NASA Technical Memorandum 89097

A LANCZOS EIGENVALUE METHOD ON A PARALLEL COMPUTER

SUSAN W. BOSTIC AND ROBERT E. FULTON

(NASA-TM-89097) A LANCZOS EIGENVALUE METHOD ON A PARALLEL COMPUTER (NASA) 20 p CSCL 09B N87-20768

Unclas
G3/62 45356

MARCH 1987

National Aeronautics and Space Administration

Langley Research Center
Hampton, Virginia 23665

A LANCZOS EIGENVALUE METHOD ON A PARALLEL COMPUTER

Susan W. Bostic and Robert E. Fulton

NASA Langley Research Center
Hampton, Virginia 23665-5225

Abstract

Eigenvalue analyses of complex structures is a computationally intensive task which can benefit significantly from new and impending parallel computers. This study reports on a parallel computer implementation of the Lanczos method for free vibration analysis. The approach used here subdivides the major Lanczos calculation tasks into subtasks and introduces parallelism down to the subtask levels such as matrix decomposition and forward/backward substitution. The method was implemented on a commercial parallel computer and results were obtained for a long flexible space structure. While parallel computing efficiency is problem and computer dependent, the efficiency for the Lanczos method was good for a moderate number of processors for the test problem. The greatest reduction in time was realized for the decomposition of the stiffness matrix, a calculation which took 70 percent of the time in the sequential program and which took 25 percent of the time on eight processors. For a sample calculation of the twenty lowest frequencies of a 486 degree of freedom problem, the total sequential computing time was reduced by almost a factor of ten using 16 processors.

Nomenclature

B	decomposition of mass matrix
D	diagonal matrix
K	stiffness matrix
L	lower triangular matrix
M	mass matrix
T_m	tridiagonal matrix of mth order
V_i	Lanczos vector
X	vector of degrees of freedom
α_i, β_i	elements of tridiagonal matrix
δ	shift parameter
λ	eigenvalue
ω^2	frequency parameter

Introduction

The eigenvalue problem associated with free vibration analysis of complex structures is one of the more computationally intensive tasks in the design of modern aerospace vehicles. With more sophisticated vehicle designs and attendant detailed analyses, structural models composed of thousands of degrees of freedom are not uncommon and the associated free vibration analyses could require hours of computing time.

Recent and impending advances in computing capabilities indicate that effective computing speeds will soon approach 10 GIGA FLOPS (10^{10} floating point operations per second) (ref. 1). Such computing speeds are being achieved by the development of innovative computer architectures consisting of arrays of processors operating in parallel. These parallel computing architectures are also being scaled down to computers whose price range is less than $1M and whose effective computer speeds approach 1 GIGA FLOPS. These trends indicate that such parallel computers can significantly improve the capability for large scale free vibration analyses. However, the hardware development of this new class of computers has surpassed the necessary software development needed to utilize fully the computational power available. The key to achieving peak performance is the modification of existing algorithms or the development of new methods tailored to parallel computers.

There are many sophisticated, efficient sequential eigenvalue solvers now available (refs. 2-9). Parlett (ref. 9) discusses these methods and their effectiveness. In order to adapt these or other algorithms to a parallel computing environment these methods must be examined to identify the possibility for parallelism in the computation steps and how the methods' performances can be improved by parallel computations. This paper focusses on one important method, the Lanczos method, (refs. 10-17) to develop a comprehensive parallel strategy for vibration analysis. The study builds on earlier work (ref. 17) with a parallel Lanczos method, introduces parallelism at lower levels of the program by subdividing tasks into subtasks and develops software capability to facilitate control of parallel eigenvalue calculations.

Parallel Implications for Eigenvalue Methods

Several of the available eigenvalue methods are roughly categorized as determinant methods, rotation methods and iterative methods (refs. 2, 4, 18-20). The type of method used for a specific application depends in large part on the type of problem being solved, the size of the problem and the desired results. Sparse matrix operations, of the type encountered in structural problems, can provide special features which some algorithms do not address. The number of eigenvalues desired and their location in the overall spectrum may also require a specific strategy. Reference 9 indicates that, due to its efficiency, the Lanczos method is a key method for structural applications. In view of its growing acceptance, the Lanczos method was selected here for study for parallel implementation.

The most comprehensive computer programs include several eigenvalue methods. An investigation of the various methods shows that many have certain calculation steps in common. These major calculation steps become areas of attention for reducing calculation time on a parallel computer. Since some methods, such as the Lanczos method, are mixed approaches, parallel algorithms which speed up calculations at the subtask level can be easily integrated into more than one method.

Levels of Parallelism

In dissecting a solution process for parallel implementation, one looks for those tasks that can be carried out concurrently with a minimum exchange of information or data dependencies. There are many levels or granularities of parallelism. An example of a high-level, or coarse-grained, degree of parallelism is the concurrent execution of large sections of a program on a number of processors. Further decomposition can lead to subsequent levels of parallelism, from the highest level to the very-fine-grained parallel execution of a few arithmetic operations simultaneously on several processors. Parallelism always introduces overhead and by examining the amount of overhead introduced, one can assess the benefits to be reaped and then decide when to further divide a task into subtasks. The development of parallel algorithms is complicated by the interactions of many processors operating concurrently. The algorithm must consider such things as: communication overhead, memory contention, critical regions which contain code that must be executed sequentially, the time to initiate tasks, data interdependency, and idle time resulting from an imbalance of the workload. Potential speedups may be relatively small at the lowest level of granularity, i.e., individual arithmetic expressions. This study investigates the level of granularity that will produce the most efficient performance.

The Lanczos Method

For the implementation in this study, the Lanczos method was used as representative of a multi-step process which lends itself to concurrent calculations. The basic Lanczos method is briefly summarized in Appendix A. The method seems well suited for the type of problems often encountered in vibration studies of large space structures.

It is most efficient when solving for a few extreme eigenvalues of a very large system. This same property makes the method applicable to parallel processing since it is efficient to use several processors to compute a few values rather than to solve for many values in a single sequential solution. An implementation of the Lanczos method where parallelism is initiated at a high level of granularity by introducing shifts and having separate processors solve for eigenvalues in different areas of the spectrum is documented in reference 17. An example of the speedups obtained for a truss vibration example problem using this strategy is shown in figure 1. Speedup is defined as the time it takes to do a calculation on one processor divided by the time it takes on multiple processors. The theoretical speedup is the optimum achievable when all processors are operating at 100 per cent efficiency. Significant speedups were obtained for up to eight processors for the truss problem in figure 1.

The next step in a parallel Lanczos implementation is to decompose the major tasks into subtasks and map them onto several processors. The sequential order of steps in the basic Lanczos procedure is shown in figure 2. The steps include initialization, which includes the introduction of the shift, decomposition of the stiffness matrix, forward solution, back substitution, calculation of the Lanczos vectors and, finally, the solution of the resulting tridiagonal system of equations by the bisection method. By computing a finite number of vectors, a tridiagonal matrix is constructed that approximates the eigenvalues of the original large problem. The parallel strategy is to assign subtasks of these major tasks to available processors as needed. An implementation of the parallel strategy was carried out on a Flexible Computer Multi-computer FLEX/32.

The FLEX/32 is a multiple-instruction multiple-data (MIMD) computer with both shared and local memory (ref. 21). The FLEX/32 configuration used in this study consists of 20 processors. Two processors are dedicated to program development and 18 are dedicated to parallel processing.

Although, in theory, all computed eigenvalues would be good approximations to the actual answers, in practice this does not hold true. Due to the loss of orthogonality in the computed vectors, only some results can be accepted. Reference 11 clarifies orthogonalization issues; methods for best determining acceptable eigenvalues is a subject of ongoing research (refs. 4, 9, 11, 14). An extensive discussion of the method and reorthogonalization procedures can be found in reference 14.

Some approaches include total or selective reorthogonalization with respect to previously calculated vectors. Another less computationally expensive approach recommended by Cullum and Willoughby (refs. 14, 17) provides an easy test for selecting valid eigenvalues from those obtained through Lanczos calculations. In this study, the Cullum and Willoughby test is used because in comparison with reorthogonalization, it was found to be reliable and efficient.

Space Mast Problem

The test problem used in this study is the 60-meter three-longeron truss structure shown in figure 3 attached as a mast to the space shuttle orbiter with an antenna attached. For this study, the mast is considered rigidly connected to the orbiter and the antenna is not considered. The mast is composed of extensional members and masses are concentrated at the nodes. The details of the mast and node members are shown in the figure where the mast is organized according to major substructures. For this problem, there are three degrees of freedom at each unconstrained node resulting in a model of 486 degrees of freedom.

Representative vibration results for the space mast are shown in figure 4 and the sequential times for each Lanczos step on a single processor are shown in figure 5. For the test problem, the decomposition of the stiffness matrix K into the product of a diagonal matrix D and a lower triangular matrix L took 70 percent of the solution time. As the problem size grows, the time taken to decompose the stiffness matrix grows accordingly. For this algorithm, the decomposition is done once for each shift value. In a nonlinear analysis, the decomposition would be done at every time step, making any reduction in calculation time even more meaningful. Since this step seemed to offer the most benefits, it was incorporated into the parallel code first.

Parallel Implementation

The first strategy in parallelizing the decomposition step is shown in figure 6 which depicts a four-processor implementation. This strategy represents a pre-scheduled assignment of tasks done in a row interleave fashion. Each processor p is assigned the calculation necessary for rows p, $p+n$, $p+2n$, ..., where n is the number of processors. The computed values are stored in shared memory where they are accessible to all processors. The processors must be synchronized to ensure that values needed for the next step are already computed by the assigned processors and stored. In this implementation, the synchronization accounts for most of the overhead associated with the parallelism.

For comparison, a second strategy was adopted to parallelize the decomposition step. In this case, a queue of tasks is set up in shared memory and each processor takes its next work assignment from this queue. The timing for this self-scheduled strategy was almost identical to the pre-scheduled task assignment. Timings given in the following figures were obtained using this self-scheduled task assignment. One advantage of the self-scheduling is that if one processor should fall behind in its calculation or experience hardware problems, the other processors would continue the calculations.

The timing results for decomposing the $n \times n$ matrix, where n is equal to 486, into a lower triangle matrix and a diagonal array by using up to 16 processors are shown in figure 7. The speedups are shown in figure 8. Significant speedups are obtained for up to eight processors for the decomposition of the stiffness matrix. Using more than eight processors for this problem did not result in significant time reduction. The size of the matrix and particularly, the size of the bandwidth determines the amount of calculation in the decomposition step. As the order of the matrix approaches infinity, the number of arithmetic operations in the decomposition step approaches $1/2\ (n)\ (s)\ (s+1)$, where s is the semi-bandwidth and n is the order of the matrix. Since the matrix is banded, the calculation of the zero elements is ignored. This means that the work becomes equal on each processsor when the assigned row number is equal to or greater than the semi-bandwidth, which in this case is 18. When 16 processors are working concurrently on the decomposition of the same matrix, each processor must wait for 15 others to compute a needed value at each step.

The forward solution and back substitution steps (fig. 9) were also parallelized in a row interleave fashion. One of the decisions that has to be made when using the Lanczos algorithm is to determine the order of the resulting tridiagonal matrix. This order represents the number of times the forward solution, back substitution and vector calculation steps are carried out. If the order is m, then m eigenvalues will be found. Not all of these eigenvalues are valid approximations to the eigenvalues of the original problem, since redundant and/or spurious values may appear (refs. 14, 17). One rule of thumb is to make m twice as large as the number of eigenvalues desired.

To study the effect of the choice of m, various values were used to calculate the eigenvalues of the space mast problem. A plot of the time for computing one acceptable eigenvalue is shown in figure 10 for three values of m. It was found that an m equal to 30 gave the most information for the least time. The plot shows about the same time for m equal to 16 as for m equal to 30. However, in addition to the acceptable values, approximate values are obtained for additional eigenvalues for the larger value of m. The first 25 eigenvalues obtained for m equal to 30 are given in table 1. Tests were then made to eliminate the multiple and spurious eigenvalues. The acceptable values are marked with an asterisk. These values compare favorably to values found using an independent structural analysis code (ref. 22).

Timing results for the forward solution with m equal to 30 are shown in figure 11. This step consists of much less computation time compared to synchronization time than in the decomposition step. The speedups shown in figure 12 show a gain on up to four processors. For this size problem, there is actually a decrease in the speedup when using eight processors for the forward solution step.

The amount of computation in the back substitution step is less than in the forward solution step. Timimg results shown in figure 13 indicate that more than four processors dedicated to this step for this problem would be ineffective.

A plot of the execution time versus the number of processors for the overall Lanczos method is shown in figure 14. A plot of the speedups for the overall method is shown in figure 15. The execution time for the various steps on several processors is shown in table 2. The decrease in execution time is due only to the parallelization of the decomposition, forward solution, and back substitution steps. Steps which have not been done in parallel include: initialization, calculation of the Lanczos vectors and the bisection. For this problem these steps took a relatively small amount of time and implementation of parallel processing is only of minor benefit.

To provide a measure of the total efficiency of a parallel Lanczos method, timing results are shown in table 3 for the test problem where four shifts are introduced and shift calculations are assigned to four separate sets of processors. An average of five valid eigenvalues was obtained for each shift region. The table shows that it takes 876 seconds to run the sequential program using four different shifts. When the four shifts are each assigned to a set of four processors for a total of 16 processors, the time is reduced to 89 seconds. Timing reductions resulting from a differing number of processors assigned to each shift are shown in figure 16.

Concluding Remarks

This study has focussed on a parallel computer implementation of the Lanczos method for free vibration analysis. The approach used extends previous work by subdividing the major Lanczos calculation tasks into subtasks and introducing parallelism at the subtask level.

Results were obtained for a long flexible space structure test problem and the method was implemented on a commercial parallel computer. The results of the study indicate the Lanczos method is a promising method for providing calculation speedups when tasks are subdivided and assigned to several processors. Since the method is most efficient when solving for a few eigenvalues at the extreme ends of the spectrum, several processors can be working concurrently in different areas of the eigenvalue spectrum. Subdividing the Lanczos tasks among processors provides the best use of the parallel resources. The decomposition step is the most time-consuming calculation step for large problems when only a few Lanczos iterations are performed. The results show that for this test problem, using eight processors for the decomposition of the stiffness matrix was the optimum. In subtasks where the calculation time is relatively short with respect to the synchronization time, such as in the forward and backward solution steps, no more than eight processors were effective. The efficiency is problem dependent and the number of processors must be tailored to the specific application.

By assigning a set of processors to an interval in the eigenvalue spectrum, eigenvalues can be obtained faster than on a sequential computer of the same speed. A combination of coarse-grained parallelism at the program level and fine-grained parallelism at the task level results in significant reductions in solution time. Parallel processors offer the structural analyst an effective tool for solving large structural problems in a timely fashion.

Appendix A

Lanczos Eigenvalue Method

The following briefly outlines the Lanczos algorithm strategy as implemented in this study. Consider the eigenvalue problem

$$KX = \omega^2 MX \tag{A1}$$

where X is the displacement vector, K and M are symmetric stiffness and mass matrices, respectively, and ω^2 is the frequency parameter. Introduce an eigenvalue shift parameter δ such that

$$\omega^2 = \delta + \bar{\omega}^2 \tag{A2}$$

and decompose M into

$$M = BB^T \tag{A3}$$

where B is a lower triangular matrix.

Equation (A1) is then transformed to the Lanczos format (ref. 7, 9-19)

$$AY = \lambda Y \tag{A4}$$

where

$$A = B^T [\bar{K}^{-1}] B \qquad Y = B^T X \tag{A5}$$

and

$$\lambda = \frac{1}{\bar{\omega}^2}, \quad \bar{K} = K - \delta M \tag{A6}$$

In equation (A4), A is symmetric since K is symmetric. The solution to equation (A4) by the Lanczos algorithm yields increasingly good approximations to the largest eigenvalues λ which correspond to the smallest $\bar{\omega}^2$ and the closest ω^2 values to a reference value δ. If a sufficient number of Lanczos cycles are carried out, the method will theoretically yield all eigenvalues.

The Lanczos transformation is

$$AV = VT \tag{A7}$$

where T is a symmetric tridiagonal matrix and V is an orthogonal rectangular matrix with $V^TV = I$.

Equation (A7) and the orthogonality condition imply

$$V^TA\,V = T$$

which transforms equation (A4) into the eigenvalue problem

$$TQ = \lambda Q \tag{A8}$$

where

$$Q = V^TY \tag{A9}$$

One of the key features of the Lanczos method is that if V is rectangular the larger eigenvalues of the reduced order equation (A8) are good approximations of the larger eigenvalues of (A4).

Let the column vectors of V be denoted $(V_1, V_2, V_3, \ldots)$ and the elements of T be denoted

$$T = \begin{bmatrix} \alpha_1 & \beta_1 & & & & \\ \beta_1 & \alpha_2 & \beta_2 & & & \\ & \beta_2 & \alpha_3 & \beta_3 & & \\ & & \cdot & \cdot & \cdot & \\ & & & \cdot & \cdot & \cdot \\ & & & \beta_{n-2} & \alpha_{n-1} & \beta_{n-1} \\ & & & & \beta_{n-1} & \alpha_n \end{bmatrix} \tag{A10}$$

Equation (A7) gives

$$AV_i = \beta_{i-1}V_{i-1} + \alpha_iV_i + \beta_iV_{i+1} \tag{A11}$$

when V_i is the ith column of V.

To obtain AV_i efficiently, note from (A5) that

$$A = B^T\,\bar{K}^{-1}B \tag{A12}$$

and decompose $\bar{K}$ into

$$\bar{K} = LDL^T \tag{A13}$$

where L is a unit triangular matrix and D is a diagonal matrix. Equation (A12) results in

$$AV_i = B^T L^{-T} (LD)^{-1} BV_i \tag{A14}$$

or

$$(B^T L^{-T})^{-1} AV_i = (LD)^{-1} BV_i = a_i \tag{A15}$$

where a_i is an intermediate vector. Equation (A15) gives

$$LD\, a_i = BV_i \tag{A16}$$

and $$B^T L^{-T} a_i = \tilde{K}^{-1} V_i \tag{A17}$$

Letting $$L^{-T} a_i = b_i \tag{A18}$$

results in $$L^T b_i = a_i \tag{A19}$$

and $$AV_i = B^T b_i \tag{A20}$$

Equations (A16), (A19), and (A20) become the equations for determining AV_i. The sequence for determining the V_i uses an arbitrary starting vector (e.g., $V_o = (1, 0, 0, 0, \ldots)$) and then calculates AV_i from equations (A16), (A19), and (A20). The remaining steps to obtain α_i, β_i and V_{i+1} are based on the Gram-Schmidt orthogonalization as follows

$$W_i = AV_i - \beta_{i-1} V_{i-1} \quad \text{where } \beta_o = 0$$
$$i = 1, 2, \ldots, n \tag{A21}$$

$$\alpha_i = V_i^T W_i \tag{A22}$$

$$C_i = W_i - \alpha_i V_i \tag{A23}$$

NOTE: For reorthogonalization insert: $c_i = C_i - \sum_{j=1}^{j=i-1} V_j^T c_i V_j$ (A23a)

$$\beta_i = [C_i^T C_i]^{\frac{1}{2}} \tag{A24}$$

$$V_{i+1} = \frac{1}{\beta_i} C_i \tag{A25}$$

The cycles are repeated to produce a set of m vectors $V_i, \ldots, V_m$ ($m \leq n$) and associated α_i and β_i. The eigenvalues may be obtained at any stage for the m by m equation

$$TQ = \lambda Q \tag{A26}$$

where T is composed of the α_i and β_i calculated to that point.

The eigenvalues of equation (A26) can be obtained by the Sturm sequence and bisection methods. Let the principal minors of equation (A26) be denoted

$$P_0 = 1 \qquad P_1(\lambda) = \alpha_1 - \lambda \tag{A27}$$

$$P_i(\lambda) = (\alpha_i - \lambda)P_{i-1}(\lambda) - (\beta)^2_{i-1}\, P_{i-2}(\lambda)$$
$$(i = 2, \ldots, m \leq n) \tag{A28}$$

Direct calculation of these sequences can result in overflow or underflow; the following sequence avoids the need to rescale.

$$q_i(\lambda) = \frac{P_i(\lambda)}{P_{i-1}(\lambda)}$$

to give

$$q_0 = 1,\ q_1 = \alpha_i - \lambda \tag{A29}$$

$$q_i(\lambda) = (\alpha_i - \lambda) - \frac{\beta^2_{i-1}}{q_{i-1}(\lambda)} \tag{A30}$$

The Sturm sequence property for equations (A27) and (A28) is that for a specific value of λ^*, the disagreements in sign between consecutive numbers of the sequence $P_i(\lambda^*)$ is equal to the number of eigenvalues smaller than λ^*.

In the bisection method, the Sturm sequence property is used to restrict the interval in which a particular eigenvalue must lie, until the eigenvalue is predicted to sufficient accuracy. An interval in question is halved and the Sturm sequence calculated to determine the number of eigenvalues in the two intervals. The interval containing the largest eigenvalue is subsequently halved and the process repeated until the largest eigenvalue has been isolated to sufficient accuracy. This approach is then applied to determine the next lower eigenvalues in order of the relative size. When in the neighborhood of an eigenvalue, it is often more efficient to switch from the bisection method to an interpolation scheme; the bisection method is also well suited for implementation on a parallel computer as any number of eigenvalues can be calculated in parallel.

For a given eigenvalue λ_K the associated eigenvector of equation (A26) Q_K is found from equation (A26). The corresponding eigenvector of (A4) Y_K is then given by

$$Y_K = V\, Q_K \tag{A31}$$

and the X_K for equation (A1) can be found from

$$B^T X_K = Y_K \tag{A32}$$

The eigenvalue is transformed to the desired frequency parameter by

$$\omega_K^2 = \delta + \frac{1}{\lambda_K}. \tag{A33}$$

References

1. Noor, A. K., Storaasli, O. O. and Fulton, R. E., "Finite Element Technology in the Future," Impact of New Computations on Computational Mechanics. (Noor and Pilkey, Editors), ASME Special Publication H00275, pp. 1-32, November 1983.

2. Ojalvo, I. U., "ALARM--A Highly Efficient Eigenvalue Extraction," The Shock and Vibration Digest, Vol. 7. No. 12, December, 1975.

3. Wilkinson, J. H., The Algebraic Eigenvalue Problem, Clarendon Press, Oxford, 1965.

4. Parlett, B., The Symmetric Eigenvalue Problem, Prentice-Hall, Englewood Cliffs, New Jersey, 1980.

5. Jennings, A., Matrix Computation for Engineers and Scientists, Wiley, New York, 1977.

6. Bathe, K. J. and Wilson, E. L., Numerical Methods in Finite Element Analysis, Prentice-Hall, Englewood Cliffs, New Jersey, 1976.

7. Golub, Gene H. and Van Loan, Charles F., Matrix Computations, The Johns Hopkins University Press, Baltimore, Maryland, 1983.

8. Ojalvo, Irving U., "Computer Methods for Determining Vibration Modes of Complex Structures," Shock and Vibration Digest, Vol. 5, Issue 5, May 1973.

9. Parlett, Beresford N., "The State-of-the-Art in Extracting Eigenvalues and Eigenvectors in Structural Mechanics Problems," Department of Mathematics, University of California, Berkeley, 1986.

10. Lanczos, C., "An Iteration Method for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operators," J. Res. Natl. Bureau of Standard, Vol. 45, pp. 255-282, 1950.

11. Paige, C. C., "Accuracy and Effectiveness of the Lanczos Algorithm for the Symmetric Ax = λBx Problem," Rep. STAN-CS-72-270, Stanford University Press, Palo Alto, CA, 1972.

12. Geradin, M., "On the Lanczos Method for Solving Large Structural Eigenvalue Problems," ZAMM, Vol. 59, pp. T127-T129, 1979.

13. Weingarten, V. I., Ramanathan, R. K., and Chen, C. N. "Lanczos Eigenvalue Algorithm for Large Structures on a Minicomputer," Computers and Structures, Pergamon Press Ltd., vol. 16, no. 1-4, pp. 253-257, 1983.

14. Cullum, J. and Willoughby, R. A., Lanczos Algorithms for Large Symmetric Eigenvalue Computations. Vol. 1 Theory, Birkhauser Boston Inc., 1985.

15. Son, H. D., "The Lanczos Algorithm for Solving Symmetric Linear Systems," Center for Pure & Applied Mathematics, University of California at Berkeley, PAM-74, April 1982.

16. Golub, G. H., Underwood, R. and Wilkinson J.H., "The Lanczos Algorithm for the Symmetric Ax = λBx Problem." Rep. STAN -CS-72-270, Stanford Univ. Press, Palo Alto, CA, 1972.

17. Bostic, S. W. and Fulton, R. E., "Implementation of the Lanczos Method for Structural Vibration Analysis on a Parallel Computer." Presented at the AIAA/ASME/ASCE/ AHS 27th SDM Conference, San Antonio, TX, May 19-21, 1986. AIAA Paper No. 86-0930-CP.

18. Fulton, R.E., "The Impact of Parallel Computing on Finite Element Computations." Reliability of Methods for Engineering Analysis. K. J. Bathe and D. R. J. Owens, Eds. Pineridge Press, Swansea, U.K., pp. 179-196, 1986.

19. Fredricksson, B. and Mackerle, J., "Partial List of Major Finite Element Programs and Description of Some of Their Capabilities." State-of-the-Art Surveys on Finite Element Technology, (Noor and Pilkey, Editors). ASME Pub. H00290, pp. 363-403, 1983.

20. Butler, T. and Michel, D., "NASTRAN, A Summary of Functions and Capabilities of the NASA Structural Analysis Computer System," NASA SP-260, 1971.

21. Matelan, N., "The Flex/32 Multicomputing Environment in Research in Structures and Dynamics - 1984," R. J. Hayduk and A. K. Noor (compilers), NASA CP 2335, 1984.

22. Lotts, C. G., Greene, W. H., McCleary, S. L., Knight, N. F., Jr., and Gillian, R. E., "Introduction to the CSM Testbed: NICE/SPAR," NASA TM 890964, 1987.

Table 1 Multiprocessor Eigenvalue Approximations

Eigenvalue shift = 0.0 Order of tridiagonal matrix = 30

Eigenvalues calculated before elimination of multiplicities	NICE/SPAR results (ref. 22)	error
10.04 *	10.04	4x10-10
10.04		
10.04		
10.04		
10.04		
10.04		
11.74 *	11.76	6x10-9
11.74		
11.74		
11.74		
11.74		
11.74		
380.2 *	380.1	1x10-3
380.2		
380.2		
442.0 *	443.8	5x10-3
442.0		
766.3	1001.	1x10-1
766.3		
766.3		
2876. *	2835.	2x10+1
3649.	3275.	2x10+1
7890.		
7890.		
18835.		

* accepted as valid approximation

Table 2 Time to solve for twenty valid eigenvalues

Number of shifts	Number of processors assigned to each shift region	Total number of processors	Time (seconds)	Speedup
4	1	1	876	1
4	1	4	227	3.9
4	2	8	130	6.7
4	4	16	87	9.8

Table 3 Execution time for Lanczos operations

		Number of processors				
		1	2	4	8	16
1.	**Initialization** $\tilde{K} = K - \delta M$	5	5	5	5	5
2.	**Decomposition** $\tilde{K} = L D L^T$	156	78	39	20	17
3.	**Forward solution** $L D a = V$	32	18	16	20	20
4.	**Back substitution** $L^T y = a$	17	10	8	15	15
5.	**Calculation of vectors** α, β, v	6	6	6	6	6
6.	**Bisection** $T_m Q = \lambda Q$, m (order of T) = 30	3	3	3	3	3
	Total	219	120	77	69	66

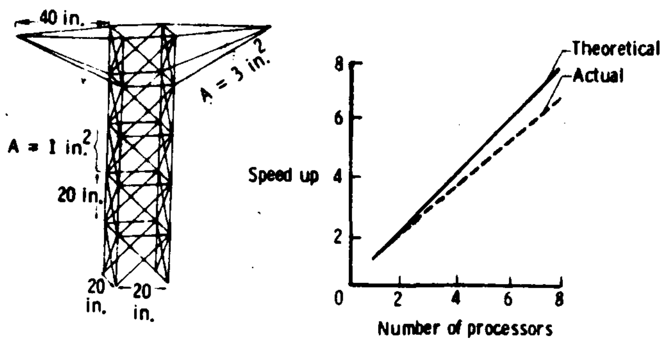


Fig. 1 Speedup for truss vibration problem.

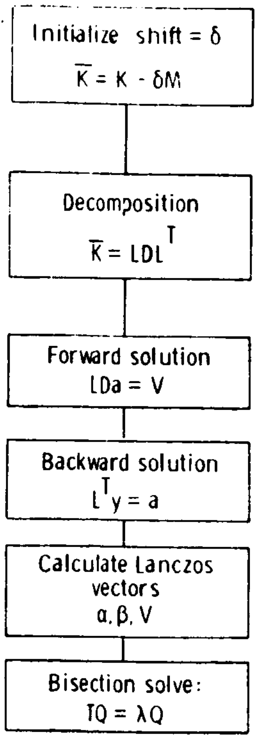
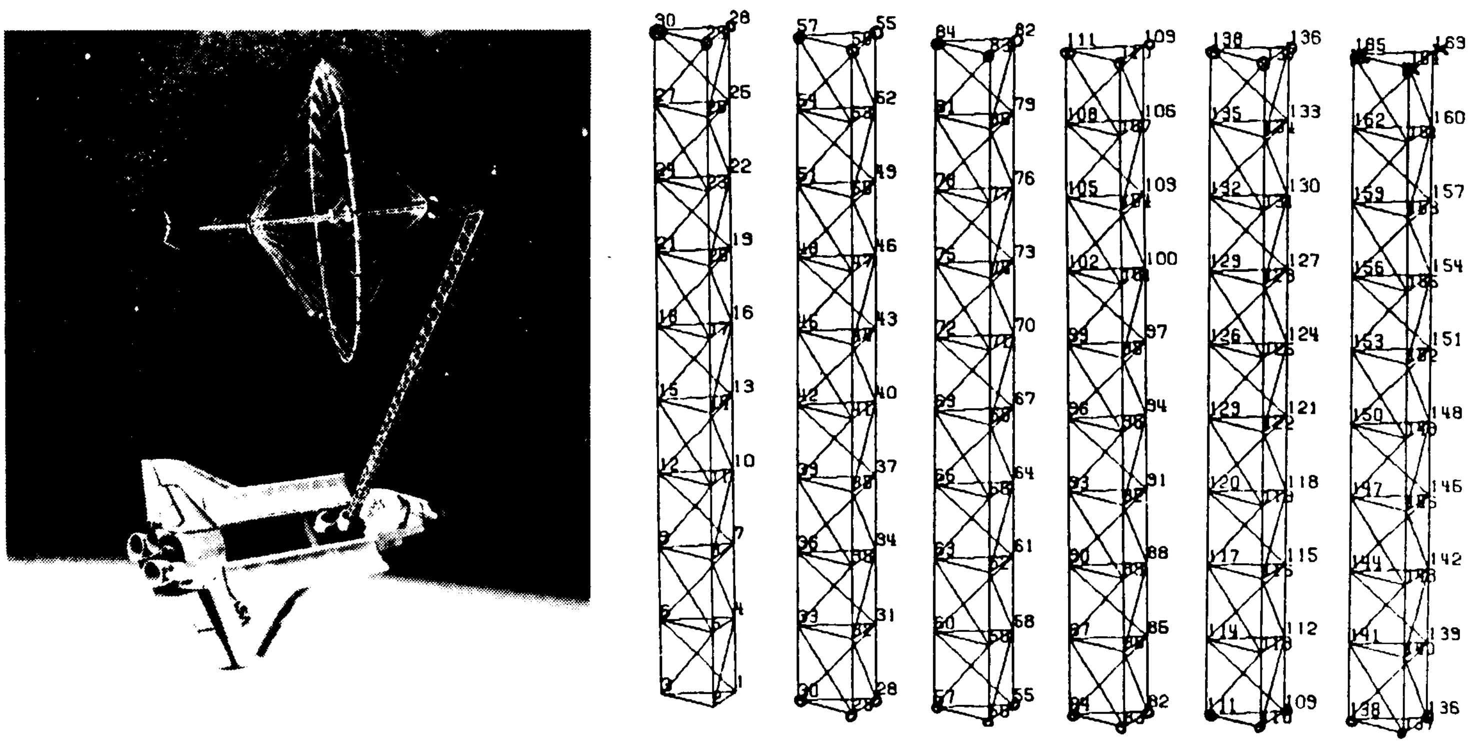


Fig. 2 Lanczos method sequential operations.



60 - meter truss
composed of 54 bays

165 nodes
486 degrees of freedom

Fig. 3 Space mast problem.

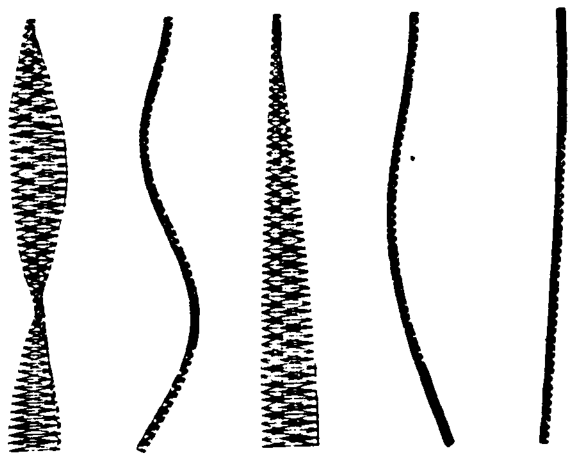


Fig. 4 Representative mode shapes of 60-meter mast.

Processor	
1	D_1
2	$L_{21}\ D_2$
3	$L_{31}\ L_{32}\ D_3$
4	$L_{41}\ L_{42}\ L_{43}\ D_4$
1	$L_{51}\ L_{52}\ L_{53}\ L_{54}\ D_5$
2	$L_{61}\ L_{62}\ L_{63}\ L_{64}\ L_{65}\ D_6$
3	$L_{71}\ L_{72}\ L_{73}\ L_{74}\ L_{75}\ L_{76}\ D_7$
4	$L_{81}\ L_{82}\ L_{83}\ L_{84}\ L_{85}\ L_{86}\ L_{87}\ D_8$
1	$L_{91}\ L_{92}\ L_{93}\ L_{94}\ L_{95}\ L_{96}\ L_{97}\ L_{98}\ D_9$
•	
•	
•	

$$L_{ij} = \frac{K_{ij} - \sum_{k=1}^{k=j-1} (L_{ik} \times L_{jk} \times D_k)}{D_j}$$

$$D_i = K_{ii} - \sum_{k=1}^{k=i-1} (D_k \times L_{ik} \times L_{ik})$$

Fig. 6 Four-processor implementation of LDL^T matrix decomposition.

Lancos method operation	Time in seconds
1. Initialization $\overline{K} = K - \delta M$	5
2. Decomposition $\overline{K} = LDL^T$	156
3. Forward solution $LDa = V$	32
4. Back substitution $L^T y = a$	17
5. Calculation of Lancoz vectors α, β, V	6
6. Bisection $T_m Q = \lambda Q$, m (order of T) = 30	3
	219

Fig. 5 Execution time for sequential Lanczos method.

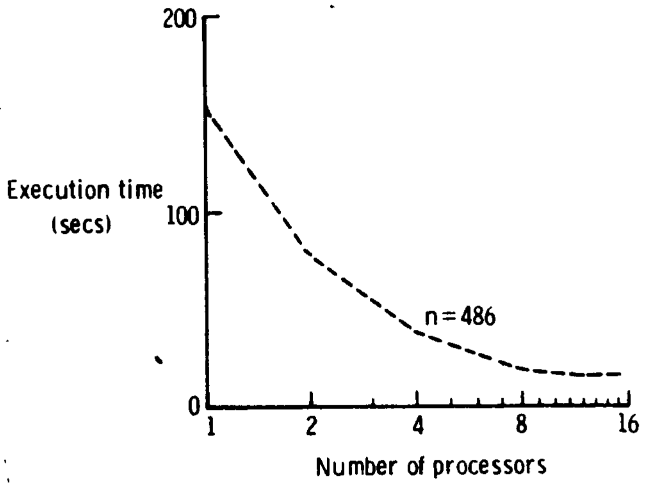


Fig. 7 Execution time for n x n matrix decomposition.

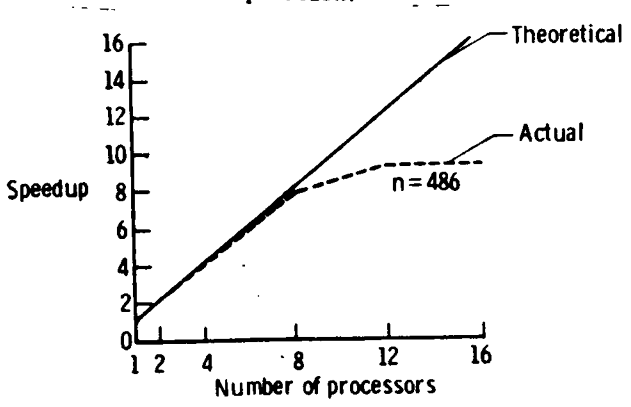


Fig. 8 Speedup for n x n matrix decomposition.

Forward solution

$$\begin{array}{c|l} \text{Processor} & \\ 1 & D_1 \\ 2 & L_{21}\ D_2 \\ 3 & L_{31}\ L_{32}\ D_3 \\ 4 & L_{41}\ L_{42}\ L_{43}\ D_4 \\ \vdots & \end{array} \quad \left\{ a \right\} = \left\{ v \right\}$$

$$a_i = v_i - \sum_{j=1}^{j<i} Lij * D_j * a_j$$

Back substitution

$$\begin{array}{c|l} \text{Processor} & \\ 1 & 1\ L_{21}\ L_{31}\ L_{41} \cdots \\ 2 & \quad 1\ L_{32}\ L_{42} \cdots \\ 3 & \qquad 1\ L_{43} \cdots \\ 4 & \qquad\quad 1 \cdots \\ \vdots & \qquad\quad\ \cdots \end{array} \quad \left\{ y \right\} = \left\{ a \right\}$$

$$y_i = a_i - \sum_{j=i+1}^{j=n} L_{ji} * y_j$$

Fig. 9 Forward and back substitution.

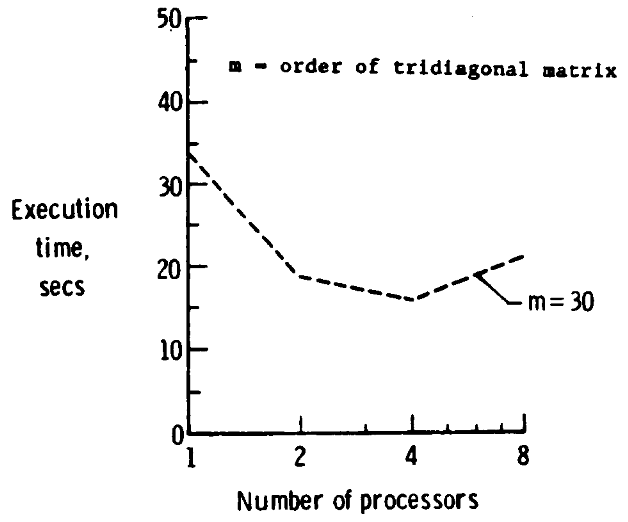


Fig. 11 Execution time for forward solution.

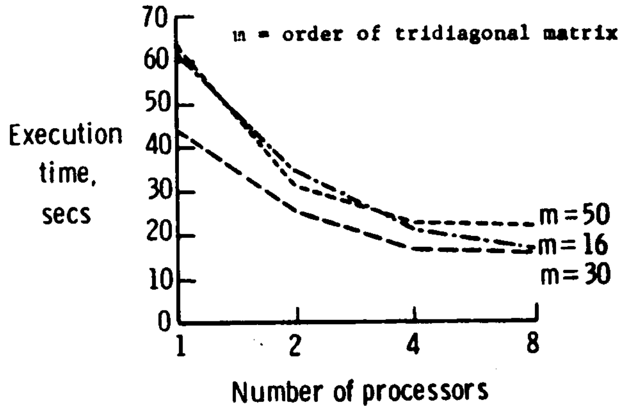


Fig. 10 Execution time for an acceptable eigenvalue.

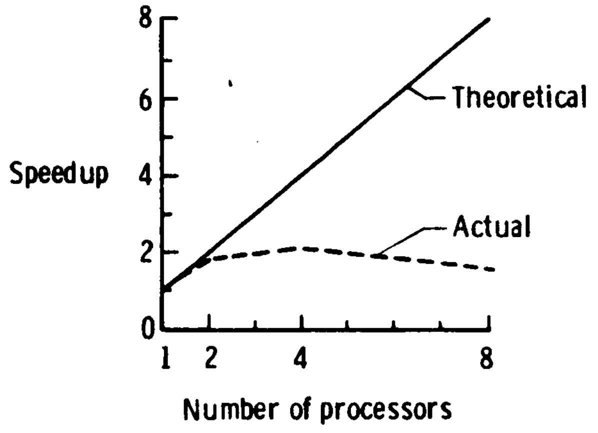


Fig. 12 Speedup for forward solution.

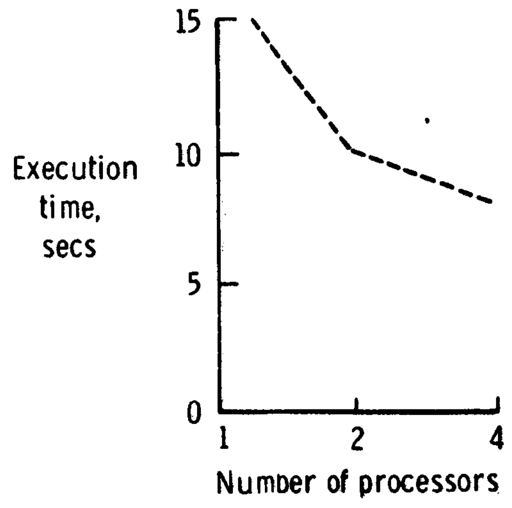


Fig. 13 Execution time for back substitution.

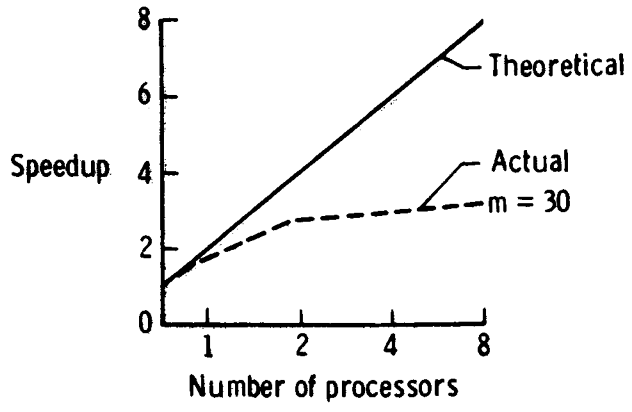


Fig. 15 Speedup for Lanczos method with one shift.

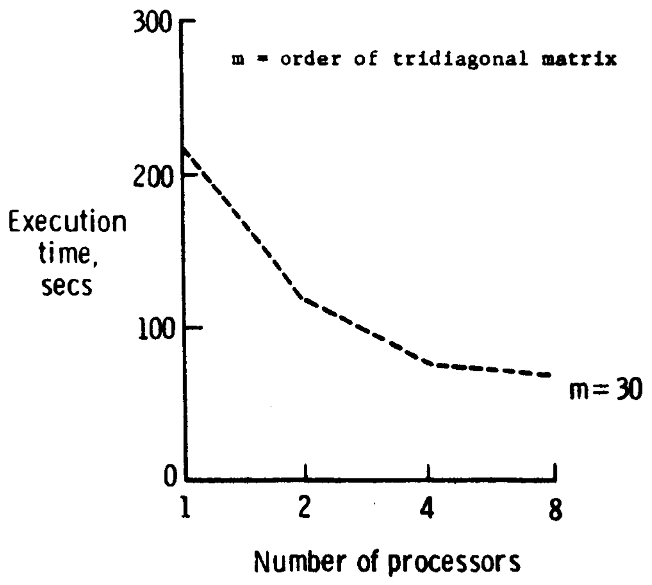


Fig. 14 Execution time for Lanczos method with one shift.

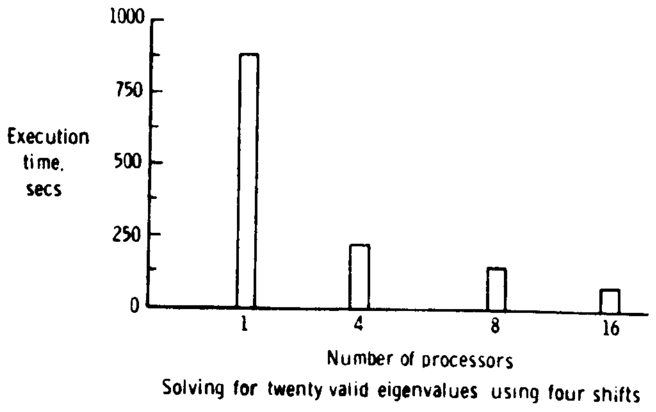


Fig. 16 Timing reductions for Lanczos method with four shifts.

Standard Bibliographic Page

1. Report No. NASA TM-89097	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle A Lanczos Eigenvalue Method on a Parallel Computer		5. Report Date March 1987
		6. Performing Organization Code 505-63-31-02
7. Author(s) Susan W. Bostic and Robert E. Fulton		8. Performing Organization Report No.
9. Performing Organization Name and Address NASA Langley Research Center Hampton, VA 23665-5225		10. Work Unit No.
		11. Contract or Grant No.
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20546		13. Type of Report and Period Covered Technical Memorandum
		14. Sponsoring Agency Code

15. Supplementary Notes

Robert E. Fulton is a Professor of Mechanical Engineering at the Georgia Institute of Technology.

16. Abstract

Eigenvalue analyses of complex structures is a computationally intensive task which can benefit significantly from new and impending parallel computers. This study reports on a parallel computer implementation of the Lanczos method for free vibration analysis. The approach used here subdivides the major Lanczos calculation tasks into subtasks and introduces parallelism down to the subtask levels such as matrix decomposition and forward/backward substitution. The method was implemented on a commercial parallel computer and results were obtained for a long flexible space structure. While parallel computing efficiency for the Lanczos method was good for a moderate number of processors for the test problem. The greatest reduction in time was realized for the decomposition of the stiffness matrix, a calculation which took 70 percent of the time in the sequential program and which took 25 percent of the time on eight processors. For a sample calculation of the twenty lowest frequencies of a 486 degree of freedom problem, the total sequential computing time was reduced by almost a factor of ten using 16 processors.

17. Key Words (Suggested by Authors(s)) Lanczos Parallel computers Vibration analysis Eigenvalue Structures		18. Distribution Statement Unclassified - Unlimited	
19. Security Classif.(of this report) Unclassified	20. Security Classif.(of this page) Unclassified	21. No. of Pages 19	22. Price A02