COMMUNICATIONS IN APPLIED NUMERICAL METHODS, Vol. 4, 425–434 (1988)

LINEAR AND NONLINEAR FINITE ELEMENT ANALYSIS ON MULTIPROCESSOR COMPUTER SYSTEMS

EDWARD L. WILSON

Department of Civil Engineering, University of California at Berkeley, CA 94720, U.S.A.

AND

CHARBEL H. FARHAT

University of Colorado, Boulder, CO 90429, U.S.A.

SUMMARY

Several general purpose computer systems with multiple processors operating concurrently are currently being commercially produced. Most of the present generation of finite element software was not designed to take advantage of this new technology. The purpose of this paper is to present the advantages of a new architecture for finite element programs which will operate efficiently on computer systems with any number of processors. Also, the basic approach is effective for multiprocessor computers with local or shared memory.

The new computer program architecture is based on an initial application of a simple algorithm which automatically subdivides the complete finite element domain into a number of subdomains equal to the number of available processors. The resulting data structure requires a minimum of communication between processors during the formation of basic element matrices, reduction of subdomain matrices and the postprocessing of element results. The assembly and solution of the global system of subdomains can be accomplished directly or iteratively, and new concurrent solution algorithms can be introduced.

Several of the ideas presented here have been tested on various types of computer. The new program architecture indicates that it is possible to obtain speed-up times of over 90 per cent of the maximum theoretical values if appropriate numerical methods are employed.

INTRODUCTION

The use of computer systems with multiple processors for the solution of finite element domains was first suggested in 1976.[1] However, it is only within the past few years that commercial multiprocessor computers have become available. The present authors have worked on the automatic subdomain algorithm,[2] concurrent algorithms for dynamic analysis,[3] concurrent iterative solution methods[4] and direct concurrent solution algorithms.[5] The purpose of this paper is to present the basic method in a more general context and to emphasize that the automatic subdomain approach has significant advantages when extended to nonlinear problems in all areas of computational mechanics.

Evolution of computer hardware

The majority of existing finite element analysis programs are based on sequential processing and computer hardware which existed over 20 years ago. At that time the central processing unit (CPU) and random access memory (RAM) were much more expensive than the other components of the computer system, and computer operating systems were therefore designed to achieve maximum utilization of the CPU and RAM. This was accomplished by allowing several programs to be executed at the same time, with memory being shared. Since the amount of RAM on these computers was limited, a large amount of data transfer was required between RAM and secondary

0748–8025/88/030425–10$05.00

Received April 1987

disc storage. As the data transfer in and out of RAM was relatively slow, the CPU could serve several programs without being overloaded.

Within the past 10 years, with the introduction of superminicomputers, this multi-user type of computer architecture has been carried to the extreme. Most of these multi-user systems automatically move data in and out of RAM (this is called paging). It is very common for the performance of these systems to degrade seriously if two or more finite element analyses are being conducted at the same time. In fact, the 'wall clock time' for many finite element analyses on modern multi-user systems is greater than if the same finite element model is run on an inexpensive, single-user microcomputer.

The CPU and RAM for microcomputers have now become very inexpensive compared with other components. For example, the CPU, 640 kbytes of RAM and a floating-point processor will cost about $300 for an IBM PC class of microcomputer. In addition, the basic floating-point arithmetic speed of a single-processor microcomputer is between a tenth and a third of the speed of the CPU on an expensive multi-user computer system, so the idea of sharing one CPU between several users is now an obsolete concept.

Evolution of finite element software

The development of finite element software during the past 25 years has been based on the assumption that only CPU is available, so programs were developed to operate sequentially. Inherent in the design of most existing finite element programs are data structures and numerical algorithms which are based on sequential processing. It is possible, within each execution phase of existing programs, to modify the algorithms to take advantage of multiprocessing. However, if the concurrent computational ability of these new computers is to be fully utilized, a completely new approach must be introduced in the basic organizational structure of finite element programs. In the selection of a new program structure we must bear in mind that the data structure and the numerical techniques used must be general with respect to the type of elements and mesh. Also, they must be efficient for both dynamic and nonlinear problems. If these objectives are achieved, the same basic approach may be taken for nonstructural types of problems in computational mechanics.

The general method presented in this paper is significantly different from the sequential approach used in existing programs. It consists in subdividing the finite element domain into the same number of subdomains (substructures) as there are available processors, N_p. (Or, if a limited number of processors are available the total number of subdomains must be an approximate multiple of N_p). The subdomains should be selected such that the computational effort is approximately equal for each subdomain. If the finite element model contains only one type of element, a subdivision which contains an approximately equal number of elements in each subdomain may be a logical basis for an automatic subdivision algorithm.

Some finite element program developers believe that the FORTRAN compilers for multiprocessor computers will be able to subdivide the computational effort between the different processors without having to change the basic architecture of the program. This approach does not recognize that new numerical algorithms are required which can be executed concurrently. Others have worked on improving algorithms for specific solution phases for the traditional sequential finite element program.[6] The subdomain approach requires a completely new program architecture if the potential of this new computer hardware is to be fully exploited.

Justification of the subdomain approach

For static linear problems with few load conditions, the total solution time on a sequential computer is the sum of the time required to process element information and the time required to solve the global equilibrium equations. The computational time required to form element stiffnesses, assemble the global stiffness and calculate element stresses is directly proportional to the number of elements. The computational time needed to solve the global equilibrium equations is proportional to the number of equations times the 'average' band-width squared.

For most two-dimensional problems the execution time, on a sequential computer, for the solution of equations is less than 50 per cent of the total computational effort. Therefore, if we use only concurrent processing for the solution of equations we will not reduce the overall computation time by more than 50 per cent. For this class of problem it is apparent that we must perform concurrent computing at the element level if a significant reduction in computer time is to be achieved.

For a two-dimensional $N \times N$ mesh the total number of elements is N^2, and total number of equations is $(N+1)^2$, with a band-width proportional to $N+1$. Therefore, the ratio of the solution time to the time required to form element stiffnesses is proportional to N^2. For three-dimensional problems with $N \times N \times N$ elements, the number of equations to be solved is $(N+1)^3$, with a band-width of $(N+1)^2$. Or, the ratio of the solution time to the time required to form element stiffnesses is proportional to N^4. Hence, large problems in three dimensions are dominated by the solution of the equation phase if direct solution methods are employed. Because of this fundamental difference between two- and three-dimensional problems, many researchers in numerical analysis are re-evaluating iterative methods for the solution phase of three-dimensional problems since iteration within each subdomain can be clearly executed concurrently.[4]

In linear dynamic response analysis, in which element stresses are evaluated as a function of time, the evaluation of element stresses may take up most of the computational effort. The use of concurrent processing would produce an increase in speed directly proportional to the number of processors. For nonlinear static or dynamic analysis, where the element stiffnesses and stresses must be computed at each load or timestep, the use of concurrent computing at the element level will be very effective.

In addition, the new program structure can be used to solve other types of field problem in mechanics, such as fluid flow or heat transfer. Also, the automatic subdomain approach may be used in the classical numerical method of finite differences to improve performance on multiprocessor computer systems.

It should be emphasized that the use of the subdomain architecture does not require existing programs for finite element analysis to be completely rewritten. The existing program modules for pre- and postprocessing, the formation of element matrices and the calculation of stresses can be directly incorporated into the new subdomain architecture. Our experience has indicated that the required program development for the automatic creation of subdomains, the reduction of subdomains and the solution of the global domain requires fewer than 1500 FORTRAN statements.

BASIC PROGRAM ARCHITECTURE

It is very important that a new finite element program has an architecture which will operate effectively on computers with any number of processors. Present supercomputers such as the CRAY have one to four processors; whereas, the hypercube design may have several hundred. In addition, some of the hardware is designed on the assumption that each processor has its own memory, and other computers share the same memory with all the processors. The subdomain approach has the advantage of being equally effective on all existing computers which have multiprocessors.

The subdomain approach requires the finite element model to be subdivided into the same number of subdomains as the number of processors, N_p, which are available on the computer system. In order to obtain maximum efficiency it is essential that all processors are assigned equal computational effort. Also, for local memory processors it is very important that there is a minimum level of data communication between processors.

Figure 1 illustrates the concurrent solution of a finite element model on a multiprocessor computer system. After the model is subdivided into N_p domains, all element and subdomain calculations are completed without the need for interprocessor communication. In a computer system with local memory, only the node coordinates, loads and element properties associated with each subdomain need to be stored, so these basic data do not need to be duplicated in the memory of the other processors. Also, the basic data are uncoupled during the stress recovery phase.

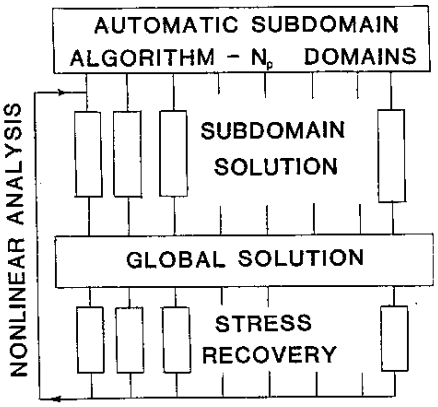


Figure 1.

AUTOMATIC CREATION OF SUBDOMAINS

The geometry of a finite element model can be numerically defined by a list of element numbers and the node numbers associated with each element. These data can be stored in two integer arrays. The MP array is the number of elements in length in which the integer MP(i) is the location in the MN array of the last node number associated with the ith element. Therefore, the node numbers associated with the ith element are stored at locations MN[MP($i-1$)+1] to MN[MP(i)], where MP(0) is defined as equal to zero. This basic data structure allows for a mixture of one-, two- and three-dimensional elements, each with a different number of nodes.

In order for the automatic subdomain algorithm to operate at increased efficiency it is necessary to create additional arrays. The NP array is the number of nodes in length in which NP(j) is the location in the NN array of the last element number associated with the jth node. Therefore, the numbers for the elements attached to the jth joint are stored in locations NN[NP($j-1$)+1] to NN[(NP(j)], where NP(0) is defined as zero. The data structures for these arrays are illustrated in Figure 2. The data in these basic arrays are not changed during the execution of the algorithm.

During the execution of the automatic subdomain algorithm, three additional integer arrays are created and modified. The NW array is equal in length to the number of nodes and contains the number of elements attached to each node. As elements are removed from the system the numbers in the NW array are reduced. The array NF contains the number of active node numbers. (An active node is one in which some of the elements have been removed.) When all elements have been removed from a node, the node number is removed from the NF array. The ME array is equal in length to the number of elements, and contains the element numbers in the sequence in which they are removed from the system.

A summary of the algorithm for the automatic creation of subdomains is given in Table I. It should be noted that all arrays can be retained in high-speed memory during the execution of the algorithm. Also, the algorithm is very fast since a minimum of array searching is required during execution.

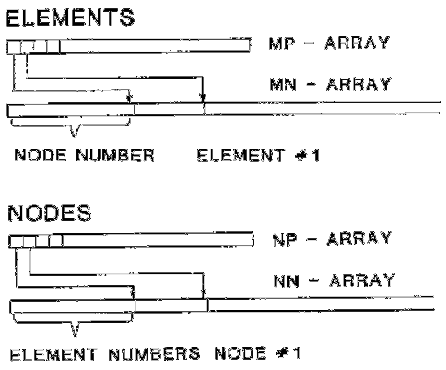


Figure 2.

Table I. Summary of the automatic subdomain algorithm

A.	NUMERICAL DEFINITION OF FINITE ELEMENT MODEL Element–node connectivity MN array containing node numbers and the MP element pointer array
B.	INITIAL CALCULATION OF NODE DATA Node-element connectivity NN array containing element numbers and the NP node pointer array
C.	DEFINITION OF WORKING ARRAYS NW array containing the number of elements connected to each node NF array containing the active node number ME array containing element numbers in order produced
D.	ALGORITHM TO PRODUCE SUBDOMAINS OF L ELEMENTS EACH 1. Zero NF array and start at node which has a minimum number of elements. 2. Remove all elements attached to node and update ME, NF and NW arrays. 3. After L elements are removed, return to Step 1. 4. Eliminate nodes with zero elements and compact NF array. 5. Return to Step 2 and use node NF(1) as next node.

DIRECT SOLUTION ALGORITHM

In this paper a concurrent direct solution method of finite element systems is summarized. One of the main reasons for restricting our discussion to noniterative methods is that they are easily extended to dynamic response analysis, using either mode superposition or direct step-by-step integration. Further research is required in iterative methods in concurrent computation before they become as general and as robust as the direct solvers.

The set of equilibrium equations for a substructure or a complete structure may be written in the form

$$\mathbf{AX} = \mathbf{B} \tag{1}$$

where $\mathbf{A}$ is the NEQ×NEQ symmetrical matrix, $\mathbf{X}$ is the matrix of the node unknowns and $\mathbf{B}$ is the matrix of node loads. It is important to note that it is not necessary to know boundary loads in order to condense the internal degrees of freedom for a subdomain. Before presenting the basic subdomain reduction algorithm, we consider the basic solution algorithm for a complete system. The first is the factorization of $\mathbf{A}$:

$$\mathbf{A} = \mathbf{LU} \quad \text{or} \quad \mathbf{A} = \mathbf{LDL}^{\mathrm{T}} \tag{2}$$

in which the jth column of $\mathbf{U}$ is evaluated from:

$$\mathbf{U}_{ij} = \mathbf{A}_{ij} - \sum_{k=1}^{i-1} \mathbf{L}_{ik}\mathbf{U}_{kj}, \quad j = 2, NEQ, \quad i = 1, j-1 \tag{3}$$

and the jth row of $\mathbf{L}$ is given by

$$\mathbf{L}_{ji} = \mathbf{U}_{ij}/\mathbf{D}_{ii}, \quad j = 2, NEQ, \quad i = 1, j \tag{4}$$

The diagonal terms $\mathbf{U}_{ii}$ and $\mathbf{D}_{ii}$ are identical since $\mathbf{L}_{ii}$ is normalized to unity. Within a computer program the terms $\mathbf{A}_{ij}$, $\mathbf{U}_{ij}$ and $\mathbf{L}_{ji}$ can be stored in the same location; it is not necessary to store $\mathbf{L}_{ii}$.

Equation (1) can be written as

$$\mathbf{LDY} = \mathbf{B} \tag{5}$$

where

$$\mathbf{Y} = \mathbf{L}^{\mathrm{T}}\mathbf{X} \tag{6}$$

Therefore, the forward reduction algorithm can be written as

$$\mathbf{Y}_i = \left(\mathbf{B}_i - \sum_{k=1}^{i-1} \mathbf{L}_{ik}\mathbf{Y}_k\right) \Big/ \mathbf{D}_{ii} \tag{7}$$

and the equation for back-substitution is

$$\mathbf{X}_i = \mathbf{Y}_i - \sum_{k=i+1}^{\mathrm{NEQ}} \mathbf{L}_{ki}\,\mathbf{X}_k, \quad i = \mathrm{NEQ},\ 1 \tag{8}$$

Also, within a computer program it is possible to store the terms $\mathbf{X}_i$, $\mathbf{Y}_i$ and $\mathbf{B}_i$ at the same location.

As described in detail in Reference 7, these equations can be modified for profile storage and substructure reduction. Equations (3), (4) and (7) can be written in the form

$$\mathbf{U}_{ij} = \mathbf{A}_{ij} - \sum_{k=kz}^{kh} \mathbf{L}_{ik}\,\mathbf{U}_{kj}, \quad j = 2,\ \mathrm{NEQ}, \quad i = 1,\ \mathrm{NEQ} \tag{9}$$

$$\mathbf{L}_{ji} = \mathbf{U}_{ij}/\mathbf{D}_{ii}, \quad j = 2,\ \mathrm{NEQ}, \quad i = 1,\ \mathrm{LEQ} \tag{10}$$

$$\mathbf{Y}_i = \left(\mathbf{B}_i - \sum_{k=kz}^{kh} \mathbf{L}_{ik}\,\mathbf{Y}_k\right)\Big/\mathbf{D}_{ii}, \quad i = 1,\ \mathrm{LEQ} \tag{11}$$

$$\mathbf{B}_i^* = -\sum_{k=kz}^{kh} \mathbf{L}_{ik}\,\mathbf{Y}_k, \quad i = \mathrm{LEQ},\ \mathrm{NEQ} \tag{12}$$

where kz is the first nonzero term in column j, and kh is the minimum of $i-1$ and LEQ. The vector $\mathbf{B}_i^*$ represents the loads which are transferred to the subdomain boundary due to the elimination of the interior unknowns of the subdomain. The reduced stiffness is automatically developed in the last NEQ−LEQ locations. Also, partial back-substitution is possible after the boundary unknowns $\mathbf{X}_{\mathrm{LEQ}+1}$ to $\mathbf{X}_{\mathrm{NEQ}}$ have been evaluated by the global solution of the complete system. Equation (8) can be written as

$$\mathbf{X}_i = \mathbf{Y}_i - \sum_{k=i+1}^{\mathrm{NEQ}} \mathbf{L}_{ki}\,\mathbf{X}_k, \quad i = \mathrm{LEQ},\ 1 \tag{13}$$

Reference 7 contains a FORTRAN listing of the direct solution algorithm which has the subdomain reduction option. It is very important to note that all three phases (factorization, forward-reduction and back-substitution) require vector operations, and that the inner do loops have been replaced with subroutine calls. If each processor has a vector processor, the subdomain operations can be made very efficient.

SUBDOMAIN REDUCTION

The automatic subdomain algorithm presented is capable of subdividing any two- or three-dimensional finite element model into the same number of domains as there are available processors. For $N_p=4$, the two-dimensional, 64-element mesh shown in Figure 3(a) would be subdivided into the four subdomains. This example also illustrates that the well-known nested dissection algorithm is a special case of the automatic subdomain algorithm.

The global equilibrium equations for the 64 element mesh are shown in Figure 3(c). As with traditional substructure analysis, it is possible to express the basic unknowns within the subdomain in terms of the unknowns on the boundary of the domain and thus to form a reduced stiffness with respect to boundary unknowns. The number of numerical operations and the required storage can be minimized within the subdomain by use of the profile-front method.[8] The boundary unknowns are the last equations to be numbered, as shown in Figure 3(b). Therefore, after the reduction of LEQ equations the reduced stiffness matrix, $\mathbf{K}_g^{(i)*}$, and the boundary loads, $\mathbf{F}_g^{(i)*}$, are produced concurrently within each domain. It is important to note that the profile method of data storage allows the triangularization phase of solution to utilize vector processors during the reduction of the subdomain.

GLOBAL SOLUTION OF SYSTEM OF SUBDOMAINS

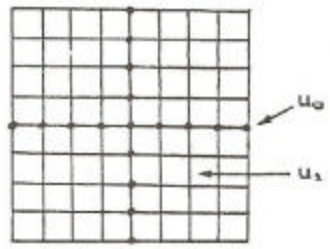
After all subdomains are reduced the global equations can be assembled by the application of the direct stiffness approach:

$$\mathbf{F}_b^* = \Sigma\, \mathbf{F}_g^{(i)} \tag{15}$$

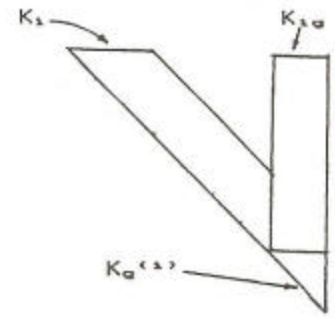
Hence, the global equilibrium equations have the form

$$\mathbf{K}_g^* \mathbf{U}_g = \mathbf{F}_g^* \tag{16}$$

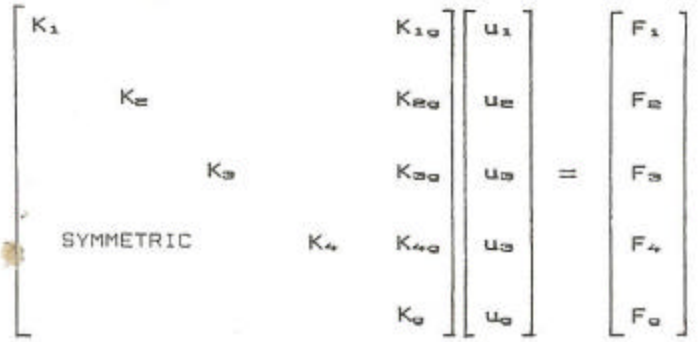
In multiprocessors with local storage the basic matrices can be assembled directly if the columns of the $\mathbf{K}_g^{(i)*}$ matrix are distributed to the different processors before the summation and the addition of the subdomain arrays are conducted concurrently.



(a) FINITE ELEMENT MODEL



(b) TYPICAL SUBDOMAIN DATA STORAGE



(c) EQUILIBRIUM EQUATIONS - TOTAL FINITE ELEMENT SYSTEM

Figure 3.

The global system of equations with respect to the subdomain boundary unknowns can be solved concurrently with a minor modification of the profile equation solver previously presented. The basic topology of the global stiffness matrix is shown in Figure 4. For this example every fourth column is assigned to be reduced by one processor. An examination of equation (9) indicates that after column n is reduced all $\mathbf{U}_{ij}$ columns greater than n can be reduced down to the row n concurrently in any order. Therefore, for large band-width problems it is possible to utilize all the processors very effectively.

With shared memory no communication is necessary, but in multiprocessors with local memory it is necessary to send the column to all other processors after it is completely reduced. In addition, it is necessary to retain a copy of $\mathbf{D}_{ii}$ in the local memory of all processors. The existence of a nonzero term on the diagonal is all the information that is required to let each processor know which terms $\mathbf{U}_{ij}$ can be calculated.

The direct application of equations (7) and (8) will produce the global displacements $\mathbf{U}_g$. It is then possible to distribute the subdomain boundary displacements to each processor for the execution of equation (13) in order to evaluate the displacements within each subdomain.

It is clear that the automatic subdomain and solution algorithms presented here will operate on either shared memory or local memory multiprocessors. Since the local memory processors require

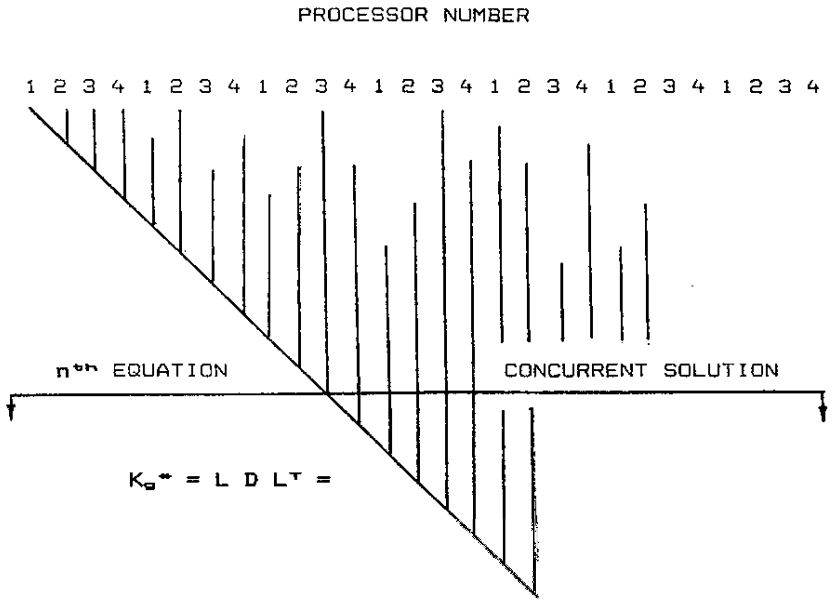


Figure 4.

additional time for passing data, it appears that the shared memory approach is the most general. However, the automatic subdomain approach, as presented here, requires all terms in the stiffness and load matrices to be stored in real memory. For very large problems this may not be possible because of address limitations of the computer. If shared storage is a limitation factor, then the local storage approach within each processor may be advantageous. For computers which automatically page storage, the use of multiprocessors appears to be totally ineffective. If the basic data cannot be retained in real memory, the algorithm can be modified so as to create a multilevel subdomain method in which all data are moved in large blocks between real and low-speed mass storage. The total number of subdomains would then be a multiple of N_p.

LINEAR DYNAMIC ANALYSIS

The automatic subdomain and solution algorithm, which has been presented in detail for static analysis, is easily extended to dynamic analysis of large finite element systems without a significant

Table II. Summary of multiprocessor dynamic analysis by direct superpositions of Ritz vectors

A.	DYNAMIC EQUILIBRIUM EQUATIONS	$\mathbf{Ma} + \mathbf{Cv} + \mathbf{Ku} = \mathbf{f}(s)\mathbf{g}(t)$
B.	INITIAL CALCULATIONS	
	1. Triangularize stiffness matrix	$\mathbf{K} = \mathbf{L}^T\mathbf{DL}$
	2. Solve for static response	$\mathbf{K}\,\mathbf{u}_0 = \mathbf{f}$
C.	GENERATE RITZ VECTORS $\mathbf{V}_1, \mathbf{V}_2, \ldots, \mathbf{V}_n, \ldots, \mathbf{V}_m$	
	1. Solve for $\mathbf{X}_i$	$\mathbf{K}\,\mathbf{X}_i = \mathbf{M}\,\mathbf{u}_i$
	2. **Use modified Gram–Schmidt orthogonalization twice** with respect to all previously calculated vectors and normalized results to	$\mathbf{V}_i^T\,\mathbf{M}\,\mathbf{V}_i = \mathbf{I}$ $\mathbf{u}_i = \mathbf{u}_{i-1} - \mathbf{c}_i\,\mathbf{V}_i$
	3. Remove new Ritz vector from static vector	where $\mathbf{c}_i = \mathbf{V}_i^T\,\mathbf{M}\,\mathbf{U}_{i-1}$
	4. Evaluate error, stop or repeat	
D.	MAKE VECTORS STIFFNESS ORTHOGONAL – OPTIONAL	
	1. Solve $N \times N$ eigenvalue problem	$[\mathbf{K}^* - \Omega\mathbf{I}]\,\mathbf{Y} = \mathbf{0}$ where $\mathbf{K}^* = \mathbf{V}^T\,\mathbf{K}\,\mathbf{V}$
	2. Calculate orthogonal Ritz vectors	$\Phi = \mathbf{V}\,\mathbf{Y}$

reduction of efficiency.[3] The basic approach is to use load-independent Ritz vectors to reduce the size of the system.[9] The latest form of this algorithm is summarized in Table II. As in static analysis, nearly all phases of the method can be carried out concurrently on multiple processors. In addition, after the modal response is evaluated the time-dependent displacements and element stresses can be calculated concurrently within each subdomain.

This approach is fundamentally different to the multiprocessor Lanczos eigenvalue method.[10] In the eigenvalue approach the complete stiffness matrix is required to be duplicated within the local memory of each processor. A different shift is used within each processor, and the eigenvalues are calculated in groups near the shifts. It is clear that this method is limited to problems where the complete stiffness matrix for the finite element system can be contained in the local memory of each processor. In addition, it has been shown that the load-dependent vectors can be generated with less numerical effort and are always more accurate than if the exact eigenvectors are used.[6]

NONLINEAR ANALYSIS

A nonlinear anlaysis of a finite element system often requires 10 to 100 times as much computation as a static linear analysis. In the most stable nonlinear solution algorithms, the load is applied incrementally and iteration is carried out within the load or timestep in order to obtain equilibrium.

In some implicit methods a direct solution of equations is not required at each load or time increment. In this approach the number of numerical operations required is directly proportional to the number of elements, so the automatic subdomain approach will tend to divide the total computational effort equally between the multiple processors.

Other methods of nonlinear analysis require the formulation and direct solution of the equilibrium equations for each increment. For this approach the methods presented in this paper for the concurrent formulation and solution of static problems on multiple processors can be used directly, as indicated in Figure 1. With shared storage all the basic data must be retained in real storage for maximum efficiency. For multiprocessor computers with local storage a small amount of information is duplicated within each processor, and the basic automatic subdomain approach requires a minimum level of communication between processors.

NUMERICAL EXAMPLES

A large number of numerical examples were run on a hypercube multiprocessor computer using the basic numerical methods summarized in this paper. (For more details see References 2–5.) The results of a typical plate bending problem are summarized in Figure 5. The speed-up is defined as the ratio of the computer time required to solve the problem using N_p processors to the computer time required using one processor. In general, the results indicate that the approach is more efficient for larger problems. It is apparent that for very small problems the use of a large number of processors may be counter-productive. Recent experience of solving this problem on a shared-memory computer indicates that speed-up ratios of over 90 per cent can be achieved.

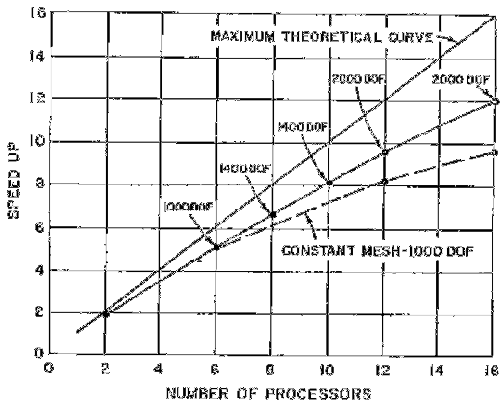


Figure 5.

CONCLUSIONS

The authors have obtained a considerable amount of direct and indirect experience during the past few years on the use of multiprocessor computer systems for the solution of finite element problems using various numerical algorithms. As a result of this work we have come to the following general conclusions:

1. The automatic subdomain algorithm can be used to divide the total computational work equally between all the processors on a multiprocessor computer system. The basic approach tends to minimize communication requirements between processors. Also, the method is effective for static, dynamic and nonlinear problems in all areas of computational mechanics.
2. The development of 'smart' compilers for multiprocessor computers will speed up individual program modules of existing programs in computational mechanics. However, if the maximum use is to be made of the hardware, a new program architecture is required so that all phases of the analysis can be executed concurrently.
3. The Gaussian elimination method, with the basic matrices stored in profile form, is a very effective approach to the concurrent reduction of the basic equations for all subdomains. The global system of equations with respect to the subdomain boundaries can be solved directly if profile storage is used. Each N_pth column is assigned to each processor, and the triangularization is carried out concurrently.
4. Iteration solution methods can also be used effectively with the automatic subdomain approach at both the subdomain and the global levels. This approach appears to have the best potential for the solution of three-dimensional problems. The basic disadvantage of the iterative approach is that it cannot be directly extended to dynamic mode superposition analysis.
5. Multiprocessor computer systems with shared memory appear to offer the most potential for the solution of problems in computational mechanics since the problem of data transfer between the local memory of the different processors is eliminated. The use of multiprocessors on computer systems which automatically page memory should be avoided.

The existence of multiprocessor computer systems is a practical reality for both micro- and supercomputer systems. The incremental cost for additional processors and memory is small, and the potential increase in performance is large. The effective use of this new technology is a challenge to researchers in numerical analysis and computational mechanics.

REFERENCES

1. E. L. Wilson, 'Special numerical and computer techniques for finite element analysis', in *Formulation and Computational Algorithms in Finite Element Analysis*, MIT Press, Cambridge, MA, 1976, pp. 2–25.
2. C. H. Farhat and E. L. Wilson, 'Solution of finite element systems on concurrent processing computers', *Eng. Computers*, **2**, 157–165 (1987).
3. C. H. Farhat and E. L. Wilson, 'Modal superposition dynamic analysis on concurrent multiprocessors', *Eng. Computations*, (1987).
4. C. H. Farhat and E. L. Wilson, 'Concurrent iterative solution of large finite element systems', *Commun. appl. numer. methods*, **3**, 319–326 (1987).
5. C. H. Farhat, 'Multiprocessors in computational mechanics', Ph.D. Dissertation, Department of Civil Engineering, University of California at Berkeley, 1986.
6. R. Fulton, 'The impact of parallel computing on finite element computations', in *Reliability of Methods for Engineering Analysis*, Pineridge Press, Swansea, 1986, pp. 179–196.
7. E. L. Wilson and H. H. Dovey, 'Solution or reduction of equilibrium equations', *Adv. Eng. Software*, **1**(1), (1978).
8. M. I. Hoit and E. L. Wilson, 'An equation numbering algorithm based on minimum front criteria', *Computers Struct.*, **16**(1–4), 225–239 (1983).
9. E. L. Wilson, M. W. Yuan and J. M. Dickens, 'Dynamic analysis by direct superposition of Ritz vectors', *Earthquake eng. struct. dyn.*, **10**, 813–823 (1982).
10. E. P. Bayo and E. L. Wilson, 'Use of Ritz vectors in wave propagation and foundation response', *Earthquake eng. struct. dyn.*, **12**, 499–505 (1984).