain, examine this graph and check if it has disjoint components. If it does not, the subdomain is mechanism-free. If it does, perform the following step.

Step 4. For each disjoint component, compute its number of elements. If this number is greater than some predefined threshold n_{min}, define this component as an additional subdomain. If not, merge this component with the subdomain that shares the largest number of faces with this disjoint component.

Note that the condition that is checked in Step 3 is a sufficient but not necessary condition for ensuring that a subdomain is mechanism-free. Note also that for each disjoint component examined in Step 4, there exists at least one subdomain that shares with it at least one face.

The mechanism buster described above is non-recursive, simple to implement, and computationally efficient. Its extension to meshes with bar elements is straightforward but code dependent. When applied to a mesh paritition containing N_s subdomains, it produces a new mesh partition that retains most of the characteristics of the original one. However, it typically increases N_s by a few subdomains. This is not a problem whatever is the target number of processors N_p, because as will be discussed in Section 6.5, the second generation of FETI methods distinguishes between the concept of a subdomain and that of a processor.

6.4. Coarse problem solvers

Two coarse problems can arise at each FETI iteration: the first one (48) is mandatory for static problems, and the second one (49) is optional but recommended for second-order elastodynamic problems and for fourth-order problems. For the second generation FETI solvers, we have opted for the solution of both coarse problems by a direct method. Such a strategy requires forming and storing the left-hand sides of

both coarse problems. Therefore, its implementation is more complex than that of the solution of these two auxiliary problems by an iterative algorithm, as was done in the first generation FETI solvers. However, such a strategy can only increase the robustness of a FETI solvers. Furthermore, when the number of subdomains N_s is very large, solving the coarse problems (48) and (49) by a direct method can be computationally more efficient than solving them by an iterative scheme, because of the reasons given in Section 5.3.

The matrix $\mathbf{G}_I^T\mathbf{Q}\mathbf{G}_I$ governing the first coarse problem (48) is in general sparse. In particular, when $\mathbf{Q} = \mathbf{I}$, its non-zero pattern is dictated by the subdomain-to-subdomain connectivity, and can be determined by assimilating each subdomain to a superelement with a number of d.o.f. equal to the number of subdomain rigid body modes. Constructing this matrix calls for subdomain-by-subdomain computations that are inherently parallel and that require inter-processor communication only between neighboring subdomains. Currently, we store $\mathbf{G}_I^T\mathbf{Q}\mathbf{G}_I$ in skyline format.

On the other hand, the left-hand side matrix of the second coarse problem (49) is in general full. For simplicity, we restrict here our discussion of the solution of this second coarse problem by a direct method to the case $\mathbf{Q} = \mathbf{I}$.

First, we note that

$$\begin{aligned}\mathbf{P}\mathbf{C} &= (\mathbf{I} - \mathbf{G}_I(\mathbf{G}_I^T\mathbf{G}_I)^{-1}\mathbf{G}_I^T)\mathbf{C} \\ &= \mathbf{C} - \mathbf{G}_I\mathbf{R}_{cg},\end{aligned} \tag{64}$$

where

$$\mathbf{R}_{cg} = (\mathbf{G}_I^T\mathbf{G}_I)^{-1}\mathbf{G}_I^T\mathbf{C} \tag{65}$$

is a full matrix that is however constructed by performing repeated sparse forward and backward substitutions. Next, we note that using Eqs. (64) and (65), the matrix governing the second auxiliary coarse problem (49) can be written as

$$\mathbf{C}^T\mathbf{P}^T\mathbf{F}_I\mathbf{P}\mathbf{C} = \mathbf{C}^T\mathbf{F}_I\mathbf{C} - (\mathbf{G}_I^T\mathbf{F}_I\mathbf{C})\mathbf{R}_{gc} - \mathbf{R}_{gc}^T(\mathbf{G}_I^T\mathbf{F}_I\mathbf{C}) + \mathbf{R}_{gc}^T(\mathbf{G}^T\mathbf{F}_I\mathbf{G})\mathbf{R}_{gc}, \tag{66}$$

which shows that the evaluation of this matrix requires forming matrices that are similar to that governing the first coarse problem (48), and performing computationally intensive matrix–matrix computations. These dense matrix–matrix multiplications are most efficiently carried out by parallelized BLAS-3 routines. Nowadays, these routines are available in optimized form on almost every advanced computational platform.

After they have been formed and assembled in parallel, the matrices governing both coarse problems (48) and (49) are also factored in parallel using the parallel skyline solver described in [47]. However, because of their limited degree of parallelism, the forward and backward substitutions that are subsequently required at each FETI iteration for solving both coarse problems (48) and (49) are currently performed in sequential mode. Hence, these serial computations ultimately limit the parallel scalability of the current version of our second generation FETI solvers.

6.5. Different subdomain and processor concepts

The software architecture of the first generation FETI solvers was such that one processor could be mapped onto one and only one subdomain, and one subdomain could be assigned to one and only one processor. In other words, past implementations of FETI have been based on the equivalence between a subdomain and a processor.

However, there are multiple reasons and incentives for providing the possibility to assign multiple subdomains to a single processor.

- as shown in Section 5.4, there exists a critical value N_s^{cr} of the number of subdomains N_s for which the CPU performance of the FETI method on a given computing hardware with N_p processors is optimal. Selecting a number of subdomains N_s that is different from N_s^{cr} can degrade the CPU performance of FETI by as much as 200%. Since in general $N_s^{cr} \neq N_p$, optimizing the performance of FETI calls for an implementation that distinguishes between the concept of a subdomain and that of a processor,

- independently from the chosen number of processors N_p, it may be desirable to increase the number of subdomains N_s for a given problem in order to reduce the memory requirements of the FETI method,
- as shown in Section 6.3, the treatment of internal mechanisms induced by mesh partitioning is simplified when the number of subdomains is not constrained by the number of available processors,
- configurations where $N_s > N_p$ are easier to load balance,
- a FETI code that allows assigning multiple subdomains to a single processor is a FETI code that runs also on a sequential machine.

Because of the above and other reasons, we have designed the software architecture of the second generation FETI solvers to allow mapping a processor onto an arbitrary number of subdomains. These solvers are packaged in a unified FETI module that can equally run on distributed shared memory (DSM) machines such as Silicon Graphics' Origin 2000, and on local memory parallel processors such as Intel's ASCI-RED machine. This module is written primarily in C++, with some low level scientific subroutines written in FORTRAN. The chief motivations for using C++ are the abstract concepts encountered in finite element codes for which the notion of polymorphism of a function in C++ is natural, the notion of a subdomain in FETI for which an object-oriented programming language such as C++ is ideal, and the fact that C++ compilers have recently improved to the point where a C++ code can deliver the same CPU performance as a FORTRAN code.

Next, we describe some key implementation issues of the second generation FETI module on both DSM and local memory parallel processors. We note that while any implementation targeted for local memory parallel processors is portable to DSM machines, our strategy is to squeeze the most performance out of a given system without sacrificing code portability.

6.5.1. A thread-based implementation for DSM parallel computers

The execution environment in a processor is usually referred to as a thread. The notion of a thread is common in parallel programming environments, and usually implies that the memory address space of the underlying machine is shared among all processors. Here, we are particularly interested in the class of shared memory parallel processors known as the DSM machines. On such computational platforms, the processors are usually organized on so-called node boards, and the memory is globally accessible from any processor.

Enforcing data locality is essential for achieving high performance, even on shared memory systems. For this reason, we distribute the address space to the node boards of a DSM machine on a "first touch" basis. Essentially, when a page is accessed for the first time by a processor, we map the memory onto the physical memory of the node board where this processor resides. Furthermore, we assign intensive work on data to the first processor that has touched this data.

We organize the FETI data as follows. All information pertaining to a subdomain – i.e. nodes, elements, element matrices, communication list – is collected in a `Subdomain` C++ class. Using a predetermined subdomain-to-processor maping, each processor instantiates as many objects of this class as there are subdomains assigned to this processor. We enforce data locality by ensuring that the objects of a given subdomain are always accessed by the same processor during all subdomain-per-subdomain computations.

Most if not all the FETI operations can be performed on subdomain-per-subdomain basis, and therefore are inherently parallel. A notable exception is the solution of the coarse problems (see Section 6.4). The main FETI PCPG loop is organized as a main thread where all global variables such as that storing the result of a dot product are allocated and updated by a single processor. Within this single thread, logical `Tasks` are created and distributed to as many parallel threads as there are processors available to the FETI solver. An example of a task is the factorization of one or several subdomain stiffness matrices $K^{(s)}$. Because the number of threads is arbitrary and independent of the number of tasks to be performed at a given step – i.e. several tasks can be assigned to the same thread – independence between the number of subdomains and the number of processors is achieved. In the extreme case, all tasks can be performed by a single thread – the main thread – in which case the program is executed in serial mode. Intersubdomain communication is carried out simply by manipulating pointers to give access to data in a neighboring subdomain.

6.5.2. A hybrid thread/message-passing implementation for other architectures

In order to also address clusters of DSM machines and/or local memory parallel processors such as Intel's ASCI-RED machine, our second generation FETI software employs MPI for communicating information across shared memory boundaries. In that case, the N_s subdomains of the given mesh partition are organized in a number of clusters equal to the number of available processors, N_p. We perform the clustering of the subdomains by applying the Greedy algorithm [52] to the given mesh partition, and compute the number of subdomains to be assigned to each cluster as to achieve load balance during the local solves.

7. Application to geometrically non-linear problems

An interesting feature of DD based iterative solvers is that they can be tuned for the solution of repeated problems with [22,44,53] or without [54,55,28] a constant left-hand side. Such problems arise, for example, in multiple load static analysis, eigenvalue analysis, linear transient dynamic analysis, and non-linear static and dynamic analyses.

Direct solvers are well-suited for the solution of problems with multiple and/or repeated right-hand sides, because after the system matrix has been factored, the multiple and/or repeated solution can be obtained by relatively inexpensive forward and backward substitutions. On the other hand, iterative solvers are in general ill-suited for these problems because they often must restart from scratch for every different right-hand side. Krylov based techniques that have been proposed for accelerating the solution of these problems by the Lanczos [56] and CG [22] methods are not computationally feasible when these two iterative solvers are applied to the global problem. However, these techniques become feasible in the context of DD, because in that case iterations are applied to the smaller size interface problem. This point highlights one of many advantages of DD based iterative methods over other iterative solvers.

DD based iterative methods can also be tailored for the efficient solution of near-by systems [54,55] – that is, repeated problems with different but related left-hand sides. Except when the Woodbury formula [57] is feasible, direct methods ignore any relationship between two consecutive left-hand sides, and therefore solve near-by systems rather inefficiently. This point highlights an advantage of DD based iterative solvers over direct methods.

In summary, DD methods can be interesting for solving efficiently problems with repeated right-hand sides, and near-by problems. The objective of this section is to discuss this topic in the context of the FETI framework discussed in this paper, and the non-linear analysis of structures. Because our main interest is in aerospace applications, we focus on geometrically non-linear static problems and their treatment by a corotational formulation [58–60].

7.1. The corotational formulation

In many applications, a structure undergoes small deformations but large rigid body motions. In such cases, a structural analysis must account for geometrically non-linear effects because the overall motion of the structure contains large displacements and large rotations. The corotational formulation [58–60] is one among several possible approaches for analyzing structures in such applications. Its key idea is to decompose the structural motion into a purely rigid component and a deformational one.

In most modern corotational finite element methods [61,62], the discretization is performed before the equation of equilibrium are established. A frame is attached to each finite element, and the motion of this (rigid) frame is used to decompose the element motion into a purely rigid body component and a deformational one. For the displacement d.o.f., this can be expressed in vector form as

$$\mathbf{u} = \mathbf{u}_R + \mathbf{u}_D, \tag{67}$$

where $\mathbf{u}_R$ and $\mathbf{u}_D$ are the rigid body and deformational components, respectively. For the rotational d.o.f., this can be written in matrix form at the node level as

$$\mathbb{R} = \mathbb{R}_D \mathbb{R}_R, \tag{68}$$

where $\mathbb{R}_D$ and $\mathbb{R}_R$ denote respectively the deformational and rigid body rotation matrices at each node.

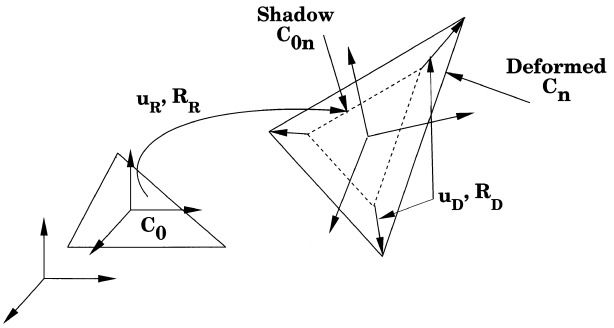

Fig. 5. Reference, shadow, and deformed configurations.

For each finite element, three configurations are considered: a reference or initial configuration $\mathscr{C}_0$, the deformed configuration $\mathscr{C}_n$ where n refers to the n-th step of the non-linear analysis, and the so-called shadow configuration $\mathscr{C}_{0n}$. This shadow configuration is obtained by applying to $\mathscr{C}_0$ the rigid body motion undergone by the element (Fig. 5). Because the structure is assumed to undergo small deformations, small deformation measures are used in the frame attached to the shadow configuration. For this reason, the standard element stiffness matrix, among other element-level quantities such as the projector introduced in [61] for filtering out the rigid body rotations, is used for evaluating the internal forces in a frame attached to $\mathscr{C}_{0n}$. These internal forces are then transformed to the global frame and assembled to obtain the non-linear equations of equilibrium

$$\mathbf{f}^{\text{int}}(\mathbf{u}) + \mathbf{f}^{\text{ext}}(\xi) = 0, \tag{69}$$

where $\mathbf{f}^{\text{int}}$ and $\mathbf{f}^{\text{ext}}$ denote, respectively, the internal and external forces, and ξ is a load parameter.

7.2. Non-linear solution strategies

We consider the solution of the non-linear problem (69) by the Newton method, and a variant that we refer to as a Newton-like method. Both methods solve this problem iteratively by constructing the successive approximations

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \delta\mathbf{u}_n, \tag{70}$$

where $\delta\mathbf{u}_n$ is the solution of the linear system of equation obtained by linearizing problem (69) around $\mathbf{u}_n$

$$\mathbf{K}_n\delta\mathbf{u}_n = -\mathbf{f}^{\text{int}}(\mathbf{u}_n) + \mathbf{f}^{\text{ext}}(\xi) \quad \mathbf{g}_n. \tag{71}$$

In most if not all corotational formulations, the tangent stiffness matrix $\mathbf{K}_n = \mathbf{K}(\mathbf{u}_n)$ is unsymmetric before convergence is reached, which prohibits the solution of the system of Eq. (71) by any of the FETI methods discussed in this paper. However, it was shown in [63] that $\mathbf{K}_n$ can be symmetrized without harming the convergence rate of the Newton method. For this reason, and because we are interested in solving (71) by FETI, we employ in this work a symmetrized tangent stiffness matrix $\mathbf{K}_n$.

We consider two non-linear solution strategies

1. a Newton-like method where the tangent stiffness matrix is frozen within a load increment. This strategy, which aims at reducing the number of updates of the tangent stiffness matrix, gives rise to a series of problems with a fixed left-hand side but different right-hand sides,
2. the "full" Newton method where the tangent stiffness matrix is updated at each non-linear step within a load increment. This strategy, which usually allows larger load increments than the previous one, gives rise to a series of near-by problems,

We show that the extension of the FETI method proposed in [22] for addressing problems with repeated right-hand sides is a two-level FETI method obtained by specifying the $\mathbf{C}$ matrix introduced in Section 3.1. We also overview the extension of the FETI method proposed in [54–56] for solving near-by problems. Using the non-linear static analysis of a stiffened wing panel from the V22 tiltrotor aircraft as an example,

we highlight the potential of these extended FETI methods for the solution of geometrically non-linear problems by both strategies outlined above.

7.3. Acceleration of the solution by FETI of a set of linearized problems characterized by a constant tangent stiffness matrix

Let

$$\mathbf{K}\delta\mathbf{u}_i = \mathbf{g}_i \quad i = 1, \ldots, N_\xi \tag{72}$$

denote a series of N_ξ successive problems associated with the Newton-like method outlined above. Solving these problems by the FETI method requires first transforming them into

$$(\mathbf{P}^{\mathrm{T}}\mathbf{F}_{\mathrm{I}}\mathbf{P})\bar{\lambda}_i = \mathbf{P}^{\mathrm{T}}\left(\mathbf{d}_i - \mathbf{F}_{\mathrm{I}}\lambda_i^0\right) \quad i = 1, \ldots, N_\xi \tag{73}$$

then computing the solutions of the above interface problems by either the one-level or two-level FETI method described in Box 1 and 2, respectively.

Both FETI solvers compute the solution of Eqs. (73) by solving the following minimization problems

$$\min_{\bar{\lambda}\in\mathbb{R}^n} \Phi_i(\bar{\lambda}) = \frac{1}{2}\bar{\lambda}^{\mathrm{T}}\mathbf{A}\bar{\lambda} - \mathbf{b}_i^{\mathrm{T}}\bar{\lambda} \quad i = 1, \ldots, N_\xi, \quad \text{where} \quad \mathbf{A} = \mathbf{P}^{\mathrm{T}}\mathbf{F}_{\mathrm{I}}\mathbf{P}, \quad \mathbf{b}_i = \mathbf{P}^{\mathrm{T}}(\mathbf{d}_i - \mathbf{F}_{\mathrm{I}}\lambda_i^0) \tag{74}$$

and n denotes the size of each of the interface problem (73). Let

$$V_i^{r_i} = \{\mathbf{p}_i^1, \mathbf{p}_i^2, \ldots, \mathbf{p}_i^k, \ldots, \mathbf{p}_i^{r_i}\} \quad i = 1, \ldots, N_\xi \tag{75}$$

denote the Krylov space consisting of the search directions $\mathbf{p}_i^k$ generated during the solution of the ith problem (73) by r_i FETI iterations, and let $\mathbf{V}_i \in \mathbb{R}^{n\times r_i}$ denote the rectangular matrix associated with $V_i^{r_i}$. From the orthogonality properties of the conjugate gradient method (in practice, from the explicit reorthogonalization enforced by a FETI solver), it follows that

$$\mathbf{V}_i^{\mathrm{T}}\mathbf{A}\mathbf{V}_i = \mathbf{D}_i \quad i = 1, \ldots, N_\xi, \tag{76}$$

where $\mathbf{D}_i$ is the diagonal matrix

$$\mathbf{D}_i = [d_{i1} \quad d_{i2} \quad \cdots \quad d_{i_{r_i}}]. \tag{77}$$

Suppose that during the solution by FETI of the first problem $\mathbf{A}\bar{\lambda}_1 = \mathbf{b}_1$, the matrices $\mathbf{V}_1$ and $\mathbf{A}\mathbf{V}_1$ are accumulated and stored. Consider next the second problem $\mathbf{A}\bar{\lambda}_2 = \mathbf{b}_2$, which can also be written as

$$\min_{\bar{\lambda}\in\mathbb{R}^n} \Phi_2(\bar{\lambda}) = \frac{1}{2}\bar{\lambda}^{\mathrm{T}}\mathbf{A}\bar{\lambda} - \mathbf{b}_2^{\mathrm{T}}\lambda. \tag{78}$$

If $\mathbb{R}^n$ is decomposed as follows

$$\mathbb{R}^n = V_1^{r_1} \oplus V_1^{r_1^*}, \dim(V_1^{r_1^*}) = n - r_1, \quad V_1^{r_1} \text{ and } V_1^{r_1^*} \text{ are } \mathbf{A}\text{-orthogonal} \tag{79}$$

then $\bar{\lambda}_2$ can be searched for in the following form

$$\bar{\lambda}_2 = \bar{\lambda}_2^0 + \mu_2, \quad \bar{\lambda}_2^0 \in V_1^{r_1}; \mu_2 \in V_1^{r_1^*}, \quad \text{and} \quad \bar{\lambda}_2^{0\mathrm{T}}\mathbf{A}\mu_2 = \mu_2^{\mathrm{T}}\mathbf{A}\bar{\lambda}_2^0 = 0. \tag{80}$$

Substituting the above expression of $\bar{\lambda}_2$ into (78), and exploiting the orthogonality condition (80) reveals that $\bar{\lambda}_2^0$ is the solution of the uncoupled minimization problem

$$\min_{\bar{\lambda}\in V_1^{r_1}} \Phi_2(\bar{\lambda}) = \frac{1}{2}\bar{\lambda}^{\mathrm{T}}\mathbf{A}\bar{\lambda} - \mathbf{b}_2^{\mathrm{T}}\bar{\lambda} \tag{81}$$

and μ_2 is the solution of the uncoupled minimization problem

$$\min_{\mu \in V_1^{r_1^*}} \Phi_2(\mu) = \frac{1}{2}\mu^{\mathrm{T}}\mathbf{A}\mu - \mathbf{b}_2^{\mathrm{T}}\mu. \tag{82}$$

In order to solve problem (81), we first note that since $\bar{\lambda}_2^0 \in V_1^{r_1}$, there exists a $\theta_2^0 \in \mathbb{R}^{r_1}$ such that

$$\bar{\lambda}_2^0 - \mathbf{V}_1\theta_2^0. \tag{83}$$

The vector θ_2^0 is determined by substituting Eq. (83) into Eq. (81) and solving the resulting minimization problem, which gives

$$(\mathbf{V}_1^{\mathrm{T}}\mathbf{A}\mathbf{V}_1)\theta_2^0 = \mathbf{V}_1^{\mathrm{T}}\mathbf{b}_2. \tag{84}$$

From Eq. (76), it follows that θ_2^0 is given by

$$\mathbf{D}_1\theta_2^0 = \mathbf{V}_1^{\mathrm{T}}\mathbf{b}_2 \Rightarrow \theta_{2j}^0 = \frac{[\mathbf{V}_1^{\mathrm{T}}\mathbf{b}_2]_j}{d_{1_j}} \quad j = 1, \ldots, r_1 \tag{85}$$

and from Eqs. (83) and (84) we conclude

$$\bar{\lambda}_2^0 = \mathbf{V}_1\mathbf{D}_1^{-1}\mathbf{V}_1^{\mathrm{T}}\mathbf{b}_2. \tag{86}$$

Note that the evaluation of θ_2^0 requires only r_1 floating point operations, and that of $\bar{\lambda}_2^0$ a single matrix–vector product.

Next, we turn to the solution of problem (82) by the FETI method. Since the decomposition (80) requires μ_2 to be $\mathbf{A}$-orthogonal to $\bar{\lambda}_2^0$, it follows that at each iteration k, the search directions $\mathbf{p}_2^k$ must be explicitly $\mathbf{A}$-orthogonalized to $\mathbf{V}_1$. This entails modifying the basic FETI solver to compute the following "enriched" search directions $\tilde{\mathbf{p}}_2^k$

$$\tilde{\mathbf{p}}_2^k = \mathbf{p}_2^k + \sum_{q=1}^{q=r_1} \eta_q \mathbf{p}_1^q, \quad \eta_q = -\frac{\mathbf{p}_1^{q^{\mathrm{T}}}\,\mathbf{A}\mathbf{p}_2^k}{\mathbf{p}_1^{q^{\mathrm{T}}}\,\mathbf{A}\mathbf{p}_1^q} = -\frac{\mathbf{p}_2^{k^{\mathrm{T}}}\,\mathbf{A}\mathbf{p}_1^q}{\mathbf{p}_1^{q^{\mathrm{T}}}\,\mathbf{A}\mathbf{p}_1^q} \tag{87}$$

instead of the usual search directions $\mathbf{p}_2^k$. Note that the right-hand sides of Eqs. (84) and (85) explain why it was assumed that $\mathbf{V}_1$ and $\mathbf{A}\mathbf{V}_1$ are accumulated and stored during the solution by FETI of the first problem $\mathbf{A}\bar{\lambda}_1 = \mathbf{b}_1$.

Since $V_1^{r_1^*}$ is only a subspace of $\mathbb{R}^n$, it follows that the FETI method equipped with the starting value given by Eq. (85), and the orthogonalization procedure specified in Eq. (87) can be expected to converge faster for the second problem than for the first one. This was extensively demonstrated in [22] where this approach was first proposed, and in [44,53] for different applications. The extension of this methodology to $N_\xi > 2$ right-hand sides is straightforward and can be found in [22,44,53]. For many applications, convergence was observed to be continuously accelerated from one right-hand side to the next one. This is because for each additional right-hand side, the number of components of the solution that are captured by the starting value of the form of $\bar{\lambda}_2^0$ increases, and the dimension of the supplementary space of the form of $V_1^{r_1^*}$ in which the FETI method is forced to iterate decreases.

It turns out that the methodology summarized above for extending the FETI method to problems with repeated right-hand sides fits the two-level FETI frame-work described in Sections 3.1 and 3.2 of this paper. Indeed, considering the solution of the second problem $\mathbf{A}\mathbf{x}_2 = \mathbf{b}_2$, if the $\mathbf{C}$ matrix introduced in Section 3.1 is set to

$$\mathbf{C} = \mathbf{V}_1 \tag{88}$$

then for $\mathbf{x}^0 = 0$, the starting value given by Eq. (36) becomes

$$\tilde{\mathbf{x}}^0 = \mathbf{V}_1\mathbf{D}_1^{-1}\mathbf{V}_1^{\mathrm{T}}\mathbf{b}_2. \tag{89}$$

This starting value is identical to that given in Eq. (86). Furthermore, using Eqs. (88) and (76), the auxiliary coarse problem (32) becomes

$$\mathbf{D}_1\gamma^k = -\mathbf{V}_1^{\mathrm{T}}\mathbf{A}\mathbf{p}_2^k. \tag{90}$$

The solution of this coarse problem is

$$\gamma_q^k = -\frac{\mathbf{p}_1^{q^{\mathrm{T}}}\,\mathbf{A}\mathbf{p}_2^k}{\mathbf{p}_1^{q^{\mathrm{T}}}\,\mathbf{A}\mathbf{p}_1^q} \quad q = 1, \ldots, r_1. \tag{91}$$

The comparison of Eqs. (85) and (91) shows that

$$\gamma^k = \begin{bmatrix} \eta_1 \\ \vdots \\ \eta_{r_1} \end{bmatrix}, \tag{92}$$

which demonstrates that for $\mathbf{C} = \mathbf{V}_1$, solving the auxiliary coarse problem (32) has the effect of orthogonalizing the search directions $\mathbf{p}_2^k$ to $\mathbf{V}_1$.

Hence, the extension of the FETI method to problems with a constant left-hand side and multiple right-hand sides is a special case of the two- level FETI method in which the $\mathbf{C}$ matrix is set to the accumulation of all the reorthogonalized search directions generated during the solution of the previous problems. From Eq. (56), it follows that this extension is equivalent to preconditioning FETI by yet another preconditioner which is the projector given in the second of Eqs. (33).

7.4. Acceleration of the solution by FETI of a set of near-by problems

Next, we consider the solution of the non-linear equations of equilibrium (69) by the full Newton method. At each load increment, this strategy requires solving a sequence of near-by linearized problems of the form

$$\mathbf{K}_i \delta\mathbf{u}_i = \mathbf{g}_i \quad i = 1, \ldots, N_\xi, \tag{93}$$

where $\mathbf{K}_i = \mathbf{K}(\mathbf{u}_i)$ is the tangent stiffness matrix introduced in Eq. (71). Solving problems (93) by the FETI method transforms them first into

$$\mathbf{A}_i \mathbf{x}_i = \mathbf{b}_i, \quad i = 1, \ldots, N_\xi, \quad \text{where} \quad \mathbf{A}_i = \mathbf{P}_i^{\mathrm{T}} \mathbf{F}_{Ii} \mathbf{P}_i, \quad \mathbf{b}_i = \mathbf{P}_i^{\mathrm{T}} (\mathbf{d}_i - \mathbf{F}_{Ii} \lambda_i^0). \tag{94}$$

The subdomain rigid body modes projector $\mathbf{P}_i = \mathbf{I} - \mathbf{Q}\mathbf{G}_{Ii}(\mathbf{G}_{Ii}^{\mathrm{T}}\mathbf{Q}\mathbf{G}_{Ii})^{-1}\mathbf{G}_{Ii}^{\mathrm{T}}$ can vary from one problem to another, because for geometrically non-linear problems, the dimension for $\ker \mathbf{K}(\mathbf{u}^i)$ can depend on the displacement solution $\mathbf{u}^i$.

Even though two successive tangent stiffness such as $\mathbf{K}_1$ and $\mathbf{K}_2$ are two different matrices, one can reasonably expect that they lead to two FETI interface matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ that are "close" enough to justify preconditioning $\mathbf{A}_2$ with a matrix constructed from data pertaining to $\mathbf{A}_1$. For example, it was shown in [53] that

$$\tilde{\mathbf{A}}_2^{-1} = \mathbf{I} - \mathbf{V}_1 \mathbf{D}_1^{-1} \mathbf{V}_1^{\mathrm{T}} \mathbf{A}_1, \tag{95}$$

where $\mathbf{V}_1$ and $\mathbf{D}_1$ have the same meaning as in Section 7.3 – is an effective preconditioner for $\mathbf{A}_2$, when $\mathbf{A}_1$ and $\mathbf{A}_2$ are two FETI interface matrices associated with two consecutive steps of an adaptive solution strategy. In [54,55], a similar idea was developed in the context of the solution by the full Newton method of elasticity problems with material non-linearities. More specifically, it was proposed in [54,55] to accelerate the solution by FETI of the second problem $\mathbf{A}_2\mathbf{x}_2 = \mathbf{b}_2$ by replacing in each preconditioning step the preconditioned residual $\tilde{\mathbf{A}}_1^{-1}\mathbf{w}_2^k$ by

$$\widehat{\tilde{A}_2^{-1}}\mathbf{w}_2^k = \tilde{\mathbf{A}}_2^{-1}\mathbf{w}_2^k + \mathbf{V}_1\theta_1^k, \tag{96}$$

where $\tilde{\mathbf{A}}_2^{-1}$ is for example the lumped or Dirichlet preconditioner, $\theta_1^k \in \mathbb{R}^{r_1}$ is a vector to be determined, and r_1 has the same meaning as in Section 7.3. If $\mathbf{A}_2$ is sufficiently close to $\mathbf{A}_1$, then the preconditioner $\widehat{\tilde{\mathbf{A}}_2^{-1}}$ is effective if it is sufficiently close to $\mathbf{A}_1^+$, which implies that $\mathbf{A}_1\widehat{\tilde{\mathbf{A}}_2^{-1}}\mathbf{w}_2^k$ should be sufficiently close to $\mathbf{w}_2^k$. For this reason, θ_1^k is determined by solving of the following minimization problem

$$\min_{\theta_1 \in \mathbb{R}^{r_1}} \Phi(\theta_1) = \frac{1}{2}(\tilde{\mathbf{A}}_2^{-1}\mathbf{w}_2^k + \mathbf{V}_1\theta_1)^{\mathrm{T}}\mathbf{A}_1(\tilde{\mathbf{A}}_2^{-1}\mathbf{w}_2^k + \mathbf{V}_1\theta_1) - \mathbf{w}_2^{k\mathrm{T}}(\tilde{\mathbf{A}}_2^{-1}\mathbf{w}_2^k + \mathbf{V}_1\theta_1), \tag{97}$$

which leads to

$$\begin{aligned} &\mathbf{D}_1\theta_1^k = \mathbf{V}_1^{\mathrm{T}}\mathbf{w}_2^k - (\mathbf{A}_1\mathbf{V}_1)^{\mathrm{T}}\tilde{\mathbf{A}}_2^{-1}\mathbf{w}_2^k \\ \Rightarrow \quad &\theta_{1_j}^k = \frac{[\mathbf{V}_1^{\mathrm{T}}\mathbf{w}_2^k - (\mathbf{A}_1\mathbf{V}_1)^{\mathrm{T}}\tilde{\mathbf{A}}_2^{-1}\mathbf{w}_2^k]_j}{d_{1_j}} \quad j = 1, \ldots, r_1. \end{aligned} \tag{98}$$

Similarly, the solution by FETI of the ith of problem (94) can be accelerated by replacing in each preconditioning step the preconditioned residual $\tilde{\mathbf{A}}_i^{-1}\mathbf{w}_i^k$ by

$$\widehat{\tilde{\mathbf{A}}_i^{-1}}\mathbf{w}_i^k = \tilde{\mathbf{A}}_i^{-1}\mathbf{w}_i^k + \sum_{q=1}^{q=i-1}\mathbf{V}_q\theta_q^k, \tag{99}$$

where $\mathbf{V}_q \in \mathbb{R}^{n\times r_q}$ is the rectangular matrix of the search directions $\mathbf{p}_q^k$ generated during the solution by FETI of the qth of problem (94) in r_q iterations. Because of the explicit full reorthogonalization enforced by a FETI solver, the following holds

$$\mathbf{V}_q^{\mathrm{T}}\mathbf{A}_q\mathbf{V}_q = \mathbf{D}_q \quad q = 1, \ldots, i-1, \tag{100}$$

where $\mathbf{D}_q$ is a diagonal matrix of the form given in (77). The vectors θ_q^k determined by solving successively the following minimization problems

$$\begin{aligned} \min_{\theta_q \in \mathbb{R}^{r_q}} \Phi(\theta_q) = \frac{1}{2}&\left(\tilde{\mathbf{A}}_i^{-1}\mathbf{w}_i^k + \sum_{l=1}^{l=q}\mathbf{V}_l\theta_l\right)^{\mathrm{T}}\mathbf{A}_q\left(\tilde{\mathbf{A}}_i^{-1}\mathbf{w}_i^k + \sum_{l=1}^{l=q}\mathbf{V}_l\theta_l\right) \\ &-\mathbf{w}_i^{k\mathrm{T}}\left(\tilde{\mathbf{A}}_i^{-1}\mathbf{w}_i^k + \sum_{l=1}^{l=q}\mathbf{V}_l\theta_l\right), \quad q = 1, \ldots, i-1, \end{aligned} \tag{101}$$

which leads to

$$\mathbf{D}_q\theta_q^k = \mathbf{V}_q^{\mathrm{T}}\mathbf{w}_i^k - (\mathbf{A}_q\mathbf{V}_q)^{\mathrm{T}}\left(\tilde{\mathbf{A}}_i^{-1}\mathbf{w}_i^k + \sum_{l=1}^{l=q-1}\mathbf{V}_l\theta_l\right), \quad q = 1, \ldots, i-1. \tag{102}$$

From Eq. (102), it follows that the strategy described above for accelerating the solution by FETI of a set of near-by problems requires storing for each of them the matrix of search directions $\mathbf{V}_i$, and the matrix–vector product $\mathbf{A}_i\mathbf{V}_i$. Because the size of these quantities is proportional to the size of the interface of the mesh partition, and because one can expect the sequence $\{r_1, r_1, \ldots, r_{N_\xi}\}$ to decrease rapidly, the storage requirements of this strategy are in general affordable.

Remark 7. *The preconditioner presented in Eq.* (99) *is not symmetric. This is not a problem however because all FETI solvers are always used with an explicit reorthogonalization procedure.*

7.5. Application to the geometrically non-linear analysis of a stiffened composite wing panel

In order to highlight the potential of the accelerated FETI methods described in the two previous sections for the solution of non-linear problems, we consider here the geometrically non-linear structural analysis of a stiffened composite wing panel from the V22 tiltrotor aircraft [64]. This panel contains interesting features such as ply drop-offs, ply interleaves, axial stiffeners, transverse ribs, clips, brackets, and a large elliptical access hole. The finite element model we have constructed for this panel is based on the data given in [64]. It contains 46 different material section properties, 147,654 nodes, 584,704 3-noded triangular shell elements, and a total of 885,924 d.o.f. A coarser version of this model is graphically depicted in Fig. 6. The panel is clamped at one end. At the midpoint of the top of the fixture brackets, the out-of-plane

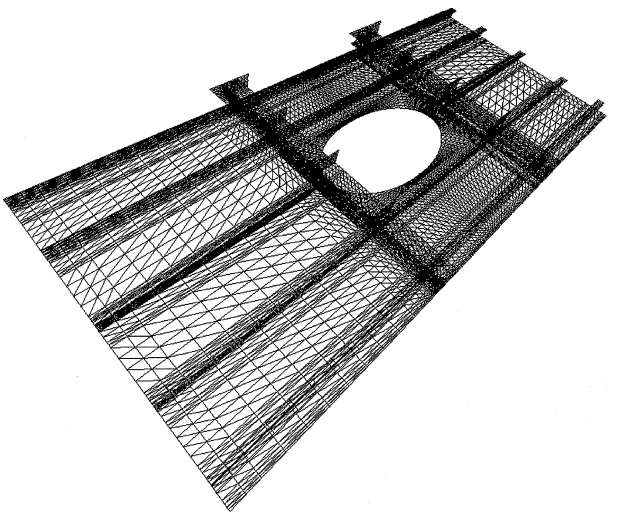

Fig. 6. Finite element discretization of a V22 stiffened wing panel.

displacements and the rotations about the longitudinal axis are constrained to be zero. A uniform end-shortening displacement field is specified at the other end of the panel to simulate the compressive motion of a test machine.

We perform several geometrically non-linear analyses of this panel using both the full Newton and Newton-like solution methods on a 16-processor Origin 2000 Silicon graphics system. In both cases, we monitor the convergence of the non-linear iterations by the following criterion

$$\|\mathbf{f}^{\text{int}}(\mathbf{u}^n) + \mathbf{f}^{\text{ext}}(\xi)\|_2 \leqslant 10^{-6}\|\mathbf{f}^{\text{int}}(\mathbf{u}^0) + \mathbf{f}^{\text{ext}}(\xi)\|_2. \tag{103}$$

We solve all systems of equations generated during the linearization process by the two-level FETI method equipped with the skyline local solver [47], the Dirichlet preconditioner, and the corner modes. For this purpose, we partition the finite element mesh into 250 subdomains using the Greedy algoritm [52] equipped with the subdomain aspect ratio optimizer [65], and set the convergence criterion of FETI to

$$\|\mathbf{K}\delta\mathbf{u} - \mathbf{g}\|_2 \leqslant 10^{-3} \times \|\mathbf{g}\|_2. \tag{104}$$

For this problem, the choice of the tolerance 10^{-3} is justified by the quadratic convergence of the full Newton method.

We report in Table 3 the performance results obtained for four different solution strategies

- FN (*the full Newton method*). For this problem, this solution strategy requires only one load increment, independently of the chosen linear equation solver. At each Newton step, the tangent stiffness matrix and the Dirichlet preconditioner are rebuilt.
- FN-AL (*the full Newton method*). This strategy differs from the previous one only in the fact that the FETI method is also equipped with the modified Dirichlet preconditioner (99) to accelerate the solution of problems with different left-hand sides.
- NL-AR (*the Newton-like method*). For this problem, this solution strategy requires two load increments, independently of the chosen linear equation solver. The tangent stiffness matrix and the Dirichlet preconditioner are frozen during all steps of the same load increment. They are rebuilt only at the beginning of each load increment. The FETI method is also equipped with the Krylov based technique discussed in Section 7.3 for accelerating the solution of problems with multiple right-hand sides.
- NL-AR&L (*the Newton-like method*). This strategy differs from the previous one only in the fact that the FETI solver is also equipped with both procedures discussed in Sections 7.3 and 7.4 for accelerating the solution of problems with multiple right- and left-hand sides, respectively. The first procedure aims at accelerating the solution of the repeated problems that arise within the same load increment. The second procedure is applied to accelerate further the solution of the problems arising in successive increments.

Table 3
Geometrically non-linear analysis of a stiffened composite wing panel with 885,924 d.o.f. on a 16-processor Origin 2000[a]

Load	Newton	FN	FN-AL	NL-AR	NL-AR& L
increment	step	FETI itr.	FETI itr.	FETI itr.	FETI itr.
1	1	143 itr.	143 itr.	143 itr.	143 itr.
	2	198 itr.	171 itr.	72 itr.	72 itr.
	3	198 itr.	108 itr.	36 itr.	36 itr.
	4	198 itr.	46 itr.	19 itr.	19 itr.
2	5	– itr.	– itr.	168 itr.	124 itr.
	6	– itr.	– itr.	64 itr.	57 itr.
	7	– itr.	– itr.	33 itr.	36 itr.
	8	– itr.	– itr.	14 itr.	16 itr.
Total itr.	FETI	737 itr.	468 itr.	549 itr.	503 itr.
Total	search directions	198	468	549	503
Total CPU	FETI	1513 s	979 s	1078 s	940 s
Total CPU	Anal.	1655 s	1139 s	1308 s	1158 s
Total Mem.	FETI	3307 Mb	3657 Mb	3417 Mb	3710 Mb
Total Mem.	Anal.	4865 Mb	5215 Mb	4800 Mb	5093 Mb

[a] The two-level FETI method equipped with the Dirichlet preconditioner and 250 sub-domains, and tailored for the solution of near-by problems with repeated right-hand sides

We emphasize that for each non-linear solution strategy, the number of load increments is set to the maximum value that can be sustained by that non-linear solution method. In particular, for the full Newton method, a single load increment is used. This corresponds to the worst case scenario for accelerating the solution by FETI of a set of near-by problems. Indeed, in that case and for a given total load, the successive tangent matrices are as "far" from each other as they can be.

From the performance results summarized in Table 3, several observations are worth noting

- in the case of the full Newton method, the solution of the system of equations arising in the first step requires less FETI iterations than the solution of each of the systems of equations arising in the subsequent Newton steps. This can be explained by the difference in construction between the first tangent stiffness matrix and the subsequent ones. In the first Newton step, the tangent stiffness matrix is not affected by the geometrical non-linearities as it is set to the elastic stiffness matrix. In all subsequent steps, the tangent stiffness matrix is constructed as the sum of the elastic stiffness matrix, and a set of geometric stiffness matrices evaluated at the previously computed displacement solution. It seems that the contribution of the geometrically non-linear effects worsens the condition number of the elastic stiffiness. Furthermore, in each floating subdomain, the dimension of ker $\mathbf{K}_1^{(s)}$ is equal to 6, whereas for $n > 1$, the dimension of ker $\mathbf{K}_n^{(s)}$ is typically equal to 3. This is because the geometric stiffnesses that contribute to the construction of the tangent stiffness matrix typically remove the rotational rigid body modes. It follows that at the first Newton step, FETI's basic coarse problem (48) is larger than at the subsequent steps, which also explains why the the FETI method converges faster during the first Newton step,
- also in the case of the full Newton method, equipping the FETI solver with the additional Krylov based preconditioner (99) reduces the total number of FETI iterations by a factor equal to 1.57, and reduces the total CPU time spent in the FETI solver by a factor equal to 1.54. On the other hand, it increases the total memory requirements of the FETI solver by 10% only,
- for the Newton-like method, the Krylov based acceleration technique summarized in Section 7.3 reduces the number of FETI iterations from 143 for the first right-hand side, to 19 iterations for the fourth right-hand side. This is an impressive result even if, for this problem, the Newton-like strategy remains slower than the full Newton one,
- in the second load increment of the Newton-like method, superposing the preconditioner (99) to the Krylov based technique for accelerating the solution of problems with multiple right-hand sides improves the

total CPU performance of the FETI solver by 12%. As a result, the Newton-like method performs almost as well as the full Newton method.

In summary, the example discussed herein illustrates the potential of the accelerated FETI methods described in Sections 7.3 and 7.4 for reducing the CPU requirements of geometrically non-linear analyses, at the price of a modest increase in memory storage.

8. Showcase problems

Before concluding this paper, we report on some parallel performance results that we have recently obtained for the second generation FETI solvers on a 32-processor Origin 2000 system with 8 Gb of memory. These performance results correspond to various large-scale stress analyses of two problems from the automotive industry. We analyze them and contrast them with the performance results obtained for the PSLDLT sparse solver applied to the solution of the same problems. We remind the reader that the PSLDLT sparse solver and its equation reordering routine are provided by Silicon Graphics on all their single-processor and parallel-processor systems.

We point out that all performance results discussed herein correspond to 64-bit arithmetic. When the FETI method is employed, its convergence is determined by the criterion (50). We also note that except for one case, all sizes of the problems discussed herein are such that memory swapping does not occur on our Origin 2000 system. Hence, except when specified, all benchmarks reported in this section are not affected by any I/O effect.

Finally, we specify that all mesh partitions employed herein have been generated by the Greedy algoritm [52] equipped with the subdomain aspect ratio optimizer described in [65].

8.1. A wheel carrier problem

First, we consider the linear static analysis of the wheel carrier shown in Fig. 7. Two finite element meshes are constructed for this purpose. The first one is denoted by M1. It contains 542,144 4-nodes tetrahedra elements, 113,610 nodes, and 340,830 d.o.f. The second mesh is finer. It is denoted by M2 and contains 4,337,152 4-noded tetrahedra, 812,368 nodes, and 2,437,104 d.o.f. Hence, the system of equations associated with mesh M2 is 7 times larger than that corresponding to mesh M1.

The solution by the PSLDLT sparse solver of the system of equations associated with mesh M1 requires 1.3 Gb of memory, and consumes 1011 s CPU on a single Origin 2000 processor. On the other hand, the solution of this system by 62 iterations of the one-level FETI method equipped with the lumped preconditioner and 300 subdomains requires only 700 Mb, and consumes only 392 s CPU on the same single Origin 2000 processor. Hence, for this problem and mesh M1, the one-level FETI method requires only half the storage space needed by the sparse solver, and is 2.6 times faster.

Also for mesh M1, the parallel performance results summarized in Fig. 8 show that when the number of processors is varied between 1 and 30, the one-level FETI solver achieves good speedups, and the PSLDLT sparse solver achieves reasonable ones. For $N_p = 30$, the parallel efficiency of the one-level FETI solver on the Origin 2000 is equal to 65%, and that of the PSLDLT sparse solver is equal to 42%. In all cases, the one-level FETI method outperforms the PSLDLT sparse solver by a factor ranging between 2.5 and 5.5. For example, using 30 processors the one-level FETI method requires 21 s CPU for solving the target 340,830 equations, whereas the PSLDLT sparse solver requires 116 s CPU for that purpose.

For mesh M2 which generates more than 2 million d.o.f., the amount of memory required by the PSLDLT sparse solver is so large that the PSLDLT routines fail to determine it. On the other hand, using as many as 1200 subdomains, the one-level FETI method equipped with the lumped preconditioner requires only 7 Gb of memory for solving this problem in 125 iterations. However, because the other steps of the finite element stress analysis require 4 additional Gb of memory, the total storage requirements for this problem add up to 12 Gb, and therefore exceed the 8 Gb of memory that are available on our O2000. It follows that for this problem, the parallel performance results of the one-level FETI solver summarized in Table 4 are tainted by memory swapping, In particular, we note that memory swapping is detrimental to parallel performance because system I/O is performed sequentially on the O2000 system. Nevertheless,

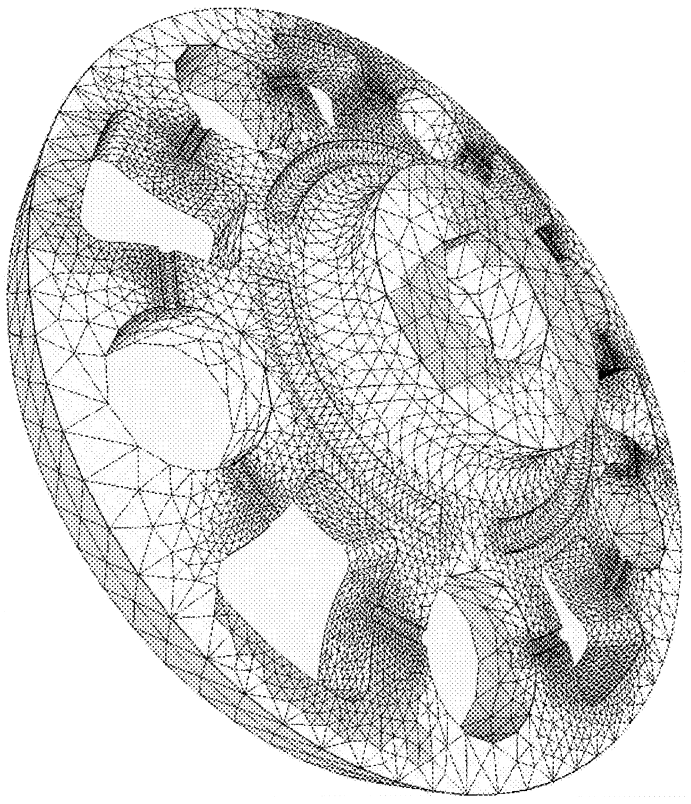

Fig. 7. Finite element discretization of a wheel carrier (mesh M1).

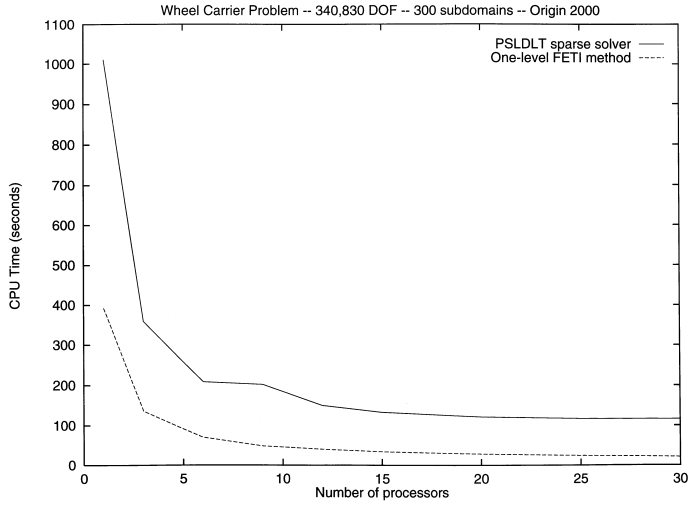

Fig. 8. Wheel carrier problem – Parallel performance results for mesh M1.

because memory swapping is a reality that one has often to deal with, we include in this paper the performance results of the FETI method for this large-scale problem.

We also point out that the jump from 62 FETI iterations for mesh M1 to 125 FETI iterations for mesh M2 is explained by the fact that the one-level FETI method equipped with the lumped preconditioner is not scalable with respect to the problem size. However, while the one-level FETI method equipped with the Dirichlet preconditioner is scalable with respect to the problem size, using the Dirichlet preconditioner for

Table 4
Performance results on a 32-processor O2000 of the one-level FETI solver equipped with the lumped preconditioner and applied to the solution of the wheel carrier problem with 2,437,104 d.o.f. (mesh M2). The mesh is decomposed in 1200 subdomains

Number of processors	CPU coarse (parallel) (s)	CPU coarse (sequential) (s)	Total CPU (%) parallel efficiency
5	153	2420	Reference
10	157	1373	88.0
15	153	1170	68.7
20	153	921	65.5
25	157	827	58.4
30	157	713	56.7

this problem increases the memory requirements of the FETI solver beyond the combined capacities of the physical memory and disk swap space of our O2000 machine.

Because of the large size of mesh M2, $N_p = 5$ is the smallest number of processors we have used in this case for performing the linear static analysis of the wheel carrier. Hence, the parallel efficiency results reported in Table 4 are computed using $N_p = 5$ as the reference point.

From the results reported in Table 4, it follows that the forward and backward substitutions performed on the coarse matrix $\mathbf{G}_I^T\mathbf{Q}\mathbf{G}_I = \mathbf{G}_I^T\mathbf{G}_I$, which define the sequential portion of the solution of the auxiliary coarse problems encountered at each FETI iteration (see Section 6.4), account for about 6% of the total CPU time when $N_P = 5$, and the 22% of the total CPU time when $N_p = 30$. This increase in the importance of the sequential component with the number of processors illustrates Amdahl's law. However, it should be pointed out that this law penalizes parallel processing only when the problem size is fixed, and the number of processors is continuously increased. For this wheel carrier problem, despite memory swapping and the serialization of the forward and backward substitutions of the coarse problems, the FETI method achieves a parallel efficiency of 56.7% on 30 Origin 2000 processors, and solves 2.4 million of equations in less than 12 min.

8.2. Back to the alloy wheel problem

Next, we consider again the alloy wheel described in Section 6.1 and illustrated in Fig. 3. Here, we consider two meshes for the linear stress analysis of this problem. The fine mesh introduced in Section 6.1 and containing 313,856 3-noded triangular shell elements and 936,102 d.o.f., and a coarser mesh containing 78,464 3-noded triangular shell elements and 235,614 d.o.f. As for the wheel carrier problem, we refer to the coarse mesh as mesh M1, and to the finer one as mesh M2.

For this problem, the parallel solution by the `PSLDLT` sparse solver of the system of equations associated with mesh M1 requires 502 Mb of memory, and consumes 42.9 s CPU when using 8 Origin 2000 processors. For mesh M2, this sparse solver requires 2601 Mb memory and consumes 301.2 s CPU on the same 8 processors. The performance results for this problem of the two-level FETI method equipped with the Dirichlet preconditioner are summarized in Table 5 for mesh M1, and Table 6 for mesh M2.

Table 5
Performance results on a 8-processor O2000 of the two-level FETI solver equipped with the Dirichlet preconditioner and applied to the solution of the alloy wheel problem with 235,6146 d.o.f. (mesh M1)

Number of subdomains	Number of iterations	Total CPU FETI (s)	Total Memory FETI (Mb)
40	67 itr.	86.6	604
80	86 itr.	65.5	521
120	86 itr.	60.4	502
140	93 itr.	69.3	500

Table 6
Performance results on a 8-processor O2000 of the two-level FETI solver equipped with the Dirichlet preconditioner and applied to the solution of the alloy wheel problem with 936,102 d.o.f. (mesh M2)

Number of subdomains	Number of iterations	Total CPU FETI (s)	Total Memory FETI (Mb)
175	98 itr.	364.4	2638
200	120 itr.	478.7	2551
250	120 itr.	490.9	2518
300	110 itr.	493.1	2481

The reader can observe that

- the iteration counts reported in Tables 5 and 6 highlight the numerical scalability of the two-level FETI method with respect to both the problem size and the number of subdomains,
- for mesh M1, the optimal number of subdomains is $N_s = 120$, for which the two-level FETI method requires 502 Mb of memory and converges in 60.4 s. Hence, for this problem and mesh M1, the two- level FETI method requires the same amount of memory as the parallel PSLDLT sparse solver which outperforms it however on 8 O2000 processors by a factor equal to 1.4,
- for mesh M2, the optimal number of subdomains is $N_s = 175$, for which the two-level FETI method requires 2638 Mb of memory and converges in 364.4 s. Hence, for this mesh, the two-level FETI method requires about the same amount of memory as the PSLDLT sparse solver, which outperforms it on 8 processors by a factor equal to 1.2. From this and the previous remark, one can reasonably conclude that for this shell problem, there exists a mesh size for and beyond which the two-level FETI method outperforms a sparse solver,
- using 8 O2000 processors, the two-level FETI method solves the 235,614 equations associated with mesh M1 in 60.4 s, and the 936,102 equations associated with mesh M2 in 364.4 s. Hence, using a fixed number of processors, the two-level FETI method solves problem M2, which is 4 times larger than problem M1, in an amount of CPU time that is 6 times larger. It follows that the two-level FETI method equipped with the Dirichlet preconditioner is not only numerically scalable, but also reasonably CPU wise scalable with respect to the mesh size – that is, for suitable values of the number of subdomains, its arithmetic complexity grows almost linearly with the size of the problem to be solved.

For $1 \leqslant N_p \leqslant 30$, the performance results of the two-level FETI method and the PSLDLT sparse solver are reported in Fig. 9 for mesh M1, and Fig. 10 for mesh M2. These results show that the two-level FETI method is more amenable to parallel processing than a sparse solver. Indeed, for this alloy wheel problem,

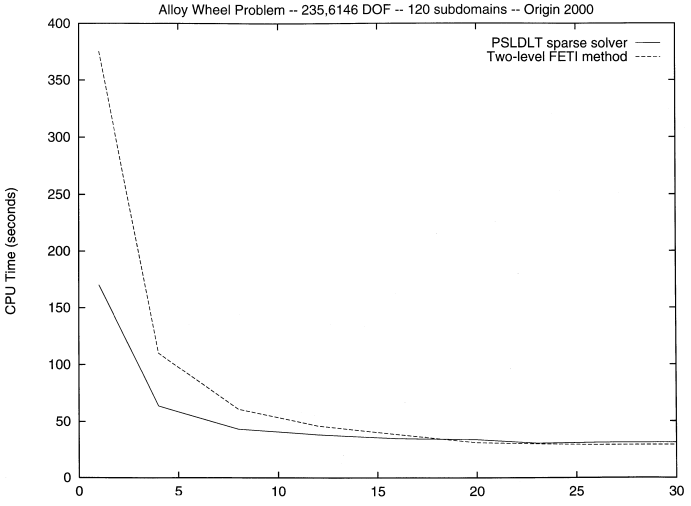

Fig. 9. Alloy wheel problem – Parallel performance results for mesh M1.

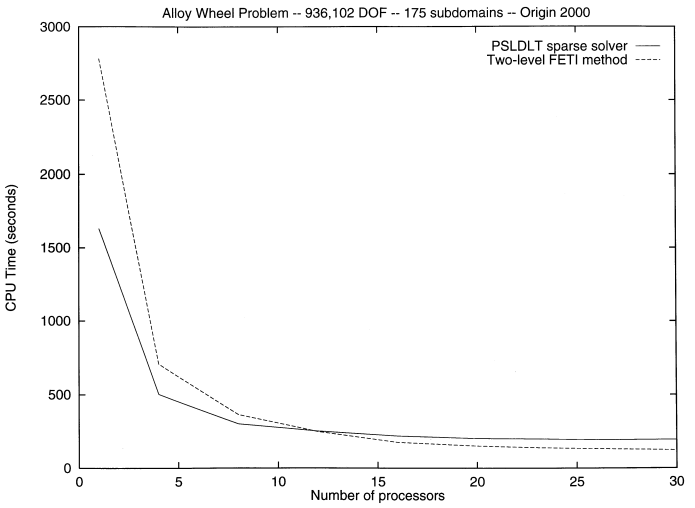

Fig. 10. Alloy wheel problem – Parallel performance results for mesh M2.

the PSLDLT sparse solver is almost twice as fast as the two-level FETI method on a single processor configuration of the O2000 machine. However, as the number of processors is increased, the two-level FETI method exhibits a better parallel scalability than the sparse solver. For mesh M1, the two-level FETI solver outperforms the PSLDLT sparse solver for $N_p \geqslant 20$, and for mesh M2, it outperforms it for $N_p \geqslant 10$.

9. Perspectives for parallel processing and domain decomposition

In the early 1980s, the first generation parallel processors were essentially distributed memory "hypercube" computers targeted at delivering 25% of the performance of a CRAY-1 system at 10% of the price. These machines never made a significant impact on the computational science infrastructure because (a) they were devoid of software, and (b) in practice, they barely sustained the performance of a VAX or an IBM mainframe of those days. However, they played a major role in training a first generation of parallel algorithms and software developers.

The appearance of the Connection Machine CM-2 in the late 1980s boosted the credibility of the potential of parallel processing. Indeed, the CM-2 was among the first parallel machines to outperform CRAY-type vector supercomputers for many scientific and a few engineering computations. Unfortunately, because of its complex architecture and programming style, the CM-2 reinforced the perception by a large segment of the computational science and engineering community of parallel processors as "exotic" machines. The late 1980s have also witnessed a refinement of earlier hypercubes into more sophisticated distributed memory computational platforms such as the intel iPSC-860 system.

During the early 1990s, distributed and cluster computing using the message-passing parallel programming paradigm flourished, thanks to the advent of first the PVM then the MPI libraries. The intel iPSC-860 system was superseded by intel's Paragon XP/S parallel processor, and IBM introduced its powerful SP2 system. Because of economic pressures, interest in tightly coupled workstation clusters as replacements for large systems also grew during this period. FDDI, HiPPi, and ATM emerged as faster alternatives to Ethernet. Numerical algorithms that exploit the scalability of this new hardware were developed, and the high potential of the parallel computing technology was finally demonstrated. Nevertheless, there was no real impact of these hardware and software developments on industrial computing, because of the following main reason. Industry uses predominantly commercial codes, and in the early and mid 1990s, parallel commercial codes were simply not available.

In the mid 1990s, tight-knit clusters of symmetric multiprocessor (SMP) shared memory systems such as the Silicon Graphics Challenge systems emerged as attractive alternatives to typical distributed memory massively parallel processors (MPP) such as the Connection Machine CM-5, the CRAY T3D, and the IBM

SP2 computers. The best features of the SMP and MPP designs were later combined to develop what is known today as the ccNUMA computer architecture, exemplified by the Origin line of servers of Silicon Graphics. Benefitting from the SMP shared memory programming model, applications require little modification for running on ccNUMA based parallel processors. This benefit is realized by the natural modularity of a single logical memory segment, represented by memory on physically separate nodes. In other words, ccNUMA parallel computing platforms can be viewed as DSM systems where cache coherence is maintained in hardware. Hence, programming these machines does not necessarily require adopting the message-passing paradigm. However, achieving higher levels of scalability on these machines requires maintaining data locality, which often requires a programming style that is similar to that of the message-passing paradigm, even if messages do not have to be explicitly exchanged between processors and/or processes. Both SMP and ccNUMA parallel computing platforms – often referred to as DSM "boxes" – offer user the possibility to run both sequential and parallel codes on the same hardware, and using the entire memory available on the system. They also offer him/her a significant flexibility in parallel programming style. For these reasons, and because these machines also offer customers the possibility of achieving application scalability at a lower cost and delivering it in an infrastructure that grows with increased business, SMP and ccNUMA parallel computing platforms have gained a noticeable acceptance and popularity in industrial computing. Several commercial codes have been ported to these platforms and partially parallelized. However, few commercial codes have been fully parallelized on these or any other parallel computing hardware, for reasons that we address later in this section.

Finally, in the late 1990s, the high-end segment of parallel computing has been re-energized by the Department of Energy's ASCI initiative. Currently, two of the three ASCI machines have a ccNUMA or ccNUMA-related architecture. The third machine is a distributed local memory massively parallel processor. The emphasis of the ASCI program is on scalability on thousands of processors for multidisciplinary applications. Hence, just when commercial computational mechanics software was about to catch up with hardware advances, new challenges have emerged.

Explicit and explicit-like research codes – that is, research codes based on numerical algorithms that require only short-range communication (i.e. communication between a subdomain and its neighbors) – have traditionally well exploited the parallel processing technology and followed closely its latest developments. On the other hand, implicit and implicit-like components of research codes, which generally require long-range communication, have traditionally lagged parallel hardware advances, especially those pertaining to local memory massively parallel processors. This is because, among other issues, implicit codes usually require the usage of an equation solver. During the last decade, the parallelization of sparse direct solvers has witnessed significant improvements. However, because of fill-in, direct solvers are not numerically scalable. Furthermore, for a constant problem size, they do not currently enjoy an acceptable level of parallel scalability, except on systems with a limited number of processors. Alternatively, simple iterative equation solvers incur explicit-like computations, and therefore scale well on massively parallel processors. However, this parallel scalability is effective only if numerical scalability is simultaneously achieved, which is hardly the case for simple iterative solvers.

One can reasonably assume that during the last two decades, most if not all computational mechanics commercial software have experienced a significant growth in size and complexity. Hence, moving today such codes to parallel computing platforms in a portable fashion requires significant resources, which perhaps explains why the majority of commercial software development houses have not yet fully embraced the parallel processing technology. Unfortunately, this in turn has and is still preventing the extra competitiveness of this technology from being used in industry. Fortunately, a few commercial finite element special purpose codes have already been parallelized. Examples include crash simulation explicit codes (see [66] for the latest developments in the parallelization of the contact problem), and some recent software modules from Centrics [67].

Domain decomposition has a lot to offer to general purpose computational mechanics software. It is a flexible software architecture by itself, and a natural route to scalable parallelism. Mesh partitioning is its most primitive form, and has been successfully used for parallelizing explicit codes. Domain decomposition is also an established strategy for constructing numerically scalable preconditioners that accelerate the convergence of iterative algorithms such as the CG and GMRES methods. Hence, domain decomposition offers the potential for blending a general purpose software architecture that scales well on massively

parallel processors, with numerically scalable iterative solvers for implicit computations. For implicit codes, the number of subdomains should not be related to the number of given processors, but to the computational complexity of the DD solver.

Today, the FETI domain decomposition method delivers a numerically scalable performance for all of the following problems: second-order elasticity and heat transfer, fourth-order plate and shell structures, elastodynamics, and acoustic scattering. It is ready for industrial computing as it is optimized for problems with multiple right-hand sides, multiple left-hand sides, and has been recently upgraded to handle multipoint constraints [68]. For three-dimensional problems discretized by solid elements, it outperforms sparse direct solvers on both sequential and parallel machines, by a factor typically ranging between three and a full-order of magnitude. For plate and shell problems, it outperforms sparse direct solvers on parallel machines. The parallel scalability of the FETI method depends on several factors including: the total size of the problem to be solved, the size of the coarse problem and therefore the specified number of subdomains, and the hardware characteristics of the given parallel processor. In the worst case, this parallel scalability is contingent on the parallel scalability of the solution of the coarse problems. For three-dimensional problems with a few million d.o.f. or less, the optimal number of subdomains is usually of the order of a thousand or less, and the solution of the coarse problems by a parallel direct method usually results in a reasonable to excellent overall parallel scalability of the FETI method, depending on the given parallel processor.

Acknowledgements

The authors acknowledge partial support by the Sandia National Laboratories under Contract BD-2435, partial support by the National Science Foundation under Grant Award ECS-9725504, and partial support by the Air Force Office of Scientific Research under Grant F49620-99-1-007.

Appendix A

The objective of this appendix is to prove that when the standard CG algorithm applied to the solution of $\mathbf{A}\mathbf{x} = \mathbf{b}$ is modified by choosing at each iteration k the search direction not in the usual Krylov space $V^{k+1}(\mathbf{w}^0, \mathbf{A})$, where $\mathbf{w}^0 = \mathbf{b} - \mathbf{A}\,\mathbf{x}^0$, but in the hybrid space $\widetilde{V}^{k+1}(\mathbf{w}^0, \mathbf{A}, \mathbf{C}) = V^{k+1}(\mathbf{P}_C^{\mathrm{T}}\mathbf{w}^0, \mathbf{P}_C\mathbf{A}\mathbf{P}_C) + \mathrm{Range}\,(\mathbf{C})$ as described in Section 3.1, the modified search directions $\tilde{\mathbf{p}}^k$ remain $\mathbf{A}$-orthogonal. This result, together with the other results derived in Section 3.1 establish that the modified CG algorithm described in Section 3.1 remains a CG algorithm. The proof goes as follows.

From Eq. (36), it follows that the iterative algorithm described in Section 3.1 starts with $\mathbf{x}^0 + \mathbf{C}(\mathbf{C}^{\mathrm{T}}\mathbf{A}\mathbf{C})^{-1}\mathbf{C}^{\mathrm{T}}(\mathbf{b} - \mathbf{A}\mathbf{x}^0)$ which is in general non-zero even when $\mathbf{x}^0 = 0$. Hence, this algorithm solves the problem $\mathbf{A}\mathbf{x} = \mathbf{b}$ by performing the splitting

$$\mathbf{x} = \mathbf{x}^0 + \mathbf{C}(\mathbf{C}^{\mathrm{T}}\mathbf{A}\mathbf{C})^{-1}\mathbf{C}^{\mathrm{T}}(\mathbf{b} - \mathbf{A}\mathbf{x}^0) + \mathbf{x}^* \tag{A.1}$$

and applying the iterations (26) to the problem

$$\mathbf{A}\mathbf{x}^* \;=\; \mathbf{P}_C^{\mathrm{T}}(\mathbf{b} - \mathbf{A}\mathbf{x}^0) \;=\; \mathbf{P}_C^{\mathrm{T}}\mathbf{w}^0. \tag{A.2}$$

The reader can check that the projector $\mathbf{P}_C$ introduced in the second of Eqs. (33) satisfies

$$\mathbf{P}_C^{\mathrm{T}}\mathbf{A}\mathbf{P}_C \;=\; \mathbf{A}\mathbf{P}_C. \tag{A.3}$$

From Eq. (33), it follows that the iterations (26) can be written as

$$\mathbf{x}^{k+1} \;=\; \mathbf{x}^k + \eta^k\tilde{\mathbf{p}}^k \;=\; \mathbf{x}^k + \eta^k\mathbf{P}_C\mathbf{p}^k. \tag{A.4}$$

This implies that the iterative algorithm described in Section 3.1 searches the solution of problem (106) in the form

$$\mathbf{x}^* \;=\; \mathbf{P}_C\hat{\mathbf{x}} \tag{A.5}$$

and computes $\hat{\mathbf{x}}$ by applying the standard CG method with search directions $\mathbf{p}^k$ to solving

$$(\mathbf{A}\mathbf{P}_C)\hat{\mathbf{x}} = \mathbf{P}_C^{\mathrm{T}}\mathbf{w}^0. \tag{A.6}$$

Using Eq. (A.3), the above problem can be rewritten as

$$(\mathbf{P}_C^{\mathrm{T}}\mathbf{A}\mathbf{P}_C)\hat{\mathbf{x}} = \mathbf{P}_C^{\mathrm{T}}\mathbf{w}^0. \tag{A.7}$$

From the above discussion, it follows that applying the modified CG algorithm described in Section 3.1 with search directions $\tilde{\mathbf{p}}^k$ to solving $\mathbf{A}\mathbf{x} = \mathbf{b}$ is equivalent to applying the standard CG method with search directions $\mathbf{p}^k$ to solving problem (A.7). Consequently, the vectors $\mathbf{p}^k$ are $(\mathbf{P}_C^{\mathrm{T}}\mathbf{A}\mathbf{P}_C)$-orthogonal, which in view of Eq. (33) implies that the modified search directions $\tilde{\mathbf{p}}^k$ are $\mathbf{A}$-orthogonal.

From the $\mathbf{A}$-orthogonality of the modified search directions $\tilde{\mathbf{p}}^k$ and Eqs. (28) and (30), we also conclude that the numerical solution algorithm presented in Section 3.1 is a true CG algorithm. This algorithm computes the solution of the problem $\mathbf{A}\mathbf{x} = \mathbf{b}$ by applying the standard CG iterations to solving $(\mathbf{P}_C^{\mathrm{T}}\mathbf{A}\mathbf{P}_C)\hat{\mathbf{x}} = \mathbf{P}_C^{\mathrm{T}}\mathbf{w}^0$. Hence, at each iteration k, in order to perform a matrix–vector product of the form $(\mathbf{P}_C^{\mathrm{T}}\mathbf{A}\mathbf{P}_C)\hat{\mathbf{x}}^k$, this CG algorithm performs a matrix–vector product of the form $\mathbf{P}_C\mathbf{x}^k$, which in view of the second of Eqs. (33) requires solving an auxiliary problem of the form $(\mathbf{C}^{\mathrm{T}}\mathbf{A}\mathbf{C})\gamma^k = -\mathbf{C}^{\mathrm{T}}\mathbf{A}\mathbf{p}^k$. Because the matrix $(\mathbf{C}^{\mathrm{T}}\mathbf{A}\mathbf{C})$ can be interpreted as a "second-level" matrix $\mathbf{A}$, we refer to the CG algorithm presented in Section 3.1 as a two-level CG method.

From Eq. (A.7), we also conclude that the two-level CG method can be described as a preconditioned CG algorithm where the preconditioner is the projector $\mathbf{P}_C$. In [40], it is shown that for any projector $\mathbf{P}_C$ of the form given in the second of Eqs. (33), the matrix $\mathbf{P}_C^{\mathrm{T}}\mathbf{A}\mathbf{P}_C$ is better conditioned than the matrix $\mathbf{A}$, which suggests that the two-level CG method can be expected to converge faster than its one-level counterpart.

References

[1] M.R. Hestnes, E. Stiefel, Method of conjugate gradients for solving linear systems, J. Res. Nat. Bur. Standards 49 (1952) 409–436.
[2] T.A. Manteuffel, Shifted incomplete Cholesky factorization, in: I.S. Duff, G.W. Stewart (Eds.), Sparse Matrix Proceedings, SIAM, 1978, pp. 41–61.
[3] M.A. Ajiz, A. Jennings, A robust incomplete Choleski-conjugate gradient algorithm, Int. J. Numer. Meth. Engrg. 20 (1984) 949–966.
[4] T.J.R. Hughes, R.M. Ferencz, J.O. Hallquist, Large-scale vectorized implicit calculations in solid mechanics on a Cray X-MP/48 utilizing EBE preconditioned conjugate gradients, Comput. Meth. Appl. Mech. Engrg. 61 (1987) 215–248.
[5] J.W. Ruge, K. Stuben, Algebraic multigrid, in: S.F. McCormick (Ed.), Multigrid Methods, Frontiers in Applied Mathematics, SIAM, 1987, pp. 73–130.
[6] P. Vanek, J. Mandel, M. Brezina, Algebraic multigrid on unstructured meshes, Computing 56 (1996) 179–196.
[7] P.E. Bjorstad, O.B. Widlund, Iterative methods for solving elliptic problems on regions partitioned into substructures, SIAM J. Numer. Anal. 23 (1986) 1097–1120.
[8] R. Glowinski, G.H. Golub, G.A. Meurant, J. Periaux (Eds.), First International Symposium on Domain Decomposition Methods for Partial Differential Equations, SIAM, Philadelphia, 1988.
[9] P. LeTallec, Domain-decomposition methods in computational mechanics, Comput. Mech. Adv. 1 (1994) 121–220.
[10] C. Farhat, F.X. Roux, Implicit parallel processing in structural mechanics, Comput. Mech. Adv. 2 (1994) 1–24.
[11] J.H. Bramble, J.E. Pasciak, A.H. Schatz, The construction of preconditioners for elliptic problems by substructuring I, Math. Comp. 47 (1986) 103–134.
[12] C. Farhat, A Lagrange multiplier based divide and conquer finite element algorithm, J. Comput. Sys. Engrg. 2 (1991) 149–156.
[13] C. Farhat, F.X. Roux, A method of finite element tearing and interconnecting and its parallel solution algorithm, Int. J. Numer. Meths. Engrg. 32 (1991) 1205–1227.
[14] J. Mandel, Balancing domain decomposition, Comm. Appl. Numer. Meth. 9 (1993) 233–241.
[15] C. Farhat, J. Mandel, F.X. Roux, Optimal convergence properties of the FETI domain decomposition method, Comput. Meth. Appl. Mech. Engrg. 115 (1994) 367–388.
[16] P. LeTallec, J. Mandel, M. Vidrascu, A Neumann–Neumann domain decomposition algorithm for solving plate and shell problems, SIAM J. Numer. Anal. 35 (1998) 836–867.
[17] C. Farhat, J. Mandel, The two-level FETI method for static and dynamic plate problems – part I: an optimal iterative solver for biharmonic systems, Comput. Meth. Appl. Mech. Engrg. 155 (1998) 129–152.
[18] C. Farhat, P.S. Chen, J. Mandel, F.X. Roux, The two-level FETI method – Part II: extension to shell problems, parallel implementation and performance results, Comput. Meth. Appl. Mech. Engrg. 155 (1998) 153–180.
[19] M. Géradin, D. Coulon, J.-P. Delsemme, Parallelization of the SAMCEF finite element software through domain decomposition and FETI algorithm, Int. J. Supercomp. Appl. 11 (1997) 286–298.

[20] C. Farhat, M. Géradin, On a component mode synthesis method and its application to incompatible substructures, Comput. Struct. 51 (1994) 459–473.
[21] C. Farhat, P.S. Chen, J. Mandel, A scalable Lagrange multiplier based domain decomposition method for implicit time-dependent problems, Int. J. Numer. Meth. Engrg. 38 (1995) 3831–3858.
[22] C. Farhat, L. Crivelli, F.X. Roux, Extending substructure based iterative solvers to multiple load and repeated analyses, Comput. Meth. Appl. Mech. Engrg. 117 (1994) 195–200.
[23] C. Farhat, D. Rixen, A new coarsening operator for the optimal preconditioning of the dual and primal domain decomposition methods: application to problems with severe coefficient jumps, in: N. Duane Melson, T.A. Manteuffel, S.F. McCormick, C.C. Douglas (Eds.), Proceedings of the Seventh Copper Mountain Conference on Multigrid Methods, 1995, pp. 301–316.
[24] D. Rixen, C. Farhat, A simple and efficient extension of a class of substructure based preconditioners to heterogeneous structural mechanics problems, Int. J. Numer. Meth. Engrg. 44 (1999) 489–516.
[25] A. de La Bourdonnaye, C. Farhat, A. Macedo, F. Magoulès, F.X. Roux, A non-overlapping domain decomposition method for the exterior Helmholtz problem, Contemporary Math. 218 (1998) 42–66.
[26] C. Farhat, A. Macedo, M. Lesoinne, F.X. Roux, F. Magoulès, A. de La Bourdonnaye, Two-level domain decomposition methods with Lagrange multipliers for the fast iterative solution of acoustic scattering problems, Comput. Meth. Appl. Mech. Engrg. 184 (2000) 213–239.
[27] M. Papadrakakis, Y. Tsompanakis, Domain decomposition methods for parallel solution of sensitivity analysis problems, Report 96-1, Institute of Structural Analysis and Seismic Research, NTUA, Athens, Greece, 1996.
[28] C. Rey, F. Lene, Generalized Krylov correction of the conjugate gradient for large-scale non-linear elasticity problems, in: Book of Abstracts of the Ninth International Conference on Domain Decomposition Methods, Bergen, Norway, 3–8 June, 1996.
[29] Q.V. Dinh, T. Fanion, Applications of dual Schur complement preconditioning to problems in computational fluid dynamics and computational electromagnetics, in: Proceedings of the Ninth International Conference on Domain Decomposition Methods, Bergen, Norway, 3–8 June, 1996.
[30] D. Soulat, F. Devries, Mechanical criteria for the subdomains decomposition: applications to heterogeneous structures and composite materials, in: Proceedings of the Ninth International Conference on Domain Decomposition Methods, Bergen, Norway, June 3–8, 1996.
[31] D. Dureissex, P. Ladevèze, A two-level and mixed domain decomposition approach for structural analysis, Contemporary Math. 218 (1998) 238–245.
[32] Z. Dostal, Domain decomposition for semicoercive contact problems, Contemporary Mathe. 218 (1998) 82–93.
[33] C. Lacour, Non-conforming domain decomposition method for plate problems, Contemporary Math. 218 (1998) 304–310.
[34] C. Farhat, P.S. Chen, F. Risler, F.X. Roux, A unified framework for accelerating the convergence of iterative substructuring methods with Lagrange multipliers, Int. J. Numer. Meth. Engrg. 42 (1998) 257–288.
[35] C. Farhat, F.X. Roux, An unconventional domain decomposition method for an efficient parallel solution of large-scale finite element systems, SIAM J. Sci. Stat. Comput. 13 (1) (1992) 379–396.
[36] D. Rixen, Dual schur complement method for semi-definite problems, Contemporary Math. 18 (1998) 341–348.
[37] R. Glowinski, G.H. Golub, G.A. Meurant, J. Périaux (Eds.), First in: International Symposium on Domain Decomposition Methods for Partial Differential Equations, SIAM, Philadelphia, PA, 1988.
[38] S.F. Ashby, T.A. Manteuffel, P.E. Saylor, A taxonomy for conjugate gradient methods, SIAM J. Numer. Anal. 27 (1990) 1542–1568.
[39] G.H. Golub, C.F. Van Loan, Matrix Computations, Johns Hopkins University Press, Baltimore, MD, 1990.
[40] Z. Dostal, Conjugate gradient method with preconditioning by projector, Int. J. Comput. Math. 23 (1988) 315–323.
[41] J. Mandel, R. Tezaur, C. Farhat, A scalable substructuring method by Lagrange multipliers for plate bending problems, SIAM J. Numer. Anal. 36 (1999) 1370–1391.
[42] D. Rixen, C. Farhat, R. Tezaur, J. Mandel, Theoretical comparison of the FETI and algebraically partitioned FETI methods, and performance comparisons with a direct sparse solver, Int. J. Numer. Meth. Engrg., submitted.
[43] C. Farhat, M. Géradin, On the general solution by a direct method of a large-scale singular system of linear equations: application to the analysis of floating structures, Int. J. Numer. Meth. Engrg. 41 (1998) 675–696.
[44] C. Farhat, P.S. Chen, Tailoring domain decomposition methods for efficient parallel coarse grid Solution and for systems with many right hand sides, Contemporary Math. 180 (1994) 401–406.
[45] M. Lesoinne, K. Pierson, An efficient FETI implementation on distributed shared memory machines with independent numbers of subdomains and processors, Contemporary Math. 18 (1998) 318–324.
[46] A. Shipley, J.C. Green, J.P. Andrews, The design and mounting of the gratings for the Far Ultraviolet Spectroscopic Explorer (FUSE), SPIE 2452 (1995) 185–196.
[47] C. Farhat, Redesigning the skyline solver for parallel/vector supercomputers, Int. J. High Speed Comput. 2 (1990) 223–238.
[48] E. Ng, private communication.
[49] W. Chan, A. George, A linear time implementation of the Reverse Cuthill Mc Kee algorithm, BIT 20 (1980) 8–14.
[50] S.W. Sloan, A Fortran program for profile and wavefront reduction, Int. J. Numer. Meth. Engrg. 28 (1989) 2651–2679.
[51] J.S. Przemieniecki, Theory of Matrix Structural Analysis, McGraw-Hill, New York, 1968.
[52] C. Farhat, A simple and efficient automatic FEM domain decomposer, Comput. Struct. 28 (5) (1988) 579–602.
[53] C. Farhat, Optimizing substructuring methods for repeated right hand sides, scalable parallel coarse solvers, and global/local analysis, in: D. Keyes, Y. Saad, D.G. Truhlar (Eds.), Domain-Based Parallelism and Problem Decomposition Methods in Computational Science and Engineering, SIAM, 1995, pp. 141–160.

[54] C. Rey, Developpement d'algorithmes paralleles de resolution en calcul non-lineaire de structures heterogenes: application au cas d'une butee acier-elastomere, Ph.D. thesis, Laboratoire de Modelisation et Mecanique des Structures, University of Paris VI, December 1994.
[55] F.X. Roux, Parallel implementation of a domain decomposition method for non-linear elasticity, in: D. Keyes, Y. Saad, D.G. Truhlar (Eds.), Domain-Based Parallelism and Problem Decomposition Methods in Computational Science and Engineering, SIAM 1995, pp. 161–175.
[56] Y. Saad, On the Lanczos method for solving symmetric linear systems with several right-hand sides, Math. Comp. 48 (1987) 651–662.
[57] W.W. Hager, Applied Numerical Linear Algebra, Prentice Hall, Englewood Cliffs, NJ, 1988.
[58] T. Belytschko, B.J. Hsieh, Non-linear transient finite element analysis with convected coordinates, Int. J. Numer. Meth. Engrg. 7 (1973) 225–271.
[59] P.G. Bergan, G. Horrigmoe, Incremental variation principles and finite element models for non-linear problems, Comput. Meth. Appl. Mech. Engrg. 7 (1976) 201–217.
[60] K.J. Bathe, S. Bolourchi, Large displacement analysis of three-dimensional beam structures, Int. J. Numer. Meth. Engrg. 14 (1979) 961–986.
[61] C.C. Rankin, F.A. Brogan, An element-independent corotational procedure for the treatment of large rotations, ASME J. Pressure Vessel Technol. 108 (1986) 165–174.
[62] B. Haugen, C.A. Felippa, A unified formulation of corotational finite elements: II. Applications, Report No. CU-CAS-95-06, Center for Aerospace Structures, Department of Aerospace Engineering, University of Colorado, Boulder, 1995.
[63] B. Haugen, Buckling and stability problems for thin shell structures using high-performance finite elements, Ph.D. Dissertation, Department of Aerospace Engineering Sciences, University of Colorado, Boulder, 1994.
[64] D.D. Davis, T. Krishnamurthy, W.J. Stroud, S.L. McCleary, An accurate non-linear finite element analysis and test correlation of a stiffened composite wing panel, American Helicopter Society National Technical Specialists' Meeting on Rotorcraft Structures, Williamsburg, Virginia, October, 1991, pp. 29–31.
[65] C. Farhat, N. Maman, G. Brown, Mesh partitioning for implicit computations via iterative domain decomposition: impact and optimization of the subdomain aspect ratio, Internat. J. Numer. Meth. Engrg. 38 (1995) 989–1000.
[66] H. Brown, S. Attaway, S. Plimpton, B. Hendrickson, Parallel strategies for crash and impact simulations, Comput. Meth. Appl. Mech. Engrg. 184 (2000) 375–390.
[67] S. Rifai, J.C. Buell, Z. Johan, T.J.R. Hughes, Automotive design applications of fluid flow simulation on parallel computing Platforms, Comput. Meth. Appl. Mech. Engrg. 184 (2000) 449–466.
[68] C. Farhat, C. Lacour, D. Rixen, Incorporation of linear multipoint constraints in substructure based iterative solvers – Part I: a numerically scalable algorithm, Int. J. Numer. Meth. Engrg. 43 (1998) 997–1016.