NAG 2-827

CENTER FOR SPACE STRUCTURES AND CONTROLS

PROGRESS REPORT: PART III
Extending Substructure Based Iterative Solvers to Multiple Load and Repeated Analyses

by

CHARBEL FARHAT

AUGUST 1993

COLLEGE OF ENGINEERING
UNIVERSITY OF COLORADO
CAMPUS BOX 429
BOULDER, COLORADO 80309

EXTENDING SUBSTRUCTURE BASED ITERATIVE SOLVERS TO MULTIPLE LOAD AND REPEATED ANALYSES

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

Direct solvers currently dominate commercial finite element structural software, but do not scale well in the fine granularity regime targeted by emerging parallel processors. Substructure based iterative solvers — often called also domain decomposition algorithms — lend themselves better to parallel processing, but must overcome several obstacles before earning their place in general purpose structural analysis programs. One such obstacle is the solution of systems with many or repeated right hand sides. Such systems arise, for example, in multiple load static analyses and in implicit linear dynamics computations. Direct solvers are well-suited for these problems because after the system matrix has been factored, the multiple or repeated solutions can be obtained through relatively inexpensive forward and backward substitutions. On the other hand, iterative solvers in general are ill-suited for these problems because they often must restart from scratch for every different right hand side. In this paper, we present a methodology for extending the range of applications of domain decomposition methods to problems with multiple or repeated right hand sides. Basically, we formulate the overall problem as a series of minimization problems over K-orthogonal and supplementary subspaces, and tailor the preconditioned conjugate gradient algorithm to solve them efficiently. The resulting solution method is scalable, whereas direct factorization schemes and forward and backward substitution algorithms are not. We illustrate the proposed methodology with the solution of static and dynamic structural problems, and highlight its potential to outperform forward and backward substitutions on parallel computers. As an example, we show that for a linear structural dynamics problem with 11640 degrees of freedom, every time-step beyond time-step 15 is solved in a single iteration and consumes 1.0 second on a 32 processor iPSC-860 system; for the same problem and the same parallel processor, a pair of forward/backward substitutions at each step consumes 15.0 seconds.

1. Introduction

Direct solvers currently dominate commercial finite element structural software, essentially because: (a) they are robust and reliable, (b) they are versatile, (c) they work well with secondary storage, and (d) they usually outperform the class of iterative algorithms that were popular when these codes were originally developed. However, with the advent of parallel processing, alternatives to direct solvers must be researched because factorization algorithms applied to systems arising from the finite element formulation of structural problems are not scalable — that is, their performance does not necessarily increase with the number of processors. This lack of scalability is illustrated by the following analysis.

Most parallel skyline solvers that have been recently reported in the literature are closely related to the parallel active column equation solver presented in [1]. In general, clusters of columns are distributed across the processors in a block wrap fashion (Fig.1). At each step k of the factorization, column k is broadcast to all processors and the entries of row k are updated in parallel.

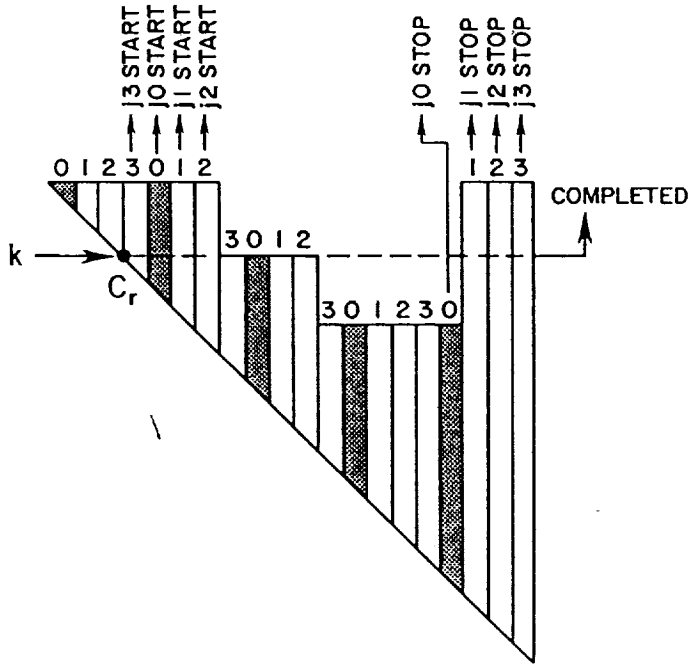

Fig. 1. Parallel skyline solver

Usually, after the mesh nodes are renumbered for optimal storage, the skyline structure becomes close to a banded one so that for all computational complexity purposes, one can reasonably assume that the system matrix is a banded one (Fig. 2).

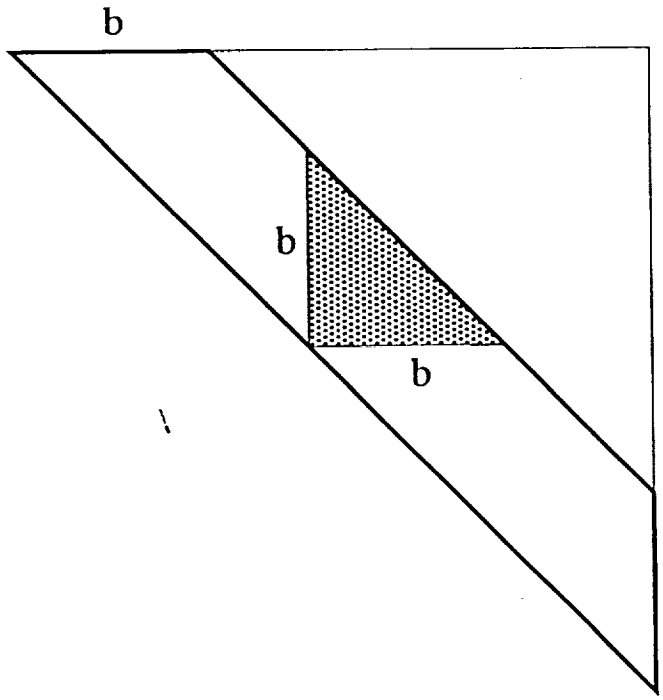

Fig. 2. Banded computational model

Let b, S_a, S_i and N_p denote respectively the system semi-bandwidth, the 64-bit floating-point arithmetic peak performance of a single processor of the parallel machine, the peak interconnect speed of that machine measured in bytes per second, and its number of processors. We assume that all real data are stored in 8-byte words. At each step of the factorization process, the computational and communication parallel time of the factorization algorithm can be evaluated as follows:

$$\begin{aligned} T_{factor} &= \frac{b^2}{N_p \times S_a} \\ T_{transmit} &= \frac{8 \times b}{S_i} \end{aligned} \tag{1}$$

Obviously, the best parallel performance is obtained when $T_{transmit} << T_{factor}$. Therefore, the saturation or balance condition is given by:

$$T_{transmit} = T_{factor} \tag{2}$$

Given a structural problem and a parallel processor, the above saturation condition holds if and only if:

$$b = 8 \times N_p \times \frac{S_a}{S_i} \tag{3}$$

Table 1 reports the values of the system semi-bandwidth that meet the saturation condition for today's emerging parallel processors. Clearly, $b = 21942$ and $b = 60262$ are unrealistic values of the matrix semi-bandwidth. For example, most finite element models related to aerospace structures have a semi-bandwidth that varies between 300 and 2000. Moreover, problems with a semi-bandwidth as large as 21942 or 60262 entail memory and CPU requirements that overwhelm even the largest of the currently available supercomputing resources.

Table 1
Saturation condition - semi-bandwidth values

Parallel processor	N_p	S_a	S_i	b
iPSC-860	128	60 Mflops	2.8 Mbytes/sec	21942
KSR-1	256	20 Mflops	8.5 Mbytes/sec	4818
CM-5	512	128 Mflops	8.7 Mbytes/sec	60262

If the saturation condition (2) is enforced for a fixed problem characterized by its semi-bandwidth b, the number of "useful" processors — that is, the number of processors beyond which any additional processor can only slow down the computations — can be computed from Eqs. (1-2) as follows:

$$N_p^u = \frac{b}{8} \times \frac{S_i}{S_a} \tag{4}$$

Table 2 reports for current massively parallel systems the number of useful processors processors for $b = 1500$. This value of the semi-bandwidth is representative of today's large-scale problems in aerospace structures.

Table 2
Number of useful processors - $b = 1500$

Parallel processor	N_p	S_a	S_i	N_p^u	N_p^u/N_p
iPSC-860	128	60 Mflops	2.8 Mbytes/sec	9	7.0 %
KSR-1	256	20 Mflops	8.5 Mbytes/sec	80	31.2 %
CM-5	512	128 Mflops	8.7 Mbytes/sec	13	2.5 %

Clearly, the relative number of processors that are kept busy by direct solvers on today's massively parallel processors is small.

In summary, the above analysis shows that parallelism in direct solvers is limited by the bandwidth and not by the size of the problem, and that direct solvers are not scalable on current parallel hardware.

REMARK 1.1. Sub-block decomposition strategies for parallel skyline factorization could provide slightly higher performance on parallel machines than distributed column methods [2]. However, the conclusions drawn above hold also for these mapping schemes.

REMARK 1.2. For some large-scale problems, sparse direct solvers can be faster than skyline ones. However, it has often been reported that sparse direct solvers are even less amenable to parallel processing than skyline ones, even on shared memory parallel processors (see, for example, [3]).

On the other hand, substructure based iterative algorithms — known also as domain decomposition methods [4] — are known to have better parallel scalability properties. Some of these algorithms also compare favorably with direct solvers even on ill-conditioned structural problems discretized with shell and beam elements [5, 6]. However, these semi-iterative algorithms must overcome several obstacles before they can be adopted by structural analysis program developers. One such obstacle is the solution of systems with many or repeated right hand sides. Such systems arise, for example, in multiple load static analyses, in implicit linear dynamics, in eigenvalue problems, and in many other structural computations. Direct solvers are well-suited for these problems because after the system matrix has been factored, the multiple or repeated solutions can be obtained through 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.

In this paper, we present a methodology for extending substructure based iterative algorithms to problems with many or repeated right hand sides. We formulate the overall problem as a series of consecutive minimization problems over K-orthogonal and supplementary subspaces, and tailor the preconditioned conjugate gradient (PCG) algorithm to solve them efficiently. Basically for each new right hand side, we first compute an optimal startup solution via the projection of the new interface problem onto an agglomerated Krylov space associated with previous right hand sides. Next, we improve this solution with an accelerated PCG algorithm where all search directions are orthogonalized with respect to the Krylov subspaces generated by previous right hand sides. The resulting solution

algorithm is scalable, whereas forward and backward substitution algorithms are not. Its convergence rate improves with the number of search directions that are stored, and therefore its performance depends on the amount of available memory. However for most structural problems, the new solution algorithm entails only a fraction of the storage requirements of direct solvers. We illustrate the proposed methodology with the solution of static and dynamic structural problems, and highlight its potential to outperform forward and backward substitutions on massively parallel computers. As an example, we show that for a linear structural dynamics problem with 11640 degrees of freedom, every time-step beyond time-step 15 is solved in a single iteration and consumes 1.0 second on a 32 processor iPSC-860 system; for the same problem and the same parallel processor, a pair of forward/backward substitutions consumes at each step 15.0 seconds. We hope that the proposed methodology will enhance the versatility of domain decomposition based iterative algorithms.

2. Problem formulation and nomenclature

For the sake of clarity, we first discuss the problem and the proposed solution methodology in the absence of any substructuring technique. In Section 4, we highlight the role of substructuring and present the substructure based solution algorithm.

Here, we are interested in solving iteratively the following problems:

$$K u_i \;=\; f_i \qquad i \;=\; 1,\; ...,\; N_{rhs} \tag{5}$$

where K, $\{f_i\}_{i\,=\,1}^{i\,=\,N_{rhs}}$, and $\{u_i\}_{i\,=\,1}^{i\,=\,N_{rhs}}$, denote respectively the stiffness matrix of a given structure, a set of N_{rhs} generalized force vectors, and the corresponding set of N_{rhs} generalized displacement vectors. Such problems arise, for example, when multiple load patterns are applied to a structure, or when a computational algorithm requires repeated solutions of a system of linear equations with the same matrix but different right hand sides.

Problems (5) above can be transformed into the following minimization problems:

$$\min_{u \in \mathcal{R}^{N_K}} \Phi_i(u) \;=\; \frac{1}{2}\, u^T K u \;-\; f_i^T u \qquad i \;=\; 1,\; ...,\; N_{rhs} \tag{6}$$

where N_K is the dimension of the stiffness matrix, $\mathcal{R}$ is the set of real numbers, and T is the transpose superscript. If each minimization problem in (6) is solved with a PCG algorithm, the following Krylov subspaces are generated:

$$\mathcal{S}_i \;=\; \{s_i^{(1)},\; s_i^{(2)},\; ...,\; s_i^{(k)},\; ...,\; s_i^{(r_i)}\} \qquad i \;=\; 1,\; ...,\; N_{rhs} \tag{7}$$

where $s_i^{(k)}$ and $r_i < N_K$ denote respectively the search direction vector at iteration k, and the number of iterations for convergence of the PCG algorithm applied to the minimization of $\Phi_i(u)$. Additionally, we introduce the following agglomerated subspaces:

$$\overline{S}_i = \bigcup_{j=1}^{j=i} S_j \qquad i = 1, \ldots, N_{rhs} \tag{8}$$

Let S_i denote the rectangular matrix associated with $\mathcal{S}_i$. From the orthogonality properties of the conjugate gradient method, it follows that:

$$S_i^T K S_i = D_i \qquad i = 1, \ldots, N_{rhs}$$

where

$$D_i = \begin{bmatrix} d_{i_1} & 0 & 0 & 0 \\ 0 & d_{i_2} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & d_{i_{r_i}} \end{bmatrix} \tag{9}$$

However, note that in general $\overline{S}_i^T K \overline{S}_i$ is not a diagonal matrix. Finally, we define $\overline{\overline{S}}_i$ as the matrix whose column vectors also span the subspace $\overline{S}_i$, but are orthogonalized with respect to the stiffness matrix K. Hence, we have:

$$range\,(\overline{\overline{S}}_i) = \overline{S}_i \qquad i = 1, \ldots, N_{rhs}$$
$$\overline{\overline{S}}_i^T K \overline{\overline{S}}_i = \overline{\overline{D}}_i \qquad i = 1, \ldots, N_{rhs}$$

where

$$\overline{\overline{D}}_i = \begin{bmatrix} \bar{\bar{d}}_{i_1} & 0 & 0 & 0 \\ 0 & \bar{\bar{d}}_{i_2} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \bar{\bar{d}}_{i_{\bar{\bar{r}}_i}} \end{bmatrix} \tag{10}$$

$$\bar{\bar{r}}_i = \sum_{j=1}^{j=i} r_j$$

3. Projection and orthogonalization

Suppose that the first problem $Ku_1 = f_1$ has been solved in r_1 PCG iterations, and that the $N_K \times r_1$ matrix S_1 associated with the Krylov subspace $\mathcal{S}_1$ is readily available.

Solving the second problem $Ku_2 = f_2$ is equivalent to solving:

$$\min_{u \in \mathcal{R}^{N_K}} \Phi_2(u) = \frac{1}{2} u^T K u - f_2^T u \tag{11}$$

If $\mathcal{R}^{N_K}$ is decomposed as follows:

$$\begin{aligned} \mathcal{R}^{N_K} &= \mathcal{S}_1 \oplus \mathcal{S}_1^* \\ dim\ (\mathcal{S}_1^*) &= N_K - r_1 \\ \mathcal{S}_1 \text{ and } \mathcal{S}_1^* &\text{ are } K-\text{orthogonal} \end{aligned} \tag{12}$$

then the solution of problem (11) can be written as:

$$\begin{aligned} u_2 &= u_2^0 + v_2 \\ &\text{where} \\ u_2^0 \in \mathcal{S}_1,\ v_2 \in \mathcal{S}_1^* &\text{ and } u_2^{0^T} K v_2 = v_2^T K u_2^0 = 0 \end{aligned} \tag{13}$$

From Eqs. (11-13), it follows that u_2^0 is the solution of the minimization problem:

$$\min_{u \in \mathcal{S}_1} \Phi_2(u) = \frac{1}{2} u^T K u - f_2^T u \tag{14}$$

and v_2 is the solution of the minimization problem:

$$\min_{v \in \mathcal{S}_1^*} \Phi_2(v) = \frac{1}{2} v^T K v - f_2^T v \tag{15}$$

First, we consider the solution of problem (14). Since $u_2^0 \in \mathcal{S}_1$, there exists a $y_2^0 \in \mathcal{R}^{r_1}$ such that:

$$u_2^0 = S_1 y_2^0 \tag{16}$$

Substituting Eq. (16) into Eq. (14) leads to the following minimization problem:

$$\min_{y \in \mathcal{R}^{r_1}} \tilde{\Phi}_2(y) = \frac{1}{2} y^T S_1^T K S_1 y - f_2^T S_1^T y \tag{17}$$

whose solution y_2^0 is given by:

$$\begin{aligned} S_1^T K_1 S_1 \; y_2^0 &= \tilde{f}_2 \\ \text{where} & \\ \tilde{f}_2 &= S_1^T f_2 \end{aligned} \tag{18}$$

From Eq. (9), it follows that the system of equations (18) is diagonal. Hence, the components $[y_2^0]_j$ of y_2^0 can be simply computed as follows:

$$[y_2^0]_j = \frac{[\tilde{f}_2]_j}{d_{1_j}} \qquad j = 1, \ldots, r_1 \tag{19}$$

Next, we turn to the solution of problem (15) via a PCG algorithm. Since the decomposition (13) requires v_2 to be K-orthogonal to u_2^0, at each iteration k, the search directions $s_2^{(k)}$ must be explicitly K-orthogonalized to S_1. This entails the computation of modified search directions $\hat{s}_2^{(k)}$ as follows:

$$\hat{s}_2^{(k)} = s_2^{(k)} + \sum_{q=1}^{q=r_1} \alpha_q s_1^{(q)}$$

where

$$\alpha_q = -\frac{s_1^{(q)^T} K s_2^{(k)}}{s_1^{(q)^T} K s_1^{(q)}} = -\frac{s_2^{(k)^T} K s_1^{(q)}}{s_1^{(q)^T} K s_1^{(q)}} \tag{20}$$

Except for the above modifications, the original PCG algorithm is unchanged.

In summary, once the first problem $Ku_1 = f_1$ has been solved in r_1 PCG iterations and the $N_K \times r_1$ matrix S_1 associated with the Krylov subspace $\mathcal{S}_1$ has been stored, the second problem $Ku_2 = f_2$ is solved in two steps as follows:

Step 1. K is projected onto S_1 and the resulting diagonal problem $S_1^T K S_1 y_2^0 = S_1^T f_2$ is trivially solved in N_K floating-point operations. Next, the partial solution $u_2^0 = S_1 y_2^0$ is formed. This partial solution u_2^0 is an optimal startup value for u_2 because: (a) it minimizes $u^T K u/2 - u^T f_2$ over $S_1 \subset \mathcal{R}^{N_K}$, and (b) it is inexpensive to compute. Note that the r_1 non-zero entries of the diagonal matrix $S_1^T K S_1$ are automatically computed during the PCG solution of the first problem $Ku_1 = f_1$. Therefore, these entries can be stored and need not be re-computed.

Step 2. The basic PCG algorithm is applied to the solution of $Ku_2 = f_2$ after it is modified to: (a) accept u_2^0 as a startup solution, and (b) orthogonalize the search directions $s_2^{(k)}$ and S_1 with respect to K.

The generalization to the case of N_{rhs} right hand sides of the two-step solution procedure described above goes as follows. Suppose that $i^* - 1 < N_{rhs}$ consecutive problems $Ku_i = f_i$, $i = 1, \ldots, i^* - 1$ have been solved with a PCG algorithm modified as described above, and that the $N_K \times \bar{\bar{r}}_{i^*-1}$ matrix $\overline{\overline{S}}_{i^*-1}$ associated with the K-orthogonalized agglomerated Krylov subspace $\overline{S}_{i^*-1}$ has been stored. The i^*-th problem $Ku_{i^*} = f_{i^*}$ is solved in two steps. First, K is projected onto $\overline{\overline{S}}_{i^*-1}$, and an optimal startup solution $u^0_{i^*} = \overline{\overline{S}}_{i^*-1} y^0_{i^*}$ is computed via the trivial solution of the diagonal system $\overline{\overline{S}}^T_{i^*-1} K \overline{\overline{S}}_{i^*-1} y^0_{i^*} = \overline{\overline{S}}^T_{i^*-1} f_{i^*}$. Next, the PCG algorithm is applied to the solution of $Ku_{i^*} = f_{i^*}$ with $u^0_{i^*}$ as a startup solution, and all search directions $s^{(k)}_{i^*}$ are K-orthogonalized to $\overline{\overline{S}}_{i^*-1}$.

Clearly, the performance of the solution method proposed here depends on the performance of the preconditioner used in the conjugate gradient algorithm. However, it should be noted that from the decomposition (12), it follows that if after the PCG solution of some problem indexed by $1 < i^* < N_{rhs}$ the dimension $\bar{\bar{r}}_{i^*}$ of the K-orthogonalized Krylov subspace $\overline{S}_{i^*}$ becomes equal to the size of one problem N_K, the proposed solution method will converge in theory to a direct solver. In that case, each remaining problem $Ku_i = f_i$, $i = i^* + 1$, N_{rhs} will be solved *directly* (zero iteration) and economically as follows (see Eq. (10)):

$$
\begin{aligned}
& u_i = \overline{\overline{S}}_{i^*} \, y^0_i \qquad i = i^* + 1, \ \ldots, \ N_{rhs} \\
& \text{where} \\
& [y^0_i]_j = \frac{[\overline{\overline{S}}_{i^*} \, f_i]_j}{\bar{\bar{d}}_{i^*_j}} \qquad j = 1, \ \ldots, \ N_K
\end{aligned}
\tag{21}
$$

In practice, the superconvergence behavior outlined above will be reached before the point where $\bar{\bar{r}}_{i^*} = N_K$, and each of the PCG solutions of the remaining $N_{rhs} - i^*$ problems will converge in two or three iterations.

REMARK 3.1. From Eqs. (11-19), it follows that if two right hand sides f_i and f_{i+1} are proportional, the solution of the problem with right hand side f_{i+1} is $u_{i+1} = u^0_{i+1} = \overline{\overline{S}}_i y^0_{i+1}$, and therefore, this solution will be found in zero iteration.

4. Primal and dual substructuring methods

Despite its elegance and simplicity, the methodology described in Section 3 can be impractical when applied to the global solution of the problems $Ku_i = f_i$, $i = 1, N_{rhs}$. Indeed, during the PCG solution of the first few problems

— that is, before superconvergence can be reached —, the cost of the orthogonalizations implied by Eq. (20) can offset the benefits of convergence acceleration via the optimal startup solution and the modified search directions $\hat{s}_i^{(k)}$. Moreover, storing every search direction $\hat{s}_i^{(k)}$ and the corresponding matrix-vector product $K\hat{s}_i^{(k)}$ can significantly increase the memory requirements of the basic PCG algorithm.

However, for symmetric elliptic problems, most powerful preconditioners are based on domain decomposition [4, 5], and for such preconditioners, the CPU and memory drawbacks outlined above become less important as discussed below.

Domain decomposition methods are essentially substructuring methods where different solution algorithms can be applied to the local and interface problems. In these methods, the substructures are seldom physical ones. In general, they are obtained by partitioning the global mesh into a number of subdomains following some specified criterion [7]. In structural mechanics, the local problems correspond to the static condensation of the substructure internal degrees of freedom, usually via a direct method, and the interface problem corresponds to the evaluation of the substructure interface boundary degrees of freedom, usually via a PCG algorithm. Depending on the choice of the substructure interface unknowns, two substructuring methods can be formulated:

1. The primal substructuring method. This is the classical substructuring method where the structural degrees of freedom are partitioned into internal ones, designated here by the subscript I, and interface boundary ones, designated here by the subscript B. The substructure equations of equilibrium are given by:

$$\begin{aligned} K_{II}^s\, u_I^s &= f_I^s - K_{IB}^s\, u_B \qquad s = 1, \ldots, N_s \\ \sum_{s=1}^{s=N_s} K_{IB}^{sT}\, u_I^s + \sum_{s=1}^{s=N_s} K_{BB}^s\, u_B &= \sum_{s=1}^{s=N_s} f_B^s \end{aligned} \tag{22}$$

where N_s denotes the number of substructures. After the internal degrees of freedom are eliminated via static condensation, Eqs. (22) above are transformed into the following interface problem:

$$[\sum_{s=1}^{s=N_s} K_{BB}^s - K_{IB}^{sT} K_{II}^{s^{-1}} K_{IB}^s\,]\, u_B = \sum_{s=1}^{s=N_s} f_B^s - K_{IB}^{sT} K_{II}^{s^{-1}} f_I^s \tag{23}$$

which can be efficiently solved with a substructure based PCG algorithm [5]. Because the interface problem (23) is written in terms of a "primal" displacement variable u_B, we refer to this substructuring method as a primal method.

2. The dual substructuring method. This substructuring method is based on a variational principle [6,8]. The interface unknowns are chosen as Lagrange multipliers representing surface tractions at the substructure interface boundaries λ. The substructure equations of equilibrium are given by:

$$\begin{aligned} K^s u^s &= f^s - B^{s^T} \lambda \qquad s = 1, \ldots, N_s \\ \sum_{s=1}^{s=N_s} B^s u^s &= 0 \end{aligned} \tag{24}$$

where B^s is the finite element matrix associated with the spatial discretization of the Lagrange multipliers λ. The substructure displacement fields u^s can be eliminated from Eqs. (25) to obtain an interface problem written in terms of the "dual" variables λ:

$$[\sum_{s=1}^{s=N_s} B^s K^{s^+} B^{s^T}] \lambda = \sum_{s=1}^{s=N_s} B^s K^{s^+} f^s \tag{25}$$

where K^{s^+} is a generalized inverse of K^s that becomes identical to $K^{s^{-1}}$ when substructure s is non-floating [6]. The dual interface problem (25) can be also solved efficiently with a substructure based PCG algorithm [6,8,9].

The size of the primal interface problem (23) is directly related to the number of interface nodes, while the size of the dual interface problem (25) is directly related to the discretization type and order of the Lagrange multipliers λ. In general, the size of the dual problem (25) is less or equal to the size of the primal problem (23) [10]. Therefore, whether a primal or dual substructuring method is chosen, the size of the interface problem is less or equal to the number of interface nodes multiplied by the maximum number of degrees of freedom per node. Moreover, computational efficiency in the iterative solution — especially on parallel processors — dictates choosing the number of substructures such that the number of interface nodes does not exceed 10 to 20 % of the total number of nodes in the finite element mesh.

Therefore, the proposed methodology for solving the multiple problems $Ku_i = f_i$, $i = 1, \ldots, N_{rhs}$ is computationally feasible when the PCG algorithm is used in a substructuring context because:

- the additional memory requirements entailed by the storage of the search directions $\hat{s}_i^{(k)}$ and the matrix vector products $K\hat{s}_i^{(k)}$ are proportional only to the reduced size of the interface problem.

- the cost of the orthogonalizations implied by Eq. (20) are negligible compared to the cost of the forward and backward substitutions that are required at each iteration k for the evaluation of the matrix-vector product $K_{II}^{s^{-1}}[K_{IB}^{s} \; u_B^{(k)}]$ (primal method), or the matrix-vector product $K^{s^+}[B^{s^T} \; \lambda^{(k)}]$ (dual method).

We have previously shown that during the solution via an iterative dual substructuring method of a single problem $Ku = f$, the spectral pattern of the governing matrix (25) causes a rapid loss of orthogonality between the computed search directions [11,12]. This result also holds for the solution of a single problem $Ku = f$ via an iterative primal substructuring method with a Neumann preconditioner [5]. The loss of orthogonality between the search directions has a disastrous consequence on the convergence rate of the PCG algorithm and can be compensated only by an explicit re-orthogonalization procedure that is identical to that of Eq. (20). In reference [12], we have shown for several large-scale structural problems that this re-orthogonalization procedure improves significantly the convergence rate of the PCG algorithm and reduces drastically its total CPU time, without dramatically increasing its storage requirements.

Hence, the orthogonalization procedure (20) has two positive and synergistic effects on the solution via a substructure based iterative methodology of the multiple problems $Ku_i = f_i$, $i = 1$, ..., N_{rhs}: (a) when invoked to K-re-orthogonalize the columns of S_1, it accelerates the convergence of the solution of the first problem, and thus reduces the number of search directions that must be stored for the solution of the next problem, and (b) when applied to K orthogonalize $s_i^{(k)}$ and $\overline{\overline{S}}_{i-1}$, it accelerates the convergence of the i-th problem and prepares the evaluation of the optimal startup solution of the i-th + 1 problem.

REMARK 4.1. In practice, the number of search directions that are stored for orthogonalization is determined by the memory space that is left available after all other storage requirements of the structural analysis have been satisfied. When only a few directions can be stored, a partial orthogonalization procedure is performed.

Finally, it should be noted that the solution methodology proposed herein involves essentially dot products and matrix-vector multiplications; therefore, it scales well in the fine granularity regime targeted by emerging parallel processors, whereas direct forward and backward substitutions do not [1].

5. Applications

Here, we apply the methodology described in the previous sections to the solution of repeated systems arising from the static analysis of a stiffened wing panel under multiple load conditions, and the linear transient analysis using an implicit time-integration scheme of a line-pinched membrane with a circular hole. The first problem illustrates the effect on convergence of the number of stored search directions, and quantifies the corresponding memory requirements. The second problem highlights the superconvergence effects of the methodology in the presence of a large number of repeated systems, and demonstrates its parallel scalability. The substructure based PCG method selected in both applications is the FETI method [6,12] with the "lumped" preconditioner [12,14]. In all cases, the convergence criterion is set to:

$$\frac{||K_i u_i - f_i||_2}{||f_i||_2} \leq 10^{-3} \tag{26}$$

We report performance results on the CRAY Y-MP and the iPSC-860 supercomputers that demonstrate the fast convergence and computational efficiency of the proposed methodology, and highlight its parallel scalability.

5.1. Multiple load static analysis

First, we consider the static analysis of a stiffened wing panel from the V22 tiltrotor aircraft [13]. The corresponding finite element model (Fig. 3) contains 9486 nodes, 18272 triangular shell elements with 6 d.o.f. per node, and a total number of 56916 degrees of freedom. The panel is clamped at one end and two different load cases are applied at the other end: a uniformly distributed bending load perpendicular to the main plane of the panel (L1), and a similar load as in (L1) but with a non-uniform spatial distribution. The finite element mesh is decomposed into 16 subdomains using the Greedy algorithm [7]. The resulting mesh partition contains 747 interface nodes. All computations are performed on a single processor CRAY Y-MP.

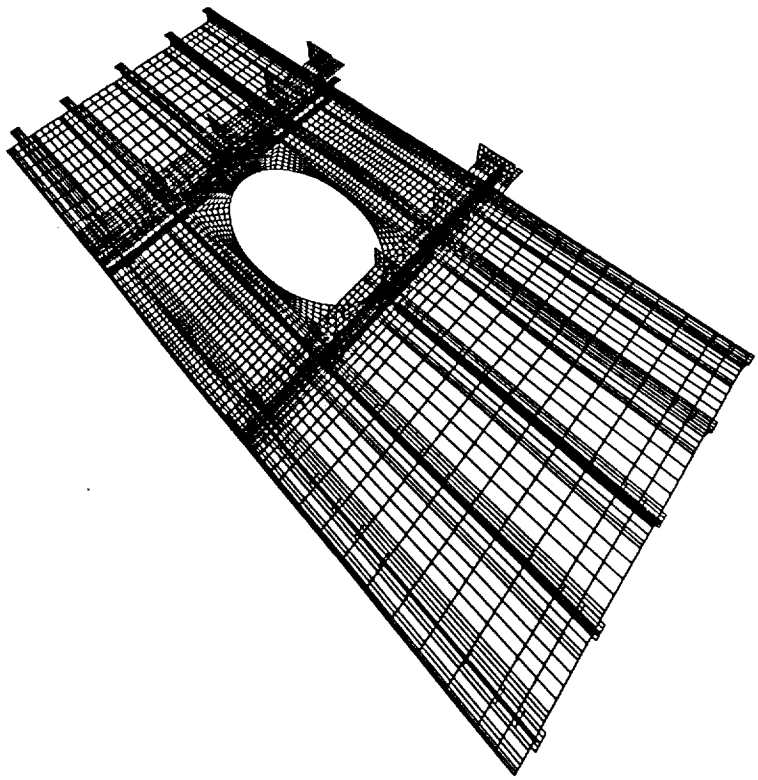

Fig. 3. Stiffened wing panel from the V22 tiltrotor aircraft

The structural problem corresponding to load case (L1) is solved using the basic FETI iterative method. The size of the dual interface problem is 4482. Convergence is achieved after 359 iterations and 120 seconds CPU. For this problem, the FETI code runs at 135 Mflops and requires 15.0 million 64 bit words (MW). An optimized direct solver running at 235 Mflops requires 51 MW and computes the solution in 199 seconds. The performance of the proposed methodology for load case (L2) is reported in Table 3 as a function of the number of stored search directions. This number is expressed as a percentage of the 359 iterations computed during the iterative solution of load case (L1).

Table 3
Stiffened wing panel from the V22 tiltrotor aircraft
Performance of the solution method for load case (L2)

# of stored (L1) directions (% of 359)	0	20	40	60	80	100
Total memory requirements (MW)	15.0	14.9	15.6	15.3	15.3	15.3
# of iterations	370	300	230	150	90	60
CPU time for orthogonalizations (secs)	5.2	5.0	4.5	3.3	2.3	1.8
Total CPU time (secs)	124.0	100.0	77.0	51.0	30.5	20.5

The results reported in Table 3 are in agreement with the theory presented in Section 3. The greater is the number of stored (L1) search directions, the better is the startup solution (17-19), the smallest is the number of iterations needed for solving the minimization problem (15), and the fastest is the convergence of the overall methodology. When all of the search directions generated during the solution of problem (L1) are stored, the solution time for problem (L2) is reduced to less than 17% of the solution time for problem (L1). The reader can also observe that for this two load case problem, the total memory requirements of the solution procedure is almost independent of the number of stored (L1) search directions. Indeed, increasing the number of stored (L1) search directions increases the storage requirements corresponding to the K orthogonalization of $s_2^{(k)}$ and $\overline{\overline{S}}_1$. However, it also minimizes the number of iterations for the solution of problem (L2), and therefore reduces the storage requirements associated with the K re-orthogonalization of S_2 (we recall the reader that the latter re-orthogonalization is an intrinsic component of the FETI method that accelerates the convergence of the PCG algorithm for a single dual interface problem (25)). In all cases, the memory requirements of the proposed iterative solution procedure do not exceed 31% of the memory requirements of a direct skyline solver.

While load cases (L1) and (L2) are not proportional, they solicit the same structural degrees of freedom. For this reason, we also report results for an analysis involving load cases (L1) and L(3), where load case (L3) corresponds to a non-uniformly distributed shearing load at the non-clamped end of the panel. Clearly, load case (L3) excites many modes of the structure that are not excited by load case (L1), which explains the slight degradation in performance reported in Table 4. However, the overall conclusions drawn for the combination (L1-L2) are shown to hold for the combination (L1-L3).

Table 4
Stiffened wing panel from the V22 tiltrotor aircraft
Performance of the solution method for load case (L3)

# of stored (L1) directions (% of 359)	0	20	40	60	80	100
Total memory requirements (MW)	15.0	14.9	15.7	15.4	15.3	15.3
# of iterations	380	310	240	165	115	85
CPU time for orthogonalizations (secs)	5.5	5.4	4.9	3.8	3.1	2.6
Total CPU time (secs)	126.0	105.0	82.0	57.0	40.0	30.5

5.2. Implicit linear dynamic analysis

Next, we consider the transient analysis via an implicit time-integration scheme of a line-pinched membrane with a circular hole. The membrane is discretized in 5680 4-node elements and 11640 degrees of freedom (Fig. 4). The finite element mesh is partitioned into 32 subdomains. The size of the dual interface problem is 1892 — that is, 16.25% of the size of the global problem. The transient analysis is carried out on a 32 processor iPSC-860 system. After all of the usual finite element storage requirements are allocated, there is enough memory left to store a total number of 891 search directions. This number corresponds to 47% of the size of the dual interface problem.

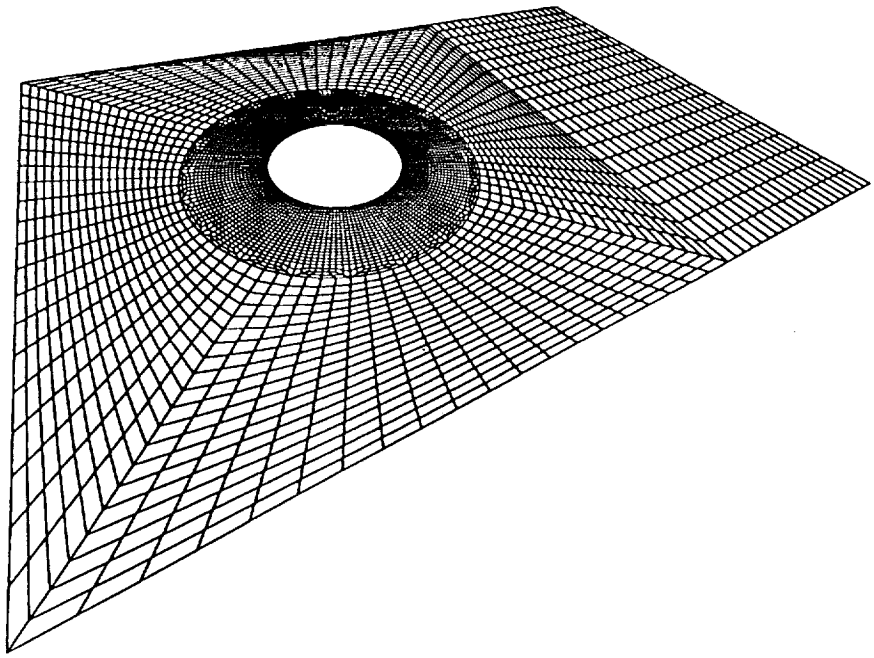

Fig. 4. Line-pinched membrane with a circular hole

The system of equations arising at the first time step is solved in 322 iterations using the FETI transient method [12]. After 3 time steps, 435 search directions are accumulated (Fig. 5) and only 20 iterations are needed for solving the fourth linear system of equations (Fig. 6). After 16 time steps, the total number of accumulated search directions is only 536 — that is, only 28% of the size of the dual interface problem, and superconvergence is triggered: all subsequent time steps are solved in 2 or 3 iterations (Fig. 7) and in less than 1.0 second CPU (Fig. 8).

When a direct solver is applied to the above problem, at each time step, the pair of forward/backward substitutions consumes 15.0 seconds on the same 32 processor iPSC-860. Therefore, the proposed solution methodology is clearly an excellent alternative to repeated forward/backward substitutions on distributed memory parallel processors.

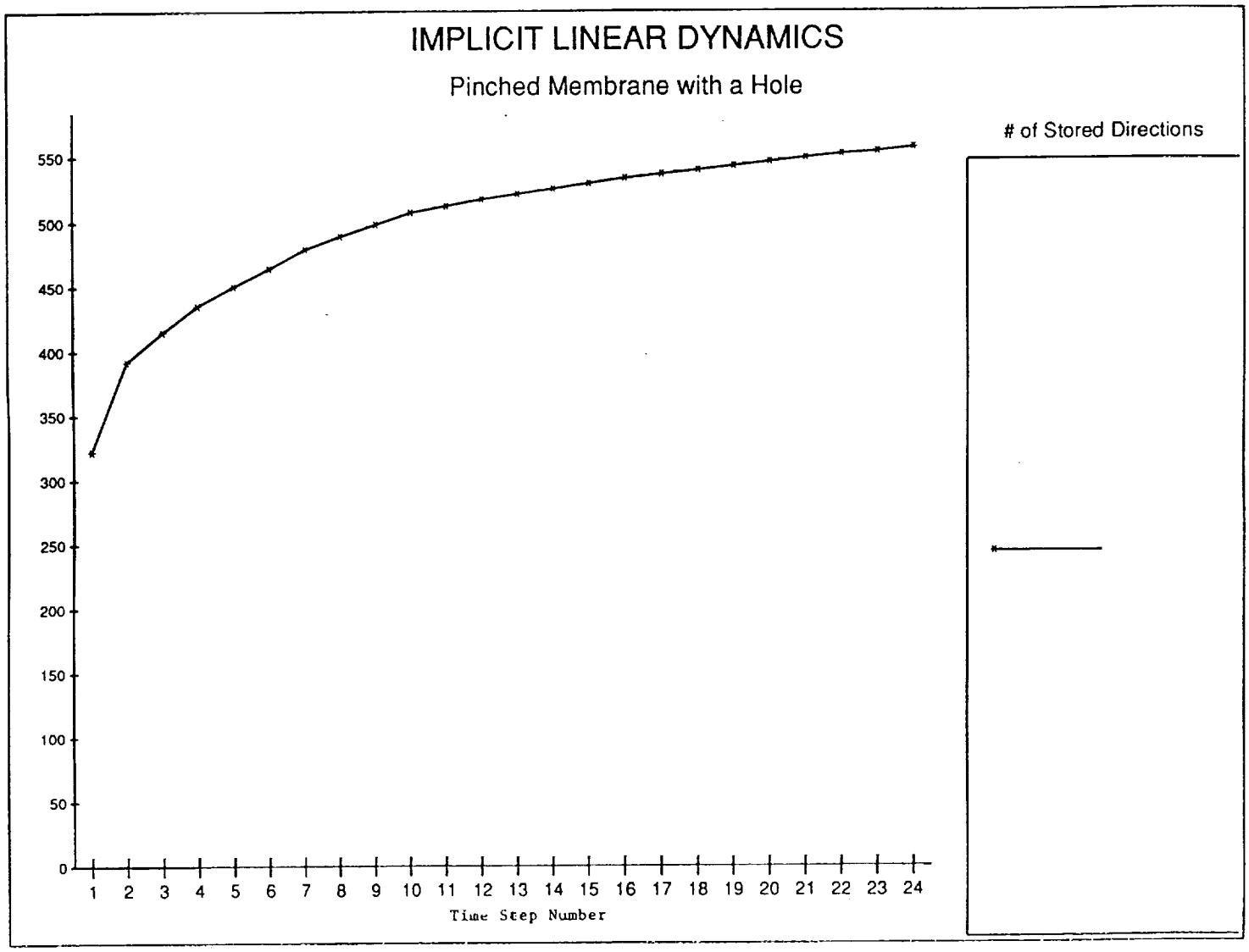

Fig. 5. Accumulation of search directions

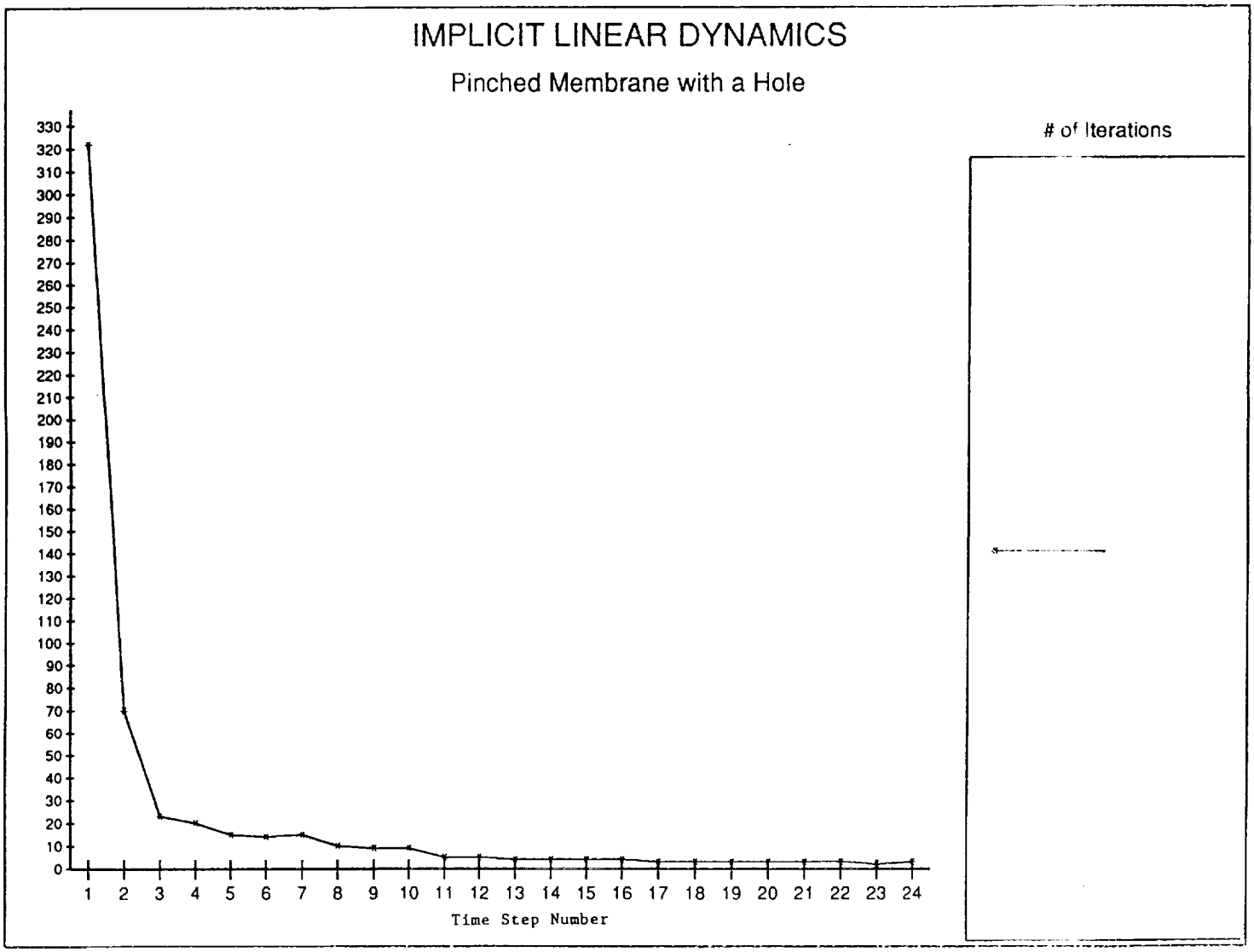

Fig. 6. Convergence rate history

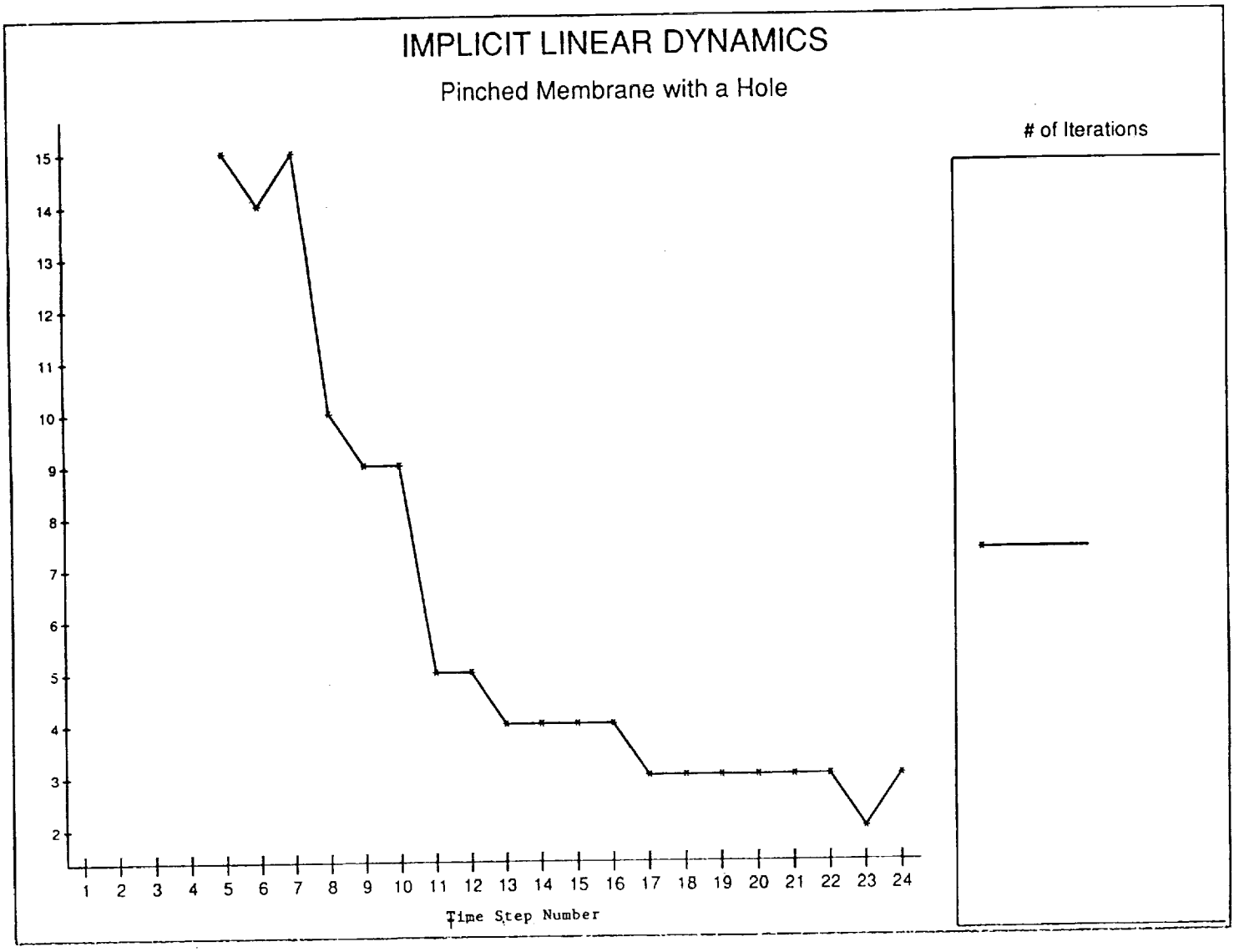

Fig. 7. Zoom on convergence rate history

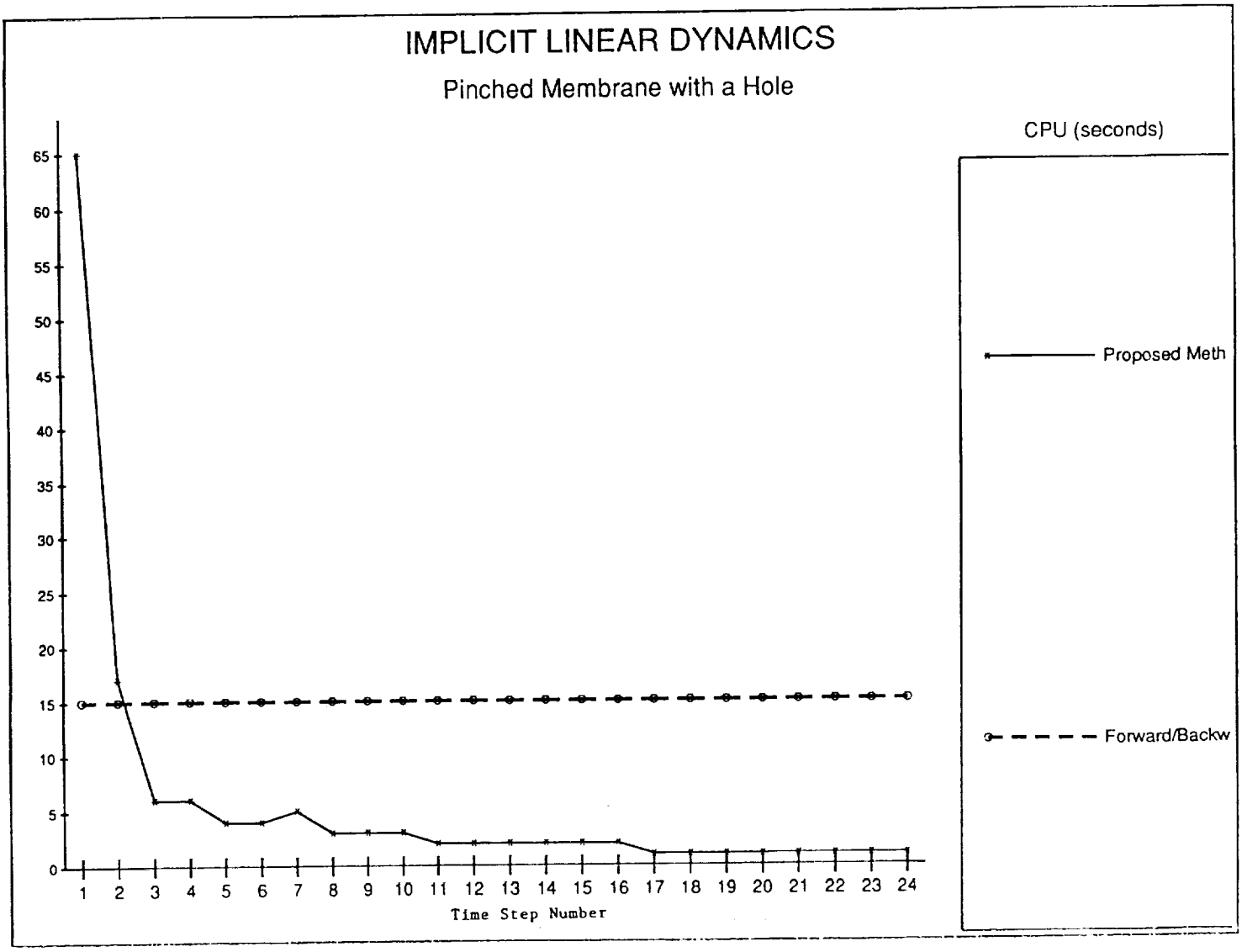

Fig. 8. CPU history

6. Conclusion

In this paper, we have presented a methodology for extending the range of applications of domain decomposition based iterative methods to problems with multiple or repeated right hand sides. Such problems arise, for example, in multiple load static analyses, in implicit linear dynamics, in eigenvalue problems, and in many other structural computations. We have formulated the global problem as a series of minimization problems over K-orthogonal and supplementary subspaces, and have tailored the preconditioned conjugate gradient algorithm to solve them efficiently. The resulting solution method is scalable in the fine granularity regime targeted by emerging parallel processors, whereas direct factorization schemes and forward and backward substitution algorithms are not. We have illustrated the proposed methodology with the solution of realistic static and dynamic structural problems, and have highlighted its potential to outperform forward and backward substitutions on parallel computers. The proposed methodology enhances the versatility of domain decomposition based iterative algorithms.

Acknowledgments

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

References

[1] C. Farhat and E. Wilson, A parallel active column equation solver, Comput. & Struct. 28 (1988) 289-304.

[2] E. Rothberg and A. Goopta, An efficient block-or iented approach to parallel sparse Cholesky factorization, (1993, private communication).

[3] H. Simon, P. Vu and C. Yang, Performance of a supernodal general sparse solver on the CRAY Y-MP:1.68 GFLOPS with autotasking, Applied Mathematics Technical Report, Boeing Computer Services, SCA-TR-117, March 1989.

[4] R. Glowinski, G. H. Golub, G. A. Meurant and J. Periaux (eds.), First international symposium on domain decomposition methods for partial differential equations, SIAM, Philadelphia (1988).

[5] P. E. Bjordstad and O. B. Widlund, Iterative methods for solving elliptic problems on regions partitioned into substructures, SIAM J. of Num. Anal. 23 (1986) 1097-1120.

[6] C. Farhat and F. X. Roux, A method of finite element tearing and interconnecting and its parallel solution algorithm, Internat. J. Numer. Meths. Engrg. 32 (1991) 1205-1227.

[7] C. Farhat and M. Lesoinne, Automatic partitioning of unstructured meshes for the parallel solution of problems in computational mechanics, Internat. J. Numer. Meths. Engrg. 36 (1993) 745-764.

[8] 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, ed., Proc. Fifth SIAM Conference on Domain Decomposition Methods for Partial Differential Equations, SIAM (1991) 271-292.

[9] C. Farhat and F.X. Roux, An unconventional domain decomposition method for an efficient parallel solution of large-scale finite element systems, SIAM J. Sc. Stat. Comp. 13 (1992) 379-396.

[10] C. Farhat and M. Geradin, Using a reduced number of Lagrange multipliers for assembling parallel incomplete field finite element approximations, Comput. Meths. Appl. Mech. Engrg. 97 (1992) 333-354.

[11] F. X. Roux, Acceleration of the outer conjugate gradient iteration by reorthogonalization for a domain decomposition method with Lagrange multipliers, in: T.F. Chan, R. Glowinski, J. Periaux, O. B. Widlund, ed., Proc. Third SIAM Conference on Domain Decomposition Methods for Partial Differential Equations, SIAM (1990) 314-321.

[12] C. Farhat, L. Crivelli and F. X. Roux, A transient FETI methodology for large-scale parallel implicit computations in structural mechanics, Internat. J. Numer. Meths. Engrg. (to appear).

[13] D. D. Davis, T. Krishnamurthy, W. J. Stroud and S. L. McCleary, An accurate nonlinear finite element analysis and test correlation of a stiffened composite wing panel, American Helicopter Society National Technical Specialists' Meeting on Rotorcraft Structures, Williamsburg, Virginia, Oct. 29-31 (1991).

[14] C. Farhat, J. Mandel and F. X. Roux, Optimal convergence properties of the FETI domain decomposition method, (submitted for publication).