INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING, VOL. 38, 3831–3853 (1995)

A SCALABLE LAGRANGE MULTIPLIER BASED DOMAIN DECOMPOSITION METHOD FOR TIME-DEPENDENT PROBLEMS

CHARBEL FARHAT AND PO-SHU CHEN

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

JAN MANDEL

Center for Computational Mathematics, University of Colorado at Denver, Denver, CO 80217-3364, U.S.A

SUMMARY

We present a new efficient and scalable domain decomposition method for solving implicitly linear and non-linear time-dependent problems in computational mechanics. The method is derived by adding a coarse problem to the recently proposed transient FETI substructuring algorithm in order to propagate the error globally and accelerate convergence. It is proved that in the limit for large time steps, the new method converges toward the FETI algorithm for time-independent problems. Computational results confirm that the optimal convergence properties of the time-independent FETI method are preserved in the time-dependent case. We employ an iterative scheme for solving efficiently the coarse problem on massively parallel processors, and demonstrate the effective scalability of the new transient FETI method with the large-scale finite element dynamic analysis on the Paragon XP/S and IBM SP2 systems of several diffraction grating finite element structural models. We also show that this new domain decomposition method outperforms the popular direct skyline solver. The coarse problem presented herein is applicable and beneficial to a large class of Lagrange multiplier based substructuring algorithms for time-dependent problems, including the fictitious domain decomposition method.

KEY WORDS: domain decomposition; dynamics; nonlinear; parallel processing

1. INTRODUCTION

Non-linear transient finite element problems in structural mechanics are characterized by the semi-discrete equations of dynamic equilibrium

$$M\ddot{u} + f^{\text{int}}(u, p_c, \theta) = f^{\text{ext}} \tag{1}$$

where M is the mass matrix, u is the vector of nodal displacements, a dot superscript indicates a time derivative, f^{int} is the vector of internal nodal forces, p_c denotes a set of control parameters, θ is a function of past history of the generalized deformation gradients, and f^{ext} is the vector of external nodal forces. In many low medium frequency dynamics applications, equation (1) is most efficiently solved using an implicit time-integration scheme. In that case, a non-linear algebraic system of equations is generated at each time step. The Newton–Raphson method and its numerous variants collectively known as 'Newton-like' methods are the most popular strategies

CCC 0029–5981/95/223831–23
© 1995 by John Wiley & Sons, Ltd.

Received 2 December 1994
Revised 28 February 1995

for solving these non-linear algebraic problems. All of these algorithms require the solution of a linear algebraic system of equations of the form

$$\begin{aligned} \tilde{K}^t(u_{n+1}^{(k)})\Delta u_{n+1}^{(k+1)} &= \tilde{r}(u_{n+1}^{(k)}) \\ \Delta u_{n+1}^{(k+1)} &= u_{n+1}^{(k+1)} - u_{n+1}^{(k)} \end{aligned} \tag{2}$$

where the subscript n refers to the nth time step, the superscript k refers to the kth non-linear iteration within the current time step, $\tilde{K}^t$ is a time-dependent symmetric positive approximate tangent matrix that includes both mass and stiffness contributions, and $\Delta u_{n+1}^{(k+1)}$ and $\tilde{r}(u_{n+1}^{(k)})$ are, respectively, the vector of nodal displacement increments and the vector of out-of-balance nodal forces (dynamic residuals).

With the advent of parallel processing, domain decomposition based direct and iterative algorithms have become increasingly popular for the solution of finite element systems of equations of the form given in equation (2). Indeed, domain decomposition provides a higher level of concurrency than parallel global algebraic paradigms, and is simpler to implement on most parallel computational platforms.[1] In general, the subdomain (or substructure) equations are solved using a direct skyline or sparse factorization based algorithm, while both direct and iterative schemes have been proposed for the solution of the interface problem.[2-6] When the reduced system of equations is solved directly, the overall domain decomposition algorithm becomes a direct frontal or multifrontal method,[7-9] and its success becomes contingent on finding a good mesh partition and/or reordered system that can achieve an optimal balance between minimizing fill-in and increasing the degree of parallelism.[10-13] When the interface problem is solved iteratively—usually, via a preconditioned conjugate gradient (PCG) algorithm—the overall domain decomposition method becomes a genuine iterative solver whose success hinges on two important properties: *numerical* scalability, and *parallel* scalability. A domain decomposition based iterative method is said to be numerically scalable if the condition number κ after preconditioning does not grow or grows 'weakly' with the ratio of the subdomain size H and the mesh size h, i.e.

$$\kappa = O\left(1 + \log^{\beta}\left(\frac{H}{h}\right)\right) \tag{3}$$

with a small constant β. Numerous authors have proved equation (3) with $\beta = 2$ for various domain decomposition methods (see, for example, References 6 and 14–17 and references therein). It is well known that in order to achieve (3), a domain decomposition method must involve a *coarse problem* with a few degrees of freedom (d.o.f.) per subdomain, that must be solved at each iteration to propagate the error globally and accelerate convergence. Parallel scalability characterizes the ability of an algorithm to deliver larger speedups for a larger number of processors. In particular, parallel scalability is necessary for massively parallel processing.

The practical implications of a condition number such as that described in equation (3) are as follows:

(I1) Suppose that a given mesh is fixed, one processor is assigned to every subdomain, and the number of subdomains (which varies as $1/H$) is increased in order to increase parallelism. In that case, h is fixed and H is decreased. From equation (3), it follows that the bound on the condition number decreases and therefore *the number of iterations for convergence is expected to decrease with an increasing number of subdomains*. In particular, for a numerically scalabe domain decomposition algorithm characterized by equation (3), increasing the number of subdomains decreases the amount of work per processor an per iteration, without increasing the number of iterations for convergence.

(I2) On most distributed memory parallel processors, the total amount of available memory increases with the number of processors. When solving a certain class of problems on such parallel hardware, it is customary to define in each processor a constant subproblem size, and to increase the total problem size with the number of processors. In that case, h and H are decreased, but the ratio H/h is kept constant. From equation (3), it follows that a numerically scalable domain decomposition algorithm can solve larger problems with the same number of iterations as smaller ones, simply by increasing the number of subdomains. However, the presence of the coarse problem may limit parallel scalability for a large number of processors.

(I3) When H/h increases, that is, the number of elements assigned to a subdomain increases, the condition number will increase only slightly. Without this property, the condition number may be too large to be practical for subdomains of a size that we wish to work with. If there are only a few substructures, the conjugate gradient algorithm might still converge quickly for some domain decomposition methods because of the presence of gaps in the spectrum of the preconditioned operator; however, for large number of subdomains, the spectrum tends to fill in, and the number of iterations tends to increase.[18]

The Finite Element Tearing and Interconnecting (FETI) method developed by Farhat and Roux[1,4,5,19,20] for the solution of self-adjoint elliptic partial differential equations is a numerically scalable domain decomposition method.[18] This method was shown to outperform direct skyline solvers and several popular iterative algorithms on both sequential and parallel computing platforms.[1,4,5,20] For static structural mechanics problems, the condition number of the unpreconditioned FETI interface problem is known to grow asymptotically as[1,18]

$$\kappa = O\left(\frac{H}{h}\right) \tag{4}$$

As was observed numerically in References 1 and 18 and proved mathematically by Mandel and Tezaur,[17] for elasticity problems discretized using plane stress/strain and/or brick elements, the condition number of the FETI interface problem preconditioned with a subdomain based Dirichlet operator[1,18,23] varies as

$$\kappa = O\left(1 + \log^{\beta}\left(\frac{H}{h}\right)\right), \quad \beta \leqslant 3 \tag{5}$$

The conditioning results (4) and (5) highlight the numerical scalability of the FETI method with respect to both the mesh size h and the number of subdomains. The parallel scalability of this domain decomposition method—that is, its ability to achieve larger speedups for larger number of processors—has also been demonstrated on current massively parallel processors for several realistic structural problems.[1,21,22]

The scalability of the FETI method for time-independent problems is due to a naturally present coarse problem in the formulation of the interface problem. For floating subdomains—that is, subdomains without enough boundary conditions to prevent substructure rigid body motions—the stiffness matrix K^s is singular. In order to guarantee the solvability of the local problems associated with floating subdomains, a small auxiliary problem with at most 6 d.o.f. per subdomain is solved at each PCG iteration. In Reference 18, it was shown that this auxiliary problem indeed plays the role of a coarse problem: it provides a satisfactory mechanism for global propagation of the error, which accelerates conver gence so that the number of iterations is practically independent of the number of subdomains.

For time-dependent problems, even when a subdomains is floating, the corresponding operator $\tilde{K}^{t^s}$ is usually non-singular. For example, when the midpoint rule is used to time-integrate the subdomain equations of dynamic equilibrium, $\tilde{K}^{t^s}$, is given by

$$\tilde{K}^{t^s} = M^s + \frac{\Delta t^2}{4} K^{t^s} \tag{6}$$

where K^{t^s} is the subdomain tangent stiffness matrix. Clearly, even when K^{t^s} is singular, it is stabilized by M^s and therefore $\tilde{K}^{t^s}$ is non-singular. Hence, a straightforward extension of the FETI method to transient problems does not inherently involve any subdomain rigid body modes, and does not 'naturally' lead to any coarse problem. Such an extension has been recently described in Reference 23. As expected, it was found that the loss of the coarse problem induces the loss of numerical scalability. The transient FETI solver described in Reference 23 was still shown to outperform other algorithms for up to 32 subdomains and 32 processors. For larger number of subdomains and processors, the FETI solver without a coarse problem converges toward a Jacobi-PCG-like algorithm and therefore loses its main interest. The main objectives of this paper are: (a) to formulate a coarsening operator for the transient FETI method in order to restore the numerical scalability of its time-independent counterpart, and (b) to demonstrate the parallel scalability of the new and improved transient FETI method and its potential for solving large-scale realistic structural problems. The reader will easily verify that the coarsening strategy presented herein is applicable and beneficial to a large class of Lagrange multiplier based substructuring algorithms for time-dependent problems, including the recently proposed fictitious domain decomposition method.[24]

The remainder of this paper is organized as follows. In Section 2, we review the formulation of the basic FETI method and its application to structural dynamics problems. In Section 3, we develop a new solution algorithm for the transient FETI method using a multigridlike strategy that propagates the error globally and accelerates convergence. The new method is mathematically justified in Section 4, and interpreted in terms of structural mechanics in Section 5. Section 6 demonstrates the recovered numerical scalability of the improved FETI method for time-dependent problems, and Section 7 overviews the efficient implementation of the coarse problem solver on massively parallel processors. In Section 8, the new time-dependent FETI method is applied to the transient analysis of large-scale finite element structural models, and the measured performance results on a Paragon XP/S and an IBM SP2 parallel systems are reported and discussed. In particular, the scalability of the new transient FETI method and its superiority to direct skyline solvers are demonstrated, and the IBM SP2 parallel processor is found to be 7–10 times faster than the Paragon XP/S computer. Finally, Section 9 concludes this paper.

2. THE FETI METHOD AND ITS ENTENSION TO DYNAMICS PROBLEMS

2.1. *The FETI method for static problems*

In order to keep this paper as self-contained as possible, we begin with an overview of the basic FETI method for the solution of the system of equations

$$Ku = f \tag{7}$$

which arises in the linear (and non-linear) finite element static analysis of a given structure. In equation (7), K, u and f denote, respectively, the stiffness matrix, and the displacement and force vectors associated with the analysed structure.

Let Ω denote the volume of the structure to be analysed, and $\{\Omega^s\}_{s=1}^{s=N_s}$ denote a partitioning (tearing) of Ω into N_s non-overlapping substructures. We denote by $\Gamma^{s,q}$ the interface boundary

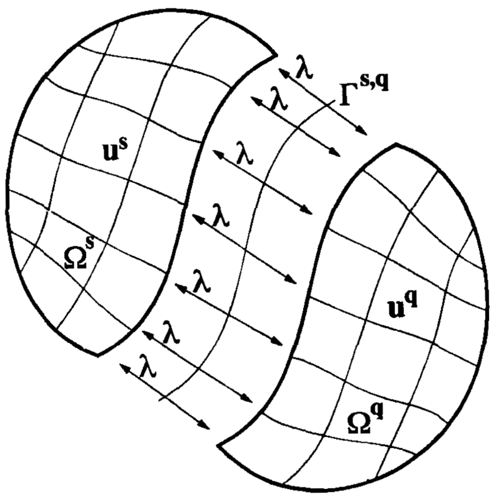

Figure 1. Tearing of a physical or computational domain

between Ω^s and a neighbouring Ω^q (Figure 1). We use an irreducible displacement formulation inside the subdomains, and independently defined Lagrange multipliers on the subdomain interfaces to join them.

The first step of the FETI method consists in writing the equations of static equilibrium for the subdom ains generated by the 'tearing' process

$$K^s u^s = f^s - B^{s^T} \lambda, \quad s = 1, \ldots, N_s \tag{8}$$

$$\sum_{s=1}^{s=N_s} B^s u^s = 0 \tag{9}$$

Here, the T superscript designates a transposed quantity, B^s is a boolean matrix with entries equal to $-1, 0, +1$ that extracts from a subdomain quantity those components that are related to the subdomain interface boundary Γ^s, and λ is the vector of Lagrange multipliers representing the traction forces needed for enforcing on the subdomain interfaces the displacement continuity (9).

In general, the partition $\{\Omega^s\}_{s=1}^{s=N_s}$ will contain N_f floating subdomains with singular stiffness matrices, and therefore the general solution of equation (8) will be given by

$$u^s = K^{s^+}(f^s - B^{s^T}\lambda) + R^s \alpha^s, \quad s = 1, \ldots, N_s \tag{10}$$

where K^{s^+} is a generalized inverse of K^s that can be quickly computed using a modified Choleski decomposition algorithm (see Reference 5, Appendix I), R^s is a rectangular matrix containing the linearly independent rigid body modes of the floating substructure Ω^s, and α^s is a vector with at most six components specifying the contribution of the subdomain rigid body modes to the substructure displacement solution. Substituting equation (10) into equation (9) leads after some algebraic manipulations to the following indefinite system of equations:

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

where

$$F_I = \sum_{s=1}^{s=N_s} B^s K^{s^+} B^{s^T}; \quad G_I = [B^1 R^1 \; \dots \; B^{N_f} R^{N_f}]$$

$$\alpha = [\alpha^{1^T} \; \dots \; \alpha^{N_f^T}]^T; \; d = \sum_{s=1}^{s=N_s} B^s K^{s^+} f^s; \quad e = [f^{1^T} R^1 \; \dots \; f^{N_f^T} R^{N_f}]^T \tag{11}$$

$$K^{s^+} = K^{s^{-1}} \quad \text{if } \Omega^s \text{ is not a floating substructure}$$

$$K^{s^+} = \text{a generalized inverse of } K^s \text{ if } \Omega^s \text{ is a floating substructure}$$

The second step of the FETI method consists in solving equations (11) via a preconditioned conjugate *projected* gradient (PCPG) algorithm with a preconditioner $\overline{F_I^{-1}}$, and the projector

$$P = I - G_I(G_I^T G_I)^{-1} G_I^T \tag{12}$$

More specially, the PCPG FETI algorithm goes as follows:[1]

```
1. Initialize
        $\lambda^0 = G_I(G_I^T G_I)^{-1} e$
        $r^0 = d - F_I \lambda^0$
2. Iterate $k = 1,2,\dots$ until convergence
    Project $w^{k-1} = P^T r^{k-1}$
    Precondition $z^{k-1} = \overline{F_I^{-1}} w^{k-1}$
    Project $y^{k-1} = P z^{k-1}$
        $\zeta^k = y^{k-1^T} w^{k-1} / y^{k-2^T} w^{k-2} \quad (\zeta^1 = 0)$
        $p^k = y^{k-1} + \zeta^k p^{k-1} \quad (p^1 = y^0)$
        $\nu^k = y^{k-1^T} w^{k-1} / p^{k^T} F_I p_k$
        $\lambda^k = \lambda^{k-1} + \nu^k p^k$
        $r^k = r^{k-1} - \nu^k F_I p_k$
```
(13)

The reader can easily check that because of the presence of the second projection step, the iterates are independent of the particular choice of the generalized inverse in equation (10).

The application of the projection operator P in (12) means that a coarse problem of the form $(G_I^T G_I)x = b$ must be solved (twice) in each FETI iteration. It was shown in References 18 and 17 that this coarse problem has the expected beneficial effect of coupling all subdomain computations and propagating the error globally, so that the condition number of the interface problem can be bounded as a function of H/h but independently of the number of subdomains.

2.2. Extension of the FETI method to non-linear dynamics problems

The extension of the FETI method to transient problems goes as follows.[23] For each subdomain, the finite element non-linear equations of dynamic equilibrium can be written as

$$M^s \ddot{u}^s + f^{\text{int}^s}(u, p_c, \theta) = f^{\text{ext}^s} - B^{s^T} \lambda \tag{14}$$

and the discrete equations expressing the continuity of the displacement field at the sub-domain interfaces remain

$$\sum_{s=1}^{s=N_s} B^s u^s = 0 \tag{15}$$

Using the notation of equation (2), the linearization of equations (14) and (15) around u^s_{n+1} can be formulated as

$$\begin{aligned} &M^s \Delta \ddot{u}^{s^{(k+1)}}_{n+1} + K^{t^s} \Delta u^{s^{(k+1)}}_{n+1} = \tilde{r}^{s^{(k)}}_{n+1} - B^{s^T} \lambda^{(k+1)}_{n+1} \quad s = 1, \ldots, N_s \\ &\sum_{s=1}^{s=N_s} B^s \Delta u^{s^{(k+1)}}_{n+1} = 0 \end{aligned} \tag{16}$$

Equations (16) are known as differential/algebraic equations (DAEs). They are more difficult to solve than the usual ordinary differential equations.[25] Let

$$\Delta v^{(k+1)}_{n+1/2} = [\Delta v^{1^{(k+1)}}_{n+1/2} \ \ldots \ \Delta v^{N_s^{(k+1)}}_{n+1/2}]$$

denote the momentum increment at iteration $k+1$ and at the midpoint between steps n and $n+1$, and let $M = [M^1 \ \ldots \ M^{N_s}]$. We have

$$\Delta v^{(k+1)}_{n+1/2} = M \Delta \dot{u}^{(k+1)}_{n+1/2} \tag{17}$$

In Reference 26, it was shown that the following time-integration algorithm for solving the DAEs (13) is second-order accurate and unconditionally stable:

1. Solve:

$$\left(M^s + \frac{\Delta t^2}{4} K^{t^s}\right) \Delta u^{s^{(k+1)}}_{n+1/2} = \frac{\Delta t^2}{4} (\tilde{r}^{s^{(k)}}_{n+1/2} - B^{s^T} \lambda^{(k+1)}) \quad s = 1, \ldots, N_s$$

$$\sum_{s=1}^{s=N_s} B^s \Delta u^{s^{(k+1)}}_{n+1/2} = 0$$

2. Update:

$$\begin{aligned} u^s_{n+1/2} &= u^{s^{(k)}}_{n+1/2} + \Delta u^{s^{(k+1)}}_{n+1/2} \\ \dot{v}_{n+1/2} &= f^{\text{ext}}_{n+1/2} - f^{\text{int}}(u_{n+1/2}, p_c, \theta) \\ u^s_{n+1} &= 2u^s_{n+1/2} - u^s_n \\ v^s_{n+1} &= v^s_n + \Delta t \dot{v}^s_{n+1/2} \end{aligned} \tag{18}$$

The computational cost of the above implicit time-integration algorithm is dominated by the cost incurred at each time step for the solution of a constrained system of the form

$$\begin{aligned} \tilde{K}^{t^s} u^s &= \tilde{g}^s - B^{s^{\mathrm{T}}} \lambda, \quad s = 1, \ldots, N_s \\ \sum_{s=1}^{s=N_s} B^s u^s &= 0 \end{aligned} \tag{19}$$

where a simplified notation has been used, and $\tilde{K}^t$ was previously defined in equation (6). After some algebraic manipulations, equations (19) above can be re-written as

$$\tilde{F}_I \lambda = \tilde{d} \tag{20}$$

where $\tilde{F}_I$ and $\tilde{d}$ are now given by

$$\tilde{F}_I = \sum_{s=1}^{s=N_s} B^s \tilde{K}^{t^{s^{-1}}} B^{s^{\mathrm{T}}}, \quad \tilde{d} = \sum_{s=1}^{s=N_s} B^s \tilde{K}^{t^{s^{-1}}} \tilde{g}^s \tag{21}$$

Note that the FETI domain decomposition method transforms the original *primal* problem described in equation (2) into a *dual* interface problem. The dual interface operator $\tilde{F}_I$ is in general symmetric positive semi-definite.[19] It has interesting spectral properties which, induce a superconvergent behaviour of a PCG algorithm applied to the solution of equation (19).[1,18,23] The parallelization of a conjugate gradient scheme applied to the solution of the dual interface problem is straightforward, because F_I is a sum of independent subdomain operators. All CG related computations can be performed in parallel on a subdomain-by-subdomain basis.

For a small number of subdomains, say $N_s < 32$, it was shown in Reference 23 that equation (21) can be solved efficiently using a PCG algorithm and either a lumped or a Dirichlet preconditioner[1,18,23] (see also Section 7 for the definition of these terms). However, for finer mesh partitions, the number of iterations for convergence was found to increase significantly with the number of subdomains. This is in contrast with the findings and the theoretical conditioning results (4) and (5) for time-independent problems. The main difference is that in the static case, the rigid body modes revealed in equation (10) and which are the building blocks of the projection operator P in equation (12) introduce a 'natural' coarse problem in the FETI methodology that must be solved at each PCG iteration. This coarse problem propagates the error globally, and accelerates convergence. In the dynamic case, the solution of the first of equations (19) does not require the introduction of the subdomain rigid body modes, and a straightforward PCG algorithm applied to the solution of the interface problem (20) does not introduce any coarse problem either. The net result for the FETI methodology is the loss in the time-dependent case of the numerical scalability exhibited and proved for time-independent problems. The main objective of this paper is to recover the numerical scalability of FETI for time-dependent problems via the construction of an 'artificial' coarse problem.

3. A NEW COARSENING STRATEGY AND THE IMPROVED TRANSIENT FETI ALGORITHM

Let $\overline{\tilde{F}_I^{-1}}$ denote a symmetric preconditioner for $\tilde{F}_I$. Since $\tilde{F}_I$ is symmetric positive definite, preconditioned conjugate gradients without any modification can be used for solving the interface problem (20). This corresponds to the original transient FETI method without a coarse problem proposed in Reference 23.

```
Original FETI for time-dependent problems
1. Initialize
        $\lambda^0 = 0$
        $r^0 = d$
2. Iterate $k = 1,2,\ldots$ until convergence
    Precondition $z^{k-1} = \overline{\tilde{F}_I^{-1}} r^{k-1}$
        $\zeta^k = z^{k-1^{\mathrm{T}}} r^{k-1} / z^{k-2^{\mathrm{T}}} r^{k-2} \quad (\zeta^1 = 0)$        (22)
        $p^k = z^{k-1} + \zeta^k p^{k-1} \quad (p^1 = z^0)$
        $\nu^k = z^{k-1^{\mathrm{T}}} r^{k-1} / p^{k^{\mathrm{T}}} \tilde{F}_I p_k$
        $\lambda^k = \lambda^{k-1} + \nu^k p^k$
        $r^k = r^{k-1} - \nu^k \tilde{F}_I p_k$
```

The improved transient FETI method we propose in this paper incorporates a coarsening operator within the preconditioning step of the above PCG algorithm. This operator will be written using a full column rank matrix Q whose columns generate the desired coarse space. The exact form of the matrix Q and its corresponding coarse space will be determined later. To accelerate the convergence of the PCG algorithm, we will require that at each iteration k, the residual r^k be orthogonal to Q:

$$Q^{\mathrm{T}} r^k = 0 \tag{23}$$

The preconditioning step consists in finding an approximate solution z of the problem $\tilde{F}_I z = r$. Using one step of a two-level multigrid algorithm where the preconditioner $\overline{\tilde{F}_I^{-1}}$ plays the role of a smoother, we can compute the preconditioned residual z as follows:

Step 1. Initialize: $z^0 = 0$.
Step 2. Compute the residual: $\tilde{r}^0 = r - \tilde{F}_I z^0 = r$.
Step 3. Perform a coarse correction: $z^1 = z^0 + Q\alpha^0$, where α^0 is such that $Q^{\mathrm{T}}\tilde{r}^1 = Q^{\mathrm{T}}(r - \tilde{F}_I(z^0 + Q\alpha^0)) = 0$. This gives $\alpha^0 = 0$ and $z^1 = z^0 = 0$.
Step 4. Perform a smoothing iteration: $z^2 = z^1 + \overline{\tilde{F}_I^{-1}}(r - \tilde{F}_I z^1)$. The result is $z^2 = \overline{\tilde{F}_I^{-1}} r$.
Step 5. Perform a second coarse correction: $z^3 = z^2 + Q\alpha^2$, where α^2 is such that $Q^{\mathrm{T}}\tilde{r}^3 = Q^{\mathrm{T}}(r - \tilde{F}_I(z^2 + Q\alpha^2)) = 0$. The outcome is $\alpha^2 = -(Q^{\mathrm{T}}\tilde{F}_I Q)^{-1} Q^{\mathrm{T}}\tilde{F}_I r$ and $z^3 = (I - Q(Q^{\mathrm{T}}\tilde{F}_I Q)^{-1} Q^{\mathrm{T}}\tilde{F}_I) z^2$.

From Steps 1–5, it follows that the approximate solution z^3 of the problem $\tilde{F}_I z = r$ can be written as

$$\begin{aligned} z^3 &= (I - Q(Q^{\mathrm{T}}\tilde{F}_I Q)^{-1} Q^{\mathrm{T}}\tilde{F}_I)\overline{\tilde{F}_I^{-1}}(I - Q(Q^{\mathrm{T}}\tilde{F}_I Q)^{-1} Q^{\mathrm{T}}\tilde{F}_I)^{\mathrm{T}} r \\ &= (I - Q(Q^{\mathrm{T}}\tilde{F}_I Q)^{-1} Q^{\mathrm{T}}\tilde{F}_I)\overline{\tilde{F}_I^{-1}} r \end{aligned} \tag{24}$$

Consequently, the operator $r \rightarrow z^3$ is symmetric and can be used in place of the preconditioner in box (22). Note that $z^1 = z^0$ and therefore Steps 2 and 3 have no effect because of the orthogonality condition $Q^T r = 0$. To guarantee that this condition is satisfied for $r = r^{k-1}$ and all k, we choose the initial solution as follows:

$$\lambda^0 = Q(Q^T \tilde{F}_I Q)^{-1} Q^T d \tag{25}$$

so that $Q^T r^0 = 0$. Then by induction, it can be easily shown that $Q^T r^k = 0$ for all k.

In the sequel, we choose $Q = G_I$ where G_I is defined in (11). Denoting

$$\tilde{P} = I - G_I (G_I^T \tilde{F}_I G_I)^{-1} G_I^T \tilde{F}_I \tag{26}$$

and using the preconditioner described in Steps 1–5 above, the new and improved transient FETI PCPG algorithm becomes as follows.

```
Improved FETI for time-dependent problems
1. Initialize
        $\lambda^0 = G_I(G_I^T \tilde{F}_I G_I)^{-1} G_I^T \tilde{d}$
        $r^0 = \tilde{d} - F_I \lambda^0$
2. Iterate $k = 1, 2, \ldots$ until convergence
    Project $w^{k-1} = P^T r^{k-1}$
    Precondition $z^{k-1} = \overline{\tilde{F}_I^{-1}} w^{k-1}$
    Project $y^{k-1} = \tilde{P} z^{k-1}$
        $\zeta^k = y^{k-1^T} w^{k-1} / y^{k-2^T} w^{k-2} \quad (\zeta^1 = 0)$
        $p^k = y^{k-1} + \zeta^k p^{k-1} \quad (p^1 = y^0)$
        $v^k = y^{k-1^T} w^{k-1} / p^{k^T} \tilde{F}_I p_k$
        $\lambda^k = \lambda^{k-1} + v^k p^k$
        $r^k = r^{k-1} - v^k \tilde{F}_I p_k$
```
(27)

Note that the first projection always gives $w^{k-1} = r^{k-1}$ because $G_I^T r^{k-1} = 0$; hence, it can be omitted in a practical implementation.

4. MATHEMATICAL JUSTIFICATION OF THE COARSENING OPERATOR

When the preconditioner for time-dependent problems $\overline{\tilde{F}_I^{-1}}$ is constructed following the same guidelines as the preconditioner for time-independent problems $\overline{F_I^{-1}}$—that is, the subdomain tangent stiffness matrix K^{t^s} is replaced by $\tilde{K}^{t^s} = M^s + (\Delta t^2/4) K^{t^s}$, it follows that $\lim_{\Delta t \to \infty} (4/\Delta t^2) \overline{\tilde{F}_I^{-1}} = \overline{F_I^{-1}}$. Keeping this in mind, the following theorem shows that the improved time-dependent FETI method with the new coarse problem converges toward the time-independent FETI method as the time step increases.

Theorem 1. Let $\lambda^k(\Delta t)$ be the iterates generated by the improved time-dependent FETI method (27) *with a time step Δt, and $\lambda^k(\infty)$ the iterates generated by the time-independent FETI* (13). *Then for all k we have*

$$\lim_{\Delta t \to \infty} \lambda^k(\Delta t) = \lambda^k(\infty) \tag{28}$$

Proof. The proof follows from the direct comparison of the FETI algorithms (13) and (27) using property

$$\lim_{\Delta t \to \infty} I - G_I(G_I^T \tilde{F}_I G_I)^{-1} G_I^T \tilde{F}_I = I - G_I(G_I^T G_I)^{-1} G_I^T \tag{29}$$

that we now prove. Let Ω^s be a floating subdomain and R_i^s be it rigid body modes, and let $\{(\mu_j^s, v_j^s)\}_{j=1}^{j=m^s}$ be all its mass normalized eigenmodes

$$K^s v_i^s = \mu_i^s M^s v_i^s, \quad v_j^{s^T} M^s v_i^s = \delta_{ij} \tag{30}$$

Let $w_i^s = M^{s^{1/2}} v_i^s$. From equations (30) it follows that

$$M^{s^{-1/2}} K^s M^{s^{-1/2}} w_i^s = \mu_i^s w_i^s, \quad w_j^{s^T} w_i^s = \delta_{ij}$$

Since both FETI algorithms (13) and (27) are invariant to the particular choice of the rigid body modes R^s, we assume without any loss of generality that the columns of R^s consist of the vectors v_i^s such that $\mu_i^s = 0$. From the spectral decomposition of the subdomain tangent dynamic stiffness matrix it follows that

$$\begin{aligned}
\tilde{K}^{t^s} &= \left(M^s + \frac{\Delta t^2}{4} K^{t^s}\right) \\
&= M^{s^{1/2}} \left(I + \frac{\Delta t^2}{4} M^{s^{-1/2}} K^{t^s} M^{s^{-1/2}}\right) M^{s^{1/2}} \\
&= \frac{\Delta t^2}{4} M^{s^{1/2}} \left(\sum_{j=1}^{j=m^s} \left(\frac{4}{\Delta t^2} + \mu_j^s\right) w_j^s w_j^{s^T}\right) M^{s^{1/2}} \\
(\tilde{K}^{t^s})^{-1} &= \frac{4}{\Delta t^2} M^{s^{-1/2}} \left(\sum_{j=1}^{j=m^s} \frac{1}{\mu_j^s + 4/\Delta t^2} w_j^s w_j^{s^T}\right) M^{s^{-1/2}} \\
&= \sum_{j=1}^{j=m^s} \frac{4/\Delta t^2}{\mu_j^s + 4/\Delta t^2} v_j^s v_j^{s^T} \\
&= R^s R^{s^T} + O\left(\frac{1}{\Delta t^2}\right), \quad \Delta t \to \infty
\end{aligned}$$

Consequently,

$$\tilde{F}_I = \sum_{s=1}^{s=N_s} B^s \tilde{K}^{t^{s^{-1}}} B^{s^T} = \sum_{s=1}^{s=N_s} B^s R^s R^{s^T} B^{s^T} + O\left(\frac{1}{\Delta t^2}\right) \quad \Delta t \to \infty,$$

which in view of equation (11) can also be written as $\tilde{F}_I = G_I G_I^T + O(1/\Delta t^2)$, $\Delta t \to \infty$. Therefore, we have

$$\begin{aligned}
\lim_{\Delta t \to \infty} I - G_I(G_I^T \tilde{F}_I G_I)^{-1} G_I^T \tilde{F}_I &= I - G_I(G_I^T G_I G_I^T G_I)^{-1} G_I^T G_I G_I^T \\
&= I - G_I(G_I^T G_I)^{-1} (G_I^T G_I)^{-1} (G_I^T G_I) G_I^T \\
&= I - G_I(G_I^T G_I)^{-1} G_I^T
\end{aligned} \tag{31}$$

Similarly one shows that $\lim_{\Delta t \to \infty} \lambda^0(\Delta t) = \lambda^0(\infty)$, which concludes the proof. □

Hence, Theorem 1 provides a mathematical justification for selecting $Q = G_I$ in (23). Since P is at the origin of the numerical scalability of the FETI method for static problems, Theorem 1 shows that $\tilde{P}$ will restore this numerical scalability for time-dependent problems.

5. MECHANICAL INTERPRETATION OF THE ALGORITHM

From a mechanical viewpoint, the contribution of the coarse space $Im(G_I)$ to the solution process can be explained as follows. First, note that the residual $r^{(k-1)}$ represents the jump of the computed displacement field $u^{(k-1)}$ across the subdomain interfaces. At the beginning of every iteration k, a specific rigid body motion is applied to each floating subdomain in order to reposition it *globally*. This rigid body correction is evaluated in the projection step where $\tilde{P}$ is applied to the residual $r^{(k-1)}$. The term $(G_I^T \tilde{F}_I G_I)^{-1}$ in $\tilde{P}$ couples all of the subdomain rigid body corrections. Therefore, it propagates the compatibility error in the displacement field across all subdomains, which accelerates convergence. After the floating subdomains are globally positioned, the *local* corrections are computed by the remainder of the PCPG algorithm.

Given that G_I has at most $6 \times N_f$ columns, $(G_I^T \tilde{F}_I G_I)$ is at most $6N_f \times 6N_f$, and the cost of each projection step is small compared to the cost of the matrix–vector product $\tilde{F}_I p_{(k)}$. However, the explicit evaluation of $(G_I^T \tilde{F}_I G_I)$ is cumbersome, and its factorization is inefficient on massively parallel processors. In Section 7, we describe a subdomain-by-subdomain implementation of the projection step that is computationally feasible and scalable to fine-grained parallel systems.

6. NUMERICAL SCALABILITY

Usually, the presence or absence of a coarse problem affects only the scalability of a domain decomposition method with respect to the number of subdomains N_s, and/or the subdomain size H. Here, we show that the improved transient FETI method with the new coarse problem is numerically scalable with respect to the number of subdomains, and is insensitive to the subdomain size. We do not offer any theoretical proof. We simply validate the asymptotic reasoning presented in Section 4 with the solution of a structural dynamics model problem. The potential of the improved time-dependent FETI method for the analysis of complex large-scale structural dynamics problems is demonstrated in Section 8 on the Paragon XP/S and IBM SP2 parallel processors.

First, we consider the transient analysis of a square membrane subjected to a planar impulse load (Figure 2). We construct three different finite element uniform discretizations using, respectively, $N_e = 6400 (h = 1/80)$, $N_e = 25\,600 (h = 1/160)$, and $N_e = 102\,400 (h = 1/320)$ four-node plane stress/strain elements with two (d.o.f.) per node. These meshes generate, respectively, $N_{eq} = 12\,800$, $N_{eq} = 51\,200$, and $N_{eq} = 20\,4800$ equations to be solved at each Newton–Raphson iteration within a time step. The coarsest mesh is partitioned into four square subdomains ($H = 1/2$), the intermediate one into 16 square subdomains ($H = 1/4$), and the finest mesh is partitioned into 64 square subdomains ($H = 1/8$). For each problem, Table I reports the number of iterations for convergence N_{itr} of the original (ORI, without a coarse problem) and new (NEW, with the coarsening operator $\tilde{P}$ (26)) FETI transient solvers, for the first Newton–Raphson outer iteration within the first time step. In all cases, the convergence results are reported for both the lumped and Dirichlet preconditioners (see Section 7), and the convergence criterion of the FETI solvers is set to

$$\frac{\|\tilde{K}^t(u_{n+1}^{(k)})\Delta u_{n+1}^{(k+1)} - \tilde{r}(u_{n+1}^{(k)})\|_2}{\|\tilde{r}(u_{n+1}^{(k)})\|_2} \leqslant 10^{-6} \tag{32}$$

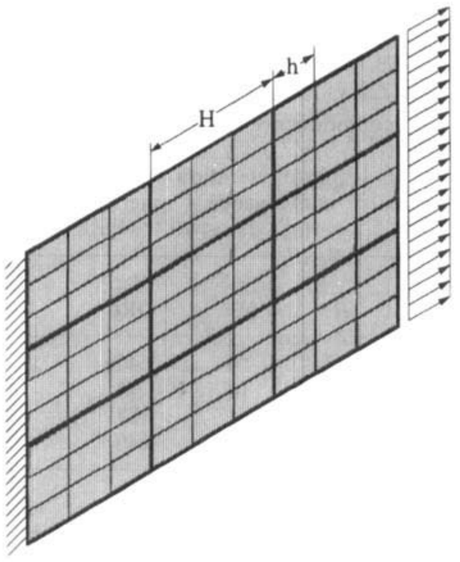

Figure 2. Finite element discretization of a membrane problem

Table I. Transient response of a square membrane to a planar impulse load: original transient FETI solver (box (22)) vs. improved transient FETI solver (box (27)); first Newton–Raphson iteration within the first time step; global convergence criterion: $\|\tilde{K}^t(u_{n+1}^{(k)})\, \Delta u_{n+1}^{(k+1)} - \tilde{r}(u_{n+1}^{(k)})\|_2 \leqslant 10^{-6}\|\tilde{r}(u_{n+1}^{(k)})\|_2$

H	h	H/h	N_{eq}	N_s	N_{itr} (ORI) lumped	N_{itr} (NEW) lumped	N_{itr} (ORI) Dirichlet	N_{itr} (NEW) Dirichlet
1/2	1/20	40	12 800	4	21	17	15	13
1/4	1/160	40	51 200	16	52	29	32	25
1/8	1/320	40	204 800	64	124	30	60	27

Note that the above convergence criterion is based on the global primal residual, and not on the dual interface residual.

For the three finite element problems described above and characterized by a constant H/h ratio, the convergence rate of the original transient FETI method with either the lumped or Dirichlet preconditioner is shown to deteriorate sublinearly with an increasing number of subdomains. On the other hand, the results depicted in Table I show that the convergence rate of the improved transient FETI method with the new coarsening operator and either the lumped or Dirichlet preconditioner is asymptotically independent of the number of subdomains. For the case of 64 subdomains, the improved FETI solver performs four times less iterations than the original one when the lumped preconditioner is used, and twice less iterations when the Dirichlet preconditioner is employed. Clearly, these results demonstrate that the improved transient FETI method is numerically scalable, and suggest that its condition number after preconditioning grows only 'weakly' with the ratio of the subdomain size H and the mesh size h.

Another popular engineering benchmark for assessing numerical scalability consists in freezing the mesh size h, and refining the mesh partition by decreasing H and therefore increasing N_s. For a fixed size problem, increasing the number of subdomains results in smaller substructures and a larger interface, which makes it harder for an iterative scheme to propagate information

Table II. Transient response of a square membrane to a planar impulse load: plane stress elements ($N_e = 102\,400$, $N_{eq} = 204\,800$); original transient FETI solver (box (22)) vs. improved transient FETI solver (box (27)); no preconditioner ($\overline{\tilde{F}_I^{-1}} = I$); effect of the projector $\tilde{P} = I - G_I(G_I^T \tilde{F}_I G_I)^{-1} G_I^T \tilde{F}_I$ on numerical scalability; first Newton–Raphson iteration within the first time step; global convergence criterion: $\| \tilde{K}^t(u_{n+1}^{(k)}) \Delta u_{n+1}^{(k+1)} - \tilde{r}(u_{n+1}^{(k)}) \|_2 \leqslant 10^{-6} \| \tilde{r}(u_{n+1}^{(k)}) \|_2$

N_{eq}	H	N_s	N_{itr} ORI	N_{itr} NEW
206 082	1/2	4	40	34
206 082	1/4	16	74	45
206 082	1/8	64	87	42
206 082	1/16	256	140	34

globally. Hence, in our case, such a test problem assesses the ability of the proposed coarsening operator to propagate the error globally and accelerate convergence. For this purpose, we consider again the transient analysis of the square membrane problem described above, and focus on the finite element discretization with $N_e = 102\,400$ elements ($h = 1/320$). We generate four different box-wise regular mesh partitions with a number of subdomains N_s varying between 4 and 256. We do not use any preconditioner ($\overline{\tilde{F}_I^{-1}} = I$) in order to highlight the sole effect of the projector $\tilde{P}$ (26). For each mesh decomposition, Table II reports the number of iterations for convergence N_{itr} of the original and improved unpreconditioned FETI transient solvers, for the first Newton–Raphson outer iteration of the first time step.

The positive effects of the coarsening operator $\tilde{P}$ on the convergence rate of the FETI method are clearly demonstrated in Table II. Whereas the original transient FETI solver exhibits a number of iterations that increases sublinearly with the number of subdomains, the improved FETI solver shows a convergence rate that improves asymptotically with an increasing number of subdomains, as it is the case for static problems.[1–18] Note that a faster convergence rate is observed for $N_s = 4$ than for $N_s = 16$, because when small number of subdomains are employed, the FETI method triggers a superconvergent behaviour of the CG scheme.[1,18]

Of course, it remains to ensure that for large-scale problems, numerical scalability is not offset by the computational overhead associated with the solution at each PCPG iteration of a coarse problem of the form $(G_I^T \tilde{F}_I G)x = b$. The issue is dealt with in the next two sections.

7. AN EFFICIENT IMPLEMENTATION AIMING AT PARALLEL SCALABILITY

In practice, numerical scalability is most interesting when parallel scalability can also be achieved. The latter property is not related to the convergence rate of an iterative domain decomposition method, but to its implementational features. Parallel scalability characterizes the ability of an algorithm to deliver larger speedups when using a larger number of processors. In particular, parallel scalability is necessary for massively parallel processing. A subdomain based iterative method that boasts both numerical and parallel scalability is clearly an 'ultimate' solution algorithm because:

(U1) from (I1), it follows that such an iterative method can compute faster solutions of a fixed mesh problem when the number of processors is increased, (U1) and from (I2), it follows that when the number of processors is increased, such an iterative method is theoretically capable of solving asymptotically larger and larger problems at a constant CPU time.

Clearly, all the computational steps in the PCPG algorithm displayed in box (27), except the projection step, can be trivially carried out on a subdomain-by-subdomain basis, and therefore are easily amenable to massively parallel processing. This includes the preconditioning step with either the lumped preconditioner[1,18,23]

$$\overline{\tilde{F}_I^{-1}}^{\mathrm{L}} = \sum_{s=1}^{s=N_s} B^s \begin{bmatrix} 0 & 0 \\ 0 & \tilde{K}^s_{\mathrm{bb}} \end{bmatrix} B^{s^{\mathrm{T}}} \tag{33}$$

or the Dirichlet preconditioner[1,18,23]

$$\overline{\tilde{F}_I^{-1}}^{\mathrm{D}} = \sum_{s=1}^{s=N_s} B^s \begin{bmatrix} 0 & 0 \\ 0 & \tilde{K}^s_{\mathrm{bb}} - \tilde{K}^{s^{\mathrm{T}}}_{\mathrm{ib}} \tilde{K}^{s^{-1}}_{\mathrm{ii}} \tilde{K}^s_{\mathrm{ib}} \end{bmatrix} B^{s^{\mathrm{T}}} \tag{34}$$

where the subscripts i and b designate, respectively, the internal and interface boundary d.o.f. Hence, special care is needed only for minimizing the computational complexity of the projection step and maximizing its parallel scalability.

Given that the matrix product

$$W = G_I^{\mathrm{T}} \tilde{F}_I \tag{35}$$

appears twice in the expression of $\tilde{P}$ (see equation (26)), a projection step of the form $z^k = \tilde{P} w^k$ is most efficiently implemented as follows:

Preprocessing:	form $W = \tilde{F}_I G_I = \tilde{F}_I [B^1 R^1 \ \ldots \ B^{N_f} R^{N_f}]$
Multiply:	$b^k = W^{\mathrm{T}} w^k$
Solve:	$(G_I^{\mathrm{T}} W) x^k = b^k$
Multiply:	$y^k = G_I x^k$
Subtract:	$z^k = w^k - y^k$

The preprocessing phase $W = \tilde{F}_I G_I$ is done only once. It involves local subdomain computations and requires interprocessor communication only between neighbouring subdomains. Assuming that a subdomain has typically six neighbours in a three-dimensional mesh partition, each with six rigid body modes, it follows that the computational overhead associated with forming explicitly W is almost equivalent to 42 additional unpreconditioned PCG iterations. Therefore, the improved transient FETI method with the new coarse problem should be used for solving those problems where the original transient FETI method[23] performs over some 80 iterations. This is typically the case for complex structural problems and large number of subdomains, say $N_s \geqslant 64$.

The computation of $W^{\mathrm{T}} w^k$ and $G_I x^k$ can also be carried out in a subdomain-by-subdomain fashion and incurs interprocessor communication between neighbouring subdomains only. Hence, special attention is needed only for the solution of the coarse problems $(G_I^{\mathrm{T}} W) x^k = b^k$.

Factoring $G_I^{\mathrm{T}} W$ once and solving $(G_I^{\mathrm{T}} W) x^k = b^k$ via repeated forward and backward substitutions is problematic for two reasons: (a) it destroys the subdomain-by-subdomain nature of all FETI computations and requires complex and cumbersome global data structures, and (b) the forward and backward substitutions that would be needed at every PCPG iteration are known to be inefficient on parallel hardware.

On the other hand, solving the coarse problems $(G_I^T W)x^k = b^k$ iteratively via a CG scheme preserves the subdomain-by-subdomain nature of the FETI computations. Indeed, a matrix–vector product of the form $y^k = (G_I^T W)x^k$ can be implemented in two steps as $\bar{y}^k = Wx^k$ and $y^k = G_I^T \bar{y}^k$, and each of these two computational steps requires only local subdomain data structures, involves only subdomain level computations, and incurs short range communication between neighbouring subdomains only. However, since a coarse problem of the form $(G_I^T W)x^k = b^k$ must be solved at each PCPG iteration, opting for an iterative strategy raises the question of how to solve iteratively and efficiently a system with a constant left-hand side matrix and repeated right-hand side vectors. This issue has been successfully addressed in Reference 27 for domain decomposition based iterative solvers. Here, we adopt the methodology proposed in Reference 27 and tailor it to the iterative solution of the repeated FETI coarse problems exposed above.

Suppose that the solution of the first encountered coarse problem $(G^T W)x^1 = b^1$ has been obtained after n^1 CG iterations. Let S^1 denote the space of the $(G^T W)$-orthogonal search directions generated during these n^1 CG iterations. If explicit re-orthogonalization is implemented in the CG algorithm, $S^{1^T}(G^T W)S^1$ is guaranteed to be a diagonal matrix. In order to compute the solution of the next coarse problem $(G^T W)x^2 = b^2$, we proceed as follows:

Step 1. We project the problem $(G^T W)x^2 = b^2$ onto S^1 and solve the trivial diagonal system $S^{1^T}(G^T W)S^1 y^{2^0} = S^{1^T} b^2$. Then, we perform a matrix–vector multiplication to obtain $x^{2^0} = S^1 y^{2^0}$. In Reference 27, it is argued that x^{2^0} is an optimal startup value for x^2 because: (a) it minimizes $x^T(G^T W)x/2 - x^T b^2$ over S^1, and (b) it is inexpensive to compute. Note that the r^1 non-zero entries of the diagonal matrix $S^{1^T}(G^T W)S^1$ are readily available from the CG solution of the previous coarse problem $(G^T W)x^1 = b^1$. Therefore, these entries can be stored and need not be re-computed.

Step 2. Next, we apply the FETI PCPG algorithm to the solution of $(G^T W)x^2 = b^2$ after it has been modified to: (a) accept x^{2^0} as a startup solution, and (b) perform the explicit orthogonalization of the new search direction and S^1 with respect to $(G^T W)$.

The solution of all subsequent coarse problems is carried out using the same two-step procedure outlined above. Essentially, the space of previous search directions is constantly enriched with the most recently computed ones, and orthogonalization with respect to $(G^T W)$ is always performed. The storage requirements associated with this methodology are minimal because it is applied to coarse and therefore small problems (see Reference 27 for further details). Because full precision is required for the solution of all coarse problems, the solution of the first one typically converges in a number of iterations equal to the size of the matrix $(G^T W)$—that is, the total number of substructure rigid body modes—and all subsequent coarse problems are solved in zero iteration, using only the optimal startup value.

8. APPLICATIONS AND PERFORMANCE RESULTS

We consider the dynamic analysis of a proposed design for a diffraction grating system that is part of a satellite borne telescope spectrograph. The grating material 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 also be able to withstand accelerations up to 15G laterally and axially during launch. The grating structure is 263 mm × 265 mm with a 165 m spherical radius cut into the face. Several designs for this system are being conceived at the University of Colorado.[28]

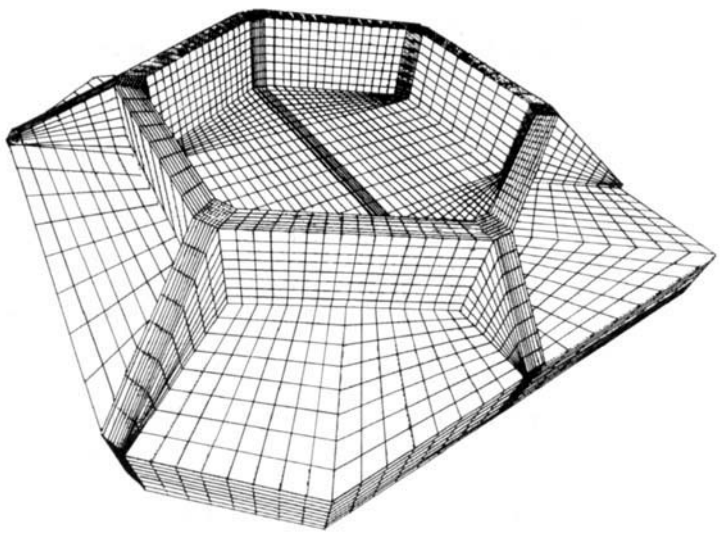

Figure 3. Finite element discretization of a diffraction grating system (model M2)

For this problem, we consider two three-dimensional finite element models[28] for the dynamic analysis on the Paragon XP/S and IBM SP2 parallel processors. Both models are constructed using 8-node brick elements. Model (M1) is relatively coarse: it contains 4416 elements and 17055 d.o.f. Model (M2) is finer: it has 35 328 elements and 120 987 d.o.f. (Figure 3). Each processor of the Paragon XP/S computer used in this work has 32 Mbytes of physical memory. On the other hand, each processor of the IBM SP2 system also used in this work is a 'wide' node with 128 Mbytes of main memory.

Several mesh partitions are generated for both finite element models using the Greedy decomposition algorithm.[29] Each mesh decomposition is also optimized for nice subdomain aspect ratios using the fact post-processing algorithm described in Reference 30. The effect of the latter optimizer on the subdomain shapes is illustrated in Plate 2.

In all cases, the time step Δt is set to

$$\Delta t = T_3/15 \tag{36}$$

where T_3 is the period of the third mode of the structure. This time step ensures an appropriate representation of the first few low modes of the structure in the computed solution, which is often sufficient for the simulation of low frequency dynamics problems. Note also that equation (36) leads to relatively large time steps for which $\tilde{K}^t$ is more ill-conditioned than for smaller time steps. We use the lumped preconditioner (33) because it is more computationally efficient that the Dirichlet one,[1,18] and the global convergence criterion given in equation (32).

We begin with a series of dynamic analyses using the coarser model M1 and 8 to 64 Paragon XP/S processors. We report in Table III the measured performance results during the first Newton–Raphson iteration of the first time-integration step, for both the original and improved time-dependent FETI methods. Throughout this section, $N_p, N_{eq}^c, T_{com}, T_W$, and T_{sol} denote, respectively, the number of processors, the size of the rigid body modes based coarse problem, the CPU time (in seconds) elapsed in interprocessor communication and synchronization, the CPU time (in seconds) elapsed in setting up $W = G_I^T \tilde{F}_I$ (35) for the new coarsening operator, and the

Table III. Diffraction grating system—model M1: 17055 d.o.f.; improved transient FETI solver (box (27)) vs. original transient FETI solver (box (22)); Lumped preconditioner $\overline{\tilde{F}_I^{-1}}^{L}$; first Newton–Raphson iteration within the first time step; global convergence criterion: $\|\tilde{K}'(u_{n+1}^{(k)})\Delta u_{n+1}^{(k+1)} - \tilde{r}(u_{n+1}^{(k)})\|_2 \leqslant 10^{-6}\|\tilde{r}(u_{n+1}^{(k)})\|_2$; parallel computations on a Paragon XP/S system

N_s, N_p	N_{eq}	N_{eq}^c	N_{itr} ORI	N_{itr} NEW	T_{com} (s) ORI	T_{com} (s) NEW	T_{sol} (s) ORI	T_W (s) NEW	T_{sol} (s) NEW
8	17055	27	72	41	31·0	61·8	237·5	200·2	397·4
16	17055	78	123	51	26·1	43·9	101·9	63·9	135·7
32	17055	174	194	49	19·9	23·1	63·1	18·6	62·7
64	17055	367	323	50	19·6	20·0	63·0	6·9	28·1

total CPU time (in seconds) corresponding to the solution of the finite element problem. Note that T_W includes the communication/synchronization time as well as the computation time corresponding to the set up of W.

From the performance results summarized in Table III, several observations are worthy noting:

(i) Whereas the convergence rate of the original transient FETI method is shown to deteriorate with the number of subdomains, the number of iterations of the new time-dependent FETI algorithm is shown to remain almost constant. This demonstrates the numerical scalability of the improved transient FETI method with respect to the number of subdomains. However, for coarse mesh partitions ($N_s = 8$ and $N_s = 16$), the original dynamic FETI solver performs more iterations than the new one but is still faster because it does not require computing $W = G_I^T \tilde{F}_I$, and its unprojected iterations are cheaper. On the other hand, for finer mesh partitions, the computational overhead associated with the new coarsening operator is offset by a superior convergence rate of the improved transient FETI solver that is also shown to outperform the original one in CPU time. As predicted earlier, the new time-dependent FETI method is interesting when the original one performs more than 80 or so iterations; for the above problem, this happens for $N_s \geqslant 32$. In particular, for $N_s = 64$, the improved transient FETI method with the new coarsening operator is reported to be twice as fast as the original one.

(ii) The communication/synchronization costs per iteration of both transient FETI solvers are shown to decrease with the number of subdomains. For the new method, the communication requirements can be separated into three categories: (a) those which are related to the matrix–vector parallel computations associated with the CG algorithm, and therefore are identical to the communication requirements of the original transient FETI method, (b) those which are associated with the setup of $W = G_I^T \tilde{F}_I$, and (c) those which are induced by each projection step. When the number of subdomains is increased, all of these communication costs decrease because the size of each subdomain decreases, and therefore the size of each message exchanged between the neighbouring subdomains also decreases.

(iii) For the new transient FETI method, the cost of constructing $W = G_I^T \tilde{F}_I$ is dominated by the cost of factoring the subdomain stiffness matrices $K^{s'}$ in order to recover the subdomain rigid body modes R^s (see Appendix A in Reference 5). Note that these computations are not needed for other purposes since the FETI operator involves the inverses of $\tilde{K}^{s'}$ and not those of $K^{s'}$. When the number of subdomains is increased, the size of the subdomain stiffnesses and therefore their factorization cost decrease, which explains the dramatic reduction of the CPU time associated with constructing W. Note also that the cost of

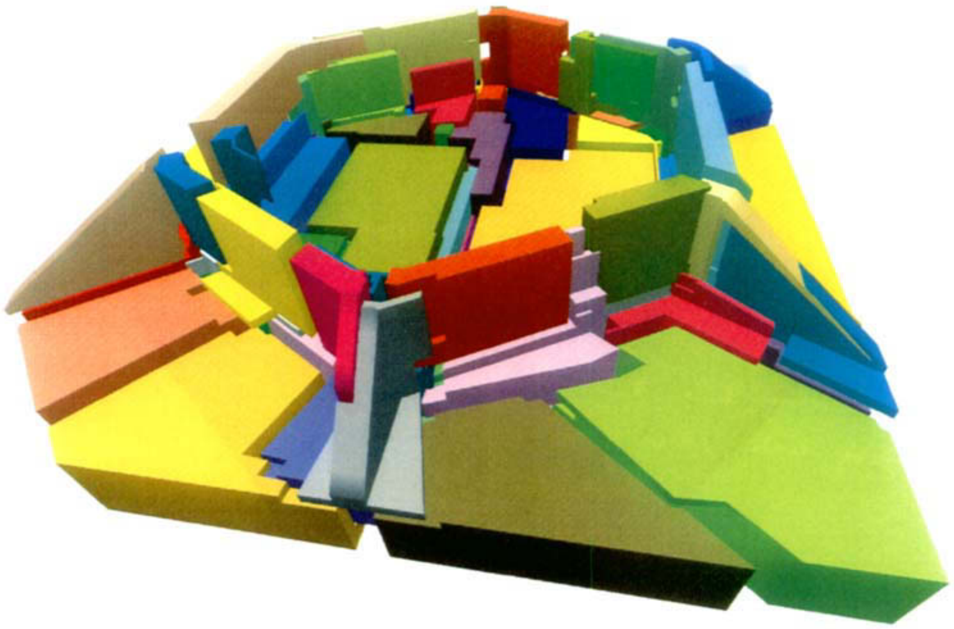

Plate 1 64-subdomain Greedy based mesh partition for model M2

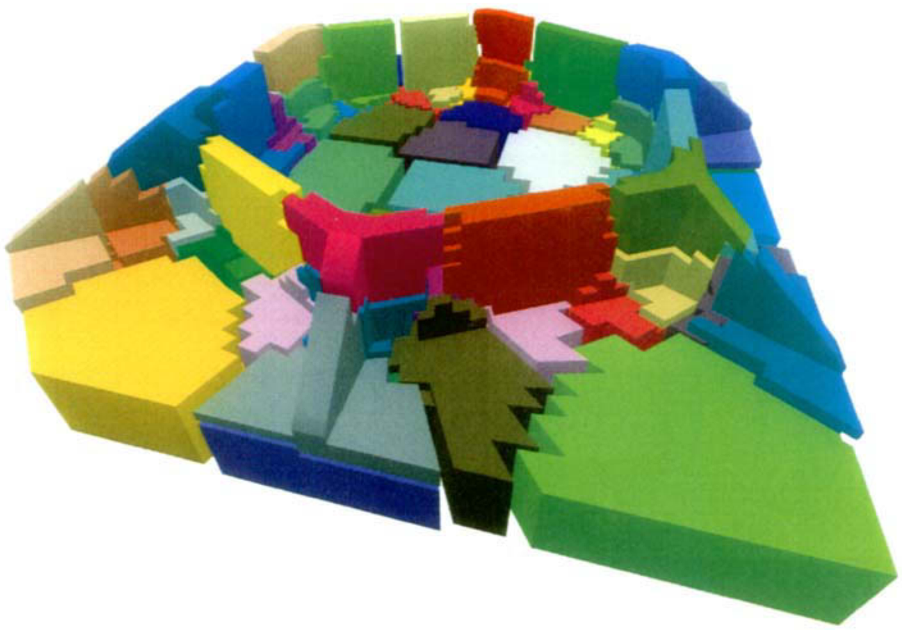

Plate 2 Optimized 64-subdomain Greedy based mesh partition for model M2

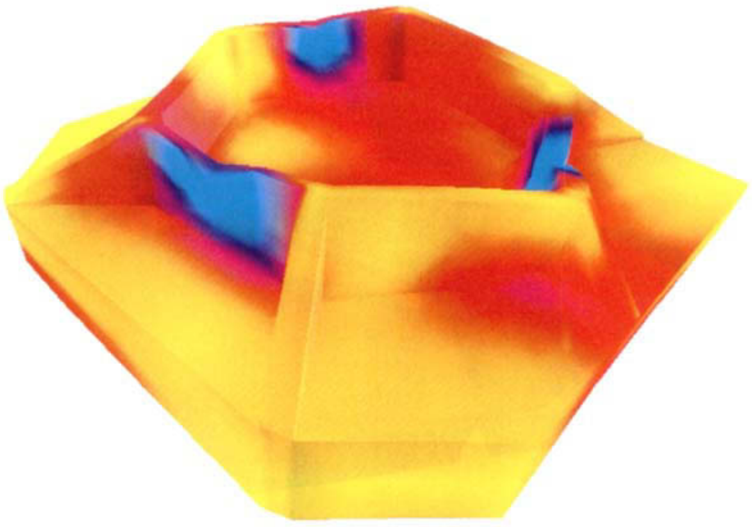

Plate 3 Amplified deflection shape and Von Mises stresses at a given time station

constructing $W = G_I^T \tilde{F}_I$ can be significantly reduced if the rigid body modes are computed from kinematics. However, the latter approach is not as robust as the factorization based one for generating the rotational rigid body modes.

(iv) For both time-dependent FETI methods, the computational complexity of each iteration and its cost significantly decrease when the number of subdomains increases, because in that case the size of each subdomain problem also decreases.

(v) For the above problem and $8 \leqslant N_s \leqslant 64$, parallel scalability is clearly demonstrated for the new time-dependent FETI method. Superlinear speedups are observed because when the number of subdomains is increased, the work per processor is decreased while the number of iterations remains almost constant.

Next, we report and discuss the performance results obtained for the dynamic analysis of the diffraction grating structure using both transient FETI methods and the finer finite element model M2 (Table IV). Originally, we intended to use 128 processors for this large-scale computation. However, we could access only 64 processors of the target Paragon XP/S computer.

Before discussing the performance results reported in Table IV, we make the following remarks. For model M1, the original transient FETI method with 64 subdomains and processors performs 323 iterations and consumes 19·6 s in interprocessor communication; hence, the average communication cost per iteration is in that case equal to 0·06 s. For model M2 and also 64 subdomains and processors, the original time-dependent FETI method performs 397 iterations and consumes 488 s in interprocessor communication; this gives an average communication cost of 1·23 s per iteration, which is more than 20 times larger than for model M1. However, the 64-subdomain mesh partitions of models M1 and M2 are topologically identical: in model M2, each subdomain has more d.o.f. than in model M1, but the same number of neighbours. Moreover, the average length of a message exchanged between two subdomains in model M2 is only 2·5 times larger than that in model M1. Given these considerations, one expects the communication cost per iteration of the original transient FETI method to increase at most by a factor equal to 3 when moving from model M1 to model M2; however, this cost is reported to increase by a factor greater than 20. We could not find a clear explanation for this strange behaviour of the Paragon XP/S system. However, we can think of two plausible causes:

1. It is possible that for a reason that is still unknown to us, some processors experience memory 'swapping' (paging to disk) when solving the M2 problem, even though the memory requirements per processor of the original transient FETI method applied to the finite element model M2 are not supposed to exceed the amount of physical memory available per processor. Since interprocessor communication is also used on the Paragon XP/S for synchronization purposes (for example, when using blocking receive calls), a load unbalance induced by memory swapping effects would increase the apparent communication costs, and consequently the total solution time.

Table IV. Diffraction grating system—model M2: 120987 d.o.f.; improved transient FETI solver (box (27)) vs. original transient FETI solver (box (22)); lumped preconditioner $\overline{\tilde{F}_I^{-1}}^L$; first Newton–Raphson iteration within the first time step; global convergence criterion: $\|\tilde{K}^t(u_{n+1}^{(k)}) \Delta u_{n+1}^{(k+1)} - \tilde{r}(u_{n+1}^{(k)})\|_2 \leqslant 10^{-6} \|\tilde{r}(u_{n+1}^{(k)})\|_2$; parallel computations on a Paragon XP/S system

N_s, N_p	N_{eq}	N_{eq}^c	N_{itr} ORI	N_{itr} NEW	T_{com} (s) ORI	T_{com} (s) NEW	T_{sol} (s) ORI	T_W (s) NEW	T_{sol} (s) NEW
64	120987	364	397	70	488	492	1499	470	991

2. In Reference 22, we have reported on the static analysis of the diffraction grating structure on the iPSC-860 system, using the time-independent FETI method and both finite element models M1 and M2. We did not observe any significant increase in the communication costs when moving from model M1 to M2, even though the communication pattern of the static FETI method is more complex than that of the original transient FETI algorithm. Therefore, a reasonable explanation for the communication costs exposed in Table IV is that the communication requirements of the finite element model M2 saturate the communication network of the Paragon XP/S system but not that of the iPSC-860 parallel processor. This is a plausible reason given that the Paragon XP/S computer has a two-dimensional grid communication network, while the iPSC-860 parallel processor has a hypercube one.

Similar observations can be made for the increase in communication costs of the new transient FETI method when moving from model M1 to model M2. In any case, the results reported in Table IV show that the improved transient FETI method with the new coarsening operator is more than 1·5 times faster than the original transient FETI algorithm.

In the structural mechanics community, it is customary to compare the performance of a proposed iterative method with that of a direct skyline solver. Indeed, direct skyline solvers are popular in computational structural mechanics, even if they do not always outperform direct sparse solvers. In this work, we compare the parallel performance of both transient FETI methods with that of the optimized parallel skyline solver described in Reference 1. After the finite element equations are renumbered for optimal skyline storage using the Reverse Cuthill–McKee algorithm, the storage requirements of the direct skyline solver are 11 million words for model M1, and 268 million words for model M2. For $N_p = 64$, this corresponds to, respectively, 1·38 and 33·5 Mbytes per processor. Given that only 24 of the 32 Mybtes of physical memory on each Paragon XP/S processor are available for the user's computations, this indicates that memory swapping will occur during the finite element analysis of model M2 via a direct skyline solver. While this affects the comparison of the performance results of the target iterative and direct methods, it is more importantly a reminder that direct skyline solvers are memory greedy.

The parallel performance results of all three solvers are reported in Table V for model M1, and in Table VI for model M2. The numbers shown between brackets ([ ]) correspond to the pure computation time—that is, the total CPU time minus the total interprocessor communication time. These numbers indicate the performance of the target solvers on sequential machines. For model M2 and $N_s = 64$, the job was terminated before the forward and backward substitutions could be completed.

Table V. Diffraction grating system—model M1: 17055 d.o.f.; improved transient FETI solver (box (27)) vs. parallel direct skyline solver; lumped preconditioner $\overline{\tilde{F}_I^{-1}}^{L}$; first Newton–Raphson iteration within the first time step; global convergence criterion: $\|\tilde{K}^t(u_{n+1}^{(k)})\,\Delta u_{n+1}^{(k+1)} - \tilde{r}(u_{n+1}^{(k)})\|_2 \leqslant 10^{-6}\|\tilde{r}(u_{n+1}^{(k)})\|_2$; parallel computations on a Paragon XP/S system

N_s, N_p	N_{eq}	T_{fac} (s)	T_{for} (s)	T_{bac} (s)	T_{tot}^{Sky} (s)	$T_{tot}^{ORI-FETI}$ (s)	$T_{tot}^{NEW-FETI}$ (s)
8	17055	262·5 [160·0]	5·5 [1·98]	11·5 [0·78]	279·5 [162·8]	237·5 [206·5]	397·4 [335·6]
16	17055	141·3 [80·3]	4·4 [0·91]	11·3 [0·38]	157·0 [81·6]	101·9 [75·8]	135·7 [91·8]
32	17055	99·1 [40·7]	4·3 [0·44]	11·5 [0·20]	114·9 [41·3]	63·1 [43·2]	62·7 [39·6]
64	17055	83·24 [20·4]	4·5 [0·22]	11·9 [0·10]	99·6 [20·7]	63·0 [43·4]	28·1 [8·1]

Table VI. Diffraction grating system—model M2: 120987 d.o.f.; improved transient FETI solver (box (27)) vs. parallel direct skyline solver; lumped preconditioner $\overline{\tilde{F}_I^{-1}}^{\mathrm{L}}$; first Newton–Raphson iteration within the first time step; global convergence criterion: $\|\tilde{K}'(u_{n+1}^{(k)})\ \Delta u_{n+1}^{(k+1)} - \tilde{r}(u_{n+1}^{(k)})\|_2 \leqslant 10^{-6}\|\tilde{r}(u_{n+1}^{(k)})\|_2$; parallel computations on a Paragon XP/S system

N_s, N_p	N_{eq}	T_{fac} (s)	T_{for}	T_{bac}	T_{tot}^{Sky} (s)	$T_{tot}^{ORI-FETI}$ (s)	$T_{tot}^{NEW-FETI}$ (s)
64	120987	3789 [1081]	NA NA	NA NA	>3789 [>1081]	1499 [1011]	991 [499]

Table VII. Diffraction grating system—model M1: 17055 d.o.f.; improved transient FETI solver (box (27)); lumped preconditioner $\overline{\tilde{F}_I^{-1}}^{\mathrm{L}}$; first Newton–Raphson iteration within the first time step; global convergence criterion: $\|\tilde{K}'(u_{n+1}^{(k)})\ \Delta u_{n+1}^{(k+1)} - \tilde{r}(u_{n+1}^{(k)})\|_2 \leqslant 10^{-6}\|\tilde{r}(u_{n+1}^{(k)})\|_2$; parallel computations on an IBM SP2 system

N_s, N_p	N_{eq}	N_{eq}^c	N_{itr} NEW	T_{com} (s) NEW	T_W (s) NEW	T_{sol} (s) NEW
8	17055	27	41	3·7	11·7	24·3
16	17055	78	51	3·5	4·2	9·3
32	17055	174	49	2·7	1·3	4·9

The performance results reported in Tables V and VI clearly show that:

(a) Except for the case of model M1 and $N_s = 8$, both parallel transient FETI methods outperform the parallel direct skyline solver. However, for model M1 and $N_s = 8$, the original transient FETI algorithm outperforms the direct skyline solver.
(b) For the relatively small finite element model M1, the parallel direct skyline solver loses to the original transient FETI method essentially because it requires more interprocessor communication.
(c) For model M1 and $N_s \geqslant 32$, and for model M2 and $N_s = 64$, the new time-dependent FETI method outperforms the parallel direct skyline solver by a factor ranging between 2 and 4. The performance results shown between brackets ([ ]) also suggest that the new time-dependent FETI method will be more than twice faster than the direct skyline solver on a sequential machine.

Finally, we report in Tables VII–IX the performance results obtained for a subset of the previous structural computations on a 64-processor IBM SP2 system. These results confirm once again the numerical and parallel scalability properties of the new transient FETI method. Note that for the structural model M1 and 32 processors, the new transient FETI method is 1·8 times faster than the direct skyline solver on the Paragon XP/S system, and 5·5 times faster than the same direct skyline solver on the IBM SP2 parallel computer. This difference can be attributed to the distinct processor-to-communication speed ratios on these two parallel systems. In all cases, the performance results reported in this paper seem to indicate that for a given number of processors, the IBM SP2 machine is 5–10 times faster than the Paragon XP/S computer for this class of unstructured finite element computations.

A sample result of the dynamic analysis of the diffraction grating system is given in Plate 3 where the displacement and Von Mises stress fields are simultaneously depicted at a given time station.

Table VIII. Diffraction grating system—model M2: 120987 d.o.f.; improved transient FETI solver (box (27)); lumped preconditioner $\overline{\tilde{F}_I^{-1}}^{\mathrm{L}}$; first Newton–Raphson iteration within the first time step; global convergence criterion: $\|\tilde{K}^t(u_{n+1}^{(k)})\,\Delta u_{n+1}^{(k+1)} - \tilde{r}(u_{n+1}^{(k)})\|_2 \leqslant 10^{-6}\|\tilde{r}(u_{n+1}^{(k)})\|_2$; parallel computations on a IBM SP2 system

N_s, N_p	N_{eq}	N_{eq}^c	N_{itr} NEW	T_{com} (s) NEW	T_W (s) NEW	T_{sol} (s) NEW
32	120987	182	53	131·8	189·7	390·4
64	120987	364	70	67·9	51·4	132·8

Table IX. Diffraction grating system—model M1: 17055 d.o.f.; improved transient FETI solver (box (27)) vs. parallel direct skyline solver; lumped preconditioner $\overline{\tilde{F}_I^{-1}}^{\mathrm{L}}$; first Newton–Raphson iteration within the first time step; global convergence criterion: $\|\tilde{K}^t(u_{n+1}^{(k)})\,\Delta u_{n+1}^{(k+1)} - \tilde{r}(u_{n+1}^{(k)})\|_2 \leqslant 10^{-6}\|\tilde{r}(u_{n+1}^{(k)})\|_2$; parallel computations on a IBM SP2 system

N_s, N_p	N_{eq}	T_{fac}	T_{for}	T_{bac} (s)	T_{tot}^{Sky} (s)	$T_{tot}^{NEW-FETI}$ (s)
16	17055	13·6 [3·4]	5·9 [0·25]	2·5 [0·02]	22·0 [3·7]	9·3 [5·8]
32	17055	12·1 [1·5]	11·9 [0·19]	2·5 [0·01]	26·5 [1·3]	4·9 [2·6]

9. CONCLUSION

In this paper, we have presented a new efficient and scalable domain decomposition method for solving implicitly linear and non-linear time-dependent problems in computational mechanics. The method is derived by adding a coarse problem to the recently proposed transient FETI substructuring algorithm in order to propagate the error globally and accelerate convergence. We have proved that in the limit for large time steps, the new method converges toward the FETI algorithm for time-independent problems. We have reported computational results that confirm that the optimal convergence properties of the time-independent FETI method are preserved in the time-dependent case. We have also presented an iterative scheme for solving efficiently the coarse problem on massively parallel processors, and demonstrated the effective scalability of the new transient FETI method with the large-scale finite element dynamic analysis on the Paragon XP/S and IBM SP2 systems of several diffraction grating finite element structural models. We have shown that for sufficiently large problems and/or fine mesh partitions, the new domain decomposition method outperforms both the original one and the popular direct skyline solver. We have also found that for an equal number of processors, the IBM SP2 parallel computer is almost an order of magnitude faster than the Paragon XP/S systems for our computations.

ACKNOWLEDGEMENTS

The first two authors acknowledge the support by RNR NAS at NASA Ames Research Center under Grant NAG 2-827. The third author acknowledges the support by the National Science Foundation under Grant ASC-9217394.

REFERENCES

1. C. Farhat and F. X. Roux, 'Implicit parallel processing in structural mechanics', *Comp. Mech. Adv.*, **2**, 1–124 (1994).
2. C. Farhat and E. Wilson, 'A new finite element concurrent computer program architecture', *Int. j. numer. methods. eng.*, **24**, 1771–1792 (1987).

3. P. E. Bjordstad and O. B. Widlund, 'Iterative methods for solving elliptic problems on regions partitioned into substructures', *SIAM J. Numer. Anal.*, **23,** 1097–1120 (1986).
4. C. Farhat, 'A Lagrange multiplier based divide and conquer finite element algorithm', *J. Comput. System Eng.*, **2,** 149–156 (1991).
5. C. Farhat and F. X. Roux, 'A method of finite element tearing and interconnecting and its parallel solution algorithm', *Int. j. numer. methods eng.*, **32,** 1205–1227 (1991).
6. J. H. Bramble, J. E. Pasciak and A. H. Schatz, 'The construction of preconditioners for elliptic problems by substructuring, I', *Math. Comp.*, **47,** 103–134 (1986).
7. I. S Duff, 'Parallel implementation of multifrontal schemes', *Parallel Comp.*, **3,** 193–204 (1986).
8. J. W. H. Liu, 'The multifrontal method for sparse matrix solution: theory and practice', *SIAM Rev.*, **34,** 82–109 (1992).
9. O. Zone, D. Vanderstraeten, P. Henriksen and R. Keunings, 'A parallel direct solver for implicit finite element problems based on automatic domain decomposition', in L. Dekker, (ed.), *Proc. Int. Conf. on Massively Parallel Processing Applications and Development*, Elsevier, Amsterdam, (in press).
10. R. E. Benner, G. R. Montry and G. G. Weigand, 'Concurrent multifrontal methods: shared memory, cache, and frontwidth issues', *Int. J. Supercomp. Appl.*, **1,** 26–44 (1987).
11. M. Lesoinne, C. Farhat and M. Géradin, 'Parallel/vector improvements of the frontal method', *Int. j. numer. methods eng.*, **32,** 1267–1282 (1991).
12. A. Pothen and C. Sun, 'A mapping algorithm for parallel sparse matrix factorization', *SIAM J. Sci. Comput.*, **4,** 1253–1257 (1993).
13. A. Pothen, E. Rothberg, H. Simon and L. Wang, 'Parallel sparse Cholesky factorization with spectral nested dissection ordering', *RNR-094-011*, NASA Ames Research Center, May 1994.
14. J. Mandel, 'Balancing domain decomposition', *Comm. appl. numer. methods*, **9,** 233–241 (1993).
15. J. Mandel, 'Hybrid domain decomposition with unstructured subdomains', in A. Quarteroni, J. Periaux, Y. A. Kuznetsov and O. Widlund (eds.), *Proc. 6th SIAM Conf. on Domain Decomposition Methods for Partial Differential Equations*, AMS, 1994, pp. 103–112.
16. M. Dryja and O. B. Widlund, 'Domain decomposition algorithms with small overlap', *SIAM J. Sci. Comput.*, **15,** 604–620 (1994).
17. J. Mandel and R. Tezaur, 'Convergence of a substructuring method with Lagrange multipliers', *UCD/CCM Report 33*, University of Colorado at Denver, 1994.
18. C. Farhat, J. Mandel and F. X. Roux, 'Optimal convergence properties of the FETI domain decomposition method', *Comput. Methods Appl. Mech. Eng.*, **115,** 367–388 (1994).
19. C. Farhat, 'A saddle-point principle domain decomposition method for the solution of solid mechanics problems', in D. E. Keyes, T. F. Chan, G. A. Meurant, J. S. Scroggs and R. G. Voigt (eds.), *Proc. 5th SIAM Conf. on Domain Decomposition Methods for Partial Differential Equations*, SIAM, 1991, pp. 271–292.
20. C. Farhat and F. X. Roux, 'An unconventional domain decomposition method for an efficient parallel solution of large-scale finite element systems', *SIAM J. Sci. Stat. Comp.*, **13,** 379–396 (1992).
21. C. Farhat, 'Optimizing substructuring methods for repeated right hand sides, scalabe parallel coarse solvers and global/local analysis', in D. Keyes, Y. Saad and D. G. Truhlar (eds.), *Domain-Based Parallelism and Problem Decomposition Methods in Computational Science and Engineering*, SIAM, 1995, pp. 141–160.
22. C. Farhat, P. S. Chen and P. Stern, 'Towards the ultimate iterative substructuring method: combined numerical and parallel scalability, and multiple load cases', *J. Comput. System Eng.*, **5**(4–6), 337–350 (1995).
23. C. Farhat, L. Crivelli and F. X. Roux, 'A transient FETI methodology for large-scale parallel implicit computations in structural mechanics', *Int. j. numer. methods eng.*, **37,** 1945–1975 (1994).
24. R. Glowinski, T. W. Pan and J. Periaux, 'A fictitious domain method for external incompressible viscous flow modeled by Navier–Stokes equations', *Comput. Methods Appl. Mech. Eng.*, in press.
25. L. Petzold, 'Differential/algebraic equations are not ODE's', *SIAM J. Sci. Stat. Comput.*, **3,** 367–384 (1982).
26. C. Farhat, L. Crivelli and M. Géradin, 'On the spectral stability of time integration algorithms for a class of constrained dynamics problems', *AIAA Paper 93-1306*, AIAA 34th Structural Dynamics Meeting, AIAA, 1993.
27. C. Farhat, L. Crivelli and F. X. Roux, 'Extending substructure based iterative solvers to multiple load and repeated analyses', *Comput. Methods Appl. Mech. Eng.*, **117,** 195–209 (1994).
28. A. Shipley, private communication.
29. C. Farhat and M. Lesoinne, 'Automatic partitioning of unstructured meshes for the parallel solution of problems in computational mechanics', *Int. j. numer. methods eng.*, **36,** 745–764 (1993).
30. C. Farhat, N. Maman and G. Brown, 'Mesh partitioning for implicit computations via iterative domain decomposition: impact and optimization of the subdomain aspect ratio', *Int. j. numer. methods eng.*, **38,** 989–1000 (1995).