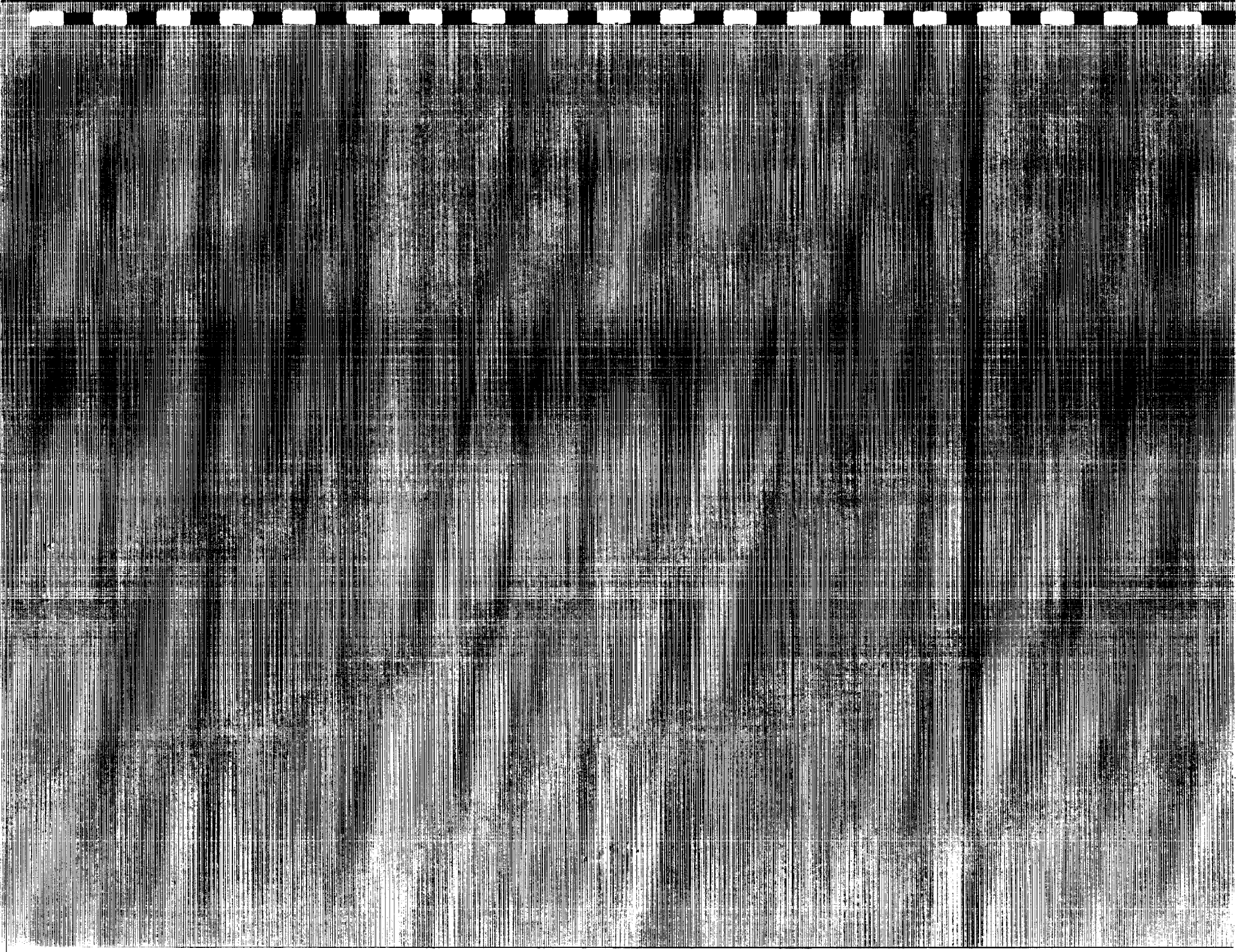
NASA-CR-194567

(NASA-CR-194567) PARALLEL PROCESSING FOR NONLINEAR DYNAMICS SIMULATIONS OF STRUCTURES INCLUDING ROTATING BLADED-DISK ASSEMBLIES Final Technical Report (Cornell Univ.) 153 p

N94-34683

Unclas

G3/62 0011759



PARALLEL PROCESSING FOR NONLINEAR DYNAMICS SIMULATIONS OF STRUCTURES INCLUDING ROTATING BLADED-DISK ASSEMBLIES

by

Shang-Hsien Hsieh

Report to NASA Lewis Research Center

Grant Number NAG 3-1063

Structural Engineering Report 93-2

School of Civil and Environmental Engineering and

Program of Computer Graphics

Cornell University

Ithaca, New York 14853-3501

Research supervised by John F. Abel

May 1993

ABSTRACT

The principal objective of this research is to develop, test, and implement coarse-grained, parallel-processing strategies for nonlinear dynamic simulations of practical structural problems. There are contributions to four main areas: (a) finite element modelling and analysis of rotational dynamics, (b) numerical algorithms for parallel nonlinear solutions, (c) automatic partitioning techniques to effect load-balancing among processors, and (d) an integrated parallel analysis system.

Two finite element approaches are implemented to account for rotational nonlinearities involved in dynamic analysis of rotating bladed-disk assemblies: the Consistent Mass (CM) and Lumped Mass (LM) approaches. It has been found that the analysis results obtained using the CM and LM approaches are in close agreement. In addition, for transient analysis using explicit time integration, the LM approach has been found to be significantly more efficient than the CM approach.

For explicit and implicit dynamic analyses, the present work implements a parallel central difference method and a parallel Newmark method, respectively. A parallel static solver is also implemented for steady-state solutions of rotational dynamics. For the parallel central difference method applied to specific test problems run on 6 processors, parallel efficiencies of over 90% have been achieved. For the parallel Newmark method and the parallel static solver, parallel efficiencies of more than 80% have been obtained.

Two automatic spectral partitioning algorithms are developed to effect load-balancing among processors. They are compared with several automatic partitioning algorithms by previous researchers. It has been found that the proposed RST partitioning algorithm with the communication graph approach gives the best results in most examples studied. In addition, interactive graphics tools are developed to allow for manual partitioning and for examining and modifying results of automatic partitioning.

A parallel analysis system is integrated to help evaluate the parallel strategies investigated, verify the finite element approaches employed, and demonstrate how advanced computer technologies can assist engineers in parallel dynamics simulations. This system takes advantage of the advanced computing environments, data structures, and interactive computer graphics to provide a useful research software testbed for study of nonlinear structural dynamics.

ACKNOWLEDGMENTS

This report is essentially a reproduction of the thesis submitted by the author in partial fulfillment for the Degree of Doctor of Philosophy. The author would like to express sincere gratitude to Professor John F. Abel, the faculty investigator for the project, for his continuous support, guidance, advice, and encouragement throughout the course of this research, and for his patience in reading and improving this report. The author is indebted to Professors Gregory G. Deierlein and Subrata Mukherjee for their patience in reviewing this report. Thanks also go to Professor Donald P. Greenberg for providing the excellent research environment at the Program of Computer Graphics which has made much of this research possible.

The author would like to express special thanks to Dr. Charles Lawrence of NASA Lewis Research Center for his advice, help, and support in the course of this research. Thanks also go to Dr. Horst D. Simon of NASA Ames Research Center who kindly provided the source code of the RSB algorithm and several valuable references, and Professor Charbel Farhat of University of Colorado at Boulder for providing corrections to the published version of the FP algorithm.

The research reported in this report benefits tremendously from the previous work of other researchers at the Program of Computer Graphics, especially that of Dr. Jerome Hajjar on parallel nonlinear solutions of structural dynamics, Dr. Sanjeev Srivastav and Dr. Brian Aubert on development of BASYS and ABREAST, and Dr. Paul Wawrzynek, Dr. Luiz Martha, and Dave Potyondy on development of FRANSYS. The author is

also deeply indebted to Glaucio Paulino for his invaluable discussions and ideas about automatic domain partitioning algorithms. The helpful advice and assistance from Sanjeev Srivastav, Brian Aubert, Paul Wawrzynek, Dave Potyondy, Gyula Greschik, Victor Balapoulos, Christopher White, DaHong Tao, Jeong-Jae Lee, Daniel Kartch, Hurf Sheldon, and Mitch Collinsworth are gratefully acknowledged.

The support of the NASA Lewis Research Center under Grant No. NAG 3-1063 has been essential to the research reported in this report and is highly appreciated. The partial support from the U.S. Air Force Office of Scientific Research under Contract No. AFOSR-90-0211 is also appreciated. The generous equipment grants from both the Digital Equipment Corporation and Hewlett-Packard to the Program of Computer Graphics are gratefully acknowledged. However, any opinions expressed herein are those of the author and do not necessarily reflect the views of the sponsors or equipment donors.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

In recent years the rapid development of novel computer hardware environments with parallel processing capabilities has created new opportunities for revolutionizing engineering computing. In computational structural dynamics, the utilization of these powerful systems may lead to increasingly complete and sophisticated simulations of advanced structural systems. Through such simulations, better engineering understanding, analysis, and design can then be achieved. The realization of advancing engineering simulations in the novel computing environments, however, often requires the difficult tasks of designing new computer codes or adapting old ones. In addition, development of new solution strategies is often required to take full advantage of these new computers. This thesis focuses on one aspect of advances in computational structural dynamics -- the development, testing, and evaluation of coarse-grained, parallel-processing strategies for nonlinear dynamics simulations of large-scale practical structural problems. The work also presents an integration of the latest advanced computing environments, data structures, and interactive computer graphics to facilitate more efficient and powerful simulations of nonlinear structural dynamics.

Computing environments with parallel processing capabilities include shared-memory supercomputers (e.g., Cray Y-MP and IBM 3090), shared-memory intermediate-sized or personal supercomputers (e.g., Convex and Alliant), distributed-memory parallel or multiprocessor computers (e.g.,

Intel iPSC and Connection Machine), and a network of high performance workstations (e.g., Apollo/HP 9000 Series 700, IBM RS/6000, and DECsystem 5000). The supercomputers are very powerful machines; however, they are expensive and are not easily accessible in the same sense as traditional batch-oriented mainframes. On the other hand, engineering workstations which are not as powerful as supercomputers are more affordable and cost-effective. With the growing availability of workstations in the workplace, it is increasingly feasible for engineers to utilize networks of workstations to achieve improved turnaround through parallel computations for computationally intensive simulations such as nonlinear dynamic analysis (Hajjar and Abel 1988, 1989a). This coarse-grained, distributed-memory networked workstation environment is the principal parallel processing environment considered in this thesis.

1.1 Background

The work reported in this thesis is a continuation of the research conducted by Dr. Jerome F. Hajjar (1987, 1988) at Cornell University. However, due to the rapid advance of computer technologies, there have been several new developments in computational structural dynamics since the time Hajjar finished his work in 1988. The exponential growth of computer speed in the last few years has allowed engineers to conduct increasingly complete dynamic analyses on more complicated engineering problems. Recent progress in the area of high speed communication networks (for example, FDDI as opposed to Ethernet) together with growing body of software tools to support distributing computing, such as PICL (Geist et al. 1991), Linda (Leler 1990), and ISIS (Birman et al. 1991), have made network-based distributing systems a viable environment for

engineering computations. Advanced modelling and analysis tools based on new software architecture, data structures, and computer graphics have also been developed to aid engineers in all phases of dynamic simulations, i.e., the data preparation, analysis monitoring, and result visualization. The present work takes advantage of these new developments and further addresses the parallel-processing strategies for nonlinear dynamics simulations.

The major contribution of Hajjar's research to the present work is the investigation and development of numerical time integration algorithms for parallel processing. After re-evaluating these algorithms, this research investigates and implements two parallel algorithms developed by Hajjar for solutions of structural dynamics: the central difference algorithm (Hajjar and Abel 1989a) and the domain decomposition algorithm (Hajjar and Abel 1988).

Using a prototypical environment consisting of one to four VAXstation II's under the VAXELN operating system, Hajjar has shown the feasibility of utilizing networked engineering workstations as a parallel computer to achieve improved turnaround for computationally intensive simulations of structural dynamics. However, the performance of the prototypical configuration was limited by the low computational speed of the VAXstation II (less than 1 Mips) and the lack of generality and portability of the VAXELN operating system. With the advance of computer technology of individual workstations, the present work implements and evaluates parallel algorithms in a more generally applicable and powerful computing environment.

The computer network interface used in this research is Ethernet, which is the same one used by Hajjar. However, faster networks such as FDDI (Fiber Distributed Data Interface) have recently become available. The approximate communication speed of the FDDI is 100 Mbits per second, which is one order of magnitude faster than Ethernet. Cornell's Program of Computer Graphics (where the writer is carrying out this research) is also in the process of installing the FDDI for its parallel processing environment at the time of this writing (March 1993).

To take advantage of parallel processing in the finite element analysis of structural dynamics, the finite element domain is usually partitioned (or decomposed) into a number of subdomains which are distributed among the processors and solved concurrently. The key problem of this approach is how to partition the domain to achieve well-balanced workload distribution among processors and to minimize the amount of interprocess communication so that significant speed-up can be obtained in the parallel analysis. Hajjar has addressed this problem by developing a set of tools with interactive computer graphics to help manual partitioning of the structural domain. However, even with the aid of interactive computer graphics, manual partitioning may be difficult for large finite element meshes with arbitrary geometries. The present work investigates and improves techniques of automatic domain partitioning to effect load-balancing among processors. Graphics tools are also provided for manual partitioning and for examining and modifying results of automatic partitioning.

The application problems of structural dynamics studied by Hajjar focus on the analysis of three-dimensional steel frames subject to

earthquake loading. Beam-column elements, which are 1-D elements, were used for modelling frames. Both geometric and material nonlinearities were considered. In addition to beam-column elements, the present work uses a 2-D shell element to model floor flexibility in the dynamic analysis of steel frames. However, only geometric nonlinearity is considered in the shell elements. A new emphasis in the present work has been on application problems dealing with finite element analysis of rotating turbine bladed-disk assemblies experiencing tip rubs. Solid elements (3-D elements) are used to represent the disks and blades. Both rotational and geometric nonlinearities are included in the finite element formulation to derive the governing equations of motion.

1.2 Objectives

The principal objective of this research is to develop, test, and implement coarse-grained, parallel-processing strategies for nonlinear dynamics simulations of practical structural problems. The parallel-processing strategies addressed include (a) numerical algorithms for parallel nonlinear solutions and (b) techniques to effect load-balancing among processors.

The second objective of the research is the application of finite element techniques for rotational dynamics. Emphasis is on the structural dynamics of rotating turbine bladed-disk assemblies.

The use of advanced computing environments, data structures, and interactive computer graphics for a more efficient and powerful simulation of nonlinear structural dynamics is the third objective of this research.

1.3 Scope

The scope of the research reported in this thesis may be summarized by the three main tasks involved, each paralleling one of the principal objectives. The first and primary task addresses parallel-processing strategies for finite element analysis of structural dynamics. The second one focuses on finite element approaches for modelling and analyzing problems of rotational as well as non-rotational dynamics although emphasis is on the rotational dynamics. The third includes the development of an integrated parallel analysis system. A more detailed discussion of these tasks is provided in the following sub-sections.

1.3.1 Parallel Processing Strategies

The parallel-processing strategies addressed include numerical algorithms for parallel nonlinear solutions of structural dynamics and techniques to effect load-balancing among processors. Although these strategies are suitable for a variety of machine environments sharing a few common features, the major configuration investigated is a coarse-grained, distributed-memory system in a message-passing environment, in particular, networked engineering workstations, each with large memory and one or more powerful processor.

For transient dynamic analyses, the numerical algorithms investigated include both parallel explicit and implicit time integration algorithms. As mentioned earlier, the present work investigates and implements the parallel central difference and parallel domain decomposition algorithms proposed by Hajjar and Abel (1989a, 1988) for explicit and implicit dynamic analyses. For steady-state stress analyses of

rotating systems, a parallel static solution method is implemented using the domain decomposition approach.

In this work, these parallel algorithms are implemented in a program called ABREAST (Srivastav 1991; Aubert 1992), which is a batch finite element analysis program for nonlinear structural dynamics. The verification and evaluation of these algorithms are studied using a variety of example problems.

To effect load-balancing among processors, the present research focuses on the automatic domain partitioning techniques for parallel finite element analysis of structural dynamics. The automatic partitioning algorithms proposed by Farhat (1988), Malone (1988), Al-Nasra and Nguyen (1991), and Simon (1991) are studied. Modified recursive spectral partitioning algorithms are then proposed. In addition, interactive graphics tools are developed to allow for manual partitioning and for examining and modifying results of automatic partitioning.

These partitioning algorithms as well as graphics tools are implemented in a program called PSAINT (Hsieh and Srivastav 1992), an interactive program that performs domain partitioning on finite element meshes. Comparative studies are conducted to evaluate and compare both efficiency and effectiveness of these partitioning algorithms.

1.3.2 Finite Element Modelling and Analysis

The finite element method is employed to study structural dynamics problems for the development, testing, and evaluation of parallel-processing strategies addressed in this work. Two classes of structural dynamics

problems are investigated. The first one includes framed structures with flexible floors subjected to seismic loading, and the second includes rotating turbine bladed-disk assemblies experiencing tip rubs.

For modelling and analysis of framed structures with flexible floors, the beam-column and shell elements already existing in the finite element library of ABREAST are employed. Geometric nonlinear analysis is conducted.

For modelling and analysis of rotating bladed-disk assemblies, a solid element is implemented in ABREAST. Both rotational and geometric nonlinearities are considered in the finite element formulation of equations of motion. Two finite element approaches for handling rotational nonlinearities are also implemented in ABREAST and compared through numerical studies.

Both modal vibration and transient dynamic analyses of rotating bladed-disk systems are investigated and discussed. In the modal vibration analysis, a steady-state stress analysis is followed by an eigensolution. A static solution capability, which was previously not provided by ABREAST, is implemented in ABREAST for the steady-state analysis. In addition, verification studies are conducted to evaluate the finite element rotational dynamics and formulations implemented in ABREAST.

1.3.3 An Integrated Parallel Analysis System

An integrated parallel analysis system is developed to help (a) evaluate the parallel strategies investigated, (b) verify the finite element approaches employed, and (c) demonstrate how advanced computer

technologies can assist engineers in parallel dynamic simulations. The system integrates four computer programs: BASYS (Srivastav and Abel 1990; Srivastav 1991) and FRANSYS (Wawrzynek et al. 1988; Martha 1989; Wawrzynek 1991) for three-dimensional modelling and visualization; PSAINT for finite element domain partitioning; and ABREAST for nonlinear dynamic solutions.

BASYS is primarily designed for modelling and visualization of buildings and other framed structures. FRANSYS was originally developed to model general, 3-D fracture processes in arbitrarily shaped solids. It has been extended to provide general tools for modelling and simulation of complex 3-D solid models. Both BASYS and FRANSYS provide the analyst an efficient way of modifying and manipulating the structural data through the use of a radial edge data representation (Weiler 1986, 1988) and a hierarchical modelling scheme. They also provide a convenient means of displaying the structure model and visualizing the response of the structure using interactive computer graphics. In the present work, graphics tools for visualization of dynamics simulations are implemented in FRANSYS.

PSAINT serves as an interface between BASYS/FRANSYS and ABREAST. Its primary job is to partition finite element domains for parallel analysis. Both automatic and manual partitioning tools are provided. It also collects the results of parallel analysis for simulation playback in BASYS/FRANSYS.

ABREAST was originally developed for analyzing framed structures consisting of either truss or beam-column elements and has been extended to include a nine-node Lagrangian shell element for modelling floors, walls,

or panels. It is capable of both geometric and material nonlinear transient dynamic analyses. In the present research, a twenty-node brick solid element is implemented in ABREAST for modelling rotating bladed-disk systems. Parallel explicit and implicit time integration methods as well as parallel steady-state (static) solution methods are also implemented using a multiple-instruction, multiple-data (MIMD) algorithm.

1.4 Organization

The organization of this thesis is briefly described in this section.

Chapter 2 addresses the finite element approach used for modelling and analysis of framed structures and rotating bladed-disk assemblies. The selection of finite elements for modelling purposes is discussed. Equations of motion for both rotational and non-rotational dynamics are formulated. Two approaches are presented to account for rotational nonlinearities in rotating bladed-disk systems. Numerical comparisons between these two approaches are also conducted. Finite element analyses of both framed structures and rotating bladed-disk assemblies are discussed. In addition, verification studies are reported on both finite elements and analysis algorithms implemented in ABREAST for this work.

Chapters 3 and 4 investigate parallel processing strategies which include parallel nonlinear solution algorithms for structural dynamics and domain partitioning techniques for load-balancing among processors. In Chapter 3, the parallel computing environment used in the present work is described first. Then, parallel solution algorithms for transient dynamics analysis as well as steady-state analysis are evaluated. The algorithms selected and implemented in this research are discussed and their

effectiveness is studied using numerical examples. In Chapter 4, load-balancing techniques based on domain partitioning are reviewed and investigated. Algorithms both proposed by previous researchers and developed in this work are studied and compared. The effectiveness of these algorithms for parallel finite element dynamic analysis is also discussed.

A parallel analysis system integrated in this research is presented in Chapter 5. An overview of the system is first described. The implementation and application of the system for each phase of parallel simulation are discussed. In addition, application examples that examine and demonstrate the efficiency and flexibility of the parallel analysis system and the parallel processing strategies developed in this research are included.

Chapter 6 summarizes and concludes the work reported in this thesis. Suggestions for future work are also provided.

Chapter 2

Finite Element Modelling and Analysis

Two classes of structural dynamics problems are studied for the development, testing, and evaluation of parallel-processing strategies addressed in this thesis. The first one includes framed structures with flexible floors subjected to seismic loading, while the second one includes rotating turbine bladed-disk assemblies experiencing tip rubs. This chapter discusses the finite element approach employed in the present work to model complex structural geometries and material properties, to account for various nonlinearities, and to formulate the governing equations of motion for these problems.

In this work, the finite element analysis capabilities of ABREAST, which has been briefly discussed in Section 1.3, are extended to model and analyze rotating turbine bladed-disk assemblies as well as other rotating or nonrotating solid structures. The implementation and verification of these new capabilities are discussed in this section. Another enhancement of the analysis capability of ABREAST is the implementation of parallel analysis algorithms, which is discussed in the next chapter.

2.1 Modelling of Framed Structures with Flexible Floors

The present work uses the pre-existing elements in the finite element library of ABREAST to model framed structures with flexible floors. To model beams and columns of steel frames, the beam-column element in ABREAST is used. To model flexible floors in steel frames, the nine-noded

Lagrangian shell finite element with 2x2 Gauss integration in ABREAST is employed.

The beam-column element is a line element with twelve degrees-of-freedom, consisting of three translations and three rotations at each end of the element. Bernoulli-Euler beam theory (Ugural and Fenster 1981) is employed in the formulation of the element stiffness with common assumptions, such as homogeneous and isotropic material, plane sections remain plane, doubly symmetric prismatic sections with no cross section distortion, and small strain theory. In linear elastic analyses, the element stiffness is a well known one (see, for example, McGuire and Gallagher 1979). For second order elastic analyses, geometric nonlinearities are handled through the use of an updated Lagrangian formulation and a geometric stiffness matrix (Argyris et al. 1979). For inelastic analyses, material nonlinearities are included through the use of a concentrated plasticity model based on the bounding surface approach and the plastic hardening reduction matrix derived by Hilmy (1984; Hilmy and Abel 1985).

The nine-noded Lagrangian shell finite element was originally developed for full nonlinear static analysis by White and Abel (1990) and further developed for geometric nonlinear dynamic analysis by Srivastav (1991). A projection operator is used to stabilize spurious zero energy modes associated with reduced integration. In elastic analyses, a two-point Gauss integration is performed through the thickness of the element for stiffness computation. For second order elastic analyses, geometric nonlinearities are handled through the use of an updated Lagrangian formulation and the geometric stiffness matrix derived by White (1988). For dynamic analyses, the diagonal element mass matrix formulated by Srivastav (1991) is used.

The verification of the beam-column element for dynamic analysis has been conducted by Hilmy (1984). The use of the nine-noded shell element for dynamic analysis has been verified by Srivastav (1991). Therefore, no further verification is conducted in this work for these two elements.

2.2 Modelling of Bladed-disk Assemblies

To achieve high performance and efficiency, advanced turbine blades have been designed to have complex geometry: thin, low aspect ratio, cambered, twisted, and swept. Some of these geometric parameters have been shown to have significant influence on the dynamic characteristics of blades (for example, Petricone and Sisto 1971; Sreenivasamurthy and Ramamurti 1981). Therefore, to obtain satisfactory results in the dynamic analysis of turbine blades, it is important to model complex blade geometry as accurately as possible.

Because of the ability to model complex structural geometry and properties along with the advancement in computer technology, the finite element method has been recognized as a promising and powerful technique for the analysis of turbine bladed-disk assemblies as well as other configurations. Therefore, it will be employed in this research to model the turbine bladed-disk assemblies.

2.2.1 Review of Previous Research

Considerable research on finite element modelling of turbine bladed-disk assemblies has been conducted. A brief review on some of these research is given below. The turbine bladed-disk assemblies considered in

this research consist of two major components: blades and disk. Different modelling strategies for these components in some of the previous research are reviewed.

2.2.1.1 Modelling of Blades

Turbine blades have been modelled using beam, plate, shell, and solid finite elements. In this section, modelling of blades using these finite elements are briefly reviewed. For detailed description and formulation of these elements, standard textbooks (for example, Zienkiewicz and Taylor 1989; Bathe 1982; Cook et al. 1989) should be consulted.

Beam Finite Elements

Developments of beam finite elements for pretwisted blades have been reviewed by Sisto and Chang (1984). They also developed a pretwisted beam element for use in vibration analysis. Recently, Abbas et al. (1987) developed a thick, tapered, pretwisted beam element to study blade vibration with root flexibility effect.

For blade modelling, the use of beam elements is simpler and requires fewer degrees of freedom than elements of other types. However, beam elements are not suitable for modelling blades with low aspect ratio and a wide range of configurations.

Plate Finite Elements

Some applications of plate elements to rotating blade problems have been reported in the literature. For example, Dokainish and Rawtani (1971) used a triangular plate element with both in-plane and bending stiffness to

determine the natural frequencies and the mode shapes of a rotating cantilever plate. This plate element is a superposition of a plane-stress element with linear displacement order and a plate-bending element with an eight term, cubic polynomial displacement order. In addition, MacBain (1975) used a quadrilateral plate bending element for vibration analysis of twisted cantilever plates.

The aforementioned plate elements are simple to formulate, easy to use, and require only a simple geometric description. However, several shortcomings are present in the application of the plate element to model blades with shell-like complex geometry (Gallagher 1976):

a) The element is unable to model the curved shell surfaces. This may introduce spurious "discontinuity" bending moments at the element juncture lines, and thus many elements may be needed to adequately model shell surfaces.
b) The coupling of membrane and bending within the elements is not included. This is departure from the true behavior.
c) When all elements adjacent to a node are coplanar, special treatment is needed to avoid a singular global stiffness matrix.

Shell Finite Elements

Some previous studies of rotating structures using shell finite elements have been reviewed briefly by Sreenivasamurthy and Ramamurti (1981). A three-noded triangular shell element has also been used by them and later by Omprakash and Ramamurti (1989, 1990a, 1990b) for blade modelling. Recently, a ten-noded triangular shell element has been used to

model cambered and twisted fan blades in vibration analysis (Khader and Abu-Farsakh 1990).

Employing standard tests proposed by MacNeal and Harder (1985), McGee (1987) has compared the overall performance of all shell elements in NASTRAN. Among other shell elements in NASTRAN, the three node triangular (TRIA3) and the four node quadrilateral (QUAD4) shell elements have then been recommended for practical blade applications.

Solid Finite Elements

Bossak and Zienkiewicz (1973) have proposed isoparametric solid elements with reduced integration for modelling turbine and compressor blades. Several tests have also been performed to demonstrate the versatility of these elements to model both "thick" and "thin" blades in the analysis of rotating machinery. The only limitation pointed out on the aspect ratio of these elements is that the ill-conditioning is liable to occur with a 48-bit word for length/thickness ratio larger than 100. Recently, Kubiak et al. (1987) have used the eight node solid element for the stress analysis of the blades.

Employing standard tests proposed by MacNeal and Harder (1985), McGee (1987) has compared the overall performance of all solid elements in NASTRAN. Among other solid elements in NASTRAN, the twenty node isoparametric brick element with reduced integration has been highly recommended for analysis of thin and moderately thick blades.

2.2.1.2 Modelling of Disks

The assumption of rigid disks is commonly used in the analysis of turbine bladed-disk systems. In modern turbine engines designed for aerospace vehicles, however, blades are usually attached to a relatively flexible disk to meet stringent weight requirements. Considerable coupling between blades and the disk may arise and the validity of rigid disk assumption is questionable. Mota Soares et al. (1976) has pointed out the considerable effect of disk flexibility on the dynamic characteristics of turbine blades. Recent research conducted by Leissa et al. (1984) and Ernst and Lawrence (1987) has also shown that the flexibility of the blade attachment to the disk has significant effect on the dynamic characteristics of the bladed disk.

Four types of finite elements, annular, sector, shell, and solid elements, have been developed to take into account the flexibility of turbine disks. In this section, modelling of turbine disks using these finite elements is briefly reviewed.

Annular Finite Elements

An extensive literature survey of the development of annular finite elements for disk modelling has been presented by Mota Soares and Petyt (1978a). Annular elements are semi-analytical elements. In their formulation, trigonometric functions expressed by Fourier series are used to represent the displacements in the angular direction, while polynomial functions are used to approximate displacements in the radial direction. By employing the thick plate theory, one may consider the effects of shear

deformation and rotary inertia. The element thickness may be constant, linear, or parabolic in the radial direction, but only constant in the angular direction.

The use of the annular element has the advantage of reducing considerably the number of degrees of freedom in the analysis and leads to computational efficiency. However, its application to bladed disk analysis is limited due to the difficulties involved in coupling annular elements with turbine blades modelled by other finite elements.

Sector Finite Elements

Previous research on development of sector finite elements for disk modelling has been reviewed by Mota Soares and Petyt (1978a). Unlike annular elements, the displacements of sector elements are approximated by two-dimensional functions. Although a larger number of degrees of freedom is used in disk modelled by sector elements than in a corresponding disk modelled by annular elements, the sector element models offer more flexibility in coupling to blades modelled by other finite elements in bladed disk analysis.

Shell Finite Elements

Omprakash and Ramamurti (1989, 1990a, 1990b) have used a three node triangular shell element to model disk for a variety of bladed disk analyses. This shell element has six degrees of freedom per node. The disks modelled by Omprakash and Ramamurti are thin disks in which the ratio of disk thickness to disk radius ranges from 0.02 to 0.05. Therefore, further research may be needed to justify the use of this element for thick disks. In

addition, special treatments are usually required for coupling this element with turbine blades modelled by not only other finite elements but also the same element if the angle of attack of the blade is not zero.

Solid Finite Elements

Hinton and Benson (1976) have developed an isoparametric parabolic solid element to study the vibrations of disks. Recently, Midturi et al. (1987) have used the standard eight node isoparametric solid brick elements (HEX8 in NASTRAN) to model disks in analysis of flexible bladed disk assemblies. Solid elements are easy to use and flexible enough to model disks of various configurations. They can be used to model the whole turbine bladed-disk assemblies and require no special treatments at the blade-disk junction (and at the disk-shaft junction if the shaft is also modelled).

2.2.1.3 Modelling of Bladed Disk

Various modelling approaches have been proposed to account for the coupling between blades and disk. Mota Soares and Petyt (1978b) have used the sector element and the eight node superparametric thick shell element to model disk and blades, respectively. Midturi et al. (1987) have modelled the blades using the plate element and the disk using the eight node solid element. In the above two approaches, the compatibility between blades and disk is achieved by means of multi-node constraints. Recently, A three node triangular shell element has been employed by Omprakash and Ramamurti (1989, 1990a, 1990b) for modelling both blades and disk. The blade and disk attachment is established by a set of constraint equations obtained by the Love-Kirchhoff hypothesis.

2.2.2 Present Approach

As discussed in previous section, the beam and plate elements are not capable of modelling a wide range of blade configurations. The annular elements are difficult to couple with other finite elements for modelling practical bladed disks. In addition, both annular and sector elements are special-purpose elements. The limited applications of these elements in modelling only disk-like structures do not serve well the purpose of development of the general-purpose finite element system targeted in this research.

Therefore, shell and solid elements are the two candidates for modelling bladed disks in this research. For shell elements, the robust and versatile nine-noded Lagrangian element already existing in the finite element library of ABREAST has been the leading candidate. This element was originally developed by White and Abel (1990) for nonlinear static analysis of structural steel sub-assemblages. The accuracy and robustness of this element has been demonstrated by White and Abel (1989). Recently, this element has been further developed and used in nonlinear dynamic analysis to study the effects of floor flexibility on seismic response of buildings (Srivastav 1991). For solid elements, the twenty node isoparametric brick element with reduced integration recommended by McGee (1987) has been considered.

With the consideration of shell and solid elements as possible choices for modelling purposes, three possible modelling approaches are considered in this research. They are (1) the use of shell elements for blade modelling and solid elements for disk modelling, (2) the use of shell elements for both

blade and disk modelling, and (3) the use of solid elements for both blade and disk modelling.

The first approach requires the use of two different types of elements and special treatment in the region of blade and disk attachment. The second and third ones both provide a unified modelling approach while the use of constraint equations to establish the compatibility between blades and disk is required in the second one. In addition to the simplicity of the third approach, the solid elements are less complicated than shell elements and easy to use. They have been shown to be capable of reproducing plate and shell behavior accurately and economically in analysis of several blade structures (Bossak and Zienkiewicz 1973). A thorough evaluation of available published results for the static and dynamic benchmark tests (especially the free vibration tests) of shells and solids also led to the choice of solid elements. Moreover, the wide range of applications of the solid element in finite element analysis serves very well the purpose of development of the general-purpose structural dynamics simulator targeted in this research.

As a result of the above consideration, the twenty node isoparametric brick element with reduced integration is selected to model all components of turbine bladed-disks studied in the present work. The element is then implemented in the finite element library of ABREAST.

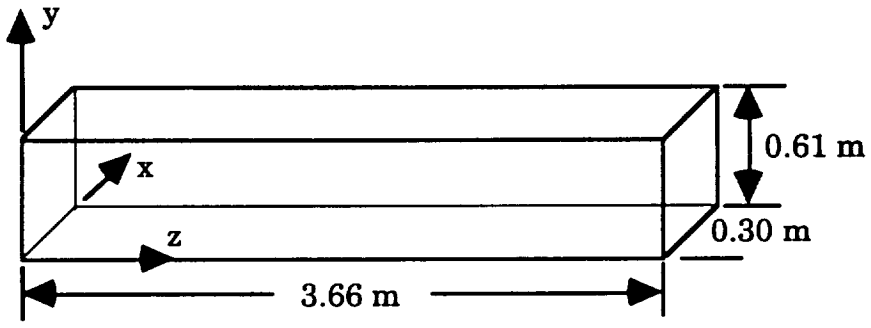
2.2.3 Verification Studies

Three example problems are used to verify the implementation of the twenty node isoparametric brick element in ABREAST for modelling turbine bladed-disks. They are free vibration analyses of a cantilever beam,

an annular plate, and twisted cantilever parallelepipeds. The numerical results from ABREAST are compared with those reported by previous researchers. All analyses performed in the present research use a) twenty-node brick elements with reduced integration, b) a lumped mass matrix formulated using the HRZ lumping scheme with full integration (Cook et al. 1989), and c) a subspace iteration method (Lin 1980) for frequency computations.

Free Vibration of a Cantilever Beam

The natural frequencies of a cantilever beam were calculated by Petyt (1990) using twenty-node solid finite elements. Figure 2.1 shows the dimensions and properties of the beam. Thirty-six elements were used in a 2 x 3 x 6 mesh. (Note that l x m x n denotes the number of elements in the x, y, and z directions, respectively.) A consistent mass matrix was employed in the vibration analysis. The computed natural frequencies of the beam were compared with results based on slender beam theory (Johnson and Field 1973).



$E = 2.068 \times 10^{11}\ N/m^2$
$\nu = 0.3$
$\rho = 8058\ Kg/m^3$

Figure 2.1 A cantilever beam used for vibration analysis

In the present research, three vibration analyses with different meshes are performed. As shown in Table 2.1, the results obtained by the present approach are in good agreement with those reported by Johnson and Field, and Petyt. The slight differences between the present results and those of Petyt are mainly due to the different mass matrices used.

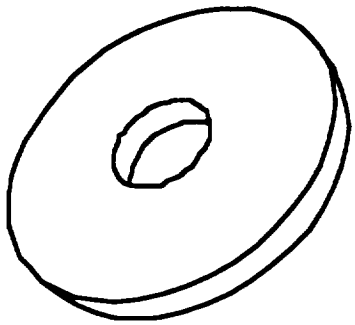
Table 2.1 Comparison of natural frequencies (Hz) of a cantilever beam obtained by the present approach with those reported by Johnson and Field (1973), and Petyt (1990)

mode no.	mode description	Johnson & Field	Petyt	present analyses		
				2x3x6	2x3x8	2x3x12
1	first bending in x direction	18.6	18.6	18.3	18.3	18.4
2	first bending in y direction	37.3	36.5	36.4	36.5	36.6
3	second bending in x direction	116.8	114.3	109.1	110.2	110.9

Free Vibration of an Annular Plate

Figure 2.2 shows an annular plate which is clamped at the inner edge while free at the outer edge. Petyt (1990) computed its natural frequencies using a two-dimensional finite element analysis with four Q8 elements. The results were compared with those reported by Rao and Prasad (1975) using plate theory. Irie et al. (1982) pointed out that Rao and Prasad's results were "incorrect, because there are probably some mistakes in analytical and computational process". They then presented their results based on the Mindlin plate theory.

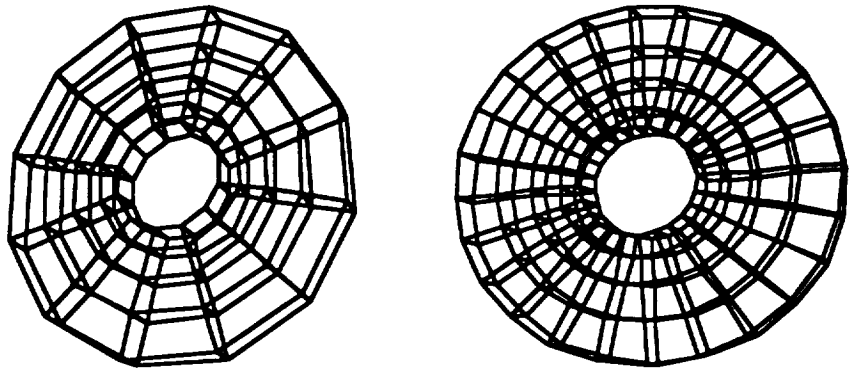
The present research uses two different meshes, as shown in Fig. 2.3, in the vibration analysis. (Note that l x m x n denotes the number of elements in the radial, circular, and thickness directions, respectively.) The present results and those reported by Rao and Prasad, Petyt, and Irie et al. are compared in Table 2.2. It can be seen that the results agree closely. However, the present results are closer to the results of Irie et al. than to those of Rao and Prasad.



inner radius = 0.3 m
outer radius = 1.0 m
thickness = 0.2 m
clamped at the inner edge
free at the outer edge

$E = 196 \times 10^9$ N/m^2
$\nu = 0.3$
$\rho = 7800$ Kg/m^3

Figure 2.2 A free-clamped annular plate used for vibration analysis



(a) 4 x 12 x 1 mesh (b) 4 x 24 x 1 mesh

Figure 2.3 Finite element meshes used in vibration analysis of an annular plate

Table 2.2 Comparison of natural frequencies (Hz) of a free-clamped annular plate obtained by the present approach with those reported by Rao and Prasad (1975), Petyt (1990), and Irie et al. (1982)

nodal circles	nodal diameters	Rao & Prasad	Petyt (2D FEM)	Irie et al.	present analyses	
					4x12x1	4x24x1
0	0	312	305	296	302	299
0	1	276	290	280	286	283
0	2	323	341	333	332	329

Free Vibration of Twisted Cantilever Parallelepipeds

The natural frequencies of twisted cantilever parallelepipeds, such as the one shown in Fig. 2.4, have been a great interest of many researchers. Recently, after giving an extensive review of previous research, McGee

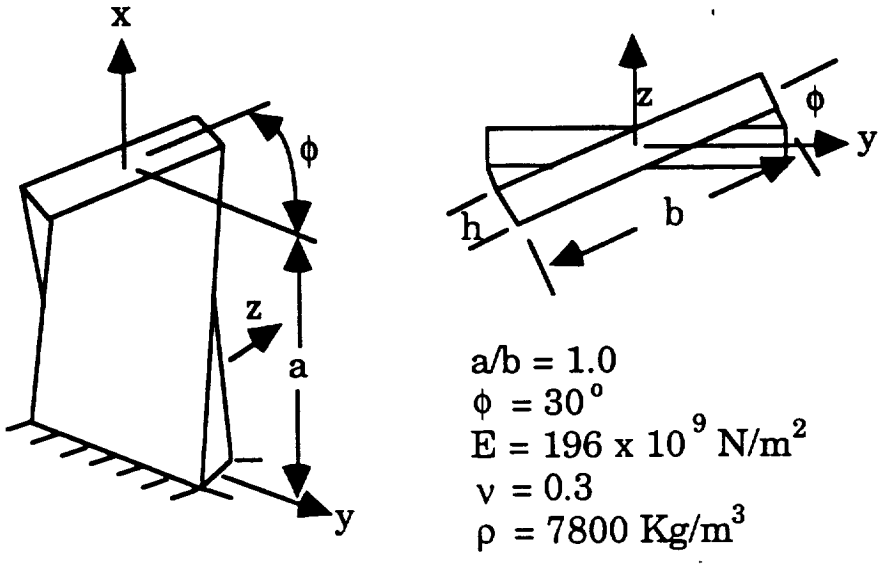


Figure 2.4 Twisted cantilevered parallelepipeds used in vibration analysis

(1992) used the 3-D Ritz method to determine the natural frequencies of a number of twisted cantilever parallelepipeds. The Ritz results were then compared with experimental results and those obtained by various 3-D finite element analyses.

In the present study, the twisted cantilever parallelepipeds shown in Fig. 2.4 are analyzed using 3-D twenty-node solid elements with 6 x 6 x 1 and 10 x 10 x 1 meshes. (Note that l x m x n denotes the number of elements in the spanwise, chordwise, and thickness directions, respectively.) In Table 2.3, the present results are compared with four sets of results, which are listed as cases A-D. Results obtained in the present study are listed as cases E and F. The thickness-independent frequency parameters, $\omega b(\rho/E)^{1/2}$, are compared. (Note that ω is the circular frequency of vibration.) for thick and thin cantilevered parallelepipeds (b/h = 5 and b/h = 20, respectively.) Case A lists experimental results reported by MacBain et al. (1985). In case B, 3-D Ritz method were employed using 6 x 4 x 4 polynomials in the x, y, and z directions, respectively McGee (1992). Standard eight- and sixteen-node isoparametric solid elements were employed in cases C and D, respectively (Kielb et al. 1985; Leissa et al. 1984). In case C, a 10 x 10 x 1 mesh and a lumped mass matrix were used. while a 14-point integration rule on a 24 x 12 x 1 mesh and a consistent mass matrix were used in case D.

For most of the modes calculated, the present results agree with the 3-D Ritz results closer than the finite element results of cases C and D although they all agree closely. The present results are also in good agreement with the experimental results (case A) for the thin (b/h = 20) twisted parallelepiped model. However, for thicker model, the present

results as well as results of cases B-D do not agree closely with the experimental results. As pointed out by McGee (1992), this is mainly due to imperfect clamping of the specimen during testing.

Table 2.3 Comparison of thickness-independent frequency parameters, $\omega b(\rho/E)^{1/2}$ of cantilevered parallelepipeds obtained by the present approach with those reported by previous researchers (a/b = 1, $\phi = 30^{o}$)

mode no.		1	2	3	4	5	6	7	8
b/h = 5	A	0.170	0.418	0.366	0.796	---	1.283	---	---
	B	0.196	0.464	0.632	1.063	1.409	1.428	1.567	1.779
	C	0.194	0.446	0.628	0.990	1.217	1.313	1.560	1.744
	D	0.217	0.479	0.640	1.313	1.453	1.502	1.574	1.789
	E	0.194	0.446	0.623	1.005	1.249	1.336	1.558	1.712
	F	0.193	0.450	0.625	1.018	1.278	1.362	1.560	1.732
b/h = 20	A	0.048	0.198	0.252	0.383	0.474	0.593	0.757	---
	B	0.050	0.214	0.272	0.387	0.469	0.620	0.681	0.695
	C	0.051	0.215	0.276	0.392	0.505	0.721	0.795	0.888
	D	0.057	0.220	0.305	0.431	0.537	0.710	0.875	0.968
	E	0.050	0.209	0.269	0.385	0.492	0.677	0.770	0.847
	F	0.050	0.212	0.271	0.398	0.505	0.688	0.797	0.864

A = experimental results (MacBain et al. 1985)

B = 3-D Ritz method (McGee 1992)

C = 3-D finite elements - 8-node solid (Kielb et al. 1985; Leissa et al. 1984)

D = 3-D finite elements - 16-node solid (Kielb et al. 1985; Leissa et al. 1984)

E = present 20-node solid elements (6 x 6 x 1 mesh)

F = present 20-node solid elements (10 x 10 x 1 mesh)

2.3 Formulation of Equations of Motion

The finite element formulation of equations of motion for dynamic analysis of both nonrotating and rotating systems is presented and discussed in this section. Since the dynamics of nonrotating objects is just a special case of that of rotating objects, the formulation presented here targets the dynamic analysis of rotating systems, in particular, rotating bladed-disks experiencing tip rubs.

For the clarity of derivation and the convenience of discussion, the equations of motion are first derived for an elastic analysis which accounts for only nonlinearity associated with kinetic energy of the rotating system. Then, by using the updated Lagrangian approach, the analysis is extended to take into account large displacements (but small strains) of flexible turbine blades, which is the geometric nonlinearity associated with potential energy of the system.

2.3.1 Coordinate Systems

Figure 2.5 shows two Cartesian coordinate systems used in the formulation, an inertial coordinate system (X-Y-Z) which is absolutely fixed in space and a undeformed body-fixed coordinate system (x-y-z) which is fixed to and rotating with the undeformed structure. The origin of x-y-z system is denoted as O'. For the rotating bladed-disk system studied in this work, it represents the center of the disk. Point **i** is the undeformed position of a typical material point in a finite element of the model, while point **i'** is the deformed position of the material point. Vectors $\mathbf{P_i}$ and $\mathbf{R}$ are position vectors of points **i** and O', respectively, observed from the X-Y-Z system, and vectors $\mathbf{r_i}$ and $\mathbf{u_i}$ are position and displacement vectors, respectively, of

point i observed from the x-y-z system. Vectors ω and α are the rotational velocity and acceleration, respectively, of the x-y-z system with respect to the X-Y-Z system. It should be noted that all vectors used in the following formulation are referred to the rotating x-y-z system.

2.3.2 Assumptions

The following assumptions are used in the derivation of equations of motion for the rotating bladed-disk system considered in the present work:

(1) there is no translational motions of the disk center of the bladed-disk system (i.e., rigid shaft and bearings are assumed and vector **R** in Fig. 2.5 is a constant vector), and

(2) the structure may undergo large deformations but strains remain small and the material remains elastic.

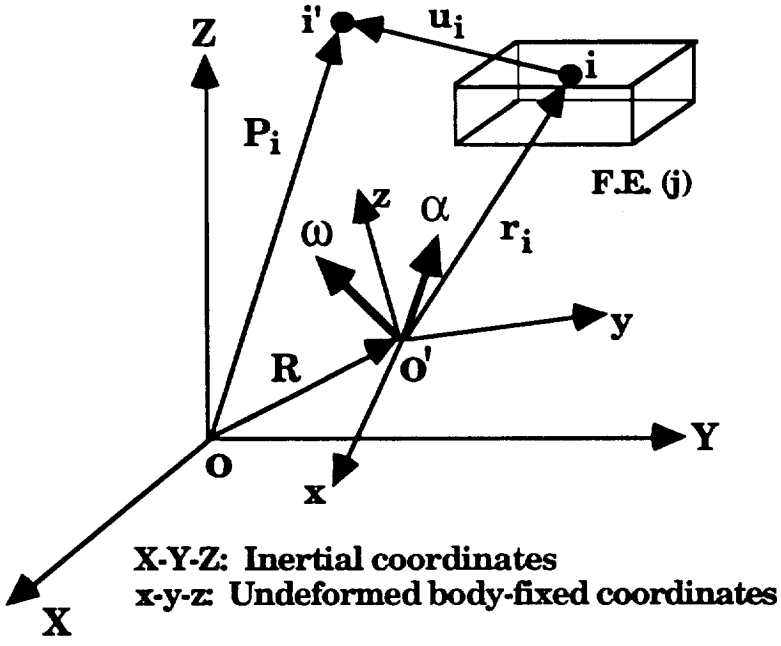


Figure 2.5 Coordinate systems used in finite element formulation

2.3.3 Rotational Nonlinearities

Nonlinearities due to rotational effects, such as centrifugal and Coriolis effects, are considered in this section for an elastic dynamic analysis. Two approaches are presented in this thesis to incorporate this nonlinearity into equations of motions of the rotating system. They are called by the writer *consistent mass* and *lumped mass* approaches. The consistent mass approach treats the structure as a continuum with mass points uniformly distributed in the structure, while the lumped mass treats the structure as a collection of discrete concentrated mass points.

It should be noted that the consistent mass approach presented here is general for all types of finite elements. The lumped mass approach presented here, however, is only applicable directly to finite elements with merely translational degrees of freedom, such as truss and solid elements. The extension of the lumped mass approach for finite elements with both translational and rotational degrees of freedom is briefly discussed in Section 2.3.3.2.

2.3.3.1 Consistent Mass Approach

In the displacement-based finite element approximation, the displacement field of an element is assumed as

$$\{u\} = [N]\{q\} \tag{2.1}$$

in which [N] is the shape function for displacements and {q} is the nodal displacement vector of the element, while the element geometry field is assumed as

$$\{r\} = [N']\{c\} \tag{2.2}$$

in which [N'] is the shape function for coordinates and {c} contains the nodal coordinates of the element. For isoparametric finite elements, [N'] and [N] are identical. The strain-displacement relationship is expressed as

$$\{\varepsilon\} = [B]\{q\} \tag{2.3}$$

in which [B] is the strain-displacement matrix. The stress-strain relationship is expressed as

$$\{\sigma\} = [C]\{\varepsilon\} \tag{2.4}$$

in which [C] is the material stiffness matrix.

As shown in Fig. 2.5, the instantaneous position vector of a material point in the element is

$$\{P\} = \{R\} + \{r\} + \{u\} \tag{2.5}$$

and the corresponding absolute velocity vector is

$$\{v\} = \{\dot{u}\} + \{\omega\} \times (\{r\} + \{u\}) = \{\dot{u}\} + [\Omega](\{r\} + \{u\}) \tag{2.6}$$

in which

$$[\Omega] = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$

Note that $\{\dot{R}\} = 0$ due to the first assumption, and the components in $\{\omega\}$ as well as in $[\Omega]$ are function of time.

The kinetic energy of the element is

$$T = \frac{1}{2}\int_{vol} \rho \{v\}^T\{v\}\, d(vol) \tag{2.7}$$

in which ρ is the density of the material. From Eqs. (2.6) and (2.7), the expression of the kinetic energy becomes

$$\begin{aligned} T = {} & \frac{1}{2}\int_{vol} \rho \{\dot{u}\}^T\{\dot{u}\}\, d(vol) + \frac{1}{2}\int_{vol} \rho \{r\}^T[\Omega^2]\{r\}d(vol) \\ & + \frac{1}{2}\int_{vol} \rho \{u\}^T[\Omega^2]\{u\}d(vol) + \int_{vol} \rho \{\dot{u}\}^T[\Omega]\{r\}d(vol) \\ & + \int_{vol} \rho \{r\}^T[\Omega^2]\{u\}d(vol) + \int_{vol} \rho \{\dot{u}\}^T[\Omega]\{u\}d(vol) \end{aligned} \tag{2.8}$$

in which $[\Omega^2] = [\Omega]^T[\Omega]$. After substituting Eqs. (2.1) and (2.2) into Eq. (2.8), one has

$$\begin{aligned} T = {} & \frac{1}{2}\int_{vol} \rho \{\dot{q}\}^T[N]^T[N]\{\dot{q}\}\, d(vol) + \frac{1}{2}\int_{vol} \rho \{r\}^T[\Omega^2]\{r\}d(vol) \\ & + \frac{1}{2}\int_{vol} \rho \{q\}^T[N]^T[\Omega^2][N]\{q\}d(vol) + \int_{vol} \rho \{\dot{q}\}^T[N]^T[\Omega]\{r\}d(vol) \\ & + \int_{vol} \rho \{r\}^T[\Omega^2][N]\{q\}d(vol) + \int_{vol} \rho \{\dot{q}\}^T[N]^T[\Omega][N]\{q\}d(vol) \end{aligned} \tag{2.9}$$

The potential energy of the element is

$$U = \frac{1}{2}\int_{vol} \{\varepsilon\}^T\{\sigma\}\, d(vol) \tag{2.10}$$

From Eqs. (2.3), (2.4), and (2.10), the expression of the (geometrically linear) potential energy becomes

$$U = \frac{1}{2}\int_{vol} \{q\}^T[B]^T[C][B]\{q\}\, d(vol) \tag{2.11}$$

Applying the Lagrange equation

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}}\right) - \frac{\partial T}{\partial q} + \frac{\partial U}{\partial q} = \{F^{ext}\} \tag{2.12}$$

one obtains the equations of motion for the element

$$[m]\{\ddot{q}\} + [c_c]\{\dot{q}\} + ([k_e] + [k_a] - [k_r])\{q\} + \{f_e\} = \{f^{ext}\} \tag{2.13}$$

in which

$[m]$ = element mass matrix = $\int_{vol} \rho\, [N]^T[N]\, d(vol)$

$[c_c]$ = element Coriolis damping matrix = $2\int_{vol} \rho\, [N]^T[\Omega][N]\, d(vol)$

$[k_e]$ = element elastic stiffness matrix = $\int_{vol} [B]^T[C][B]\, d(vol)$

$[k_a]$ = element centripetal stiffness matrix = $\int_{vol} \rho\, [N]^T[A][N]\, d(vol)$

$[k_r]$ = element centrifugal stiffness matrix = $\int_{vol} \rho\, [N]^T[\Omega^2][N]\, d(vol)$

$\{f_e\}$ = element rotational force vector

$$= \int_{vol} \rho\, [N]^T[A][N']\{c\}\, d(vol) \; - \int_{vol} \rho\, [N]^T[\Omega^2][N']\{c\}\, d(vol)$$

$\{f^{ext}\}$ = element external load vector

and

$$[A] = [\dot{\Omega}] = \begin{bmatrix} 0 & -\alpha_z & \alpha_y \\ \alpha_z & 0 & -\alpha_x \\ -\alpha_y & \alpha_x & 0 \end{bmatrix}$$

If additional damping of the structural system is to be considered, it is frequently modelled by an element equivalent viscous damping matrix, $[c_v]$, which is added to the coefficient of the velocity term $\{\dot{q}\}$.

After assembly, the final global equations of motion will be of the form

$$[M]\{\ddot{Q}\} + ([C_v] + [C_c])\{\dot{Q}\} + ([K_e] + [K_a] - [K_r])\{Q\} + \{F_e\} = \{F^{ext}\} \quad (2.14)$$

It should be noted that the formulation presented above is similar to the one reported by Omprakash and Ramamurti (1989) except that the assumption of constant ω (rotational velocity vector) is not used in the present formulation. The formulation is different from the one presented by Davis (1989). In his approach, the system model is first discretized into finite elements with either concentrated masses described by a lumped mass matrix or distributed masses specified by a consistent mass matrix. Using the generalized Newton's second law, Davis then derived the kinetic equations of motion based on the already discretized finite element model without the use of shape functions of the finite element for interpolating fields within the element. However, the lumped mass approach presented below is similar to his approach with a lumped mass matrix except that the present formulation mainly considers finite elements with merely translational degrees of freedom.

2.3.3.2 Lumped Mass Approach

Consider the finite element shown in Fig. 2.5 with masses already lumped at its nodes. With the first assumption applied, the nodal instantaneous position vector of the element is

$$\{P_n\} = \{c\} + \{q\} \tag{2.15}$$

and the corresponding absolute nodal velocity vector is

$$\{v_n\} = \{\dot{u}\} + \{\omega\} \times (\{c\} + \{q\}) = \{\dot{q}\} + \langle[\Omega]\rangle (\{c\} + \{q\}) \tag{2.16}$$

in which $\langle[\Omega]\rangle$ is a block diagonal matrix with $[\Omega]$'s on the diagonal.

The kinetic energy of the element is

$$T = \frac{1}{2} \{v_n\}^T [m_L] \{v_n\} \tag{2.17}$$

in which $[m_L]$ is the lumped element mass matrix. From Eqs. (2.16) and (2.17), the kinetic energy is expressed as

$$\begin{aligned} T = {} & \frac{1}{2} \{\dot{q}\}^T [m_L] \{\dot{q}\} + \frac{1}{2} \{c\}^T \langle[\Omega]\rangle^T [m_L] \langle[\Omega]\rangle \{c\} \\ & + \frac{1}{2} \{q\}^T \langle[\Omega]\rangle^T [m_L] \langle[\Omega]\rangle \{q\} + \{\dot{q}\}^T [m_L] \langle[\Omega]\rangle \{c\} \\ & + \{c\}^T \langle[\Omega]\rangle^T [m_L] \langle[\Omega]\rangle \{q\} + \{\dot{q}\}^T [m_L] \langle[\Omega]\rangle \{q\} \end{aligned} \tag{2.18}$$

The potential energy of the element is

$$U = \frac{1}{2} \int_{vol} \{q\}^T [B]^T [C] [B] \{q\} \, d(vol) \tag{2.19}$$

Applying the Lagrange equation (Eq. 2.12) and taking into account equivalent viscous damping of the element, one has the equations of motion for the element

$$[m_L]\{\ddot{q}\} + ([c_v] + [c_c])\{\dot{q}\} + ([k_e] + [k_a] - [k_r])\{q\} + \{f_e\} = \{f^{ext}\} \qquad (2.20)$$

in which

$[c_v]$ = element equivalent viscous damping matrix

$[c_c]$ = element Coriolis damping matrix = $[m_L]\langle[\Omega]\rangle - \langle[\Omega]\rangle^T[m_L]$

$= [m_L]\langle[\Omega]\rangle + \langle[\Omega]\rangle[m_L]$

$[k_e]$ = element elastic stiffness matrix = $\int_{vol} [B]^T[C][B]\, d(vol)$

$[k_a]$ = element centripetal stiffness matrix = $[m_L]\,\langle[A]\rangle$

$[k_r]$ = element centrifugal stiffness matrix = $\langle[\Omega]\rangle^T[m_L]\langle[\Omega]\rangle$

$\{f_e\}$ = element rotational force vector

$= [m_L]\,\langle[A]\rangle\,\{c\} - \langle[\Omega]\rangle^T[m_L]\langle[\Omega]\rangle\,\{c\}$

$\{f^{ext}\}$ = element external load vector

and

$$\langle[A]\rangle = \langle[\dot{\Omega}]\rangle$$

After assembly, the final global equations of motion will be of the form

$$[M_L]\{\ddot{Q}\} + ([C_v] + [C_c])\{\dot{Q}\} + ([K_e] + [K_a] - [K_r])\{Q\} + \{F_e\} = \{F^{ext}\} \quad (2.21)$$

By assuming that the lumped masses have no rotational inertia (i.e., the mass associated with any rotational degrees of freedom is zero), the above formulation can be easily extended for finite elements with both translational and rotational degrees of freedom.

It can be seen that the lumped mass approach is computationally more efficient than the consistent one. However, since the mass coupling between element nodes is neglected in the formulation, this approach is not expected to be as accurate as the consistent one. Later in this chapter, numerical comparison between the lumped mass and consistent mass approaches will be conducted to examine the questions of accuracy and applicability of the lumped mass approach.

2.3.4 Geometric Stiffness Effects

As indicated by Lawrence and Kielb (1984), and Simo and Vu-Quoc (1987), a geometric nonlinear analysis is usually required for accurately predicting dynamic behavior of rotating blades. The present research adopts the formulation presented by Bathe (1982) with slight modification to account for geometric nonlinearity in the analysis using the updated Lagrangian approach.

Figure 2.6 shows the motion of a typical body in the undeformed body-fixed coordinate system (x-y-z). Configuration 0 (C_0) represents the original undeformed state; configuration 1 (C_1) represents the current (known) deformed state; and configuration 2 (C_2) represents the desired (unknown) deformed state.

Similar to the tensor notation used by Bathe (1982), both left subscripts and superscripts on a symbol are used to denote the three configurations shown in Fig. 2.6. A left superscript denotes in which configuration the quantity occurs. The absence of such a superscript indicates that the quantity is an increment between C_1 and C_2. A left subscript denotes in which configuration the quantity is measured. In the notation adopted, a comma denotes differentiation with respect to the coordinate following; thus, for example, ${}^{2}_{1}u_{i,j} = \dfrac{\partial^2 u_i}{\partial {}^1x_j}$

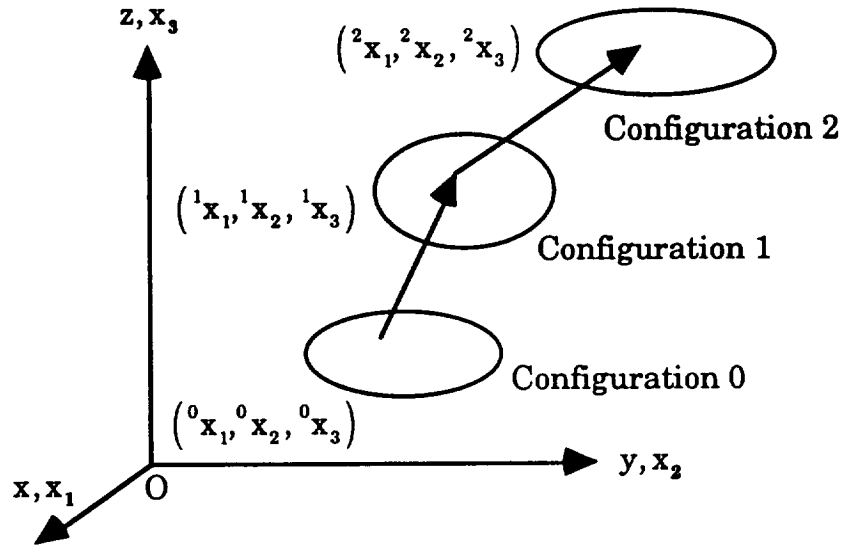


Figure 2.6 Motion of body in body-fixed undeformed coordinate system

The principle of virtual displacements gives the equilibrium of the body expressed in the deformed configuration C_2 being sought

$$\int_{{}^2V} {}^2\tau_{ij}\ \delta_2 e_{ij}\ {}^2dV = {}^2_2R \tag{2.22}$$

in which ${}^{2}\tau_{ij}$ is the Cauchy stress tensor, $\delta_2 e_{ij}$ is the variation of an infinitesimal strain tensor, and ${}^{2}_{2}R$ is the external virtual work, with

$$ {}^{2}_{2}R = \int_{{}^{2}V} {}^{2}f_i^b \; \delta_2 u_i \; {}^{2}dV + \int_{{}^{2}S} {}^{2}f_i^s \; \delta_2 u_i \; {}^{2}dS \qquad (2.23) $$

in which ${}^{2}f_i^b$ and ${}^{2}f_i^s$ are the components of the externally applied body and surface force vectors, respectively, and $\delta_2 u_i$ is the variation of the displacement component.

In the updated Lagrangian formulation, Eq. (2.22) can be transformed to (Bathe 1982)

$$ \int_{{}^{1}V} {}^{2}_{1}S_{ij} \; \delta_1 \varepsilon_{ij} \; {}^{1}dV = {}^{2}_{1}R \qquad (2.24) $$

in which ${}^{2}_{1}S_{ij}$ is the 2nd Piola-Kirchhoff stress tensor and $\delta_1 \varepsilon_{ij}$ is the variation of the Green-Lagrange strain tensor. The incremental decomposition of stress tensor is expressed as

$$ {}^{2}_{1}S_{ij} = {}^{1}\tau_{ij} + {}_{1}S_{ij} \qquad (2.25) $$

and the strain increments can be decomposed into

$$ {}_{1}\varepsilon_{ij} = {}_{1}e_{ij} + {}_{1}\eta_{ij} \qquad (2.26) $$

in which ${}_{1}e_{ij}$ and ${}_{1}\eta_{ij}$ are the linear and nonlinear components of the Green-Lagrange incremental strain tensor, respectively. Using the linearized constitutive relationship between stress and strain increments,

$$ {}_1S_{ij} = {}_1C_{ijkl}\, {}_1e_{kl} \tag{2.27} $$

and substituting Eqs. (2.25) and (2.26) into Eq. (2.24), one obtains the incremental equilibrium equation

$$ \int_{{}^1V} {}_1C_{ijkl}\ {}_1e_{kl}\ \delta_1 e_{ij}\ {}^1dV + \int_{{}^1V} {}^1\tau_{ij}\ \delta_1\eta_{ij}\ {}^1dV $$

$$ = {}^2_1R - \int_{{}^1V} {}^1\tau_{ij}\ \delta_1 e_{ij}\ {}^1dV \tag{2.28} $$

After applying finite element approximation, one has the matrix form of the incremental equilibrium equation (Bathe 1982)

$$ ({}^1_1[K_e] + {}^1_1[K_g])\{\Delta Q\} = {}^2_1R - {}^1_1\{F\} \tag{2.29} $$

in which

$$ {}^1_1[K_e] = \text{elastic stiffness matrix} = \int_{{}^1V} {}^1_1[B_L]^T\ {}_1[C]\ {}^1_1[B_L]\ {}^1dV $$

$$ {}^1_1[K_g] = \text{geometric stiffness matrix} = \int_{{}^1V} {}^1_1[B_{NL}]^T\ {}^1[\tau]\ {}^1_1[B_{NL}]\ {}^1dV $$

$$ {}^1_1\{F\} = \text{internal force vector} = \int_{{}^1V} {}^1_1[B_L]^T\ {}^1\{\tau\}\ {}^1dV $$

and ${}^1_1[B_L]$ and ${}^1_1[B_{NL}]$ are linear and nonlinear strain-displacement transformation matrices, respectively.

Incorporating Eq. (2.29) into Eq. (2.14) for dynamic analysis, one then has the equations of motion for geometric nonlinear analysis:

$$[M]\,{}^{t+\Delta t}\{\ddot{Q}\} + ({}^{t}_{t}[C_v] + {}^{t}_{t}[C_c])\,{}^{t+\Delta t}\{\dot{Q}\} + ({}^{t}_{t}[K_e] + {}^{t}_{t}[K_g] + {}^{t}_{t}[K_a] - {}^{t}_{t}[K_r])\{\Delta Q\}^{i}$$

$$= {}^{t+\Delta t}\{F^{ext}\} - {}^{t+\Delta t}_{t+\Delta t}\{F\}^{i-1} - {}^{t}_{t}\{F_e\} - ({}^{t}_{t}[K_a] - {}^{t}_{t}[K_r])\,{}^{t}\{Q\} \qquad (2.30)$$

for implicit time integration, and

$$[M]\,{}^{t}\{\ddot{Q}\} + ({}^{t}_{t}[C_v] + {}^{t}_{t}[C_c])\,{}^{t}\{\dot{Q}\} = {}^{t}\{F^{ext}\} - {}^{t}_{t}\{F\} - {}^{t}_{t}\{F_e\} - ({}^{t}_{t}[K_a] - {}^{t}_{t}[K_r])\,{}^{t}\{Q\} \qquad (2.31)$$

for explicit time integration, in which [M] is the time-independent mass matrix and the symbols t+Δt and t denotes the configurations at time t+Δt and t, respectively. The right superscript in the implicit scheme indicates the iteration number with respect to which the quantity is evaluated in the iterative procedure.

In addition, for steady state stress analysis, the equilibrium equations are expressed as

$$({}^{t}_{t}[K_e] + {}^{t}_{t}[K_g] - {}^{t}_{t}[K_r])\{\Delta Q\}^{i} = {}^{t+\Delta t}\{F^{ext}\} - {}^{t+\Delta t}_{t+\Delta t}\{F\}^{i-1} - {}^{t}_{t}\{F_e\} + {}^{t}_{t}[K_r]\,{}^{t}\{Q\} \qquad (2.32)$$

For undamped vibration analysis and with the Coriolis matrix neglected, the equations of motion have the form of

$$[M]\{\ddot{Q}\} + ([K_e] + [K_g] - [K_r])\{Q\} = \{0\} \qquad (2.33)$$

which is employed for vibration analysis after the determination of the static stresses from Eq. (2.32).

It should be noted that Eqs. (2.30), (2.31) and (2.32) assume the use of initial undeformed coordinates to evaluate ${}^{t}_{t}\{F_e\}$. If the updated coordinates

are used (this is usually the case in the updated Lagrangian approach), the term $({}^{t}_{t}[K_a] - {}^{t}_{t}[K_r])^{t}\{Q\}$ in both Eqs. (2.30) and (2.31), and the term $({}^{t}_{t}[K_r])^{t}\{Q\}$ in Eq. (2.32) are automatically taken into account in ${}^{t}_{t}\{F_e\}$ through the process of coordinate update. In this case, however, it should be also noted that the shape function in ${}^{t}_{t}\{F_e\}$ is different from the shape function in both ${}^{t}_{t}[K_a]$ and ${}^{t}_{t}[K_r]$ if the consistent mass approach is employed and the finite element used is not isoparametric.

For nonrotating systems (i.e., $\{\omega\} = \{0\}$ in Eq. 2.6), the formulation reverts to the standard forms, i.e., the equations of motions are

$$[M]\,{}^{t+\Delta t}\{\ddot{Q}\} + {}^{t}_{t}[C_v]\,{}^{t+\Delta t}\{\dot{Q}\} + ({}^{t}_{t}[K_e] + {}^{t}_{t}[K_g])\{\Delta Q\}^{i} = {}^{t+\Delta t}\{F^{ext}\} - {}^{t+\Delta t}_{t+\Delta t}\{F\}^{i-1} \qquad (2.34)$$

for implicit time integration, and

$$[M]\,{}^{t}\{\ddot{Q}\} + {}^{t}_{t}[C_v]\,{}^{t}\{\dot{Q}\} = {}^{t}\{F^{ext}\} - {}^{t}_{t}\{F\} \qquad (2.35)$$

for explicit time integration, while for undamped vibration analyses, the equations of motion have the form of

$$[M]\{\ddot{Q}\} + ([K_e] + [K_g])\{Q\} = \{0\} \qquad (2.36)$$

2.4 Analysis of Framed Structures

The present work takes advantage of the capabilities already provided in ABREAST for dynamic analysis of framed structures. These capabilities include eigensolvers for undamped vibration analysis governed by Eq. (2.36) and direct time integration solvers for transient analysis governed by Eqs. (2.34) and (2.35). They are briefly discussed in this section. More detailed descriptions have been provided by Srivastav (1991)

and Aubert (1992). The enhancement of the time integration algorithms for parallel transient analysis in this work is discussed in Section 3.2.1.

For free vibration analyses, Eq. (2.36) is usually transformed into an eigenproblem of the form (Bathe 1982)

$$[K]\{\Phi\} = \omega^2 [M]\{\Phi\} \tag{2.37}$$

in which [K] and [M] are the stiffness and mass matrices of the system, and ω and $\{\Phi\}$ are the eigenvalue and eigenvector corresponding to the angular vibration frequency and mode shape of a mode. In ABREAST, two eigensolvers are provided for solving Eq. (2.37). The first solver uses a rational QL method (Wilkinson and Reinsch 1971) to solve the full set of eigenvalues and eigenvectors of the system. The second one uses a subspace iteration algorithm (Lin 1980) to extract only a desired number of low modes of vibration. In the present work, the subspace iteration solver is used due to its feasibility and efficiency for large structural problems.

Both explicit and implicit integration methods are available in ABREAST for transient dynamic analysis. The explicit time integration uses the central difference method to solve the equations of motion given by Eq. (2.35). The central difference method approximates velocity and acceleration with second-order accuracy by

$$^{t}\{\dot{Q}\} = (\, ^{t+\Delta t}\{Q\} - \, ^{t-\Delta t}\{Q\} \,) / (2\Delta t) \tag{2.38}$$

$$^{t}\{\ddot{Q}\} = (\, ^{t+\Delta t}\{Q\} - 2 \, ^{t}\{Q\} + \, ^{t-\Delta t}\{Q\} \,) / (\Delta t^2) \tag{2.39}$$

in which Δt is the constant time step size. Substitution of Eqs. (2.38) and (2.39) into Eq. (2.35) yields

$$((1/\Delta t^2)[M] + (1/(2\Delta t))\,{}^{t}_{t}[C_v])\,{}^{t+\Delta t}\{Q\} = {}^{t}\{F^{ext}\} - {}^{t}_{t}\{F\} + (2/\Delta t^2)[M]\,{}^{t}\{Q\} - ((1/\Delta t^2)[M] - (1/(2\Delta t))\,{}^{t}_{t}[C_v])\,{}^{t-\Delta t}\{Q\} \qquad (2.40)$$

in which the damping matrix $[C_v]$ is assumed to be the linear combination of the mass and stiffness matrices (Rayleigh damping). With the use of lumped masses and either no damping or mass proportional damping, Eq. (2.40) is a set of uncoupled equations and its solution can proceed on a degree-of-freedom level without assembly of the global stiffness matrix and solution of simultaneous equation. In the case of nondiagonal damping (i.e., the stiffness-proportional damping is included), Eq. (2.38) is replaced by a first-order accurate approximation for the velocity

$${}^{t}\{\dot{Q}\} = ({}^{t}\{Q\} - {}^{t-\Delta t}\{Q\}) / \Delta t \qquad (2.41)$$

Substitution of Eqs. (2.41) and (2.39) into Eq. (2.35) yields

$$(1/\Delta t^2)[M]\,{}^{t+\Delta t}\{Q\} = {}^{t}\{F^{ext}\} - {}^{t}_{t}\{F\} + ((2/\Delta t^2)[M] - (1/\Delta t)\,{}^{t}_{t}[C_v])\,{}^{t}\{Q\} - ((1/\Delta t^2)[M] - (1/\Delta t)\,{}^{t}_{t}[C_v])\,{}^{t-\Delta t}\{Q\} \qquad (2.42)$$

With the use of lumped masses, solution of Eq. (2.42) can also proceed on a degree-of-freedom level. Since the explicit central difference method is only conditionally stable, the stability conditions for Eq. (2.40) and (2.42) are $\Delta t \le 2/\omega_{max}$ and $\Delta t \le (2/\omega_{max})(\sqrt{1+\xi^2} - \xi)$, respectively, where ω_{max} is the highest undamped natural frequency of the system and ξ is the fraction of critical damping at the highest natural frequency, ω_{max} (Cook et al. 1989).

The implicit integration in ABREAST uses the Newmark family of schemes to solve the equations of motion given by Eq. (2.34). The Newmark method relates displacements, velocities, and accelerations by (Bathe 1982)

$$ {}^{t+\Delta t}\{\dot{Q}\} = {}^{t}\{\dot{Q}\} + ((1 - \delta)\,{}^{t}\{\ddot{Q}\} + \delta\,{}^{t+\Delta t}\{\ddot{Q}\})\,\Delta t \tag{2.43} $$

$$ {}^{t+\Delta t}\{Q\} = {}^{t}\{Q\} + {}^{t}\{\dot{Q}\}\Delta t + ((1/2 - \alpha)\,{}^{t}\{\ddot{Q}\} + \alpha\,{}^{t+\Delta t}\{\ddot{Q}\})\Delta t^2 \tag{2.44} $$

The present work uses the constant average acceleration scheme, in which $\delta = 1/2$ and $\alpha = 1/4$, because it is unconditionally stable and second-order accurate. Substitution of Eqs. (2.43) and (2.44) into Eq. (2.34) and rearrangement of terms result in the incremental equilibrium equation of the form

$$ {}^{t}[\hat{K}]\,\{\Delta Q\}^{i} = {}^{t+\Delta t}\{\hat{R}\} \tag{2.45} $$

in which

$$ {}^{t}[\hat{K}] = (1/(\alpha\Delta t^2))[M] + (\delta/(\alpha\Delta t))\,{}^{t}_{t}[C_v] + {}^{t}_{t}[K_e] + {}^{t}_{t}[K_g] \tag{2.46} $$

and

$$ \begin{aligned} {}^{t+\Delta t}\{\hat{R}\} = {}& {}^{t+\Delta t}\{F^{ext}\} - {}^{t+\Delta t}_{t+\Delta t}\{F\}^{i-1} - \\ & [M]((1/(\alpha\Delta t^2))({}^{t+\Delta t}\{Q\}^{i-1} - {}^{t}\{Q\}) - (1/(\alpha\Delta t))\,{}^{t}\{\dot{Q}\} - ((1/2 - \alpha)/\alpha)\,{}^{t}\{\ddot{Q}\}) - \\ & {}^{t}_{t}[C_v]((\delta/(\alpha\Delta t))({}^{t+\Delta t}\{Q\}^{i-1} - {}^{t}\{Q\}) - (\delta/\alpha + 1)\,{}^{t}\{\dot{Q}\} - (\delta/\alpha + 1)\Delta t\,{}^{t}\{\ddot{Q}\}) \end{aligned} \tag{2.47} $$

A modified-Newton iterative solution scheme is used to obtain the equilibrium solution of Eq. (2.47).

The implementation of the above analysis capabilities in ABREAST has been verified by Srivastav (1991) and Aubert (1992). Therefore, no further verification is conducted in this work.

2.5 Analysis of Rotating Bladed-disk Assemblies

In recent years the trend for turbine bladed-disk assemblies to have higher efficiency, performance, and reliability has significantly increased

complexities and difficulties in the structural analysis and design of rotating turbine bladed-disk assemblies. To achieve better understanding and prediction of behavior of turbine bladed-disk assemblies, considerable research has been conducted to study the steady-state responses and modal vibrations of rotating turbine bladed-disk systems. However, little research has been conducted to investigate the transient responses of rotating bladed-disk assemblies during a unsteady motion induced by events such as the start-up, blade tip rubbing, speed or load changes, and passing through resonant frequencies. The reason for this is usually the lack of both capable analysis tools and adequate computing power to carry out this type of analysis due to the complexity and nonlinearity involved in the analysis and the large size of the problem.

This section discusses the analysis approaches used in this work and the enhancement of analysis tools in ABREAST for both modal vibration and transient dynamic analyses of rotating bladed-disks modelled by solid finite elements. Verification studies of the current implementation are also provided. The use of parallel processing in a network of powerful workstations to provide the considerable computing power needed in the analysis is investigated in Chapters 3 and 4. In addition, the use of interactive computer graphics to facilitate the modelling and visualization of the dynamic simulation is presented in Chapter 5.

Due to the attempt in modern design of turbine bladed-disk assemblies to minimize the clearance between blades and housing for efficiency and performance optimization, the probability of blade tip rubbing has been greatly increased. The present work emphasizes the transient dynamic analysis of rotating bladed-disk assemblies experiencing tip rubs.

However, the modal vibration analysis of rotating bladed-disk systems studied in this work plays an important role in verifying the present approach and implementation. The present study also contributes additional sets of data of 3-D finite element vibration analyses of rotating turbine blades and annular disks.

In the following subsections, a brief review of previous research is given first. The approach used in the present work is then discussed. Results obtained using the present approach are compared with those obtained by previous researchers to assess the accuracy of the present approach. Furthermore, results obtained using the lumped mass approach are compared with those obtained using the consistent mass approach in the same verification studies.

2.5.1 Review of Previous Research

Previous research on both modal vibration and transient dynamic characteristics of rotating turbine bladed-disk assemblies is brief reviewed as follows.

Modal Vibration Analysis

Free vibration characteristics of rotating turbine blades have been studied by many researchers. Earlier research on the analysis of rotating blades has been reviewed by Ramamurti and Balasubramanian (1984) and Rao (1987). In most previous research, beam theory was used in the vibration analysis of blades often idealized as cantilever beams. Although many effects such as pre-twist, hub radius, setting angle, hub flexibility, and tip mass were investigated in the analysis, the effects of shear

deformation and rotary inertia were often neglected. Recently, Yokoyama (1988) used the finite element method to determine the bending frequencies of a rotating uniform Timoshenko beam. Khulief and Bazoune (1992) calculated the first three frequencies of rotating tapered Timoshenko beams with different boundary conditions. However, it has been recognized that the use of beam theory is not sufficient for modelling blades with low aspect ratio and complex configurations. Some previous applications of plate and shell theories to study frequencies of rotating blades were briefly reviewed by Sreenivasamurthy and Ramamurti (1981). More recent examples are studies conducted by Omprakash and Ramamurti (1989, 1990a, 1990b).

Some research has been reported on the vibration analysis of rotating disks. A review has been given by Omprakash and Ramamurti (1988). In the previous work, Kirchhoff-Love thin-plate theory is often used. The theory neglects the shear deformation and rotary inertia effects and, therefore, is limited to modelling thin disks. A recent study conducted by Sinha (1987) has used the Mindlin's plate theory to account for both effects in the analysis.

It has been recognized that the dynamic behavior of a bladed-disk system can not be predicted accurately without considering the coupling effect between the blades and the disk. Although a few studies have been conducted on the vibration analysis of nonrotating bladed-disks, little research has been reported on the vibration analysis of rotating bladed-disks. Recently, after reviewing previous research, Omprakash and Ramamurti (1990a) studied the coupled vibration characteristics of rotating bladed-disk systems.

Transient Dynamic Analysis

Although transient dynamic analysis is important in predicting the responses of rotating bladed-disk assemblies during events such as start-up, blade tip rubbing, speed changes, and traversing through system critical speeds, it has received little attention. Using an extended beam theory, Irretier (1985) performed a spectral analysis to simulate the run-up of a turbine blade subjected to partial admission. Davis (1989) implemented analysis tools in an existing finite element program to address nonlinear transient analysis of rotating bladed-disk-shaft systems subjected to blade rubbing. Due to the large amount of computation involved, only bladed-disk models with coarse meshes were studied in the sample analyses.

Recently, the transient characteristics of a bladed-disk during run-up were studied by Omprakash and Ramamurti (1990b) in a spectral analysis using the finite element method. A three-noded triangular shell element was used to model the bladed-disk. The cyclic symmetry and modal superposition approaches were used to reduce computational burden. The spectral analysis employed the lowest three frequencies that are interpolated quadratically at each time step from the frequencies computed by actual eigensolution at selected rotational speeds.

2.5.2 Present Approach

As discussed in Section 2.2, the present work uses twenty-node brick finite elements with reduced integration to model bladed-disk systems (although both eight- and twenty-node brick finite elements with either full or reduced integration have been implemented in ABREAST). The quadratic displacement model provided by the element goes beyond those

provided by Mindlin/Timoshenko theory and implicitly takes into account effects of shear deformation and rotary inertia. However, it should be noted that numerical ill-conditioning may occur if the aspect ratio of the element is too large, e.g., exceeding about 50. Such ill-conditioning has been encountered by the writer in cases where coarse meshes are used to model very thin blades or disks.

The finite element formulation presented in Section 2.3 has been implemented in all analysis modules of ABREAST for the brick finite elements. For rotational dynamics, the current implementation uses the rotating x-y-z coordinate system shown in Fig. 2.5 as the global coordinate system. The current implementation also assumes that the time-varying rotational-velocity vector $\{\omega\}$ is the multiplication of a time-varying scalar and a time-independent reference vector, i.e., the rotational acceleration vector is colinear with the rotational velocity vector. The user is allowed to specify in the analysis input file a) the type of formulation for accounting for rotational nonlinearity (i.e., consistent- or lumped-mass), b) the reference vector of rotational velocity in terms of three components in the global coordinates (with the assumption that the vector passes through the origin of the global coordinate system), and c) the name of the history file in which the time history of the scalar of rotational velocity is defined.

Modal Vibration Analysis

A two-stage analysis is employed in the present work to carry out modal vibration analyses of rotating bladed-disk systems. The first stage involves a nonlinear static analysis to obtain a steady-state solution serving as the initial condition for the second-stage eigenvalue analysis.

For the steady-state solution, a static analysis capability, which did not previously exist in ABREAST, has been implemented. The solution schemes implemented include simple incremental, Newton-Raphson iterative incremental, and Modified-Newton iterative incremental schemes. Both Gauss elimination and preconditioned conjugate gradient equation solvers are available. Since the steady-state solution of a rotating bladed-disk system involves both rotational and geometric nonlinearities (as shown in Eq. 2.32), the present work uses either the Newton-Raphson or the Modified-Newton iterative incremental scheme to ensure the satisfaction of equilibrium. In addition, the load vector (i.e., the right hand side of Eq. 2.32) is nonlinear and displacement-dependent. The current approach updates the load vector at the end of each load increment after equilibrium iteration is completed. At the end of the final increment, a recursive procedure that updates the load vector and then performs equilibrium iterations is used until the update of the load vector no longer affects the equilibrium. Upon completion of the analysis, an initial condition file, which contains the results for the steady-state nonlinear analysis, is created for the second-stage vibration analysis.

The second-stage vibration analysis requires the solution of Eq. (2.33). The system stiffness is computed based on the final equilibrium state of the system obtained from the first-stage analysis. The present research uses the subspace iteration algorithm in ABREAST to solve for a desired number of low modes of vibration.

Transient Dynamic Analysis

Both the explicit central difference and implicit Newmark integration methods in ABREAST have been extended to account for rotational dynamics for the brick elements. In the present central difference analysis for rotational dynamics, since the Coriolis damping matrix, $[C_c]$, in Eq. (2.31) is nondiagonal but skew-symmetric, the velocity and acceleration are approximated by Eq. (2.41) and Eq. (2.39), respectively. Substitution of Eqs. (2.41) and (2.39) into Eq. (2.31) results in

$$\begin{aligned}(1/\Delta t^2)[M]\,{}^{t+\Delta t}\{Q\} = {}^{t}\{F^{ext}\} - {}^{t}_{t}\{F\} - {}^{t}_{t}\{F_e\} - ({}^{t}_{t}[K_a] - {}^{t}_{t}[K_r])\,{}^{t}\{Q\} \\ + ((2/\Delta t^2)[M] - (1/\Delta t)({}^{t}_{t}[C_v] + {}^{t}_{t}[C_c]))\,{}^{t}\{Q\} \\ - ((1/\Delta t^2)[M] - (1/\Delta t)\,({}^{t}_{t}[C_v] + {}^{t}_{t}[C_c]\,))\,{}^{t-\Delta t}\{Q\} \end{aligned} \tag{2.48}$$

The diagonal mass matrix is used to avoid solution of simultaneous equations. In the present research, the explicit central difference method is used for dynamic analysis of rotating bladed-disk assemblies experiencing tip rubs, which is a short-duration dynamic problem. In this case, similar to the modal vibration analysis discussed earlier, a two-stage analysis is employed to save computational time. In the first stage, a nonlinear static analysis is performed to obtain a steady-state solution of the system at a given rotational speed. This solution then serves as the initial condition for the second-stage transient analysis which computes system responses during tip rubbing.

In the implicit Newmark analysis for rotational dynamics, the use of the constant average acceleration scheme results in the incremental equilibrium equation of the same form as Eq. (2.45)

$$ {}^{t}[\hat{K}]\,\{\Delta Q\}^{i} = {}^{t+\Delta t}\{\hat{R}\} $$

in which

$$ {}^{t}[\hat{K}] = (1/(\alpha\Delta t^2))[M] + (\delta/(\alpha\Delta t))\,{}^{t}_{t}[C] + {}^{t}_{t}[K] \tag{2.49} $$

$$ \begin{aligned} {}^{t+\Delta t}\{\hat{R}\} = {}& {}^{t+\Delta t}\{F^{ext}\} - {}^{t+\Delta t}_{t+\Delta t}\{F\}^{i-1} - {}^{t}_{t}\{F_e\} - ({}^{t}_{t}[K_a] - {}^{t}_{t}[K_r])\,{}^{t}\{Q\} \\ & - [M]((1/(\alpha\Delta t^2))({}^{t+\Delta t}\{Q\}^{i-1} - {}^{t}\{Q\}) - (1/(\alpha\Delta t))\,{}^{t}\{\dot{Q}\} - ((1/2 - \alpha)/\alpha)\,{}^{t}\{\ddot{Q}\}) \\ & - {}^{t}_{t}[C]((\delta/(\alpha\Delta t))({}^{t+\Delta t}\{Q\}^{i-1} - {}^{t}\{Q\}) - (\delta/\alpha + 1)\,{}^{t}\{\dot{Q}\} - (\delta/\alpha + 1)\Delta t\,{}^{t}\{\ddot{Q}\}) \end{aligned} \tag{2.50} $$

$$ {}^{t}_{t}[C] = {}^{t}_{t}[C_v] + {}^{t}_{t}[C_c] \tag{2.51} $$

$$ {}^{t}_{t}[K] = {}^{t}_{t}[K_e] + {}^{t}_{t}[K_g] + {}^{t}_{t}[K_a] - {}^{t}_{t}[K_r] \tag{2.52} $$

Since ${}^{t}_{t}[C_c]$ and ${}^{t}_{t}[K_a]$ are skew-symmetric, the effective stiffness matrix ${}^{t}[\hat{K}]$ is no longer symmetric, resulting in a significant increase of storage requirement for ${}^{t}[\hat{K}]$. To avoid this problem, the current implementation replaces Eq. (2.49) by

$$ {}^{t}[\hat{K}] = (1/(\alpha\Delta t^2))[M] + (\delta/(\alpha\Delta t))\,{}^{t}_{t}[C_v] + {}^{t}_{t}[K_e] + {}^{t}_{t}[K_g] - {}^{t}_{t}[K_r] \tag{2.53} $$

Since the current approach uses a modified-Newton iterative solution scheme, the use of Eq. (2.53) does not change the final equilibrium solution although the convergence rate may be affected. The implicit Newmark method is suited to long-duration dynamic problems such as dynamic analyses of the run-up of rotating bladed-disk systems.

2.5.3 Verification and Comparative Studies

As discussed earlier, very little research on transient dynamic analysis of rotating bladed-disk assemblies has been conducted using direct

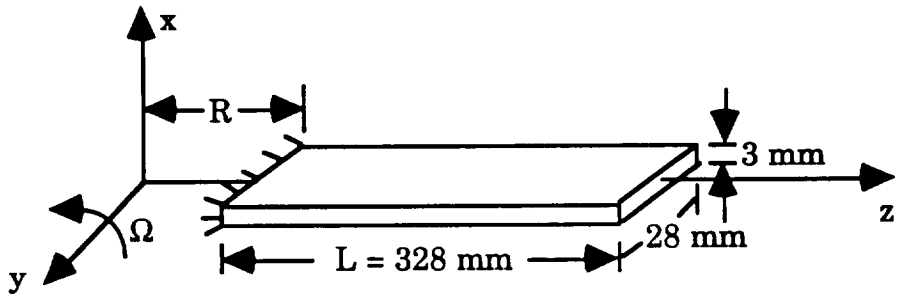
time integration solution methods. None of the numerical results of which the writer is aware provides sufficient information which can be used in the present implementation to reproduce the results adequately. Therefore, the present research relies mainly on using some modal vibration results of rotating beams and plates published in the literature to verify the implementation in ABREAST for analysis of rotating turbine bladed-disk assemblies. Both steady-state and vibration analysis capabilities implemented are directly verified. The transient dynamic analysis capabilities implemented are verified indirectly and partially because many portions of them share the same routines with the steady-state or vibration analysis modules (for example, routines for mass matrix formation and assembly, stiffness matrix formation and assembly, and stress recovery). In addition to verification studies, numerical comparisons are conducted between the consistent mass approach and the lumped mass approach for accounting for rotational nonlinearities.

Five example problems are used here for verification and comparative studies. They are (a) the in-plane vibration of a rotating cantilever beam, (b) the out-of-plane vibration of a rotating cantilever beam, (c) the vibration of a rotating tapered cantilever beam, (d) the vibration of a rotating annular plate, and (e) the transient response of a rotating cantilever beam subjected to an impact load (for comparative study only). The numerical results from ABREAST are compared with those published in the literature. All analyses performed in the present research use (i) twenty-node brick elements with reduced integration, (ii) a lumped mass matrix formulated using the HRZ lumping scheme with full integration (Cook et al. 1989), and (iii) a Newton-Raphson iterative method for steady-state solutions, followed

by either a subspace iteration method (Lin 1980) for frequency computations or a central difference method for time integration of transient responses. The present results obtained using the consistent mass approach for taking into account rotational nonlinearities are denoted as Present-CM, while those obtained using the lumped mass approach are denoted as Present-LM.

In-plane Vibration of a Rotating Cantilever Beam

The in-plane vibrations of the rotating cantilever beam shown in Fig. 2.7 are studied in the present research over a wide range of rotational speeds. An 1 x 2 x 8 mesh is used in the study. (Note that l x m x n denotes the number of elements in the x, y, and z directions, respectively.) In Table 2.4, the present results are compared with those obtained by Putter and



$E = 2.17 \times 10^{11} N/m^2$ $\rho = 7850\ kg/m^3$ $\nu = 0.3$

Figure 2.7 A rotating cantilever beam for in-plane vibration analysis

Manor (1978) and Yokoyama (1988). Putter and Manor used five beam elements with a fifth-order displacement function and considered effects of shearing force and rotary inertia, while Yokoyama used eight Euler-Bernoulli beams with a cubic displacement function. The first and second normalized frequencies, $\omega(\rho AL^4/EI)^{1/2}$, of vibrations in the x-z plane are compared for different values of normalized rotational speeds,

$\Omega(\rho AL^4/EI)^{1/2}$. (Note that ω is the circular frequency of vibration; ρ is the mass density; A is the area of cross-section; L is the length; E is the Young's modulus of elasticity; I is the moment of inertia; and Ω is the rotational speed.)

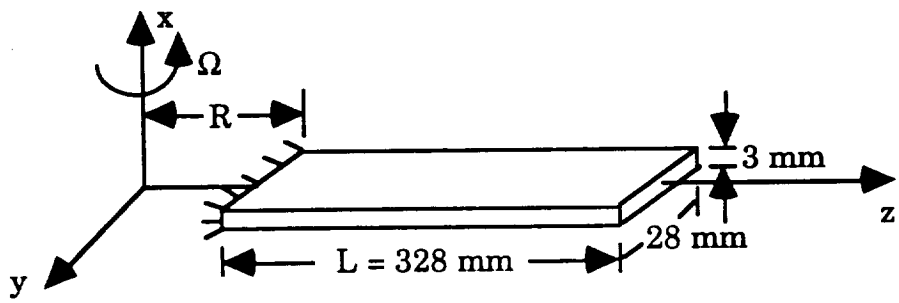
Table 2.4 Comparison of in-plane normalized frequencies, $\omega(\rho AL^4/EI)^{1/2}$ of a rotating cantilever beam obtained by the present study with those reported by Putter and Manor (1978) and Yokoyama (1988)

$\Omega(\rho AL^4/EI)^{1/2}$ [Ω in rpm]	R/L	0		1	
	mode	1st	2nd	1st	2nd
2.0 [809]	Present-LM	3.68	22.59	4.45	23.33
	Present-CM	3.68	22.59	4.45	23.31
	Putter and Manor	3.61	22.53	4.40	23.28
	Yokoyama	3.62	22.53	4.40	23.28
5.0 [2,022]	Present-LM	4.16	24.97	7.47	28.85
	Present-CM	4.16	24.94	7.46	28.78
	Putter and Manor	4.07	24.95	7.41	28.92
	Yokoyama	4.07	24.95	7.41	28.93
10.0 [4,044]	Present-LM	5.22	32.03	13.34	42.89
	Present-CM	5.22	31.92	13.32	42.67
	Putter and Manor	5.05	32.12	13.26	43.23
	Yokoyama	5.05	32.12	13.26	43.24
20.0 [8,088]	Present-LM	7.17	51.01	25.43	72.08
	Present-CM	7.17	50.66	25.39	72.68
	Putter and Manor	6.78	51.35	25.29	76.59
	Yokoyama	6.79	51.37	25.32	76.66
50.0 [20,220]	Present-LM	12.32	115.25	62.47	179.65
	Present-CM	12.44	113.83	62.50	176.39
	Putter and Manor	10.48	116.20	61.64	181.94
	Yokoyama	10.90	116.42	61.88	182.39

Both Present-CM and Present-LM results agree well with those of Putter and Manor and of Yokoyama except the first frequencies when R/L = 0 and $\Omega(\rho AL^4/EI)^{1/2}$ = 20, 50. It is believed that the disagreements are mainly due to the different approaches used for taking into account centrifugal forces between the present work and those of Putter and Yokoyama. Both Putter and Yokoyama compute the centrifugal forces based on the undeformed beam configuration while the present research accounts for the deformed beam configuration. As a result, the differences become more significant as the rotational speed increases (i.e., deformation of the beam increases). It can be seen that all of the present first-mode results are higher than those of Putter and Manor, and Yokoyama. Another observation is that in all but two cases, the results of Present-LM are either equal to or slightly higher than those of Present-CM.

Out-of-plane Vibration of Rotating Cantilever Beams

The second example studied is the out-of-plane vibrations of the rotating cantilever beam shown in Fig. 2.8 over a range of rotational speeds.



$E = 2.17 \times 10^{11} N/m^2$ $\rho = 7850\ kg/m^3$ $\nu = 0.3$

Figure 2.8 A rotating cantilever beam for out-of-plane vibration analysis

This is the same beam used in the previous example (Fig. 2.7) but with a different rotational axis. An 1 x 2 x 8 mesh is used in the study. (Note that 1 x m x n denotes the number of elements in the x, y, and z directions, respectively.) In Table 2.5, the present results are compared with those obtained by Yokoyama (1988) using eight Euler-Bernoulli beam elements with a cubic displacement function. The first and second normalized frequencies, $\omega(\rho AL^4/EI)^{1/2}$, of vibrations in the x-z plane are compared for different values of normalized rotational speeds, $\Omega(\rho AL^4/EI)^{1/2}$. (Note that notations used here are the same as those in the previous example.)

Table 2.5 Comparison of out-of-plane normalized frequencies, $\omega(\rho AL^4/EI)^{1/2}$ of a rotating cantilever beam obtained by the present study with those reported by Yokoyama (1988)

$\Omega(\rho AL^4/EI)^{1/2}$ [Ω in rpm]	R/L	0		1	
	mode	1st	2nd	1st	2nd
2.0 [809]	Present-LM	4.19	22.68	4.88	23.41
	Present-CM	4.18	22.67	4.87	23.40
	Yokoyama	4.14	22.62	4.83	23.37
4.0 [1,618]	Present-LM	5.64	24.31	7.52	26.93
	Present-CM	5.63	24.28	7.51	26.87
	Yokoyama	5.59	24.28	7.48	26.96
6.0 [2,427]	Present-LM	7.42	26.81	10.50	31.91
	Present-CM	7.40	26.74	10.47	31.79
	Yokoyama	7.36	26.81	10.44	32.03
8.0 [3,236]	Present-LM	9.33	29.96	13.57	37.73
	Present-CM	9.30	29.84	13.54	37.55
	Yokoyama	9.26	30.00	13.51	37.96
10.0 [4,044]	Present-LM	11.28	33.56	16.67	44.04
	Present-CM	11.25	33.39	16.64	43.77
	Yokoyama	11.20	33.64	16.61	44.38

Both Present-CM and Present-LM results agree closely with those of Yokoyama. It is observed again that the results of Present-LM are slightly higher than those of Present-CM in all cases.

Vibration of a Rotating Tapered Cantilever Beam

The frequencies of rotating tapered beams with different taper ratios, boundary conditions, and rotational speeds were studied by Khulief and Bazoune (1992) using a tapered Timoshenko beam element. In the present research, the bending frequencies of the particular rotating tapered beam shown in Fig. 2.9 are computed. An 8 x 2 x 2 mesh is used in the analysis. (Note that l x m x n denotes the number of elements in the x, y, and z directions, respectively.) In Table 2.6, the present results are then compared with those obtained by Khulief and Bazoune using sixteen tapered Timoshenko beam elements. The first and second normalized frequencies, $\omega(\rho A_0 L^4/EI_0)^{1/2}$, of vibrations in the x-y plane are compared for different values of normalized rotational speeds, $\Omega(\rho A_0 L^4/EI_0)^{1/2}$. (Note

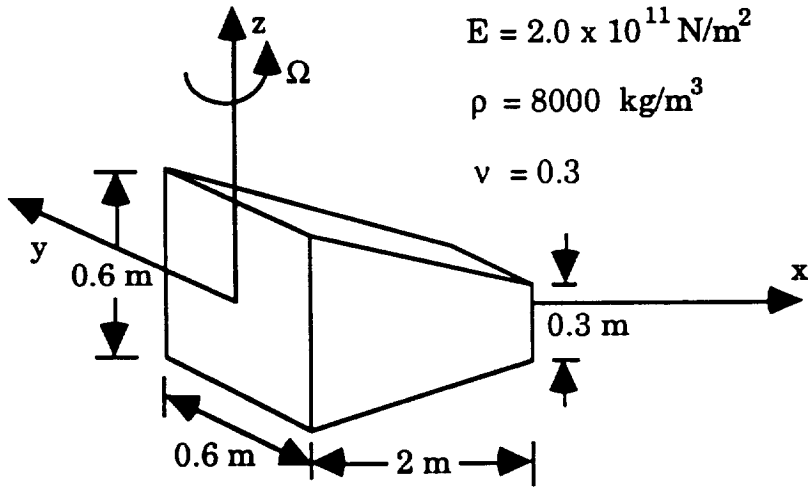


Figure 2.9 A rotating tapered beam for vibration analysis

that ω is the circular frequency of vibration; ρ is the mass density; A_0 is the area of cross-section at x = 0; L is the length; E is the Young's modulus of elasticity; I_0 is the moment of inertia at x = 0; and Ω is the rotational speed.)

It can be seen that both Present-CM and Present-LM results are in good agreement with those of Khulief and Bazoune. The results of Present-LM are either equal to or slightly lower than those of Present-CM for the first mode while the trend is reversed for the second mode.

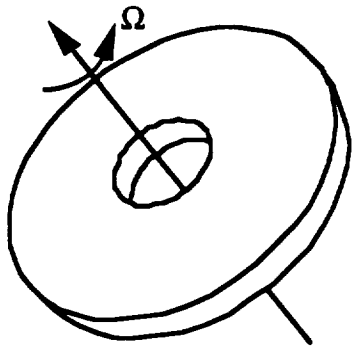
Table 2.6 Comparison of normalized frequencies, $\omega(\rho A_0 L^4/EI_0)^{1/2}$ of a rotating tapered beam obtained by the present study with those reported by Khulief and Bazoune (1992)

mode	$\Omega(\rho A_0 L^4/EI_0)^{1/2}$ [Ω in rpm]	0 [0]	1 [2,924]	3 [8,772]	5 [14,619]
1st	Present-CM	4.37	4.41	4.68	5.26
	Present-LM	4.37	4.40	4.68	5.23
	Khulief & Bazoune	4.38	4.43	4.83	5.55
2nd	Present-CM	15.41	15.52	16.46	18.35
	Present-LM	15.41	15.52	16.48	18.37
	Khulief & Bazoune	15.98	16.04	16.48	17.34

Vibration of a Rotating Annular Plate

Figure 2.10 shows a rotating annular plate which is clamped at the inner edge while free at the outer edge. The frequencies of the rotating plate are computed in the present research using 3-D twenty-node brick elements with a 4 x 12 x 1 mesh as shown in Fig. 2.3(a). In Table 2.7, the present results are compared with those obtained by Sinha (1987) who used Mindlin's plate theory and a modified Rayleigh-Ritz method with a

numerical trial function. The nondimensional frequencies, $\omega(\rho h r_0^4/D)^{1/2}$, are compared for different values of nondimensional angular velocities of rotation, $\Omega(\rho h r_0^4/8D)^{1/2}$. (Note that ω is the circular frequency of vibration; is the mass density; h is the plate thickness; r_0 is the inner radius; Ω is the angular velocity of rotation; and $D = (Eh^3)/[12(1-\nu^2)]$ where E and ν are Young's modulus of elasticity and Poisson's ratio, respectively.)



inner radius = 0.25 m
outer radius = 1.0 m
thickness = 0.2 m
clamped at the inner edge
free at the outer edge

$E = 196 \times 10^9$ N/m^2
$\nu = 0.3$
$\rho = 7800$ Kg/m^3

Figure 2.10 A rotating annular plate for vibration analysis

Again, both Present-CM and Present-LM results agree well with those of Sinha. The results of Present-LM are either equal to or slightly higher than those of Present-CM for all three modes.

Transient Response of a Rotating Cantilever Beam Subjected to an Impact Load

The transient responses of the rotating cantilever beam shown in Fig. 2.7 (in this example, R = 0 and Ω = 4,000 rpm) subjected to an impact load are studied for comparison of the consistent mass and lumped mass approaches for a transient dynamic problem. The impact load is a uniform traction in the -x direction applied on the face at the free end of the beam and has the history shown in Fig. 2.11. The mesh used in the study is

shown in Fig. 2.12. The transient responses are computed for a duration of 0.0039 sec. and a time step of 0.0000001 sec. is used. The results are output every 0.000005 sec.

Table 2.7 Comparison of nondimensional frequencies, $\omega(\rho h r_0^4/D)^{1/2}$ of a rotating annular plate obtained by the present study with those reported by Sinha (1987)

nodal circles	nodal diameters	$\Omega(\rho h r_0^4/D)^{1/2}$ [Ω in rpm]	0 [0]	2 [16,387]	4 [32,773]
0	0	Present-CM	5.54	7.80	12.66
		Present-LM	5.54	7.92	12.79
		Sinha	5.56	7.87	12.33
0	1	Present-CM	5.08	8.03	13.96
		Present-LM	5.08	8.16	14.13
		Sinha	5.12	8.26	13.87
0	2	Present-CM	6.21	10.07	17.64
		Present-LM	6.21	10.23	18.04
		Sinha	6.30	10.25	17.13

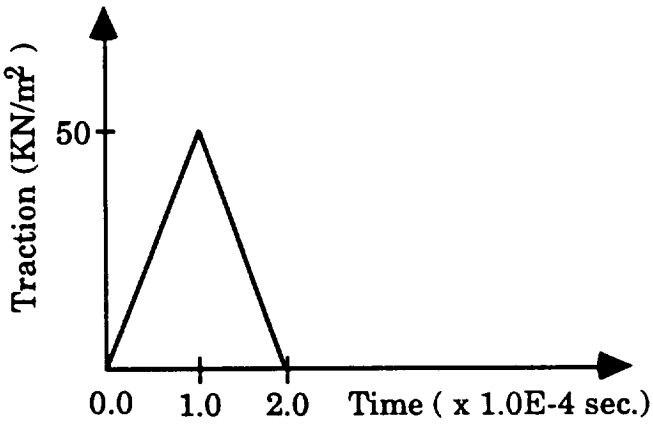


Figure 2.11 History of the impact load applied to the rotating cantilever beam of Fig. 2.7

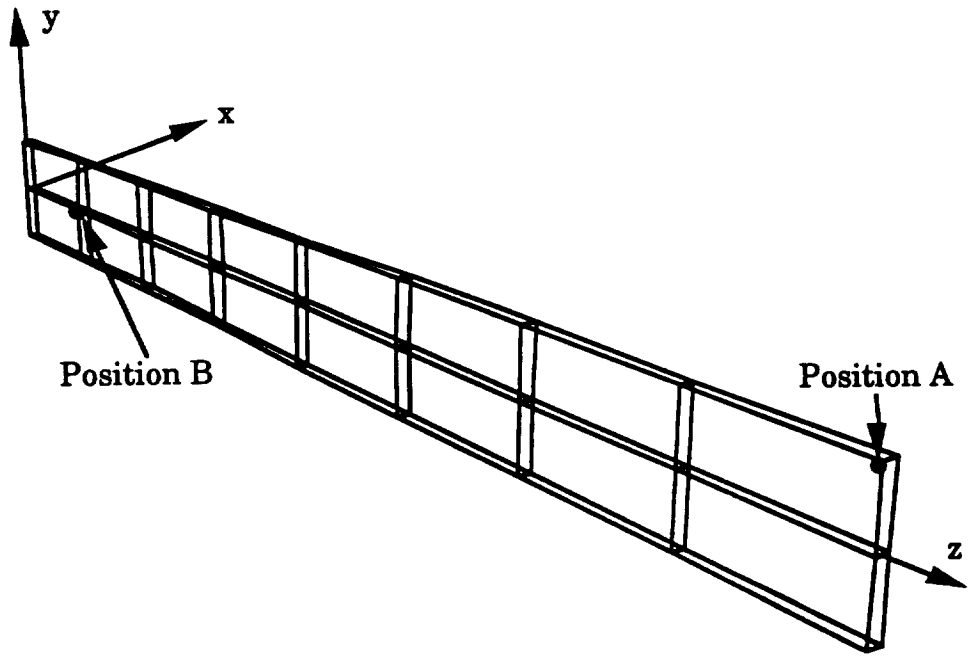


Figure 2.12 Finite element mesh used in the transient analysis of the rotating cantilever beam of Fig. 2.7

The transient displacements at two different locations of the rotating beam are monitored in the present study (see positions A and B in Fig. 2.12). In Figs. 2.13 - 2.18, the Present-LM results are plotted against the Present-CM results. It can be seen that the Present-LM results agree closely with the Present-CM results in Figs. 2.13 and 2.16. In Figs. 2.14, 2.15, and 2.18, the differences between the Present-LM results and the Present-CM results mainly come from different displacements obtained at the end of the steady-state analysis between the Present-LM and the Present-CM approaches, despite the fact that the same tolerance ($1x10^{-8}$ in this example) is used for Newton-Raphson equilibrium iterations in both cases. The transient characteristics of the Present-LM results are in good agreement with those of the Present-CM results. In Fig. 2.17, it should be

noted that the displacements are so small that the differences between the Present-LM results and the Present-CM results may be neglected.

2.5.4 Closure

Although the lumped mass approach for taking into account rotational nonlinearities is not expected to be as accurate as the consistent mass approach due to its neglect of mass coupling, the modal vibration results obtained using the lumped mass approach in all of the examples studied in Section 2.5.3 are in close agreement with those obtained using the consistent mass approach. In most cases, the frequency results from the lumped mass approach are slightly higher than those from the consistent mass approach. In addition, the transient displacements predicted using the lumped mass approach in the example studied agree well with those predicted using the consistent mass approach.

On the other hand, the lumped mass approach is expected to be computationally more efficient than the consistent mass approach. Although this is usually the case in all of the modal analysis examples studied in Section 2.5.3, the computational times for modal analyses using these two different approaches differ by only about five to ten percent. This is probably due to the fact that the calculation of nonlinear rotational forces is not the major computational cost when compared with the stiffness matrix formation, equation solving, and eigensolution in modal vibration analyses. However, the computational time for transient analysis using the consistent mass approach is about 8.7 times larger than that using the lumped mass approach in the example studied.

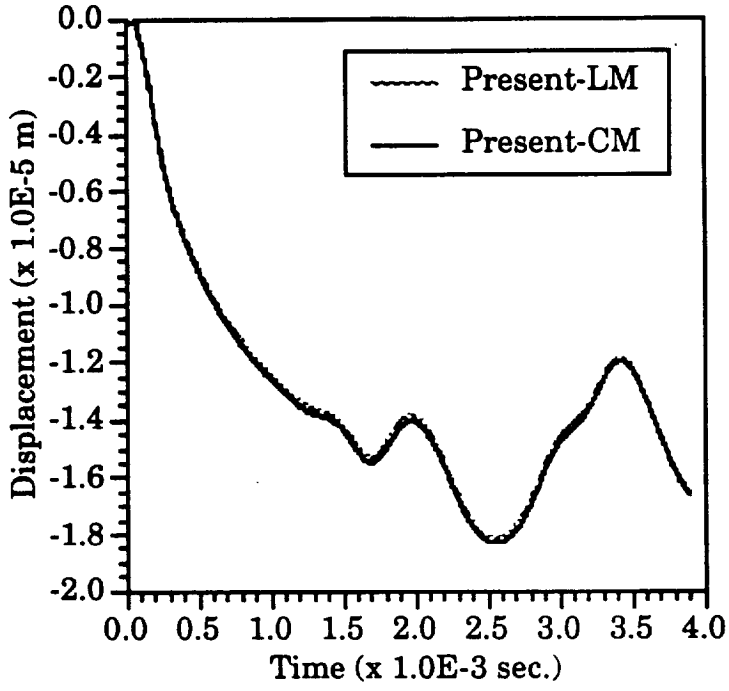


Figure 2.13 Transient displacements in the x direction at position A of the rotating beam in Fig. 2.12

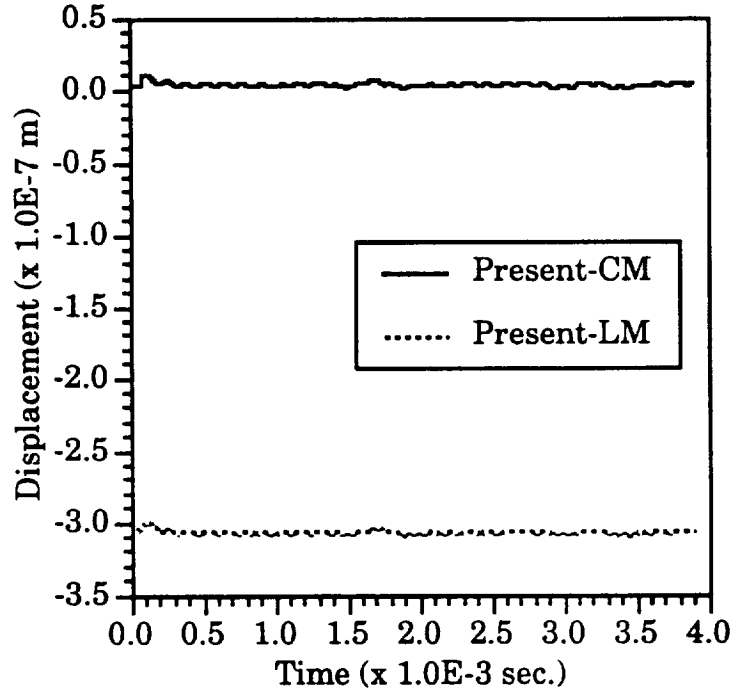


Figure 2.14 Transient displacements in the y direction at position A of the rotating beam in Fig. 2.12

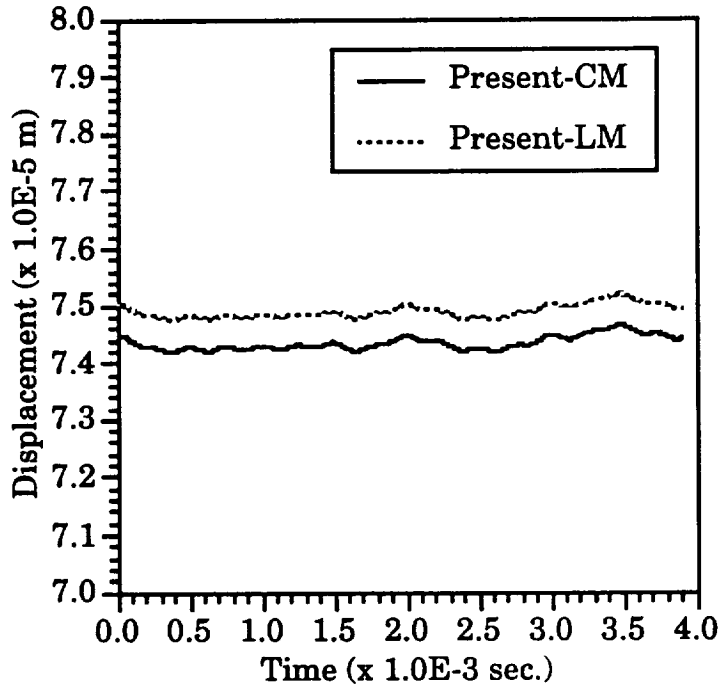


Figure 2.15 Transient displacements in the z direction at position A of the rotating beam in Fig. 2.12

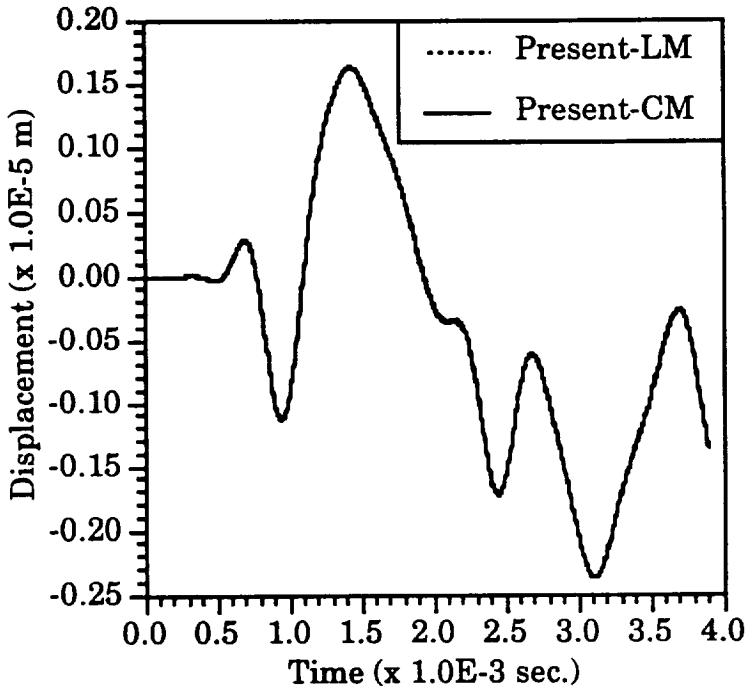


Figure 2.16 Transient displacements in the x direction at position B of the rotating beam in Fig. 2.12

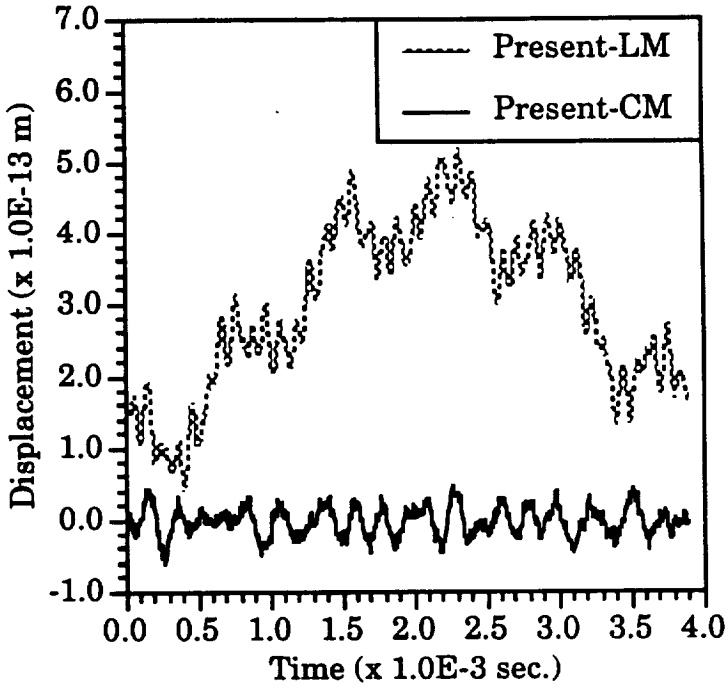


Figure 2.17 Transient displacements in the y-direction at position B of the rotating beam in Fig. 2.12

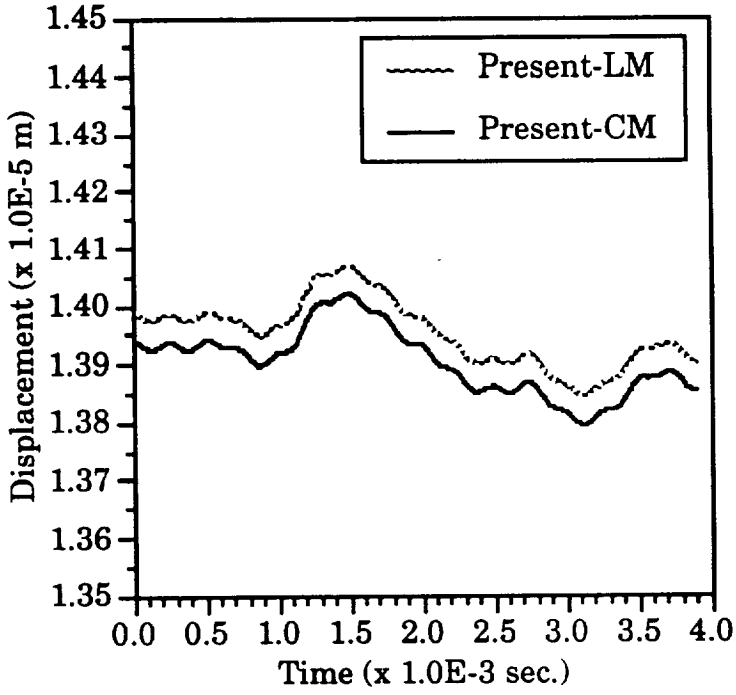


Figure 2.18 Transient displacements in the z direction at position B of the rotating beam in Fig. 2.12

Chapter 3

Parallel Nonlinear Solution Algorithms

The parallel processing strategies investigated in the present work include both numerical algorithms for parallel nonlinear solutions and techniques to effect load-balancing among processors. This chapter discusses the parallel solution algorithms for transient dynamic analysis as well as steady-state analysis. The load-balancing techniques based on domain partitioning are explored in the next chapter.

The realistic computer simulation of nonlinear structural dynamics of large finite element systems often requires a significant amount of computing time and memory. One possible way to reduce the elapsed wall clock time needed for an analysis is the use of parallel processing which divides the computational work among several processors running concurrently. For large structural problems, the analysis may often requires the use of virtual memory which usually slows down the analysis significantly. With the workload divided among processors in the parallel analysis, the computation may be performed in each processor without the use of virtual memory, resulting in additional time saving.

The present research investigates the parallelization of time integration algorithms and equation solvers for the solution of the governing equilibrium equations of structural dynamics discussed in Chapter 2. Focus is placed on parallel algorithms suitable for the present coarse-grained, message passing environment where, as mentioned in Chapter 1 and will be

discussed further in Section 3.1, minimization of the interprocess communication and parallelization at the substructure level are two key strategies for the algorithms to achieve good parallel performance.

In this chapter, the parallel computing environment used is described first. Then, a brief review of previous research on parallel solution algorithms is given. The parallel solution algorithms investigated in the present work are discussed. Numerical studies are also performed to evaluate the effectiveness of the present parallel solution algorithms.

3.1 Parallel Computing Environment

The computing environment for parallel processing in this work consists of either up to six DECsystem 5000's or up to twelve Apollo/HP 9000 series 720's. Each DECsystem 5000 has 16 Mbytes of memory and a central processor running at the speed of approximately 24 Mips, while each Apollo/HP 720 has 128 Mbytes of memory and a central processor running at the speed of approximately 59 Mips. The DEC workstations utilize the ULTRIX operating system (DEC's version of UNIX), while the HP workstations utilize the HP-UX operating system (HP's version of UNIX). They are connected by Ethernet and communicate via the TCP/IP (Transmission Control Protocol/Internet Protocol) and DECnet protocol (for DEC workstations only). Any processor may communicate directly with any of the other processors. The approximate realizable communication speed of the Ethernet ranges from 0.1 to 1.0 Mbyte per second.

The communication and synchronization between processors are achieved in a message passing environment provided by ISIS (Birman et al. 1990), a UNIX-based parallel application manager developed in the

Computer Science Department at Cornell University. Two of the most basic facilities that ISIS provides for interprocess communication are process-group and broadcast mechanisms. The process-group mechanism provides a means of grouping and naming processes as a unit (note that a process can be a member of more than one process group). The broadcast mechanism allows a process to send a message to a process group and to reply to or forward a message received. Another important feature provided by ISIS is a programming paradigm called virtual synchrony. Virtual synchrony ensures that all members of a process group receive the messages sent to the group in the same order as they were sent but not necessarily at the same time. This approach avoids the difficulty and cost involved in maintaining real synchrony but still provides sufficient synchronicity for parallel applications. A research version of ISIS is in the public domain (available via anonymous FTP through the Internet from ftp.cs.cornell.edu) and has been distributed and used by hundreds of research institutes and universities. The latest version is now sold commercially.

The parallel computing environment presented above is a coarse-grained, distributed-memory environment where the number of processors used is small and no global memory is shared among processors. In the parallel paradigm, this environment is usually classified as a multiple-instruction, multiple-data (MIMD) environment where each processor, with a different set of data, does not necessarily execute the same instruction simultaneously. It should be noted that the parallel processing strategies discussed in this chapter are not limited to the networked workstation environment investigated in this work, but instead are suitable for a variety

of machine environments which share the aforementioned features. The major advantages of this type of environments are their

a) extendibility: it is easy to add more processors to increase processing power. In addition, as the computer technology advances, the processors as well as network interfaces in the environment can be upgraded gradually and in a relatively less expensive fashion.

b) heterogeneity: computers with different features can be put together to meet different computational needs and to maximize the utilization of computer resources. For example, the parallel simulation of structural dynamics may need a workstation with high-performance graphics for visualization and a number of fast processors for parallel computations.

In the present networked workstation environment, the communication among processors is the major bottleneck for parallel applications due to slow communication speed of Ethernet. Although the replacement of the Ethernet by a faster FDDI network will provide improvement in the near future (as mentioned in the Chapter 1), it is believed that the interprocess communication is still the major bottleneck because the speed of processors is currently advancing in a faster rate than that of communication network. Therefore, this research considers parallel algorithms which minimize communication overhead. In addition, the parallelization of finite element computations in this type of coarse-grained environment is usually best devised at the substructure (or subdomain) level (as opposed to the parallelization at the element level or at the degree-of-freedom level as in a fine-grained environment).

3.2 Review of Previous Research

Previous research on both parallel time integration algorithms and parallel equation solvers is briefly reviewed in the following subsections.

3.2.1 Parallel Time Integration Algorithms

Considerable research has been conducted on various time integration algorithms to take advantage of parallel processing in transient dynamic analysis. At Cornell University, a preliminary study on a number of time integration algorithms had been performed to evaluate their potential in parallel analysis (Hajjar 1987, 1988). In this subsection, existing time integration algorithms are briefly re-evaluated for parallel processing based on the preliminary studies at Cornell and more recent research work published in the literature.

Explicit algorithms

Among explicit algorithms, the central difference method (see, for example, Bathe 1982; Cook et al. 1989) is probably the most commonly used. The central difference algorithm is a conditionally stable explicit time integration method inherently amenable to parallel processing. With standard selections of the finite difference relations and the use of lumped masses, the solution may proceed on a degree-of-freedom level without assembly of the global stiffness matrix and solution of simultaneous equations. In addition, groups of degrees of freedom may be readily apportioned to different processors by substructuring. Finally, with appropriate preparation of substructure data for each processor, a minimum

amount of interprocess communication may be achieved, and this communication needs to occur only between adjacent processors.

Malone (1988, 1990) formulated a parallel central difference algorithm which does not duplicate computational work among processors, but requires interprocess communication in the computation of nodal quantities at nodes shared by two or more processors. The hardware configuration considered by Malone was a 32-processor Intel iPSC/d5 hypercube. In a networked workstation environment, Hajjar and Abel (1989a) presented a parallel implementation of the central difference algorithm which minimizes interprocess communication at the expense of duplicating the computation of element quantities in elements shared by two or more processors. Chiang and Fulton (1990) also investigated central difference method for two different parallel computers, the FLEX/32 shared memory multicomputer and the Intel iPSC Hypercube local memory computer. All of the above studies show good parallel performance of the central difference algorithm.

Implicit algorithms

Implicit algorithms (see, for example, Bathe 1982; Owen 1980) may be formulated to be unconditionally stable. Compared to explicit methods, these methods allow a larger time step size to be selected based on accuracy requirements exclusively. However, the solution of a set of simultaneous equations is required at each time step, making the implementation of the procedure in parallel more difficult than explicit methods. The parallel simultaneous equation solution also requires significant amount of interprocess communication, both nearest-neighbor and global.

Hajjar and Abel (1988) used the implicit Newmark-β constant average acceleration algorithm with domain decomposition for the parallel solution of nonlinear structural dynamics in a networked workstation environment. The domain decomposition strategy employed uses substructuring techniques and a preconditioned conjugate gradient (PCG) algorithm for the iterative solution of the reduced set of unknowns along the substructure interfaces. The PCG algorithm seems attractive for parallel processing because it requires less interprocess communication and is easier to balance the workload among processors than direct methods. Chiang and Fulton (1990) investigated implicit Newmark type methods with a skyline Cholesky decomposition strategy for the FLEX/32 shared memory multicomputer and the Intel iPSC Hypercube local memory computer. It was shown that the shared database nature of the decomposition algorithm made the FLEX/32 multicomputer a more efficient parallel environment than the Hypercube computer.

Mixed-Time Integration Algorithms

Mixed-time integration algorithms use simultaneously different time integration methods with different time steps in different domains of the problem to minimize the computational cost. These methods are suitable for problems consisting of definitive domains of different stiffness properties, such as soil-structure interaction problems.

A variety of implicit-explicit methods have been proposed. A review of these methods was given by Hughes and Belytschko (1983) and Liu (1987). Since the solution of a reduced set of simultaneous equations is still required in the implicit integration and different time steps are used in

different integration methods, parallel implementation of these methods are more challenging than that of implicit methods, especially in the area of balancing the computational loads among processors. Liu (1987) discussed some aspects of parallel implementation of these methods in a shared memory, parallel processing environment.

Explicit-explicit subcycling (Liu and Belytschko 1982; Liu 1987) is also one type of mixed-time integration methods. These methods use only explicit integration with different time steps selected for different domains of the problem without the solution of simultaneous equations. Therefore, they are more amenable to parallel processing than implicit-explicit methods. However, with the use of different time steps in different domains, their parallel implementation faces challenging tasks for load-balancing and synchronization among processors to achieve desirable efficiency, especially in a distributed memory environment.

Semi-Implicit Algorithms

Semi-implicit algorithms first presented by Trujillo (1977) for structural dynamics are similar to an operator splitting algorithm, termed the left-right (or right-left) technique (Saul'yev 1964), developed in finite difference analysis. These methods take the form of implicit time integration algorithms and split the stiffness and damping matrices into strictly lower and upper triangular matrices. Both symmetric and unsymmetric splitting algorithms have been presented. With the use of a diagonal mass matrix, the solution of a set of simultaneous equations is avoided and only a back substitution or a forward reduction is required at each time step.

Trujillo (1977) proved the unconditional stability of the symmetric splitting method for the undamped case, and referred the method as an unconditionally stable method. However, the method was shown to have poor accuracy performance by several researchers (Mullen and Belytschko 1983; Park and Housner 1982; Hughes and Belytschko 1983). The unsymmetric splitting method was shown more accurate than the symmetric splitting, but only conditionally stable (Trujillo 1977; Park and Housner 1982). Using the concept of a penalty matrix for approximate factorization, Park (1982) presented a strategy to improve the accuracy of the symmetric splitting method.

It has been pointed out by Hajjar (1988) that these algorithms are not ideal for parallel implementation due to the inherently sequential nature of back substitution.

Group Explicit Algorithms

The group explicit algorithms (Evans 1984, 1985; Abdullah and Evans 1986; Hajjar 1988) use an explicit approach, such as the central difference method, with a symmetric splitting operator formulated from the superposition of the left-right and right-left operators. With appropriate partitioning of the problem domain into subdomains, the stiffness matrix can be made block diagonal, and the solution can be computed independently in each subdomain and simultaneously in all subdomains. Similar to the central difference method, these methods are suitable for parallel processing and only nearest neighbor communication is required in a time step due to the explicit solution performed in the subdomain interfaces. Although the implicit solution performed in each subdomain

allows a larger time step than that of the purely explicit solution, Hajjar (1988) has shown that these methods are only conditionally stable and the stability limit governed mainly by the explicit solution in the interfaces is not increased sufficiently to justify the added computation of the implicit solution.

The alternating group explicit algorithms (Evans 1984; Abdullah and Evans 1986; Hajjar 1988) perform implicit solution in the interfaces every other step to achieve unconditional stability. The solution of a reduced set of simultaneous equations is therefore required. In addition, these methods have been reported to provide severe amplitude decay for time steps near and above the explicit stability limit by Hajjar and Abel (1989b).

Group Implicit Algorithms

In the group implicit algorithms (Ortiz and Nour-Omid 1986; Ortiz et al. 1988; Hajjar 1988), the problem domain is first partitioned into subdomains and the implicit operator, such as the Newmark-β constant average acceleration algorithm, is used to obtain a local solution in each subdomain. The computed results in the subdomain interfaces are then weighted by using the mass matrix and averaged to produce a unique solution.

These methods may be formulated to be unconditionally stable and are inherently amenable to parallel processing. The implicit solution of each subdomain may be performed independently of all other domains and only nearest neighbor communication is required for the averaging procedure in each time step. However, these methods have limited range of applicability (Ortiz 1991) and have been shown to provide inadequate

accuracy for practical time step sizes in dynamic analyses of framed structures (Hajjar and Abel 1989b) due to conditional consistency in these methods (Farhat and Sobh 1990).

Summary of Algorithms

As discussed previously, the semi-implicit algorithms are inherently sequential and not suitable for parallel implementation. Both the group explicit and group implicit algorithms are well suited for parallel processing, but the group explicit algorithms suffer from inaccuracy problems, and the group implicit algorithms have limited range of applicability. In addition, the efficiency of parallel mixed-time integration methods depends mainly on how the load-balancing issues are addressed. In a distributed memory environment such as the networked workstations investigated in this work, the inherent high communication overhead may make the load-balancing task difficult and expensive.

The explicit central difference method is inherently amenable to parallel processing, but is conditionally stable and often requires at least an order-of-magnitude more time steps than an unconditionally stable implicit method. Therefore, for certain dynamics problems, such as those involving short-duration loadings (such as impact) where a short time step is necessary to capture the dynamic phenomena, the parallel central difference analysis can be a powerful and efficient time integration algorithm.

The implicit algorithms are not as amenable to parallel processing as the explicit central difference method because a global simultaneous solution is needed in each time step. However, the unconditional stability of these methods makes them more suitable for long-duration problems than

the central difference method. The major challenge in the parallel implicit analysis is the devising of an efficient parallel solution technique for the set of simultaneous equations relating the inter-substructure degrees of freedom.

3.2.2 Parallel Equation Solvers

The solution of the linear system of equations is required both in a number of time integration methods (as discussed in Section 3.2.1.2) and in steady-state (static) analyses (for example, see Eq. 2.32). Since the equation solving can be the most time consuming task in an analysis, the efficiency of its parallel implementation greatly affects the performance of the entire parallel analysis. Many techniques for parallel solution of simultaneous equations have been studied in previous research and are reviewed briefly in this subsection.

Direct Algorithms

Direct algorithms obtain the solution of the system of equations in a known number of arithmetic operations and within machine precision (i.e., with only the rounding and critical-arithmetic errors introduced in the computation). The most widely used direct algorithms are Gaussian elimination and its variants, such as Cholesky (LL^T), and LDL^T decompositions. Formulations of these methods are well known (see, for example, Golub and Van Loan 1989; Bathe 1982) and, therefore, not repeated here. Because the system of equations arising in many finite element formulations is symmetric, positive definite, and banded, these direct methods are usually more efficient than iterative methods on sequential computers.

A significant amount of research has been devoted to parallelization of direct algorithms on both shared and distributed memory parallel environments. For example, a parallel row-oriented Gaussian elimination was devised by Farhat (1987) for both the Intel iPSC Hypercube (local memory) and the Cray X-MP (shared memory) computers. A parallel active column solver was developed by Farhat and Wilson (1988) on both the Intel iPSC Hypercube (local memory) and the Encore Multimax (shared memory) computers. Parallel Cholesky decomposition solvers were studied by Farhat (1987) on the Intel iPSC Hypercube (local memory) computer and by Chiang and Fulton (1990) on both the FLEX/32 (shared memory) and the Intel iPSC Hypercube (local memory) computers. In addition, a parallel frontal solver was presented by Zhang and Lui (1991) on the Alliant FX/80 (shared memory) computer.

In addition to the difficulty involved in the parallel implementation of direct algorithms due to the sequential procedures inherent in the algorithms, parallel direct algorithms usually require extensive interprocess communication and entail difficult load balancing tasks. In the context of finite element analysis, the partitioning of data for the parallel direct solvers is generally different from that for other phases of the analysis, such as element formation and stress recovery. Furthermore, to achieve high parallel efficiency, different phases of the parallel direct solution (for example, forward reduction and backward substitution) may require different data partitioning strategies. All of the above complexities plus the shared database nature of the decomposition procedure make the shared memory environment a more efficient parallel environment for parallel direct algorithms than the distributed memory environment.

A popular strategy used in the parallel implementation of direct solvers is substructuring. The substructuring strategy first partitions the structure into a number of subdomains and assigns each subdomain to a separate processor. Then, assembly and static condensation are performed independently and concurrently within each subdomain without any interprocess communication. Finally, a condensed set of equations associated with unknowns along subdomain interfaces is solved by a desired parallel direct algorithm. This approach greatly reduces the cost of interprocess communication and, therefore, is more efficient than the outright use of parallel direct solvers for the solution of the entire set of original equations.

Iterative Algorithms

Starting with an initial approximation of the solution, iterative algorithms calculate successive approximations of the solution until the approximation converges to the exact solution with the desired accuracy. Some commonly used iterative algorithms include Jacobi, Gauss-Seidel, successive overrelaxation, dynamic relaxation, and conjugate gradient methods. Descriptions of these iterative methods are available elsewhere (see, for example, Hageman and Young 1981; Jennings 1977; Golub and Van Loan 1989; Underwood 1983) and are not repeated here. When iterative methods converge sufficiently fast, they generally require less amount of computation than direct methods. For problems with a considerable number of unknowns (say, for example, over 10,000) and large bandwidth, iterative methods appear to be especially more effective than direct methods. However, the efficiency of iterative methods depends heavily on how fast they converge which is usually not known *a priori*.

Considerable research has been conducted on the parallel implementation of iterative algorithms. For example, a parallel block SOR iteration method was developed by Farhat (1987) using a recursive substructuring technique. Evan and Yousif (1992) reviewed and presented asynchronous parallel iterative methods, such as asynchronous parallel Jacobi and Gauss-Seidel methods as well as a purely asynchronous parallel iterative method. A synchronous parallel Jacobi method was also implemented for comparison with the asynchronous methods. It was shown that asynchronous methods were more effective than synchronous methods because they had less synchronization overhead and it was easier to achieve load-balancing among processors. A parallel explicit dynamic relaxation method was implemented by Farhat and Crivelli (1989). Although this method is robust and amenable to parallel processing, it may be slow in some applications. In addition, Nour-Omid et al. (1987) investigated a parallel preconditioned conjugate gradient method with two different preconditioners: diagonal scaling and incomplete LU factorization. It was found that the incomplete LU preconditioning was more effective in reducing the number of iterations than the diagonal preconditioning, but spent more time per iteration and resulted in an overall increase in the solution time.

Hybrid Algorithms

Hybrid algorithms combine advantages of both direct and iterative methods. These algorithms start with partitioning the structure into a number of subdomains and assign each subdomain to a separate processor. Then, a direct decomposition method is used for substructure condensation which is carried out on each processor independently and concurrently

without any interprocess communication. Finally, a condensed set of system equations associated with unknowns along subdomain interfaces is solved by a parallel iterative algorithm. Examples of parallel implementation of hybrid algorithms can be found in Nour-Omid et al. (1987) and Hajjar (1987, 1988).

3.3 Present Approaches

This section discusses the parallel solution algorithms investigated and implemented in this work for simulations of structural dynamics. For parallel explicit transient analysis, the parallel central difference method developed by Hajjar and Abel (1989a) is adopted in this research. For parallel implicit transient analysis, the parallel Newmark algorithm with domain decomposition developed by Hajjar and Abel (1988) is adopted with slight modifications. The implementation of the former algorithm has been carried out in collaboration with Dr. Brian H. Aubert, while the implementation of the latter algorithm has been carried out in collaboration with Dr. Sanjeev Srivastav, both at the Cornell Program of Computer Graphics. For steady-state analysis, a parallel Newton-Raphson iterative incremental algorithm has been implemented. This algorithm uses the same approach as the parallel implicit algorithm except that the time stepping outer loop is replaced by the load increment outer loop.

3.3.1 Parallel Explicit Transient Solution

The present research adopts the parallel central difference method developed by Hajjar and Abel (1989a) for explicit transient solution of structural dynamics. Formulation of the central difference method has already been presented in Section 2.4 (see Eqs. 2.38 - 2.42). As discussed

previously, the central difference algorithm is inherently amenable to parallel processing. With the use of lumped masses to yield a diagonal mass matrix, solution may proceed on a degree-of-freedom level without assembly of the global matrices and solution of simultaneous equations (see, for example, Eq. 2.42).

To take advantage of parallel processing, the finite element domain is first partitioned into a number of subdomains which are then distributed among the processors, and the computation involved in each subdomain is carried out by a separate processor. The structural partitioning needed by the present central difference algorithm is shown in Fig. 3.1 for a simple two-dimensional frame. The elements in a particular substructure are either interior elements if both their nodes lie completely within the boundary of the substructure, or border elements if one of their nodes is resident in a neighboring substructure. The interior nodes in a particular substructure are nodes lying within the boundary of the substructure. The boundary nodes are a subset of the interior nodes and connected to at least one border element. The adjacent nodes are also connected to at least one border element, but lie outside the boundary of the substructure.

With this partition of structural data, the interprocess communication can be minimized by including the border elements on both of the processors associated with neighboring substructures. In this case, only one nearest-neighbor communication for exchanging the displacements of the boundary nodes is required per time step. This communication efficiency is achieved at the expense of a duplication of effort to perform the element calculations for the border elements on the processors sharing these elements. This approach is suitable for parallel analyses in which the

number of border elements is significantly smaller than the number of interior elements and the cost of interprocess communication is relatively high. This is usually the case for the type of problem and environment characterizing the current work.

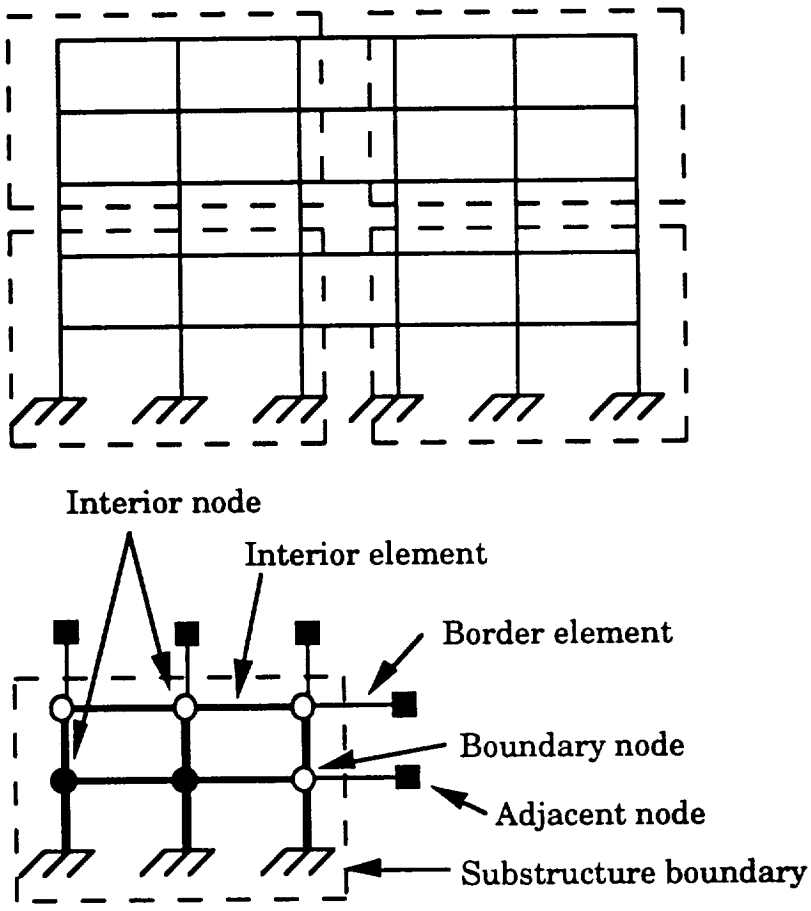


Figure 3.1 The partition of structural data for the parallel explicit algorithm (Hajjar and Abel 1989a)

The parallel central difference method has been implemented in ABREAST using a multiple-instruction, multiple-data (MIMD) algorithm in a message passing environment provided by ISIS. Each processor executes identical code, but asynchronously and on different data (i.e., on a different

substructure). Within each time step, the one nearest-neighbor communication for exchanging the displacements of the boundary nodes is achieved through ISIS process-group and broadcast mechanisms. Table 3.1 briefly outlines the present implementation of this parallel algorithm. It should be noted that synchronization only between adjacent processors is required at Steps (5b) and (5c) to avoid race conditions.

Table 3.1 Parallel explicit algorithm

(1) Input and setup basic analysis data.
(2) Compute internal force vector.
(3) Compute load vector.
(4) Solve for displacements.
(5) Exchange the displacements of the boundary nodes with adjacent substructures through message passing.
  (5a) Send requests to adjacent substructures for the displacements of the adjacent nodes.
  (5b) Wait for requests from adjacent substructures, then send out the displacements of the boundary nodes.
  (5c) Wait until displacements for all the adjacent nodes are received.
(6) Compute velocities and accelerations.
(7) Recover stresses and strains.
(8) Go to (2) for next time step.

In addition, to maximize the portability of the code for any UNIX or UNIX-like operating system and for alternative message passing environments, an effort has been taken to provide a layer of software to isolate the ISIS message passing routines from the application code (this has been done for all parallel implementations in this work). To illustrate

what and how ISIS routines are used in this work to achieve interprocess communication and synchronization for all implemented parallel algorithms, some example segments extracted directly from the analysis program are provided in Appendix A.

3.3.2 Parallel Implicit Transient Solution

The present research investigates the parallel Newmark constant average acceleration algorithm with domain decomposition described by Hajjar and Abel (1988) for implicit transient solution of structural dynamics. Formulation of the Newmark constant average acceleration algorithm has already been presented in Section 2.4 (see Eqs. 2.43 - 2.47). Within the Newmark time stepping outer loop, the domain decomposition approach employs a hybrid algorithm as discussed previously in Section 3.2.1.2 for the solution of a set of dynamic equilibrium equations (see, for example, Eq. 2.45). For the sake of discussions, formulation of the domain decomposition approach is briefly reviewed here.

The domain decomposition approach requires that the finite element domain be partitioned into a number of subdomains for substructuring analysis. The structural partitioning needed is shown in Fig. 3.2 for the same two-dimensional frame of Fig. 3.1. The nodes in a particular subdomain are either interior nodes, which lie within the boundary of the subdomain, or boundary nodes, which lie along the interfaces between subdomains. The boundary nodes are further categorized into primary and secondary boundary nodes. Each boundary node is a primary node in only one subdomain but, at the same time, a secondary boundary node in all other subdomains which share it. The interior elements in a particular

subdomain are connected to at least one interior node, while the boundary elements are not connected to any interior nodes but boundary nodes.

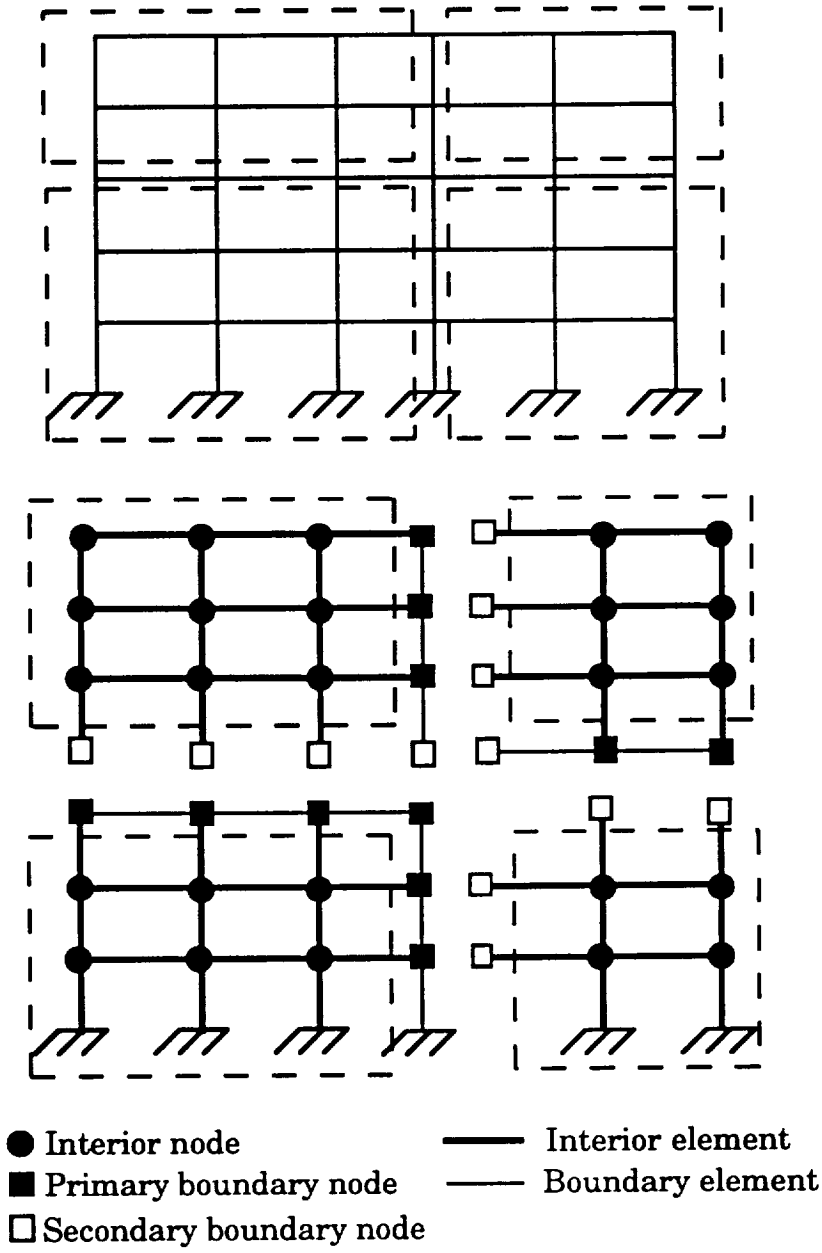


Figure 3.2 The partition of structural data for the parallel implicit algorithm (Hajjar and Abel 1988)

For each subdomain s, which has interior degrees-of-freedom i and boundary degrees-of-freedom h, the partitioning takes the form

$$\begin{bmatrix} [\hat{K}^s_{ii}] & [\hat{K}^s_{ih}] \\ [\hat{K}^s_{hi}] & [\hat{K}^s_{hh}] \end{bmatrix} \begin{Bmatrix} \{\Delta u^s_i\} \\ \{\Delta u^s_h\} \end{Bmatrix} = \begin{Bmatrix} \{\hat{R}^s_i\} \\ \{\hat{R}^s_h\} \end{Bmatrix} \tag{3.1}$$

In this work the modified decomposition algorithm developed by Han and Abel (1984) is used for the condensation of the interior degrees-of-freedom. The condensation results in a reduced set of equations

$$[\hat{K}^s_H]\{\Delta u^s_h\} = \{\hat{R}^s_H\} \tag{3.2}$$

in which

$$[\hat{K}^s_H] = [\hat{K}^s_{hh}] - [N^s_{hi}]\,[N^s_{hi}]^T \tag{3.3}$$

$$[N^s_{hi}]^T = [L^s_{ii}]^{-1}[\hat{K}^s_{ih}] \tag{3.4}$$

$$[\hat{K}^s_{ii}] = [L^s_{ii}]\,[L^s_{ii}]^T \tag{3.5}$$

In the current implementation, the skyline format is used for matrices $[\hat{K}^s_{ii}]$, $[\hat{K}^s_{ih}]$, $[N^s_{hi}]$, and $[L^s_{ii}]$. The Cholesky decomposition is performed in Eq. (3.5) to obtain $[L^s_{ii}]$, then the following formula is used directly to compute $[N^s_{hi}]$

$$N_{jk} = \left(K_{jk} - \sum_{n=1}^{k-1} N_{jn} L_{kn} \right) / L_{jj} \tag{3.6}$$

in which N_{jk} is the element at the j^{th} row and k^{th} column of $[N^s_{hi}]$, K_{jk} is the element at the j^{th} row and k^{th} column of $[\hat{K}^s_{ih}]$, and L_{kn} is the element at the k^{th} row and n^{th} column of $[L^s_{ii}]$. To take advantage of the skyline format, the summation index n in Eq. (3.6) does not actually start from one but, instead, starts from the first nonzero product.

Upon completion of the condensation, Eqs. (3.2) for all subdomains are assembled to form a set of global equations for the unknowns along the subdomain interfaces. In this research, a parallel preconditioned conjugate gradient method is used for solution of these unknowns. Then, the following equation is used by each subdomain to solve for $\{\Delta u_i^s\}$:

$$\{\Delta u_i^s\} = [L_{ii}^s]^{-T}[L_{ii}^s]^{-1}(\{\hat{R}_i^s\} - [\hat{K}_{ih}^s]\{\Delta u_h^s\}) \tag{3.7}$$

The present research investigates two different preconditioners for the parallel preconditioned conjugate gradient method. The first one, which was used by Hajjar and Abel (1988), factors the coefficient matrix $[\hat{K}_H^s]$ (see Eq. 3.2) of each subdomain, and the factored preconditioner is constructed as

$$[P]^{-1} = \sum_{j=1^A}^{N_{sub}} \left([\hat{K}_H^j]^{-1} \right) \tag{3.8}$$

in which N_{sub} is the number of subdomains and Σ_A is the global finite element assembly operator. It should be noted that $[\hat{K}_H^s]$ in Eq. (3.8) may not be positive definite for subdomains that does not have enough prescribed displacements to avoid local rigid body motions. In dynamic analyses, however, the contribution of the mass matrix (to the effective stiffness matrix of Eq. 2.46) may in many cases help make $[\hat{K}_H^s]$ positive definite. Generally speaking, the successful construction of this preconditioner is not always guaranteed.

The second preconditioner investigated here assembles only the diagonals of $[\hat{K}_H^s]$ from all subdomains to construct the diagonal scaling preconditioner of the form

$$[P] = \sum_{j=1^A}^{N_{sub}} \text{diag}\left(\left[\hat{K}_H^j\right]\right) \tag{3.9}$$

in which N_{sub} is the number of subdomains, Σ_A is the global finite element assembly operator, and diag(.) retains only the diagonal terms in a matrix. This preconditioner is selected in this research over other preconditioners, such as incomplete Cholesky and block diagonal scaling preconditioners, because it is simple and requires less computation and interprocess communication.

The parallel Newmark constant average acceleration algorithm with domain decomposition described above has been implemented in ABREAST using a MIMD algorithm. Within each time step, the substructure condensation for each subdomain is performed on a separate processor independently and concurrently with no interprocess communication required. The solution of the unknowns along the subdomain interfaces is then computed using the parallel preconditioned conjugate gradient algorithm which requires both nearest-neighbor and global interprocess communication. For nonlinear analysis, the present research uses a modified-Newton iterative solution scheme to obtain the equilibrium solution at the end of each time step. This equilibrium check also requires both nearest-neighbor and global interprocess communication. Tables 3.2 - 3.5 briefly outline the present implementation of this parallel implicit algorithm. It should be noted that a host processor is selected among processors automatically by the program at the beginning of the analysis to facilitate global interprocess communication whenever it is needed. In addition, the dot products needed by the conjugate gradient algorithm in Table 3.4 are computed on the primary boundary nodes.

Table 3.2 Parallel implicit algorithm

(1) Input and setup basic analysis data.
(2) Compute internal force vector.
(3) Compute global stiffness matrix (as of Eq. 2.46).
(4) Compute load vector (as of Eq. 2.47).
(5) Solve for displacements.
  (5a) Condense the interior degrees-of-freedom.
  (5b) Assemble condensed load vector (see Table 3.3).
  (5c) Use parallel preconditioned conjugate gradient method (see Table 3.4) to solve for boundary degrees-of-freedoms.
  (5d) Solve for interior degrees-of-freedoms.
(6) Compute velocities and accelerations.
(7) Recover stresses and strains.
(8) Check equilibrium for nonlinear analysis.
  (8a) Assemble internal force and load vectors (see Table 3.3).
  (8b) Compute unbalanced force vector.
  (8c) Assemble unbalanced force norm (see Table 3.5).
  (8d) Go to (9) if convergence is achieved.
  (8e) Solve for displacements (same as Step 5).
  (8f) Compute velocities and accelerations (same as Step 6).
  (8g) Recover stresses and strains (same as Step 7) and go to (8a).
(9) Go to (2) for next time step.

Table 3.3 Assembly of the vector at subdomain boundary

(1) Send requests to adjacent subdomains for their contributions to the vector being assembled for the current subdomain.
(2) Wait for requests from adjacent subdomains, then send out the contributions of the current subdomain to adjacent subdomains.
(3) Wait until contributions from all adjacent subdomains are received and assembled.

Table 3.4 Parallel preconditioned conjugate gradient algorithm

$[K] = \left[\hat{K}_H^s\right]$; k = iteration number = 0; $\{u\}_o = \{0\}$; $\{r\}_o = \{R\} = \left\{\hat{R}_H^s\right\}$;

While($\|\{r\}_k\|_2 \le (\text{tol.})\|\{R\}\|_2$)

Solve $[P]\{z\}_k = \{r\}_k$

$k = k + 1$

Assemble $\{z\}_k$ (see Table 3.3) and compute average values over neighboring subdomains

Assemble the dot product $\{r\}_k^T\{z\}_k$ (see Table 3.5)

if (k = 1)

$\{p\}_1 = \{z\}_o$

else

$\beta_k = \{r\}_{k-1}^T\{z\}_{k-1} / \{r\}_{k-2}^T\{z\}_{k-2}$

$\{p\}_k = \{z\}_{k-1} + \beta_k\{p\}_{k-1}$

end

Assemble $\{S\} = [K]\{p\}_k$ (Table 3.3)

Assemble the dot product $\{p\}_k^T\{S\}$ (see Table 3.5)

$\alpha_k = \{r\}_{k-1}^T\{z\}_{k-1} / \{p\}_k^T[K]\{p\}_k$

$\{u\}_k = \{u\}_{k-1} + \alpha_k\{p\}_k$; $\{r\}_k = \{r\}_{k-1} - \alpha_k[K]\{p\}_k$

Assemble the dot products $\{r\}_k^T\{r\}_k$ and $\{R\}^T\{R\}$ (see Table 3.5)

end

$\{u\}$ = solution vector = $\{u\}_k$

Table 3.5 Assembly of the dot product

(1) Signal adjacent subdomains that the current processor is ready for assembling the dot product.

(2) Wait for signals from adjacent subdomains, then send out local dot product to host processor.

(3) If the current processor is the host processor, wait until local dot products from all processors are received and assembled, then send assembled global dot product to all processors.

(4) Wait until global dot product is received from the host processor.

3.3.3 Parallel Steady-State Solution

The present research implements a parallel Newton-Raphson iterative incremental algorithm for steady-state solution of rotational dynamics. The algorithm uses the same domain decomposition approach as the parallel implicit algorithm discussed previously. The algorithm has also been implemented in ABREAST and is outlined briefly in Table 3.6.

Table 3.6 Parallel steady-state (static) algorithm

(1) Input and setup basic analysis data.
(2) Compute global stiffness matrix.
(3) Compute load vector.
(4) Solve for displacements.
  (4a) Condense the interior degrees-of-freedom.
  (4b) Assemble condensed load vector (see Table 3.3).
  (4c) Use parallel preconditioned conjugate gradient method (see Table 3.4) to solve for boundary degrees-of-freedoms.
  (4d) Solve for interior degrees-of-freedom.
(5) Recover stresses and strains.
(6) Check equilibrium for nonlinear analysis.
  (6a) Assemble internal force and load vectors (see Table 3.3).
  (6b) Compute unbalanced force vector.
  (6c) Assemble unbalanced force norm (see Table 3.5).
  (6d) Go to (7) if convergence is achieved.
  (6e) Compute global stiffness matrix (same as Step 2).
  (6f) Solve for displacements (same as Step 4).
  (6g) Recover stresses and strains (same as Step 5) and go to (6a).
(7) If at the end of final increment, go to (8); else go to (2).
(8) Perform (8a) and (8b) recursively until the update of the load vector no longer affects the equilibrium.
  (8a) Update load vector (see Table 3.3).
  (8b) Perform equilibrium iterations (same as Step 6).

3.4 Effectiveness of Parallel Analysis

This section evaluates and discusses the effectiveness of the parallel algorithms implemented in this research. The implementation of the parallel algorithms have been verified by comparing results of parallel analyses with those of serial analyses. The following definitions of speed-up, S, and efficiency, E, are used to evaluate the performance of the parallel algorithm:

$$S = \frac{\text{Elapsed wall clock time for solution on one processor}}{\text{Elapsed wall clock time for solution on } N_p \text{ processors}} \tag{3.10}$$

$$E\,(\%) = \frac{\text{Speed-up}}{N_p} \times 100 \tag{3.11}$$

in which N_p is the number of processors used in the parallel analysis. The wall-clock time is used in Eq. (3.10) because it yields the most conservative measures of speedup and efficiencies. For example, communication delays and overhead are fully accounted for with this definition. Nevertheless, it should be stated that the results reported in this thesis were obtained by runs overnight while there was little other traffic on the network.

In the investigation of both the parallel implicit and steady-state analyses, it should be noted that the single-processor analysis does not use the substructuring (or domain decomposition) approach but, instead, uses a direct Gauss elimination method for the solution of the total set of original equations. For the solution of problems which are not too large to be accommodated in the core memory of a single processor, this direct approach is believed to be more efficient than the substructuring approach with either a direct or an iterative equation solver in most cases. This also leads to more conservative measures of speed-up and efficiency.

3.4.1 Parallel Explicit Transient Analysis

Three structures of increasing size have been used in the preliminary study (Abel et al. 1991; Aubert 1992) to evaluate the effectiveness of the parallel central difference algorithm described in Section 3.2.2.1. The first structure is a transmission tower shown in Fig. 3.3. The tower has 434 truss elements, 160 nodes, and 468 unrestrained degrees of freedom. The

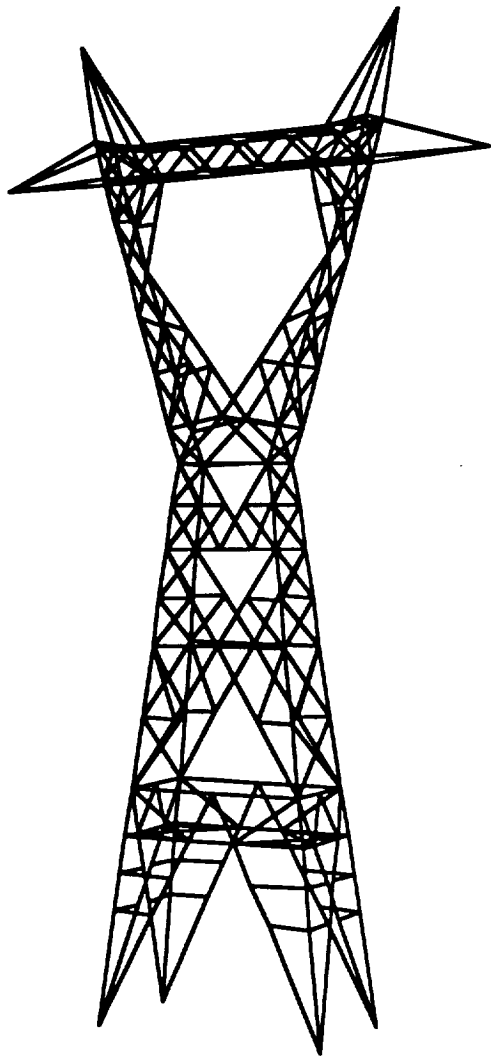


Figure 3.3 The finite element model of a transmission tower with 434 elements, 160 nodes, and 468 unrestrained degrees of freedom

structure is subjected to the El Centro earthquake. Both geometrical and material nonlinearities are considered in the analysis. Up to four DEC5000's are used.

The second structure is shown in Fig. 3.4 which is an unsupported space station with 1,428 truss elements, 304 nodes, and 912 unrestrained degrees of freedom. The structure is loaded with self-equilibrating external loads (Aubert 1992; Aubert et al. 1992) and is analyzed with the consideration of geometrical nonlinearity. Up to four DEC5000's are used.

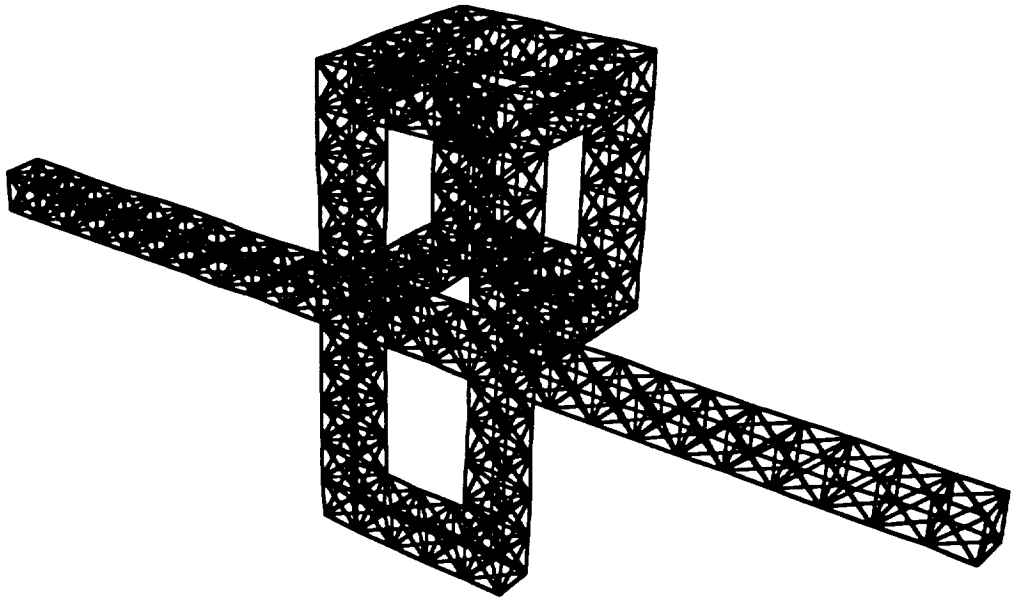


Figure 3.4 The finite element model of a space station with 1,428 elements, 304 nodes, and 912 unrestrained degrees of freedom

The third structure is a thirty-story building shown in Fig. 3.5. The structure consists of 1,200 beam-column elements, 496 nodes, and 2,880 unrestrained degrees of freedom and is subjected to the El Centro

earthquake. Both geometrical and material nonlinearities are considered in the analysis. Up to five DEC5000's are used.

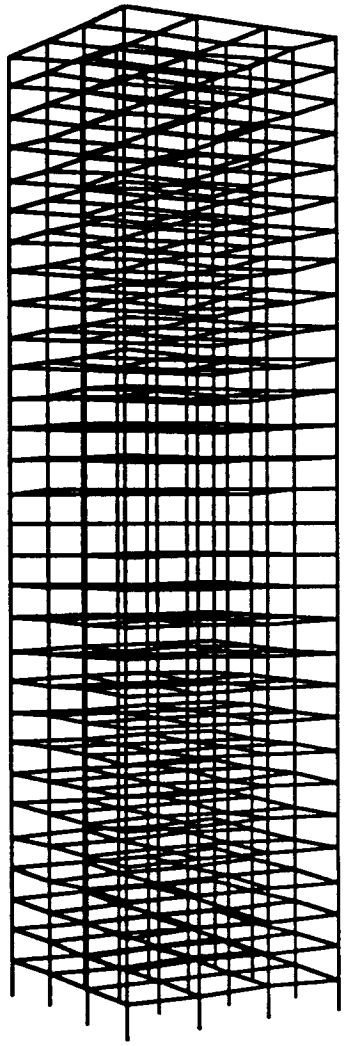


Figure 3.5 The finite element model of a thirty-story building with 1,200 elements, 496 nodes, and 2,880 unrestrained degrees of freedom

Figure 3.6 plots the speed-up results obtained in the parallel analyses versus the linear (optimal) speed-up. For the transmission tower, which is the smallest structure in these three examples, the speed-up ranges from

1.7 for two processors to 2.7 for four processors. For the 30-story building, the largest one in the three examples, the speed-up ranges from 1.8 for two processors to 4.1 for five processors.

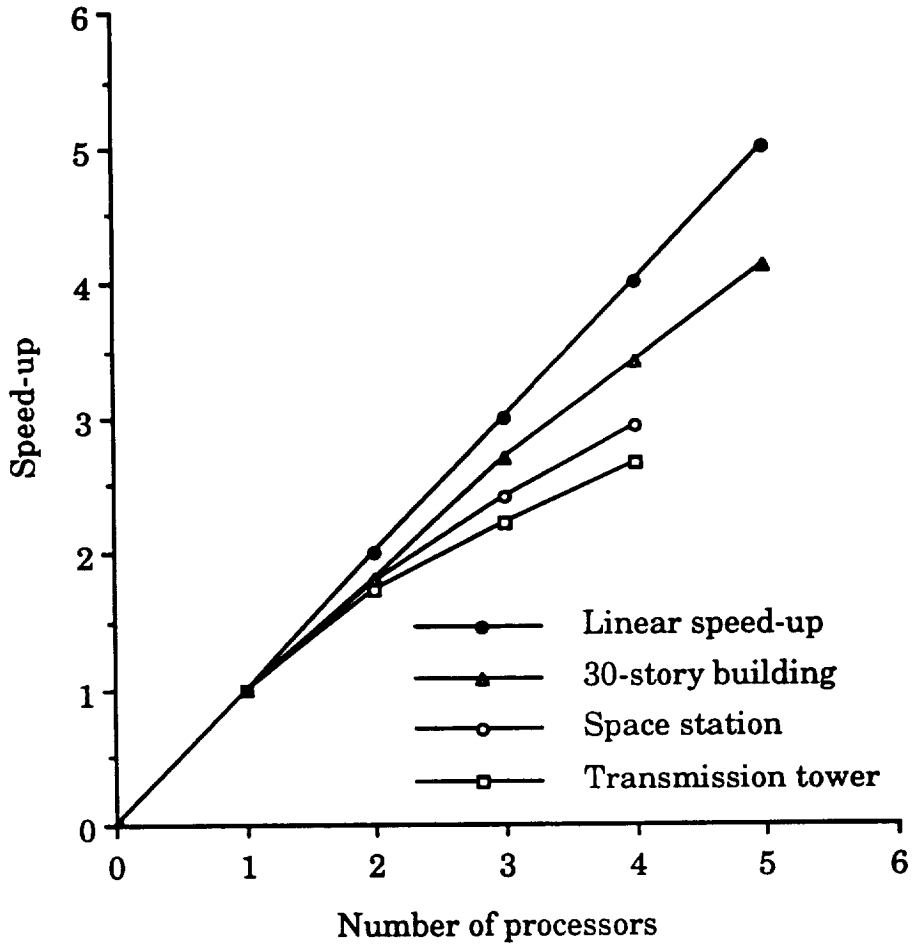


Figure 3.6 Speed-up results obtained for the parallel analysis on the structures of Figs. 3.3 - 3.5

In addition to the above studies, several geometrically nonlinear analyses using the consistent mass approach have been performed of the rotating bladed-disk problem with the finite element model of Fig. 3.7. The

model consists of 504 twenty-node brick solid elements, 3,828 nodes, and 10,044 unrestrained degrees of freedom. Up to six DEC5000's or up to twelve HP9000/720's are used. A time step of 0.00001 seconds is used, and 100 time steps were run. The speed-up and efficiency results obtained from the parallel analyses are given in Table 3.7. It takes about five hours and thirty-two minutes for a single DEC5000 to complete the analysis, while it takes about two hours and fifty-five minutes for a single HP9000/720. The speed-up results of each analysis are plotted versus the linear (optimal) speed-up in Fig. 3.8.

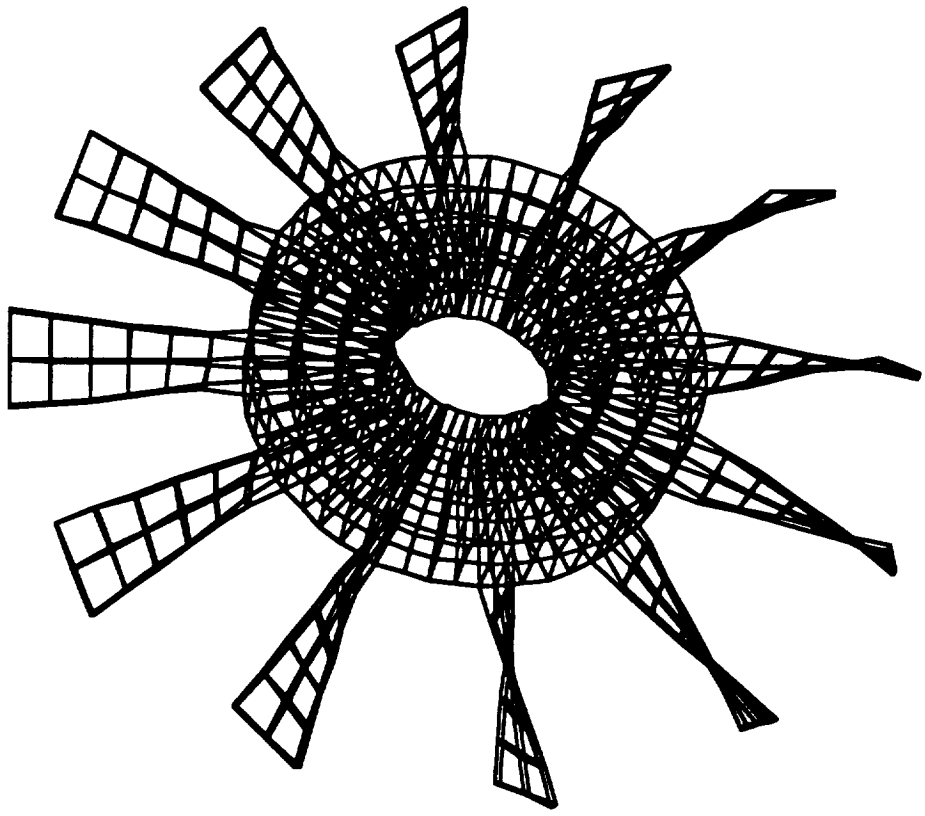


Figure 3.7 A finite element model of a 12-bladed turbine disk with 504 elements, 3,828 nodes, and 10,044 unrestrained degrees of freedom

Table 3.7 The speed-up and efficiency of the parallel central difference algorithm for analysis of the rotating bladed-disk problem of Fig. 3.7

N_p	DEC5000		HP9000/720	
	Speed-up	Efficiency (%)	Speed-up	Efficiency (%)
2	1.9	96	1.9	96
3	2.9	96	2.9	96
4	3.8	94	3.8	95
6	5.4	90	5.6	93
12	--	--	9.7	81

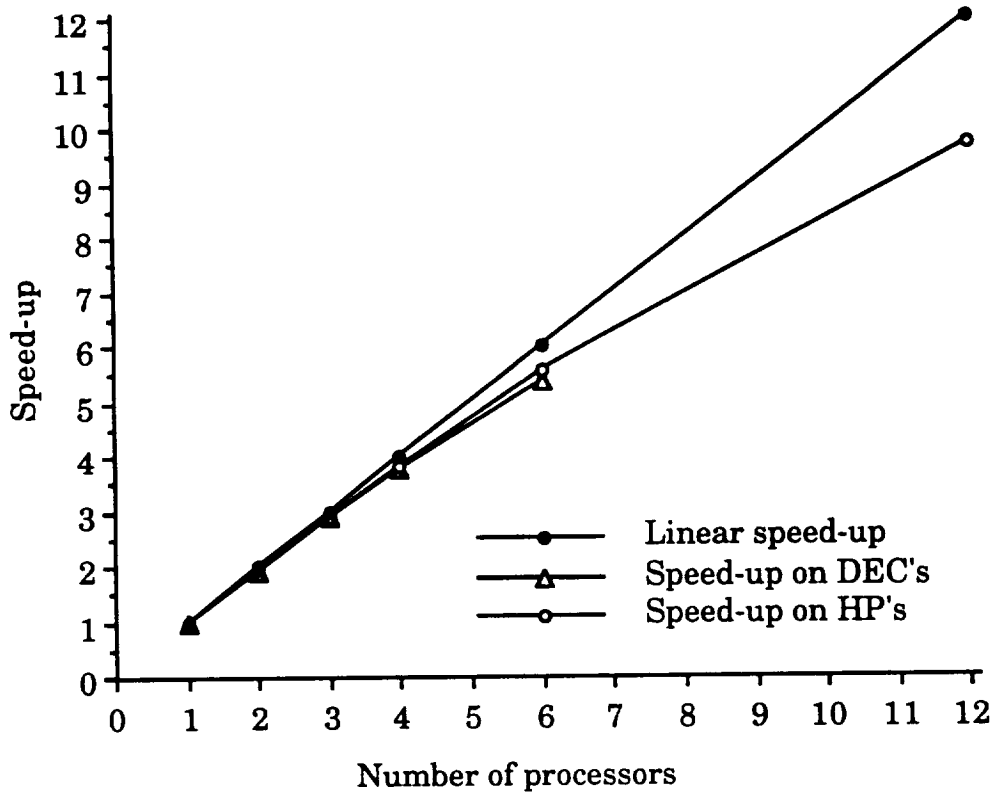


Figure 3.8 Speed-up of the parallel central difference method applied to the bladed-disk of Fig. 3.7

From Fig. 3.6, Table 3.7, and Fig. 3.8, the effectiveness of the parallel central difference method is apparent. It should be noted that the analyses cannot achieve linear speed-up, even with no communication overhead, because element calculations for the border elements are duplicated in the processors sharing these border elements. However, as long as the ratio of the number of border elements to the number of interior elements in each substructure is small, the parallel central difference algorithm can achieve high speed-up and efficiency.

3.4.2 Parallel Implicit Transient Analysis

Two structures have been analyzed to investigate the effectiveness of the parallel implicit algorithm. The first one is the same thirty-story building shown in Fig. 3.5. The building is subjected to the El Centro earthquake. Geometrically nonlinear analyses are performed. The time step is 0.01 seconds and 5 seconds of the earthquake are analyzed. The convergence tolerance for the modified-Newton iterations is $1x10^{-5}$, while the convergence tolerance for the conjugate gradient iterations is $1x10^{-8}$. Up to three DEC5000's are used for the analyses.

The speed-up and efficiency results obtained from the parallel analyses are given in Table 3.8. ANCGI denotes the average number of conjugate gradient iterations, while NIDOF denotes the total number of degrees of freedom on the subdomain interfaces. The average number of modified-Newton iterations per time step for all analyses is three. The analysis on a single DEC5000 takes about four hours and twenty-three minutes.

From Table 3.8, it can be observed that both preconditioners accelerate the convergence of the conjugate gradient iteration, but the diagonal scaling preconditioner of Eq. (3.9) is more effective in this case. In addition, as the number of processors increases from two to three, both the number of interface degrees of freedom and the number of conjugate gradient iterations increase, resulting in an increase of the total amount of synchronization and communication overhead. Although, at the same time, the amount of computation per processor decreases, the increase of overhead dominates in this case and the speed-up factors drop. This means that nothing is gained by having more than two processors in the parallel analysis of this example.

Table 3.8 The speed-up and efficiency of the parallel implicit algorithm for analysis of the thirty-story building of Fig. 3.5 (with the use of modified-Newton iteration solution scheme)

N_p	Preconditioner	ANCGI	S	E (%)	NIDOF
2	Diagonal	6	1.4	68	96
	Eq. (3.8)	20	1.1	56	
	None	29	1.1	52	
3	Diagonal	7	1.3	44	192
	Eq. (3.8)	31	0.7	24	
	None	44	0.7	23	

It should be noted that the use of modified-Newton iteration scheme is unfavorable to the conjugate gradient solution method because, in the equilibrium iteration loops, only backward substitution is required for the direct Gauss elimination method (which is the case of $N_p = 1$), while it is not

the case for the conjugate gradient method. To demonstrate this point, the same analyses are performed using the simple incremental solution scheme, instead of modified-Newton iteration scheme, for the cases of $N_p = 1$ and $N_p = 2$. The results are given in Table 3.9. It takes about three hours and twenty-five minutes for the analysis on a single DEC5000. It can be seen that the average number of conjugate gradient iterations decreases and better speed-up factors are obtained. Surprisingly, however, the preconditioner of Eq. (3.8) does not reduce but increases the number of iterations in this case.

Table 3.9 The speed-up and efficiency of the parallel implicit algorithm for analysis of the thirty-story building of Fig. 3.5 (with the use of simple incremental solution scheme)

N_p	Preconditioner	ANCGI	S	E (%)	NIDOF
2	Diagonal	3	1.5	75	96
	Eq. (3.8)	17	1.4	69	
	None	16	1.4	71	

The second structure analyzed is a twelve-story L-shaped building shown in Fig. 3.9. The finite element model of the structure consists of 468 beam-column elements, 72 nine-node Lagrangian shell elements, 482 nodes, and 2,508 unrestrained degrees of freedom. Again, the structure is subjected to the El Centro earthquake and geometrically nonlinear analyses are performed. The time step is 0.01 seconds and only 0.5 seconds of the earthquake are analyzed. The convergence tolerance for the modified-Newton iterations is $1x10^{-5}$, while the convergence tolerance for the

conjugate gradient iterations is $1x10^{-8}$. Up to four DEC5000's or up to four HP9000/720's are used for the analyses.

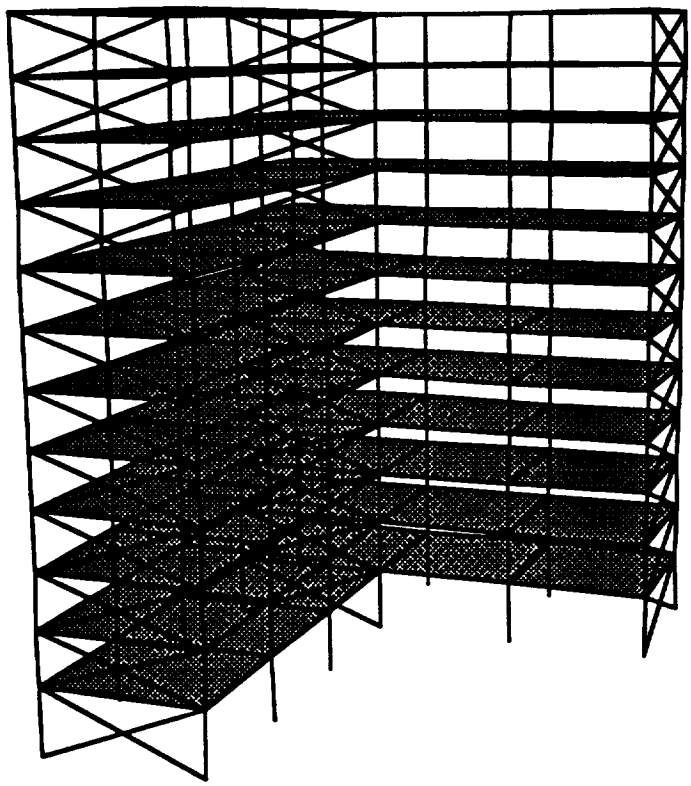


Figure 3.9 A finite element model of a 12-story L-shaped building with 468 beam-column elements, 72 shell elements, 482 nodes, and 2,508 unrestrained degrees of freedom

Table 3.10 reports the speed-up and efficiency results obtained from the parallel analyses. ANCGI denotes the average number of conjugate gradient iterations, while NIDOF denotes the total number of degrees of freedom on the subdomain interfaces. The average number of modified-Newton iterations per time step for all analyses is three. The analysis takes about three hours and forty-four minutes on a single DEC5000, while it takes about two hours and nine minutes on a single HP9000/720. In this

example, the preconditioner of Eq. (3.8) is not used because singular $\left[\hat{K}_H^s\right]$ is encountered during the analysis. All analyses performed on HP9000/720's use the diagonal scaling preconditioner of Eq. (3.9). In addition, the speed-up results of each analysis are plotted versus the linear speed-up in Fig. 3.10.

Table 3.10 The speed-up and efficiency of the parallel implicit algorithm for analysis of the twenty-story L-shaped building of Fig. 3.9

N_p	Preconditioner	ANCGI	DEC5000		HP9000/720		NIDOF
			S	E (%)	S	E (%)	
2	Diagonal	13	2.1	103	2.1	104	84
	None	85	1.8	91	--	--	
3	Diagonal	13	2.5	84	2.5	84	168
	None	126	1.6	54	--	--	
4	Diagonal	13	3.3	81	3.3	83	252
	None	140	1.6	40	--	--	

From table 3.10 and Fig. 3.10, three observations can be made. First, the diagonal scaling preconditioner of Eq. (3.9) is again very effective in reducing the number of iterations. As the number of processors increases, the average number of conjugate gradient iterations increases in the unpreconditioned analyses but remains the same in the preconditioned analyses.

Second, the speed-up factors for the unpreconditioned analyses decreases when the number of processor increases, while those for the

preconditioned analyses increase. This shows the significance of the preconditioning.

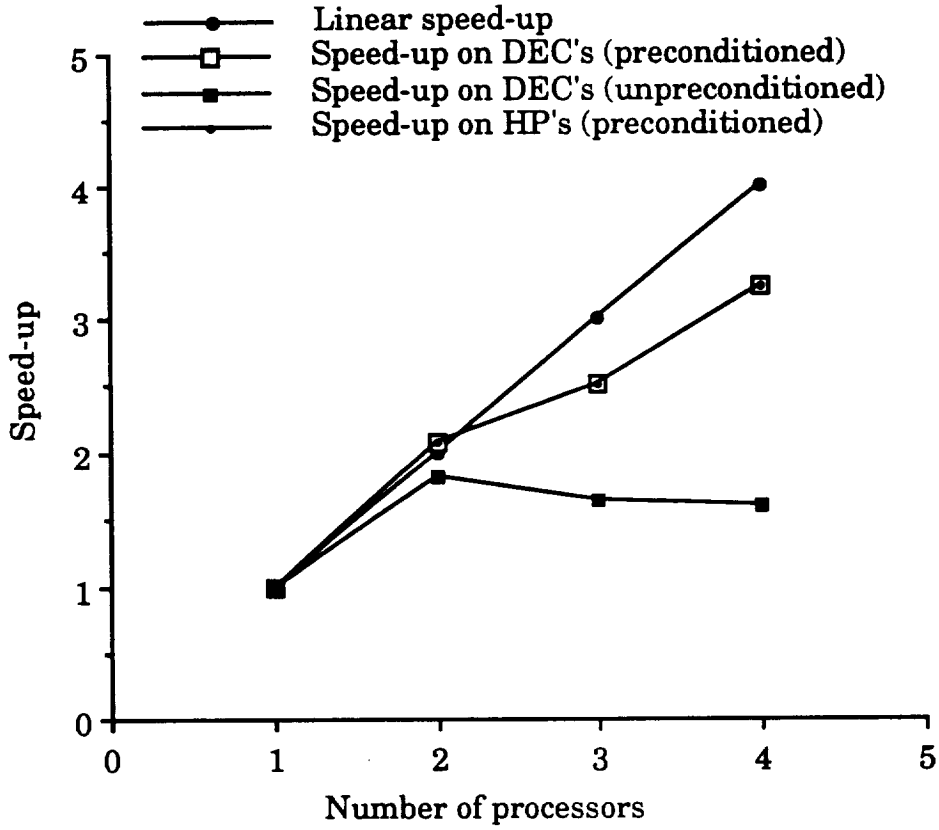


Figure 3.10 Speed-up for the twelve-story L-shaped building of Fig. 3.9 using the parallel implicit algorithm

Third, superlinear speed-up is achieved in the preconditioned analysis for the case of $N_p = 2$. This is probably due to the fact that the profile (or skyline) of the coefficient matrix for the whole structure in the serial analysis is not as well minimized as that of the coefficient matrix for each subdomain in the parallel analysis. The nodal numbering for the whole structure is currently done in BASYS by a crude bandwidth

minimization algorithm (Srivastav 1991), while nodal renumbering is performed for each subdomain using a modified version of the Gibbs-King profile reduction algorithm (Paulino 1988) in PSAINT. If the same profile reduction algorithm used for nodal renumbering in each subdomain is used to renumber the nodes in the whole structure, it is believed that the superlinear speed-up would not occur in this example.

Generally speaking, the parallel implicit algorithm is not as efficient as the parallel explicit algorithm because significantly more synchronization and communication overhead is required in the analysis. Nevertheless, it has been shown in the above study that the parallel implicit algorithm is reasonably effective for parallel nonlinear solution of structural dynamics, especially when the problem size is large and an appropriate number of processors is used. In addition, the preconditioning often accelerates the convergence rate of the conjugate gradient method, leading to better performance of the parallel implicit algorithm. The effectiveness of the diagonal scaling preconditioner is evident in both examples studied.

3.4.3 Parallel Steady-State Analysis

The steady-state analysis of the rotating bladed-disk with the finite element model of Fig. 3.7 is conducted to investigate the effectiveness of the parallel steady-state algorithm. The structure is rotating at a constant speed of 100 rpm. Geometrically nonlinear analyses are required. The consistent mass approach is employed to account for rotational nonlinearities. The convergence tolerance for the Newton-Raphson iterations is $1x10^{-5}$, while the convergence tolerance for the conjugate

gradient iterations is $1x10^{-10}$. Up to six DEC5000's or up to six HP9000/720's are used for the analyses.

The speed-up and efficiency results obtained from the parallel analyses are reported in Table 3.11. ANCGI denotes the average number of conjugate gradient iterations, while NIDOF denotes the total number of degrees of freedom on the subdomain interfaces. A total of fourteen Newton-Raphson iterations is required for all analyses to obtain final equilibrium solutions. The analysis on a single DEC5000 takes about three hours and eight minutes, while it takes about one hour and forty-six minutes on a single HP9000/720. In addition, Fig. 3.11 plots the speed-up results of each preconditioned analysis versus the linear speed-up.

Table 3.11 The speed-up and efficiency of the parallel steady-state algorithm for analysis of the rotating bladed-disk of Fig. 3.7

			DEC5000		HP9000/720		
N_p	Preconditioner	ANCGI	S	E (%)	S	E (%)	NIDOF
2	Diagonal	37	1.4	71	1.5	74	144
	None	48	1.4	70	--	--	
3	Diagonal	49	2.0	68	2.2	74	216
	None	52	2.0	65	--	--	
4	Diagonal	40	3.0	74	3.2	80	288
	None	49	3.0	74	--	--	
6	Diagonal	48	3.9	66	3.9	65	432
	None	54	3.9	65	--	--	

From Table 3.11 and Fig. 3.11, two observations are obtained. First, the diagonal preconditioning successfully reduces the number of iterations

required in conjugate gradient method but the reduction is less significant for this example than for those examples studied in previous section. In addition, the speed-up and efficiency of the preconditioned analysis is only slightly better than the unpreconditioned analysis. This may be because the additional communication overhead required in the preconditioned analysis offsets the reduction of communication overhead due to the reduction of iteration number.

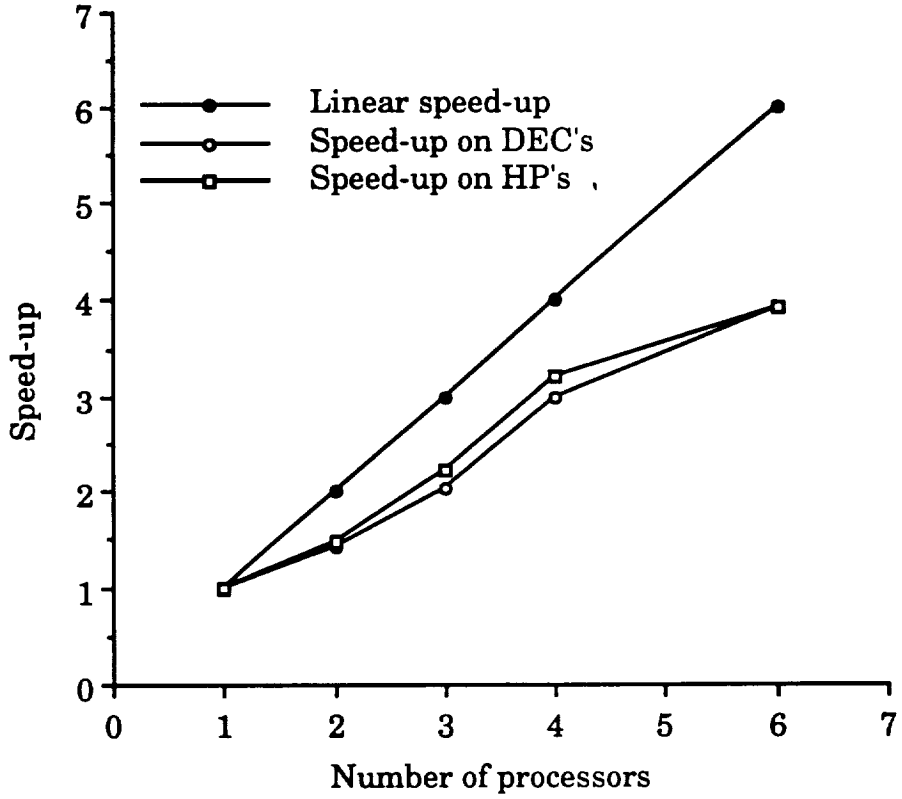


Figure 3.11 Speed-up for the 12-bladed turbine disk of Fig. 3.7 using the parallel steady-state algorithm

Second, as the number of processors increases, the speed-up factors increase. It should be noted, however, that the efficiency factors increase

significantly and reach the peak when the number of processors goes from three to four. The average number of conjugate gradient iterations required for the case of $N_p = 4$ is also smaller than those for the case of $N_p = 3$. This is because the nature of the structural domain partitioning affects both the efficiency of the substructuring condensation in each subdomain and the condition number of the condensed coefficient matrix. In this case, it happens to be that the partitioning for the case of $N_p = 4$ results in the best parallel performance of the algorithm among all cases.

Because the parallel steady-state algorithm is very similar to the parallel implicit algorithm, the examples presented in both the present and previous sections may be studied together to evaluate the effectiveness of the same parallel approach used by both algorithms. Although high synchronization and communication costs are inherent in the present networked workstation environment, the results obtained in these examples are encouraging. These results also indicate that both the implicit dynamic and steady-state analyses of either framed structures or rotating turbine bladed-disks can benefit from parallel processing.

Chapter 4

Load Balancing Among Processors

As discussed in Chapter 3, to take advantage of parallel processing in finite element analysis, the domain is usually partitioned (or decomposed) into a number of subdomains which are distributed among the processors and the computation involved in a subdomain is carried out by a separate processor. The key problems of this approach are how to partition the domain to achieve well-balanced workload distribution among processors and how to minimize the amount of interprocess communication so that significant speed-up can be obtained in the parallel analysis.

Load-balancing techniques have frequently been classified as either static or dynamic. Static load balancing techniques assume *a priori* knowledge of the static characteristics of the tasks and the system to distribute the tasks among processors. The task distribution is done only once and before the actual computation starts (i.e., in the preprocessing phase). Dynamic load balancing (sometimes referred to as adaptive load balancing) techniques utilize short-term knowledge of current state information of the tasks and the system to dynamically distribute the tasks among processors during the task execution. The tasks originally assigned to a processor may be migrated (or redistributed) to other processors at any time during the computation to improve the load balance. Dynamic load balancing may be important in problems involving material nonlinearity, self-adaptive mesh refinement, crack propagation, etc.

In this research, attention is limited to the static load balancing techniques for the domain partitioning problems involved in parallel finite element analysis of structural dynamics. Because the methods considered must be applicable to finite element meshes which are large and generic (irregular shapes, including multiply connected and/or branched), manual partitioning may be difficult (if not impossible) even with the help of interactive computer graphics. Simon (1991) has shown that visual perception may be inadequate for the task of partitioning large three-dimensional structures. Therefore, the focus of this work is on automatic algorithms, although interactive graphics tools are also developed to allow for manual partitioning and for examining and modifying results of automatic partitioning. The discussion of development of the graphics tools is deferred until Chapter 5.

Generic finite element meshes can be defined by the following four characteristics:

a) large meshes which may lack topological or geometrical regularity, e.g., meshes which have holes, loops, and/or branches.
b) meshes combining finite elements of different geometrical dimensions, e.g., 1-D line elements, 2-D quadrilateral elements, and 3-D brick elements.
c) meshes with finite elements of different shapes, e.g., triangular elements with three nodes and quadrilateral elements with four nodes.
d) meshes with finite elements of different interpolation orders, e.g., linear and quadratic.

The above definition gives an idea of the complexity involved in the problem of efficient partitioning of generic finite element meshes.

Different solution methods used in the analysis may require different strategies for domain partitioning. Two types of partitioning strategies may be classified: *element-based partitioning* and *node-based partitioning*. The element-based algorithms focus on partitioning elements in finite element meshes, while the node-based algorithms focus on partitioning nodes. For parallel solutions of structural dynamics, implicit solution methods often require element-based partitioning (for example, Hajjar and Abel 1988), while explicit methods may require either type depending on the parallel implementation (for example, Malone 1988; Hajjar and Abel 1989a). Both types of partitioning algorithms are addressed in this research. However, emphasis is placed on algorithms suitable for use with the parallel solution methods investigated extensively in this work as discussed in Chapter 3.

In this chapter, a brief review of previous research is given first, and some basic concepts and definitions of graph theory needed for the purpose of the present discussions are introduced. Then, domain partitioning algorithms to effect load balancing among processors proposed by previous researchers and developed by the present research are studied. Finally, these partitioning algorithms are evaluated and compared using different types of structures modelled by 1D, 2D, and 3D finite elements.

4.1 Review of Previous Research

There has been a great amount of research on load balancing in the field of operating systems. Numerous algorithms and techniques, both static and dynamic, have been studied and developed to maximize system

performance (i.e., minimize job turnaround) and to optimize processor utilization. Several types of system architectures have been also considered in the literature. However, relatively little research has been conducted addressing load balancing for parallel finite element analysis of structural dynamics.

The problem of domain partitioning of finite element meshes is equivalent to the problem of partitioning the graph associated with the mesh. Graph partitioning is an important NP-complete problem (Garey and Johnson 1979) and so is the general nature of mesh partitioning. Therefore, obtaining optimal solutions is practically intractable, but is also unnecessary because satisfactory near optimal solutions can always be sought at a relatively low cost by the use of heuristic algorithms.

Using interactive computer graphics, Hajjar (1987) developed a set of tools to help manually partition the structural domain for parallel dynamics solutions of three-dimensional framed structures. This approach works well for regular meshes with a small number of partitions. However, it is not in general effective and feasible for meshes that are complex, irregular, and large, especially if they are multi-connected and/or branched.

Flower et al. (1987) presented a simulated annealing method for mapping irregular finite element domains to parallel processors. Although these types of methods can give almost optimal results, they are usually expensive and time-consuming especially for problems of large size.

After reviewing several decomposition algorithms proposed by previous researchers, Malone (1988) proposed an automated mesh decomposition method for transient analysis on hypercube multiprocessor

computers using an explicit time integrator. The method is called Reduced Bandwidth Decomposition (RBD) and is based on a scheme which reduces the bandwidth of the matrix representation of the connectivities in the mesh. Only element-based partitioning is addressed.

Farhat (1988) proposed an automatic finite element domain decomposer which seeks to decompose a finite element mesh into a set of balanced domains sharing a minimum number of common nodal points. The decomposer is suitable for both shared memory and local memory multiprocessors. This algorithm explicitly addresses the element-based partitioning. The node-based partitioning is briefly discussed, but no algorithm is suggested. A drawback of this algorithm is that it may create domain splitting situation, i.e., portions of a domain may be unconnected or not contiguous. The domain splitting increases the size of subdomain boundaries as well as communication overhead in parallel computations. As an attempt to avoid the domain splitting situation, Al-Nasra and Nguyen (1991) incorporated the geometry information of the finite element meshes into an automatic decomposition algorithm similar to the one proposed by Farhat (1988). Padovan and Kwang (1991) also developed a direct element connect filling (DECF) scheme which employs multiple starting nodes and proceeds with an algorithm similar to Farhat's simultaneously for each starting node to avoid possible domain splitting. The symmetry of the finite element domain is considered in the selection of starting nodes and during the partitioning process.

Simon (1991) compared three partitioning algorithms: recursive coordinate bisection, recursive graph bisection, and recursive spectral bisection algorithms, and shows the superiority of the new spectral bisection

algorithm over the other two. Only element-based partitioning is addressed. The bisection nature of these algorithms is suitable for hypercube computers in which the number of processors is an integer power of two, but it is too restrictive for applications on coarse-grained parallel computing environments consisting of an arbitrary number of processors.

In addition to the comparison similar to that made by Simon (1991), Venkatakrishnan et al. (1991) compared two variants of the partitioning strategies based on each of the three algorithms introduced by Simon. The two variants were referred to as domainwise and stripwise decompositions. It was found that the domainwise partitioning strategies attempt to reduce the transmission costs of interprocess communication, while the stripwise partitioning strategies attempt to reduce the start-up costs of communication.

Recently, Hendrickson and Leland (1992) developed a new spectral graph partitioning algorithm which combines a recursive spectral octasection algorithm with a generalized Kernighan-Lin algorithm. In the example problem studied, it was shown that this algorithm produced better results than the recursive spectral bisection algorithm proposed by Simon (1991). However, the octasection nature of the algorithm is again best suited for hypercube architectures and is not amenable for applications on parallel computing environments consisting of an arbitrary number of processors.

4.2 Theoretical Background and Definitions

For the purpose of discussion of domain partitioning algorithms, it is necessary to introduce some essential concepts and definitions of graph

theory and to establish their relationship to matrix analysis and finite element analysis. These concepts and definitions are presented in this section. Different graph representations of finite element meshes are also discussed.

A graph G = (V, E) consists of a non-null finite set of vertices $V = \{v_1, v_2, \ldots, v_n\}$ together with a set of edges $E = \{e_1, e_2, \ldots, e_m\}$ which are unordered pairs of distinct values from V; $E = \{ (v_i,v_j) \mid v_i \in V, v_j \in V \}$. Edges are elements of the set E and vertices are elements of the set V. A graph obeying this definition is an undirected graph because the set E is comprised of unordered pairs of vertices.

The correspondence between a graph and a matrix can be established by considering a symmetric matrix K of order N with non-null diagonal components k_{ii}. The ordered and undirected graph of K is denoted by $G^K = (V^K, E^K)$. This graph has N vertices numbered from 1 to N, and $\{v_i, v_j\} \in E^K$ if and only if $k_{ij} = k_{ji} \neq 0, i \neq j$, where v_i denotes the node of E^K with label i.

The vertices u and v in G are adjacent vertices if $\{u, v\} \in E$.

The adjacent set of the subset W of the vertices of G, Adj(W), is defined as

$$Adj(W) = \{ u \in (V-W) \mid (u,v) \in E, v \in W, W \subset V \} \tag{4.1}$$

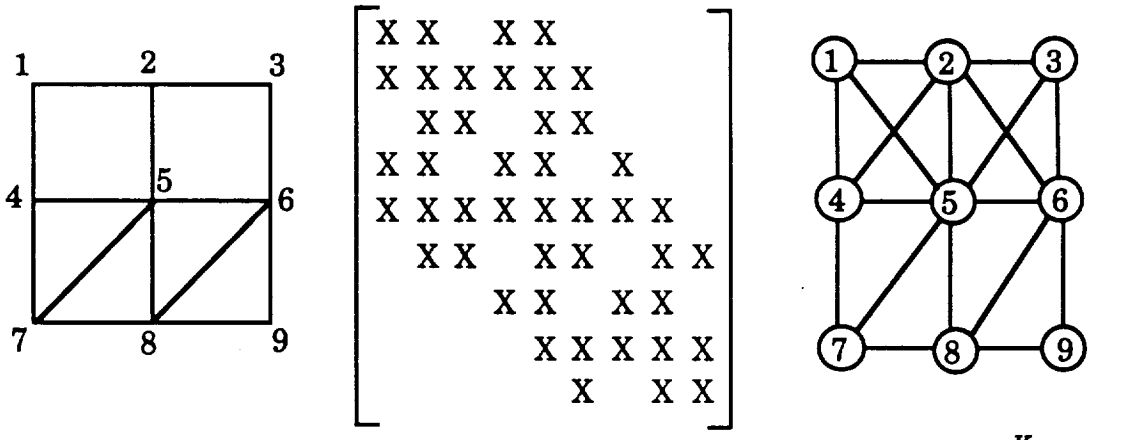
If W = {v}, Adj(W) = Adj(v). The degree of the set W is defined as

$$Deg(W) = \mid Adj(W) \mid \tag{4.2}$$

where $|\cdot|$ denotes the number of components of the set. Similarly, the degree of a vertex v is defined as

$$\text{Deg}(v) = |\ \text{Adj}(v)\ | \tag{4.3}$$

Consider the finite element mesh of Fig. 4.1(a) with nine nodes, four T3 elements, two Q4 elements, and no boundary conditions. For the sake of simplicity, assume one degree of freedom (dof) per node. The stiffness matrix representation associated with this mesh is given in Fig. 4.1(b) by the symmetric matrix K (of order nine) with elements k_{ij}. With respect to Fig. 4.1(c), $V^K = \{ 1, \ldots, 9 \}$ and $E^K = \{ \{1,2\}, \{1,4\}, \{1,5\}, \ldots, \{8,9\} \}$, where each pair $\{v_i, v_j\}$ contains two adjacent vertices. If W = {1, 2, 3}, then Adj(W) = {4, 5, 6} and Deg(W) = 3. For the vertex No. 6, Adj(6) = {2, 3, 5, 8, 9} and Deg(6) = 5.



FEM mesh
(9 nodes; 6 elements)
(a)

Matrix K
(9 x 9)
(b)

Graph G^K
(9 vertices; 18 edges)
(c)

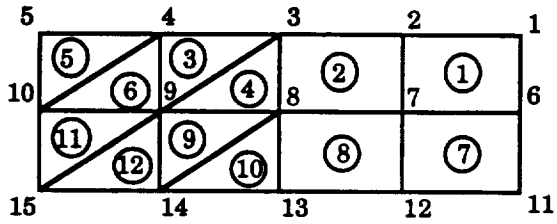
Figure 4.1 Correspondence among mesh, matrix, and graph

Next, consider the finite element mesh of Fig. 4.2(a) with fifteen nodes, eight T3 elements, four Q4 elements, and no boundary conditions. The connectivity of the nodes can be represented topologically by the node graph of Fig. 4.2(b). In order to establish the connectivity of the finite elements in a topological sense, a dual graph representation is used as illustrated by Fig. 4.2(c).

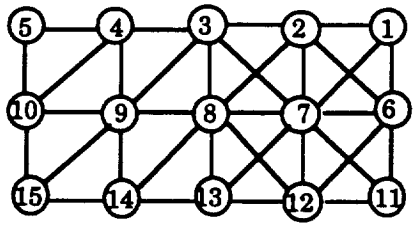
The geometric aspect of the dual graph is used here for the purpose of representing the connectivity of the finite elements in a generic mesh. The nodes in the dual graph represent finite elements in the original mesh. The edges in the dual graph represent adjacent finite elements that share a common boundary in the original mesh. According to this definition, a finite element of dimension n (n = 1, 2, 3) has boundaries of dimension (n-1). As an example, by applying this definition to the finite element mesh of Fig. 4.2(a), the dual graph of Fig. 4.2(c) is obtained.

A spectral method, based on algebraic properties of the graph associated with the finite element mesh, can be used to solve the partitioning problem (Pothen et al. 1990). The spectral method associates an adequate graph representation (G) to the finite element mesh and forms the Laplacian matrix L(G). Here, G is assumed to be a connected graph. A particular eigenvector of this matrix can be used to partition the vertices of the graph into two sets.

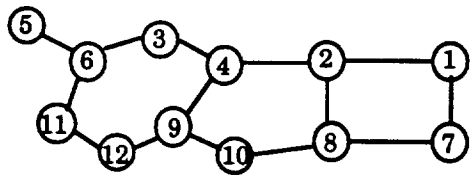
The Laplacian matrix L(G) is a symmetric matrix of order N, where N is the number of the vertices of the graph G. The components l_{ij} of L(G) are defined as



(a) Finite Element Mesh (arbitrary nodal and element numbering)



(b) Node graph G^{A}



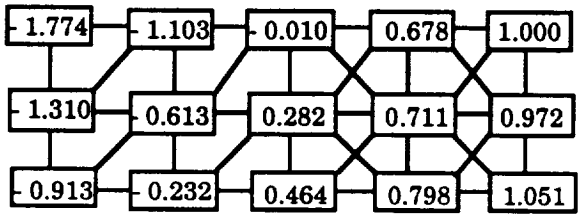
(c) Dual graph G^{*A}

$$
\begin{bmatrix}
-3 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & -5 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & -5 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & -4 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & -2 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & -5 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 1 & -8 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 1 & -7 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & -6 & 1 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & -4 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & -3 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & -5 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & -4 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & -4 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & -3
\end{bmatrix}
$$

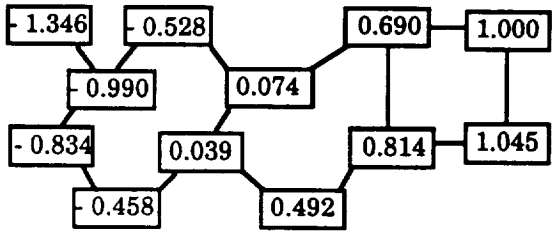
(d) $L(G^{A})$

$$
\begin{bmatrix}
-2 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & -3 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & -2 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & -3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & -3 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & -2 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & -3 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -3 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -2 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -2
\end{bmatrix}
$$

(e) $L(G^{*A})$



(f) Fiedler vector for $L(G^{A})$
(λ_2 = - 0.639)



(g) Fiedler vector for $L(G^{*A})$
(λ_2 = - 0.264)

Figure 4.2 Spectral analysis

$$l_{ij} = \begin{cases} 1 & \text{if and only if } \{v_i, v_j\} \in E \\ -\text{Deg}(v_i) & \text{if and only if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

From this definition, it follows that

$$L(G) = -D(G) + A(G) \tag{4.5}$$

where D(G) is a diagonal matrix of order N with diagonal components $d_{ij} = \text{Deg}(v_i)$ and A(G) is an adjacency matrix of order N with components $a_{ij} = 1$ if $\{v_i, v_j\} \in E$ and zero otherwise. Fig. 4.2(d) shows the Laplacian matrix $L(G^A)$ associated with the graph G^A and Fig. 4.2(e) shows the Laplacian matrix $L(G^{*A})$ associated with the dual graph G^{*A}.

The Laplacian matrix L(G) is negative semidefinite. Let the eigenvalues of L(G) be ordered as $\lambda_1 = 0 \geq \lambda_2 \geq \ldots \lambda_n$. The largest eigenvalue is $\lambda_1 = 0$ and the associated eigenvector $\mathbf{y}_1$ has all of its elements equal to 1. If G is connected, the second eigenvalue is always nonzero and negative. The special properties of the second eigenvalue λ_2 and its corresponding eigenvector $\mathbf{y}_2$ have been studied by Fiedler (1975). The second eigenvector $\mathbf{y}_2$ is called the *Fiedler vector* by Simon (1991). The eigenvalue λ_2 is designated "algebraic connectivity" and is related to the vertex and edge connectivities of the graph. The components of $\mathbf{y}_2$ are associated with the corresponding vertices of the graph. Differences in the values of the components of $\mathbf{y}_2$ give topological distance information about the vertices of the graph. The components of $\mathbf{y}_2$ also provide a weighting for these vertices which can be used for partitioning the graph G. For example, all vertices with weights below the mean weight may be assigned to one partition, and the rest to the other partition.

Figs. 4.2 (f) and (g) show the algebraic connectivity λ_2 and the Fiedler vector $\mathbf{y_2}$ for the graph G^A and the dual graph G^{*A}. In both cases, the mean value of the components of $\mathbf{y_2}$ is zero.

Graph Representation of Finite Element Meshes

One approach to partition a generic finite element mesh is to associate a proper graph representation with the mesh and to partition the graph. There are several ways to associate graphs with meshes.

An advantage of using the dual graph for partitioning of finite element meshes is that the number of edges in the connectivity graph is reduced, leading to a smaller set of data storage (compare Figs. 4.2 (b) and (c)). This is due to the fact that the dual graph defines the connectivity of the finite elements by means of their boundaries, instead of their nodes. However, for framed structures consisting of only 1-D finite elements, the elements are connected to their adjacent elements through 0-D boundary nodes. Therefore, there is no clear advantage in using the dual graph approach for domain partitioning of framed structures.

A disadvantage of the dual graph approach is that the dual graph does not represent the true interprocess communication in parallel finite element analysis. The true interprocess communication occurs across shared nodes in the original finite element mesh, but not shared boundaries (e.g., edges or faces). To better describe the communication pattern, Venkatakrishnan et al. (1991) proposed the use of a communication graph which is defined as follows. The nodes in the communication graph represent finite elements in the original mesh. The edges in the

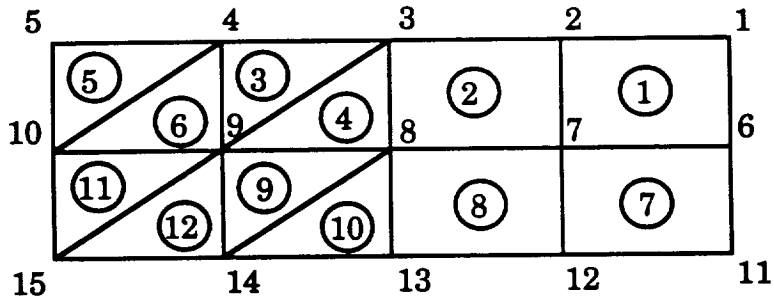
communication graph represent adjacent finite elements that share a common node in the original mesh. Figs. 4.3 (a) and (b) show the same finite element mesh and its associated dual graph (G^{*A}) as used in Figs. 4.2 (a) and (c), respectively. Fig. 4.3 (c) shows the associated communication graph ($G^{\bullet A}$).

Both the dual graph and the communication graph do not differentiate among different types of finite elements. For example, if one or more Q4 elements in Fig. 4.3(a) are replaced by Q8 elements, both the dual graph and the communication graph remain the same as Figs. 4.3 (b) and (c), respectively. Therefore, the partitioning algorithms employing these graphs may not produce partitions with well-balanced computational loads for meshes with mixed finite element types. Nevertheless, the communication graph is expected to provide better results than the dual graph.

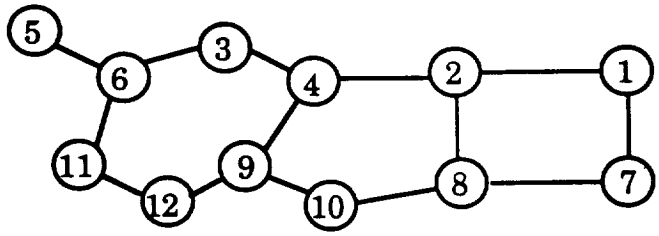
For node-based partitioning, since the focus is on partitioning nodes in finite element meshes, the node graph seems to be a natural choice. The use of the node graph is advantageous for framed structures consisting of only 1-D finite elements because structures of this type usually have a fewer number of nodes than of elements, resulting in less requirement of data storage. However, this becomes a disadvantage for structures consisting of 2D or 3D elements, especially of higher order.

The best choice among different possible graph representations depends on several factors such as the type of partitioning desired (node-based or element-based), the parallel solution algorithms of the dynamic system (explicit or implicit), and the effectiveness and computational cost of

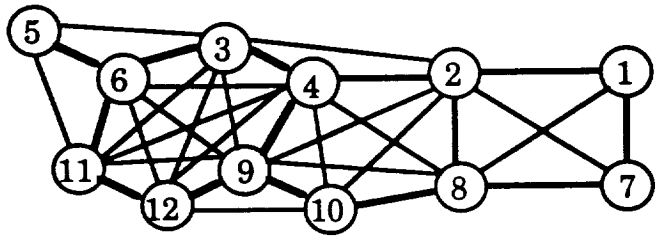
the partitioning algorithm. This topic is further investigated in the present research.



(a) Finite element mesh



(b) Dual graph G^{*A}



(c) Communication graph $G^{\bullet A}$

Figure 4.3 Correspondence among mesh, dual graph, and communication graph

4.3 Domain Partitioning Algorithms

Four existing automatic algorithms for domain partitioning are investigated in this research. They are the algorithms proposed by Farhat (1988), Malone (1988), Al-Nasra and Nguyen (1991), and Simon (1991). This section illustrates the approaches used by these algorithms and discusses their strengths and weaknesses. Some aspects of their implementation in this research are described. Two algorithms have been developed in this research based on extensions to the spectral partitioning method used by Simon (1991). They are also discussed in detail in this section.

4.3.1 Farhat's Algorithm

Farhat (1988) presented an algorithm for automatic decomposition of arbitrary finite element meshes. The algorithm is designed to meet three basic requirements: (a) it must be capable of handling arbitrary finite element meshes; (b) it must generate a set of balanced subdomains (often in terms of number of elements) to ensure as even as possible distribution of overall computational load among processors; (c) it must minimize the amount of interface nodes to reduce the synchronization and/or communication overhead.

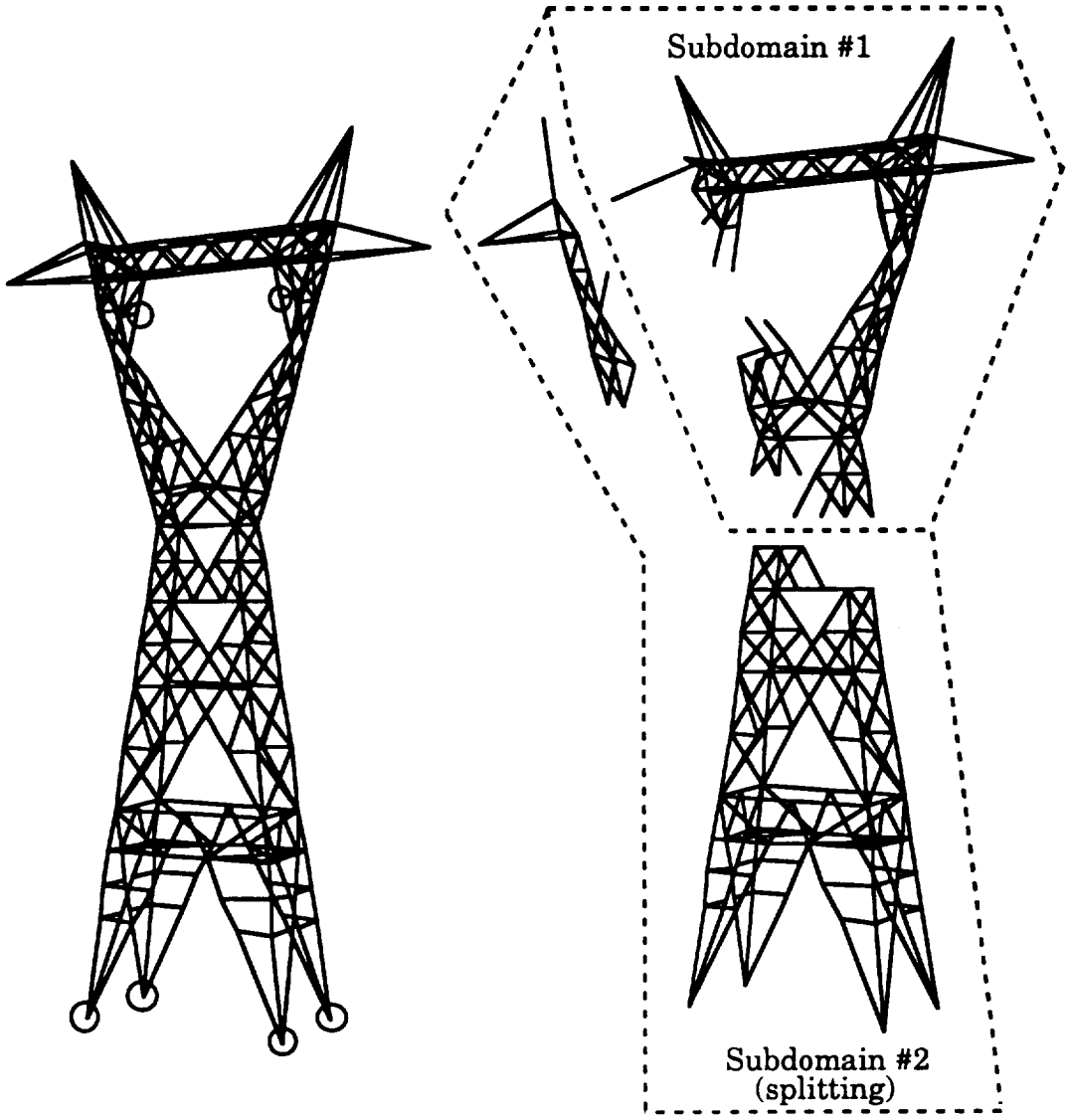
Farhat's partitioning algorithm (here designated FP) is summarized in Table 4.1. The weight of a node is defined as the number of unassigned elements connected to it. The interior boundary of a subdomain is defined as the subset of its boundary that connects to other subdomains. The present implementation uses the routines provided by Farhat (1988) with some corrections (Farhat 1992).

Table 4.1 Farhat's partitioning algorithm (here designated FP) (Farhat 1988)

(1) Locate a node that belongs to the boundary of the previously defined subdomains (for the first time, the whole domain is used) and has a nonzero minimal weight.
(2) Assign unassigned elements that are connected to this node to the current subdomain. Recursively, assign unassigned elements that are adjacent to the elements in the current subdomain to the current subdomain until the number of elements equals to the total number of elements divided by the number of processors.
(3) Repeat (1) and (2) until all subdomains are defined.

According to Farhat (1988), this algorithm is independent of both element and nodal numbering, since only adjacency information of elements is utilized. However, the decomposition obtained from the algorithm is actually not independent of nodal numbering. In Step (1), there may be several nodes all with same minimal weight and the first node encountered with minimal weight is arbitrarily selected. In this case, the initial nodal numbering determines which starting node is to be selected. It has been found that the selection of the node with minimal weight in Step (1), especially the very first one in the algorithm, greatly affects the decomposition result. In some cases, whether domain splitting occurs in the decomposition may depend on the selection in Step (1). For example, in the transmission tower shown in Fig. 4.4(a) (this is the same one shown in Fig. 3.3), the six vertices enclosed by a circle are nodes with minimal weight for the very first step of the algorithm. Fig. 4.4(b) shows the partitioning

results for two subdomains with a splitting situation occurring in subdomain #2. In this case, the algorithm uses the upper right minimum-weighted node as the starting point. If a different nodal numbering is used which makes the algorithm start from any of the four support nodes (minimum-weighted nodes), domain splitting will be avoided for this case.



(a) Transmission tower (b) Partitioning by FP algorithm

Figure 4.4 Partitioning of a transmission tower using FP algorithm

4.3.2 Al-Nasra and Nguyen's Algorithm

Recently, Al-Nasra and Nguyen (1991) presented an algorithm similar to Farhat's FP (1988) for automatic domain decomposition in finite element analysis. They incorporated the geometry information of the finite element meshes in the algorithm as an attempt to avoid domain splitting problems. The algorithm uses the geometry information to identify the overall long and short directions of the structure and then discourages spreading of elements in a subdomain in the long but short direction.

Al-Nasra and Nguyen's partitioning algorithm (here designated ANP) is summarized in Table 4.2. Al-Nasra and Nguyen (1991) stated in the paper that domain splitting problems did not occur in their algorithm for all applications they had tested. However, for some applications the writer tested, the splitting situation did occur. For example, Fig. 4.5(a) shows an 8-bladed turbine disk modelled by solid finite elements, and Fig. 4.5(b) presents the partitioning results for two subdomains obtained by the ANP algorithm. It can be seen that splitting occurs in subdomain #2. Moreover, when the splitting occurs in a subdomain other than the last one, Step (3) of the algorithm may go into an endless search for the minimum-weighted node. Therefore, the present implementation uses the routines provided by Al-Nasra and Nguyen (1991) with a slight modification in Step (3) to avoid the problem of endless search for the minimum-weighted node. The modified Step (3) in the algorithm then becomes

(3) Locate a node that has a nonzero minimal weight and is not located at the boundaries with other subdomains. If all nodes in the current subdomain are not qualified, perform a search among all unassigned nodes in the mesh.

Table 4.2 Al-Nasra and Nguyen's partitioning algorithm (here designated ANP) (Al-Nasra and Nguyen 1991)

(1) Assign an initial weight to each node. The initial weight of a node is defined as the number of elements connected to it.

(2) Adjust the initial weight of each node based on its geometric location in the mesh such that extra weight is added increasingly along the long direction of the mesh.

(3) Locate a node that has a nonzero minimal weight and is not located at the boundaries with other subdomains.

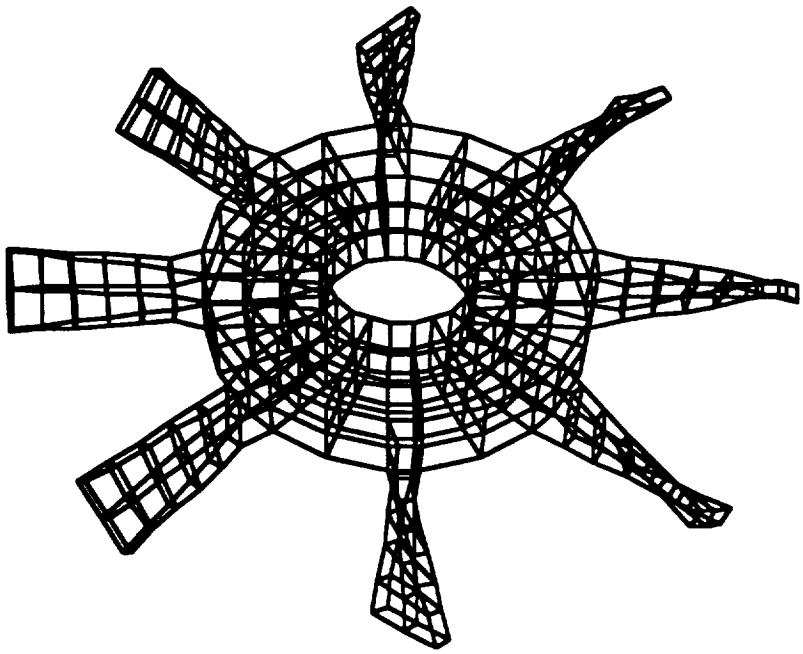
(4) Assign unassigned elements that are connected to this node and their associated nodes to the current subdomain, and reduce the weight of the nodes by one.

(5) Locate a node with minimum weight and with at least one adjacent unassigned element in the current subdomain.

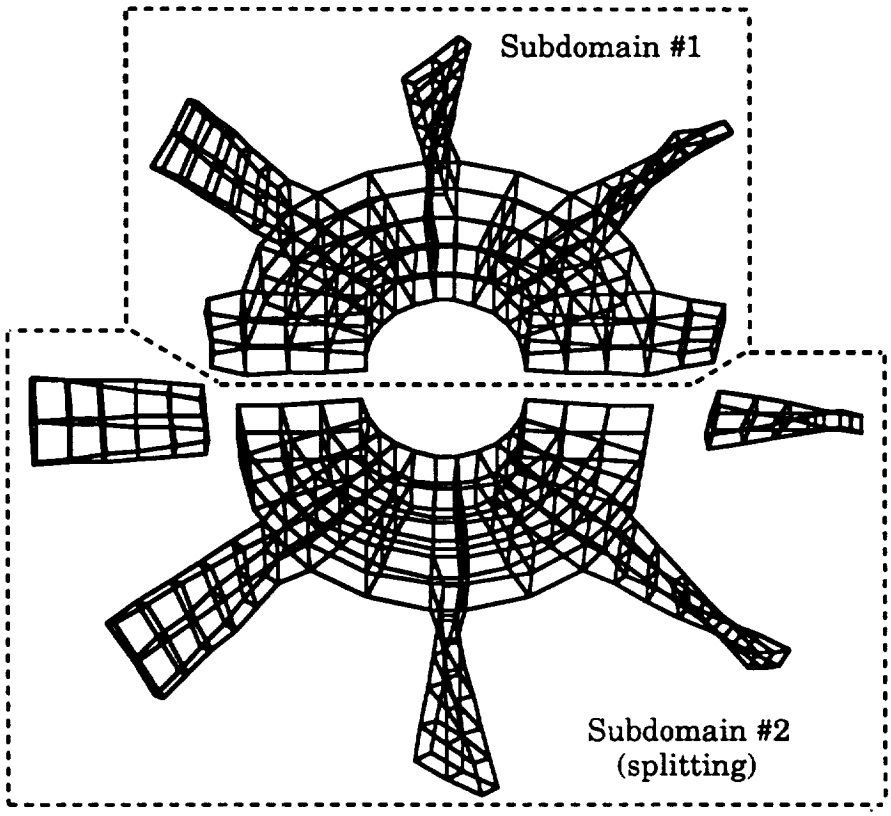
(6) Repeat (4) and (5) until the number of elements equals to the total number of elements divided by the number of processors.

(7) Repeat (3) - (6) until all subdomains are defined.

In addition, the long and short directions defined in the algorithm are in terms of the directions of global axes used to build the finite element model of the structure. This means that different choices of the global axes and different ways of orienting the structure with respect to the chosen global axes may result in different partitioning results. Therefore, caution is needed in preparation of input data for this algorithm.



(a) 8-bladed disk model



(b) Partitioning by ANP algorithm

Figure 4.5 An 8-bladed disk and its partitioning by ANP algorithm

4.3.3 Malone's Algorithm

Malone (1988) also presented an algorithm for automatic decomposition of finite element meshes. The algorithm is called Reduced Bandwidth Decomposition (RBD) algorithm by Malone (1988) and is summarized in Table 4.3.

Table 4.3 Malone's reduced bandwidth decomposition (RBD) algorithm (Malone 1988)

(1) Reduce bandwidth of the matrix representing the nodal connectivities of the finite element mesh.
(2) Reorder elements in ascending sequence of their lowest numbered nodes.
(3) To each processor, assign the elements in order until the number of elements equals to the total number of elements divided by the number of processors.

In Step (1) of the RBD algorithm, it is stated by Malone that a modified version of the Collins' automatic nodal renumbering algorithm for bandwidth reduction (Collins 1973). The present implementation uses a modified version of the Collins algorithm developed by Paulino (1988). However, Collins' algorithm is not as "effective for most meshes" as stated by Malone (1988, pp. 42), who presented only meshes with constant strain triangles (CST). This algorithm was reported to be unsuccessful for meshes with eight-noded quadrilateral elements (Collins 1973). Moreover, all of the examples presented by Collins (1973) are too small when compared to large

meshes that demand parallel analysis. Therefore, if a more efficient and effective heuristic algorithm is used in Step (1) to reduce the bandwidth of the system matrix, better partitioning results may be obtained. However, further research is needed to justify this idea.

There is no guarantee in the RBD algorithm that domain splitting does not occur in the subdomains created. In Fig. 4.6, the space station shown previously in Fig. 3.4 is partitioned by the RBD algorithm into four subdomains. It can be seen that splitting occurs in subdomains #2 and #4.

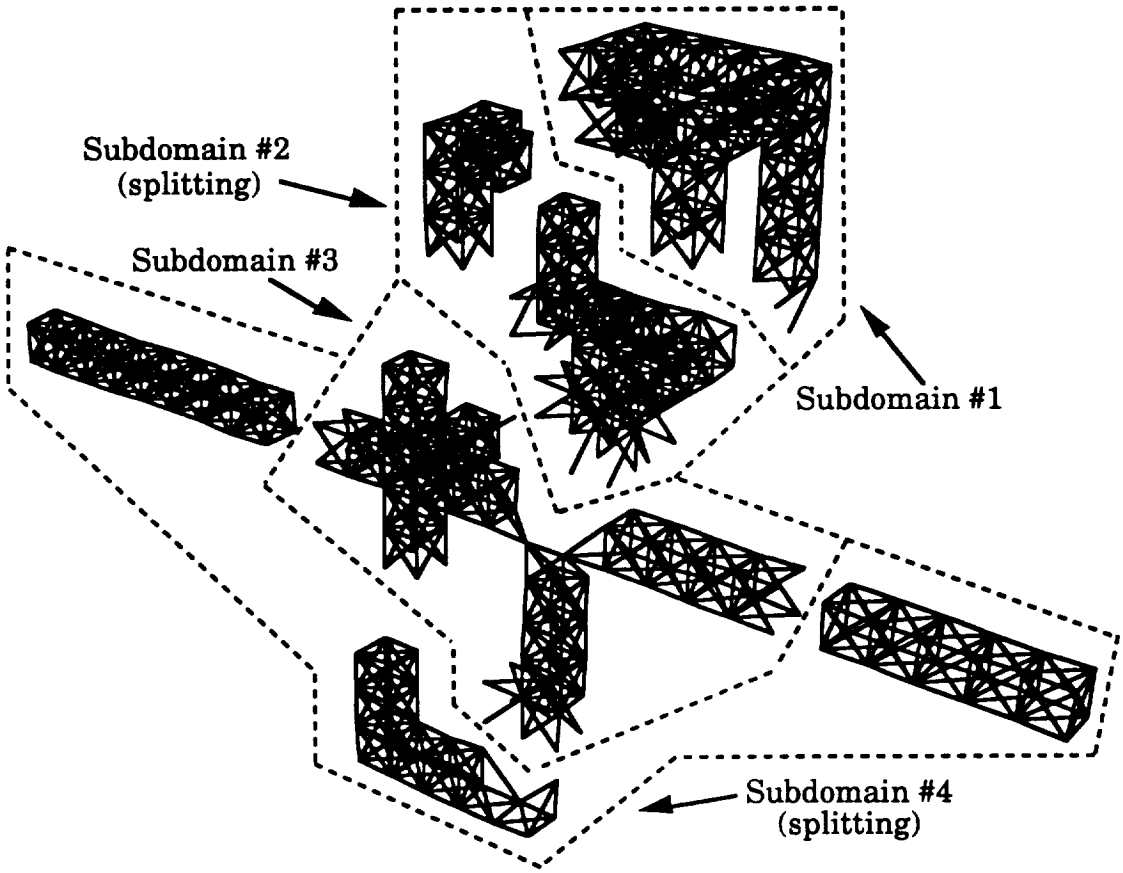


Figure 4.6 Partitioning of the space station shown in Fig. 3.4 by the RBD algorithm.

In the RBD algorithm originally presented by Malone (1988), there is a Step (4) to assign uniquely nodes along the interfaces of subdomains to processors which is required by the parallel dynamic algorithm used by Malone. It is omitted from Table 4.3 because it is not generally needed by parallel algorithms and is not implemented in this research. However, when there is a need to assign uniquely nodes along the interfaces of subdomains to processors, this research uses an approach different from Malone's. The approach is discussed later in this chapter.

4.3.4 Simon's Algorithm

Based on the spectral partitioning method proposed by Pothen et al. (1990), Simon (1991) presented a recursive bisection algorithm for automatic partitioning of unstructured grids. To establish the partitioning of the grid, a dual graph representation has been used. The partitioning algorithm is called Recursive Spectral Bisection (RSB) algorithm by Simon (1991) and is summarized in Table 4.4.

Table 4.4 Simon's recursive spectral bisection (RSB) algorithm (Simon 1991)

(1) Construct the dual graph associated with the finite element mesh.
(2) Compute the second eigenvector of the Laplacian matrix (called the *Fiedler vector* by Simon) of the graph using the Lanczos algorithm.
(3) Sort vertices of the graph according to the value of their associated components in the Fiedler vector.
(4) Assign half of the vertices to each subdomain.
(5) Repeat recursively for each subdomain.

This algorithm has two features that are worth mentioning. First, the dual graph is used to construct the partitioning problem. As discussed previously in Section 4.2, this approach decreases significantly the size of the eigenproblem associated with the Laplacian matrix. As a result, the computational storage and execution time required to solve the eigensystem are reduced. Second, this algorithm employs the global information of the graph provided by the Fielder vector to obtain an edge separator which partitions the graph into two subgraphs of nearly equal size (one vertex difference at most).

Very good partitioning results have been obtained using this algorithm, as reported by Simon (1991). It has also been shown by Pothen et al. (1990) that the separators computed by this algorithm compare quite favorably with separators computed by other previously proposed algorithms.

However, there are some problems associated with the RSB algorithm. First, the dual graph associated with a finite element mesh may be a disconnected graph. This situation may happen when n-dimensional finite elements are not connected through all their (n-1)-dimensional boundaries and when the adjacent finite elements do not have the same geometric dimensions (Fenves and Law 1983). Although theoretically this algorithm is capable of handling non-connected graphs (Pothen et al. 1990), this situation may not only create difficulties and complexities in the partitioning process but also deteriorate the partitioning results.

Second, as discussed in Section 4.2, the dual graph does not represent the true interprocess communication in parallel finite element analysis. To

better describe the communication pattern, the communication graph may be used. In the present work, both the dual graph and communication graph approaches are implemented. For node-based partitioning, the node graph approach is also implemented.

Third, the bisection nature of the RSB algorithm is suitable for hypercube computers in which the number of processors is an integer power of two, but it is too restrictive for computing environments consisting of an arbitrary number of processors, such as the networked workstation environments investigated in this research. In Section 4.3.5, two algorithms developed in this research are presented, both of which generalize the RSB algorithm for an arbitrary number of partitions. They are called by the author the *recursive spectral sequential-cut* (RSS) algorithm and the *recursive spectral two-way* (RST) algorithm, respectively. Detail descriptions of these two algorithms are given in Section 4.3.5.

The last problem associated with the RSB method is related to its computational cost. In general, the solution of the second eigenvector for large meshes can be quite expensive, even when the dual graph approach is used. Venkatakrishnan et al. (1991) have compared the execution time of the RSB algorithm and two other algorithms for a graph with 15,606 vertices on a Silicon Graphics workstation (Iris 4D/70). It is found that the RSB algorithm is "quite expensive" (1,750 seconds for the RSB algorithm, while 4 and 3 seconds for the other two algorithms, respectively) although the performance of the algorithm improves considerably on a vector computer such as the Cray Y-MP because matrix vector products in the Lanczos algorithm can be vectorized. Recently, Barnard and Simon (1992) have developed a fast multilevel implementation of the RSB algorithm

which is reported to attain "about an order-of-magnitude improvement in run time on typical examples." This multilevel approach has not been implemented in the present research.

4.3.5 Present Algorithms

Two algorithms have been developed in this research based on the same spectral partitioning method used by the RSB algorithm. Unlike the RSB algorithm in which the number of partitions is restricted to an integer power of two, both the present algorithms can yield an arbitrary number of partitions.

Recursive Spectral Sequential-Cut (RSS) Partitioning Algorithm

The recursive spectral sequential-cut (RSS) algorithm partitions the graph in such a way that subgraphs are cut out of the original graph one by one (or sequentially) in a recursive fashion. The algorithm is given in Table 4.5. The present implementation allows the use of the dual, communication, or node graph.

Recursive Spectral Two-way (RST) Partitioning Algorithm

Instead of using a bisection approach, the recursive spectral two-way (RST) algorithm uses a two-way partitioning approach which partitions the graph into two parts not necessarily equal in size. The algorithm is given in Table 4.6. The number of vertices in each subdomain D_i when the partitioning task is completed is denoted by m_i. It is computed in advance by sequentially employing the following equation for $i = 1, \ldots, N_p$:

$$m_i = \{ [N - \sum_{j=1}^{i-1} m_j] / [N_p - (i - 1)] \} \ (+ 1 \text{ if remainder} \neq 0)$$

Table 4.5 Recursive spectral sequential-cut (RSS) algorithm

(1) Construct the dual, communication, or node graph desired graph associated with the finite element mesh.
(2) Compute the second eigenvector of the Laplacian matrix (called the *Fiedler vector* by Simon) of the graph using the Lanczos algorithm.
(3) Sort vertices of the graph according to the value of their associated components in the Fiedler vector.
(4) Assign $1/N_p$ of the vertices to one subdomain and the remaining to the other, in which N_p is the number of partitions desired (or number of processors).
(5) Repeat recursively for the larger subdomain in the previous step until all subdomains are defined.

in which N_p is the number of available processors and N is the total number of vertices of the whole domain. The number of partitions desired in the intermediate subdomain in each two-way partitioning step is denoted by np (initially np = N_p for the whole domain). Each intermediate subdomain maintains a list l of subdomain numbers associated with D_i which has been assigned to it (initially the list is $\{1, \ldots, N_p\}$ for the whole domain). The k-th component of this list is denoted by $l(k)$. Figure 4.7 demonstrates the process of partitioning a graph with seventeen vertices into five subdomains using this algorithm.

From Table 4.6 and Fig. 4.7, it should not be difficult to see that the RST algorithm degenerates to the RSB algorithm of Table 4.4 when the number of processors (N_p) is equal to an integer power of two.

Table 4.6 Recursive spectral two-way (RST) algorithm

(1) Construct the dual or communication graph associated with the finite element mesh.

(2) Compute the second eigenvector of the Laplacian matrix (called the *Fiedler vector* by Simon) of the graph using the Lanczos algorithm.

(3) Sort vertices of the graph according to the value of their associated components in the Fiedler vector.

(4) Compute the following integers:

p1 = np/2 (discard remainder)

p2 = np - p1

$$n = \sum_{k=1}^{p1} m_{l(k)}$$

Assign n vertices and the list of the first p1 components in $l(k)$ to one subdomain, and set np = p1 for this subdomain.

Assign the remaining vertices and the list of the remaining components in $l(k)$ to the other subdomain, and set np = p2 for this subdomain.

(5) Repeat recursively for each subdomain with np > 1.

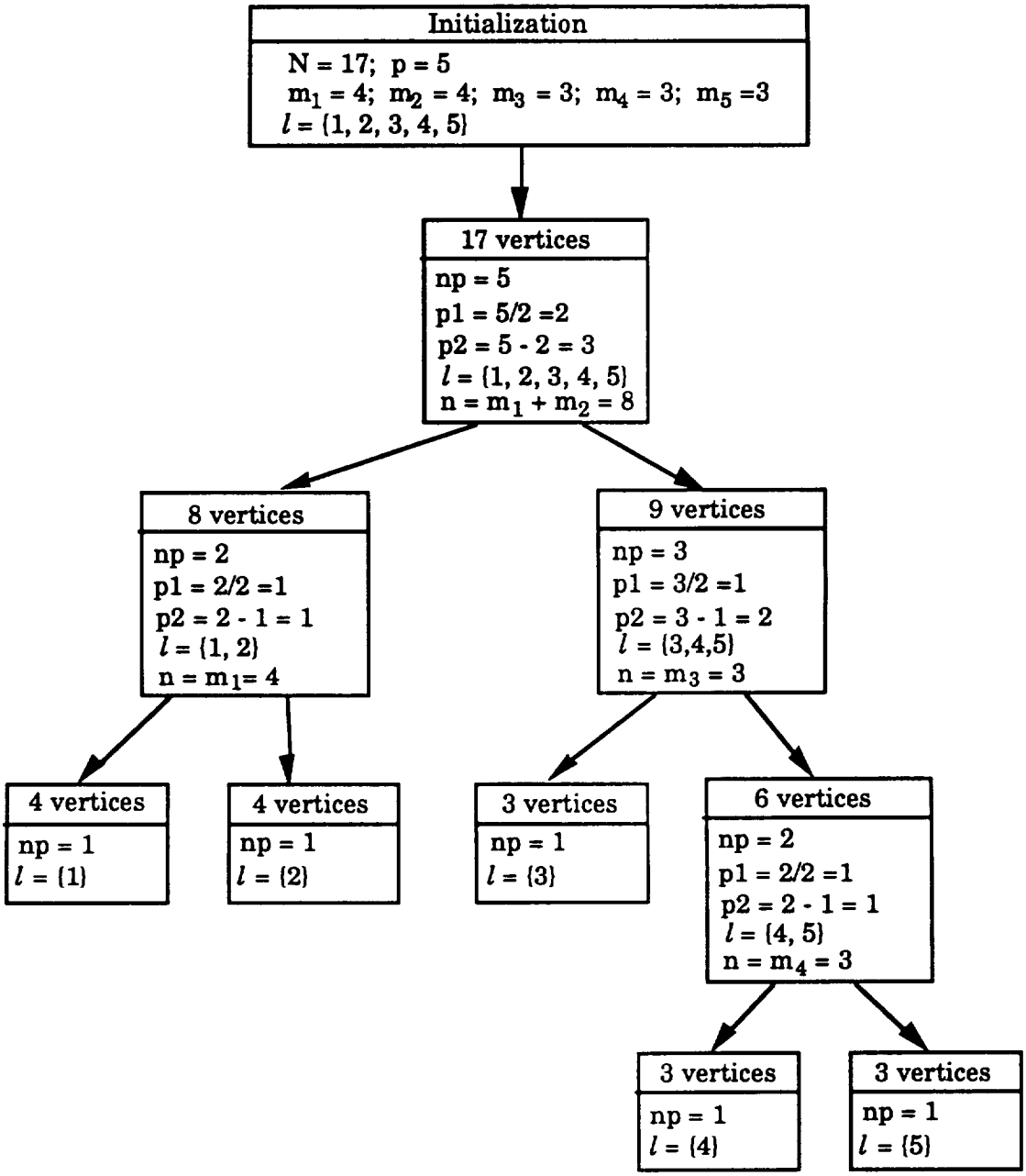


Figure 4.7 Example of the RST partitioning process

4.4 Load Balancing for Parallel Implicit Analysis

There are three major computational components involved in the parallel implicit analysis (also in the parallel steady-state analysis) that require attention for load balancing among processors. First, the element calculations, such as element formation and stress recovery, take a significant portion of the total computational time, especially in nonlinear analyses of structures modelled by 2D and 3D elements. The load balancing of this computation among processors requires an essentially even distribution of elements among subdomains (if a homogeneous parallel environment is assumed and all elements in the mesh are of the same type). In the present implementation, the above load balancing requirement is fulfilled by all of the partitioning algorithms discussed in Section 4.3 because they all assign equal or nearly equal numbers of elements to each subdomain. Although this approach does not in general produce optimal load balancing for meshes with different types of elements, satisfactory results are still obtained for examples studied in the present work for most partitioning algorithms (see examples in Section 4.4.3).

Second, the substructure condensation represented by Eqs. (3.3) - (3.7) may also demand a considerable amount of computational time. At the outset, the profile for each subdomain should be minimized to increase computational efficiency within each subdomain. In addition, the load balancing of this computation requires that the minimized profile of interior degrees of freedom in each subdomain be equal. None of the partitioning algorithms discussed in Section 4.3 attempt to satisfy this load balancing requirement and instead they all focus on the load balancing of element calculations. This is mainly due to the following two reasons: 1) a

complicated and expensive iterative strategy may be required to obtain satisfactory load balancing for condensation because the profile cannot be known until the subdomain is defined, and 2) for nonlinear analyses of structures modelled by 2D and 3D elements (which are the major cases in the present study), it has been found that the load balancing of element calculations often has a more significant effect on parallel efficiency. However, for framed structures modelled by 1D elements, it has also been found that the effect of the load balancing for condensation is dominant.

Third, the efficiency of solving the unknowns along subdomain interfaces by the parallel preconditioned conjugate gradient method greatly affects the overall efficiency of parallel analysis. Although the load balancing of this solution phase requires that all subdomain have an equal number of boundary nodes, it is more important that the total number of boundary nodes shared by subdomains be minimized to reduce the synchronization and communication overhead. The smaller the number of the boundary nodes, the fewer the number of equations involved in the parallel preconditioned conjugate gradient analysis. This not only reduces the size of messages in interprocess communication but also may decrease the number of iterations required in the conjugate gradient analysis. Furthermore, if the total number of boundary nodes is minimized, the workload unbalance due to uneven distribution of boundary nodes among subdomains may be neglected.

All of the partitioning algorithms discussed in the previous section can readily be employed to generate the element-based partitioning, as shown in Fig. 3.2, required by both the parallel implicit and steady-state algorithms. However, after the elements are partitioned among subdomains

(or processors), the partitioning shown in Fig. 3.2 also requires that the elements in each subdomain be labelled as either interior or boundary elements, while the nodes be labelled as interior, primary boundary, or secondary boundary nodes. Moreover, to achieve maximum computational efficiency within each subdomain for the condensation required in the domain decomposition approach, nodal renumbering is needed to reduce either the bandwidth or profile of the subdomain coefficient matrix (see Eq. 3.2). This section briefly describes the present approaches for the element and node labelling and for subdomain renumbering. Furthermore, comparative studies are conducted to evaluate the performance of the partitioning algorithms discussed in the previous section for the element-based partitioning.

4.4.1 Element and Node Labelling

After the elements in the structure are partitioned among subdomains by any of the automatic partitioning algorithms, the nodes in the structure are either labelled as interior nodes if they are shared by elements residing in the same subdomain or labelled as boundary nodes if otherwise. Then, the elements are checked to determine if they are connected to at least one interior node. If they are, they are labelled as interior elements. Otherwise, they are boundary elements.

Next, the boundary nodes are further partitioned into primary and secondary boundary nodes. The following approach is used in the present work:

All boundary nodes are labelled as secondary boundary nodes initially. Then, boundary nodes common to any two subdomains are

labelled as primary boundary nodes in the subdomain which has fewer nodes in the subdomain.

4.4.2 Subdomain Renumbering

As mentioned in Section 3.3.2, most of the matrices involved in the substructure condensation are stored in the skyline format. Although nodal renumbering for the whole domain is performed for bandwidth or profile reduction in BASYS/FRANSYS, this nodal ordering may not result in optimal profile for each subdomain. In this work, the subdomain is therefore renumbered to minimize the computational cost within each subdomain for substructure condensation. Consequently, the overall computational cost of the parallel analysis is reduced.

The present work uses a modified version of the Gibbs-King profile reduction algorithm (Paulino 1988) for subdomain renumbering. This algorithm first finds the endpoints of a pseudo-diameter of the node graph (i.e., a pair of nodes that are at nearly maximal distance apart in the node graph) using the approach proposed by Gibbs et al. (1976). Then, one of the endpoints (arbitrarily selected) is used as the starting node and the nodes in the graph are numbered by a modified King algorithm (King 1970), which reduces the profile of the sparse symmetric matrix associated with the graph. Finally, a test is run to determine if the profile based on new numbering is smaller than that based on the original numbering. If it is not, the original numbering is retained. Otherwise, the new numbering is used.

In the present work, only interior nodes in the subdomain are renumbered. This is due to two reasons. First, the condensation of the

interior nodes (see Eq. 3.1) requires that the interior nodes be numbered before the boundary nodes. The present renumbering algorithm is currently incapable of handling this constraint. Second, the major computational cost in the modified decomposition algorithm used for the condensation is usually the Cholesky decomposition of Eq. 3.5, in which only the interior nodes are involved. Therefore, if the ratio of the number of boundary nodes to the number of interior nodes in each subdomain is kept small (which is usually the case in the present work), renumbering of interior nodes alone should provide sufficient improvement in computational efficiency.

4.4.3 Comparative Studies Among Algorithms

Six structures of different types have been used to evaluate and compare the partitioning algorithms discussed in Section 4.3. They include three framed structures modelled by 1D elements, a building structure modelled by 1D and 2D elements, and two solid structures modelled by 3D elements. Some typical results are reported in this section. The following notations are used in the present study:

N_p = the number of partitions,

N_b = the number of boundary nodes in a subdomain,

N_{e1} = the number of 1D elements in a subdomain,

N_{e2} = the number of 2D elements in a subdomain,

N_{e3} = the number of 3D elements in a subdomain,

N_d = the number of subdomains that have disconnected regions,

Tot(x) = total values of x in the whole domain,

Max(x) = maximum value of x among subdomains,

Min(x) = minimum value of x among subdomains,

CG = communication graph,

DG = dual graph, and

T_{cpu} = the CPU time (sec.) required, which includes time spent in data preparation for the algorithm, execution of the algorithm, and setting results into the database.

Framed Structures Modelled by 1D Elements

Three framed structures of increasing irregularity in geometry have been used to evaluate the performance of different partitioning algorithms. They are the thirty-story building of Fig. 3.5, the transmission tower of Fig. 3.3, and the space station of Fig. 3.4. The results are given in Tables 4.7 - 4.9. In addition, Fig. 4.8 shows the characteristics of the partitionings resulted from different partitioning algorithms on the thirty-story building for $N_p = 4$. It should be noted that the RSB, RSS, and RST algorithms produce the same results in this particular case.

From Tables 4.7 - 4.9. and Fig. 4.8, the following observations are obtained:

(1) Compared to the execution time required by a dynamic analysis, the CPU time spent by all the algorithms are negligible.

(2) As expected, the RST algorithm produces the same results as the RSB algorithm when N_p is an integer power of two.

(3) Except for one case ($N_p = 3$ in Table 4.9), the RST algorithm consistently gives partitions with smallest $Tot(N_b)$. However, even for that case, $Tot(N_b)$ resulted from the RST algorithm is very close to the smallest.

(4) Although it is undesirable to have fragmented subdomains because they usually result in larger $Tot(N_b)$, it should be noted that having nonfragmented subdomains (i.e., $N_d = 0$) is not sufficient in itself to obtain smaller $Tot(N_b)$ (see the case of $N_p = 4$ in Table 4.9).

Table 4.7 Comparison of different algorithms for element-based partitioning of the thirty-story building of Fig. 3.5 ($N_p = 4$)

Parameter	ANP	FP	RBD	RSB	RSS	RST
T_{cpu}	7	3	2	7	9	8
$Tot(N_b)$	51	58	51	48	48	48
$Max(N_b)$	34	39	34	32	32	32
$Min(N_b)$	17	19	17	16	16	16
N_d	0	0	0	0	0	0
$Max(N_{el})$	300	300	300	300	300	300
$Min(N_{el})$	300	300	300	300	300	300

Table 4.8 Comparison of different algorithms for element-based partitioning of the transmission tower of Fig. 3.3 ($N_p = 3$)

Parameter	ANP	FP	RBD	RSS	RST
T_{cpu}	0.9	0.5	0.3	1.9	1.7
$Tot(N_b)$	13	29	24	13	13
$Max(N_b)$	13	28	24	13	13
$Min(N_b)$	4	16	9	4	5
N_d	0	1	0	0	0
$Max(N_{el})$	145	145	145	145	145
$Min(N_{el})$	144	144	144	144	144

Table 4.9 Comparison of different algorithms for element-based partitioning of the space station of Fig. 3.4

N_p	Parameter	ANP	FP	RBD	RSB	RSS	RST
3	T_{cpu}	7	3	2	--	15	14
	$Tot(N_b)$	36	51	47	--	38	38
	$Max(N_b)$	28	45	47	--	28	28
	$Min(N_b)$	17	25	20	--	20	20
	N_d	0	1	1	--	0	0
	$Max(N_{e1})$	476	476	476	--	476	476
	$Min(N_{e1})$	476	476	476	--	476	476
4	T_{cpu}	7	3	2	15	18	15
	$Tot(N_b)$	78	55	68	44	50	44
	$Max(N_b)$	58	36	47	32	37	32
	$Min(N_b)$	28	17	21	16	17	16
	N_d	0	2	2	1	1	1
	$Max(N_{e1})$	357	357	357	357	357	357
	$Min(N_{e1})$	357	357	357	357	357	357

A Building Structure Modelled by 1D and 2D Elements

The twelve-story L-shaped building of Fig. 3.9 has been partitioned using different algorithms. The results are reported in Table 4.10. As discussed previously, the present implementation of all the algorithms assumes that all elements in the mesh are of the same type. However, in this example which represents a mixture of 1D and 2D elements, most algorithms still produce a satisfactorily balanced distribution of elements among subdomains. The exceptions are the RSS(DG), RSB(DG), and RST(DG) algorithms. In addition, both the RSS(CG) and RST(CG) give partitions with smallest $Tot(N_b)$ for both cases.

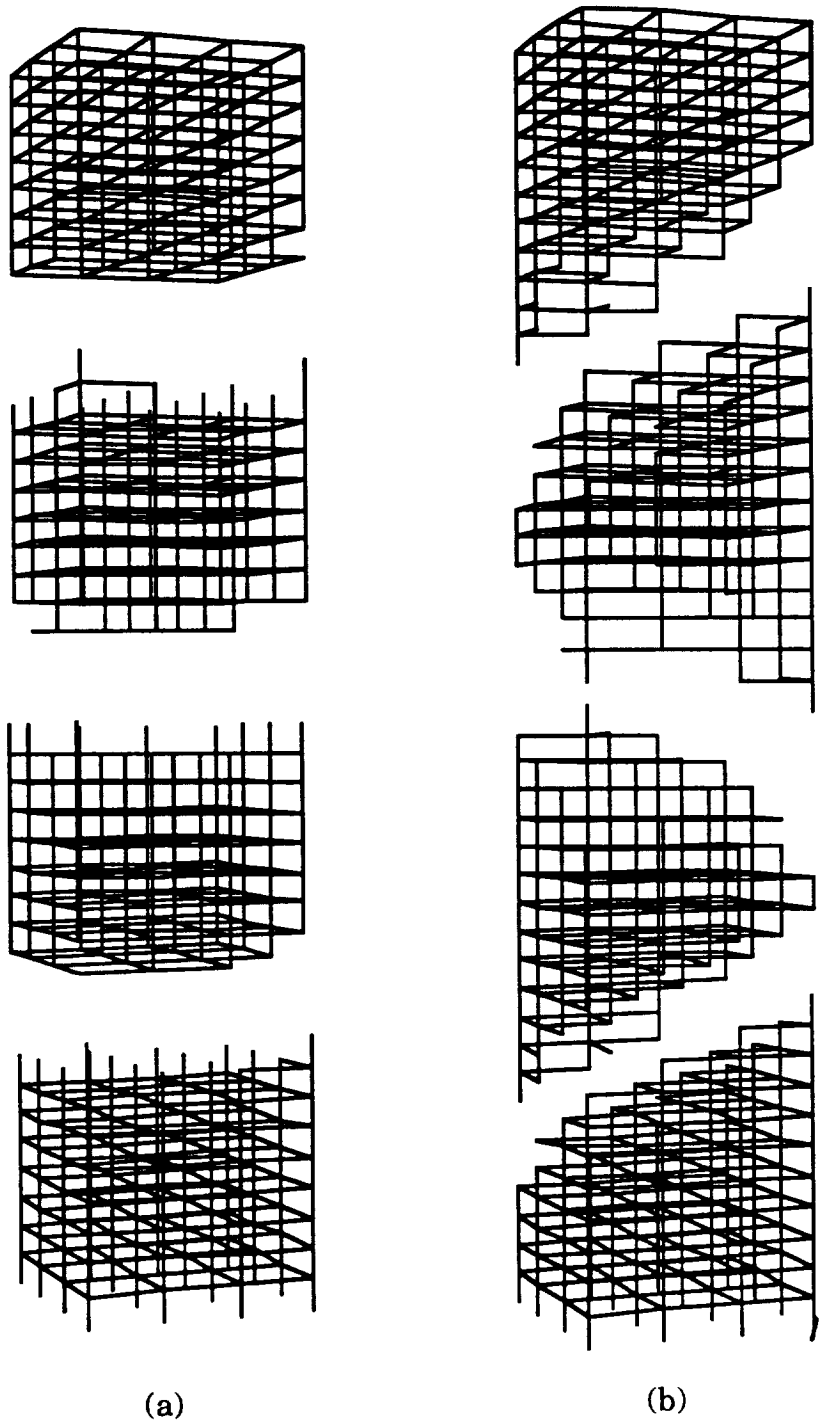
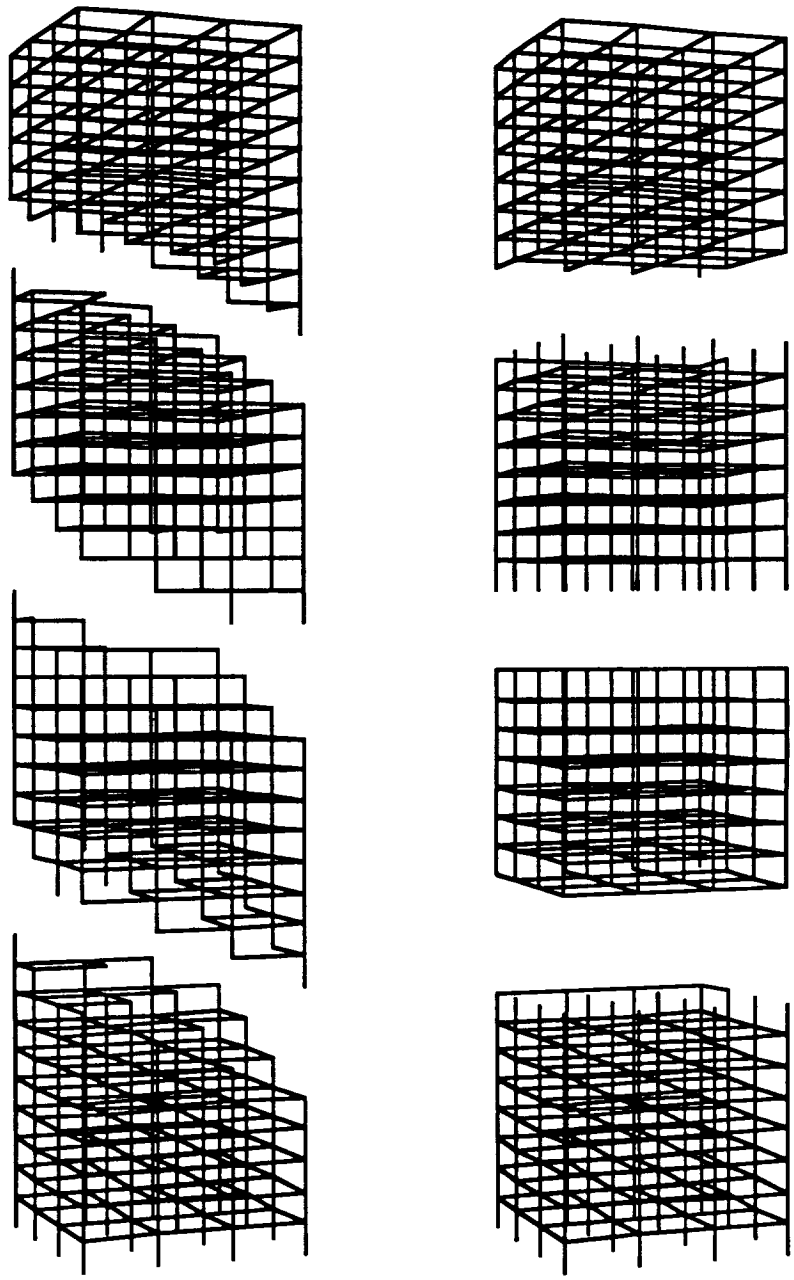


Figure 4.8 Partitionings of the thirty-story building of Fig. 3.5 by the (a) ANP, (b) FP, (c) RBD, and (d) RSB, RSS, or RST algorithms



(c) (d)

Figure 4.8 (Continued)

Table 4.10 Comparison of different algorithms for element-based partitioning of the 12-story L-shaped building of Fig. 3.9

N_p	Parameter	ANP	FP	RBD	RSB (DG)	RSB (CG)	RSS (DG)	RSS (CG)	RST (DG)	RST (CG)
3	T_{cpu}	3	2	2	--	--	3	3	3	3
	$Tot(N_b)$	30	39	38	--	--	139	28	139	28
	$Max(N_b)$	30	38	38	--	--	139	28	139	28
	$Min(N_b)$	15	20	19	--	--	75	14	75	14
	N_d	0	0	0	--	--	1	0	1	0
	$Max(N_{e1})$	156	156	159	--	--	180	156	180	156
	$Min(N_{e1})$	156	156	153	--	--	108	156	108	156
	$Max(N_{e2})$	24	476	476	--	--	72	24	72	24
	$Min(N_{e2})$	24	476	476	--	--	0	24	0	24
4	T_{cpu}	3	2	2	3	4	4	4	3	4
	$Tot(N_b)$	46	50	57	155	42	155	42	155	42
	$Max(N_b)$	31	32	38	156	28	156	28	156	28
	$Min(N_b)$	15	19	19	56	14	56	14	56	14
	N_d	0	0	0	1	0	1	0	1	0
	$Max(N_{e1})$	117	120	120	135	117	135	117	135	117
	$Min(N_{e1})$	117	113	114	63	117	63	117	63	117
	$Max(N_{e2})$	18	22	21	72	18	72	18	72	18
	$Min(N_{e2})$	18	15	15	0	18	0	18	0	18

Solid Structures Modelled by 3D Elements

Two solid structures have also been used in the present comparative studies. The first one is the twelve-bladed turbine disk of Fig. 3.7, while the second one is the turbine blade of Fig. 4.9 which was created by Wawrzynek (1991) for his work. The partitioning results are given in Tables 4.11 and 4.12. In addition, some typical partitionings on the twelve-bladed disk for the case of $N_p = 3$ are shown in Fig. 4.10.

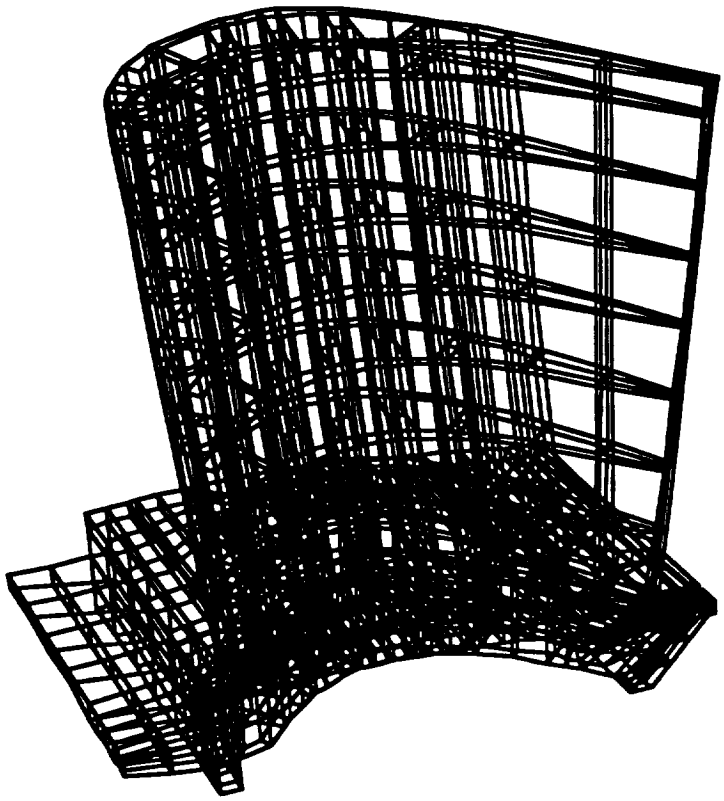


Figure 4.9 A finite element model of a turbine blade with 944 20-noded elements and 6,427 nodes

From Tables 4.11 and 4.12, the following observations are obtained:

(1) Unlike the previous examples, all the spectral methods (i.e., the RSB, RSS, and RST algorithms), which require eigensolutions, deliver partitions in a shorter CPU time than the non-spectral algorithms (i.e., the ANP, FP, and RBD algorithms).

(2) In most cases studied, the DG approach produces smaller $Tot(N_b)$ than the CG approach for the RSS algorithm, while the CG approach produces smaller $Tot(N_b)$ than the DG approach for the RST algorithm.

(3) For the bladed disk example, the optimal partitioning for $N_p = 3$ or $N_p = 6$ seem to be a trivial task for the human. However, it is

not the case for the partitioning algorithms. Only the RSS(DG), RST(DG), and RST(CG) algorithms deliver the optimal solutions.

(3) Except for the case of N_p = 8 in Table 4.12, the RST(CG) algorithm gives partitions with smallest *Tot*(N_b).

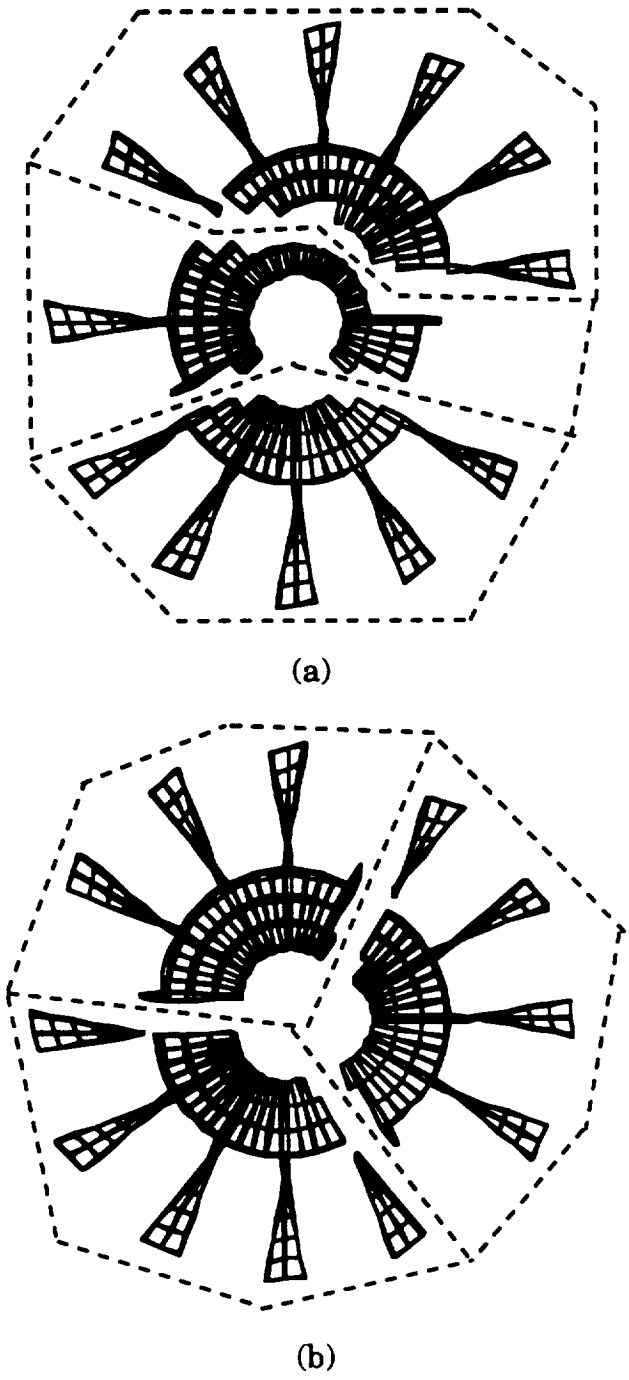
Table 4.11 Comparison of different algorithms for element-based partitioning of the 12-bladed turbine disk of Fig. 3.7

N_p	Parameter	ANP	FP	RBD	RSB (DG)	RSB (CG)	RSS (DG)	RSS (CG)	RST (DG)	RST (CG)
3	T_{cpu}	74	49	123	--	--	40	40	39	38
	Tot(N_b)	395	142	194	--	--	87	87	87	87
	Max(N_b)	395	100	194	--	--	58	58	58	58
	Min(N_b)	116	92	89	--	--	58	58	58	58
	N_d	1	2	2	--	--	0	0	0	0
	Max(N_{e3})	168	168	168	--	--	168	168	168	168
	Min(N_{e3})	168	168	168	--	--	168	168	168	168
6	T_{cpu}	75	48	120	--	--	40	41	39	38
	Tot(N_b)	724	268	481	--	--	174	258	174	174
	Max(N_b)	379	97	210	--	--	58	92	58	58
	Min(N_b)	130	84	97	--	--	58	74	58	58
	N_d	1	4	5	--	--	0	3	0	0
	Max(N_{e3})	84	84	84	--	--	84	84	84	84
	Min(N_{e3})	84	84	84	--	--	84	84	84	84
8	T_{cpu}	76	48	121	39	41	42	42	41	39
	Tot(N_b)	883	369	645	304	276	308	387	304	276
	Max(N_b)	473	113	189	76	69	79	121	76	69
	Min(N_b)	129	69	92	76	69	76	81	76	69
	N_d	1	5	7	4	0	3	4	4	0
	Max(N_{e3})	63	63	63	63	63	63	63	63	63
	Min(N_{e3})	63	63	63	63	63	63	63	63	63

Table 4.12 Comparison of different algorithms for element-based partitioning of the turbine blade of Fig. 4.9

N_p	Parameter	ANP	FP	RBD	RSB (DG)	RSB (CG)	RSS (DG)	RSS (CG)	RST (DG)	RST (CG)
6	T_{cpu}	241	129	278	--	--	117	126	124	123
	$Tot(N_b)$	1301	1055	1127	--	--	881	868	839	821
	$Max(N_b)$	589	534	516	--	--	435	381	382	377
	$Min(N_b)$	216	210	203	--	--	169	180	166	160
	N_d	2	1	0	--	--	0	0	0	0
	$Max(N_{e3})$	158	158	158	--	--	158	158	158	158
	$Min(N_{e3})$	157	157	157	--	--	157	157	157	157
8	T_{cpu}	243	128	272	113	118	119	125	113	118
	$Tot(N_b)$	1430	1255	1460	1138	1179	989	1043	1138	1179
	$Max(N_b)$	495	491	504	444	460	374	365	444	460
	$Min(N_b)$	194	174	174	195	155	144	143	195	155
	N_d	1	1	0	0	0	0	0	0	0
	$Max(N_{e3})$	118	118	118	118	118	118	118	118	118
	$Min(N_{e3})$	118	118	118	118	118	118	118	118	118

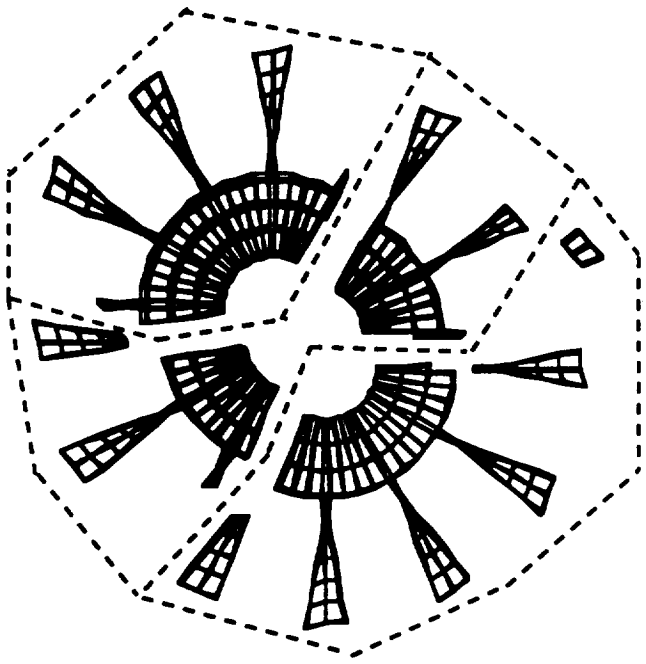
To examine the effect of different partitions on the performance of the parallel analysis, actual parallel steady-state analyses have been carried out on the bladed-disk of Fig. 3.7 partitioned into three subdomains by the FP and RST(CG) algorithm. The partitioning results of these two algorithms have already been given in Table 4.11. It can be seen that the total number of boundary nodes in the partitions obtained from the FP algorithms is almost twice as large as that obtained from the RST(CG) algorithm. In addition, two subdomains in the partitions produced by the FP algorithm have disconnected regions. The results of parallel analyses on DECsystem 5000's are reported in Table 4.13.



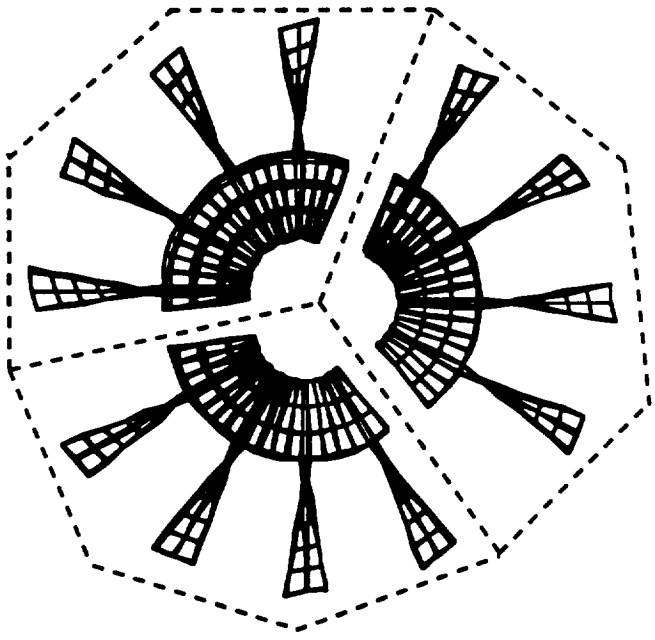
(a)

(b)

Figure 4.10 Partitioning results of the (a) ANP, (b) FP, (c) RBD, (d) RST algorithms for the 12-bladed turbine disk of Fig. 3.7 (N_p = 3)



(c)



(d)

Figure 4.10 (Continued)

Table 4.13 Performance on DEC5000's of the parallel steady-state analysis of the bladed-disk problem of Fig. 3.7 with domain partitioned by the FP and RST(CG) algorithms (N_p = 3)

Algorithm	Speed-up	Efficiency (%)	Average # of conjugate gradient iterations	Total # of dof's on the subdomain interfaces
FP	0.51	17	270	429
RST(CG)	2.03	68	46	216

Summary of Results

The results of the above studies are summarized as follows:

(1) All the algorithms deliver partitions in a very small amount of time compared to the computational time of a dynamic analysis.

(2) Generally speaking, the spectral algorithms (i.e., the RSB, RSS, and RST algorithms) give better results than the non-spectral algorithms (i.e., the ANP, FP, and RBD algorithms).

(3) In most cases, the RST(CG) algorithm gives the best partitioning results among all the considered algorithms.

4.5 Load Balancing for Parallel Explicit Analysis

In Section 4.3, only the spectral partitioning algorithms with the node graph approach have addressed the node-based partitioning. For those algorithms which address only element-based partitioning, the present work performs a simple extra step at the end of the algorithms to assign uniquely common boundary nodes of two or more subdomains to a subdomain so that

the algorithms can be also used for node-based partitioning. In this extra step, boundary nodes common to any two subdomains are assigned to the subdomain which has fewer nodes in its subdomain.

In the rest of this section, a brief description is given first for the labelling of the elements and nodes in each subdomain required by the parallel explicit algorithm (as shown in Fig. 3.1). Results of comparative studies conducted to evaluate the performance of different partitioning algorithms for the node-based partitioning are then reported.

4.5.1 Element and Node Labelling

After the nodes in the structure are partitioned among subdomains, the elements in the structure are either labelled as interior elements if all of their nodes reside in the same subdomain or labelled as border elements if otherwise. For a particular subdomain, all of its nodes are initially labelled as interior nodes. Then, the nodes are checked to determine if they are connected to at least one border element. If they are, they are also labelled as boundary nodes. Once the boundary nodes are identified, all of the nodes of the border elements connected to these boundary nodes are checked to determine if they are residing in this particular subdomain. If they are not, they are labelled as adjacent nodes of this subdomain.

4.5.2 Comparative Studies Among Algorithms

Numerical comparative studies among the partitioning algorithms discussed in Section 4.3 have been conducted for node-based partitioning using three structures of different types: (a) the space station of Fig. 3.4 consisting of only 1D elements, (b) the twelve-story L-shaped building of

Fig. 3.9 consisting of both 1D and 2D elements, and (c) the twelve-bladed turbine disk of Fig. 3.7 consisting of 3D elements. The partitioning results are given in Tables 4.14 through 4.16. The following notations are used in these tables:

N_p = the number of partitions,

N_b = the number of boundary nodes in a subdomain,

N_i = the number of nodes in a subdomain,

N_{e1} = the number of 1D elements in a subdomain,

N_{e2} = the number of 2D elements in a subdomain,

N_{e3} = the number of 3D elements in a subdomain,

N_d = the number of subdomains that have disconnected regions,

Sum(x) = summation of values of x over all subdomains,

Max(x) = maximum value of x among subdomains,

Min(x) = minimum value of x among subdomains,

NG = node graph,

CG = communication graph,

DG = dual graph, and

T_{cpu} = the CPU time (sec.) required.

Sum(N_b) is related to the amount of interprocess communication, while *Max*(N_b)/*Min*(N_b) shows the balancing of communication loads among processors. *Max*(N_i)/*Min*(N_i) and *Max*(N_e)/*Min*(N_e) indicate the balancing of the computational loads of equation solving and element calculations, respectively, among processors. T_{cpu} includes time spent in data preparation for the algorithm, execution of the algorithm, and setting results into the database.

Intuitively, N_d is a significant parameter because, generally speaking, compact nonfragmented subdomains would appear to lead to the most efficient parallelization. In this connection, Fig. 4.11 shows some typical partitionings of the twelve-bladed turbine disk of Fig. 3.7 for $N_p = 4$. It should be noted that border elements shared by adjacent subdomains are included in each subdomain shown.

From Tables 4.14 - 4.16 and Fig. 4.11, the following observations can be obtained:

(1) In all examples studied, the CPU time spent by all the algorithms is very small compared to the execution time of a dynamic analysis.

(2) In the example of the space station, the node graph approach produces better results than the dual and communication approaches for the spectral algorithms. Among all the considered algorithms, the RSS(NG) algorithm gives the best results.

(3) In the example of the twelve-story L-shaped building, the RSS(CG) and RST(CG) algorithms give partitions with smallest *Sum*(N_b), while the RST(NG) algorithm produces partitions with most balanced N_b, N_i, N_{e1}, and N_{e2} among subdomains. Among all the considered algorithms, the RSS and RST algorithms with the dual graph approach (i.e., the RSS(DG) and RST(DG) algorithms) give the worst results.

(4) In the example of the twelve-bladed disk, although the optimal partitioning into four or six subdomains seem to be a trivial task for the human, it is not the case for the partitioning algorithms. Only the RSS(DG), RSS(NG), RSB(DG), RSB(CG), RST(DG), and

RST(CG) algorithms give the optimal solution for the case of N_p = 4. The RSS(DG), RST(DG), and RST(CG) algorithms also give the optimal solution for the case of N_p = 6. For the case of N_p = 8 which does not have a trivial optimal solution, the RST(NG) algorithm gives the best results among all the considered algorithms. In addition, the communication graph approach produces better or at least not worse partitioning results than the dual graph approach for the RSB and RST algorithms, while it is opposite for the RSS algorithm.

(5) Although it is undesirable to have fragmented subdomains because they usually result in longer subdomain boundaries (see Figs. 4.11(a) & (c)), it should be noted that having nonfragmented subdomains (i.e., N_d = 0) is not sufficient in itself to obtain shorter subdomain boundaries (see Table 4.16 and Figs. 4.11(a) & (b)).

(6) In all the cases studied, the RST(CG) algorithm, which consistently delivers good partitioning results, has the best overall performance among all the considered algorithms.

To show the effect of different partitions on the performance of the parallel central difference method, actual parallel analyses have been carried out on the bladed-disk model (shown in Fig. 3.7) partitioned into four and six subdomains by different algorithms. The partitioning results have already been given in Table 4.16. The parallel analysis results for DECsystem 5000's are given in Table 4.17. It can be seen that the results in Table 4.17 verify the trends observed in Table 4.16.

Table 4.14 Comparison of different algorithms for node-based partitioning of the space station of Fig. 3.4 ($N_p = 3$)

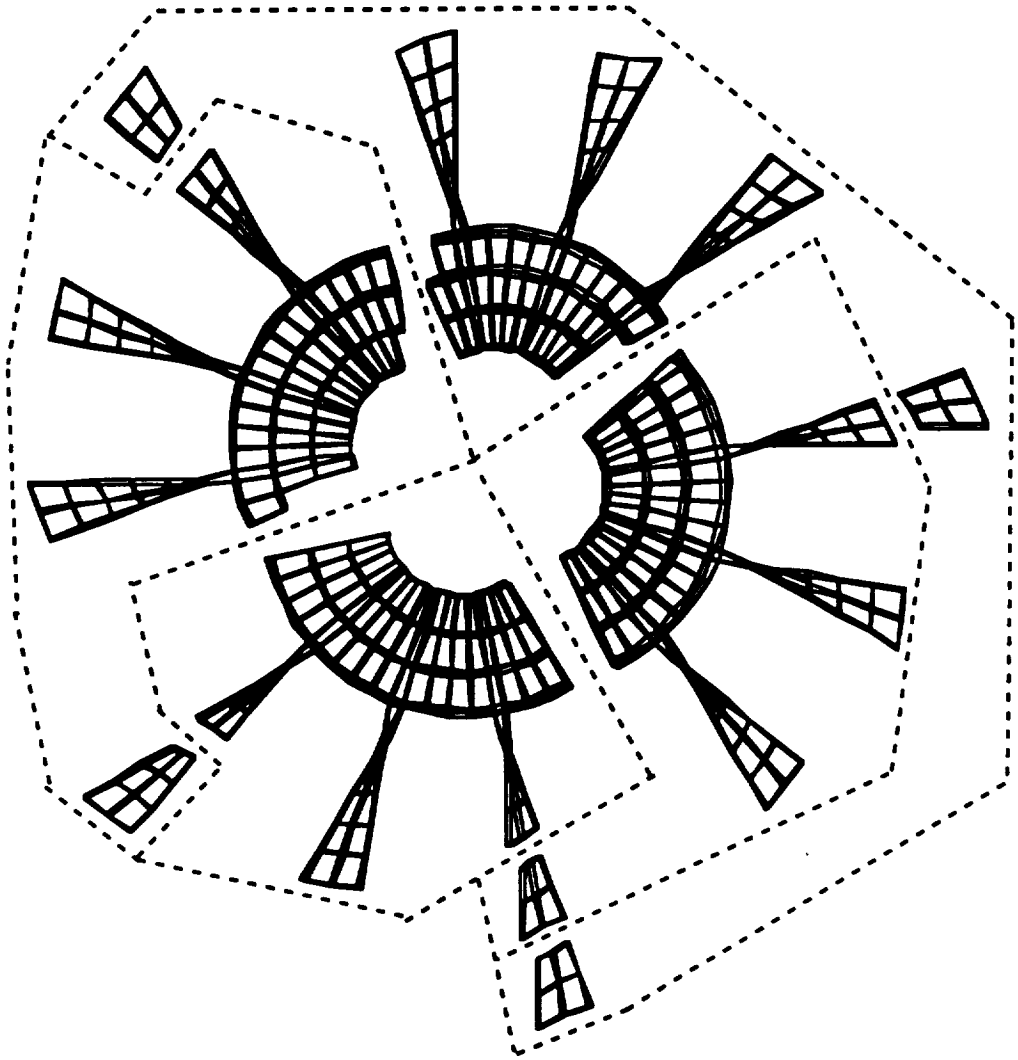
Parameter	ANP	FP	RBD	RSS		RST	
				NG	DG & CG	NG	DG & CG
T_{cpu}	7	2	2	2	16	2	16
$Sum(N_b)$	98	97	83	72	83	75	83
$Max(N_b)$	45	42	42	29	39	29	39
$Min(N_b)$	17	18	18	18	18	20	18
$Max(N_i)$	151	153	139	132	156	132	156
$Min(N_i)$	119	119	117	120	115	120	115
$Max(N_{e1})$	565	568	548	521	605	523	605
$Min(N_{e1})$	473	476	474	497	456	494	456
N_d	0	0	1	0	0	0	0

Table 4.15 Comparison of different algorithms for node-based partitioning of the twelve-story L-shaped building of Fig. 3.9 ($N_p = 3$)

Parameter	ANP	FP	RBD	RSS			RST		
				NG	DG	CG	NG	DG	CG
T_{cpu}	3	1	1	3	3	3	3	3	3
$Sum(N_b)$	98	137	134	90	439	81	87	439	81
$Max(N_b)$	49	60	67	39	290	39	34	290	39
$Min(N_b)$	14	27	19	18	64	14	19	64	14
$Max(N_i)$	205	226	223	218	474	223	214	474	223
$Min(N_i)$	176	187	182	175	232	170	176	232	170
$Max(N_{e1})$	181	195	193	207	211	215	204	211	215
$Min(N_{e1})$	154	151	152	153	95	137	155	95	137
$Max(N_{e2})$	29	31	31	30	72	30	29	72	30
$Min(N_{e2})$	24	28	27	24	34	24	24	34	24
N_d	0	0	0	0	1	0	0	1	0

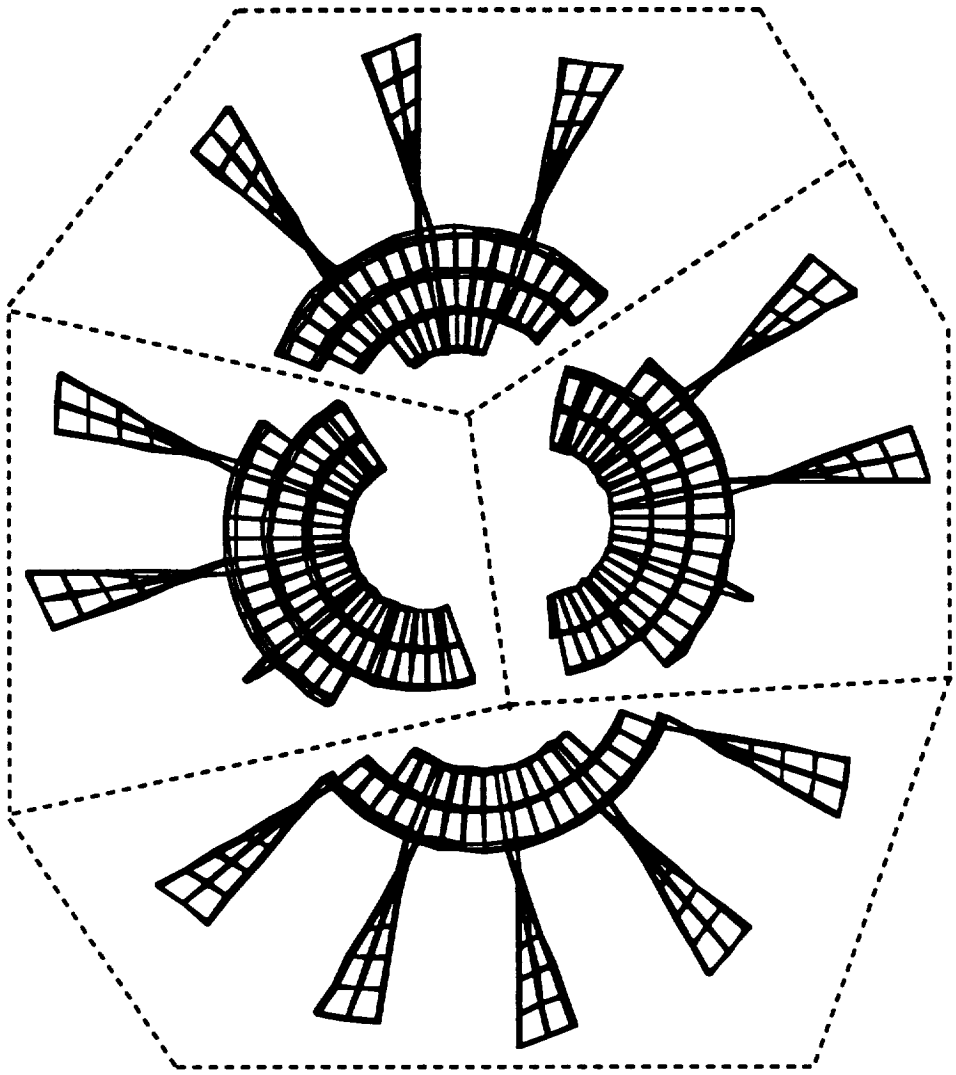
Table 4.16 Comparison of different partitioning algorithms on the bladed-disk problem of Fig. 3.7

N_p	Parameter	FP	RBD	ANP	RSS			RST & RSB($N_p \neq 6$)		
					NG	DG	CG	NG	DG	CG
4	T_{cpu}	37	109	64	139	28	27	123	27	26
	Sum(N_b)	484	637	826	280	280	426	356	280	280
	Max(N_b)	161	237	261	70	70	131	89	70	70
	Min(N_b)	92	92	184	70	70	79	89	70	70
	Max(N_i)	1141	1271	1292	1027	1027	1101	1046	1027	1027
	Min(N_i)	1033	1044	1018	1027	1027	1023	1046	1027	1027
	Max(N_{e3})	146	160	170	132	132	144	134	132	132
	Min(N_{e3})	126	126	126	132	132	126	134	132	132
	N_d	3	3	0	0	0	1	0	0	0
6	T_{cpu}	37	110	66	190	28	29	145	29	28
	Sum(N_b)	624	1107	1569	675	420	624	566	420	420
	Max(N_b)	121	246	378	118	70	131	130	70	70
	Min(N_b)	89	97	154	105	70	79	80	70	70
	Max(N_i)	766	880	1219	768	708	779	768	708	708
	Min(N_i)	722	756	747	740	708	725	718	708	708
	Max(N_{e3})	94	104	164	98	90	100	98	90	90
	Min(N_{e3})	90	84	84	88	90	84	88	90	90
	N_d	3	5	1	2	0	3	1	0	0
8	T_{cpu}	38	111	69	234	31	33	148	29	27
	Sum(N_b)	908	1567	1722	795	759	947	672	777	740
	Max(N_b)	140	246	361	126	116	149	94	130	116
	Min(N_b)	81	92	136	82	76	81	66	76	69
	Max(N_i)	699	711	1051	594	610	635	572	711	613
	Min(N_i)	545	573	581	553	537	542	544	531	529
	Max(N_{e3})	86	84	125	74	78	84	74	87	80
	Min(N_{e3})	65	63	63	66	63	63	66	63	63
	N_d	5	7	1	3	3	4	0	4	0



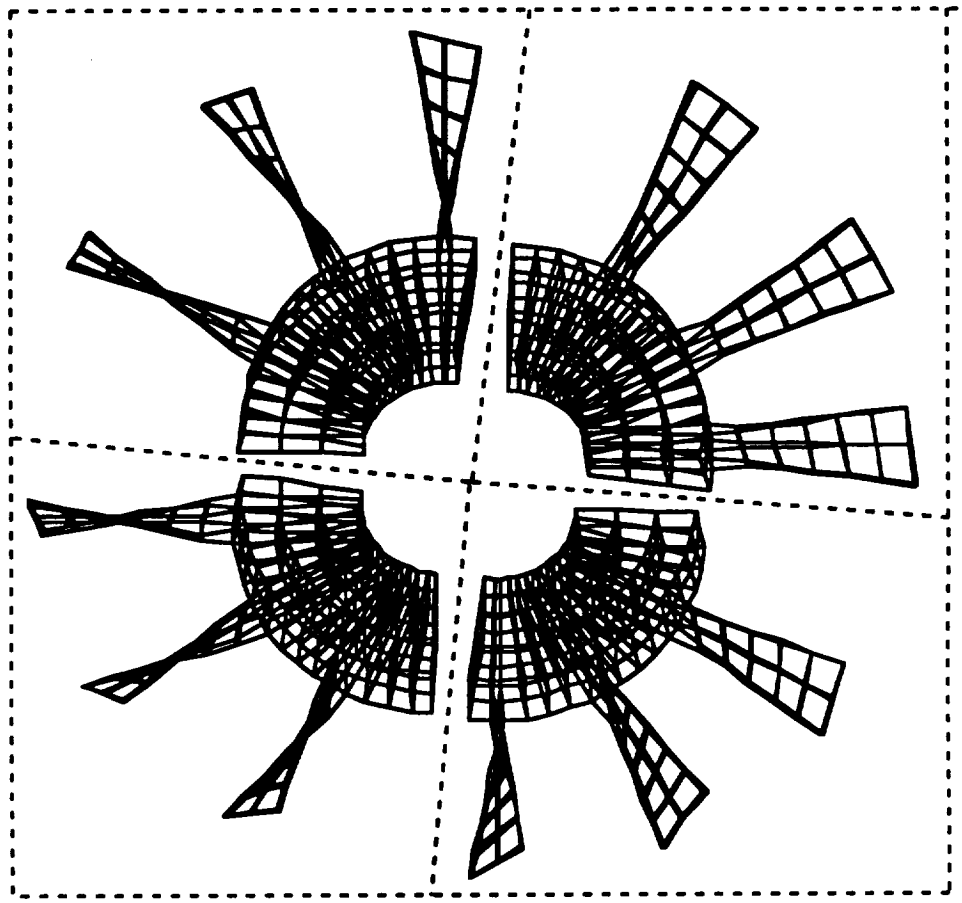
(a) Partitioning results for the FP algorithm

Figure 4.11 Partitioning results of the FP, ANP, and RST algorithms for the 12-bladed turbine disk of Fig. 3.7 ($N_p = 4$)



(b) Partitioning results for the ANP algorithm

Figure 4.11 (Continued)



(c) Partitioning results for the RST algorithm

Figure 4.11 (Continued)

Table 4.17 Performance on DEC5000's of the parallel central difference analysis of the bladed-disk problem of Fig. 3.7 with domain partitioned by different algorithms

Algorithm	$N_p = 4$		$N_p = 6$	
	Speed-up	Efficiency (%)	Speed-up	Efficiency (%)
FP	3.4	85	5.2	87
RBD	1.8	45	4.6	76
ANP	1.8	44	3.0	50
RSB (CG & DG)	3.8	94	--	--
RST (CG & DG)	3.8	94	5.4	90

Chapter 5

An Integrated Parallel Analysis System

An integrated parallel analysis system is developed in this research to help evaluate the parallel strategies investigated, verify the finite element approaches employed, and demonstrate how advanced computer technologies can assist engineers in all phases of parallel dynamics simulations. The system takes full advantage of the advanced computing environments, data structures, and interactive computer graphics and provides a research software testbed for study of nonlinear structural dynamics.

This chapter discusses the implementation and application of this system for various stages involved in parallel nonlinear simulations of structural dynamics. The emphasis is on the enhancement of several software applications previously developed at Cornell Program of Computer Graphics, the development of new applications, and most importantly, the integration of these applications into an efficient and powerful analysis system. In addition, application examples that examine and demonstrate the efficiency and flexibility of the system are presented.

5.1 System Overview

The parallel analysis system consists of four major software applications: BASYS (Srivastav and Abel 1990; Srivastav 1991) and FRANSYS (Wawrzynek et al. 1988; Martha 1989; Wawrzynek 1991) for interactive three-dimensional modelling and visualization, PSAINT (Hsieh

and Srivastav 1992) for finite element domain partitioning, and ABREAST (Srivastav 1991; Aubert 1992) for parallel nonlinear solutions. Figure 5.1 illustrates the organization of the system and indicates the relationship among these applications.

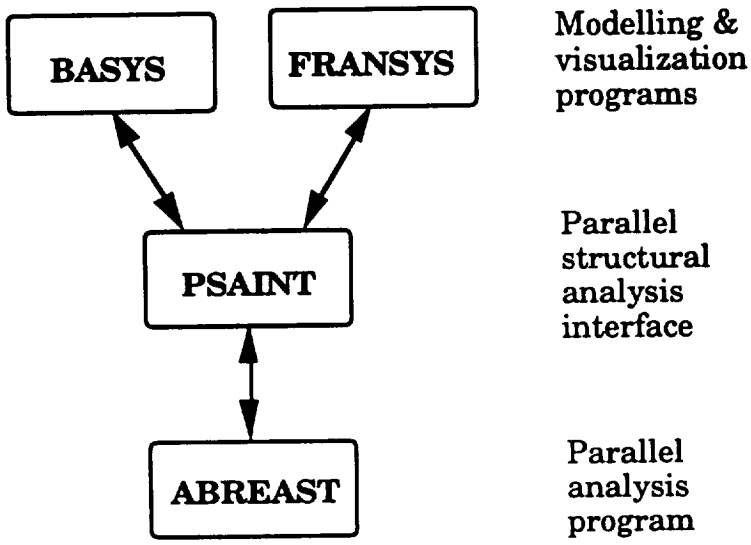


Figure 5.1 Organization of the parallel analysis system

BASYS is primarily designed for modelling and visualization of buildings, space structures, and other framed structures. FRANSYS was originally developed to model general, 3-D fracture processes in arbitrarily shaped solids and has been extended to provide general tools for modelling and simulation of complex 3-D solid models. It has been further extended in the present work to include graphics tools for visualization of dynamics simulations.

Both BASYS and FRANSYS provide the analyst an efficient way of modifying and manipulating the structural data through the use of a topological data representation called the radial edge data structure (Weiler

1986, 1988) and a hierarchical modelling scheme. The radial edge data structure supports an unambiguous boundary representation of non-manifold topologies of the structural model and therefore, allows BASYS and FRANSYS to model complex three-dimensional structures. Because the adjacency relationships of the structural entities are explicitly stored in the radial edge database, the time required for modelling operations such as determination of the elements sharing a particular node is considerably reduced. This contrasts with the use of a geometrically based database in which time-intensive searches through a connectivity table or time-consuming floating-point computations and comparisons are required for the same kinds of modelling operations. In addition, the use of a hierarchy of topological models to represent the structure allows analysts to work with the structural model at various levels of abstraction according to their needs at various stages of analysis and design. The inheritance of structural attributes (e.g., material properties, boundary conditions, etc.) from the top down in the hierarchy of models provides a natural and efficient way for the analyst to assign the structural attributes. For example, material properties only need to be assigned to entities in the geometry model at the top of the hierarchy which is the coarsest discretization of the structure, and all of the entities in the mesh model at the bottom of the hierarchy automatically inherit the material properties from their parent entities in the geometry model from which they are derived. This is opposed to "the mesh is the model" approach in which tedious assignments of structural attributes to elements and nodes of the mesh are unavoidable.

Both BASYS and FRANSYS also provide a friendly interface for user-program interaction and a convenient means of displaying the structure

model and visualizing the response of the structure using interactive computer graphics. The graphical user interfaces provided include pop-up menus, dialog boxes, message boxes, toggle buttons, data input tools using a variety of mouse- and keyboard-based techniques, and 2-D and 3-D graphical windows for displaying 2-D and 3-D objects, respectively. Two versions of graphics implementation are currently available for both programs. For convenience, they are simply referred here as X and PEX versions. The X version is based on the X Window System[1] (Scheifler and Gettys 1986) for 2-D graphical displays and a simple in-house 3-D graphics package (Wawrzynek 1987) for transformation of 3-D objects into 2-D images. To achieve a better 3-D graphical display, the PEX version uses PEX[2] (PHIGS[3] extensions to X) (Rost et al. 1989) instead of the in-house 3-D graphics package for the 3-D graphical display, but the 2-D graphics is still based on the X Window System. The current implementation of PEX in BASYS/FRANSYS uses a primitive version of PEX implemented by Digital Equipment Corporation (Potyondy 1992) and is operable only on DECstations.

PSAINT is developed in this work to serve as an interface between BASYS/FRANSYS and ABREAST for parallel analysis. It is also the key component that glues the whole system together. Two major tasks are

---

[1] X is a network-based graphics window system developed jointly by M.I.T. and Digital Equipment Corporation. It is currently supported by most of the major computer vendors.

[2] PEX is a 3-D graphics package developed to support 3-D graphical display in a network windowing environment. It has been recently gaining wider support from the major computer vendors.

[3] PHIGS is one of the available 3-D graphics standards. It is now widely used in the industry.

performed in PSAINT: (1) the partitioning of structural domains for parallel analysis in ABREAST and (2) the collection and merging of the parallel analysis results from the subdomains into a single set of results for simulation playback in BASYS/FRANSYS. All of the domain partitioning algorithms discussed in Section 4.3 have been implemented for automatic partitioning. Interactive graphics tools are also provided for manual partitioning of the structure and for evaluation and modification of the results of automatic partitioning. The program automatically generates various statistics related to partitioning results and allows for visualization of individual subdomains. Like BASYS and FRANSYS, both X and PEX versions of graphics implementation are available for PSAINT.

ABREAST is a batch analysis program capable of both geometric and material nonlinear transient dynamic analyses. Geometric nonlinear behavior is modelled using an updated Lagrangian formulation with geometric element stiffness matrices. Material nonlinear response is included for beam-column elements only through a concentrated plasticity model which is based on a bounding surface model in three-dimensional force space (Hilmy 1984; Hilmy and Abel 1985). The original version was primarily geared towards frame structures consisting of either truss or beam-column elements. A nine-node Lagrangian shell element is also available for modelling floors, walls, or panels. For modelling rotating bladed-disk systems, ABREAST has been extended in the present research to include a twenty-noded isoparametric brick solid element. Furthermore, the parallel nonlinear solution algorithms discussed in Section 3.3 for both transient dynamic and steady-state (static) analyses have been implemented using a multiple-instruction, multiple-data (MIMD) algorithm.

Efforts have been taken in the development of these programs to maximize their portability. For example, BASYS and ABREAST have been ported from Digital VAXstations running the VMS operating system to both Digital DECstations running the ULTRIX operating system and Hewlett-Packard computers running the HP-UX operating system. FRANSYS has been ported from the VMS operating system to the ULTRIX operating system. PSAINT has been ported from the ULTRIX operating system to the HP-UX operating system.

5.2 Interactive Modelling

For the sake of discussion, the modelling of the structure is divided into two phases. The first phase creates the geometry of the structure. Only the coarsest discretization of the model that accurately characterizes the structural geometry is required. The second phase involves the assignment of the structural attributes, such as material properties, boundary conditions, etc., and the generation of a mesh for finite element analysis.

Geometry Creation of Framed Structures

Until recently, BASYS relied on a program called CU-PREPF (McGuire et al. 1989) for creation of the geometry model of framed structures. CU-PREPF (Cornell University PREProcessor for Frames) is an interactive program for the definition of two- and three-dimensional framed structures. It provides a menu-driven interface to help the analyst construct structural geometry and specify structural attributes. Once the basic definition of the structure is completed in CU-PREPF, an output data file is created that can be read into BASYS for further refinements.

Although CU-PREPF is a complete preprocessor for framed structures, its effectiveness and efficiency are limited by the use of geometrically based database and "the mesh is the model" approach as discussed previously. The use of CU-PREPF for creation of the geometry model for BASYS has been an expedient in the ongoing software evolution. Recently, a model creator has been implemented in BASYS (White and Lee 1993) which eliminates the dependence of BASYS on CU-PREPF. Taking advantage of the hierarchical topology database and the graphical user interfaces in BASYS, the model creator provides a set of flexible and powerful tools for the analyst to create and edit complex 3-D framed and/or panel structures (consisting of line and/or face elements). Nevertheless, the interface in BASYS for input of models created in CU-PREPF is still available to allow for retrieval of structural models created in CU-PREPF at the early stage of this research.

Attribute Assignment and Mesh Generation of Framed Structures

The hierarchical topology database in BASYS includes three discrete levels of data representation:[4] the structural element (STR) level, the subdivision (SDV) level, and the mesh (MSH) level. The geometry model of the structure is first built up in the STR level either using the tools provided by the model creator in BASYS or based on the model created in CU-PREPF. Structural attributes are then assigned to structural elements in the STR level. If the attributes have already been specified in CU-PREPF along with the geometry model, they are automatically set to the attribute

[4] There was a forth representation known as the substructure (SBS) level which does not use topological representation and has been recently removed from BASYS.

database when the input data file from CU-PREPF is read. A set of graphics tools is also provided for interactive definition and modification of structural attributes.

To generate a finite element mesh model for the analysis, the topological edges of the STR model are first subdivided in the SDV level to define the desired mesh density throughout the structure. Then, the SDV model is further discretized in the MSH level to obtain the final finite element mesh (Srivastav 1991). The attributes associated with the elements in the MSH model are automatically inherited from those associated with their parent elements in the STR model.

Geometry Creation of 3-D Solids

FRANSYS requires as input a collection of surface patches that represents the boundary (or skin) of the 3-D solid being modelled. An object-oriented, boundary-representation, non-manifold solid modeller called OSM (Potyondy 1991) is used to help the analyst create such surface patches. Through a scripting language, the user can construct surface patches in OSM from the simpler entities such as points, lines, and curves. The statements in the scripting language may be either entered one by one directly by the user or read from a file prepared in advance. Multiple 3-D graphics windows may also be activated by the user to display any already defined entities. Once the desired surface patches are constructed, an output data file is created that can be read into FRANSYS for further modelling operations.

For creation of the bladed-disk models with different geometry (see, for example, Figs. 3.7 and 4.5) in the present research, a simple C program

called BDM is written to generate automatically the statements in the scripting language that can be read into OSM to construct the surface patches of the desired bladed-disk model. BDM allows the user to specify the following ten parameters for a bladed-disk model (see also Fig. 5.2): the total number of blades (N_b), inner radius of the disk (R_i), outer radius of the disk (R_o), thickness of the disk (H_d), length of the blades (L_b), thickness of the blades (H_b), width of the blades (W_b), length of the blade-disk joint (L_j), thickness of the blade-disk joint at the disk end (H_j), and pre-twisted angle of the blades (T_a).

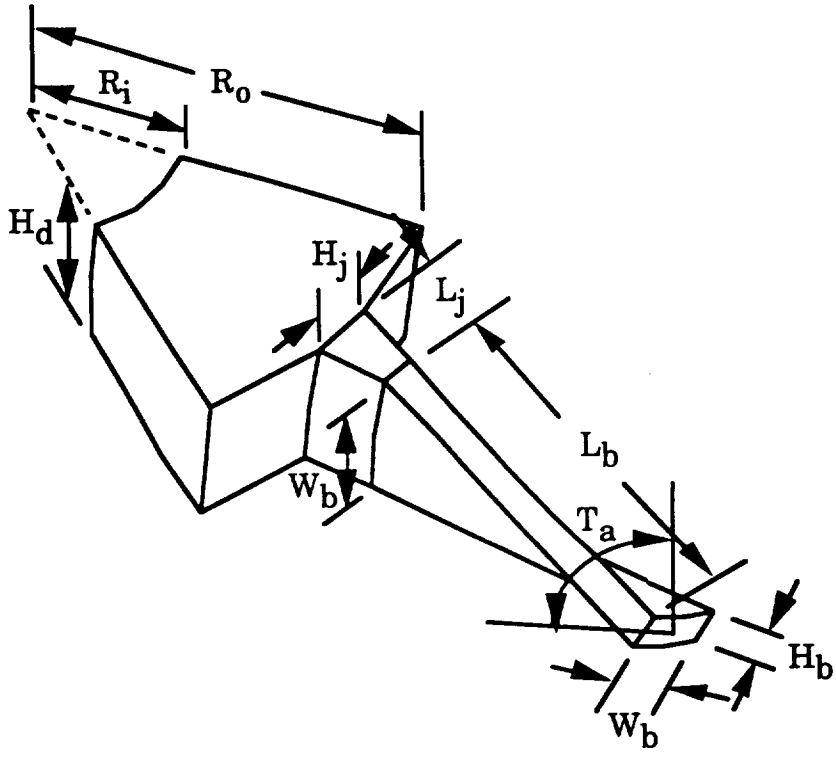


Figure 5.2 Parameters for definition of a bladed-disk model

Attribute Assignment and Mesh Generation of 3-D Solids

FRANSYS uses a hierarchy of topological models to represent the structure at different levels of discretization. The hierarchy consists of the

following five models: the geometry (GEO) model, the volume decomposition or subdomain (SDM) model, the face decomposition or subregion (SRG) model, the edge decomposition or subdivision (SDV) model, and the mesh (MSH) model. After the initial GEO model of the structure is constructed from the surface patches created in OSM, FRANSYS allows the analyst to modify the GEO model, and assign attributes to the GEO model. A set of interactive tools are also provided for refinements of the GEO model into the SDM model and further into the SRG model.

To generate a finite element mesh model for the analysis, the topological edges of the SRG model are first subdivided in the SDV level to define the desired mesh density throughout the structure. Then, the topological faces and volumes in the SDV model are meshed in the MSH level to obtain the final finite element mesh. Several meshing tools based on a variety of techniques are provided for meshing of faces and volumes. For meshing faces, these include bi-linear transfinite mapping, collapsed bi-linear transfinite mapping, tri-linear transfinite mapping, and general triangulation. For meshing volumes, the tools available are tri-linear transfinite mapping and general sweeping. In addition, the attributes associated with the elements in the MSH model are automatically inherited from those associated with their parent elements in the GEO model.

5.3 Domain Partitioning

Once the modelling in BASYS or FRANSYS is completed, an output data file, called the structural analysis (SA) data file, is created which is already a valid input file for ABREAST if only a serial finite element analysis is required. For parallel analysis, the SA file is instead read into

PSAINT for partitioning of the finite element domain defined in BASYS or FRANSYS into a number of subdomains. Both automatic and manual partitioning tools are provided in PSAINT for such partitioning. The partitioning results are appended at the end of the SA file. The updated SA file can then be read into ABREAST for parallel analysis.

Automatic Domain Partitioning

All the automatic partitioning algorithms discussed in Section 4.3 have been implemented in PSAINT for both element-based and node-based partitioning. The studies in Chapter 4 show that, in most cases, the RST(CG) algorithm gives the best partitioning results among all the studied algorithms. However, in many cases, comparisons among results from different algorithms may be necessary to assure that the best possible results among all implemented algorithms are obtained.

Manual Domain Partitioning

A set of graphics tools is provided in PSAINT under the "Interactive Partitioning" menu for manual partitioning. To define a subdomain, the user first specifies the corresponding subdomain number of the subdomain through a dialog box. The subdomain is then defined by collecting nodes in the structural domain using a set of data collectors. If the node-based partitioning is required, the collected nodes are assigned to the subdomain. If the element-based partitioning is required, those elements with all of their nodes collected are assigned to the subdomain. Upon exit from the partitioning menu when all subdomains are defined, the elements and nodes in subdomains are then labelled following the procedures discussed in Sections 4.4.1 and 4.5.1 for the element-based partitioning and the node-

based partitioning, respectively. Various collectors are available for the user to collect nodes in the structural domain:

(1) MANY NODES: nodes are collected one by one simply by pointing to them.

(2) NODES ON LINE: Two nodes are selected to define a line. All nodes lying on the line are then collected automatically.

(3) NODES ON PLANE: Three nodes are selected to define a plane. All nodes lying on the plane are then collected automatically.

(4) NODES IN CUBE: A rectangular parallelepiped with faces aligned to the global coordinate axes is defined by selecting the endpoints of its diagonal. All nodes in the parallelepiped are then collected automatically. The two endpoints can be either existing nodes in the structure or points at the coordinates specified by the user.

(5) NODES IN SECTOR: A sector is defined by specifying the following parameters as shown in Fig. 5.3: a reference point P which defines the local origin of the sector, a reference vector $\underline{v}$ which is aligned to the face, a rotation axis $\underline{u}$, the angle of rotation, ϕ, and the thickness of the sector, h. All nodes in the sector are then collected automatically. The current implementation requires that vectors $\underline{u}$ and $\underline{v}$ align with the global coordinate axes.

In addition, the above tools can be used to modify previous partitions, say, resulting from automatic partitioning. In this case, the modification is achieved by re-defining the subdomain numbers associated with the nodes (if node-based partitioning) or the elements (if element-based partitioning)

in the structural domain. First, the user first specifies the current active subdomain. Then, the collected nodes (if node-based partitioning) or the elements with all of their nodes collected (if element-based partitioning) are assigned to the active subdomain. Finally, when the re-definition of nodes or elements is completed, the elements and nodes in each subdomain are re-labelled accordingly.

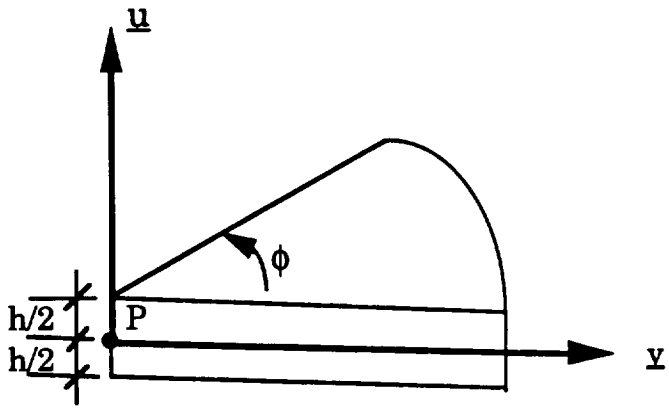


Figure 5.3 Parameters for definition of a sector

Examination of Partitioning Results

PSAINT also provides graphics tools under the "Display Substructures" menu for the user to examine partitioning results. Upon activation of this menu, subdomains in a partitioned domain are displayed in different colors. The user is then allowed to turn on and off display of any subdomain, to display subdomains individually, or to display them in a sequential order for better examination of partitioning results. Message boxes containing statistics of partitioning results are also displayed. For element-based partitioning, the statistics provided for each subdomain include the total number of nodes, the number of interior nodes, the number of primary boundary nodes, the number of secondary boundary nodes, and

the number of elements of different types. For node-based partitioning, the statistics provided for each subdomain include the total number of nodes, the number of interior nodes, the number of boundary nodes, the number of adjacent nodes, and the number of elements of different types.

5.4 Parallel Nonlinear Analysis

After reading the SA file prepared by BASYS/FRANSYS and PSAINT, ABREAST performs the desired parallel nonlinear analysis specified in the SA file. As already discussed in Section 3.3, the parallel solution algorithms implemented in ABREAST for this work include a parallel central difference algorithm for explicit transient analysis, a parallel Newmark algorithm with domain decomposition for implicit transient analysis, and a parallel Newton-Raphson iterative-incremental algorithm for steady-state (static) analysis. The effectiveness of these parallel algorithms implemented has also been investigated in Section 3.4.

A UNIX shell script is used to help the user run parallel analyses. The shell script takes two arguments as input: the name of the SA file (without file extension) and the name of a user-specified file containing a list of processors available for use in parallel analyses. The script extracts the number of subdomains (or processors), N_p, from the SA file and remotely invokes executable image of ABREAST on each of the first N_p processors in the list to start the parallel analysis.

All copies of ABREAST running on separate processors are identical and read identical copies of the SA file. The order that each ABREAST joins the ISIS process group is used to automatically determine the distribution of subdomains among processors (i.e., subdomain No. 1 is assigned to the

first member in the process group, and so on). Because each processor executes identical code, but asynchronously and on different data, this is a MIMD (multiple instruction, multiple-data) approach. As discussed in Section 3.1, the communication and synchronization between processors are achieved in a message passing environment provided by ISIS.

5.5 Response Visualization

Once the parallel analysis is complete, an individual response file is created by each ABREAST on a separate processor which contains analysis results associated with the subdomain assigned to that particular processor. PSAINT is then used to gather these response files into a single file for visualization in BASYS or FRANSYS.

There are three major utilities provided in BASYS and FRANSYS for visualization of dynamic responses of the structure. The first one is for animation of modal vibration of the structure. The eigenvectors (or mode shapes) obtained from an ABREAST eigensolution can be scaled and dynamically displayed. The second utility provides two-dimensional plots for selected response data at specific points in the finite element model. The user can interactively specify the time interval for which the desired response quantities are to be plotted. The user can also choose from the following quantities for the ordinate and abscissa axes in the plots: time, displacements, velocities, accelerations, stresses, strains, and stress resultants. The third utility is for dynamic simulation of the structure. The results of displacements and stresses/strains in the response file (which are calculated at the user-specified time intervals) are used to construct a sequence of displays that animates the deforming motions of the structure

with color contours showing the changing of a selected stress or strain quantity. The user is allowed to specify the time interval and the displacement amplification factor for the dynamic simulation.

5.6 Interactive Monitoring and Steering

The capabilities for interactively monitoring and steering parallel analysis have not been implemented in the current version of parallel analysis system. However, the idea for monitoring the progress of parallel analysis has been investigated in the preliminary version of parallel analysis system consisting of BASYS and ABREAST at the early stage of this research (Abel et al. 1991; Aubert 1992). At that time, PSAINT had not been created and temporary graphics tools were provided in BASYS for interactive domain partitioning. Visualization tools in BASYS were extended to include a response monitor and process monitors for use in monitoring the progress of parallel analysis of framed structures. The response monitor provides a means to display analysis results (i.e., deformed shape of the structure with color contours of the selected stress/strain quantity) as they are being calculated, while the process monitors display the most current status of parallel processors, such as the name of the computing node, the time spent for the actual computations, the time spent for interprocess communication, etc. The user is also allowed to specify the update frequencies (in terms of time steps) of the monitors. Up to six DECsystem 5000's were used.

Figure 5.4 shows the interactive monitoring of parallel analysis using the preliminary version of parallel analysis system in a networked workstation environment. ABREAST's are run on a number of workstations

and communicate through ISIS. Also, through ISIS, BASYS collects the most recent analysis results as they are being calculated by the ABREAST's and monitors the progress of the parallel analysis. In addition, the graphics display of BASYS can be visualized on any workstation specified by the user through the X Window system.

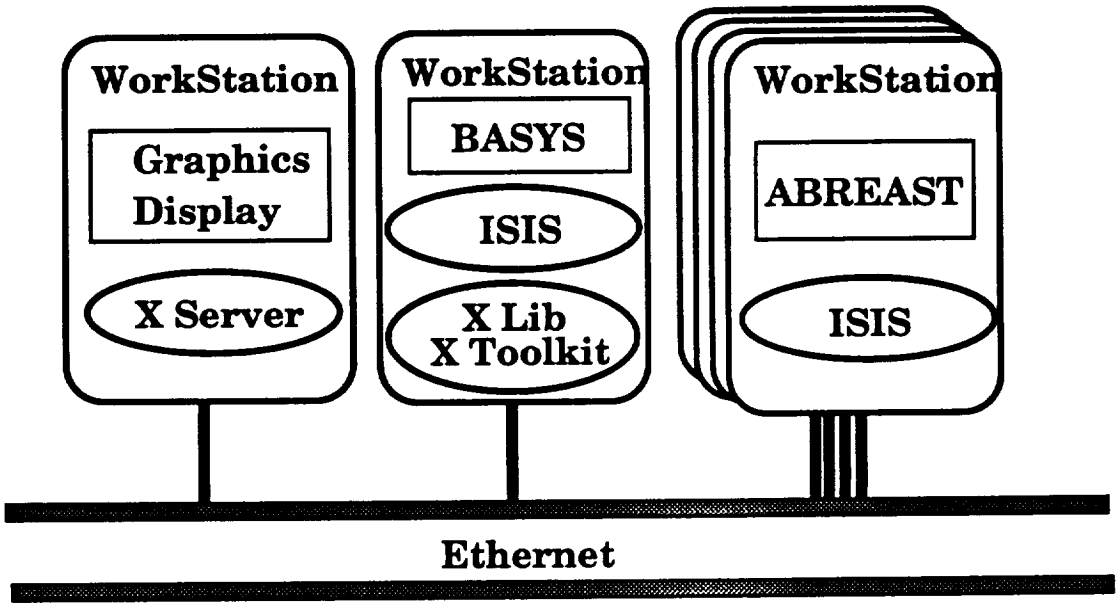


Figure 5.4 Interactive monitoring of parallel analysis using the preliminary version of the parallel analysis system

It was found that the performance of the parallel analysis was degraded when the interactive monitors (especially the response monitor) in BASYS were used. This is mainly due to the increase of communication overhead that is required for the various ABREAST's to send calculated responses of the structure and timing statistics of the parallel analysis to BASYS. The amount of response information that needs to be sent to the response monitor becomes larger as the size of the structure being analyzed

increases. It was also found that the time required to update the display of the response monitor might increase considerably so that a responsive interactive display environment in BASYS could not be maintained.

However, as the processing power and the speed of communication network continue to increase, interactive monitoring and steering of parallel analyses in the current parallel analysis system will become more feasible. The future development of interactive monitoring and steering capabilities in the current parallel analysis is suggested in Section 6.2.

5.7 Application Examples

Two examples are used in the present work to examine and demonstrate the efficiency and flexibility of the parallel analysis system presented above. The first example analyzes a framed structure subjected to seismic loading, while the second one analyzes a rotating turbine bladed-disk experiencing tip rubs.

5.7.1 Framed Structure Subjected to Seismic Loading

A twelve-story L-shaped building is analyzed to demonstrate the capabilities of the parallel analysis system for dynamic simulations of framed structures. The structure has been studied by Srivastav (1991) to examine effects of floor flexibility on building responses and is similar to the one shown in Fig. 3.9 but with each floor panel modelled by four shell elements. Only linear elastic analyses are performed by Srivastav.

Detailed descriptions of the design of the structure and assumptions used in the analysis have been given by Srivastav (1991) and, therefore, are not repeated here. Four different strategies were used for modelling floor

slabs of different flexibility in Srivastav's work. In the present work, the floor slabs are modelled by shell finite elements with the stiffness of a typical reinforced concrete slab.

The geometry model and attributes of the structure are defined in CU-PREPF, while the finite element meshes of the structure are generated in BASYS. The resulting finite element model consists of 696 beam-column elements, 288 nine-node Lagrangian shell elements, 1,514 nodes, and 7,668 unrestrained degrees of freedom.

The structure is subjected to the El Centro earthquake. Both linear and geometric nonlinear parallel analyses are performed using six HP9000/720's. The time step is 0.01 seconds and 20 seconds of structural responses are analyzed, i.e., 2,000 steps are performed. The structure is partitioned by floors into six subdomains using the interactive tools provided in PSAINT. Each subdomain consists of two floors of the structure. The parallel Newmark implicit time integration in ABREAST is used for dynamic solutions. For each time step, the equilibrium solution is achieved using modified-Newton iterations. The diagonal scaling preconditioner of Eq. 3.9 is used in the parallel preconditioned conjugate gradient (PCG) iterations. The convergence tolerance for the modified-Newton iterations is 10^{-5}, while the convergence tolerance for the conjugate gradient iterations is $1x10^{-8}$. The results are output every 0.05 seconds.

The measured wall clock time required for the parallel analyses using six HP9000/720's is given in Table 5.1. If only a single HP9000/720 is used, it is estimated that the linear analysis would take about 13 hours and 14 minutes, while the geometric nonlinear analysis would take about 53 days

13 hours and 32 minutes. These estimations are extrapolated from time required for actual analysis runs of twenty time steps. In this example, superlinear speed-up is observed in both the linear and nonlinear parallel analyses. As discussed in Section 3.4.2, this is probably due to the fact that the profile (or skyline) of the coefficient matrix for the whole structure in the serial analysis is not as well minimized as that of the coefficient matrix for each subdomain in the parallel analysis. Moreover, for analysis of this particular structure, the substructuring approach with the PCG solver in the parallel analysis may require fewer computational operations (i.e., may be more effective) than the direct approach in the serial analysis because only a very small number of PCG iterations is required in the analysis (see Table 5.1). This also accounts for the superlinear speed-up of the analyses.

Table 5.1 The time required for parallel implicit analyses of the twelve-story L-shaped building studied in Section 5.7.1 (six HP9000/720's are used)

Analysis type	Linear	Geometric nonlinear
Wall clock time (days hours:minutes)	2:09	3 15:40
Average number of equilibrium iterations per time step	--	6
Average # of PCG iterations	7	12
Number of boundary d.o.f.'s	420	420

Finally, the response files of the parallel analysis are gathered by PSAINT into a single response file for response visualization in BASYS. In this example, the gathered response file takes up approximately 86 Mbytes of disk space. Fig. 5.5 shows a simulation playback of dynamic responses of the structure in BASYS.

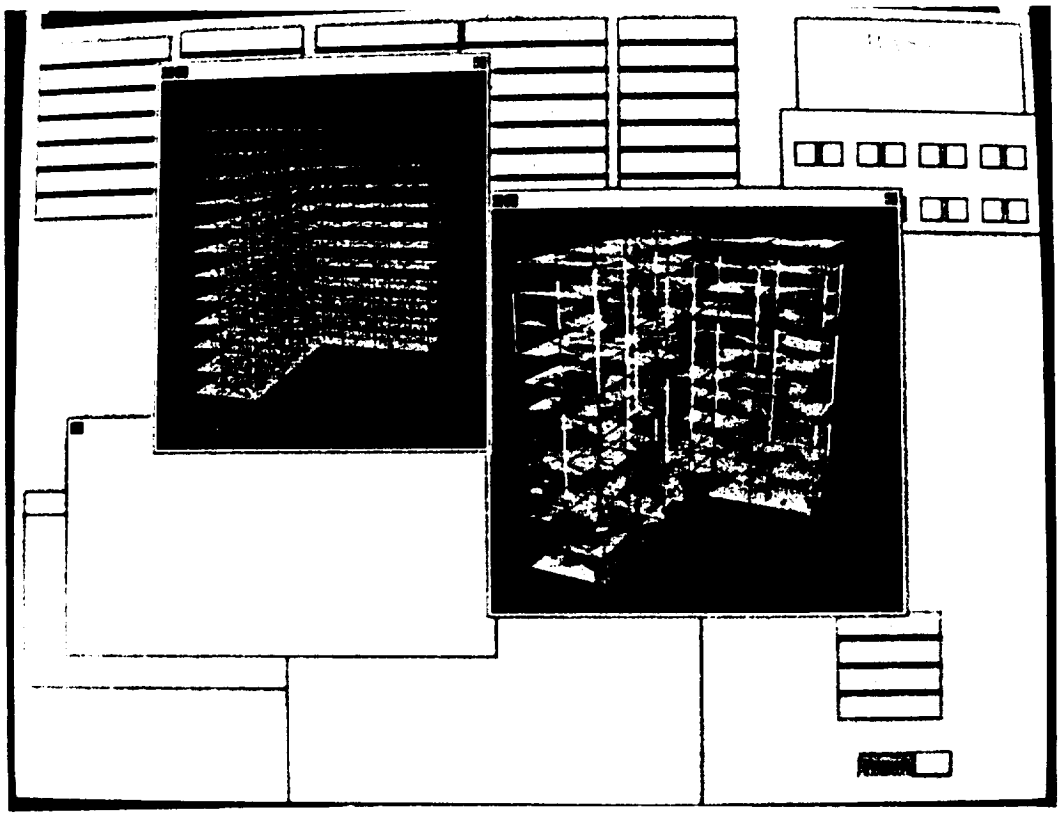


Figure 5.5 A simulation playback of dynamic responses of the 12-story L-shaped building in BASYS

5.7.2 Rotating Bladed-disk Experiencing Tip Rubs

The rotating turbine 12-bladed disk problem with the finite element model of Fig. 3.7 is analyzed to demonstrate the capabilities of the parallel analysis system for transient simulations of rotational dynamics of solid models. The bladed-disk rotates at the speed of 12,000 rpm and experiences a rubbing impact at the tip of one of its blades.

Geometrical Configuration and Material Properties

The geometry model of the structure is constructed using the BDM and OSM programs discussed in Section 5.2. The geometric configuration of the model is described by the following parameters (see also Fig. 5.2):

1) the inner radius of the disk (R_i) = 0.048 m,
2) the outer radius of the disk (R_o) = 0.12 m,
3) the thickness of the disk (H_d) = 0.05 m,
4) the length of the blades (L_b) = 0.10 m,
5) the thickness of the blades (H_b) = 0.0025 m,
6) the width of the blades (W_b) = 0.05 m,
7) the length of the blade-disk joint (L_j) = 0.02 m,
8) the thickness of the blade-disk joint (H_j) = 0.0075 m, and
9) the pre-twisted angle of the blades (T_a) = 45^0.

The properties of the aluminum are used for the entire structure and are represented by the following parameters:

1) the Young's modulus = 6.9×10^{10} N/m^2,
2) the density = 2687.36 kg/m^3, and
3) the poison ratio = 0.33.

Finite Element Mesh

The finite element mesh model of the structure is generated in FRANSYS. In the present example, after the topological edges of the model are subdivided, the automatic meshing capabilities provided by FRANSYS are used to construct both surface and volume meshes. The resulting finite element model consists of 504 twenty-node brick solid elements, 3,828 nodes, and 10,044 unrestrained degrees of freedom. The twenty-node brick element uses the reduced integration scheme in all analyses performed in the present example.

Domain Partitioning

The structure is partitioned into six subdomains using the RST automatic partitioning algorithm in PSAINT. For partitions used in parallel steady-state analysis, the partitioning type specified is the element-based partitioning, while the node-based partitioning is specified for partitions used in parallel central difference analysis.

Parallel Steady-State Analysis

To obtain the steady-state solution of the bladed-disk model rotating at the speed of 12,000 rpm, parallel steady-state analyses are performed using six DEC5000's. Both rotational and geometric nonlinearities are considered in the analyses and a lumped mass matrix is employed. Two sets of analyses using the lumped mass (LM) and consistent mass (CM) approaches for the formulation of the rotational terms, respectively, are performed. The fractional load step size used is 0.2, and Newton-Raphson iterations are used to achieve equilibrium solutions for each load increment.

The measured wall clock time required for the parallel analyses is given in Table 5.2. The maximum radial displacement at the blade tips is about 0.5 mm.

Serial analyses using a single DEC5000 are also performed. The analysis with the LM approach takes about 8 hours and 31 minutes, while the analysis with the CM approach takes about 7 hours and 26 minutes. Based on these results, a speed-up of 3.8 and an efficiency of about 64% can be calculated for the parallel analyses with both the LM and CM approaches.

From Table 5.2, it can be seen that the analysis with the CM approach takes less time than that with the LM approach. This is mainly because the former requires fewer Newton and PCG iterations in this example despite the greater computational effort per Newton step to account for rotational nonlinearity.

Table 5.2 The time required for parallel steady-state analyses of the rotating turbine bladed-disk studied in Section 5.7.2 (six DEC5000's are used)

Approach	LM	CM
Wall clock time (hours:minutes)	2:16	1:56
Total number of Newton iterations	39	33
Average # of PCG iter. per Newton iter.	78	75
Number of boundary d.o.f.'s	432	432

Modal Vibration Analysis

The modal vibration characteristics of the bladed-disk model are examined. Two cases are considered in the analysis. In the first case, the model does not rotate (Ω = 0 rpm), while the model rotates at the speed of Ω = 12,000 rpm in the second case. For the case of Ω = 12,000, the results of the previous steady-state analysis are used as initial conditions for the analysis. The subspace iteration method in ABREAST is used for eigensolutions. A lumped mass matrix is used. The computed frequencies for the first thirty-six modes are given in Table 5.3. The results are then read into FRANSYS for visualization of mode shapes. Each of the eigenanalyses takes about 2 hours and 57 minutes on a single DEC5000.

Table 5.3 Modal frequencies of the turbine 12-bladed disk studied in Section 5.7.2

Mode	Frequency (Hz)		
	Ω = 0 rpm	Ω = 12,000 rpm	
		LM approach	CM approach
1	181	278	278
2 ~ 12		279	279
13 ~ 17	838	413	412
18~ 24	839		
25 ~ 28	1147	625	620
29		626	621
30 & 31		629	624
32 & 33		632	627
34 ~ 36		633	628

From Table 5.3, it can be observed that, for both nonrotating and rotating cases, the frequency results fall into three clusters. The first cluster includes the first twelve modes, the second one includes the next twelve modes, and so on. In the first cluster, the frequencies of the rotating case are higher than those of the nonrotating case. However, the frequencies of the rotating case in the second and third clusters are lower than those of the nonrotating case in the second cluster. It should be noted also that the mode shapes of the same mode number in both the nonrotating and rotating cases may not necessarily be the same.

Parallel Transient Dynamic Analysis

The transient responses of the rotating bladed-disk model are analyzed. The model rotates at the speed of 12,000 rpm and experiences a rubbing impact at the tip of one of its blades. Parallel central difference analysis using the LM approach is performed using six DEC5000's. Both rotational and geometric nonlinearities are considered in the analyses and a lumped mass matrix is employed. The results of the previous steady-state analysis are used as initial conditions for the analysis.

The blade tip rubs are modelled by a set of applied impact forces based on a dry friction rub model. In this rub model, the frictional (or circumferential) force, F_t, is proportional to the normal (or radial) force, F_n. That is

$$F_t = \mu F_n \tag{5.1}$$

in which μ is the coefficient of friction. In this example, $\mu = 0.2$ is used and uniform tractions T_t and T_n are applied to the blade tip surface for the impact forces F_t and F_n, respectively. Figure 5.6 shows the time history of

the normal traction T_n which acts in the inward radial direction. The circumferential traction T_n acts in the direction opposite to the tangential blade tip velocity.

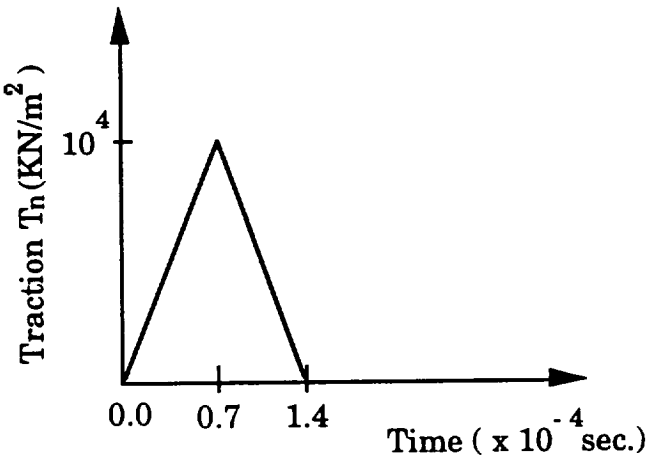


Figure 5.6 Time history of the normal traction T_n used in the transient dynamic analysis of Section 5.7.2

The transient responses are computed for a duration of 10^{-3} sec. and a time step of 10^{-7} sec. is used. The results are output every $5x10^{-6}$ sec., i.e., every 50 time steps. The measured wall clock time required by the parallel analysis with the LM approach is about 12 hours and 19 minutes. Serial analysis with the LM approach using a single DEC5000 is also performed. The analysis takes about 2 days 15 hours and 12 minutes. Based on the above results, a speed-up of 5.1 and an efficiency of 86% can be calculated. For analyses with the CM approach, it is estimated that the parallel analysis using six DEC5000's would take about 4 days 6 hours and 53 minutes, while the serial analysis using a single DEC5000 would take about

23 days 1 hour and 47 minutes. These estimations are extrapolated from the analysis results obtained in Section 3.4.1 for the same turbine 12-bladed disk model but for only 100 time steps.

Finally, the response files of the parallel analysis are gathered by PSAINT into a single response file for response visualization in FRANSYS. In this example, the gathered response file takes up approximately 58 Mbytes of disk space. Fig. 5.7 shows a simulation playback of dynamic responses of the structure in FRANSYS.

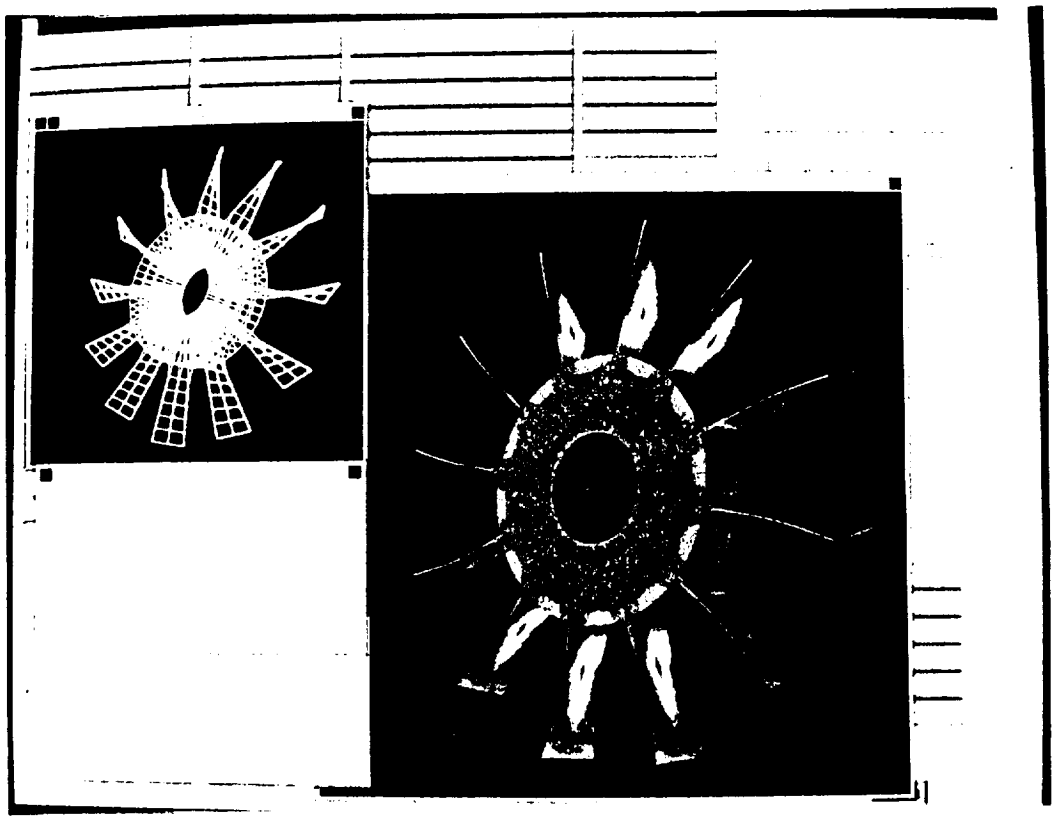


Figure 5.7 A simulation playback of dynamic responses of the turbine 12-bladed disk in FRANSYS

Chapter 6

Summary and Conclusions

The principal objective of this research is to develop, test, and implement coarse-grained, parallel-processing strategies for nonlinear dynamics simulations of practical structural problems. The parallel-processing strategies considered include numerical algorithms for parallel nonlinear solutions and techniques to effect load-balancing among processors. Finite element techniques are employed for modelling and analysis of structural dynamics problems studied in this work. Two classes of problems are investigated: framed structures with flexible floors subjected to seismic loading and rotating turbine bladed-disk assemblies experiencing tip rubs. However, emphasis is placed on the structural dynamics of rotating turbine bladed-disk assemblies. To facilitate more efficient and powerful simulations of nonlinear structural dynamics, an integrated parallel finite element analysis system is presented. This chapter summarizes the research work reported in this dissertation, draws conclusions, and suggests directions for future work.

6.1 Summary

There are four main components included in the present work. The first of them deals with finite element approaches for modelling and analyzing structural dynamics problems of rotating turbine bladed-disk assemblies as well as framed structures. Both the second and the third address parallel-processing strategies for finite element analysis of

structural dynamics. The second investigates numerical algorithms for parallel nonlinear solutions, while the third studies techniques to effect load-balancing among processors. Finally, the last one entails the development of an integrated parallel analysis system.

Finite Element Modelling and Analysis

Two classes of structural dynamics problems are studied for the development, testing, and evaluation of parallel-processing strategies addressed in this work. The first one includes framed structures with flexible floors subjected to seismic loading, while the second one includes rotating turbine bladed-disk assemblies experiencing tip rubs. The finite element approach is employed to model complex structural geometries and material properties, to account for both geometric and rotational nonlinearities, and to formulate the governing equations of motion for these problems. The emphasis is on the finite element formulation for handling geometric nonlinear analysis of rotating multi-bladed disk problems.

For modelling and analysis of framed structures with flexible floors, the finite elements and analysis capabilities already provided in ABREAST are employed. The beam-column element is used to model beams and columns of framed structures, while the nine-noded Lagrangian shell element is used to model floor panels. The dynamic analysis capabilities of ABREAST include eigensolvers for undamped vibration analysis and both explicit and implicit direct time integration solvers for transient analysis. Geometric nonlinearity is considered in the analysis.

For modelling the turbine bladed-disk, the twenty-noded isoparametric brick element with reduced integration is used after

consideration of several alternative modelling strategies as discussed in Section 2.2. The present implementation of this element in the finite element library of ABREAST has been verified using three example problems in Section 2.2.3.

Because turbine blades are relatively flexible and normally respond to the centrifugal forces with considerable deflections, a geometrically nonlinear analysis is usually required for accurately predicting dynamic behavior of rotating blades. Furthermore, the centrifugal loads are also dependent upon displacements in that they are proportional to the instantaneous radius from the rotational axis. The present research accounts for geometric nonlinearities through the use of the geometric stiffness and an updated Lagrangian formulation. Two finite element approaches presented in Section 2.3.3 are implemented in ABREAST to incorporate rotational nonlinearities into equations of motion of the rotating system: the consistent mass (CM) and lumped mass (LM) approaches. The CM approach treats the structure as a continuum with mass points uniformly distributed in the structure, while the LM approach treats the structure as a collection of discrete concentrated mass points. In Section 2.5.3, numerical studies have been conducted in to verify the present implementation of both the CM and LM approaches for rotational dynamics. In addition to verification studies, numerical comparisons are performed between the CM and LM approaches for accounting for rotational nonlinearities.

A two-stage analysis is employed to carry out modal vibration analyses of rotating bladed-disk systems. The first stage involves a nonlinear static analysis to obtain a steady-state solution which serves as

the initial condition for the second-stage eigenvalue analysis. For transient dynamic analysis of rotating bladed-disk assemblies experiencing tip rubs, a similar two-stage analysis is also employed to save computational time. The first stage analysis is the same as that in the modal vibration analysis. The solution of the first-stage analysis then serves as the initial condition for the second-stage transient analysis which computes structural responses during tip rubbing. A static analysis capability, which did not previously exist in ABREAST, has been implemented for the steady-state solution.

Parallel Nonlinear Solution Algorithms

The parallel computing environment used in this work consists of either up to six DECsystem 5000's or up to twelve Apollo/HP 9000 series 720's. These UNIX workstations are connected by Ethernet and communicate via the TCP/IP. The interprocess communication and synchronization are achieved in a message passing environment provided by ISIS. This is a coarse-grained, distributed-memory environment where the number of processors used is small and no global memory is shared among processors. The present research focuses on parallel solution algorithms suitable for this type of environment where minimization of communication overhead and parallelization at the substructure level (as opposed to the parallelization at the element level or at the degree-of-freedom level as in a fine-grained environment) are two key strategies for the algorithms to achieve good parallel performance.

For parallel explicit transient analysis, the parallel central difference method developed by Hajjar and Abel (1989a) is adopted in this research and implemented in ABREAST. The central difference algorithm is

inherently amenable to parallel processing. With the use of lumped masses to yield a diagonal mass matrix, the solution may proceed on a degree-of-freedom basis without treatment of simultaneous equations and with only a minimum amount of communication required between adjacent processors.

For parallel implicit transient analysis, the parallel Newmark algorithm with domain decomposition presented by Hajjar and Abel (1988) is adopted with slight modifications and implemented in ABREAST. This parallel implicit algorithm uses a domain decomposition approach within the Newmark time stepping outer loop. The domain decomposition approach starts with partitioning the structure into a number of subdomains and assign each subdomain to a separate processor. Then, the substructure condensation is carried out on each processor independently and concurrently without any interprocess communication. Finally, a condensed set of system equations associated with unknowns along subdomain interfaces is solved by a parallel diagonally-preconditioned conjugate gradient algorithm.

For parallel steady-state analysis, a parallel Newton-Raphson iterative incremental algorithm is implemented in ABREAST. The algorithm uses the same approach as the parallel implicit algorithm except that the time stepping outer loop is replaced by the load increment outer loop.

To evaluate the effectiveness of the parallel algorithms implemented in this work, numerical studies have been performed in Section 3.4 using structures of different types.

Load Balancing Among Processors

To effect load-balancing among processors, the present research focuses on the automatic domain partitioning techniques for parallel finite element analysis of structural dynamics. The domain partitioning techniques partition the domain of a structure into a number of subdomains which are distributed among the processors and the computation involved in a subdomain is carried out by a separate processor. Well-balanced distribution of computations among subdomains and minimization of interprocess communication are sought so that significant speed-up can be obtained in the parallel analysis. Because the partitioning is done only once and before the actual computation starts (i.e., in the preprocessing phase), the domain partitioning methods are classified as static (as opposed to dynamic) load balancing techniques.

The automatic partitioning algorithms proposed by Farhat (1988), Malone (1988), Al-Nasra and Nguyen (1991), and Simon (1991) are investigated in this research. Two generalized versions of Simon's recursive spectral bisection (RSB) partitioning algorithm (1991) are then proposed in Section 4.3.5. They are called the recursive spectral sequential-cut (RSS) and the recursive spectral two-way (RST) partitioning algorithms. Unlike the RSB algorithm in which the number of partitions is restricted to an integer power of two, both the RSS and RST algorithms can yield an arbitrary number of partitions. In addition, interactive graphics tools are developed to allow for manual partitioning and for examining and modifying results of automatic partitioning.

Comparative studies have been conducted in Sections 4.4 and 4.5 to evaluate and compare both efficiency and effectiveness of the automatic partitioning algorithms investigated. Different types of structures modelled by 1D, 2D, and 3D finite elements are employed in the studies. The use of different graph representation of the finite element meshes in the spectral partitioning algorithms is also investigated.

An Integrated Parallel Analysis System

An integrated parallel analysis system has been developed in this work to help evaluate the parallel strategies investigated, verify the finite element approaches employed, and demonstrate how advanced computer technologies can assist engineers in parallel dynamics simulations. Using the advanced computing environments, data structures, and interactive computer graphics, this system provides an efficient and powerful environment for simulations of nonlinear structural dynamics.

The system integrates four in-house, general-purpose computer programs: BASYS and FRANSYS for three-dimensional modelling and visualization, PSAINT for finite element domain partitioning, and ABREAST for nonlinear dynamic solutions. In Chapter 5, the efficiency and flexibility of the system for parallel dynamics simulations have been demonstrated.

6.2 Conclusions

The main contributions of this work are: (a) implementation and comparison of two finite element approaches for handling rotational

nonlinearities in dynamic analysis of rotating multi-bladed disks, (b) investigation and implementation of three parallel solution algorithms for analysis of structural dynamics in a modern networked workstation environment, (c) generalization of the Recursive Spectral Bisection (RSB) partitioning algorithm (Simon 1991) for an arbitrary number of partitions, (d) evaluation and comparison of several automatic domain partitioning techniques for load balancing among processors, and (e) integration of a parallel finite element analysis system.

Comparison of Two Finite Element Approaches

Two finite element approaches have been sucessfully implemented in this work to account for rotational nonlinearities involved in dynamic analysis of rotating bladed-disk assemblies: the Consistent Mass (CM) and Lumped Mass (LM) approaches. In both cases, the inertial properties (mass) are represented by a diagonal (lumped) mass matrix. However, the different methods cited are based on whether the mass effects in the rotational terms are either considered as lumped (LM) or distributed (CM). Based on the numerical comparative study conducted in Section 2.5.3 and results observed in the application examples of Section 5.7.2, the following conclusions may be made:

(1) Although the LM approach is not expected to be as accurate as the CM approach due to its neglect of mass coupling, it has been found that the modal vibration results obtained using the LM approach are in close agreement with those obtained using the CM approach. In most cases, the frequency results from the LM approach are slightly higher than those from the CM approach.

The transient displacements predicted using the LM approach also agree well with those predicted using the CM approach.

(2) On the other hand, the LM approach is expected to be computationally more efficient than the CM approach. This has been found to be the case for transient dynamic analysis using explicit time integration. In the examples studied, the computational time required in the analysis using the CM approach has been found to be about 9 times more than that using the LM approach. However, for modal vibration analyses, the computational times of analyses using these two different approaches do not differ significantly.

Parallel Nonlinear Solution Algorithms

It has been shown in Section 3.4.1 that the parallel central difference algorithm can achieve high parallel speed-up and efficiency. For example, in the analysis of the 12-bladed turbine disk of Fig. 3.7, the speed-up ranges from 1.9 for two processors (i.e., 96% efficiency) to 9.7 for twelve processors (i.e., 81% efficiency) in a network of HP9000/720 workstations. However, the central difference method is conditionally stable and often requires at least an order-of-magnitude more time steps than an unconditionally stable implicit method. Nevertheless, for structural dynamics problems of short-duration nature where a short time step is necessary to capture the dynamic phenomena, the central difference analysis can be a cost-effective time integration algorithm. One example is the structural dynamics of rotating turbine bladed-disk assemblies experiencing tip rubs investigated in this work. Due to the short-duration, impact-like nature of the tip rubs, the use of a small time step is necessary in analysis to capture the dynamic

phenomena. Furthermore, the size of the problem is usually large, some of the coefficient matrices are skew-symmetric rather than symmetric, and significant computation and storage are required for matrix assembly and solution of system equations if implicit methods are used.

In addition to more computation and storage needed for stiffness assembly and solution of system equations, the parallel implicit algorithm requires significantly more synchronization and communication overhead than the parallel explicit algorithm and, therefore, is less efficient. However, its unconditional stability makes it more suitable for long-duration problems such as the structural dynamics of framed structures subjected to seismic loading investigated in this work. It has been shown in Section 3.4.2 that, with the use of the parallel implicit algorithm, nonlinear analysis of structural dynamics can benefit from parallel processing, especially when the problem size is large and an appropriate number of processors is used. For example, in the analysis of the 12-story L-shaped building of Fig. 3.9, a speed-up of 3.3 (83% efficiency) is achived when four HP9000/720's are used. In addition, the diagonal scaling preconditioner has been shown to accelerate effectively the convergence rate of the parallel conjugate gradient algorithm in the examples studied.

The synchronization and communication overhead required in the parallel steady-state algorithm is smilar to that of the parallel implicit algorithm. Good parallel performance of this algorithm has also been obtained in the example studied. For example, in the analysis of the 12-bladed turbine disk of Fig. 3.7, a speed-up of 3.2 (80% efficiency) is achived when four HP9000/720's are used.

Automatic Domain Partitioning Algorithms

As presented in Section 4.3.5, two partitioning algorithms have been developed in this research, both of which generalize the RSB algorithm for an arbitrary number of partitions. They are the Recursive Spectral Sequential-cut (RSS) and the Recursive Spectral Two-way (RST) algorithms. In Sections 4.4.3 and 4.5.2, comparative studies are conducted among these algorithms and several other algorithms proposed by previous researchers. From these comparative studies, the following conclusions may be made:

(1) It has been found that in most cases the RST algorithm with the communication graph approach gives the best partitioning results among all the considered algorithms.

(2) all the considered algorithms deliver partitions in a very small fraction of the computational time for the dynamic analysis. For coarse-grained problems studied, the spectral partitioning methods require about the same computational time as the non-spectral methods.

(3) Although it is undesirable to have fragmented subdomains because they usually result in longer subdomain boundaries, it should be noted that having nonfragmented subdomains is not sufficient in itself to obtain shorter subdomain boundaries.

An Integrated Parallel Analysis System

A parallel analysis system has been integrated in this work. Its implementation and application for various stages involved in parallel nonlinear simulations of structural dynamics have been discussed in

Chapter 5. From the examples presented in Section 5.7, the system has been shown to provide a useful research software testbed for study of nonlinear structural dynamics.

6.3 Suggestions for Future Research

There are several areas in which research should continue to improve parallel processing strategies for simulations of nonlinear structural dynamics. There are also several possible enhancements of the parallel analysis system integrated in this work. Some suggestions for future work are discussed as below.

Parallel Modal Vibration Analysis

To study dynamic characteristics of a structure, modal vibration analysis is often required. For large structural problems, the eigensolution involved in the modal vibration analysis is a computationally intensive task and, therefore, is a potential candidate to benefit from parallel processing.

This work has investigated parallel time integration algorithms for transient dynamic analysis and parallel equation solvers for steady-state analysis, but has not studied parallel eigensolution algorithms for modal vibration analysis. Future research should be conducted to investigate parallel solution algorithms for large eigenproblems suitable for the present coarse-grained, distributed-memory environment. The parallelization of commonly used algorithms such as the subspace iteration method, the Lanczos method, and the determinant search method could be studied and compared.

Parallel Implicit Analysis

In the present study of the parallel implicit algorithm, the single-processor analysis does not use the substructuring approach but, instead, uses a direct Gauss elimination method for the solution of the total set of original equations. In many cases where the problem size and bandwidth are not too large, this approach can lead to more conservative measures of the parallel speed-up and efficiency. However, to obtain a better understanding of synchronization and communication overhead inherent in the algorithm, research should be done to use the same substructuring approach in the serial analysis as in the parallel analysis.

Load Balancing Among Processors

As discussed in Chapter 4, a generic finite element mesh may include finite elements of different types. In addition, a general networked computing environment may consist of computers with different processing power. To handle generic finite element meshes and to maximize the utilization of computer resources in a network, further development and enhencement of the present domain partitioning approaches are required.

The domain partitioning methods are static load balancing techniques which distribute computational loads among processors only once and before the actual computation starts. This approach is suitable for problems in which the distribution of computational loads among processors is constant throughout the course of the analysis. However, for dynamic analysis involving localized nonlinear responses, the presence of localized nonlinearity may create significantly unbalanced workloads among processors, resulting in reduction in parallel efficiency. Therefore, there is a

need to study and develop dynamic load balancing techniques that can redistribute computational loads among processors during parallel analysis in a cost-effective fashion. In the present networked workstation environment where the cost of interprocess communication is high, the development of suitable dynamic load balancing techniques is a particularly challenging research task.

Fault Tolerance of Parallel Analysis

Parallel nonlinear analyses are usually expensive. Since the parallel analysis requires interprocess communication and synchronization among several processors in the computing network, the chances of failure due to a hardware, operation system, or communication problem are higher than those during a serial analysis. For example, failures due to some communication problems have been experienced by the writer during this research. To avoid costly reanalysis of the entire problem, future research should be conducted to provide fault tolerance capabilities in the present parallel analysis system.

One simple solution for the fault tolerance problem is a restart capability for parallel analysis. Essential data for analysis recovery are written out to data files at the user-specified points (often called the checkpoints) in the parallel analysis. If a failure occurs, the analysis can then be restarted by the user at the nearest checkpoint prior to the point of failure.

More complicated approaches may be devised to automate the restart sequence. For example, each processor used in the parallel analysis may be associated with a backup processor. If the primary processor fails, it will be

automatically replaced by its backup processor. During the analysis, the analysis recovery data at the checkpoints are either written to data files or sent to backup processors. An automatic mechanism is required to detect processor failures should they occur, to activate backup processors. and to restart the analysis at the nearest checkpoint prior to the point of failure.

Interactive Monitoring and Steering of Parallel Analysis

To provide a better interactive parallel analysis environment, the present parallel analysis system should be further developed to include interactive monitoring and steering capabilities. The suggested implementation of these capabilities in a networked workstation environment is explained using Fig. 6.1. To carry out a parallel analysis, copies of ABREAST are run on a number of workstations and communicate through ISIS. Also, through interprocess communication provided by ISIS, the following interactions between FRANSYS/BASYS, PSAINT, and the copies of ABREAST can be achieved:

(a) After requested by FRANSYS/BASYS, PSAINT collects the most recent analysis results as they are being calculated by the various ABREAST's and, then, sends them to FRANSYS/BASYS for interactive monitoring and visualization of the parallel analysis.

(b) FRANSYS/BASYS may steer the parallel analysis by requesting the ABREAST's (either through PSAINT or directly) to start/stop the analysis and modify specific analysis parameters during the analysis.

In addition, the graphics display of FRANSYS/BASYS can be visualized on any workstation specified by the user through the X Window system.

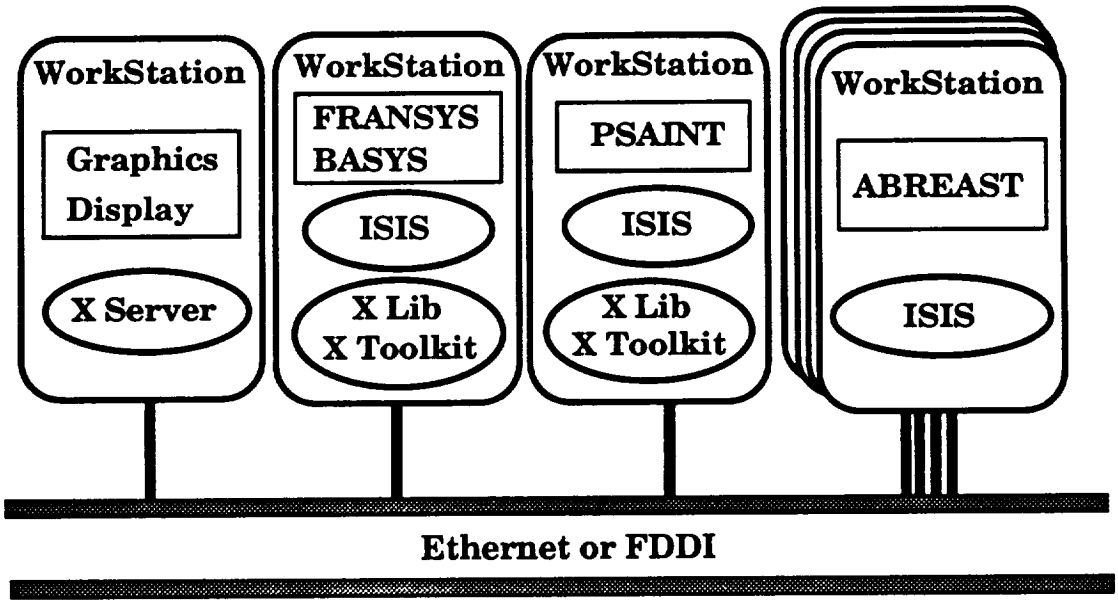


Figure 6.1 Interactive monitoring and steering of parallel analysis

Closure

In the foreseeable future, the rapidly advancing computer technologies promise to provide researchers and engineers increasingly powerful computing environments. Research should be conducted continuously to take advantage of these technologies for more efficient and powerful simulations of nonlinear structural dynamics.

Appendix A

Parallel Implementation Using ISIS

This appendix provides a brief description of what and how ISIS routines are used in the present research to achieve interprocess communication and synchronization for all implemented parallel algorithms discussed in Section 3.3. Example code segments extracted directly from the analysis program ABREAST are provided with all ISIS routine calls highlighted in bold. For detailed description of these ISIS routines, the ISIS manual (Birman et al. 1991) should be consulted.

A.1 General Purpose Routines

In the present implementation, processes are assigned unique ID numbers based on the sequence they join the ISIS process group. The following routine returns the ID number of the calling process in the process group specified by the groupview pointer gv:

```
int dpp_GetMyID( t_gv )
   char *t_gv ;  /* groupview ptr of the process group */
{
 groupview *gv = (groupview *)t_gv ;
 int index = 0 ;

 while( !addr_ismine( &gv->gv_members[index] ) )
  index++ ;

 return( ++index ) ;
}
```

If there are N processes in the process group, the ID number returned from the above routine for the first process is 1 and that for the last one is N.

When a message is received by a process, it is also necessary for the process to know the message sender's ID number so that the data in the message can be handled correctly. The following routine returns the message sender's ID number:

```
int dpp_GetSenderID( t_gv, t_msg_p )
   char *t_gv ; /* groupview ptr of the process group */
   char *t_msg_p ; /* ptr to message received */
{
 groupview *gv= (groupview *)t_gv ;
 message   *msg_p = (message *)t_msg_p ;
 address   *sender_addr ;  /* message sender's address */
 int       index = 0 ;

 sender_addr = msg_getsender( msg_p ) ;

 while( !addr_isequal(sender_addr,&gv->gv_members[index]) )
  index++ ;

 return( ++index ) ;
}
```

A.2 Initialization

To run ABREAST under ISIS, a start sequence is required which initializes the ISIS system (i.e., connects the application to ISIS), registers all processes in an ISIS process group, and declares all ISIS entry points for interprocess communication. For example, the following initialization routine is implemented for the parallel central difference algorithm:

```
static address   *gaddr_p ;  /* addr ptr of the ISIS process
                        group 'pp_cd_pg' */
static groupview *gview_p ;  /* groupview ptr of the ISIS
                        process group 'pp_cd_pg' */
void dpp_cd_Init()
{
   /* (Function declarations are omitted) */
   int      my_sbs_id, n_sbs ;
   char     pp_cd_pg[PG_GLEN] ;
          /* name of process group containing all
          ** processes running parallel CD algorithm */

/* Initialize ISIS */
   isis_remote_init( (char*)0, 0, 0, ISIS_USEITIMER) ;

/* Finish ISIS startup */
   isis_start_done() ;

/* Joint the process group (Note: n_sbs is used to detect
** the completion of process-group joining.) */
   ads_analysis( GET_PROB_NAME, pp_cd_pg ) ;
   n_sbs = (int) ads_css( GET_NUM_SBS ) ;
   gaddr_p = pg_join( pp_cd_pg, 0 ) ;
   gview_p = pg_getview( gaddr_p ) ;

   while( (int) gview_p->gv_nmemb < n_sbs )
      isis_accept_events( ISIS_BLOCK ) ;
      /* wait (block) until all members have joined */

/* Get a unique ID for this process */
   my_sbs_id = dpp_GetMyID( gview_p ) ;
   ads_css( SET_SBS_ID, 0, my_sbs_id ) ;

/* Declare ISIS entries for interprocess communication */
   isis_entry( PP_CD_RECEIVE, dpp_cd_ReceiveDispl,
                        "dpp_cd_ReceiveDispl" ) ;
```

```
  isis_entry( PP_CD_NOTIFY, dpp_cd_ReceiveNote,
                    "dpp_cd_ReceiveNote" ) ;
}
```

The initialization routine implemented for the parallel implicit domain decomposition algorithm follows the same steps as the one given above.

A.3 Interprocess Communication and Synchronization

This section illustrates the interprocess communication and synchronization routines implemented for both parallel central difference algorithm and parallel implicit domain decomposition algorithm.

Parallel Central Difference Algorithm

Within each time step of the parallel central difference algorithm, interprocess communication is required between adjacent processors to exchange the displacements of the boundary nodes (see Step (5) in Table 3.1). The routines implemented to achieve this interprocess communication are given as follows:

```
static int      n_note_got = 0 ;
static int      n_msg_got = 0 ;

void dpp_cd_Commun()
{
  /* (Function declarations are omitted) */
  int    n_adj_sbs = (int)ads_css( GET_N_ADJ_SBS, 0 ) ;

/* notify all adjacent processors that sending displacements
** is now allowed. (Note: this is to avoid the possible race
** condition that adjacent processors may send displacements
** right before this processor escapes from the "ISIS_BLOCK"
** loop after calling dpp_cd_SendDisp().) */
```

```
   dpp_cd_Notify() ;

/* Wait until all replies needed have been received */
   while ( n_note_got < n_adj_sbs ) {
      isis_accept_events( ISIS_BLOCK ) ;
      }
   n_note_got = 0 ;          /* Reset n_note_got */

/* Send displacements of "boundary nodes" to adjacent
** substructures */
   dpp_cd_SendDispl() ;

/* wait until displacements of "adjacent vertices" from all
** adjacent substructures have been received (Note: the
** routine dpp_cd_ReceiveDispl() for receiving displacements
** is defined in dpp_cd_Init() as an ISIS entry which is
** activated automatically by ISIS whenever a message for it
** arrives.) */
   while ( n_msg_got < n_adj_sbs ) {
      isis_accept_events( ISIS_BLOCK ) ;
      }
   n_msg_got = 0 ;           /* Reset n_msg_got */
}

void dpp_cd_Notify()
{
   int i ;
   int n_adj_sbs  = (int)ads_css( GET_N_ADJ_SBS, 0 ) ;
   int *adj_sbs_l = (int *)ads_css( GET_ADJ_SBS_LIST, 0 ) ;
   address send_to ;

/* Send notes to all adjacent processes (substructures)*/
   for ( i=0 ; i < n_adj_sbs ; i++ ) {
      send_to = gview_p->gv_members[ adj_sbs_l[i] ] ;
      bcast( &send_to, PP_CD_NOTIFY, "%d", 1, 0 ) ;
      }
}
```

```
void dpp_cd_ReceiveNote( msg_p )
message   *msg_p ;
{
   int    note ;

   msg_get( msg_p, "%d", &note ) ;
   n_note_got++ ;
}

void dpp_cd_SendDispl()
{
   /* (Function declarations are omitted) */
   int        n_adj_sbs, *adj_sbs_l ;
   int        msg_len ;  /* length of msg_buf[] */
   double     *msg_buf ; /* ptr to message buffer */
   address    send_to ;

/* Query basic information */
   n_adj_sbs    = (int)ads_css( GET_N_ADJ_SBS, 0 ) ;
   adj_sbs_l    = (int *)ads_css( GET_ADJ_SBS_LIST, 0 ) ;

/* Send displacements to all adjacent processes */
   for ( i=0 ; i < n_adj_sbs ; i++ ) {
      /* For the current adjacent substructure, compute
        msg_len, allocate memory for msg_buf using
        calloc(), and put the displacements of boundary
        nodes into msg_buf */
      /* (Codes omitted) */

      /* broadcast the displacements */
      send_to = gview_p->gv_members[ adj_sbs_l[i] ] ;
      bcast( &send_to, PP_CD_RECEIVE, "%*G", msg_buf,
                              msg_len, free, 0 ) ;
      }
 }
```

```
void dpp_cd_ReceiveDispl( msg_p )
message    *msg_p ;
{
   /* (Function declarations are omitted) */
   int        sender_id ;
   double     *msg_buf ;
   address    send_to ;

/* Query basic information */
   /* (Codes omitted) */

/* Receive displacements from adjacent substructures */
   msg_get( msg_p, "%+G", &msg_buf, &n_elem ) ;

/* get the message sender's ID */
   sender_id = dpp_GetSenderID( gview_p, msg_p ) ;

/* set the displacements into database */
   /* (Codes omitted) */

/* Free the message buffer and increase n_msg_got by 1 */
   free( (void *) msg_buf ) ;
   n_msg_got++ ;

}
```

Parallel Implicit Domain Decomposition Algorithm

Two types of interprocess communication and synchronization are required for the parallel implicit domain decomposition algorithm. The first one is the assembly of the vector at substructure boundary by adding contributions from adjacent substructures (See also Table 3.3), while the second one is the computation of a global dot product by summing contributions from all substructures (See also Table 3.5). The routines

implemented to achieve both types of interprocess communication and synchronization are given as follows.

Assembly of the Vector at Substructure Boundary

```
static address *dd_ga ;  /* ptr to addr of the process
                         ** group containing all processes
                         ** running the parallel domain
                         ** decomposition algorithm*/
static groupview *dd_gv ; /* Ptr to groupview of the process
                          ** group containing all processes
                          ** running the parallel domain
                          ** decomposition algorithm */
static int *n_bdry_vec_note_got=0;/
static int *n_bdry_vec_got=0;
static double **vec_own = 0; /* Ptrs for temporarily holding
                             ** the addresses of boundary
                             ** vectors (of this process) */

static void AssembleBoundaryVector(id, v)
   BdryVecType id ; /* type id of the boundary vector */
   double     *v ; /* the boundary vector of type id */
{
   int   n_sbs = (int)ads_css( GET_NUM_SBS ) ;
   int   n_adj_sbs = (int)ads_css( GET_N_ADJ_SBS, 0 ) ;
   int  i ;

/* put the boundary vector into buffer vec_own[type_id][] */
   for(i=0; i<n_h_dof; i++)
      vec_own[(int)id][i] = v[i] ;

/* Synchronize all processors */
   NotifySendingBoundaryVector( id );
   while( n_bdry_vec_note_got[id] < n_sbs )
      isis_accept_events( ISIS_BLOCK ) ;
   n_bdry_vec_note_got[id] = 0 ;  /* reset the counter */
```

```
/* Send boundary vector */
   SendBoundaryVector( id, v ) ;

/* Wait until contributions from all adjacent processes
** have been received */
   while( n_bdry_vec_got[id] < n_adj_sbs )
      isis_accept_events( ISIS_BLOCK ) ;
   n_bdry_vec_got[id] = 0 ;  /* reset the counter */

/* put the assembled vector into v */
   for(i=0; i<n_h_dof; i++)
      v[i] = vec_own[(int)id][i] ;
}

static void NotifySendingBoundaryVector( id )
   BdryVecType id; /* type id of the boundary vector */
{
   int     i ;
   int     n_sbs  = (int) ads_css( GET_NUM_SBS ) ;
   address  send_to ;

/* Send notes to all processes for synchronization */
   for(i=0; i<n_sbs; i++){
      send_to = dd_gv->gv_members[ i ] ;
      bcast( &send_to, PP_DD_RecvBdryVecNote, "%d,%d",
                                 1, id, 0 ) ;
      }
}

static void RecvBdryVecNote( t_msg_p )
   char  *t_msg_p ;  /* ptr to the message */
{
   message    *msg_p = (message *)t_msg_p ;
   int        note ;
   BdryVecType id ;

   msg_get( msg_p, "%d,%d", &note, &id ) ;
```

```
    n_bdry_vec_note_got[id] += note ;
}

static void SendBoundaryVector(id, v)
    BdryVecType id ; /* type id of the boundary vector */
    double     *v ; /* the boundary vector to be sent */
{
    int     i ;
    address  send_to ;
    double  *msg_buf;
    int     msg_len ;
    int   n_adj_sbs  = (int) ads_css(GET_N_ADJ_SBS, 0) ;
    int   *adj_sbs_l = (int *)ads_css(GET_ADJ_SBS_LIST, 0) ;

/* Send the vector to adjacent processes */
    for(i=0; i<n_adj_sbs; i++){
      /* compute msg_len, allocate memory for msg_buf
        using malloc(), and put v into msg_buf */
      /* (Codes omitted) */
      send_to = dd_gv->gv_members[ adj_sbs_l[i] ] ;
      bcast( &send_to, PP_DD_RecvBdryVec, "%d,%*G", id,
                    msg_buf, msg_len, free,  0 ) ;
      }
}

static void RecvBdryVec( msg_p )
    message  *msg_p ;
{
    int         sender_id ;
    BdryVecType  id;
    double      *v ;
    int         n_h_dof_recv ;

/* Receive the message */
    msg_get( msg_p, "%d,%+G", &id, &v, &n_h_dof_recv ) ;
```

```
/* Figure out which adjacent substructure sends this data */
   sender_id = dpp_GetSenderID( dd_gv, msg_p );

/* Assemble the contributions of the adjacent boundary
** vector into buffer vec_own.*/
   /* (Codes omitted) */

   n_bdry_vec_got[id]++;
   free(v) ;
}
```

Assembly of a Global Dot Product

```
static int    n_ldotprod_note_got=0;
static int    n_ldotprod_got=0;
static int    n_gdotprod_got=0;
static int    n_gdotprod_note_got=0;
static double  gdot ;  /* value of global dot product*/

double dpp_DD_AsmDotProduct(a, b)
   double *a, *b ;
{
int    n_sbs = (int) ads_css( GET_NUM_SBS ) ;
double  ldot ;
int    uno = 1;
int    my_id ;

   gdot = 0.0 ; /* initialize global dot product */

/* Synchronize all processes */
   NotifySendingLDotProd();
   while( n_ldotprod_note_got < n_sbs )
      isis_accept_events( ISIS_BLOCK ) ;
   n_ldotprod_note_got = 0 ; /* reset the counter */

/* Compute own share of dot product */
   ldot = DDOT(&n_pr_h_dof, a, &uno, b, &uno) ;
```

```
/* Send own share of dot product */
   SendLDotProd(ldot);

/* If this is the host processor, collect all the dot
** product contributions */
   my_id = ads_css( GET_SBS_ID, 0 ) ;
   if((my_id - 1) == 0) { /* 0th is designated the host */
      while( n_ldotprod_got < n_sbs )
         isis_accept_events( ISIS_BLOCK ) ;
      n_ldotprod_got = 0 ;  /* reset the counter */
      /* send assembled global dot product to all
        processes */
      SendGDotProd() ;
      }

/* Wait until the global dot product has been received */
   while( n_gdotprod_got < 1 )
      isis_accept_events( ISIS_BLOCK );
   n_gdotprod_got = 0 ;  /* reset the counter */

   return( gdot ) ;
}

static void NotifySendingLDotProd()
{
   int     i ;
   int     n_sbs  = (int)ads_css( GET_NUM_SBS ) ;
   address  send_to ;

/* Send notes to adjacent processes */
   for(i=0; i<n_sbs; i++){
      send_to = dd_gv->gv_members[i] ;
      bcast(&send_to,PP_DD_RecvLDotProdNote,"%d",1,0) ;
      }
}
```

```
static void RecvLDotProdNote( msg_p )
   message  *msg_p ;
{
   int       note ;
   BdryVecType id ;

   msg_get( msg_p, "%d", &note ) ;
   n_ldotprod_note_got += note ;
}

static void SendLDotProd( d )
   double  d ;
{
   address  send_to ;

/* Send the local dot product to the host processor */
   send_to = dd_gv->gv_members[0] ;
                  /* 0th is designated the host */
   bcast( &send_to, PP_DD_RecvLDotProd, "%G[1]", &d, 0 ) ;
}

static void SendGDotProd()
{
   int       i ;
   int       n_sbs = (int)ads_css( GET_NUM_SBS ) ;
   address     send_to ;

/* Send the global dot product to all processes */
   for(i=0; i<n_sbs; i++){
     send_to = dd_gv->gv_members[i] ;
     bcast(&send_to,PP_DD_RecvGDotProd,"%G[1]",&gdot,0) ;
   }
}

static void RecvLDotProd(msg_p)
   message  *msg_p ;
```

```
{
   double  dot ;

/* receive a local dot product */
   msg_get( msg_p, "%g", &dot ) ;

/* Add the local dot product into the global dot product */
   gdot += dot ;

   n_ldotprod_got++ ;
}

static void RecvGDotProd( msg_p )
   message  *msg_p ;
{
   double  dot ;

/* Receive the global dot product */
   msg_get( msg_p, "%g", &dot ) ;
   gdot = dot ;

   n_gdotprod_got++ ;
}
```

A.4 Termination

After the parallel analysis is complete, a termination sequence is required to disconnect the application from ISIS. For example, the following termination routine is implemented for the parallel central difference (CD) algorithm:

```
void dpp_cd_Done()
{
   int        n_sbs = (int)ads_css(GET_NUM_SBS) ;
   address    send_to ;
   int        i ;
```

```
/* Notify all members in the process group 'pp_cd_pg' that
** this process is leaving */
   for(i=0; i<n_sbs; i++){
     send_to = gview_p->gv_members[i] ;
     bcast( &send_to, PP_CD_NOTIFY, "%d", 2, 0 ) ;
   }

/* Wait until all members are ready to leave the process
** group */
   while( n_note_got < (int) gview_p->gv_nmemb ) {
     isis_accept_events( ISIS_ASYNC ) ;
     }
   n_note_got = 0 ;  /* reset the counter n_note_got */

/* Leave the process group */
   pg_leave( gaddr_p ) ;
}
```

The termination routine implemented for the parallel implicit domain decomposition (DD) algorithm follows the same steps as the one given above.

Bibliography

Abbas, B. A. H. and K. M. Kamal (1987). "Vibration of Turbomachinery Blades with Root Flexibility Effect", *Bladed Disk Assemblies*, ASME Publication DE-Vol. 6, No. H00406, 31-41.

Abdullah, A. R., and D. J. Evans (1986). "A Weighted Group Explicit Method for the Diffusion Equation," *Computer Methods in Applied Mechanics and Engineering*, Vol. 55, No. 3, 221-238.

Abel, J. F., B. H. Aubert, S. H. Hsieh (1991). "On the Use of a Workstation Network for the Parallel Solution of Nonlinear Structural Dynamics," presented in the First U.S. National Congress on Computational Mechanics, Chicago, IL, July 21-24.

Al-Nasra, M., and D. T. Nguyen (1991). "An Algorithm for Domain Decomposition in Finite Element Analysis," *Computers and Structures*, Vol. 39, No. 3/4, 277-289.

Argyris, J., O. Hilpert, G. Malejannakis, and D. Scharpf (1979). "On the Geometrical Stiffness of a Beam in Space," *Computer Methods in Applied Mechanics and Engineering,* Vol. 20, No. 1, 105-131.

Aubert, B. H. (1992). "Numerical Simulation of the Transient Nonlinear Dynamics of Actively Controlled Space Structures," Ph.D. dissertation, Cornell University, Ithaca, New York.

Aubert, B. H., S. H. Hsieh, and J. F. Abel (1992). "Simulation of the Dynamics of Large Space Structures," in N. K. Srivastava, A. N. Sherbourne, and J. Roorda, Eds., *Innovative Large Span Structures*, Vol. 1, Toronto, Canada, 570-580.

Barnard, S. T., and H. D. Simon (1992). "A Fast Multilevel Implementation of Recursive Spectral Bisection for Partitioning Unstructured Problems," Report RNR-92-033, NAS Systems Division, Applied Research Branch, NASA Ames Research Center, Mail Stop T045-1, Moffett Field, CA 94035.

Bathe, K. J. (1982). *Finite Element Procedures in Engineering Analysis*, Prentice Hall, New Jersey.

Birman, K. P., R. Cooper, T. A. Joseph, K. P. Kane, and F. Schmuck (1991). *ISIS - A Distributed Programming Environment, Version 3.0 - User's Guide and Reference Manual.* Department of Computer Science, Cornell University, Ithaca, New York.

Bossak, M. A. J., and O. C. Zienkiewicz (1973). "Free Vibration of Initially Stressed Solids, with Particular Reference to Centrifugal-force

Effects in Rotating Machinery", *Journal of Strain Analysis*, Vol. 8, 245-252.

Chiang, K. N., and R. E. Fulton (1990). "Structural Dynamics Methods for Concurrent Processing Computers," *Computers and Structures*, Vol. 36, No. 6, 1031-1037.

Collins, R. J. (1973). "Bandwidth Reduction by Automatic Renumbering," *International Journal for Numerical Methods in Engineering*, Vol. 6, 345-356.

Cook, R. D., D. S. Malkus, and M. E. Plesha (1989). *Concepts and Applications of Finite Element Analysis*, 3rd Ed., Wiley, New York.

Davis, R. R. (1989). "Practical Nonlinear Simulation of Rotating Machinery Dynamics With Application to Turbine Blade Rubbing," Ph.D. dissertation, University of California at Davis, Davis, California, June 1989.

Dokanish, M. A., and S. Rawtani (1971). "Vibration Analysis of Rotating Cantilever Plates," *International Journal for Numerical Methods in Engineering*, Vol. 3, 233-248.

Ernst, M. A., and C. Lawrence (1987). "Hub Flexibility Effects on Propfan Characteristics", NASA TM-89900.

Evans, D. J. (1984). "The Group Explicit Method for the Numerical Solution of Non-linear Partial Differential Equations," in R. W. Lewis, E. Hinton, P. Bettess, and B. A. Schrefler, Eds., *Numerical Methods for Transient and Coupled Problems*, Proceedings of the International Conference on Numerical Methods for Transient and Coupled Problems, Venice, Italy, 9-13 July 1984, Pineridge Press, Swansea, U. K., 801-815.

Evans, D. J. (1985). "The Numerical Solution of Non-linear Parabolic Equations on Parallel Computers by Group Explicit Methods," in J. Middleton, G. N. Pande, and A. A. Balkema, Eds., *NUMETA '85 Numerical Methods in Engineering: Theory and Applications*, Proceedings of the International Conference on Numerical Methods in Engineering, Vol. 2, Swansea, U. K., 7-11 January 1985, 973-980.

Evans, D. J., and N. Y. Yousif (1992). "Asynchronous Parallel Algorithms for Linear Equations," in H. Adeli, Ed., *Parallel Processing in Computational Mechanics*, Marcel Dekker, Inc., New York, 69-130.

Farhat, C. (1987). "Multiprocessors in Computational Mechanics," Ph.D. dissertation, University of California, Berkeley, California.

Farhat, C. (1988). "A Simple and Efficient Automatic FEM Domain Decomposer," *Computers and Structures*, Vol. 28, No. 5, 579-602.

Farhat, C. (1992). Personal Communication.

Farhat, C., and L. Crivelli (1989). "A General Approach to Nonlinear FE Computations on Shared Memory Multiprocessors," *Computer Methods in Applied Mechanics and Engineering*, Vol. 72, No. 2, 153-172.

Farhat, C., and N. Sobh (1990). "A Consistency Analysis of a Class of Concurrent Transient Implicit/Explicit Algorithms," *Computer Methods in Applied Mechanics and Engineering*, Vol. 84, 147-162.

Farhat, C., and E. Wilson (1988). "A Parallel Active Column Equation Solver," *Computers and Structures*, Vol. 28, No. 2, 289-304.

Fenves, S. J., and K. H. Law (1983). "A Two-Step Approach to Finite Element Ordering," *International Journal for Numerical Methods in Engineering*, Vol. 19, 891-911.

Fiedler, M. (1975). "A Properties of Eigenvectors of Nonnegative Symmetric Matrices and Its Application to Graph Theory," *Czechoslovak Mathematics Journal*, Vol. 25, No. 100, 607-618.

Flower, J., S. Otto, and M. Salama (1987). "Optimal Mapping of Irregular Finite Element Domains to Parallel Processors," in A. K. Noor, Ed., *Parallel Computations and Their Impact on Mechanics*, ASME, New York, 239-250.

Gallagher, R. H., "Problems and Progress in Thin Shell Finite Element Analysis," in D. G. Ashwell and R. H. Gallagher, Eds., *Finite Elements for Thin Shell and Curved Members*, Wiley, New York.

Garey, M. R., and D. S. Johnson (1979). *Computers and Intractability: A Guide to The Theory of NP-Completeness*, W. H. Freeman and Company, New York.

Geist, G. A., M. T. Heath, B. W. Peyton, and P. H. Worley (1991). "A Users' Guide to PICL - A Portable Instrumented Communication Library," ORNL/TM-11616, Oak Ridge National Laboratory, Oak Ridge, Tennessee 37831.

George, A. (1971). "Computer Implementation of the Finite Element Method," Technical Report STAN-CS-71-208, Computer Science Department, Stanford University.

Gibbs, N. E., W. G. Poole, Jr., and P. K. Stockmeyer (1976). "An Algorithm for Reducing the Bandwidth and Profile of a Sparse Matrix," *SIAM Journal of Numerical Analysis*, Vol. 13, No. 2, 237-250.

Golub, G. H., and C. Van Loan (1989). *Matrix Computations*, 2nd Ed., The Johns Hopkins University Press, Baltimore, Maryland.

Hageman, L. A., and D. M. Young (1981). *Applied Iterative Methods*, Academic Press, New York.

Hajjar, J. F. (1987). "Parallel Processing for Transient Nonlinear Structural Dynamics of Three-Dimensional Framed Structures," Department of Structural Engineering Report No. 87-5, Cornell University, Ithaca, New York, November 1987.

Hajjar, J. F. (1988). "Parallel Processing for Transient Nonlinear Structural Dynamics of Three-Dimensional Framed Structures," Ph.D. dissertation, Cornell University, Ithaca, New York, January 1988.

Hajjar, J. F., and J. F. Abel (1988). "Parallel Processing for Transient Nonlinear Structural Dynamics of Three-Dimensional Framed Structures Using Domain Decomposition," *Computers and Structures*, Vol. 30, No. 6, 1237-1254.

Hajjar, J. F., and J. F. Abel (1989a). "Parallel Processing of Central Difference Transient Analysis for Three-Dimensional Nonlinear Framed Structures," *Communications in Applied Numerical Methods*, Vol. 5, 39-46.

Hajjar, J. F., and J. F. Abel (1989b). "On the Accuracy of Some Domain-by-domain Algorithms for Parallel Processing of Transient Structural Dynamics," *International Journal for Numerical Methods in Engineering*, Vol. 28, 1855-1874.

Han, T. Y., and J. F. Abel (1984). "Substructure Condensation Using Modified Decomposition," *International Journal for Numerical Methods in Engineering*, Vol. 20, No. 11, 1959-1964.

Hendrickson, B., and R. Leland (1992). "An Improved Spectral Graph Partitioning Algorithm for Mapping Parallel Computations," Technical Report SAND92-1460, Sandia National Laboratories, Albuquerque, New Mexico.

Hinton, E., and P. R. Benson (1976). "A Thick Finite Strip Solution for Static, Free Vibration and Stability Problems," *International Journal for Numerical Methods in Engineering*, Vol. 10, 665-678.

Hilmy, S. I. (1984). "Adaptive Nonlinear Dynamic Analysis of Three-Dimensional Steel Framed Structures with Interactive Computer Graphics," Ph.D. dissertation, Cornell University, Ithaca, New York.

Hilmy, S., and J. F. Abel (1985). "A Strain-Hardening Concentrated Plasticity Model for Nonlinear Analysis of Steel Buildings," *NUMETA 85 Numerical Methods in Engineering: Theory and Application*; Proceedings of International Conference on Numerical Methods in Engineering, Swansea, U. K., Vol. 1, 305-314.

Hsieh, S. H., and S. Srivastav (1992). "PSAINT: An Parallel Structural Analysis Interface," unpublished internal document, Program of Computer Graphics, Cornell University, Ithaca, New York.

Hughes, T. R. J., and T. Belytschko (1983). "A Precis of Developments in Computational Methods for Transient Analysis," *Journal of Applied Mechanics*, Vol. 50, No. 46, 1033-1041.

Irie, T., G. Yamada, and K. Takagi (1982). "Natural Frequencies of Thick Annular Plates," *Journal of Applied Mechanics*, Vol. 49, 633-638.

Irretier, H. (1985). "Computer Simulation of the Run-Up of a Turbine Blade Subjected to Partial Admission," ASME Paper 85-DET-128.

Jennings, A. (1977). *Matrix Computation for Engineers and Scientists*, John Wiley and Sons, New York.

Johnson, S. E., and E. I. Field (1973). "Three Isoparametric Solid Elements for NASTRAN," NASA TM x-2893, *NASTRAN: User's Experiences*, 423-437.

Kielb, R. E., A. W. Leissa, and J. C. MacBain (1985). "Vibrations of Twisted Cantilever Plates - A Comparison of Theoretical Results," *International Journal of Numerical Methods in Engineering*, Vol. 21, 1365-1380.

Khader, N., and G. Abu-Farsakh (1990). "A Triangular Shell Element for Vibration Analysis of Cambered and Twisted Fan Blades," *Finite Elements in Analysis and Design*, Vol. 6, 287-301.

Khulief, Y. A., and A. Bazoune (1992). "Frequencies of Rotating Tapered Timoshenko Beams with Different Boundary Conditions," *Computers and Structures*, Vol. 42, No. 5, 781-795.

Kubiak, J. A., O. Yrigoyen, A. Carnero, and J. Aguirre (1987). "Analysis of the Last Stage Blade Group", *Bladed Disk Assemblies*, ASME Publication DE-Vol. 6., No. H00406, 101-111.

Lawrence, C., and R. E. Kielb (1984) "Nonlinear Displacement Analysis of Advanced Propeller Structures Using NASTRAN", NASA TM-83737.

Leissa, A. W., J. C. MacBain, and R. E. Kielb (1984). "Vibrations of Twisted Cantilever Plates - Summary of Previous and Current Studies," *Journal of Sound and Vibration*, Vol. 96, No. 2, 159-173.

Leler, W. (1990). "Linda meets Unix," *Computer,* Vol. 23, No. 2, 43-54.

Lin, T. W. (1980). *Basic Application Programs in Engineering* (in Chinese), New Knowledge Education Press, Taipei, Taiwan.

Liu, W. K. (1987). "Parallel Computations for Mixed-time Integrations," in R. W. Lewis, E. Hinton, P. Bettess, and B. A. Schrefler, Eds., *Numerical Methods for Transient and Coupled Problems*, Chapter 13, John Wiley & Sons Ltd., 261-277.

Liu, W. K., and T. Belytschko (1982). "Mixed-Time Implicit-Explicit Finite Elements for Transient Analysis," *Computers and Structures*, Vol. 15, No. 4, 445-450.

MacBain, J. C. (1975). "Vibrating Behaviour of Twisted Cantilever Plates," *Journal of Aircraft*, Vol. 12, 343-349.

MacBain, J. C., R. E. Kielb, and A. W. Leissa (1985). "Vibrations of Twisted Cantilever Plates - Experimental Investigation," *Journal of Engineering for Gas Turbines and Power*, Vol. 107, 187-196.

MacNeal, R. H., and R. L. Harder (1985). "A Proposed Standard Set of Problems to Test Finite Element Accuracy," *Finite Element in Analysis and Design*, Vol. 1, 3-20.

Malone, J. G. (1988). "Automatic Mesh Decomposition and Concurrent Finite Element Analysis for Hypercube Multiprocessor Computers," *Computer Methods in Applied Mechanics and Engineering*, Vol. 70, 27-58.

Malone, J. G. (1990). "Parallel Nonlinear Dynamic Finite Element Analysis of Three-Dimensional Shell Structures," *Computers and Structures*, Vol. 35, No. 5, 523-539.

Martha, L. F. C. R. (1989). "Topological and Geometrical Modeling Approach to Numerical Discretization and Arbitrary Fracture Simulation in Three-Dimensions," Ph.D. dissertation, Cornell University, Ithaca, New York.

McGee, O. G. (1992). "Performance of Continuum and Discrete Three-Dimensional Vibration Analyses of Twisted Cantilevered Parallelepipeds," *Computers and Structures*, Vol. 42, No. 2, 211-227.

Midturi, S., M. L. Soni, W. A. Stange, and J. D. Reed (1987). "On Model Generation and Modal Analysis of Flexible Bladed Disk Assemblies", *Bladed Disk Assemblies*, ASME Publication DE-Vol. 6, No. H00406, 49-54.

McGee, O. G. (1987). "Finite Element Analysis of Flexible Rotating Blades", NASA TM-89906.

McGuire, W., and Gallagher, R. H. (1979). *Matrix Structural Analysis*, John Wiley & Sons, New York.

McGuire, W., G. G. Deierlein, T. K. Sooi, and Y. Zhao (1989). "Illustrated Primer for CU-PREPF, CU-STAND, and CU-QUAND," Structural Engineering Report 89-12, School of Civil and Environmental Engineering, Cornell University, Ithaca, New York.

Mota Soares, C. A., M. Petyt, and A. M. Salama (1976). "Finite Element Analysis of Bladed Disks," in A. V. Srinivasan and Pratt & Whitney Aircraft, Eds., *Structural Dynamic Aspects of Bladed Disk Assemblies*, ASME Winter Annual Meeting, New York, 73-91.

Mota Soares, C. A., and M. Petyt (1978a). "Finite Element Dynamic Analysis of Practical Discs," *Journal of Sound and Vibration*, Vol. 61, No. 4, 547-560.

Mota Soares, C. A., and M. Petyt (1978b). "Finite Element Dynamic Analysis of Practical Bladed Discs," *Journal of Sound and Vibration*, Vol. 61, No. 4, 561-570.

Mullen, R., and T. Belytschko (1983). "An Analysis of an Unconditionally Stable Explicit Method," *Computers and Structures*, Vol. 16, No. 6, 691-696.

Nour-Omid, B., A. Raefsky, and G. Lyzenga (1987). "Solving Finite Element Equations on Concurrent Computers," in A. K. Noor, Ed., *Parallel Computations and Their Impact on Mechanics*, ASME, New York, 209-227.

Omprakash, V., and V. Ramamurti (1988). "Analysis of Bladed Disks - A Review," *Shock and Vibration Digest*, Vol. 20, No. 11, 14-21.

Omprakash, V., and V. Ramamurti (1989). "Dynamic Stress Analysis of Rotating Turbo-machinery Bladed-Disk Systems," *Computers and Structures*, Vol. 32, No. 2, 477-488.

Omprakash, V., and V. Ramamurti (1990a). "Coupled Free Vibration Characteristics of Rotating Tuned Bladed Disk Systems," *Journal of Sound and Vibration*, Vol. 140, No. 3, 413-435.

Omprakash, V., and V. Ramamurti (1990b). "Spectral Analysis of the Transient Characteristics of a Bladed Disk During Run-up," *Computers and Structures*, Vol. 37, No. 6, 983-992.

Ortiz, M. (1991). "Discussion on 'A Consistency Analysis of a Class of Concurrent Transient Implicit/Explicit Algorithms', by C. Farhat and N. Sobh," *Computer Methods in Applied Mechanics and Engineering*, Vol. 92, 397-398.

Ortiz, M., and B. Nour-Omid (1986). "Unconditionally Stable Concurrent Procedures for Transient Finite Element Analysis," *Computer*

Methods in Applied Mechanics and Engineering, Vol. 58, No. 2, 151-174.

Ortiz, M., B. Nour-Omid, and E. D. Sotelino (1988) "Accuracy of a Class of Concurrent Algorithms for Transient Finite Element Analysis," *International Journal of Numerical Methods in Engineering*, Vol. 26, 379-391.

Owen, D. R. J. (1980). "Implicit Finite Element Methods for the Dynamic Transient Analysis of Solids with Particular Reference to Non-Linear Situations," in J. Donea, Ed., *Advanced Structural Dynamics*, Applied Science, London, 123-190.

Padovan, J., and A. Kwang (1991). "Hierarchically Parallelized Constrained Nonlinear Solvers with Automated Substructuring," *Computers and Structures*, Vol. 41, No. 1, 7-33.

Park, K. C. (1982). "An Improved Semi-Implicit Method for Structural Dynamics Analysis," *Journal of Applied Mechanics*, Vol. 49, 589-593.

Park, K. C., and J. M. Housner (1982). "Semi-Implicit Transient Analysis Procedures for Structural Dynamics Analysis," *International Journal for Numerical Methods in Engineering*, Vol. 18, 609-622.

Paulino, G. H. (1988). "Preprocessing of Three-Dimensional Space Frames, with Nodal Reordering, using Interactive Computer Graphics," M.Sc. Dissertation, Department of Civil Engineering, PUC, Rio de Janeiro (in portuguese).

Petricone, R., and F. Sisto (1971). "Vibration Characteristics of Low Aspect Ratio Compressor Blades," *Journal of Engineering for Power*, Vol. 93, 103-110.

Petyt, M. (1990). *Introduction to Finite Element Vibration Analysis*, Cambridge University Press.

Pothen, A., H. Simon, and K.-P. Liou (1990). "Partitioning Sparse Matrices with Eigenvectors of Graphs," *SIAM Journal of Matrix Analysis and Application*, Vol. 11, No. 3, 430-452.

Potyondy, D. (1992). "A PEX Implementation for 3-D Graphics in FRANSYS," unpublished internal document, Program of Computer Graphics, Cornell University, Ithaca, New York.

Potyondy, D. (1991). "Object-oriented Solid Modeller (OSM): Initial Specification," unpublished internal document, Program of Computer Graphics, Cornell University, Ithaca, New York.

Putter, S., and H. Manor (1978). "Natural Frequencies of Radial Rotating Beams," *Journal of Sound and Vibration*, Vol. 56, No. 2, 175-185.

Ramamurti, V., and P. Balasubramanian (1984). "Analysis of Turbomachine Blades -- A Review," *Shock and Vibration Digest*, Vol. 16, No. 8, 13-28.

Rao, J. S. (1987). "Turbomachine Blade Vibration," *Shock and Vibration Digest*, Vol. 19, No. 5, 3-10.

Rao, S. S., and A. S. Prasad (1975). "Vibrations of Annular Plates Including the Effects of Rotatory Inertia and Transverse Shear Deformation," *Journal of Sound and Vibration*, Vol. 42, No. 3, 305-324.

Rost, R. J., J. D. Friedberg, and P. L. Nishimoto (1989). "PEX: A Network-transparent 3D Graphics System," *IEEE Computer Graphics and Applications*, 14-26.

Saul'yev, V. K. (1964). *Integration of Equations of Parabolic Type by the Method of Nets*, The MacMillan Company, New York.

Scheifler, R. W., and J. Gettys (1986). "The X Window System," *ACM Transactions on Graphics*, Vol. 5, No. 2, 79-109.

Simo, J. C., and L. Vu-Quoc (1987). "The Role of Non-linear Theories in Transient Dynamic Analysis of Flexible Structures," *Journal of Sound and Vibration*, Vol. 119, No. 3, 487-508.

Simon, H. D. (1991). "Partitioning of Unstructured Problems for Parallel Processing," *Computing Systems in Engineering*, Vol. 2, No. 2/3, 135-148.

Sinha, S. K. (1987). "Determination of Natural Frequencies of a Thick Spinning Annular Disk Using a Numerical Rayleigh-Ritz's Trial Function," *Journal of Acoustical Society of America*, Vol. 81, No. 2, 357-369.

Sisto, F. and A. T. Chang (1984). "A Finite Element for Vibration Analysis of Twisted Blades Based on Beam Theory," *AIAA Journal*, Vol. 22, No. 11, 1646-1651.

Sreenivasamurthy, S. and V. Ramamurti (1981). "A parametric Study of Vibration of Rotating Pre-twisted and Tapered Low Aspect Ratio Cantilever Plates", *Journal of Sound and Vibration*, Vol. 76, No. 3, 311-328.

Srivastav, S. (1991). "Three-Dimensional Modelling and Simulation of Buildings for Seismic Analysis," Ph.D. dissertation, Cornell University, Ithaca, New York.

Srivastav, S., and J. F. Abel (1990). "3-D Modelling of Buildings for Nonlinear Seismic Analysis," Proceedings of Eurodyn 90, European Conference on Structural Dynamics.

Trujillo, D. M. (1977). "An Unconditionally Stable Explicit Algorithm for Structural Dynamics," *International Journal for Numerical Methods in Engineering*, Vol. 11, 1579-1592.

Ugural, A. C., and S. K. Fenster (1981). *Advanced Strength and Applied Elasticity*, Elsevier, New York.

Underwood, P. (1983). "Dynamic Relaxation," in T. Belytschko and T. J. R. Hughes, Eds., *Computational Methods for Transient Analysis*, Vol. I, Mechanics and Mathematical Methods, A Series of Handbooks, First Series: Computational Methods in Mechanics, Elsevier Science Publishers B. V., Amsterdam, 245-265.

Venkatakrishnan, V., H. D. Simon, and T. J. Barth (1991). "A MIMD Implementation of a Parallel Euler Solver for Unstructured Grids," Report RNR-91-024, NAS Systems Division, Applied Research Branch, NASA Ames Research Center, Mail Stop T045-1, Moffett Field, CA 94035.

Wawrzynek, P. A. (1987). "FRANSYS: Initial Graphics Specification," unpublished internal document, Program of Computer Graphics, Cornell University, Ithaca, New York.

Wawrzynek, P. A. (1991). "Discrete Modeling of Crack Propagation: Theoretical Aspects and Implementation Issues in Two and Three Dimensions," Ph.D. dissertation, Cornell University, Ithaca, New York.

Wawrzynek, P. A., L. F. Martha, and A. R. Ingraffea (1988). "A Computational Environment for the Simulation of Fracture Processes in Three Dimensions," in A. J. Rosakis et al., Eds., *Analytical, Numerical, and Experimental Aspects of Three Dimensional Fracture Processes*, Vol. 91, 321-327.

Weiler, K. (1986). "Topological Structures for Geometric Modelling," Ph.D. dissertation, Rensselaer Polytechnic Institute, Troy, New York, 1986.

Weiler, K. (1988). "The Radial-edge Structure: A Topological Representation for Non-manifold Geometric Boundary Representations," *Geometric Modelling for CAD Applications*, 3-36.

White, C., and J. J. Lee (1993). "Implementation of a Model Creator for Complex 3-D Framed Structures in BASYS," unpublished internal document, School of Civil and Environmental Engineering, Cornell University, Ithaca, New York.

White, D. W. (1988). "Analysis of Monotonic and Cyclic Stability of Steel Frame Subassemblages," Ph.D. dissertation, Cornell University, Ithaca, New York.

White, D. W., and J. F. Abel (1989). "Testing of Shell Finite Element Accuracy and Robustness," *Finite Element in Analysis and Design*, Vol. 6, 129-151.

White, D. W., and J. F. Abel (1990). "Accurate and Efficient Nonlinear Formulation of a Nine-node Shell Element with Spurious Mode Control," *Computers and Structures*, Vol. 35, No. 6, 621-641.

Wilkinson, J. H., and C. Reinsch (1971). *Handbook for Automatic Computation: Vol. II Linear Algebra*, 1st Ed., Springer-Verlag, Berlin.

Yokoyama, T. (1988). "Free Vibration Characteristics of Rotating Timoshenko Beams," *International Journal of Mechanical Sciences*, Vol. 30, No. 10, 743-755.

Zhang, W., and E. M. Lui (1991). "A Parallel Frontal Solver on the Alliant FX/80," *Computers and Structures*, Vol. 38, No. 2, 203-215.

Zienkiewicz, O. C., and R. L. Taylor (1989). *The Finite Element Method*, 2 vols., 4th Ed., McGraw-Hill, New York.