NASA Technical Memorandum 102450
ICOMP-89-31
AIAA-89-1395

Eigensolution of Finite Element Problems in a Completely Connected Parallel Architecture

Fred A. Akl
Ohio University
Athens, Ohio

and Institute for Computational Mechanics in Propulsion
Lewis Research Center
Cleveland, Ohio

and

Michael R. Morel
Ohio University
Athens, Ohio

Prepared for the
30th Structures, Structural Dynamics and Materials Conference
cosponsored by the AIAA, ASME, ASCE, AHS, and ACS
Mobile, Alabama, April 3-5, 1989

EIGENSOLUTION OF FINITE ELEMENT PROBLEMS IN A COMPLETELY CONNECTED PARALLEL ARCHITECTURE

Fred A. Akl[*]
Department of Civil Engineering
Ohio University, Athens, Ohio 45701-2979
and Institute for Computational Mechanics in Propulsion
Lewis Research Center, Cleveland, Ohio 44135-3191

and

Michael R. Morel[†]
Department of Civil Engineering
Ohio University
Athens, Ohio 45701-2979

Abstract

This paper presents a parallel algorithm for the solution of the generalized eigenproblem in linear elastic finite element analysis, $[K][\Phi] = [M][\Phi][\Omega]$, where: $[K]$ and $[M]$ are of order N, and $[\Omega]$ is of order q. The parallel algorithm is based on a completely connected parallel architecture in which each processor is allowed to communicate with all other processors. The algorithm has been successfully implemented on a tightly coupled multiple-instruction-multiple-data (MIMD) parallel processing computer, Cray X-MP. A finite element model is divided into m domains each of which is assumed to process n elements. Each domain is then assigned to a processor, or to a logical processor (task) if the number of domains exceeds the number of physical processors. The macrotasking library routines are used in mapping each domain to a user task. Computational speed-up and efficiency are used to determine the effectiveness of the algorithm. The effect of the number of domains, the number of degrees-of-freedom located along the global fronts and the dimension of the subspace on the performance of the algorithm are investigated. For a 64-element rectangular plate, speed-ups of 1.86, 3.13, 3.18 and 3.61 are achieved on two, four, six and eight processors, respectively.

Nomenclature

$[B]_l$	right-hand side at the l^{th} iteration $= [M][V]$
$[B]_{FF}$	assembled global front right-hand sides
$[K]$	stiffness matrix of order $N.N$
$[K]^*_d$	domain subspace stiffness matrix
$[K]^*$	subspace stiffness matrix of order $q.q$
$[M]$	mass matrix of order $N.N$
$[M]^*_d$	domain subspace mass matrix
$[M]^*$	subspace mass matrix of order $q.q$
N	total number of degrees-of-freedom of system
q	dimension of the subspace $\leq N$
$[Q]$	eigenvectors of the auxiliary eigenproblem
$[V]^*_{l+1}$	eigenvectors obtained at the l^{th} iteration
$[V]_{FF}$	unknown variables on global front
β_l	over-relaxation factor
$[\Omega]$	eigenvalues of required subspace of order $q.q$
$[\Phi]$	eigenvectors of required subspace of order $N.q$

[*]Associate Professor
[†]Graduate Student

Introduction

Large finite element models used in the analysis and design of complex structures are not uncommon and usually require enormous amounts of computing time to solve the generalized eigenproblem. As a result, the capabilities of sequential computers are quickly reaching their ultimate peaks and efficient parallel algorithms must be investigated to meet these computing needs.

A major advancement in computer hardware based upon the unique architecture of parallel processing has the potential to decrease execution time by several orders of magnitude. However in order to successfully improve the performance, one must select and develop numerical processes that take advantage of the parallel architecture of this new generation of computers. This paper presents the development, description

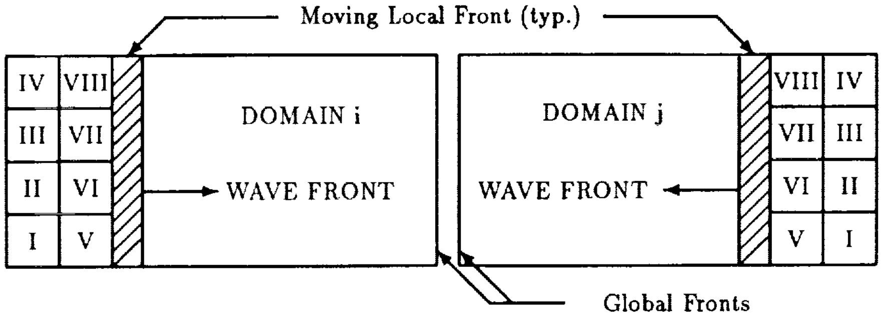


Figure 1: Finite Element Model Divided into m Domains

and results of a new parallel numerical algorithm using a multi-frontal subspace method to determine the natural frequencies and mode shapes of large structures.

Numerical Techniques

The generalized eigenproblem for a structural system is defined as:

$$[K][\Phi] = [M][\Phi][\Omega] \tag{1}$$

in which $[K]$ and $[M]$ are symmetric, $[\Phi]$ is a modal matrix where each column is an eigenvector and $[\Omega]$ is a diagonal matrix containing eigenvalues. For the generalized eigenproblem described in Eq. 1, the eigenvalues are real and positive and the eigenvectors are orthogonal with respect to $[K]$ and $[M]$. The three most time consuming procedures in the solution of large eigenproblems are the creation of element stiffness and mass matrices, the solution of linear simultaneous equations and the extraction of eigenpairs. The efficiency and robustness of the frontal method[1] for the solution of linear simultaneous equations and the modified subspace method[2] for the extraction of the least dominant eigenpairs, have prompted the authors to incorporate them in the concurrent solution of large eigenproblems.

The classical subspace method is reported to provide an efficient algorithm for the solution of large problems in parallel and sequential processing[3,4]. The rate of convergence of the modified subspace method used in this paper is faster by an average of 33% compared to the classical subspace method[2]. There are certain advantages in using the multi-frontal solution method in parallel processing[5]. First, since the bandwidth in the frontal solution depends on the numbering of elements, there is no need to renumber the nodes within each domain to minimize the bandwidth of the submatrices of the domain. In addition, the element numbering scheme for both sequential and parallel solutions may be left unchanged, thereby forgoing preprocessing of the finite element for parallel execution. Second, load balancing is dependent on the frontwidth and the number of elements in each domain. Load balancing is therefore relatively easier to achieve using the multi-frontal solution method.

Parallel Architecture

A finite element model is divided into m domains each of which consists of n elements (Fig. 1). Each domain is then assigned to a physical processor, or to a logical processor (task) if the number of domains exceeds the number of physical processors. The macrotasking library routines[6] are used in mapping each domain to a user task. The parallel algorithm is based on a completely connected parallel architecture (Fig. 2) in which each processor is allowed to communicate with all other processors.

Parallel Algorithm

Fig. 3 shows the logical structure of the parallel algorithm. Each processor creates the stiff-

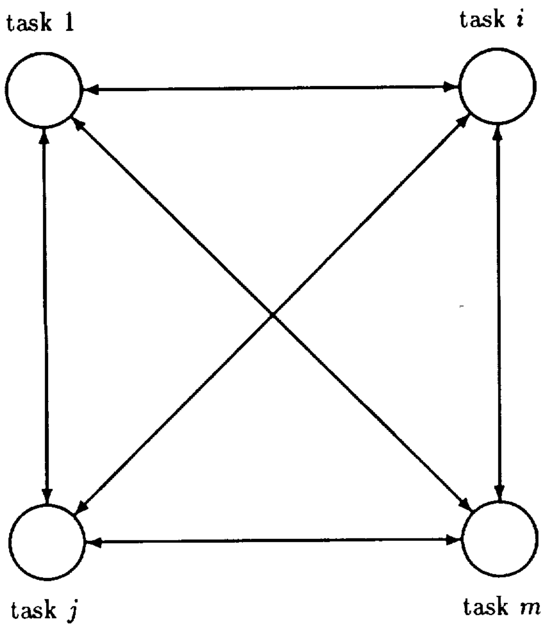


Figure 2: Network of Completely Connected m-Concurrent Processors (Tasks)

ness and mass matrices, $[K]^e$ and $[M]^e$, of the elements located within its assigned domain, and calculate the corresponding right-hand sides, $[B]_l^e$. Random numbers are used to generate $N.q$ starting eigenvectors $[V]_1^e$. Each processor then assembles the element matrices, $[K]^e$ and $[B]_l^e$, and simultaneously eliminates the equations corresponding to the degrees-of-freedom not located along the global fronts (boundaries):

$$[K]^i[V]_{l+1}^{*i} = [B]_l^i \tag{2}$$

where: $[K]^i, [V]_{l+1}^{*i}$ and $[B]_l^i$ are the stiffness matrix, approximate eigenvectors and the corresponding right-hand sides, respectively, of the i^{th} domain just after the assembly of matrices and before the elimination of the degrees-of-freedom during the l^{th} iteration. Although Eq. 2 is never formed in the frontal solution, it is given here to illustrate the algorithm in a more concise manner. At the conclusion of the assembly and elimination steps, two matrix equations are obtained for the i^{th} domain in which the subscript 'F' refers to the degrees-of-freedom located along the global fronts and the subscript 'd' refers to all other degrees-of-freedom within the domain (Fig. 1):

$$[U]_d[V]_d^* + [K]_d[V]_d^* = [B]_d \tag{3}$$
$$[K]_F[V]_F^* = [B]_F \tag{4}$$

A synchronization point is established at this stage in which each processor waits for all other processors to calculate and communicate $[K]_F$ and $[B]_F$, and to assemble the global front matrices $[K]_{FF}$ and $[B]_{FF}$. The solution for the degrees-of-freedom located along the global fronts, $[V]_{FF}$, is obtained and the process of back-substitution within each domain proceeds concurrently until $[V]_{l+1}^{*e}$ is calculated at the l^{th} iteration for each element. Concurrent processing continues to calculate the projection of the stiffness and mass matrices onto the required subspace, $[K]_d^{*i}$ and $[M]_d^{*i}$ of order $q.q$ for the i^{th} domain. This is the second and last synchronization point in the parallel algorithm at which the contribution from all other domains are required before proceeding to solve the auxiliary eigenproblem of the modified subspace[7,8], $[K]^*[Q] = [M]^*[Q][\Omega]$. The selection of the factor β_l is documented[2,8] and will not be repeated here. More accurate approximation of the eigenvectors $[V]_l^e$ is obtained using:

$$[V]_{l+1}^e = [V]_{l+1}^{*e}[Q] \tag{5}$$

The algorithm either terminates or continues to iterate until a test of convergence is satisfied. In this paper a tolerance level of 10^{-7} is imposed on the highest order eigenvalue in the subspace, ω_q^2.

Analysis of Performance

To measure the success of the parallel algorithm on the Cray X-MP/24 supercomputer, two measures are used:

$$\text{Speed-up} = \text{SP} = \frac{T_s}{T_p} \; (\geq 1) \tag{6}$$

$$\text{Efficiency} = \frac{SP}{m} \; (\leq 100\%) \tag{7}$$

where: T_s is the time of sequential algorithm.
T_p is the time of parallel algorithm.
m is the number of processors used in the parallel solution.

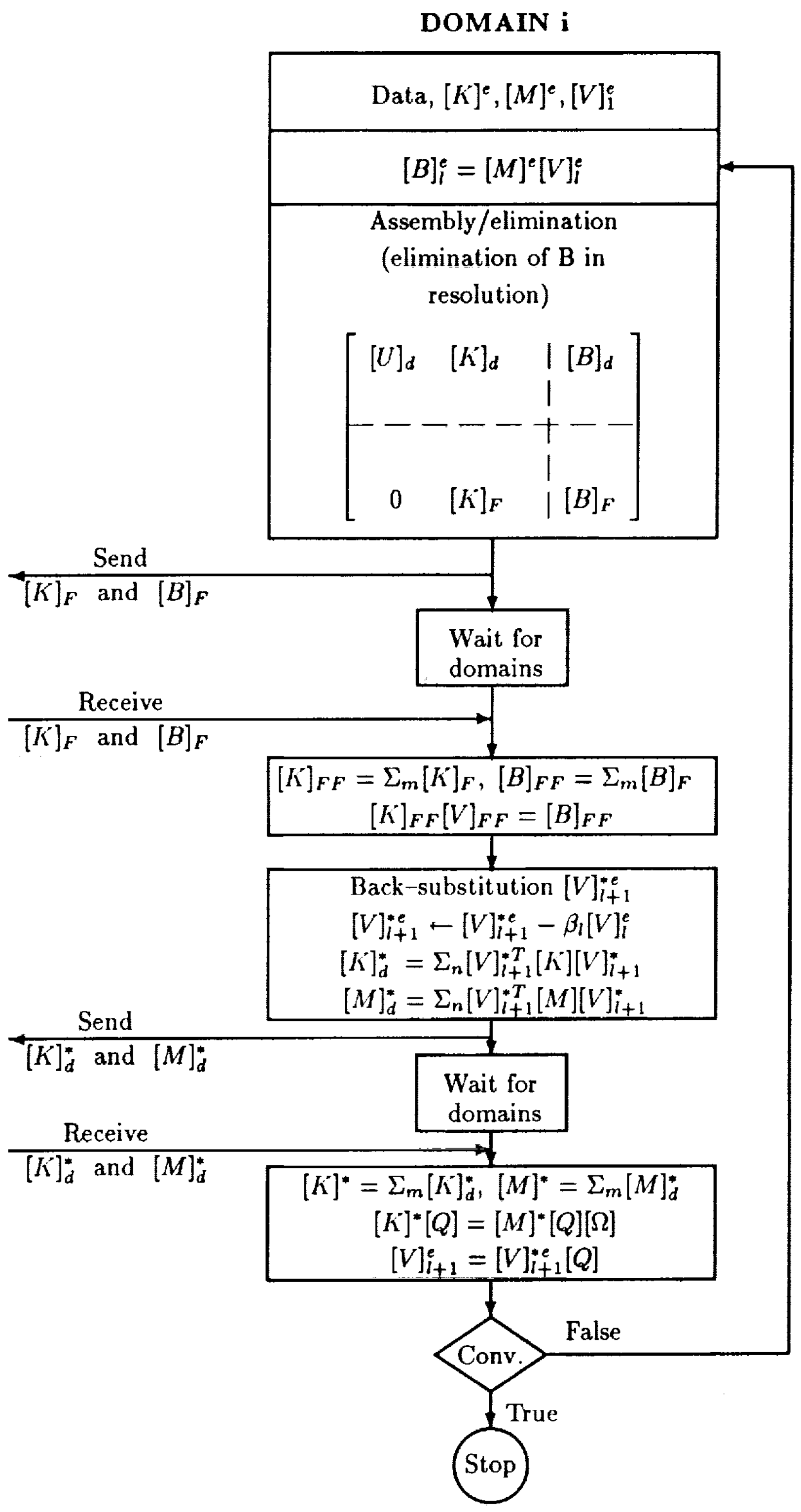

DOMAIN i
Data, $[K]^e, [M]^e, [V]_1^e$
$[B]_l^e = [M]^e[V]_l^e$
Assembly/elimination (elimination of B in resolution)
$\left[\begin{array}{cc|c} [U]_d & [K]_d & [B]_d \\ \hline 0 & [K]_F & [B]_F \end{array}\right]$
Send $[K]_F$ and $[B]_F$
Wait for domains
Receive $[K]_F$ and $[B]_F$
$[K]_{FF} = \Sigma_m[K]_F,\ [B]_{FF} = \Sigma_m[B]_F$
$[K]_{FF}[V]_{FF} = [B]_{FF}$
Back-substitution $[V]_{l+1}^{*e}$
$[V]_{l+1}^{*e} \leftarrow [V]_{l+1}^{*e} - \beta_l[V]_l^e$
$[K]_d^* = \Sigma_n[V]_{l+1}^{*T}[K][V]_{l+1}^*$
$[M]_d^* = \Sigma_n[V]_{l+1}^{*T}[M][V]_{l+1}^*$
Send $[K]_d^*$ and $[M]_d^*$
Wait for domains
Receive $[K]_d^*$ and $[M]_d^*$
$[K]^* = \Sigma_m[K]_d^*,\ [M]^* = \Sigma_m[M]_d^*$
$[K]^*[Q] = [M]^*[Q][\Omega]$
$[V]_{l+1}^e = [V]_{l+1}^{*e}[Q]$
Conv.
False
True
Stop

Figure 3: Parallel Algorithm for the i^{th} Domain

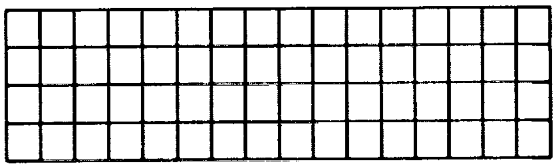


Figure 4: 64–Element Plate, All Edges Clamped.

These two measures are computed for a rectangular plate with 64 identical elements (Fig. 4). The plate is modelled using an isoparametric square plate element of length 2.0 *in*; the element consists of four corner nodes and four mid–side nodes amounting to 16 degrees–of–freedom per element. The plate properties are: Young's modulus is 1.0 *psi*, Poisson's ratio is 0.3, the mass density is 1.0 $lb\ sec^2/in^4$ and 1.0 *in* equaling the thickness. The algorithm developed is tested against a similar sequential algorithm: FEDA[9]. Table 1 shows the first eight eigenvalues for the plate.

Table 1: Predicted Eigenvalues

Order of Eigenvalue	Parallel and Sequential Solution (Tol=10^{-7})
1	$0.1171\text{x}10^{-1}$
2	$0.1306\text{x}10^{-1}$
3	$0.1569\text{x}10^{-1}$
4	$0.2017\text{x}10^{-1}$
5	$0.2731\text{x}10^{-1}$
6	$0.3814\text{x}10^{-1}$
7	$0.5401\text{x}10^{-1}$
8	$0.7662\text{x}10^{-1}$

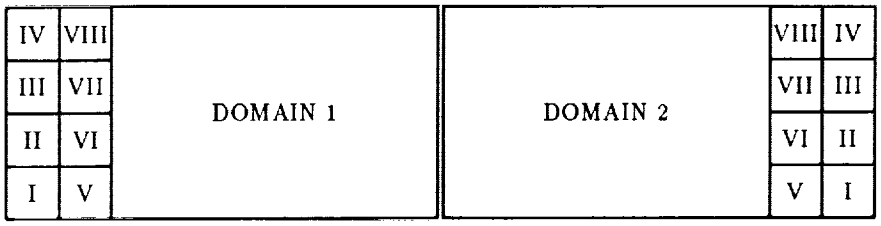
A number of test runs were analyzed to examine the variables influencing the speed–up and efficiency of the algorithm in a dedicated mode. The following sections present the results obtained from a number of example problems for the 64–element rectangular plate with all edges clamped.

Number of Domains

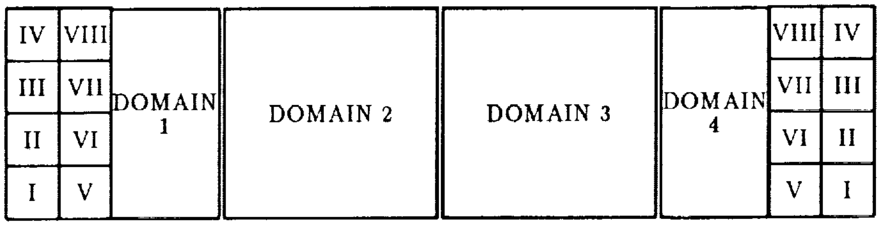
Ideal speed–up for the parallel algorithm should be equivalent to the number of domains the finite element model has been subdivided into. The rectangular plate shown in Fig. 4 is tested to determine the speed–up and efficiency on two, four, six and eight processors. The decoupled plates are shown in Fig. 5 to describe the global fronts and element numbering layout. Favorable results were obtained on the two and four processor models (Table 2). The six and eight processor models showed performance degradation due to the relatively high number of degrees–of–freedom on the global front.

Table 2: Performance of Various Domains with $q = 2$

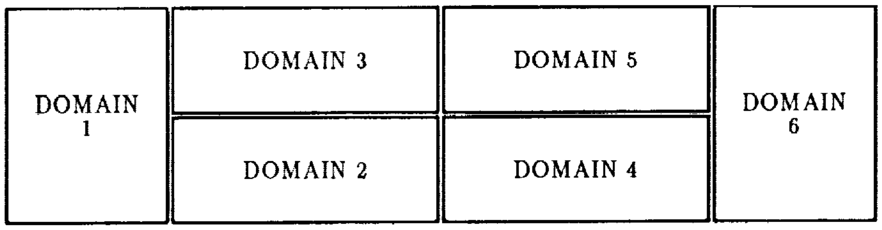
No. of Processors	Figure No.	Number of Iterations	Speed–up (Tol=10^{-7})	Efficiency
1	4	16	1.00	100%
2	5–a	16	1.86	93%
4	5–b	14	3.13	78%
6	5–c	16	3.18	53%
8	5–d	14	3.61	45%



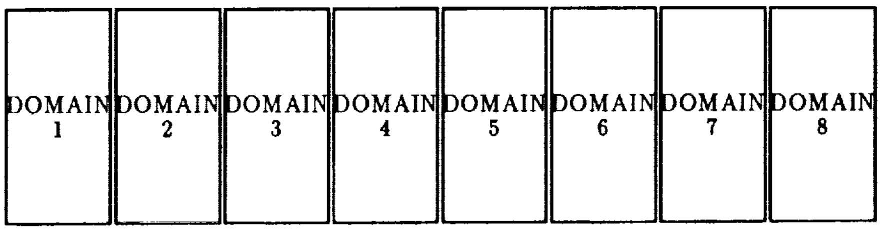
a: Two-Domain Configuration



b: Four-Domain Configuration



c: Six-Domain Configuration



d: Eight-Domain Configuration

Figure 5: 64-Element Plate Configurations

Table 3: Performance of Two Subspace Iterations

No. of	Figure	Speed-up	Efficiency
Processors	No.	$q = 2$	
1	4	1.00	100%
2	5-a	1.68	84%
4	5-b	2.84	71%
6	5-c	2.88	48%
8	5-d	2.83	35%

The two-processor model (Fig. 5a) performed very well with a speed-up of 1.86. When comparing the results of Tables 2 and 3, the two-domain model gains momentum as the number of iterations increases. The global front has only 13 degrees-of-freedom which is the lowest possible number for the two domain model. This is the only system where an accurate evaluation of the communication links could be verified. It was found that transmitting information from one processor to another through common blocks accumulated minimal overhead.

Table 2 also shows that the results for the four-domain model in Fig. 5b benefited greatly from a lower number of iterations in the parallel solution. The number of iterations taken to achieve tolerance plays a big part in determining the overall speed-up of the algorithm. Difference in roundoff error between the sequential and parallel solutions is suspected to be the reason for the different number of iterations.

Subroutine Evaluation

To initially start the multitasking package a main program was developed to set the synchronization points and map out all domain processors. The time taken to perform this task was calculated to be between 20 to 60 milliseconds, which does not have a big impact on the total execution time. At the first of three synchronization points all input data is read into task one and passed to the other tasks to avoid overhead due to single-threaded I/O on the Cray computer. This causes a 1.0 second delay until all processors can move forward again. A description of the subroutines used by parallel FEDA (*p*FEDA) is located in Appendix A.

In DMATRON, the subroutine that determines the first and last appearance of all nodes in its domain has a very low speed-up for all sizes of domains. This subroutine takes about 2% of the total execution time to complete. Some overhead is accumulated in this subroutine but is not critical to the total execution time. The creation of element matrices, $[K]^e$ and $[M]^e$, is the first place where significant speed-up is achieved because the finite element model is substructured into an equal number of elements in each task; the individual tasks should have an ideal speed-up of 2.0, 4.0, 6.0 and 8.0 for two, four, six and eight processors in subroutine ESTIFF. Referring to Table 4, the two, four and eight-domain structures show efficiencies of 89%, 90% and 90% for subroutine ESTIFF which means some overhead has been compiled at this point due to parallel processing. In the unbalanced six-processor model (Fig. 5c) the efficiency is only 80% as a result of the extra elements in domains one and six.

Table 4: Subroutine Speed-ups for $q = 6$

	Number of Processors			
Subroutine	Two	Four	Six	Eight
DMATRON	1.27	1.96	2.13	2.40
ESTIFF	1.78	3.59	4.78	7.17
First Iteration				
DFRONT	1.60	1.96	1.21	0.91
DCONDS	1.86	3.63	4.75	6.96
RELOAD	1.92	3.81	5.08	7.65
Second Iteration				
DFRONT	1.83	2.55	2.37	2.34
DCONDS	1.86	3.64	4.81	6.97
RELOAD	1.92	3.84	5.06	7.64

After the element matrices have been generated, the program is ready to begin the solution-resolution process to determine the natural frequencies of the system. The most critical subroutine is DFRONT where the multi-frontal technique is implemented along with the assembly and elimination of the global front. In addition, the second synchronization point is located within this subroutine to send/receive data on the interface matrices (Fig. 3).

Number of Degrees-of-Freedom along Global Fronts:

Success of the parallel algorithm is dependent upon the number of degrees-of-freedom on the global front; as the number of domains increase so does the number of degrees-of-freedom on the global front. Subroutine DFRONT behaves progressively worse as the number of degrees-of-freedom on the global front and domains increase which can be seen in Table 4. For example, in the two-domain problem (Fig. 5) with 13 degrees-of-freedom on the global front and 115 degrees-of-freedom remaining in each domain, subroutine DFRONT in the first and second iteration take up 19% of the total execution time. In contrast, the eight-domain problem with 91 degrees-of-freedom on the global front and 19 degrees-of-freedom remaining in each domain takes 64% of the total execution time. The execution time of DFRONT in the first iteration is always greater than that of the subsequent subspace iterations because of a lower number of calculations that are required in subsequent iterations.

The remaining two subroutines DCONDS and RELOAD perform the calculations of the modified subspace method. Some overhead is associated with these subroutines but overall their speed-ups are consistent and performed very well. Imbedded in DCONDS is the final synchronization point to send/receive all domain subspace matrices.

In summary, the sources of overhead associated with pFEDA are:

1. The extra coding to implement parallel processing.
2. Input of data and the map of the pre-front needed in the solution.
3. Communication links used to pass data.
4. Assembly and elimination of the global front performed within each task.
5. Solution of the auxiliary eigenproblem.

Dimension of the Subspace

The number of eigenvalues and mode shapes is increased to determine its impact on the algorithm. All factors are kept constant when using the 64-element plate shown in Figs. 4 and 5 with test runs limited to two subspace iterations. Displayed in Fig. 6 is the speed-up relative to the increasing number of eigenvalues (q = 2, 4, 6, 8 and 10). It shows a steady increase in the speed-up from 1.68 to 1.75 for the largest two subspace dimensions even though the auxiliary eigenproblem is solved sequentially. This increase in overall speed-up is the result of higher speed-ups attained by subroutines DFRONT and RELOAD which outweigh the lower speed-up of subroutine DCONDS where the auxiliary eigenproblem is solved. In conclusion, for larger finite element problems increasing the subspace size adds no extra overhead and shows a steady increase in speed-up.

Conclusions

The parallel program described in this paper was found to be an accurate and effective algorithm to solve linear finite element eigenproblems on the Cray X-MP computer and demonstrated that speed-ups in execution time can be achieved when compared to a similar sequential algorithm (Fig. 7). In the course of this research, the following conclusions have emerged:

1. pFEDA takes advantage of the shared and "private" memory on the MIMD Cray computer while successfully using a completely connected architecture to transmit information from one processor to another.

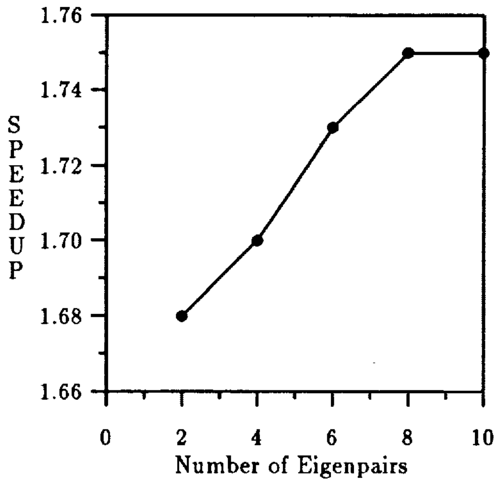


Figure 6: Impact of Increasing the Subspace Dimension

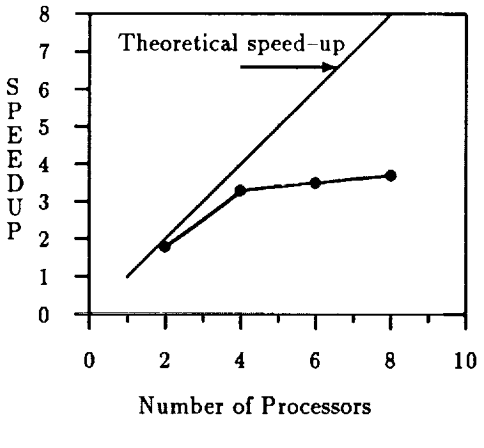


Figure 7: Effectiveness of pFEDA, $q = 6$ and Six Subspace Iterations

2. Communication links appear to be optimally synchronized. Overhead due to data contention and library routines calls are found to be of minimal impact on the performance.

3. Performance for the creation of the stiffness and mass matrices and the modified subspace method were extremely encouraging and indicate the effectiveness of multitasking environment on the Cray X-MP computer.

4. The major deficiency of pFEDA was the elimination of the degrees-of-freedom on the global front, as the domains increased so did the degrees-of-freedom on the boundary. The extra sequential calculations performed by each task to handle the global front lowered the speed-up and efficiency significantly.

5. When subdividing a finite element model into m domains one should choose the configuration with the lowest possible degrees-of-freedom on the global front for this will increase speed-up and efficiency of the parallel solution.

6. Increasing the size of the subspace creates no extra overhead even though the auxiliary eigenproblem is solved sequentially.

7. Load balancing, i.e. assigning an equivalent amount of work to each task by keeping the number of elements and frontwidth equal in all domains, is very important in the performance of pFEDA.

8. Element numbering is an important aspect in lowering the frontwidth for the frontal technique.

Acknowledgments

The authors wish to thank the Structural Dynamics Branch at NASA Lewis Research Center for their financial support and for providing access to their facilities to perform this research. The second author wishes to acknowledge the financial support received from Ohio University and the C. Paul Stocker Endowment. Support received from NASA/Case ICOMP by the first author in support of the early stages of this research is greatly appreciated.

References

1. Irons, B. M., "A Frontal Solution Program for Finite Element Analysis," *International Journal for Numerical Methods in Engineering* 2, pp. 5-32, 1970.

2. Akl, F., "Convergence of Modified Subspace Eigenanalysis Algorithm," *Proceedings of the 8th Conference on Electronic Computation, ASCE*, Houston, Texas, pp. 416-421, February 1983.

3. Storaasli, O., Bostic, S., Patrick, M., Mahajan, U. and Shing, M., "Three Parallel Computation Methods for Structural Vibration Analysis," AIAA Paper 88-2391, *AIAA/ASME/ASCE/AHS 29th Structres, Structural Dynamics and Materials Conference*, Williamsburg, VA, pp. 1401-1411, April 1988.

4. Bathe, K. and E. L. Wilson, *Numerical Methods in Finite Element Analysis*, Prentice-Hall, Inc., 1976.

5. Melhem, R. G., *A Modified Frontal Technique Suitable for Parallel Systems*, Technical Report ICMA-85-84, July 1985.

6. Cray Research, Inc., *Cray X-MP Multitasking Programmer's Reference Manual*, SR-0222, July 1987.

7. Akl, F., W. H. Dilger and B. M. Irons, "Acceleration of Subspace Iteration," *International Journal for Numerical Methods in Engineering*, Vol. 18, No. 4, pp. 583-589, 1982.

8. Akl, F., W. H. Dilger and B. M. Irons, "Over-relaxation and Subspace Iteration," *International Journal for Numerical Methods in Engineering*, Vol. 14, No. 4, pp. 629-630, 1979.

9. Akl, F., Dilger, W. H. and Irons, B. M., *"FEDA-A Finite Element Dynamic Analysis Program"*, Research Report, Department of Civil Engineering, The University of Calgary, Canada, September 1979.

Appendix A

A brief description is given for specific subroutines in pFEDA. Similar subroutines used in FEDA[1,9] are documented and will not be repeated here.

1. Subroutine DMATRON:
   The data is checked for fatal or non-fatal errors. This routine then determines the last appearance of each node (pre-front). The size of the global front is also calculated.

2. Subroutine ESTIFF:
   Creates the element stiffness and mass matrices. It also creates the element initial eigenvectors.

3. Subroutine DFRONT:
   Assembles and eliminates the stiffness matrix for the degrees-of-freedom in their last appearance up to the global front. After the global front is reached and boundry interface matrices are transmitted to all tasks, the global front can be assembled and eliminated. Immediately afterward back-substitution begins to calculate the unknown variables within each domain.

4. Subroutine DCONDS:
   Calculates the projection of $[K]$ and $[M]$ onto the current subspace for each iteration and communicates them to all other user tasks. After this is completed , the auxiliary eigenproblem is solved. Next, a better M-orthonormalized approximation of the required eigenvectors is constructed.

5. Subroutine RELOAD:
   After each iteration, a set of new fictitious inertia loads is calculated for each element to be used in the new subspace iteration step.

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No. NASA TM-102450 ICOMP-89-31; AIAA-89-1395	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Eigensolution of Finite Element Problems in a Completely Connected Parallel Architecture		5. Report Date
		6. Performing Organization Code
7. Author(s) Fred A. Akl and Michael R. Morel		8. Performing Organization Report No. E-5235
		10. Work Unit No. 505-62-21
9. Performing Organization Name and Address National Aeronautics and Space Administration Lewis Research Center Cleveland, Ohio 44135-3191		11. Contract or Grant No.
		13. Type of Report and Period Covered Technical Memorandum
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, D.C. 20546-0001		14. Sponsoring Agency Code

15. Supplementary Notes

Presented at the 30th Structures, Structural Dynamics and Materials Conference cosponsored by the AIAA, ASME, ASCE, AHS, and ACS, Mobile, Alabama, April 3-5, 1989. Fred A. Akl, Ohio University, Dept. of Civil Engineering, Athens, Ohio 45701-2979 and Institute for Computational Mechanics in Propulsion, Lewis Research Center (work funded by Space Act Agreement C99066G). Michael R. Morel, Ohio University, Dept. of Civil Engineering. Space Act Monitor, Louis A. Povinelli.

16. Abstract

(omega)

This paper presents a parallel algorithm for the solution of the generalized eigenproblem in linear elastic finite element analysis, $[K][\Phi] = [M][\Phi][\Omega]$, where: $[K]$ and $[M]$ are of order N, and $[\Omega]$ is of order q. The parallel algorithm is based on a completely connected parallel architecture in which each processor is allowed to communicate with all other processors. The algorithm has been successfully implemented on a tightly coupled multiple-instruction-multiple-data (MIMD) parallel processing computer, Cray X-MP. A finite element model is divided into m domains each of which is assumed to process n elements. Each domain is then assigned to a processor, or to a logical processor (task) if the number of domains exceeds the number of physical processors. The macrotasking library routines are used in mapping each domain to a user task. Computational speed-up and efficiency are used to determine the effectiveness of the algorithm. The effect of the number of domains, the number of degrees-of-freedom located along the global fronts and the dimension of the subspace on the performance of the algorithm are investigated. For a 64-element rectangular plate, speed-ups of 1.86, 3.13, 3.18 and 3.61 are achieved on two, four, six and eight processors, respectively.

- (K)(phi) = (M)(phi)(omega),

17. Key Words (Suggested by Author(s)) Eigenvalue; Eigenvector; Parallel computers; Multifront; Finite elements; Vibration; Dynamics; Subspace	18. Distribution Statement Unclassified – Unlimited Subject Category 39

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price*
Unclassified	Unclassified	11	A03

NASA FORM 1626 OCT 86

*For sale by the National Technical Information Service, Springfield, Virginia 22161