Nonlinear Substructuring Method for Concurrent Processing Computers

Olaf O. Storaasli*
NASA Langley Research Center, Hampton, Virginia
and
Pål Bergan†
Det Norske Veritas, Høvik, Norway

This paper proposes a method, based on substructuring, to solve nonlinear structural analysis problems on multiprocessor computers. Background information is given on the use of substructuring in large-scale finite-element programs, and computational time distributions for the major components of an example nonlinear finite-element analysis are discussed. The implementation of the substructuring method on a typical multiprocessor computer is described, and estimates are made of expected reductions in computation times based on nonlinear substructuring results obtained on a single-processor computer.

Nomenclature

C_{ij}	= damping matrix
e_0	= iteration convergence criterion
$f(i)$	= vector of forces or loads
$F_b(j)$	= force vector for boundary nodes
j	= index indicating substructure number
$[K]$	= global stiffness matrix
$K_{bb}(j)$	= matrix of boundary node terms for j
$K_{ib}(j)$	= matrix of interior boundary terms for j
$K_{ii}(j)$	= matrix of interior node terms for j
K_t	= tangent stiffness matrix
L_j	= lower triangular matrix
M_{ij}	= mass matrix
N	= number of concurrent substructures
Q_j	= resulting term from decomposition
q_j	= decomposed force vector
R	= speedup for concurrent equation solution
U_j	= upper triangular matrix
W_b	= work for parallel band solver
W_s	= work for parallel substructuring solver
$X(i)$	= vector of displacements
λ	= load parameter

I. Introduction

THE development and use of substructuring and superelement techniques has played an important role in large-scale linear analysis, primarily because it extends the size of finite-element models that can be solved. Substructuring has been incorporated in many well-known general purpose finite-element codes used for the analysis of aerospace, marine, and off-shore structures, some with finite-element models ranging up to 750,000 equivalent degrees of freedom. Recently, nonlinear analysis has been required for some of these structures that exhibit complex behavior or must meet stringent design requirements. This nonlinear analysis requires considerable computation time to generate stiffness matrices and solve the load displacement equations for many iterations within each load step.

This paper proposes a method to reduce nonlinear analysis computation time by using substructuring on concurrent processing computers. The development of such methods is important, as concurrent processing computers offer the possibility for significant gains in computational speed for typical structural analysis programs.[1,2] However, unlike powerful single-processor engineering workstations,[3] the power of vector and concurrent processing computers cannot be fully exploited without making significant changes in existing finite-element codes. Experience in parallel processing on NASA Langley's first multiple instruction multiple data (MIMD) computer (the Finite Element Machine of Refs. 4-9) has shown that the greatest computational gains are obtained by writing special-purpose codes based on "rethinking" the solution method; somewhat smaller gains have resulted from "recoding" an existing algorithm, and no gain has resulted from the "dusty deck" approach of just running an existing program on a parallel computer. This situation exists because the compilers and operating systems of most parallel computers currently lack any automated capability to exploit parallelism, although such features may become available in the coming years.

The objective of this paper is to describe substructuring and concurrent processing methods currently used and to propose a method, based on substructuring, to solve nonlinear analysis problems on multiprocessor computers, and to assess its computational benefits.

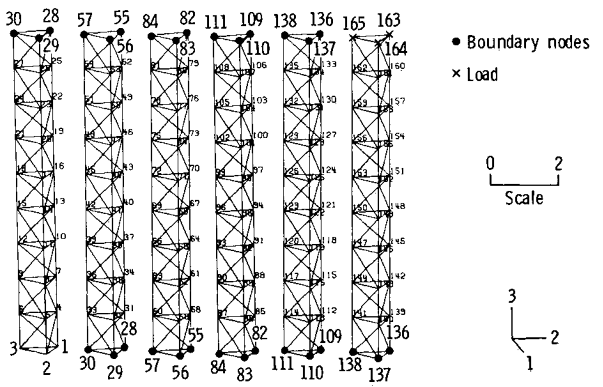


Fig. 1 Single-level substructuring of space frame.

Received July 22, 1986; revision received Oct. 21, 1986.

*Research Scientist, Structural Mechanics Branch. Associate Fellow AIAA.

†Director of Research.

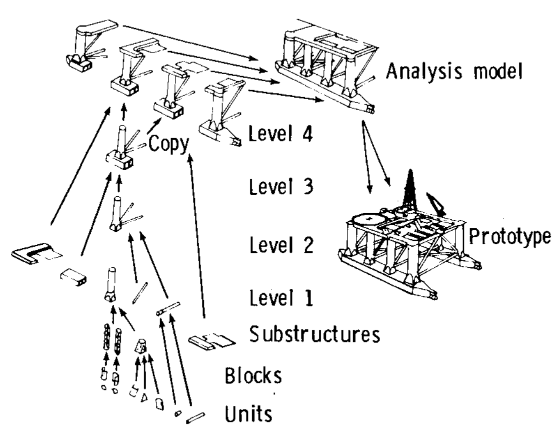


Fig. 2 Multilevel substructuring of oil platform.

II. Solution Methods

Substructuring Methodology

The principal motivation for the use of substructuring in linear analysis is to extend the size of finite-element models that can be analyzed. Examples of single-level and multilevel substructuring are shown in Figs. 1 and 2 respectively.

Figure 1 shows, for single-level substructuring, how a space truss model is divided into six similar substructures, each with only three structural nodes (termed boundary nodes) connecting it to the adjacent substructures on each end. Figure 2 shows, for multilevel substructuring, how a large repetitive substructure (or superelement) is assembled from successive substructuring levels down to the basic finite elements. Multilevel substructuring permits the stiffness matrix for large repetitive structural sections to be built up by levels, assembled only once, and used for each section. A static condensation to eliminate the internal nodes for each substructure is performed to form a condensed stiffness matrix and modified load vector.

Substructuring can reduce the computation time by replacing time-consuming operations on large matrices with faster operations on smaller matrices, eliminating matrix assembly time for repetitive assemblies, and simplifying the data organization. The global stiffness matrix representing the six substructures in Fig. 1 may be written as

$$[K] = \begin{bmatrix} K_{ii}(1) & 0 & 0 & 0 & 0 & 0 & K_{ib}(1) \\ 0 & K_{ii}(2) & 0 & 0 & 0 & 0 & K_{ib}(2) \\ 0 & 0 & K_{ii}(3) & 0 & 0 & 0 & K_{ib}(3) \\ 0 & 0 & 0 & K_{ii}(4) & 0 & 0 & K_{ib}(4) \\ 0 & 0 & 0 & 0 & K_{ii}(5) & 0 & K_{ib}(5) \\ 0 & 0 & 0 & 0 & 0 & K_{ii}(6) & K_{ib}(6) \\ K_{bi}(1) & \cdot & \cdot & \cdot & \cdot & K_{bi}(6) & K_{bb} \end{bmatrix} \tag{1}$$

where $K_{ii}(j)$ represents the contribution from the jth substructure, K_{bb} the stiffness due to the boundary nodes, and K_{ib} the connections between the interior and boundary nodes. Most linear finite-element programs, even those with substructuring capability, use bandwidth-oriented matrix equation solution techniques to solve the load displacement equation

$$[K]X = f \tag{2}$$

where $[K]$ is the global stiffness matrix in Eq. (1), X the displacement vector, and f the load vector. When substructuring is present, block elimination procedures based on the natural decoupling of the substructures [i.e., $K_{ii}(j)$] may also be used. An example of one such block solution procedure[10] is given here and discussed later relative to concurrent processing. In a block solution procedure, the stiffness blocks $K_{ii}(j)$, which represent the stiffness of each substructure due to the internal nodes, are each factored into lower and upper triangular matrices

$$K_{ii}(j) = L_j U_j \tag{3}$$

Using the same $L_j U_j$ factors, this leads to

$$K_{ib}(j) = L_j U_j Q_j \tag{4}$$

where Q_j is the resulting term, and

$$f_i(j) = L_j U_j q_j \tag{5}$$

where q_j is the resulting load vector. The boundary contributions are then formed by summing the contributions from each substructure j and factoring

$$\bar{K}_{bb}(j) = K_{bb}(j) - K_{bi}(j) Q_j \tag{6}$$

$$F_b(j) = f_b(j) - K_{bi}(j) q_j \tag{7}$$

$$\bar{K}_{bb} = L_b U_b \tag{8}$$

yielding first the displacements X_b at the boundary nodes from

$$L_b U_b X_b = F_b \tag{9}$$

and then the displacements $X_i(j)$ at the interior nodes for substructures $j = 1$ to n from

$$L_j U_j X_i(j) = -K_{ib}(j) X_b(j) + f_i(j) \tag{10}$$

A tradeoff exists between the number of substructures used and the amount of additional computation introduced as more substructures are added. Also, the matrix $\bar{K}_{bb}(j)$ in Eq. (6) tends to fill with nonzero terms and theoretically grows very large for structures with a complex interdependency between nodes such as an n-dimensional cube.[11]

An alternative approach, incorporated in some finite-element analysis codes, is to use substructuring for the elemental and global matrix assembly followed by sparse matrix solution procedures. Research on substructuring methodology for both linear and nonlinear analysis focusing on single-processor computers (e.g., Refs. 12 and 13) is being conducted. This paper extends this work to suggest a nonlinear analysis substructuring method applicable to multiprocessor computers.

Nonlinear Analysis

Linear finite-element programs cannot accurately predict the response of structures with large displacements and rotations or where the load-displacement behavior is nonlinear (i.e., systems with stiffening, softening, limit loads, and bifurcation points). Typical of the equations to be satisfied for nonlinear elasticity (as in the FENRIS formulation, Ref. 14) are the system equilibrium equations

$$f(\text{internal}) = f(\lambda) - M\ddot{X} - C\dot{X} \tag{11}$$

where X is the nodal displacement vector, f (internal) are the internal reaction forces for the current state of internal stress, $f(\lambda)$ are the external applied loads, M is the mass matrix, and C is the damping matrix. The solution of the nonlinear equations assumes a linearized incremental form of the equilibrium equations based on two configurations close to each other. These equations are

$$K_t \Delta X = \Delta f(\lambda) - M \Delta \ddot{X} - C \Delta \dot{X} \tag{12}$$

where K_t is the tangent stiffness matrix calculated for the current stress-strain state and deformation of the structure. For static analysis problems, the mass matrix and damping terms are absent and the equations are solved for the displacement ΔX, using the skyline profile block solver[15] or other popular linear equation solvers. The geometry is then updated by adding the displacements to the previous nodal coordinates:

$$X_{i+1} = X_i + \Delta X \tag{13}$$

This process is repeated for this load step until the updated displacement differs from the previous displacement by no more than e_0 times the norm of the updated displacement:

$$\|X_{i+1} - X_i\| < \|X_{i+1}\| e_0 \tag{14}$$

When this convergence criterion is satisfied, the next load step is calculated.

Most nonlinear programs are modifications to linear programs and retain the same substructuring technique used in their linear version. However, some nonlinear programs such as FENRIS have been completely rewritten and streamlined for the efficient solution of nonlinear structural analysis by including a different substructuring method than for linear analysis. In FENRIS, for example, the stiffness contributions of each substructure are added directly to form the system global stiffness matrix, so that Eq. (2) can be solved for all degrees of freedom simultaneously. This approach complements recent concurrent processing research which has concentrated on the development of general-purpose direct and iterative methods for solving large sparse systems of equations in parallel.[16-22]

Computation Distribution for Nonlinear Analysis

Insight into the distribution of computations for an example nonlinear analysis problem is given in Figs. 3-5.

A cylinder with end diaphragms, corresponding properties, and applied load is shown in Fig. 3. A nonlinear analysis was performed using FENRIS for this model for two $m \times n$ mesh configurations: 16×8 and 8×4. Nonlinear analysis computations were performed to obtain the load displacement behavior for the 16×8 mesh with 16 elements longitudinally ($m = 16$) and 8 circumferentially ($n = 8$). The model contained 128 quadrilateral shell elements, 153 nodes, and 818 degrees of freedom. The model was subjected to 7 load steps each with up to 5 iterations, and a new tangent stiffness matrix was calculated with every other iteration and at every load step. A convergence criterion of $e_0 = 0.0001$ was used for the iterative procedure. For comparison purposes, this pinched cylinder problem was solved for a 8×4 mesh with 8 elements longitudinally ($m = 8$) and 4 circumferentially ($n = 4$). This reduced model contained 32 elements, 45 node points, and 215 degrees of freedom and was subjected to 8 load steps. A convergence study for this problem[23] provides more details of the solution and comparisons with other published results.

The distribution of computation and I/O time needed to perform a nonlinear analysis of the 818-degree-of-freedom pinched cylinder problem on an Apollo DN330 computer is shown in Fig. 4. A similar distribution of computation time for the 215-degree-of-freedom reduced model is shown in Fig. 5. The majority of computation time for both models (Figs. 4 and 5) is associated with the generation of the nonlinear stiffness matrix.

Since the percentage of computation associated with stiffness matrix generation is greater for the smaller model (78% in Fig. 5) than for the larger model (68% in Fig. 4), one would expect even larger models to have even smaller percentages of total time taken for stiffness matrix generation. In contrast, the time taken for equation solution grows from 17% for the smaller problem (Fig. 5) to 28% for the larger problem (Fig. 4), so one would expect the equation solution to percentally become the dominant computation factor as the problem size continues to increase.

Since the larger example problem took considerable time to solve (1304 CPU s and 657 I/O s on a VAX 11/785), one can expect both the matrix generation and the equation solution to take significant computation time for problems taking approximately an hour to solve. The proposed substructuring method for the solution of nonlinear analysis problems on concurrent processing computers addresses both the stiffness matrix generation and the equation solution.

Concurrent Processing Methods

Concurrent processing implies executing two or more processes simultaneously on separate computers. Such computers set new demands on data structure, data management, organization, program coding, and adaptability considerations. For existing programs, the suitability for concurrent processing must be carefully evaluated before making any

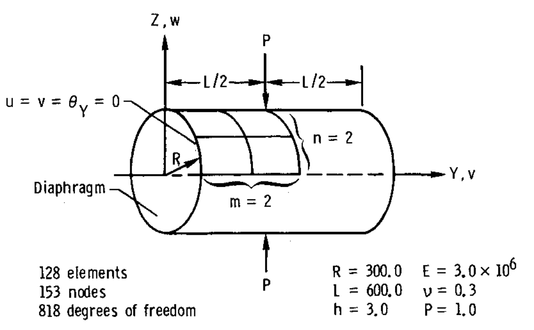


Fig. 3 Nonlinear bending of a pinched cylinder.

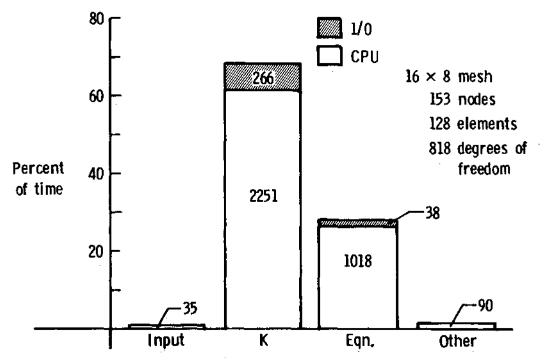


Fig. 4 Results for 818-degree-of-freedom cylinder.

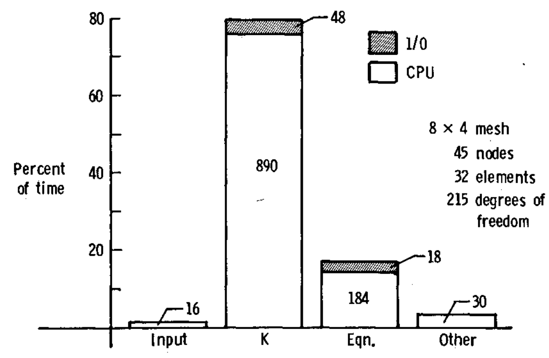


Fig. 5 Results for 215-degree-of-freedom cylinder.

significant modifications. In many cases, it may not be worth the investment to enhance a code for concurrent processing. Significant gains should occur for general purpose codes that are used extensively for the solution of large-scale nonlinear problems. However, the speedups to be realized depend to a great extent on the architecture and properties of the program.

Even though significant software support is still in its infancy for most multiprocessor computers, research results from the Finite Element Machine have shown significant computational speedups (proportional to the number of processors) for the geometrically nonlinear dynamic analysis[5,6] and the materially nonlinear stability analysis of structural sections.[7,8]

A meaningful comparison between a substructuring technique [Eqs. (1-10)] and a banded technique for the concurrent solution of Eq. (2) is difficult, since it is strongly influenced by the complexity and interdependence of the nodal points, constraints, and boundary conditions. However, a theoretical comparison was made for an n-dimensional cube example.[11] The results showed that the computational work required to solve Eq. (2) by a substructuring method was

$$W_s = 108C_a n^4 d + 54C_c n^2 d \tag{15}$$

compared to

$$W_b = 36C_a n^5 + 18C_c n^3 \tag{16}$$

for a band solver, where C_a is the cost of one floating point multiply/add and C_c is the cost of communicating $B+1$ floating point numbers where B is the bandwidth. Reference 11 concludes that a substructuring technique requires less computation and communication time than a band solver where the number of node points n in each direction of the substructure satisfies

$$n^{3/8} < N < n/3 \tag{17}$$

where N is the number of substructures running concurrently. Although these results were for a particular linear structures problem with a complex nodal interconnection, similar reductions in computation time can be expected for nonlinear problems where the solution of Eq. (2) is required for multiple iterations at each load step.

III. Proposed Method

The proposed method employs substructuring to reduce the computation time associated with stiffness matrix generation as well as concurrent matrix solution strategies based on the characteristics of the finite-element model being solved. The organization of the nonlinear solution procedure is shown in Fig. 6, where *SSN* denotes substructure *N*.

The general approach in this method is to process concurrently the geometry, nodes, elements, and boundary conditions for each substructure and to specify the common boundary nodes. Although the time reduction gained by performing this input processing may not appear significant, such substructuring is already built into most general purpose nonlinear analysis codes. The advantages occur in eliminating the need to transfer data among processors to form the individual substructure matrices in parallel. The global stiffness matrix representing these substructures may be expressed in terms of the load-deflection equation as

$$\begin{bmatrix} K_{ii}(1) & & K_{ib}(1) \\ & \ddots & \vdots \\ & K_{ii}(n) & K_{ib}(n) \\ K_{bi}(1) \ldots K_{bi}(n) & & K_{bb} \end{bmatrix} \begin{bmatrix} X_i(1) \\ \vdots \\ X_i(n) \\ X_b \end{bmatrix} = \begin{bmatrix} f_i(1) \\ \vdots \\ f_i(n) \\ f_b \end{bmatrix} \tag{18}$$

The following steps, illustrated in Fig. 6, outline a proposed concurrent substructuring method utilizing both local and shared memory:

1) Install at a minimum that portion of a nonlinear finite-element program that defines and develops the stiffness matrix for each substructure on N concurrent processors.

2) Subdivide the finite-element structural model to be analyzed into N computationally equivalent substructures (i.e., a similar number of reduced degrees of freedom).

3) For all substructures, define the load step increments, convergence criteria, and equation solution option (i.e., band, block, or iteration solver) based on the degree of interconnection complexity of the substructures.

4) Process input for each substructure defined (i.e., nodes, elements, and boundary nodes) on each processor in parallel.

5) From the nonlinear stiffness matrix (including geometric and material nonlinearity terms) in parallel for each substructure as shown in Eq. (18).

6) Eliminate boundary nodes and form the global stiffness matrix in common memory based on the individual substructure matrices computed in step 5.

7) Form the tangent stiffness matrix K_t by subtracting components of the previous stiffness matrix from the updated stiffness matrix. During the first iteration only the linear $[K]$ matrix is available.

8) Solve for displacements according to Eq. (12) by using the concurrent solver (i.e., band, block, or iterative) selected in step 3.

9) Update the geometry (based on the displacements just calculated) and return to step 5 unless convergence is achieved or the iteration limit is reached for this load step.

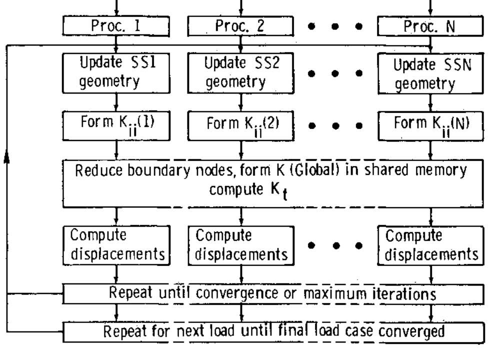
10) Repeat steps 5-9 for each load step until the maximum displacement or final load step is reached.

Step 1 assumes that sufficient local memory exists on each processor to store the executable nonlinear finite-element program, to process the substructure input to the program, and to form the stiffness matrices for each substructure. The feasibility of step 1 is partially demonstrated by the installation of the comprehensive FENRIS software system (200,000 lines of FORTRAN 77) on an engineering workstation[3] with similar memory and computational characteristics to one processor of a concurrent processing computer.

To maximize the efficiency gains from step 2, the structure should be sufficiently large to warrant division into substruc-

Table 1 Solution times vs number of substructures

Substructures	1 (16×8)		4 (16×2)	
Time (s)	CPU	I/O	CPU	I/O
Input	11.990	18.060	15.380	16.253
Form $[K]$	691.240	453.433	670.710	16.880
Eq. solution	591.180	156.327	584.860	0.187
Other	9.950	29.046	9.070	1.767



Analysis code + substructure N (SSN) input to processor N
Proc. 1
Proc. 2
Proc. N
Update SS1 geometry
Update SS2 geometry
Update SSN geometry
Form $K_{ii}(1)$
Form $K_{ii}(2)$
Form $K_{ii}(N)$
Reduce boundary nodes, form K (Global) in shared memory compute K_t
Compute displacements
Compute displacements
Compute displacements
Repeat until convergence or maximum iterations
Repeat for next load until final load case converged

Fig. 6 Concurrent substructuring analysis method.

tures, each comprising a large number of internal nodes with a minimal number of boundary nodes. This requires insight and judgment on the part of the analyst. Dividing a large structure into computationally balanced substructures may someday be automated. However, at present this process must take into account the reduced computational load for substructures where constraints are present and the increased computational load in substructures with complex elements. For load balancing considerations, if more than N substructures are required, then the number of computationally balanced substructures should be a multiple of the number of processors available. Then, once a processor has completed its work on one substructure, it can proceed to work on the next one until work on all substructures is completed.

One can expect significant computation time reductions (in proportion to the number of concurrent substructures) by performing steps 2–4 in parallel. This reduction is important since the stiffness matrix generation can consume most of the computation time, as in the example nonlinear finite-element analysis problem shown in Figs. 3–5.

The nonlinear stiffness matrix in step 5 [shown in Eq. (18)], although similar in form to the linear matrix of Eq. (1), is much more complex, since each entry reflects contributions due to geometric and material nonlinearities. Thus, as the load incrementation progresses, the nonlinear geometry and material behavior terms are calculated and included in the matrix in addition to the linear stiffness contributions. The tangent stiffness matrix K_t, which is the difference between subsequent stiffness matrices, is calculated for subsequent load steps. This process [shown in Eq. (12)] is repeated for all load steps and involves significantly more computations than linear analysis.

Multiprocessing in step 8 may be exploited in conjunction with different methods for solving linear equations. The stiffness matrix may be divided into hypermatrices, which may be operated on concurrently. Blocking may be accomplished as in Eqs. (1–10) or in groups of columns. The so-called "element-by-element" methods of Ref. 24 may also be used for substructures. The factorization of hypermatrices along the main diagonal combined with iterative corrections from outside terms is also feasible. Also, the conjugate gradient method with preconditioning is a reliable iterative method applicable for most structural applications. Details of these and other methods are contained in Refs. 16–22.

The method just described is sufficiently general to be implemented in most existing nonlinear finite-element analysis systems such as FENRIS with minimum (but not negligible) software modification. Such adaptability is an important consideration since many years have been invested to develop such comprehensive analysis systems, and significant modification could require more effort than developing a new code suitable for concurrent processing computers.

Since many nonlinear finite-element programs also perform an eigenvalue analysis of nonlinear structures, the method proposed may be extended to the solution for eigenvalues and eigenvectors. This requires the generation of the mass matrix in step 4 and the nonlinear solution for eigenvalues in step 8 [according to Eq. (12)] by a concurrent technique such as that proposed in Ref. 9.

IV. Concurrent Substructuring Simulation Results

To determine the potential computation speedup gained by the proposed method on a concurrent processing computer, solutions to the pinched cylinder problem (Fig. 3) were obtained on a VAX 11/785 computer for both one and four substructures and the resulting computation times compared. The 16×8 mesh cylinder was divided into four 16×2 regions with 17 boundary nodes along each edge. The computation times attributed to both CPU and I/O for the one- and four-substructure pinched cylinder nonlinear analysis are given in Table 1.

The time shown in Table 1 for the stiffness matrix generation for four substructures represents the time taken for each of the four substructure analyses run sequentially. Each substructure took approximately 168 s for stiffness generation, which is less than one quarter of the stiffness generation time for one substructure (691.24 in Table 1). This reduction represents a potential speedup of 4.043 for the stiffness generation if the four processors are computing concurrently. Since the stiffness generation on each processor is decoupled except for minimal communications associated with boundary node calculations, one can expect the computation speedup to be proportional to the number of processors on concurrent processing computers. The results in Table 1 may be conservative, as a division of the cylinder into four 8×4 regions (with even fewer boundary nodes) would be likely to give even lower computation time than for the 16×2 mesh. Greater speedups can be expected for problems where the number of internal nodes is large with respect to boundary nodes.

Table 1 also shows that the disk I/O time is reduced significantly by dividing the problem into substructures. This advantage is important for single-processor computers where the disk I/O is often a major computation expense for large problems. It is likely that such reductions in I/O will correspond to similar reductions in interprocessor communication time for concurrent processing computers. Assuming that one can expect a computational speedup for the equation solution of R, the total computational speedup for N concurrent substructures is

$$\text{Speedup } (N) = \frac{T_{\text{INPUT}} + T_{\text{OTHER}} + T_K + T_{\text{EQ}}}{T_{\text{INPUT}} + T_{\text{OTHER}} + T_K/N + T_{\text{EQ}}/R} \tag{19}$$

where T_K, T_{EQ}, T_{INPUT}, and T_{OTHER} represent the time taken on one processor for stiffness generation, equation solution, input, and other calculations as shown in Table 1.

The time taken for input and other calculations is quite small even for the pinched cylinder example, so as the problem size increases, one can expect the computation speedup for N processors to approach

$$\text{Speedup } (N) = \frac{T_K + T_{\text{EQ}}}{T_K/N + T_{\text{EQ}}/R} \tag{20}$$

V. Concluding Remarks

This paper proposes a method based on substructuring for the solution of nonlinear finite-element analysis problems on concurrent processing computers. It describes a method by which the input processing and generation of stiffness matrices is performed in parallel with the parallel equation solution technique selected by the user based on the problem characteristics.

Distributions of computation time for the input processing, stiffness generation, equation solution, and other calculations are given for two discretizations of a nonlinear analysis problem. Since the distributions show the stiffness generation to contain the most computations, one can expect the initial gains to be the greatest by performing the nonlinear stiffness generation in parallel. The trend in the distributions also shows that the computations associated with equation solution increase rapidly with problem size and will contain the major computations once the stiffness generation is performed in parallel.

Simulation results are presented which show the stiffness generation to speed up by a factor of N, the number of substructures running concurrently. An additional speedup factor results from the use of general purpose concurrent matrix solution routines. The combination of these two speedups available in the proposed method can significantly reduce the total solution time for nonlinear structural analysis on parallel computers. The method described is sufficiently general to implement in existing finite-element analysis codes with substructuring capability.

Acknowledgments

Much of this research was done from July 1984 to July 1985 while the first author was on a NASA research fellowship at the Norwegian Institute of Technology. The close cooperation of personnel from the Norwegian Institute of Technology, Det Norske Veritas, and the Norwegian Government Foundation of Scientific and Industrial Research (SINTEF) was extremely valuable and greatly appreciated by the authors.

References

[1]Noor, A. K., Storaasli, O. O., and Fulton, R. E., "Impact of New Computing Systems on Finite Element Computations," *State-of-the-art Surveys on Finite Element Technology*, edited by Noor and Pilkey, ASME Special Publication H00290, Nov. 1983, pp. 499–530. (Also in *Impact of New Computing Systems on Computational Mechanics*, edited by Noor and Pilkey, ASME Special Publication H00275, Nov. 1983, pp. 1–32.)

[2]Noor, A. K., Storaasli, O. O., and Fulton, R. E., "Impact of New Computing Systems on Computational Mechanics and Flight-Vehicle Structures Technology," The Influence of Large Scale Computing on Aircraft Structural Design, AGARD Rept. 706, Sienna, Italy, April 1984, pp. 2-1–2-29.

[3]Storaasli, O. O. and Bergan, P. G., "Nonlinear Structural Analysis on an Engineering Workstation," *Proceedings of the 9th ASCE Conference on Electronic Computation*, Feb. 1986, Birmingham, AL, pp. 394–405.

[4]Storaasli, O. O., Peebles, S. W., Crockett, T. W., Knott, J. D., and Adams, L., "The Finite Element Machine: An Experiment in Parallel Processing," 1982, NASA TM 84514, pp. 201–217.

[5]Storaasli, O. O., Ransom, J., and Fulton, R. E., "Structural Dynamic Analysis on a Parallel Computer: The Finite Element Machine," *Proceedings of the AIAA/ASME/ASCE/AHS 25th Structures, Structural Dynamics and Materials Conference*, Palm Springs, CA, May 1984, pp. 537–543.

[6]Ransom, J. P., Storaasli, O., and Fulton, R., "Application of Concurrent Processing to Structural Dynamic Response Computations," *Symposium on Advances and Trends in Structures and Dynamics*, Washington, DC, Oct. 1984, NASA CP 2335, pp. 31–44.

[7]Darbhamulla, S. P., Razzaq, Z., and Storaasli, O. O., "Concurrent Processing for Nonlinear Analysis of Hollow Rectangular Structural Sections," *Proceedings of the AIAA/ASME/ASCE/AHS 26th Structures, Structural Dynamics and Materials Conference*, Orlando, FL, April 1985, pp. 463–470.

[8]Darbhamulla, S. P., Razzaq, Z., and Storaasli, O. O., "Concurrent Processing in Nonlinear Structural Stability," *Proceedings of the AIAA/ASME/AHS/ASCE 27th Structures, Structural Dynamics and Materials Conference*, San Antonio, TX, May 1986, pp. 545–550.

[9]Bostic, S. W. and Fulton, R. E., "A Concurrent Processing Implementation for Structural Vibration Analysis," AIAA Paper 85-0783-CP, *AIAA/ASME/ASCE/AHS 26th Structures, Structural Dynamics and Materials Conference*, Orlando, FL, April 1, 1985, pp. 566–572.

[10]Noor, A., Kamel, H., and Fulton, R., "Substructuring Techniques—Status and Projections," *Computers and Structures*, Vol. 8, 1978, pp. 621–632.

[11]Adams, L. and Voigt, R. G., "A Methodology for Exploiting Parallelism in the Finite Element Process," NASA CR 172219, Sept. 1983.

[12]Ryu, Y. S. and Arora, J. S., "Review of Nonlinear FE Methods with Substructures," *Journal of Engineering Mechanics*, ASCE, Vol. 111, Nov. 1985, pp. 1361–1379.

[13]Dodds, R. H. and Lopez, L. A., "Substructuring in Linear and Nonlinear Analysis," *International Journal for Numerical Methods in Engineering*, Vol. 15, 1980, pp. 583–597.

[14]Bergan, P. G. and Arnesen, A., "FENRIS—A General Purpose Nonlinear Finite Element Program," *Proceedings of the 4th International Conference on Finite Element Systems*, Southampton, July 1983.

[15]Wilson, E. L. and Dovey, H. H., "Solution or Reduction of Equilibrium Equations for Large Complex Structural Systems," *Advances in Engineering Software*, Vol. 1, No. 1, Dec. 1978.

[16]Ortega, J. M. and Voigt, R. G., "Solution of Partial Differential Equations on Vector and Parallel Computers," *SIAM Review*, Vol. 27, No. 2, June 1985, pp. 149–240.

[17]Reed, D. and Patrick, M., "Parallel Iterative Solution of Sparse Linear Systems: Models and Architectures," ICASE Rept. 84-35, NASA Langley Research Center, Hampton, VA.

[18]Reed, D. and Patrick, M., "A Model of Asynchronous Iterative Algorithms for Solving Large Sparse Linear Systems," *Proceedings of the 1984 International Conference on Parallel Processing*, pp. 402–409.

[19]Adams, L., "Iterative Algorithms for large Sparse Linear Systems on Parallel Computers," Ph.D. Thesis, Univ. of Virginia, Charlottesville, VA. (Also NASA CR-166027, NASA Langley Research Center, Hampton, VA.)

[20]Adams, L., "An M-Step Preconditioned Conjugate Gradient Method for Parallel Computation," *Proceedings of the 1983 International Conference on Parallel Processing*, pp. 36–43.

[21]Adams, L. and Ortega, J., "A Multi-Color SOR Method for Parallel Computation," *Proceedings of the International Conference on Parallel Processing*, pp. 53–56.

[22]Kowalik, J., Lord, R., and Kumar, S., "Design and Performance of Algorithms for MIMD Parallel Computers," *Proceedings of the NATO Workshop on High Speed Computations*, edited by Kowalik, NATO ASI Series, Vol. F-7, Springer-Verlag, Berlin, Germany.

[23]Bergan, P. G. and Nygard, M. K., "Nonlinear Shell Analysis Using Free Formulation Finite Elements," *Proceedings of the Europe-US Symposium on Finite Element Methods for Nonlinear Problems*, Vol. 2, The Norwegian Institute of Technology, Trondheim, Norway, Aug. 12–15, 1985.

[24]Hughes, T. J. R., Winget, J., Levit, I., and Tezduyar, T. E., "New Alternating Direction Procedures in Finite Element Analysis Based on EBE Approximate Factorization," *Computer Methods for Nonlinear Solids and Structural Mechanics*, edited by S. Atluri and N. Perrone, AMD, Vol. 54, 1983, pp. 75–109.