Three Parallel Computation Methods for Structural Vibration Analysis

Olaf Storaasli* and Susan Bostic†
NASA Langley Research Center, Hampton, Virginia
and
Merrell Patrick,‡ Umesh Mahajan,§ and Shing Ma§
Duke University, Durham, North Carolina

Three parallel algorithms, based on existing sequential methods for vibration analysis, were developed and evaluated for the solution of three example problems on a parallel computer. The three methods—Lanczos, multisectioning, and subspace iteration—maintain reasonable accuracy in the computation of vibration frequencies and show significant reductions in computation time as the number of processors increases. An analysis of the performance of each method was carried out to determine its strengths and weaknesses and to identify the key parameters that affect its computational efficiency.

Introduction

THE determination of the fundamental (lowest) natural frequencies and associated mode shapes is a key step used to uncover and correct potential failures or problem areas in most complex structures. However, the computation time taken by finite-element codes to evaluate these natural frequencies is significant, often the most computationally intensive part of structural analysis calculations. There is a continuing need to reduce this computation time. This study addresses this need by developing methods for parallel computation.

Since Von Neumann (sequential) computers are approaching their limit of computation speed due to physical constraints (speed of electricity), a new generation of parallel scientific computers is being introduced to increase computation power by increasing the number of processors. With such parallel computers, time-consuming calculations of vibration analysis (i.e., matrix factorization) can be performed using many processors operating in parallel. However, methods that perform well on sequential computers do not necessarily perform well on parallel computers. For that reason, the NASA computational structural mechanics (CSM) initiative began the development of new parallel algorithms to speed up structural analysis computations on the emerging class of parallel computers.[1-5] The work described herein extends this previous work.

The objective of this paper is to describe and evaluate the performance of three parallel computation methods developed for the solution of structural vibration problems on a parallel computer. The three methods are the Lanczos method, subspace iteration, and sectioning.[6-15] To evaluate their performance, each method was used to solve representative vibration analysis problems on a parallel computer.

Presented as Paper 88-2391 at the AIAA/ASME/ASCE/AHS/ACS 29th Structures, Structural Dynamics, and Materials Conference, Williamsburg, VA, April 18-20, 1988; received Aug. 4, 1988; revision received Nov. 24, 1988.

*Aerospace Engineer, Computational Mechanics Branch. Associate Fellow AIAA.
†Electronics Engineer, Computational Mechanics Branch. Member AIAA.
‡Professor, Department of Computer Science.
§Graduate Student, Department of Computer Science.

Methods for Vibration Analysis

The solution of structural vibration problems has become more complex as the structures to be analyzed, such as aircraft and flexible space structures, have increased in size and complexity.[6] The usual objective of a vibration analysis is to find a few of the lowest vibrational frequencies and the associated mode shapes of a structure. The approach is to solve the structural eigenvalue problem

$$Kx = \lambda Mx \tag{1}$$

where K and M are symmetric, positive-definite stiffness and mass matrices, respectively, x is the displacement vector (the eigenvector), and λ (the eigenvalue) determines the frequency. In this study, three parallel methods have been developed and tested on representative problems of present and future interest. The method of subspace iteration is included in this study as programs based on it are in widespread use throughout the engineering community for eigenvalue extraction.[6] The Lanczos method is included as an emerging competitor of subspace iteration.[6,15] A method based on sectioning is amenable to parallel solution of the standard eigenvalue problem,[15] so its suitability as a parallel method for the generalized eigenproblem is studied here. Methods based on Jacobi, block Jacobi, and QR transformations are not included since they solve for all eigenvalues. For our applications, only a few of the smallest eigenvalues are required. A brief description of each method follows.

Lanczos Method

The Lanczos method, a technique to solve Eq. 1, involves transforming the original large eigenvalue problem into a small, easier-to-solve problem.[11] The major calculation tasks in the Lanczos method include decomposition of the stiffness matrix and the solution of the system of equations by a forward solution followed by a backward solution. The elements

of the resulting tridiagonal system and the Lanczos vectors are then calculated. The bisection method, a reliable and efficient method for the solution of tridiagonal systems, was then used to solve for the first few (lowest) eigenvalues. The bisection method is easily adapted to parallel computation and computes eigenvalues that accurately approximate the eigenvalues of interest in the original large problem.

The strength of the Lanczos method is that the few eigenvalues of interest can be found with a minimum amount of computation. One of the method's weaknesses is that when the algorithm is implemented on a finite-precision arithmetic computer, the computations gradually decrease in accuracy due to loss of orthogonality in the computed vectors. Many Lanczos procedures have been developed to overcome this loss of orthogonality by including a total or selective reorthogonalization with respect to the previously calculated vectors. Another approach, recommended by Cullum and Willoughby,[9] provides an easy test for selecting valid eigenvalues from those obtained through Lanczos calculations. The results reported in the present study are based on use of the Cullum and Willoughby test because, in comparison with reorthogonalization, the test was found to be faster and as reliable.

Subspace Iteration Method

One of the more frequently used eigensolution techniques is subspace iteration, which is a combination of inverse iteration and the Rayleigh-Ritz procedure. Subspace iteration begins with the somewhat difficult choice of the dimension m of the subspace and of the m starting vectors $(x_1, x_2, \ldots, x_m) = X_O$ with $(X_O)^T M X_O = I$. Given a shift σ, the usual implementation[15] is for $k = 1, 2, \ldots$ until convergence, repeat steps 1–5.

1) Solve $(K - \sigma M) Y_k = M X_{k-1}$ for Y_k.

2) Compute the reduced matrices $K_k = (Y_k)^T K Y_k$, $M_k = (Y_k)^T M Y_k$.

3) Solve the small problem $(K_k - \theta_i M_k)\, g_i = 0$ for eigenvalues θ_i and normalized eigenvectors g_i, $i = 1, 2, \ldots, m$. Set $G_k = (g_1, g_2, \ldots, g_m)$.

4) Set $X_k = Y_k G_k$.

5) Test for convergence.

Steps 2–4 of subspace iteration carry out the Rayleigh-Ritz procedure that delivers the best approximations to eigenvalues of $K - \lambda M$ from the space spanned by Y_k. Within this implementation, $\sigma = 0$ is used in step 1 and unshifted inverse iteration is used in step 3 to solve the small eigenvalue problem. Hence, the computation is broken into phases, the primary ones being a Choleski factorization of K followed by a sequence of Rayleigh-Ritz approximations and inverse iterations for the m eigenvalues and eigenvectors.

Sectioning Method

The sectioning method computes either the m smallest frequencies or all of the frequencies in a specified interval. The frequencies are isolated, in parallel, using sectioning in which the Sturm sequence property of leading principal minors of matrices $K - \sigma M$ and an appropriate search technique are used.[12] Isolated frequencies and their corresponding vibration modes are computed, to the desired accuracy, using combined inverse and Rayleigh-quotient iterations.[13]

The Sturm sequence for the generalized eigenvalue problem can be computed using a modified elimination procedure with a special pivoting scheme.[12] If multiple or clustered eigenvalues exist, the sectioning process will continue until the subintervals containing these eigenvalues are sufficiently small. Eigenvectors corresponding to these eigenvalues are computed using inverse iteration and orthogonalization techniques.[11]

Once a single eigenvalue has been isolated in an interval $(\gamma - \eta, \gamma + \eta)$, a superlinear algorithm[12] is used to compute the eigenpair accurately. The algorithm is a combination of inverse and Rayleigh quotient iterations of the form

$$(K - \sigma M) y_k = M z_k$$

$$\omega_{k+1} = [(y_{k+1})^T M y_{k+1}]^{-1/2}$$

$$z_{k+1} = \omega_{k+1} y_{k+1}$$

where $\sigma = \gamma$ in the case of inverse iteration and $\sigma = \sigma_k = z_k^T K z_k$ for the Rayleigh quotient case. Inverse iteration converges linearly to the eigenvalue closest to γ, whereas Rayleigh quotient iteration exhibits cubic convergence. The choice of whether to use either $\sigma = \gamma$ or $\sigma = \sigma_k$ is based on a criterion that guarantees superlinear convergence of the algorithm to the eigenvalue isolated in the interval.

Parallel Implementations

The three methods were implemented on a parallel computer (FLEX/3 multicomputer[16]) at NASA Langley Research Center. The FLEX/32 consists of 20 processors, each with 4 Mbytes of local memory and 4.5 Mbytes of shared memory (accessed by all processors). In the configuration used in this study, two of the processors are dedicated to program development and run under the UNIX operating system. The other 18 processors run under a parallel operating system. A brief discussion of the parallel implementation of each method follows.

Parallel Lanczos

The parallel Lanczos algorithm used in this paper is an extended and optimized version of the method more fully described in Refs. 2–4. A self-scheduled task assignment algorithm is the basic strategy used to parallelize the major computational tasks of the Lanczos method (i.e., matrix decomposition, forward, and backward solutions). In this strategy, a queue of tasks is set up in shared memory and each processor takes its next work assignment from this queue. The assignment of tasks is done in a row interleaved fashion. Each task represents a row of the matrix, and the next available processor is assigned the calculations necessary for that row. The computed values are stored in shared memory where they are accessible to all processors. The processors must be synchronized to insure that values needed for the next step are already computed and stored. This synchronization accounts for most of the overhead associated with the parallelism. One advantage of self-scheduling is that if one processor should fall behind in its calculations or experience hardware problems, the other processors would continue the calculations.

Parallel Subspace Iteration

The subspace iteration algorithm involves operations on data in the form of matrices and vectors. Operations such as matrix-matrix multiplication, matrix-vector multiplication, and vector-vector (inner product) multiplication have been parallelized. For example, assuming p processors, matrix-vector multiplication A^*x (where A is an $n \times n$ matrix and x is a vector of length n) is performed by assigning rows, i, $i+p$, $i+2p, \ldots$ to processor i and, for all i, having processor i multiply its rows by x. The Choleski factorization and the forward and backward substitution phases are also carried out in parallel using the same row interleaved form of partitioning the work among the processors. In contrast to the self-scheduled workload assignment used in the parallel Lanczos method, a prescheduled workload assignment is used here. In this approach, work is preassigned to processors in a fixed manner rather than processors beginning work on the next available task. All communication between processors is carried out via shared memory.

Parallel Sectioning

The parallel implementation of the sectioning method is based on processors working in parallel to isolate the eigenval-

ues in intervals. Each so-called "slice" of the spectrum requires a triangular factorization of the matrix $K-s_iM$ where s_i denotes the point at which the spectrum is sliced. The processors i, $i=1,2,\ldots,p$, can perform the factorization at slices s_i in parallel. Once the eigenvalues are isolated (i.e., each eigenvalue lies in a different interval), the corresponding eigenvector can be accurately determined by the superlinear method, which combines the inverse and Rayleigh-quotient iterations. The eigenvalues can be computed independently in different intervals by different processors. The intervals containing the isolated eigenvalues are stored in a queue in shared memory. Whenever a processor is available, it computes eigenvalues on the next interval, if any, in the queue. Hence, the parallel tasks of extracting eigenvalues are self-scheduled. This is like having "eigenvalue engines" computing eigenvalues in parallel, with each allotted a particular search interval.

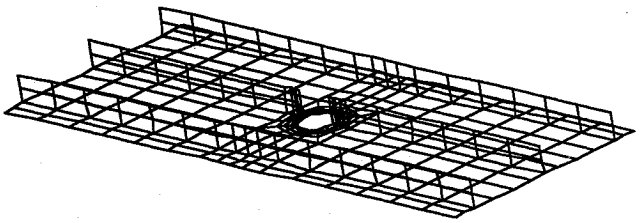

Fig. 1 Blade-stiffened panel example problem.

Structural Analysis Focus Problems

Three example problems, a stiffened panel with a circular cutout (Fig. 1) and two deployable truss beams—space Mast (Fig. 2), and Mini-Mast (Fig. 3)—were studied. To determine the performance of the parallel methods on different problem sizes and matrix characteristics, two models of the stiffened panel problem and the two truss-beam problems were solved. This was done by interfacing the methods with the model definition, matrix generation, and node resequencing utilities of the sequential CSM testbed.[17] The vibration frequencies were obtained for the example problems by the three parallel methods and their accuracy compared with frequencies obtained using the CSM testbed.

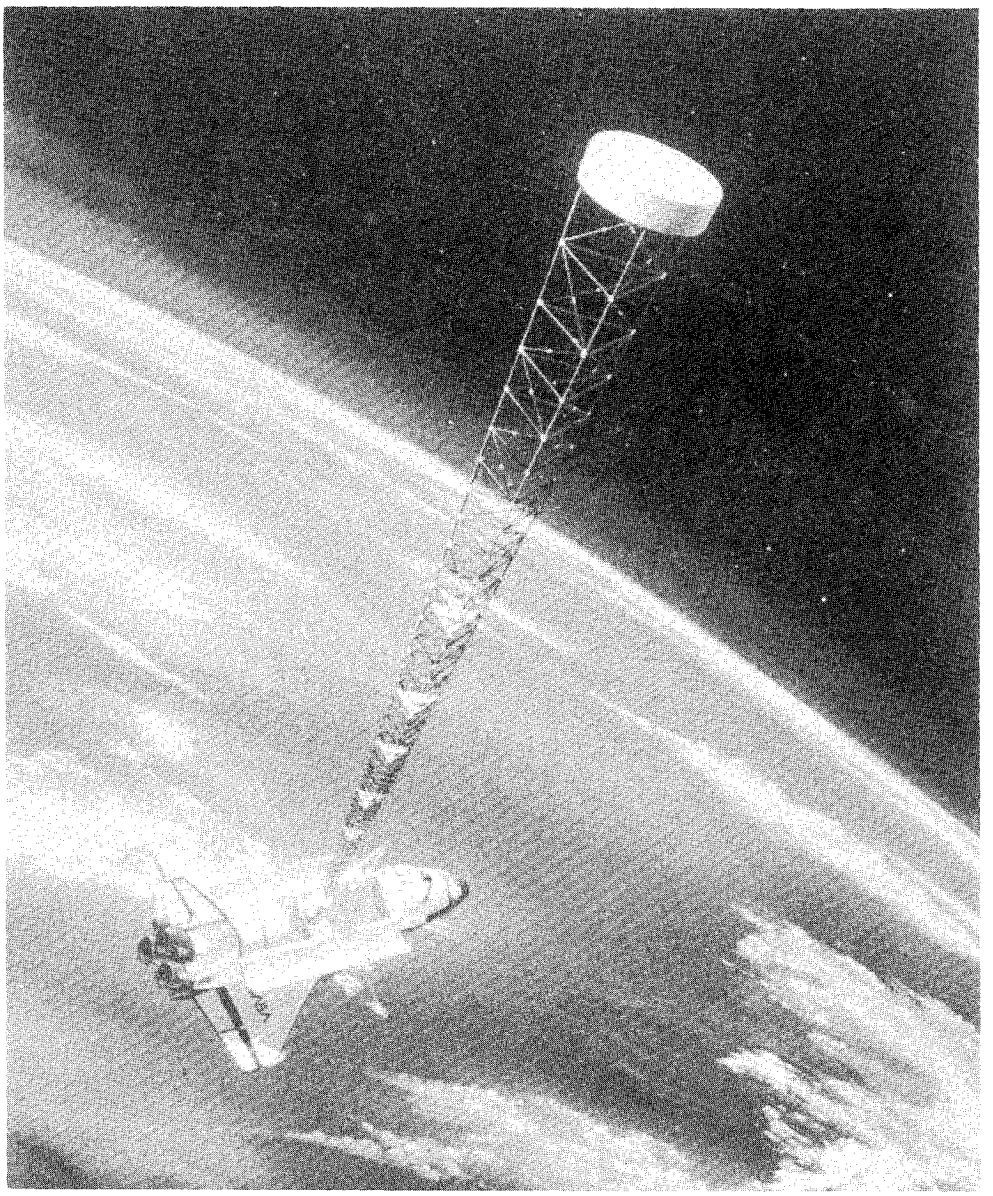

Fig. 2 Deployable space Mast example problem.

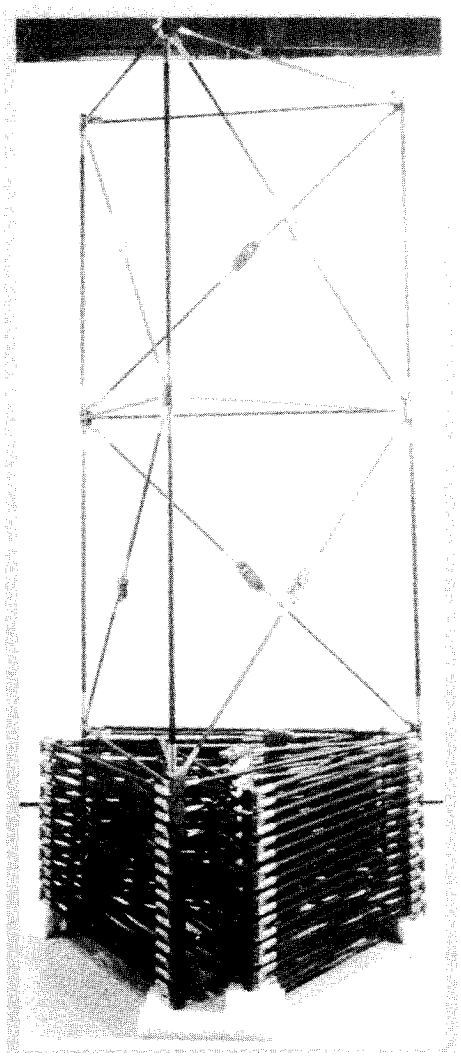

Fig. 3 Deployable Mini-Mast example problem.

Blade-Stiffened Panel

A model of a 76.2×29.2 cm rectangular, blade-stiffened, aluminum panel with a 5.1-cm hole in the center is shown in Fig. 1. The panel contains three 3.56-cm-high stiffeners spaced 11.43 cm apart. The thicknesses of the plate and stiffeners are 0.254 cm. A more detailed desciption of the finite-element model used (including input data) is contained in Appendix C of Ref. 16. Two finite-element models of this stiffened panel were used in this study: a coarse model (648 degrees-of-freedom) and a medium-sized model (2328 degrees-of-freedom). The stiffness matrix for the coarse 108-node model of the panel contains rows with a semibandwidth of 118, whereas that for the 2328-degree-of-freedom model has 1824 rows with a semibandwidth of 240.

Deployable Space Mast

The space Mast problem is based on a proposed Space Shuttle experiment to explore the dynamic characteristics of a 60-m, 54-bay, 3-longeron, deployable truss beam (Fig. 2). Details of the model definition and results are published.[18] In the present study, a 495-degree-of-freedom model of the Mast was analyzed. The stiffness matrix for this model had a semibandwidth of only 18.

Deployable Mini-Mast

A second, more detailed one-third length beam model of the space Mast with mid-point hinges, referred to as the Mini-Mast,[19] was deployed and tested for static and dynamic characteristics at the NASA Langley Research Center. The Mini-Mast (Fig. 3) consists of 18 bays containing thin graphite-epoxy tubular longerons, battens, and diagonal members each with titanium-tip connectors. The mass of the 111 titanium joints (0.7775 kg) is significant when compared to the lightweight tubular members. Thus, the Mini-Mast is referred to as a "joint-dominated structure." The Mini-Mast is fixed at the three base points, which makes only 1980 of the total 1998 degrees-of-freedom active in the solution. Examination of the element interconnection pattern for joints reveals that the Mini-Mast has a small bandwidth (60) when compared to the panel problems (118 and 240).

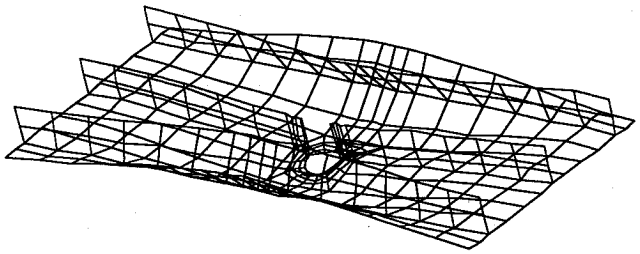

Fig. 4 Second vibration mode for 2328-degree-of-freedom panel.

Computational Results

Computational results were obtained for the example problems by each of the methods. The first 10 vibration frequencies were obtained by all parallel methods and generally agreed to within 3–5 digits with the testbed results. Some test cases, (i.e., the 2328-degree-of-freedom panel) were not run using the sectioning and subspace methods due to computer memory constraints. However, for these methods, double precision was used to calculate more accurately the first 10 frequencies. The double-precision calculations increased the accuracy, but also increased the execution time and memory requirements.

The second vibration mode of the 2328-degree-of-freedom stiffened panel and the first four modes of the 1998-degree-of-freedom Mini-Mast are shown in Figs. 4 and 5. The two lowest vibration modes for the Mini-Mast (far left of Fig. 5) are bending about the base in the x and y directions (to the left and into the paper). The bending mode into the paper is barely recognizable by the gap between the elements forming the upper platforms. The torsion and second bending modes are shown on the right of Fig. 5.

To identify specific characteristics shown by particular methods, the results are discussed individually, by method.

Computational Results for the Lanczos Method

The parallel Lanczos method was used to solve the space Mast problem, the Mini-Mast problem, and two discretizations of the stiffened panel problem.

The execution time to solve the small space Mast problem with an increasing number of processors is given in Table 1 and also shown as the lower curve in Fig. 6. The computation time decreased from 68 s on one processor to 45 s on two processors. The addition of more processors did not significantly reduce the execution time, and, in fact, it took longer for eight processors to solve the problem than for four processors to solve the same problem. This increase in time is attributed to the problem having so few indendent computations to perform. This emphasizes the need to scale the resources to the amount of work to be performed and the need to consider the overhead attributed to subdividing tasks.

The number of arithmetic operations in the decomposition step is dependent on the semibandwidth and the order of the matrix. When the matrix is banded, the calculation of the zero elements is ignored. Therefore, when the assigned row number

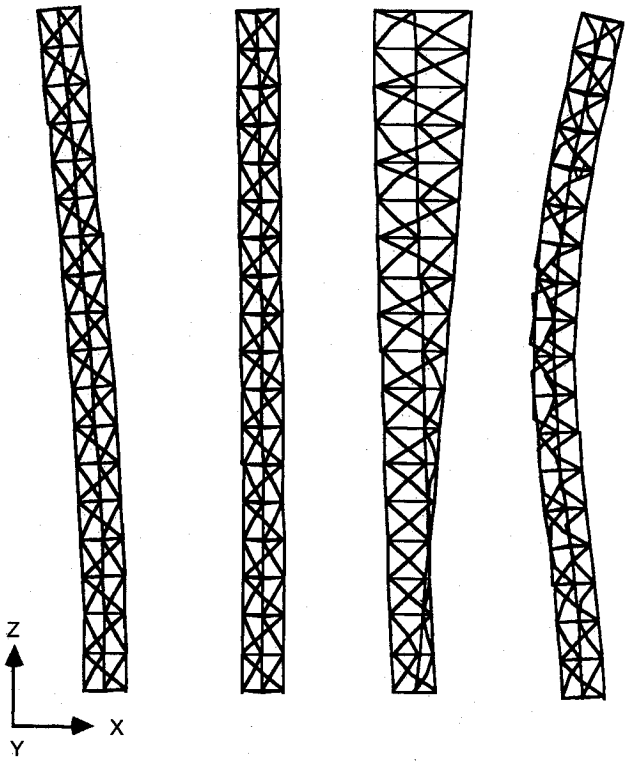

Fig. 5 Vibration modes 1-4 for 1998-degree-of-freedom panel.

Table 1 Performance comparison for 495-degree-of-freedom Mast

Performance	No. of processors	Parallel method		
		Lanczos	Sectioning	Subspace
Computation time, s	1	68.0	1492.1	1138.4
	2	45.0	—	612.7
	4	40.0	480.0	359.2
	8	45.0	336.5	236.3
	16	59.0	199.4	192.8
Computation speedup	1	1.0	1.0	1.0
	2	1.5	—	1.9
	4	1.7	3.1	3.3
	8	1.5	4.4	5.3
	16	1.2	7.5	5.9

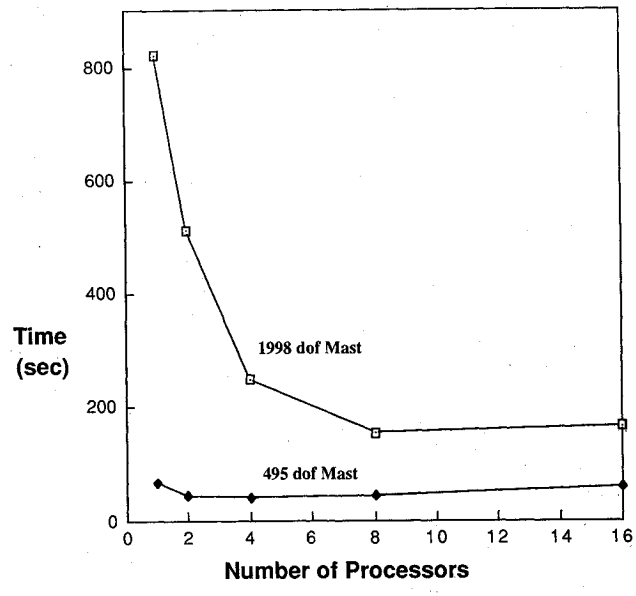

Fig. 6 Lanczos execution time for Mast and Mini-Mast.

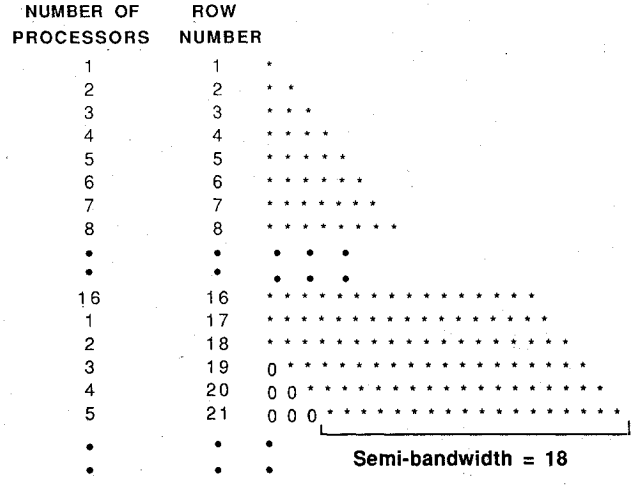

Fig. 7 LDL^T decomposition of Mast stiffness matrix.

is equal to or greater than the semibandwidth, the work becomes equal on each processor. An example of the work assignment of the decomposition of the space Mast stiffness matrix on 16 processors is shown in Fig. 7. From the figure, one sees that row 20 is assigned to processor 4. The calculation for this row will begin with the third element in this row. To compute the elements for row 20, some results from the previous rows are needed. Processor 4 must synchronize with the other 15 processors to insure that the necessary calculations have been completed. The more processors working at one time and having to synchronize with each other, the more parallel overhead is introduced. The small amount of calculations for each processor for this particular problem (18 elements per row) plus the amount of synchronization (with 15 other processors), explains the results shown in Fig. 6. When the Lanczos method was used to solve the larger Mini-Mast problem, the performance proved much more satisfactoy. The computation times for the Mini-Mast are shown as the upper curve in Fig. 6. The execution time for the solution of the Mini-Mast problem with 1998 degrees-of-freedom is shown to decrease significantly for up to eight processors with no improvement for 16 processors. This problem had K and M matrices of dimension 1980 with semibandwidth 60.

The stiffened panel problem was solved using two finite-element discretizations. The execution times for these two solutions are given in Table 2 and shown in Fig. 8. The time to solve the 2328-degree-of-freedom panel problem was reduced from 3189 s on one processor to 246 on 16 processors, giving a speedup of 12.96. The 648-degree-of-freedom panel problem was less computationally demanding and had a speedup of only 6.18 for 16 processors.

Computational Results for Subspace Iteration

The parallel subspace iteration method was used to solve the space Mast problem with 495 degrees-of-freedom and the coarse model panel problem with 648 degrees-of-freedom. In both cases, the first 10 eigenvalues were computed with a maximum error of 10^{-6}.

The impact of subspace size on the execution time for the space Mast is shown in Fig. 8. With more computational work (which can be distributed more evenly), the larger subspace case yields the greatest reduction in execution time as the

number of processors increases. However, the smaller subspace case required less execution time but used the processors less efficiently. The maximum reducton in computation time occurs when the number of processors is a divisor of the subspace size (Fig. 9).

The execution time vs the number of processors for the 648-degree-of-freedom panel problem with a subspace size of 18 is shown in Fig. 10. Just as for the space Mast (Fig. 9), the time reductions are significant as the number of processors increases, with the maximum reductions occurring at divisors of the subspace size.

Computational Results for Parallel Sectioning

The parallel sectioning algorithm was used to solve both Mast problems and the coarse model panel problem. In all cases, 10 eigenvalues were computed with a maximum error of 10^{-6}. Execution times vs the number of processors for the space Mast problem with 495 degrees-of-freedom are shown in Fig. 11. The speedup and execution times for parallel subspace iteration and Lanczos methods are also included to compare all three methods.

The Mini-Mast problem with 1998 degrees-of-freedom and the coarse model panel problem with 648 degrees-of-freedom were run with 18 processors. These problems were not run using fewer numbers of processors because the execution time with 18 processors already indicated that parallel sectioning would not be a competitive method with subspace iteration and Lanczos for larger bandwidth problems. The parallel sectioning method on 18 processors required roughly 5 h of time to solve the Mini-Mast problem with 1998 degrees-of-freedom (matrices of dimension 1980 and semibandwidth 60) and roughly 2-1/3 h to solve the coarse model panel problem with 648 degrees-of-freedom (K and M of dimension 476 with semibandwidth 118).

The parallel subspace iteration method requires less execution time for 18 processors than parallel sectioning for the space Mast problem (Fig. 11) with 495 degrees-of-freedom (K and M of dimension 486 with semibandwidth 18). One can see the impact on computation time of the reduced bandwidth by comparing Fig. 11 to Fig. 10. Even though the parallel sectioning algorithm appears less efficient for larger bandwidth problems, it is considerably more efficient than parallel subspace iteration for tridiagonal matrices (Fig. 12), which makes it a likely candidate for finding the eigenvalues of the tridiagonal matrices arising in the Lanczos method.

Table 2 Lanczos results for stiffened-panel models

Performance	No. of processors	Panel (degrees of freedom) 648	2328
Computation time, s	1	414.0	3189.0
	2	235.0	1607.0
	4	138.0	827.0
	8	92.0	430.0
	16	67.0	246.0
Computation speedup	1	1.0	1.0
	2	1.76	1.98
	4	3.0	3.86
	8	4.50	7.41
	16	6.18	12.96

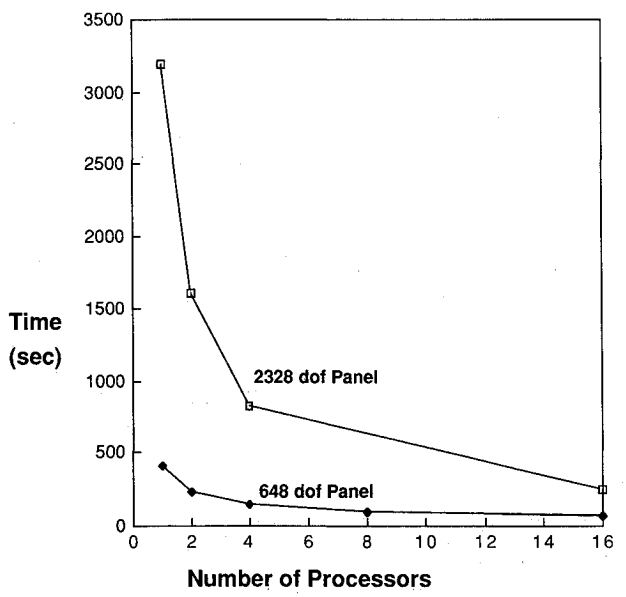

Fig. 8 Lanczos execution time for panels.

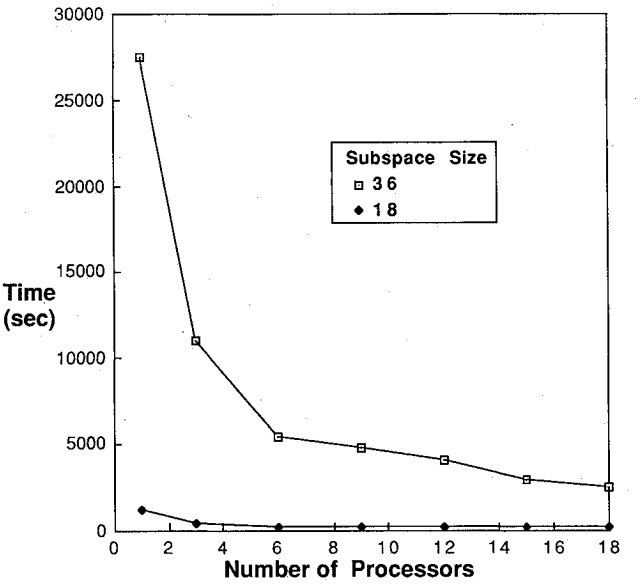

Fig. 9 Subspace execution times for Mast (semibandwidth = 18).

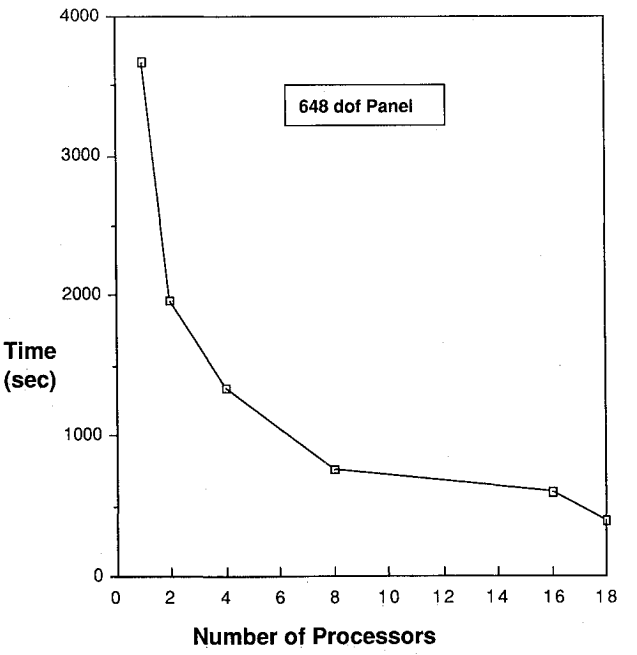

Fig. 10 Subspace execution time for panel (semibandwidth = 118).

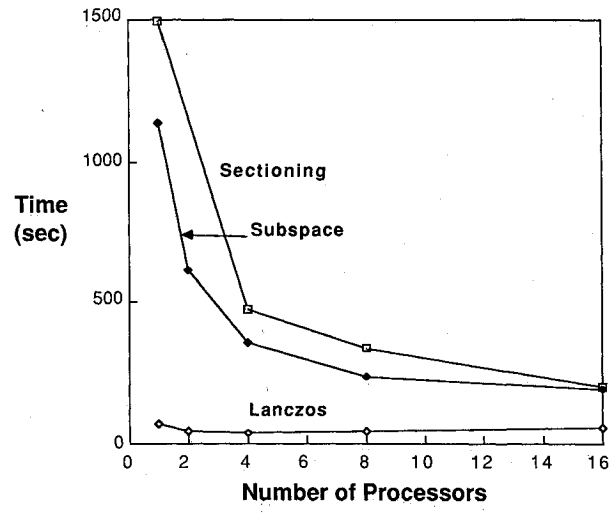

Fig. 11 Execution times for 495-degree-of-freedom Mast.

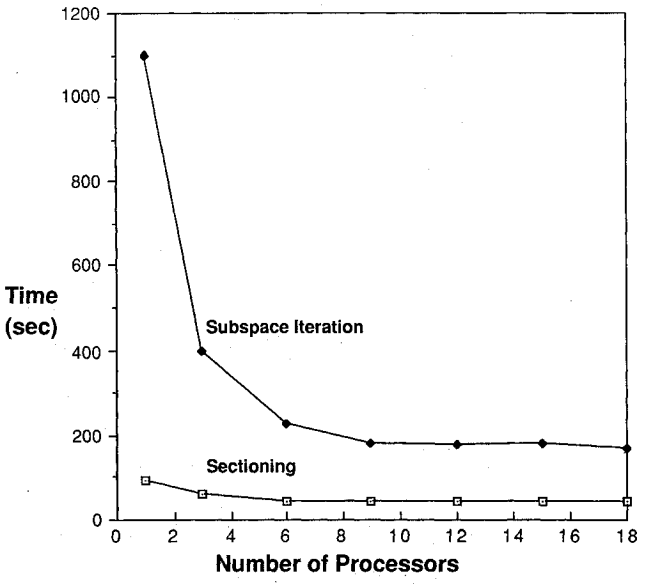

Fig. 12 Subspace and sectioning execution times for tridiagonal matrix.

Conclusions

Three parallel methods for structural vibration analysis—Lanczos, multisectioning and subspace iteration—were developed, and their performance was tested on a parallel computer. All methods demonstrated significant reductions in computation time as the number of processors was increased. The parallel performance was found to vary with the problem size (degrees of freedom and matrix bandwidth). The methods performed best on the panel problems, which had larger matrix bandwidths and sufficient computations to keep all of the processors busy. For example, the Lanczos method achieved a speedup of 12.96 on 16 processors for the largest panel problem.

The Lanczos method took substantially less computation time than the subspace iteration and sectioning methods to solve the example problems tested. The subspace iteration method showed a reduction in computation time even for 18 processors. This may indicate that the subspace method would perform well for very large problems. The sectioning method yields speedups which are restricted to be less than the number of eigenvalues being determined.

References

[1]Knight, N. F., and Stroud, W. J., "Computational Structural Mechanics: A New Activity at the NASA Langley Research Center," NASA TM-87612, Sept. 1985.

[2]Bostic, S. W., and Fulton, R. E., "A Concurrent Processing Implementation for Structural Vibration Analysis," *Proceedings of the 26th AIAA/ASME/ASCE/AHS Structures, Structural Dynamics and Materials Conference*, AIAA, New York, 1985, pp. 566–572.

[3]Bostic, S. W., and Fulton, R. E., "Implementation of the Lanczos Method for Structural Vibration Analysis on a Parallel Computer," *Computers & Structures*, Vol. 25, No. 3, 1987, p. 395.

[4]Bostic, S. W., and Fulton, R. E., "A Lanczos Eigenvalue Method on a Parallel Computer," *Proceedings of the 28th AIAA/ASME/ASCE/AHS Structures, Structural Dynamics and Materials Conference*, AIAA, New York, 1987; also, AIAA Paper 87-0725, 1987.

[5]Storaasli, O. O., Poole, E. L., Ortega, J. M., Cleary, A. J., and Vaughan, C. T., "Solution of Structural Analysis Problems on a Parallel Compuer," *Proceedings of the 29th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, AIAA, Washington, DC, 1988, pp. 596–605; also, AIAA Paper 88-2287, 1988.

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

[7]Bathe, K., *Finite Element Procedures in Engineering Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1982.

[8]Lanczos, C., "An Iteration Method for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operators," *Journal of Research of the National Bureau of Standards*, Vol. 45, 1950, pp. 255–282.

[9]Cullum, J., and Willoughby, R. A., *Lanczos Algorithms for Large Symmetric Eigenvalue Computations*, Vol. 1—Theory, Birkhauser, Boston, 1985.

[10]Parlett, B. N., *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, NJ, 1980.

[11]Golub, E. H., and Van Loan, C., *Matrix Computations*, Johns Hopkins Univ. Press, Baltimore, MD, 1983.

[12]Peters, G., and Wilkinson, J. H., "Eigenvalues of $Ax=\lambda Bx$ with Band Symmetric A and B," *Computing Journal*, Vol. 12, 1969, pp 398–404.

[13]Szyld, D. B., "A Two-level Iterative Method for Large Sparse Generalized Calculations," Ph.D. Thesis, Dept. of Mathematics, New York Univ., New York, 1983.

[14]Lo, S., Philippe, B., and Sameh, A., "A Multiprocessor Algorithm for the Symmetric Tridiagonal Problem," *SIAM Journal of Scientific and Statistical Computing*, Vol. 8, 1987, pp. 155–166.

[15]Parlett, B. N., "How To Solve $(K-\lambda M)$ For Large K and M," Mathematics Department and Computer Science Division Rept., Univ. of California, Berkeley, CA, 1984.

[16]Matelan, N., "The Flex/32 Multicomputing Environment," *Research in Structures and Dynamics 1984*, compiled by R. J. Hayduk and A. K. Noor, NASA CP-2335, 1984.

[17]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 89096, 1987.

[18]Horta, L. G., Walsh, J. L., Horner, G. C., and Bailey, J. P., "Analysis and Simulation of the MAST (COFS-1 Flight Hardware)," NASA CP 2447, Part 1, Nov. 18–21, 1986, pp. 515–532.

[19]Adams, L. R., "Design, Development and Fabrication of a Deployable/Retractable Truss Beam Model for Large Space Structures Application," NASA CR 178287, June 1987.