NASA Contractor Report 185166

Parallel Eigenanalysis of Finite Element Models in a Completely Connected Architecture

F.A. Akl and M.R. Morel
Ohio University
Athens, Ohio

November 1989

Prepared for
Lewis Research Center
Under Grant NAG3-762

National Aeronautics and
Space Administration

Contents

1 Introduction **1**

1.1 Overview 1

1.2 Present State of Knowledge 2

1.3 Parallel Computers 3

1.4 Objective 4

2 Overview of Parallel Algorithm **5**

2.1 Terminology: 5

2.2 Load Balancing 6

2.3 Overview of Multitasking: 6

2.3.1 Program Multi 10

3 Parallel Solution **13**

3.1 Parallel Architecture 13

3.2 Concurrent Processing 14

3.2.1 Multifront Method 14

3.2.2 The Modified Subspace Method 19
3.3 Outline of *p–feda*: 20
3.3.1 Dynamic Array Management 20
3.3.2 *p–feda* 24

4 Numerical Experiments **32**
4.1 Purpose 32
4.2 Description of Test Problems 33
4.2.1 Two-Dimensional Beam 33
4.2.2 Space Truss 34
4.2.3 Plane Stress Example 39
4.2.4 Isoparametric Plate 40

5 Performance of the Parallel Algorithm **42**
5.1 Preview 42
5.2 Background to Testing 43
5.3 Evaluation of Varying Domains 49
5.4 Examination of Subroutines 55
5.5 Impact of Increasing Elements 57
5.6 Subspace Dimension 59
5.7 Size of Global Front 62

6 Conclusions and Recommendations **65**

Bibliography 69

A The Modified Subspace Method 74

A.1 The Eigenproblem in Structural Dynamics 75

A.2 Description of the Sequential Algorithm 75

A.3 Behavior of the Subspace Method 78

A.4 Convergence of The Modified Subspace Method 78

B Variable List 82

C Management of Files 88

D Error Messages 89

E Data Input for *p–feda* 93

F Output of *p–feda* 98

Nomenclature

A	area
$[B]$	right-hand side = [M] [V]
d	domain
E	Young's modulus of elasticity
I	moment of inertia
$[K]$	stiffness matrix of size N.N
$[K]^e$	element stiffness matrix
$[K]^*$	subspace stiffness matrix of size q.q
L	length
$[M]$	mass matrix of size N.N
$[M]^e$	element mass matrix
$[M]^*$	subspace mass matrix of size q.q
m	number of domains
n	number of domains, processors or tasks
N	total number of degrees of freedom of system
q	number of eigenpairs or size of subspace $\leq$ N
$[Q]$	eigenvectors of the auxiliary eigenproblem
t	thickness
$[V]$	eigenvectors at the $\ell - th$ iteration
$[V]^e$	element eigenvector at the $\ell - th$ iteration
$[\Omega]$	eigenvalues of required subspace
$[\Phi]$	eigenvectors of required subspace
λ_i	$i - th$ eigenvalue
$\{\phi\}_i$	$i - th$ eigenvector
ρ	mass density
υ	Poisson's ratio

Chapter 1

Introduction

1.1 Overview

Research work in computational structural mechanics is generally concerned with innovative techniques in numerical methods and software algorithms with the aim of achieving more efficient and accurate solutions. During the past decade, the architecture of computer hardware began to emerge as an important factor in computational structural mechanics. Parallel processing architecture represents a major advance in computer hardware when compared with the previous generation of single instruction single data (SISD) sequential computers. Issues dealing with high level language programming, compilers, communication links and numerical algorithms play a major role in the performance of parallel processors.

Since the development of the finite element method, several approaches of subdividing relatively large scale structures into a number of substructures have been used to overcome the limitations of the computer technology of the time. By all measures, the finite element method together with increasingly resourceful digital computers led to a revolutionary progress in many engineering and scientific disciplines. Innovative techniques in dealing with sparse band matrices allowed the solution of relatively larger order finite element systems. Creative data management approaches led to the frontal

method [24,29].

Vibration analysis of finite element models is typically time consuming and memory demanding. This is particularly true in data intensive applications generally encountered in aerospace applications. Sequential computers are rapidly achieving the physical limit of their processing power [32,37,38]. This study offers a general purpose eigenproblem solver for finite element analysis in an emerging parallel computer architecture. By utilizing this new computer architecture, it is expected that the multifrontal and the modified subspace methods will enhance the computational capabilities available to engineers. It is also hoped that this study will offer researchers in the area of parallel computational methods in engineering mechanics a thorough understanding of a new and potentially optimum method for the solution of large finite element eigenproblems on parallel computers. The new method is expected to be useful for future research work in nonlinear and structural stability problems.

1.2 Present State of Knowledge

Parallel algorithms for the solution of the static analysis problem $[K]\{V\} = \{B\}$ were implemented on multiple instruction multiple data (MIMD) computers using Jacobi iteration, successive over-relaxation (SOR) and conjugate gradient methods [19,41]. The Jacobi iterative method exhibited no guarantee for convergence [6,12,41]. However the conjugate gradient and SOR methods are reported to give a speedup of 2.8 and processor efficiency of 71% in the solution of a plane stress problem on four processors. In general, these three approaches are based on assigning one or more nodes to each processor. Nodal topology is mapped onto the communications links between processors. Direct solution techniques are also documented [21,34,37]. Salama *et al.* [34] concluded that among the direct solution methods considered, LR–Gauss appears to be the best suited for applications on a hypothetical limited processor efficient machine which closely resembles realistic parallel computers.

Research work on the solution of materially nonlinear structural stabil-

ity of imperfect columns [20] indicates that the computational efficiency decreases as the number of processors increase, suggesting an optimum number of processors.

Application of conventional substructuring techniques is recommended in Reference [40] for the solution of nonlinear large scale finite element problems. In this work, it is noted that the suggested substructuring algorithm may result in a very large dense stiffness matrix along the boundaries. As a result a tradeoff exists between the number of substructures and the amount of additional computation introduced to solve the resulting large and dense matrix along the boundaries. A speedup factor of an order equal to the number of substructures is reported in simulated parallel solution of nonlinear bending of pinched cylinder on a VAX 11/785.

Recently, structural vibration analysis in parallel processing environment has been studied using the inverse iteration method [15,39] and the Lanczos method [14]. These studies dealt with assigning each processor the task of solving for a specified bandwidth of the eigenvalue spectrum. This is accomplished by imposing a different shift region for each processor and solving the resulting eigenproblem.

A new multifrontal/subspace method of parallel processing of large eigenproblems is described in Reference [1]. This new approach utilizes the architecture of parallel computers in the solution of large eigenproblems generally encountered in aerospace applications.

1.3 Parallel Computers

Parallel processing is defined as:

> *an efficient form of computation which emphasizes the exploitation of concurrent events in the computing process. Concurrency implies parallelism, simultaneity, pipelining. It is in contrast to sequential processing* [22].

Parallel computers are classified in a variety of ways. Hwang and Briggs [22] divide parallel computers into two broad categories: synchronous and asynchronous. Synchronous parallel computers include pipelined machines (SISD) in which temporal parallelism is utilized, e.g. Cray 1 and Cyber 205, and array processors (SIMD) in which spatial parallelism is used, e.g. Illiac IV. Asynchronous parallel computers are multiprocessor machines (MIMD) in which either memory is shared among tightly coupled processors with high degree of interaction, e.g. Cray X-MP, or distributed loosely coupled processors, e.g. transputers. Several topologies of parallel computer networks, such as star, ring, pipeline and trees, can be implemented at the hardware and/or software level [22,31,37].

1.4 Objective

The objective of this study is to investigate the performance of the multifrontal/subspace method [1] in solving the generalized eigenproblem:

$$[K][\Phi] = [M][\Phi][\Lambda] \tag{1.1}$$

where: [K] is positive definite square matrix of order N
[M] is positive semi-definite square matrix of order N
[Φ] is a rectangular matrix of eigenvectors of order $N.q$, where $q \leq N$.
[Λ] is a diagonal matrix of the subdominant q eigenvalues such that $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_q$

Numerical experimentation using the CRAY X-MP/24 computer at NASA Lewis Research Center is conducted on typical problems to investigate the behavior of the multifrontal/subspace method and compared it to similar sequential solutions [3,5].

Chapter 2

Overview of Parallel Algorithm

Before we start offering a detailed description of the algorithm implemented in the multifrontal/subspace method, an overview of the overall organization of the parallel architecture will be first presented in this chapter.

2.1 Terminology:

Definitions are given for the following terms to clarify their meaning:

- Domain – a section of a subdivided finite element model considered as an independent structure except for a common boundary that connects it to the remainder of the finite element model.
- Speedup – the ratio between the time needed for a sequential algorithm to solve the problem divided by the time taken to execute the same problem using a parallel algorithm.
- Task – a computational model consisting of the code and data of the program, whose instructions must be processed in a sequential order. A separate task will be assigned to each domain.

2.2 Load Balancing

The three most time consuming processes in the solution of large eigenproblems are:

1. the creation of element stiffness and mass matrices.
2. the solution of linear simultaneous equations.
3. the extraction of eigenpairs.

Depending on the size of the problem and the number of eigenpairs required, one of the preceding processes will consume the bulk of the computational activities. Parallelizing of the first process is quite a straight forward and obvious procedure. However parallelizing the second and third processes has been the subject of a number of investigations as previously mentioned in Section 1.2. The multifrontal solution and the modified subspace method investigated in this report offer an effective parallel algorithm.

A certain amount of overhead should be expected to degrade the performance of parallel solution algorithms. This overhead is system dependent. In Chapter 5 it will be shown that the cumulative effect of overhead due to the implementation of the multitasking environment is neglegible.

On the other hand bottlenecks due to single threaded I/O operations on algorithms which are I/O intensive can lead to an appreciable performance degradation.

2.3 Overview of Multitasking:

Multitasking, defined as the structuring of a program into two or more tasks, which can execute concurrently, is Cray Research Inc.'s implementation of parallel processing. There are two methods of multitasking available. The first is macrotasking, best suited for programs with larger long running tasks

Table 2.1: Cray Multitasking Utilities

Tasks	Events	Locks	Barriers
TSKSTART	EVASGN	LOCKASGN	BARASGN
TSKWAIT	EVWAIT	LOCKON	BARSYN
TSKTUNE	EVPOST	LOCKOFF	BARREL
TSKVALUE	EVCLEAR	LOCKREL	
TSKLIST	EVREL		

(coarse granularity). The second method, refered to as microtasking, is beneficial for programs with shorter running tasks. All references, examples and developments of multitasking herein shall refer to the method of macrotasking. The programmer must explicitly code his/her FORTRAN subroutines so they can run in parallel. Multitasking subroutines can be used to decrease execution time of a complete program; but a parallel job not efficiently multitasked could take more time than a job that is sequential, due to unbalanced concurrent tasks.

The Cray X-MP computer is a tightly coupled multiple instruction multiple data (MIMD) machine which can execute different instructions and operate on different data, i.e. possesses m independent processors each having its own control unit. Memory on the Cray X-MP multiprocessor system can be accessed independently or in parallel during execution. The system has low overhead of task initiation for multitasking and has proven to be very efficient [22].

Table 2.1 shows the names of most of the multitasking library routines available on the Cray computer system [18]. These library routines provide the basic tools necessary for the implementation of the multifrontal solution and the modified subspace iteration in parallel architecture at the software level. In order to provide the reader with a grasp of the utility of the multitasking library routines, let us consider a finite element problem broken down into m domains. Each domain will be assigned a task to be initiated by a

TSKSTART subroutine. When calling the TSKSTART subroutine, a taskarray (task control array) and a name (entry point at which task execution begins) must be passed in the parameter list. An optional list of arguments can also be passed. In our case, the specific domain will be passed along with the taskarray and name. Therefore, the following call statement and do loop are needed to produce a multitasking environment with each domain assigned a separate central processing unit.

```
      DO 10 I = 1, m
         CALL TSKSTART (PROCESS(1,I), BEGIN, DOMAIN(I))
10    CONTINUE
```

Not all domains will be completed at the same time. For each TSKSTART called, a TSKWAIT must be called so that all subprograms end at the same time. The taskarray is the only parameter needed in the call list. The TSKWAIT subroutine is called through a do-loop similar to that used for TSKSTART subroutine.

```
      DO 20 I = 1, m
         CALL TSKWAIT (PROCESS(1,I))
20    CONTINUE
```

The TSKSTART and TSKWAIT subroutines create the parallel environment needed to solve the finite element problem using multitasking. Each domain will have its own task to be performed by a separate CPU. This will enable the parallel finite element algorithm to execute faster than a sequential finite element algorithm.

Communication between tasks/domains is needed to solve the finite element problem. As the tasks are executing certain variable values must be transmitted between tasks/domains. To guarantee that the values are computed in one task before they are used in another, the correspondence must take place at a synchronization point, defined as a point in time at which a task has received the go-head to proceed with its processing. Therefore, one task computes the value before the synchronization point, and the other tasks reference the value only after the synchronization point.

The facility that allows signaling between tasks is called an event, which has two states: cleared and posted. When an event is posted it has reached the synchronization point and the variable can be used in other tasks. If the event is cleared, no waiting is needed because the variable has already been posted and cleared for all tasks to continue. The event is identified by an integer variable passed through the subroutine EVASGN. An event variable cannot be used unless this subroutine is called before any other event subroutines. Therefore, EVASGN passes an integer variable used as an event and an optional value if needed. The following example initiates m events:

```
      DO 30 I = 1, m
         CALL EVASGN(EVENT(I))
30    CONTINUE
```

Three other subroutines will be called along with the EVASGN subroutine: EVWAIT, EVPOST and EVCLEAR. Each of these subroutines is needed to complete the process of communication between tasks when a variable is needed by more than one task working in parallel. The three subroutines also must pass the same integer variable as the EVASGN subroutine to link all subroutines to the same event.

The EVWAIT subroutine waits until the specified event is posted, but the task resumes execution without waiting if the event is already posted. Subroutine EVPOST returns control to the calling task after the subroutine posts the event. By the event being posted, all other tasks waiting on that event may resume execution. In addition, EVCLEAR clears an event and returns control to the calling task, but if the variable is already cleared then execution continues.

The following example program MULTI will help show the use of these multitasking subroutines. It is assumed that the program will run on a MIMD computer possessing a hypothetical 20 physical processors. For clarification, a horizontal row of dots in the example program takes the place of executable FORTRAN statements that are unimportant in the presentation of the multitasking technique at this time.

2.3.1 Program Multi:

```
c+++ This program shows how multitasking can be achieved
c+++ in generating a stiffness matrix for 20 elements concurrently.
c+++ For each element a task is assigned, a global
c+++ task will receive all element stiffness matrices and assemble
c+++ them into a global matrix. The calculation of the
c+++ element stiffness matrices should be 20 times as fast as
c+++ a sequential algorithm performing the same computations.
c+++ domain = number of the task/element.
c
      EXTERNAL BEGIN
      INTEGER EVENT1(20), PROCESS(1,20), DOMAIN(20)
      COMMON/EVENTS/EVENT1
c
c+++ data declaration
c
      DO 5 I = 1,20
         PROCESS(1,I) = 3
         DOMAIN(I) = I
    5 CONTINUE
c
c+++ event assignments
c
      DO 10 I = 1,20
         CALL EVASGN(EVENT1(I))
   10 CONTINUE
c
c+++ start domain tasks
c
      DO 20 I = 1,20
         CALL TSKSTART(PROCESS(1,I), BEGIN, DOMAIN(I))
   20 CONTINUE
c
c+++ start global task
c
```

```
        CALL ASSEMBLE
c
c+++ task completion
c
        DO 30 I = 1,20
            CALL TSKWAIT(PROCESS(1,I))
   30   CONTINUE
c
c+++ clear all events
c
        DO 40 I = 1,20
            CALL EVCLEAR(EVENT1(I))
   40   CONTINUE
c
        STOP
        END
```

```
        SUBROUTINE ASSEMBLE
        INTEGER EVENT1(20)
        COMMON/EVENTS/EVENT1
c
c+++ wait and clear all events.
c
        DO 10 I = 1,20
            CALL EVWAIT(EVENT1(I))
            CALL EVCLEAR(EVENT1(I))
   10   CONTINUE
c
c+++ read all of the element stiffness matrices from the tape
c+++ to which it was written in subroutine begin.
c
                .....
c
```

```
c+++ assemble the elements into a global stiffness matrix.
c
                .....
c
        RETURN
        END
```

```
        SUBROUTINE BEGIN (DOMAIN)
        INTEGER EVENT1(20)
        COMMON/EVENTS/EVENT1
        INTEGER DOMAIN
c
c+++ read the necessary data.
c
                .....
c
c+++ compute the stiffness matrix for the element.
c
                .....
c
c+++ write the element stiffness matrix to a tape, thereby
c+++ allowing the subroutine assemble to access the information.
c
                .....
c
c+++ post the events as they are finished
c
        CALL EVPOST(EVENT1(DOMAIN))
c
        RETURN
        END
```

Chapter 3

Parallel Solution

The parallel algorithm developed in this study is robust. First, it is not problem dependent; and second its architecture is implemented at the software level and therefore it can be altered to investigate alternative networks.

It must also be emphasized that the speedup realized by this algorithm is exclusively due to the parallelization of the solution method at the macro level, i.e. by exploiting the coarse granularity of the finite element model. Additional speedup can certainly be achieved through the implementation of the microtasking features of the Cray computer system and through automatic vectorization. This approach allows us to isolate the impact of the parallel architecture investigated in this study on the performance of the parallel algorithm.

3.1 Parallel Architecture

The parallel algorithm used in this study is based on a completely connected parallel architecture (Figure 3.1) in which each processor is allowed to communicate with all other processors. A finite element model is divided into m domains each of which is assumed to process n elements (Figure 3.2). Each domain is then assigned to a processor and the macrotasking library

routines [18] are used in mapping each domain to a user task.

3.2 Concurrent Processing

3.2.1 Multifront Method

Concurrent analysis is realized by using each processor to create the stiffness and mass matrices of the elements located within its assigned domain, and by performing assembly/forward elimination and back-substitution for each domain.

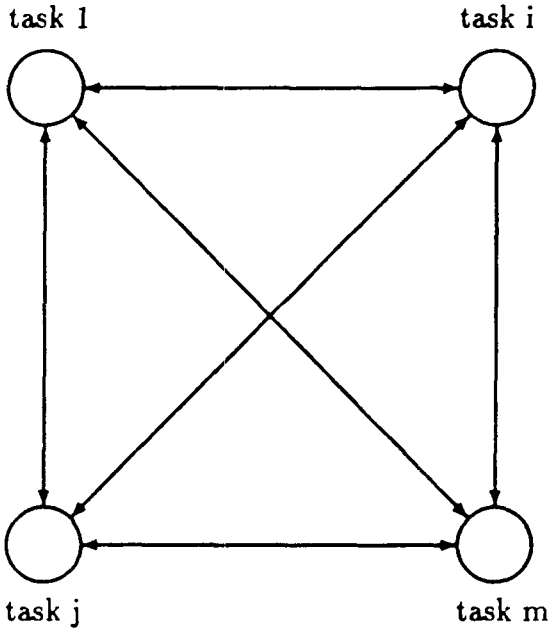

Figure 3.1: Network of Completely Connected m-Concurrent Tasks

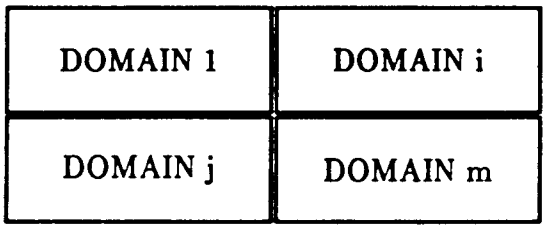

Figure 3.2: Finite Element Model Subdivided into m Domains

Figure 3.3 shows the logical structure of the parallel algorithm. Each processor creates the stiffness and mass matrices, $[K]^e$ and $[M]^e$, of the elements located within its assigned domain. Random numbers are used to generate an $N.q$ starting eigenvectors $[V]_1^e$ and to calculate the corresponding right-hand-sides, $[B]_1^e$. Each processor then assembles the element matrices, $[K]^e$ and $[B]_1^e$, and eliminate the equations corresponding to the degrees-of-freedom not located along the global fronts (boundaries) using the frontal method [24]:

$$[K]^i[V]^{*i} = [B]^i \tag{3.1}$$

where: $[K]^i, [V]^{*i}$ and $[B]^i$ are the stiffness matrix, approximate eigenvectors and right–hand sides, respectively, for the i^{th} domain just after the assembly of matrices and before the elimination of degrees–of–freedom.

The frontal method solves a set simultaneous linear equations in a unique way based on the Gaussian elimination and back-substitution algorithm [29]. Expanding Equation 3.1, we get:

$$\begin{bmatrix} k_{11} & \cdots & k_{1s} & \cdots & k_{1n} \\ \vdots & & \vdots & & \vdots \\ k_{s1} & k_{sj} & k_{ss} & & k_{sn} \\ \vdots & & \vdots & & \vdots \\ & k_{ij} & k_{is} & & \\ k_{n1} & \cdots & & \cdots & k_{nn} \end{bmatrix} \begin{Bmatrix} v_1 \\ \vdots \\ v_s \\ \vdots \\ v_i \\ v_n \end{Bmatrix} = \begin{Bmatrix} b_1 \\ \vdots \\ b_s \\ \vdots \\ b_i \\ b_n \end{Bmatrix} \tag{3.2}$$

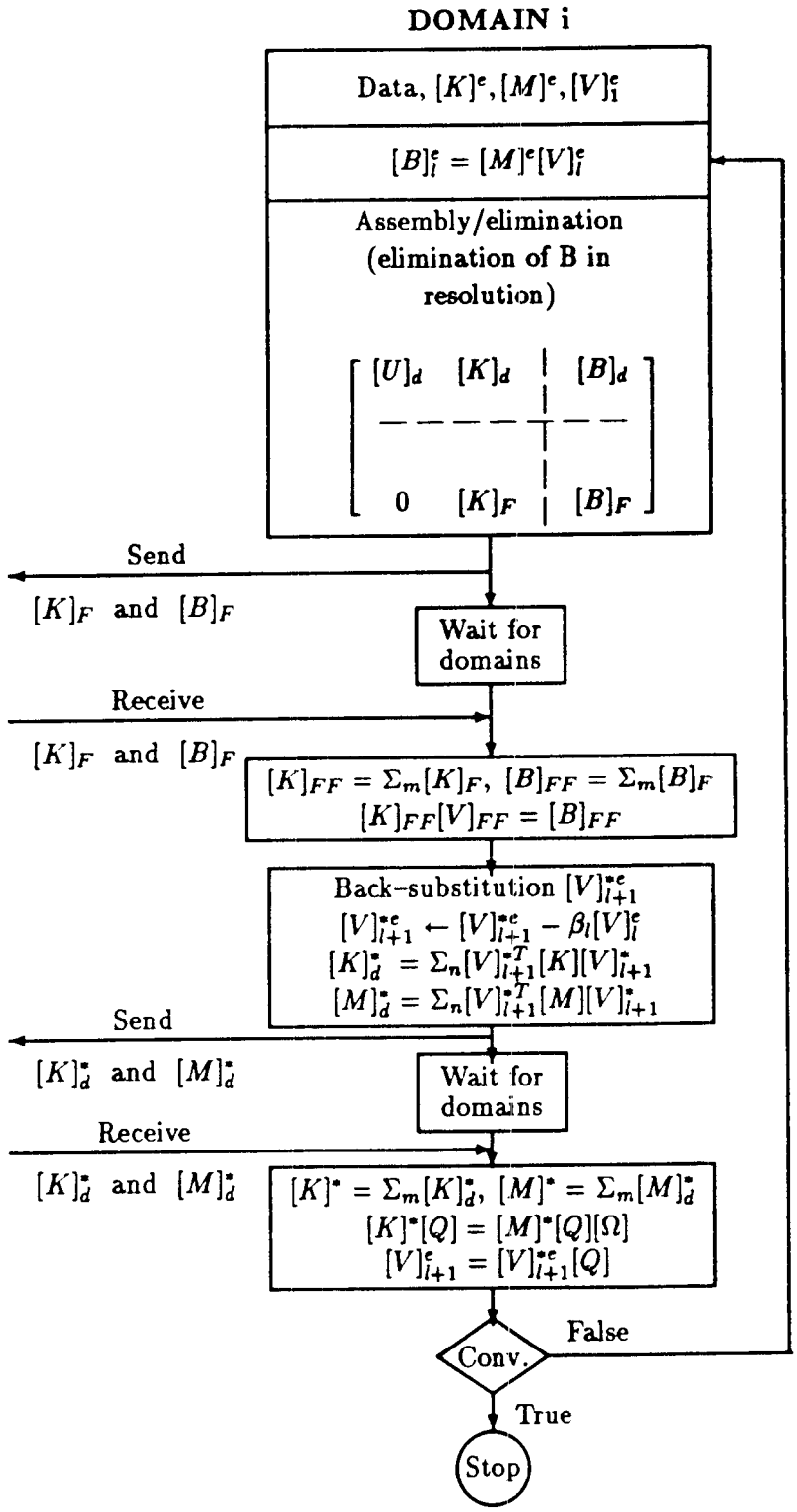

Figure 3.3: Parallel Algorithm for the i^{th} Domain

To eliminate variable v_s, the Gaussian method gives:

$$k_{ij} \leftarrow k_{ij} - \Sigma\left\{\frac{k_{is}k_{sj}}{k_{ss}}\right\} \tag{3.3}$$

$$b_i \leftarrow b_i - \Sigma\left\{\frac{k_{is}v_s}{k_{ss}}\right\} \tag{3.4}$$

Irons [24] observed that if the elements in the curly brackets shown in Equations 3.3 and 3.4 are fully assembled, the second term in the above two equations can be immediately calculated. The contribution of elements to k_{ij} and v_i can then be assembled regardless of the order in which these contributions are made. The main idea of the frontal technique is to assemble the equations and eliminate the variables at the same time. As soon as all contributions to a node are made and assembled from all relevant elements, each degree–of–freedom for this node is eliminated. As a result, the global stiffness matrix is never formed.

The wave front, i.e. the active nodes on the front, divides the domain into two substructures with three sections of elements (Figure 3.4). The first section includes the elements that have already been processed, the second section is the active elements on the front and the third is the elements that have yet to be processed. The front begins at one end of the domain and advances, engulfing one element at a time, eliminating the nodes on the element that are fully assembled until it has swept over the whole domain. After all nodes within the domain are eliminated the front reaches the domain's boundary (global front).

The frontal solution possesses certain advantages over other direct techniques and has proven to be a very effective and powerful means for solving the positive definite symmetric equations arising in standard finite element analysis. The band matrix methods are the chief competitor of the frontal solution. A comparison between the two shows that for small problems the frontal and band routines are about the same, because of the extra coding required by the frontal routine. However, with larger analyses, the frontal routine is superior in terms of speed and core requirements [24,29]. In order to optimize the solution, the band matrix methods require optimum numbering of nodes, while element numbering is not important. On the other

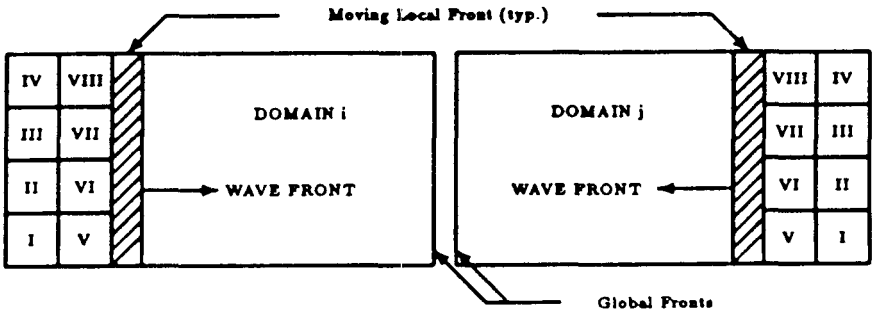

Figure 3.4: Finite Element Model Divided into m Domains

hand, the frontal solution requires optimum numbering of the elements, while node numbering is immaterial with respect to solution optimization. Optimum numbering of elements will reduce the largest frontwidth, defined as the maximum degrees of freedom on the wave front at any point in time as the wave front sweeps across the structure.

There are additional advantages in using the multifrontal solution method in parallel processing [1]. First, there is no need to renumber the nodes within each domain to minimize the bandwidth of the submatrices of the domain, because the bandwidth in the frontal solution depends on the numbering of elements. In addition, the element numbering scheme for both sequential and parallel may be unchanged, thereby forgoing preprocessing of the finite element for parallel execution. Second, load balancing is dependent on the frontwidth and the number of elements in each domain. Load balancing is therefore relatively easier to achieve using the multifrontal solution method.

Although Equation 3.1 is never formed in the frontal solution, it is given here to illustrate the algorithm in a more concise manner. At the conclusion of the assembly and elimination steps, two equations are obtained for the i^{th} domain in which the subscript (F) refers to the degrees–of–freedom located along the global fronts and the subscript (d) refers to all other degrees–of–freedom in a domain (Figure 3.2):

$$[U]_d[V]_d^* + [K]_d[V]_d^* = [B]_d \tag{3.5}$$

$$[K]_F[V]^*_F = [B]_F \tag{3.6}$$

A synchronization point is established at this stage in which each processor waits for all other processors to calculate and communicate $[K]_F$ and $[B]_F$ and to assemble matrices $[K]_{FF}$ and $[B]_{FF}$. The solution for the degrees-of-freedom located along the global fronts, $[V]_{FF}$ is obtained and the process of back-substitution within each domain proceeds concurrently until $[V]^*_{l+1}$ is calculated at the l^{th} iteration for each element.

3.2.2 The Modified Subspace Method

Concurrent processing continues to calculate the projection of the stiffness and matrices onto the required subspace, $[K]^*_d$ and $[M]^*_d$ of order $q.q$ for the i^{th} domain. This is the second and last synchronization point in the parallel algorithm at which the contribution from all other domains are required before proceeding to solve the auxiliary eigenproblem of the modified subspace, $[K]^*[Q] = [M]^*[Q][\Omega]$. More accurate approximation of the eigenvectors $[V]^c_l$ is obtained using:

$$[V]^c_{l+1} = [V]^*_{l+1}[Q] \tag{3.7}$$

The algorithm either terminates or continues to iterate until a test of convergence is satisfied.

The modified subspace method [2,4] is used to solve for the least dominant eigenpairs, $[\Phi]$ and $[\Omega]$, of order q, where $q \leq N$, the total number of degrees-of-freedom of the finite element model. The classical subspace method is reported to provide an efficient algorithm for the solution of large problems in sequential and parallel processing [10,39]. The rate of convergence of the modified subspace method used in this report is faster by an average of 33% compared to the classical subspace method [2]. In Appendix A, detailed presentation of the modified subspace method is given.

3.3 Outline of *p–feda*:

A broad outline of a parallel finite element analysis program hereinafter referred to as *p–feda* is given in this section showing the implementation of the parallel algorithm described in Section 3.2.

3.3.1 Dynamic Array Management

Dynamic management of arrays is used within each domain (task) to accommodate the varying demands of problem analyses in an efficient manner. Two large vectors, *VEC* and *VEC1*, are declared at the outset for each domain with a size of *LENVEC* and *LENVEC1*, respectively. Two integer vectors, *NVEC* and *NVEC1*, are equivalenced with *VEC*, and *VEC1*, respectively.

Tables 3.1 and 3.2 show the addresses calculated in *p–feda* for arrays *VEC/NVEC* and *VEC1/NVEC1*. Detailed explanation of the variable names is given In Appendix B. Appendices C and D provide a brief summary of file management and error messages used in the program, respectively.

The reader should note that the Cray computer running the FTN2 FORTRAN compiler under COS assigns integer and floating point numbers an equal number of words. However, the FT77 FORTRAN compiler running under COS assigns an integer number a length of 42 bits versus 64 bits for a floating point number. The latter word assignment is the default under the UNICOS operating system regardless of the FORTRAN compiler used. It is therefore necessary to specify in the compiler command a word length of 64 bits for integer numbers in order to achieve correct mapping between *VEC/NVEC* and *VEC1/NVEC1*.

Table 3.1: Arrays in VEC/NVEC

Address	Name	Dimension
1	ELSTIF	LCOEF=(LVAB*(LVAB+1))/2
I1	ELOAD	LVAB*NRHS
I2	ELDISP	LVAB*NRHS
I3	ELCORD	NODEL*NDIM
J1	NDF	NODEL*JPROP
J2	LTYPE	NEL
J3	LLOAD	NEL
J4	NDMAIN	NGLOBE
J5	LDMAIN	NEL
J6	LNODS	NODEL*NEL
J7	GLNODS	GNODEL*NFRONT
J8	GNDF	NPOIN
J9	GNDFRO	NGLOBE/NFRONT
K1	COORD	NPOIN*NDIM
K2	VFIX	NFIX*NDFMAX
K21	REACTN	NFIX*NDFMAX*NRHS
K3	VSTIF	NEXTIF*NDFMAX
K4	VLOAD	NLOAD*NDFMAX*NRHS
K5	SLOAD	JLOAD*NRHS
K6	VPROP	IPROP*JPROP
K7	EXCOD	NEL*NDIM

Table 3.1 Arrays in VEC/NVEC (cont'd)

Address	Name	Dimension
L1	NODFIX	NFIX
L2	KODFIX	NFIX
L3	NOSTIF	NEXTIF
L4	NODLOD	NLOAD
L5	LPROP	NEL
L6	NODFRO	LIMFRO
	NDFRO	NEL
L7	LDEST	LVAB
M1	SUSTIF	NSTIF=(LIMFRO(LIMFRO+1))/2
M2	SUBLOD	LIMFRO*NRHS
M3	SIGDIG	LIMFRO
M4	TOTLOD	LIMFRO*NRHS
M5	EQ	LIMFRO*NBUFZ
M6	EQR	NBUFZ*NRHS
M7	EQRTOT	NBUFZ*NRHS
M8	EQSIG	NBUFZ
N1	NPIVOT	NBUFZ
N2	NAME	NBUFZ
N3	MDEQ	NBUFZ
N4	STREGY	NRHS
N5	POTEGY	NRHS
N6	DIAGY	NRHS
N7	ERRGY	NRHS

Table 3.2: Arrays in VEC1/NVEC1

Address	Name	Dimension
1	ELMASS	LCOEF = (LVAB(LVAB+1))/2
IG1	EIGEN1	LVAB*NEIGEN
IG2	OLGNVL	NEIGEN
IG3	SKSTR	NEIGEN*NEIGEN
IG4	SMSTR	NEIGEN*NEIGEN
IG5	CJAC	NEIGEN*NEIGEN*3
IG6	ELMST1	LVAB*NEIGEN
IG7	ELKST1	LVAB*NEIGEN
IG8	ELMSTR	NEIGEN*NEIGEN
IG9	ELMKSR	NEIGEN*NEIGEN
IG10	EIGSHP	NPOIN1*NEIGEN
IG11	FULMAS	LVAB*LVAB
IG12	FULSTF	LVAB*LVAB
IG13	NDIGEN	NPOIN1
IG14	EIGEN2	LVAB*NEIGEN

3.3.2 *p–feda*

MAIN PROGRAM

```
c
c+++ a parallel frontal solution of finite element systems
c+++ using a modified subspace approach to extract the eigenpairs
c


      INTEGER PROCESS(3,10), DOMAIN(10)
      INTEGER EVENT(10,10)
c
c+++ put event in common block so that all domains can access it.
c

      COMMON /EVENTS/ EVENT
c
c+++ data declaration
c

      DO 10 I = 1,NFRONT
         PROCESS(1,I) = 3
         DOMAIN(I) = I
   10 CONTINUE
c
c+++ event assignments, identifies the integer variables that
c+++ the program intends to use as an event.
c

      DO 20 I = 1,NFRONT
      DO 20 J = 1,NFRONT
         CALL EVASGN(EVENT(I,J))
   20 CONTINUE
c
c+++ start all domain tasks except domain 1.
c
      DO 30 I = 2,NFRONT
         CALL TSKSTART(PROCESS(1,I), DOMFRONT, DOMAIN(I))
```

```
     30 CONTINUE
c
c+++ start domain 1, which is called separately to limit the
c+++ input data to just one domain.
c
        CALL DOMFRONT(DOMAIN(1))
c
c+++ upon task completion, tell each domain that domain 1 has
c+++ been completed.
c
        DO 40 NF = 1,NFRONT
            CALL EVPOST(EVENT(1,NF))
     40 CONTINUE
c
c+++ wait for tasks to be completed that were called by tskstart.
c
        DO 50 NF = 2,NFRONT
            CALL TSKWAIT(PROCESS(1,I))
     50 CONTINUE
c
        STOP
        END
```

```
        SUBROUTINE DOMFRONT(DOMAIN)
c
        INTEGER DOMAIN
c
        IF( DOMAIN .EQ. 1 ) THEN
c
c+++ read initial data.
c+++ diagnose the initial data, if fatal or nonfatal errors occur
c+++ call doctor, and setup housekeeping for dynamic dimensioning
c+++ of vector arrays. the subroutine doctor will identify and
c+++ list the error messages found in the data.
c
```

```
        CALL DNURSE
c
c+++ read the remaining data, e.g. element types, nodal
c+++ coordinates etc., for the problem and assign them to
c+++ variables in a common block, that will be accessed by all
c+++ domains.
c
        CALL FINPUT
c
c+++ tell all domains, except domain 1, that all data has
c+++ been read in.
c
        DO 10 NF = 1,NFRONT
            CALL EVPOST(EVENT(1,NF))
    10  CONTINUE
c
        ELSE
c
c+++ wait for the data to be read into domain 1.
c
        CALL EVWAIT(EVENT(1,DOMAIN))
c
c+++ setup housekeeping for dynamic dimensioning of vector
c+++ arrays.
c
        CALL DNURSE
c
c+++ assign all data received in the common block from
c+++ domain 1 to the correct variables.
c
        CALL DINPUT
c
c+++ clear events so that they maybe used again.
c
        CALL EVCLEAR(EVENT(1,DOMAIN))
c
        ENDIF
c
c+++ check for the last appearance of each node, when it is
```

```
c+++ found make the node negative (creating the pre-front).
c+++ also, the size of the global front is determined and a
c+++ few more dimensions are calculated that are needed in
c+++ subroutine dfront.
c
        CALL DMATRON
c
c+++ creates the element stiffness and mass matrix files
c+++ for the specific element type, in addition, creates
c+++ the element load and eigenvector files.
c
        CALL ESTIFF
c
        DO 6 NRESOL = 1,NCASE
c
c+++ solves the set of linear simultaneous equations using
c+++ the frontal technique.
c
        CALL DFRONT
c
```

SUBROUTINE DFRONT

```
c
c+++ assemble the stiffness matrix and eliminate the
c+++ degress-of-freedom in their last appearance up to the local front.
c
        .....
c
c+++ after the local front is reached begin work on the
c+++ global front.
c
        .....
c
c+++ tell all of the other domains that you have completed the
c+++ assembly of the local front located within this domain.
```

```
c
        DO 100 NF = 1,NFRONT
            IF ( NF .NE. DOMAIN ) THEN
            CALL EVPOST(EVENT(DOMAIN,NF))
            ENDIF
    100 CONTINUE
c
c+++ wait for all domains to reach their local fronts.
c
        DO 200 NF = 1,NFRONT
            IF ( NF .NE. DOMAIN ) THEN
            CALL EVWAIT(EVENT(NF,DOMAIN))
            ENDIF
    200 CONTINUE
c
c+++ clear the event for future use.
c
        DO 300 NF = 1,NFRONT
            IF (NF .NE. DOMAIN ) THEN
            CALL EVCLEAR(EVENT(NF,DOMAIN))
            ENDIF
    300 CONTINUE
c
c+++ assemble the global stiffness matrix and perform a gauss
c+++ jordan elimination to solve the unknown variables
c+++ located on the global front.
c
        .....
c
c+++ begin the back-substitution to solve for all unknown
c+++ variables in the domain.
c
        .....
c
        RETURN



c
c+++ solve for the eigenpairs.
```

```
c
        CALL DCONDNS
c
```

SUBROUTINE DCONDNS

```
c
c+++ project the stiffness and mass matrices onto the current
c+++ subspace for each iteration.
c
        .....
c
c+++ tell all domains that this task has been completed.
c
        DO 100 NF = 1,NFRONT
            IF ( NF .NE. DOMAIN ) THEN
            CALL EVPOST(EVENT(DOMAIN,NF))
            ENDIF
    100 CONTINUE
c
c+++ wait for domains to reach this point.
c
        DO 200 NF = 1,NFRONT
            IF ( NF .NE. DOMAIN ) THEN
            CALL EVWAIT(EVENT(NF,DOMAIN))
            ENDIF
    200 CONTINUE
c
c+++ clear the event.
c
        DO 300 NF = 1,NFRONT
            IF ( NF .NE. DOMAIN ) THEN
            CALL EVCLEAR(EVENT(NF,DOMAIN))
            ENDIF
    300 CONTINUE
c
```

```
c+++ assemble [k]*d and [m]*d from all domains.
c
        .....
c
c+++ solve the auxiliary eigen problem for each iteration by a
c+++ pseudo-jacobi method .
c
        CALL EIGN
c
c+++ test the eigenvalues for convergence. if tolerance is
c+++ met then set nstop = 10. a better m-orthonormalized
c+++ approximation of the required eigen vectors is constructed.
c
        .....
c
        RETURN



c
        IF ( NSTOP .NE. 0 ) GO TO 70
c
c+++ after each iteration, calculate a new set of eigenvectors
c+++ that will be used in the next iteration.
c
        CALL RELOAD
c
     6  CONTINUE
c
c+++ prints the output for mode shapes at prescribed nodes
c+++ rather than the customary element-by-element output.
c
    70  CALL PLOTING
c
c+++ wait for tasks to be completed and clear events.
c
        IF ( DOMAIN .NE. 1 ) THEN
            CALL EVWAIT(EVENT(1,DOMAIN))
            CALL EVCLEAR(EVENT(1,DOMAIN))
        ENDIF
```

```
C
      RETURN
      END
```

Chapter 4

Numerical Experiments

4.1 Purpose

This chapter has a three fold purpose:

1. Validate the algorithm's accuracy by comparing the results to the following: a sequential frontal/subspace algorithm called "FEDA" [3], MSC/NASTRAN (NAsa STRuctural ANalysis program) and in some cases analytical results. Note: All MSC/NASTRAN eigenpairs were solved by the Tridiagonal (Givens) method [36].
2. To show the capabilities and wide range of problems the algorithm can handle.
3. Attract attention to the speedup attained for each finite element problem by subdividing the structure into two domains. As a consequence of only two physical CPUs on the Cray X-MP/24 computer, two domains were chosen to get the exact speedup between the sequential and parallel algorithm. The ideal or theoretical speedup for the parallel solution would be 2.0.

Therefore, in the following pages a description, diagram, input properties, speedup and a table of eigenvalues solved by the different procedures for each

structure analyzed will be presented. For all problems solved a tolerance of 10^{-7} was imposed on each eigenvalue in the subspace. Located in Appendix E is a description of the input data used for *p–feda*.

4.2 Description of Test Problems

4.2.1 Two-Dimensional Beam

The first problem solved was a beam clamped at both ends (Figure 4.1) and having 20 elements with 57 degrees–of–freedom (dof), i.e. three dof at each node; this rather simple problem was chosen to assist in the initial developmental efforts. The global front for this beam consisted of one node located at the center of the beam. As a result, there are two domains/tasks, the right half is domain one and the left half is domain two. In MSC/NASTRAN, the beam was modeled using the CBAR element [36] along with a consistent mass matrix. Speedup for the beam is 1.49; due to the simplicity and small number of elements the speedup is low. Total number of iterations for the parallel and sequential programs where 14 and 15 respectively. Table 4.1 contains the four lowest eigenvalues solved by *p–feda*, FEDA, MSC/NASTRAN and a closed-form solution.

Table 4.1: Clamped-Clamped Beam Eigenvalues

Order of Eigenvalue	Eigenvalues Predicted by			
	Parallel feda (q = 8)	Sequential feda (q = 8)	NASTRAN	Closed Form [30]
1	19.55	19.55	19.55	19.55
2	148.6	148.6	148.6	148.6
3	571.1	571.1	570.9	571.0
4	1560.8	1561.0	1559.6	1560.3

Beam with Both Ends Clamped (Speedup = 1.49)

Number of Nodes = 21	Number of Elements = 20
Input Properties:	
E = 1.0E15 kN/m^2	A = 1.0E10 m^2
I = 1.0 m^4	t = 1.0 m
ρ = 1.0 $kNsec^2/m^4$	L = 2.0 m

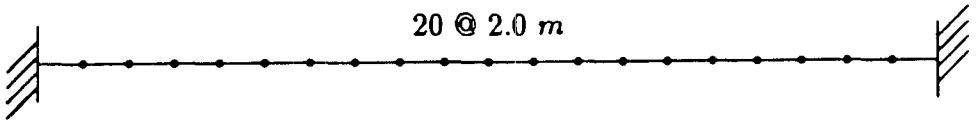

Figure 4.1: Clamped–Clamped Beam Idealization

4.2.2 Space Truss

Two structures were analyzed using the three dimensional truss element, with all members having three degrees of freedom at each node. Both space trusses were constructed on MSC/NASTRAN and use the CBAR element [36] with all rotations fixed and a lumped mass matrix. The first space truss [14] used in solving for the eigenpairs has 88 members and 26 joints with the four base nodes fixed, shown in Figure 4.2, as a result the truss contains 66 dof. There are four nodes on the global front, located atop the third tier from the bottom, and an equal number of elements in their respective domains. Property set one belongs to all members that makeup the five box subtrusses (horizontal and vertical elements are 20 ft in length) while the eight members that protrude from the sides pertain to property set two and have a horizontal length of 40 ft. For the space truss, a speedup of 1.77 was achieved with both parallel and sequential algorithms performing eight iterations until convergence was met. In addition, a comparison of the six lowest eigenvalues are located in Table 4.2.

Space Truss (Speedup = 1.77)

Number of Nodes = 26 Number of Elements = 88

Input Properties:

Property Set 1	Property Set 2
E = 3.9385E10 psi	E = 3.9385E10 psi
A = 1.0 in^2	A = 3.0 in^2
I = 1.0E02 in^4	I = 1.0E02 in^4
ρ = 1.0 $lb\ sec^2/in^4$	ρ = 1.0 $lb\ sec^2/in^4$

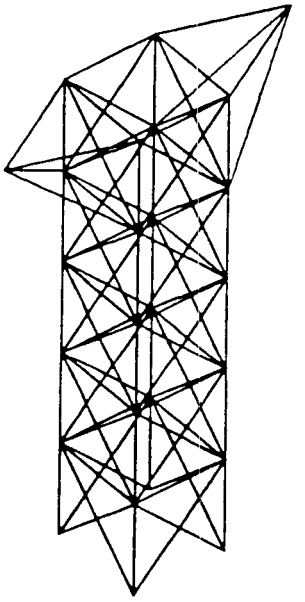

Figure 4.2: Idealization of Space Truss

Table 4.2: Space Truss Eigenvalues

Order of Eigenvalue	Eigenvalues Predicted by			
	Parallel feda (q = 12)	Sequential feda (q = 12)	NASTRAN	Lanczos Method [14]
1	2.104E04	2.104E04	2.104E04	2.104E04
2	3.075E04	3.075E04	3.076E04	3.076E04
3	3.624E04	3.624E04	3.524E04	3.625E04
4	1.819E05	1.819E05	1.819E05	1.819E05
5	3.869E05	3.869E05	3.869E05	3.869E05
6	9.168E05	9.168E05	9.168E05	9.170E05

The second space truss considered is an open helicopter tail-boom structure [7]. There are 108 truss members and 28 nodes as shown in Figure 4.3, the four left end nodes are fixed with the structure possessing 72 dof. Keeping both domains balanced an equivalent number of elements are assigned to each separate task. The structure is subdivided at its midpoint and consists of four nodes on the global front. An abnormally high speedup of 2.10 was calculated for the tail-boom structure because of the smaller number of iterations taken by the parallel program (15) compared to the sequential program (20). Figure 4.4 shows the geometry and lengths of the finite element model, furthermore, Table 4.3 contains the eigenvalues of the tail-boom.

Table 4.3: Eigenvalues for Helicopter Tail-Boom

Order of Eigenvalue	Eigenvalues Predicted by			
	Parallel feda (q = 12)	Sequential feda (q = 12)	NASTRAN	Subspace Iteration [7]
1	1.848E04	1.848E04	1.846E04	1.880E04
2	2.076E04	2.076E04	2.077E04	2.113E04
3	3.091E05	3.091E05	3.090E05	4.075E05
4	3.723E05	3.723E05	3.719E05	4.370E05
5	4.160E05	4.160E05	4.163E05	4.553E05
6	1.574E06	1.574E06	1.574E06	1.593E06

Helicopter Tail-Boom (Speedup = 2.10)

Number of Nodes = 28 Number of Elements = 108

Input Properties:
E = 1.05E08 psi A = 1.0 in^2
I = 1.0E02 in^4 ρ = 2.588E-03 $lb\ sec^2/in^4$

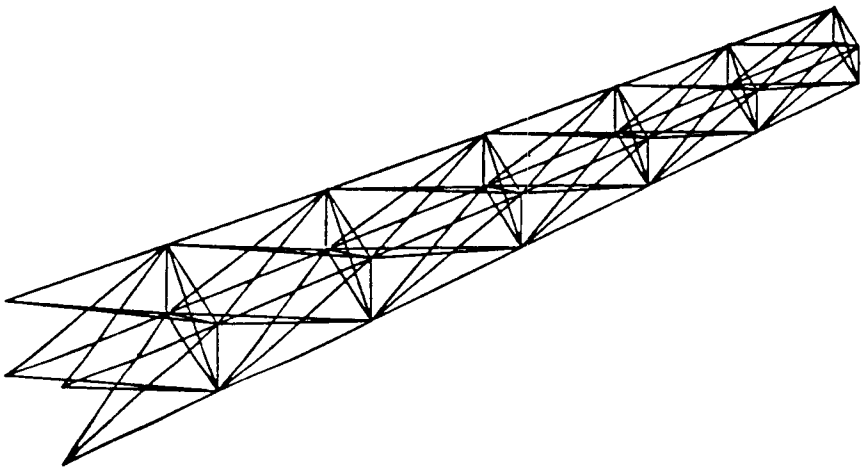

Figure 4.3: Helicoptor Tail-Boom Idealization

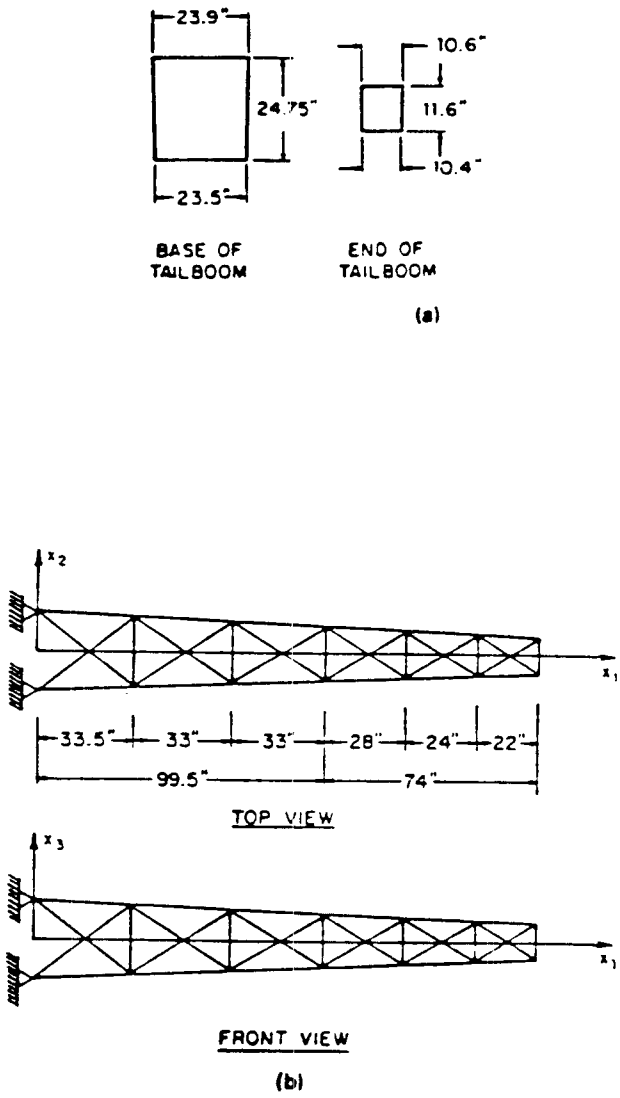

Figure 4.4: Helicopter Tail-Boom Structure: (a) Geometry of Tail-Boom (b) Finite Element Model for the Tail-Boom Structure [7]

4.2.3 Plane Stress Example

Referring to Figure 4.5, a rectangular plane stress element comprising of eight elements in the form of an inverted T is analyzed. The T-section was modeled on MSC/NASTRAN using a QUAD8 element with the rotations fixed and a coupled mass matrix. Shown in Table 4.4 are the eigenvalues determined by *p–feda*, FEDA and MSC/NASTRAN.

For the structure, each square element has eight nodes with two dof each and a length of 2.0 *in*. Moving across the bottom horizontally, all nodes are clamped which leaves 60 dof to displace. Substructures are formed by a vertical global front in the center of the structure which includes seven joints. Due to the lower number of iterations for the parallel program (16) matched against the sequential program (21), speedup equaling 2.31, once again was jutting above the theoretical speedup.

Plane Stress T-Section (Speedup = 2.31)

Number of Nodes = 39	Number of Elements = 8
Input Properties:	
E = 1.0 *psi*	$\nu = 0.4$
t = 1.0 *in*	$\rho = 1.0\ lb\ sec^2/in^4$

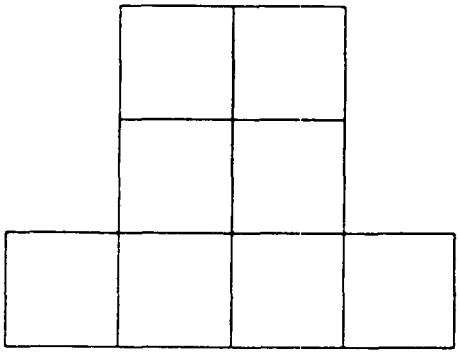

Figure 4.5: Idealization of T-Section

Table 4.4: Eigenvalues for Plane Stress T-Section

Order of Eigenvalue	Eigenvalues Predicted by		
	Parallel feda (q = 12)	Sequential feda (q = 12)	NASTRAN
1	3.806E-03	3.806E-03	3.822E-03
2	2.054E-02	2.054E-02	2.055E-02
3	2.344E-02	2.344E-02	2.341E-02
4	7.223E-02	7.223E-02	7.151E-02
5	9.195E-02	9.195E-02	9.165E-02
6	1.233E-01	1.233E-01	1.229E-01

4.2.4 Isoparametric Plate

The final problem solved is a cantilevered plate which has a hole within the structure shown in Figure 4.6. Free vibration analysis of the 116 element plate on MSC/NASTRAN modeled by the QUAD4 element and an uncoupled mass matrix was performed. The left end rotations and translations are fixed, moving in the vertical direction from top to bottom for all nodes. The nodal configuration for a square isoparametric thin plate used in analysis comprised of four corner deflections and 12 slope variables (two at each corner node and one at the midside nodes), in addition, there are 701 dof for the total structure and 12 nodes lying on the global front located at the center of the plate along a horizontal line. The length of each plate element is 2.0 *in*. Speedup of the parallel algorithm is 1.84 with the number of iterations (21) being equivalent in the parallel and sequential algorithms. Due to the slight differences in the stiffness matrices determined by *p–feda*, FEDA and MSC/NASTRAN, the eigenvalues in Table 4.5 are not equivalent but have a difference of 1-3% for each mode.

Cantilevered Plate with Hole (Speedup = 1.84)

Number of Nodes = 430 Number of Elements = 116

Input Properties:

E = 7.7E10 *psi*
ν = 0.33
t = 1.0 *in*
ρ = 2.8E03 *lb. sec*^{2}/*in*^{4}

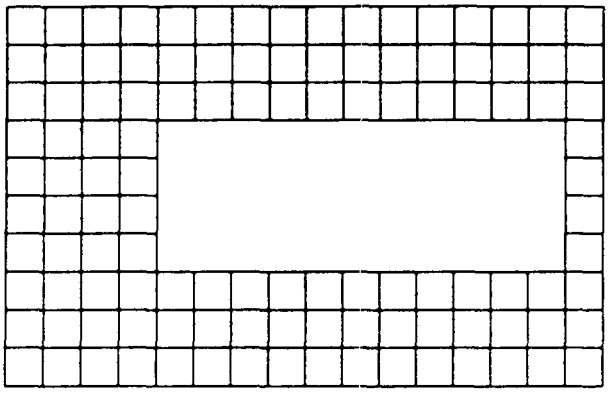

Figure 4.6: Plate with Hole Idealization

Table 4.5: Eigenvalues for a Plate with a Hole

Order of Eigenvalue	Eigenvalues Predicted by		
	Parallel feda (q = 10)	Sequential feda (q = 10)	NASTRAN
1	504.8	504.8	511.9
2	3798.0	3798.0	3815.1
3	17730.0	17730.0	17583.0
4	34510.0	34510.0	34843.0
5	1.326E06	1.326E06	1.292E06

Chapter 5

Performance of the Parallel Algorithm

5.1 Preview:

To measure the success of the parallel algorithm on the Cray X-MP/24 supercomputer, two important factors will be determined: speedup (see Section 2.1) and efficiency which measures the utilization of the parallel machine, e.g. if the processors are idle or require extra calculations introduced through parallelization of the problem the speedup and efficiency decrease [37], the subsequent equations represent the speedup and efficiency:

$$\text{SPEEDUP} = \text{SP} = \frac{T_s}{T_p} \ (\geq 1) \tag{5.1}$$

$$\text{EFFICIENCY} = \frac{\text{SP}}{\text{m}} \ (\leq 100\%) \tag{5.2}$$

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

These evaluation tools were computed for the following size plates 8, 16, 24, 32, 40 and 64 elements, shown in Figures 5.1 and 5.2. Variables affecting the assessment of the algorithm are:

1. the number of domains chosen.
2. total number of elements and degrees–of–freedom (dof).
3. formulation of the global front relative to the number of degrees–of–freedom on the local fronts.
4. direction the domain fronts move controlled by the element numbering scheme.
5. number of iterations taken to achieve the required tolerance level.
6. total number of eigenpairs (q) predicted.

As a result, a number of test runs were analyzed to examine these variables influencing the speedup and efficiency of the algorithm in a dedicated mode. This chapter presents the results obtained from a number of example problems for rectangular plate structures with all edges clamped (c). All structural plates use an isoparametric square plate element [8] of length 2.0 *in*; the model consists of four corner nodes and four mid-side nodes amounting to 16 degrees–of–freedom per element. The input properties for all plates are: Young's modulus is 1.0 *psi*, Poisson's ratio is 0.3, the mass density is 1.0 $lb\ sec^2/in^4$ and 1.0 *in* equaling the thickness. Either a tolerance level of 10^{-7} or 10^{-4} was placed upon all q eigenvalues.

5.2 Background to Testing:

Time functions were inserted into the algorithm at strategic points to define accurately the time needed to perform the calculations. The time elapsed from one station to another is wall-clock time and not the CPU time charged to the job [18]. A distinction must be made between them as a consequence of a multiple processor job having a greater CPU time than an equivalent

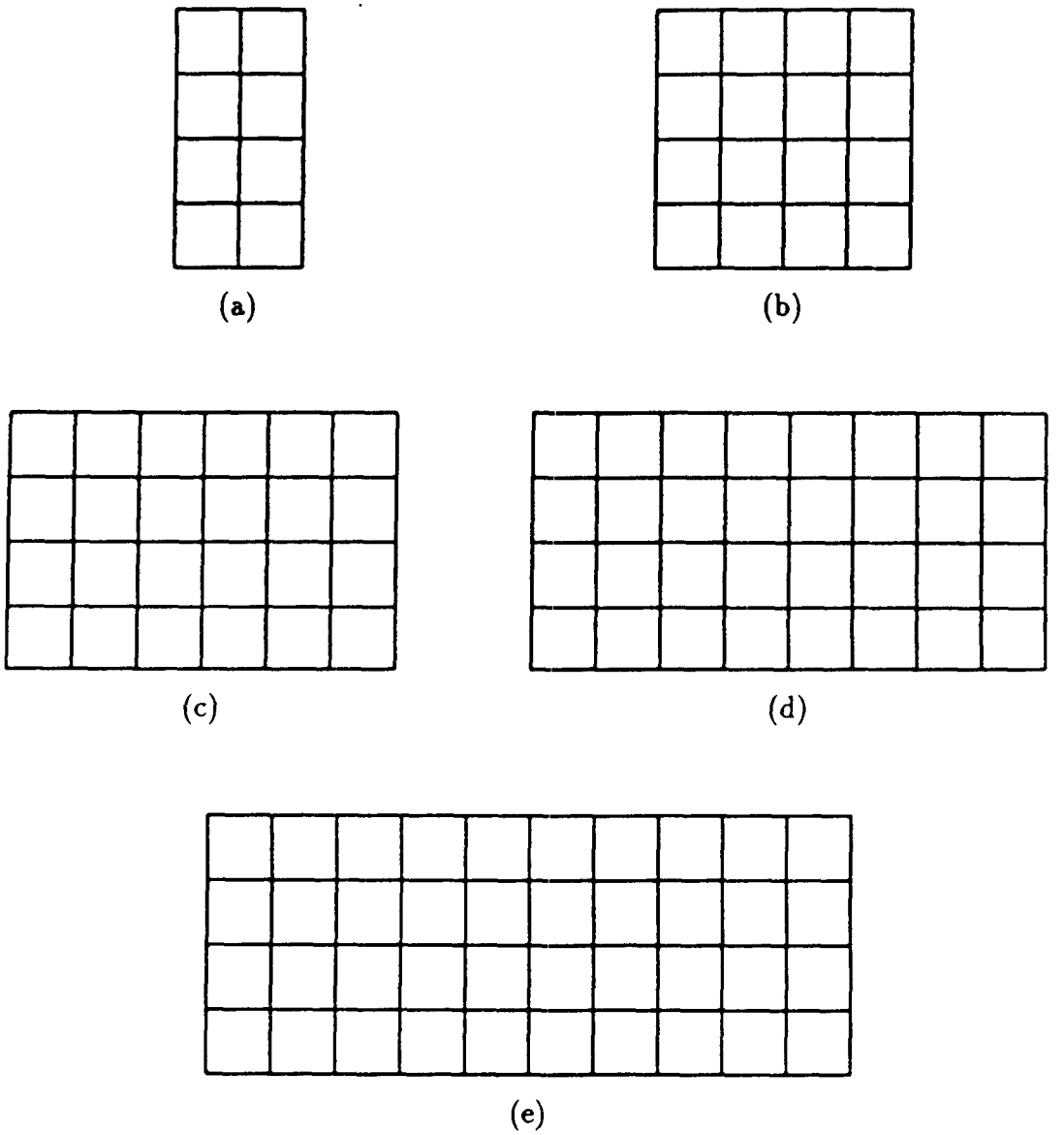

Figure 5.1: Different Size Plates Used in Analysis with all Edges Clamped: (a) 8 Element Plate (67 dof); (b) 16 Element Plate (115 dof); (c) 24 Element Plate (163 dof); (d) 32 Element Plate (211 dof); (e) 40 Element Plate (259 dof)

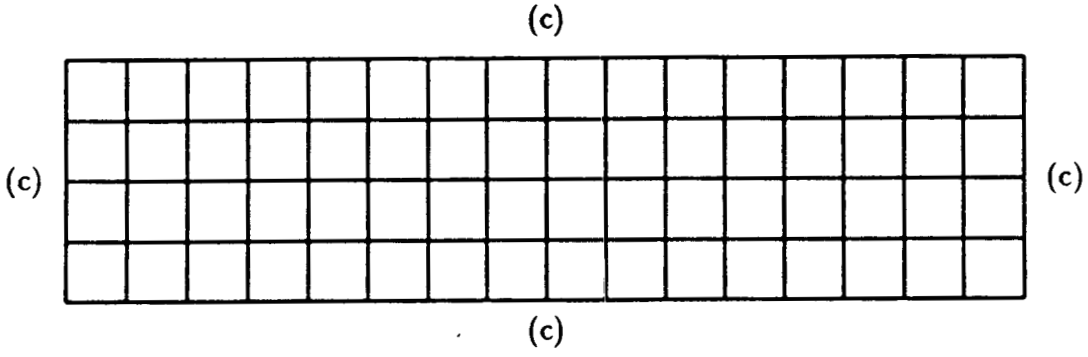

Figure 5.2: A 64 Element Plate Clamped (c) on all Edges

job on a single processor. The total CPU time for a multi-processor job will be labeled work done by the system and will not be equal to the wall-clock (execution) time, in contrast wall-clock and CPU time for a sequential job are nearly equivalent, i.e. total execution time will be the sum of the subroutine times for the slowest task in the parallel solution.

As reported earlier, the Cray X-MP/24 computer has only two physical CPUs but the program can handle up to eight logical processors, i.e. when the number of processors used in the parallel solution exceed the number of processors on the machine (physical processors), the processors are called logical processors. This must be kept in mind when assessing the speedup for domains/tasks greater than two on account of the extra waiting introduced at synchronization points; this time must be eliminated to get a more precise execution time.

Shown in Figure 5.3 is a time chart of a four logical CPU system working on a two physical CPU machine with all tasks assigned the same amount of work. Tasks 3 and 4 cannot begin execution until tasks 1 and 2 reach the first synchronization point (AT1). The two task processors working simultaneously will not finish exactly at the same time, but within 10-30 milliseconds of each other, therefore it will be assumed that the task processors are nearly equivalent in time. In a two processor system it was identified that minimal time, approximately 0.02 milliseconds, was taken to post, wait and clear an

event at the place of communication if the event had been previously posted by the other task. At AT2 task 3 and 4 reach the same synchronization point that task 1 and 2 did earlier at AT1 and all four tasks are ready to continue execution again. Following the logic of *p– feda's* main program, task one will always continue over the other tasks after all tasks reach the same synchronization point along with the last task to post the same event, this can be seen at point AT2 if task 4 posted its event after tasks 2 and 3. Therefore, when calculating the total wall-clock time, for domains greater than two, only the actual time computing will be summed and not the time accumulated by the idle domains waiting for an open processor to continue execution. The total execution time for the four tasks in Figure 5.3 will be calculated as if there are m physical processors located on the machine, shown in Figure 5.4. The following conclusions can be drawn from Figures 5.3 and 5.4:

1. Figure 5.3 represents a two physical CPU system with four logical CPUs being used. Figure 5.4 shows the interpretation of how the total execution time is resolved.
2. Time begins at AT0 and DT0 and ends at AT4 and DT2.
3. AT1, AT2 and DT1 are associated with the same synchronization point.
4. Refer to Figures 5.3 and 5.4.

$$T11 \cong T21 \cong T31 \cong T41 \cong D11 \cong D21 \cong D31 \cong D41 \tag{5.3}$$

$$T12 \cong T22 \cong T32 \cong T42 \cong D12 \cong D22 \cong D32 \cong D42 \tag{5.4}$$

$$T11 + T12 \cong D11 + D12 \tag{5.5}$$
$$T21 + T22 \cong D21 + D22 \tag{5.6}$$
$$T31 + T32 \cong D31 + D32 \tag{5.7}$$
$$T41 + T42 \cong D41 + D42 \tag{5.8}$$

Input and output (I/O) for all sample runs were kept to a minimum amount. The SSD solid-state storage device used to read and write information on tapes/disks was bypassed because when one processor gains access to the device the other processor becomes idle. When the problem of

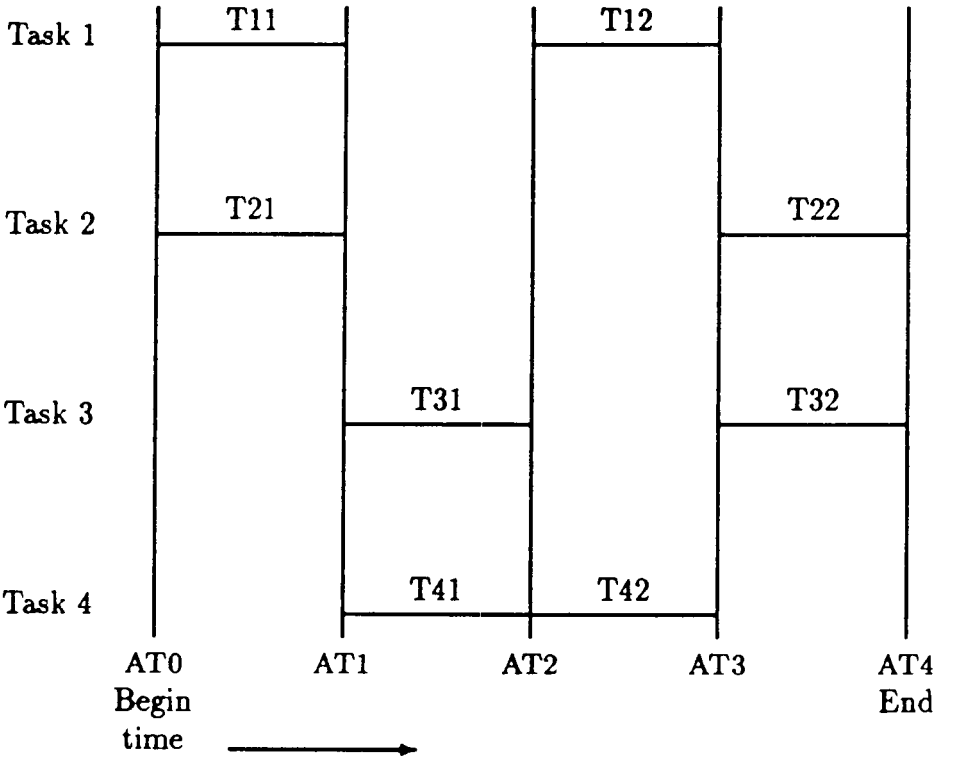

Figure 5.3: Time Chart of Four Logical Processor System Running on a Two Physical Processor Machine

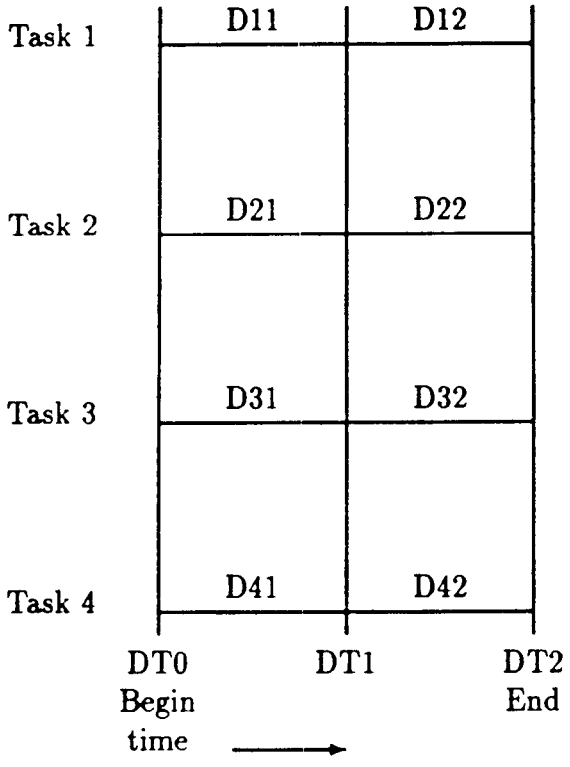

Figure 5.4: Time Chart Showing the Interpretation of a Four Logical Processor System Running on a Two Physical Processor Machine

Table 5.1: Analysis of Various Domains for a 64 Element Plate Using a Subspace (q) of 2

Number of Processors	Figure Number	Speedup (Tol=10^{-4})	Efficiency (Tol=10^{-4})	Speedup (Tol=10^{-7})	Efficiency (Tol=10^{-7})
1	5.2	1.00 (13)	100%	1.00 (16)	100%
2	5.5	1.85 (13)	93%	1.86 (16)	93%
4	5.7	3.86 (9)	96%	3.13 (14)	78%
6	5.11	3.02 (13)	50%	3.18 (16)	53%
8	5.12	4.15 (9)	52%	3.61 (14)	45%

The value in ( ) is the total number of iterations to achieve the prescribed tolerance.

single-threaded I/O is resolved *p-feda* will incorporate the SSD device to significantly enhance its performance.

5.3 Evaluation of Varying Domains:

The most important feature of this research is the speedup obtained by the parallel solution. In putting all other factors aside, the bottom line is to examine the substructured finite element model faster and accurately on a concurrent machine compared to a sequential machine while keeping the overhead to a minimum degree. It is expected that the speedup will increase as the number of processors increase with a theoretical limit set to m where m is the number of domains the finite element model has been subdivided into. As a reminder if m>2 then the m domains will be performed on m logical processors.

Table 5.2: Analysis of Various Domains for a 64 Element Plate Using a Subspace (q) of 6

Number of Processors	Figure Number	Speedup (Tol=10^{-4})	Efficiency (Tol=10^{-4})	Speedup (Tol=10^{-7})	Efficiency (Tol=10^{-7})
1	5.2	1.00 (9)	100%	1.00 (15)	100%
2	5.5	1.83 (9)	92%	2.08 (13)	104%
4	5.8	2.67 (9)	67%	3.10 (13)	77%
6	5.11	4.04 (7)	67%	5.30 (9)	88%
8	5.12	2.27 (17)	28%	3.19 (20)	40%

A 64 element rectangular plate, Figure 5.2, containing 403 dof is tested to determine the speedup and efficiency on two, four, six and eight processors shown in Table 5.1 for q = 2 and Table 5.2 for q = 6. The decoupled plates are shown in Figures 5.5–5.13 to show the global fronts and element numbering layout. A sample output of program *p- -feda* for the two domain configuration is located in Appendix F. Favorable results were obtained on the two and four processor models with the six and eight processor models developing trouble due to the high number of dof on the global front. The number of iterations taken to achieve tolerance plays a big part in determining the overall speedup of the system, e.g. a greater number of iterations in the parallel solution compared to the sequential solution will significantly lower the speedup. A thorough evaluation will be made on the various number of domains to show the advantages and deficiencies of *p–feda* compared to sequential FEDA by looking at specific subroutines and communication links.

The two processor model (Figure 5.5) performed consistently well, average speedup of 1.84, and even advanced above the theoretical limit for q = 6 and a tolerance of 10^{-7} because of a lower number of iterations, Tables 5.1 and

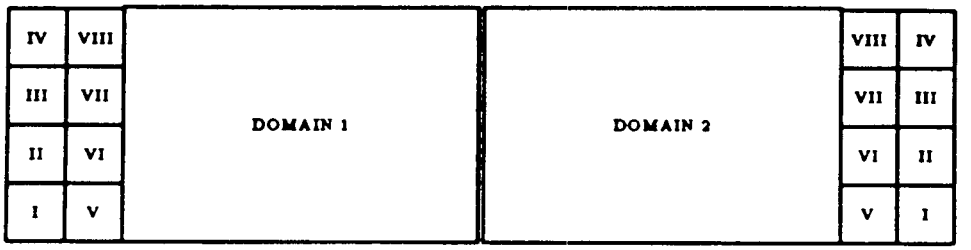

Figure 5.5: Two Domain Configuration of 64 Element Plate with Element Numbering Scheme

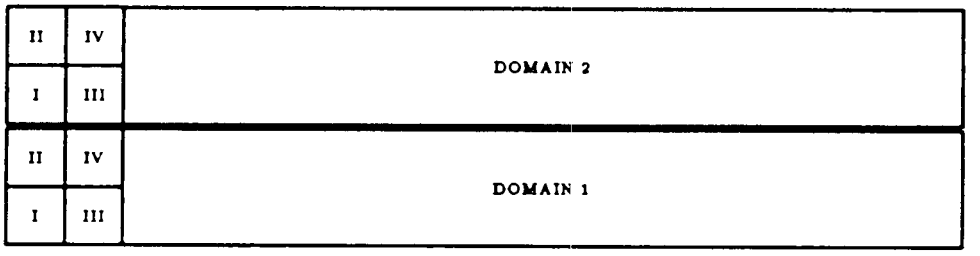

Figure 5.6: Two Domain Configuration with Horizontal Global Front for 64 Element Plate

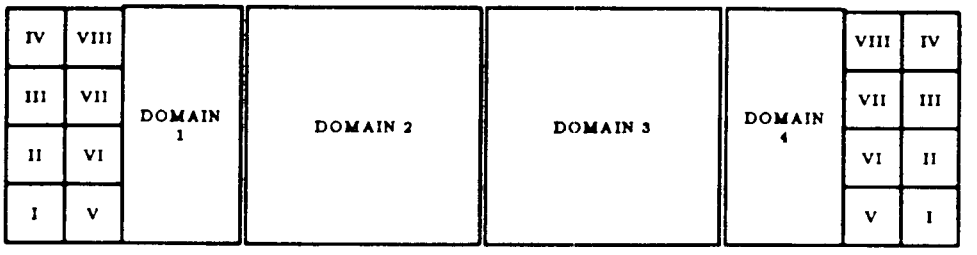

Figure 5.7: Four Domain Idealization of 64 Element Plate with Element Numbering Scheme and Vertical Local Fronts

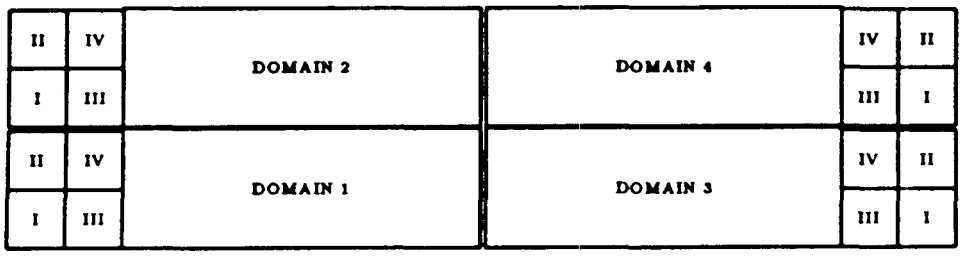

Figure 5.8: Four Domain Idealization with a Cross (+) Front for the 64 Element Plate

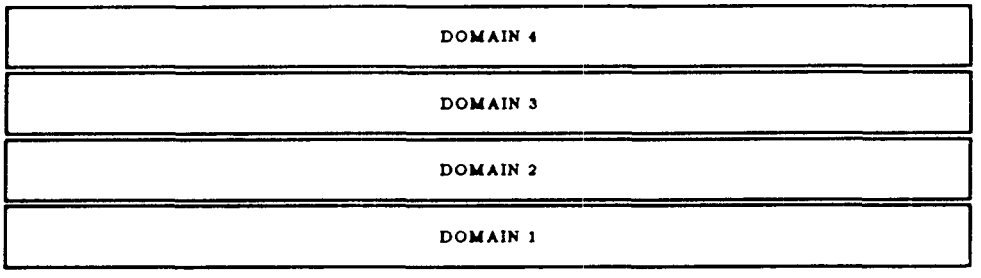

Figure 5.9: Horizontal Global Front with Four Domains on a 64 Element Plate

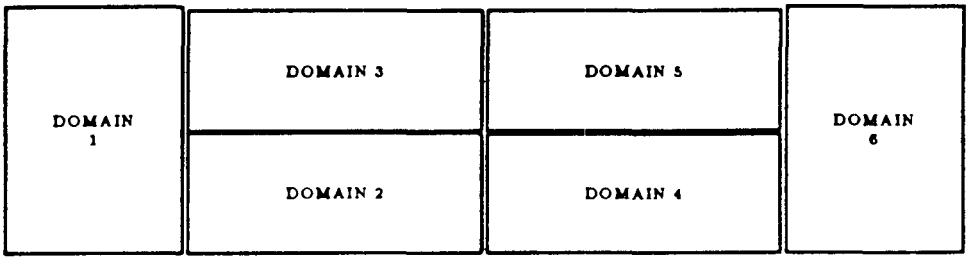

Figure 5.10: Six Domain Configuration of the 64 Element Plate

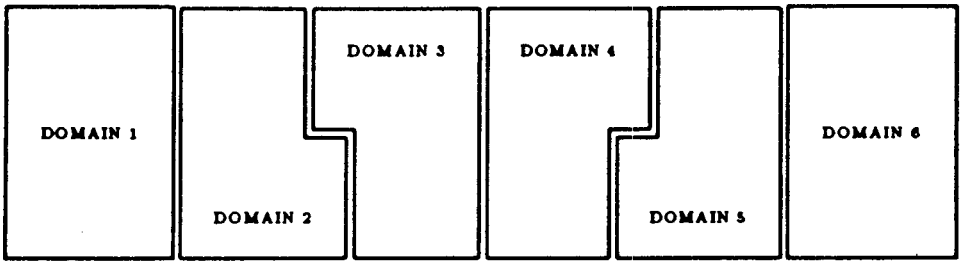

Figure 5.11: Six Domain Idealization of a 64 Element Plate, Domains 1 and 6 have 12 Elements Each with all Others Containing Ten Elements

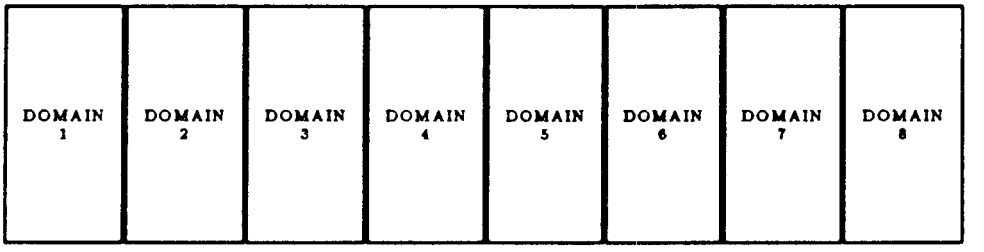

Figure 5.12: Eight Domain Configuration of the 64 Element Plate with Vertical Global Fronts

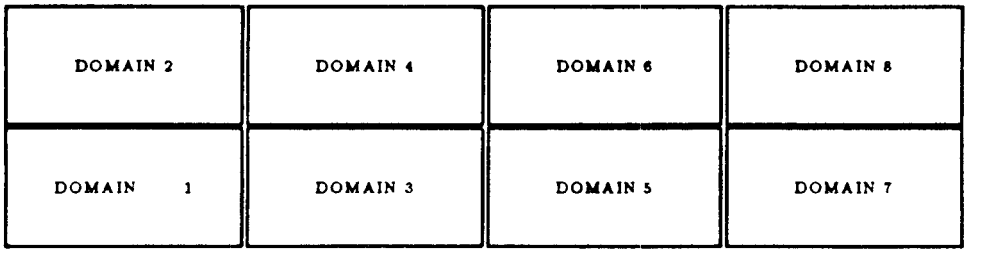

Figure 5.13: Eight Domain Configuration of 64 Element Plate

Table 5.3: Evaluation of 64 Element Plate Limited to Two Subspace Iterations

Number of Processors	Figure Number	Speedup	Efficiency	Speedup	Efficiency
		q = 2		q = 6	
1	5.2	1.00	100%	1.00	100%
2	5.5	1.68	84%	1.75	88%
4	5.7	2.84	71%	2.91	73%
6	5.11	2.88	48%	3.00	50%
8	5.12	2.83	35%	3.18	40%

5.2. When comparing the results of Tables 5.1, 5.2 and 5.3 the two domain model gains momentum as the number of iterations increase where the total number of sequential iterations are equivalent to the total number of parallel iterations. The global front has only 13 dof which is the lowest possible number for a two domain model.

This is the only system where an evaluation of the communication links could be verified. It was found that the overhead associated with transmitting information from one processor to another through common blocks was minimal.

Scanning Tables 5.1 and 5.2 shows that the results from the four domain models in Figures 5.7 and 5.8 benefited greatly from a lower number of iterations in the parallel solution. In Table 5.2 the domain model in Figure 5.8 was used over the model in Figure 5.7 because of the abnormally high number of iterations, 21 and 27, taken by the model in Figure 5.7 to converge. The reason for the lower speedup in Table 5.2 is a consequence of a greater number of degrees–of–freedom (71) for Figure 5.8 compared to 39 dof in Figure 5.7. This concern will be addressed later in the chapter. Therefore, to get a more comparable speedup look to Table 5.3 where a limited number of iterations were placed upon a four domain problem.

5.4 Examination of Subroutines:

To initially start the multitasking package a main program was developed to set events and map out all domain processors. The time taken to perform this task was calculated to be between 20 to 60 milliseconds, which does not have a big impact on the total execution time. As mentioned earlier in Section 3.2, at the first synchronization point all input data is read into task one and passed to the other tasks because of problems with I/O (only single-threaded I/O available) on the Cray. This causes a 1.0 second delay until all processors can move forward again. The subroutine that handled the dynamic dimensioning of arrays has no speedup and takes approximately the same amount of time regardless of the number of processors but since it takes less than one millisecond to perform all calculations, the subroutine will be assumed negligible in the total execution time.

In DMATRON, the subroutine that determines the first and last appearance of all nodes in its domain has a very low speedup for all sizes of domains. This subroutine takes about 2% of the total execution time to complete, some overhead is accumulated in this subroutine but is not critical to the total execution time. The creation of model matrices, $[K]^e$ and $[M]^e$, is the first place where significant speedup is achieved because the finite element model is substructured into an equal number of elements in each task; the individual tasks should have an ideal speedup of 2.0, 4.0, 6.0 and 8.0 for two, four, six and eight processors in subroutine ESTIFF.

Referring to Table 5.4, the two, four and eight domain structures had efficiencies of 89%, 90% and 90% for subroutine ESTIFF which means some overhead has been complied at this point due to parallel processing. In the unbalanced six processor model (Figure 5.11) the efficiency is only 80% as a result of the extra elements in domains one and six.

After the element matrices have been generated, the program is ready to begin the solution-resolution process to determine the natural frequencies of the system. The first and most critical subroutine is DFRONT where the multi-frontal technique is implemented along with the assembly and elimination of the global front. The success of the parallel algorithm is dependent upon the amount of dof on the global front, as the number of domains in-

Table 5.4: Subroutine Speedups for Varying Domain Sizes with q = 6

Subroutine	Number of Processors			
	Two	Four	Six	Eight
DMATRON	1.27	1.96	2.13	2.40
ESTIFF	1.78	3.59	4.78	7.17
PLOT	1.27	2.16	2.74	4.04
First Iteration				
DFRONT	1.60	1.96	1.21	0.91
DCONDS	1.86	3.63	4.75	6.96
RELOAD	1.92	3.81	5.08	7.65
Second Iteration				
DFRONT	1.83	2.55	2.37	2.34
DCONDS	1.86	3.64	4.81	6.97
RELOAD	1.92	3.84	5.06	7.64

crease so does the number of dof on the global front. This subroutine gets progressively worse as the number of dof on the global front and domains increase which can be seen in Table 5.4. The first iteration's execution time in DFRONT will always be greater than the remaining subspace iteration's execution time because of a lower number of calculations. All equations needed in resolve that were previously calculated are saved for future use, e.g. in the Gauss-Jordan method for the elimination of the global front all variables divided by the pivot in their respective equations are saved for resolution. A more comprehensive investigation will be conducted later in Section 5.6. The remaining two subroutines DCONDS and RELOAD perform the calculations of the modified subspace method. Some overhead is accompanied with these subroutines but overall their speedups were consistent and performed very well.

For the two processor model (Figure 5.5) the percentage of total execution time used by the different subroutines in *p-feda* are presented in Table 5.5. The sum of all values in Table 5.5 is 96% which leaves 4% of the total execution time due to the overhead of parallel processing. In conclusion,

Table 5.5: Percentage of Execution Time taken by *p–feda* for the Two Domain Model in Figure 5.5

Subroutine Name	Time (%)	
	1^{st} Iteration	2^{nd} Iteration
FINPUT	1%	
DNURSE	Negligible	
DMATRON	3%	
ESTIFF	32%	
DFRONT	12%	7%
DCONDS	12%	12%
RELOAD	5%	5%
PLOT		7%

a recap of the overhead associated with *p–feda* is given:

1. extra coding to implement the parallel processing.
2. extra storage requirements used by the separate task processors.
3. input of data and the map of the pre-front needed in the solution.
4. communication links used to pass information.
5. assembly and elimination of the global front performed within each task.
6. calculations that are not performed on an element by element basis.

5.5 Impact of Increasing Elements:

The effect of increasing the number of elements and degrees–of–freedom (dof) on the overall plate will be investigated. All other factors are kept constant, i.e. q = 6, two or six subspace iterations, always dividing the plate into two

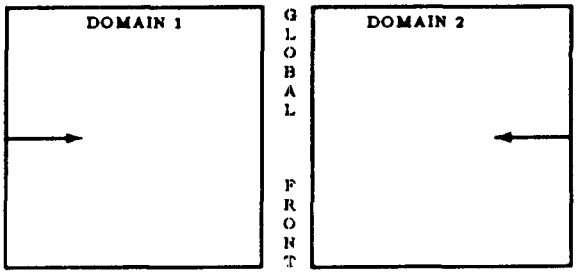

Figure 5.14: Subdivided Plate for all Test Runs with the Direction of the Local Fronts Shown

balanced domains, 13 dof on the global front, direction the wave front sweeps across the domain as shown in Figure 5.14 and the size of the wave front. Plates consisting of 8, 16, 24, 32, 40 and 64 elements, Figures 5.1 and 5.2 were used in this test. Referring to Figure 5.15, the speedup is increasing at a constant rate for both the two and six subspace iteration test cases as the number of elements increase. These results show that for large finite element problems a significant increase in computational speedup can be achieved if the number of elements per domain are large enough to overcome the deficiency of eliminating the global front. The percentage of execution time taken by DFRONT becomes less of a factor as the number of iterations and elements increase. The 8 and 16 element plates for two subspace iterations are affected by the single–threaded I/O more than the larger size plates.

Looking at the execution time shows a vast improvement in the parallel algorithm with the added elements. However, another determinant to be evaluated is the work done or CPU time for the solution process; the speedup for the work done is displayed in Figure 5.18. The ideal speedup would be 1.00 for the parallel solution, i.e. the amount of work done would be equal for both the parallel and sequential solutions. Although the work done for the parallel solution is greater than the sequential solution in all cases, the parallel CPU

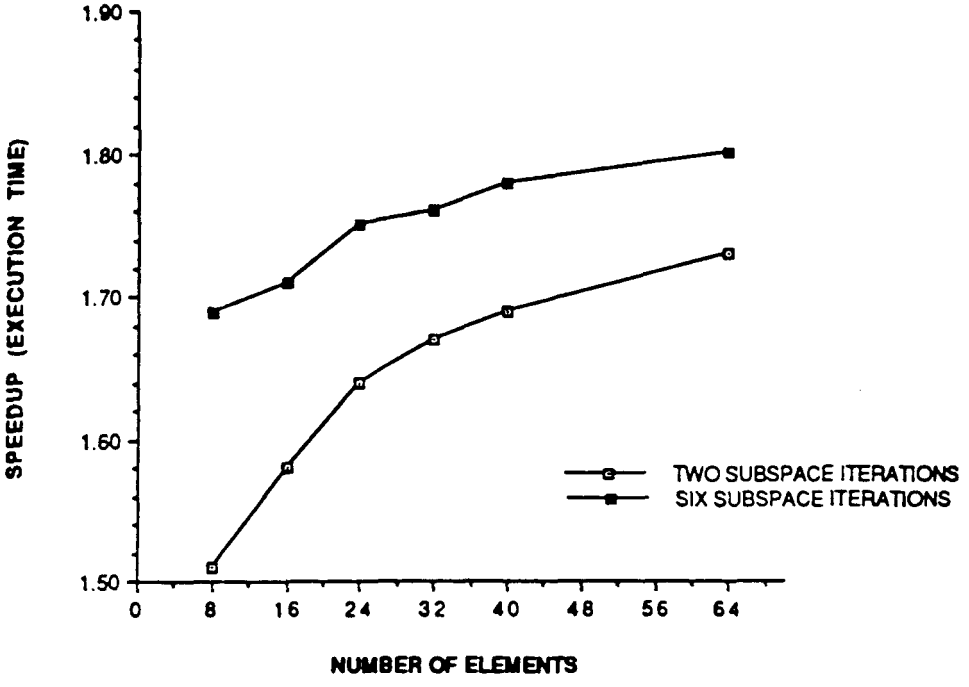

Figure 5.15: Effect of Increasing the Number of Elements on the Execution Time with all Other Factors Constant

time is approaching the sequential CPU time at a steady pace as the number of elements are expanded. This is a very notable factor when applying the parallel algorithm to enormous finite element models because the work done by the parallel solution is not increasing as rapidly as the sequential solution. Therefore, the overhead associated with parallel processing is becoming less noticeable as the number of elements increase with all major factors that affect the solution set at constant values.

5.6 Subspace Dimension:

In this study the number of eigenvalues and mode shapes will be increased to determine its impact on the algorithm. All factors are kept constant when using the 64 element plate shown in Figures 5.2 and 5.5 with test runs limited to two subspace iterations. Displayed in Figure 5.17 is the speedup relative to the increasing eigenvalues (q = 2, 4, 6, 8 and 10), a speedup of

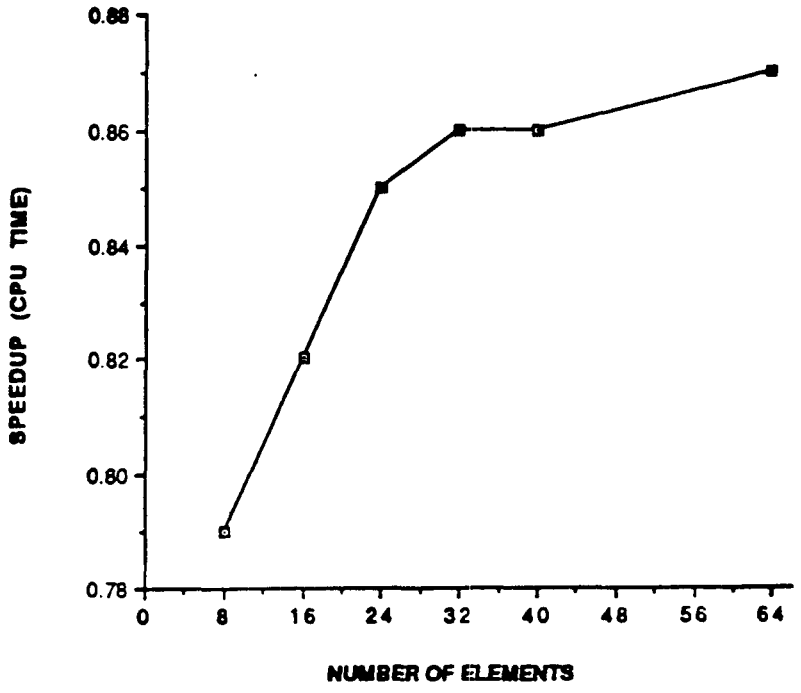

Figure 5.16: CPU Time Affected by Extra Elements with all Major Factors Constant a Limit of Two Subspace Iterations were Used

execution time shows a steady increase in the speedup from 1.68 to 1.75 for the largest two subspace dimensions. In conclusion, for large finite element problems increasing the subspace size adds no extra overhead and shows a steady increase in speedup for a higher number of eigenpairs.

Determining the effect of extra eigenpairs in the subspace on the CPU time will be the next goal. Referring to Figure 5.18, the speedup for the CPU time slowly builds up as the eigenvalues expand in number.

Consequently, this leads to the same conclusions found in the previous section, i.e. the overhead associated with parallel processing is becoming less noticeable as the number of eigenvalues and mode shapes increase while keeping all major factors constant.

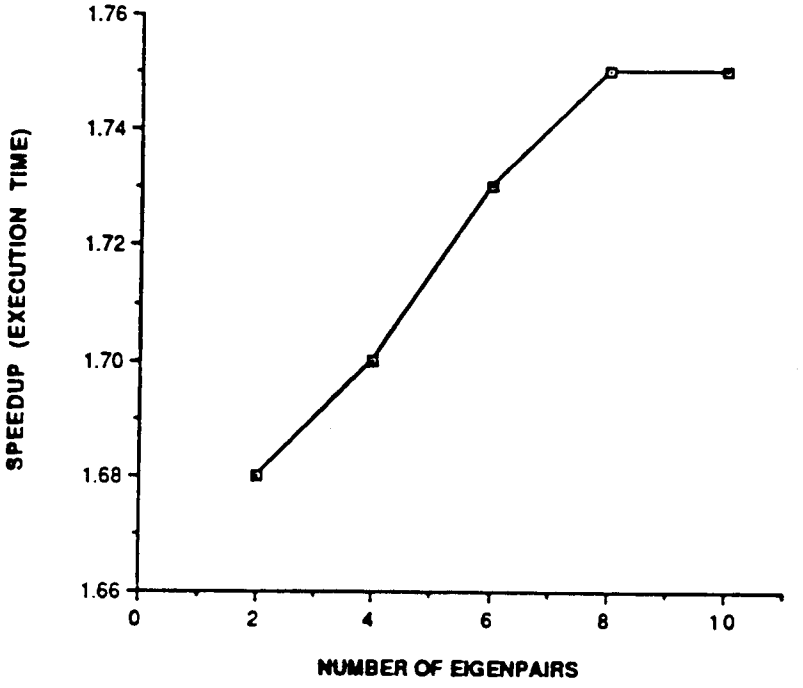

Figure 5.17: Impact on Execution Time with an Increasing Subspace Dimension with all Other Factors Constant

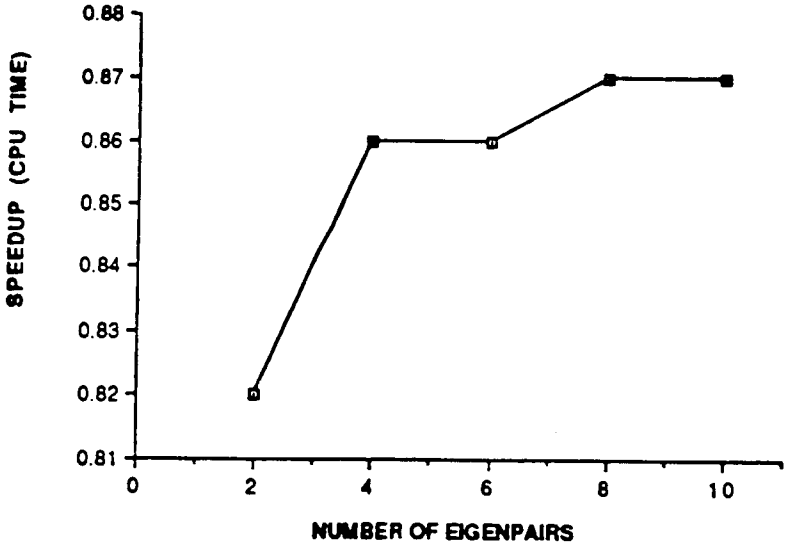

Figure 5.18: Influence of Increased Eigenpairs on the CPU Time Keeping all Factors Constant

5.7 Size of Global Front:

A primary section of additional coding for *p–feda* is associated with the assembly and elimination of the global front. A major portion of the overhead is accumulated at this section of the domain task.

Keeping the degrees–of–freedom on the global front to the lowest possible total by subdividing the finite element model appropriately will increase the speedup and efficiency of the overall problem. When the number of domains increase so too does the dof on the global front and the best one can hope for is an equivalent number of dof on the boundaries when the finite element problem is subdivided into a greater number of domains compared to the simplest two domain substructure.

This has proven to be a deficiency in *p–feda* which can be seen in Table 5.6 where the global front varies for different test cases. For example, in the two domain problem (Figure 5.5) with 13 dof on the global front and 115 dof remaining in each domain, subroutine DFRONT in the first and second iteration take up 19% of the total execution time. In contrast, the eight domain problem with 91 dof on the global front and 19 dof remaining in each domain takes 64% of the total execution time.

The four domain model shown in Figures 5.7-5.9 will be investigated to determine the impact of increasing the dof on the global front with all other factors constant, i.e. q = 6 and two subspace iterations. In Figure 5.19 the results of the four domain models located in Table 5.6 are plotted. The degrees–of–freedom in one domain are the sum of the global front dof and the dof remaining in one domain, e.g. in Table 5.6 for Figure 5.7, the dof for each separate domain is 51 and the dof on the global front is 39 with a total dof in one domain equaling 89 for this test case, the other totals are 114 dof for Figure 5.8 and 198 dof for Figure 5.9. Clearly shown in Figure 5.19 is the impact of increasing the global front in *p–feda*, a significant loss in speedup is witnessed as the dof on the global front front increase and the remaining dof in each domain decrease.

In Figures 5.7 and 5.9, the problem of an idle domain or waiting by one task will arise in DFRONT even though all domains have an equal number

Table 5.6: Impact of Degrees-of-Freedom on Global Front for q = 6 and Two Subspace Iterations

m	Figure Number	DOF Located on		Speedup		
		Global Front	One Domain	Overall	DFRONT	
					(1)	(2)
2	5.5	13	115	1.75	1.60	1.83
	5.6	61	91	1.38	0.66	1.02
4	5.7	39	51	2.91	1.42	2.55
	5.8	71	43	2.37	0.93	1.95
	5.9	183	15	0.40	0.06	0.38
6	5.10	73	35	3.00	1.13	2.34
	5.11	73	35	2.94	1.21	2.37
8	5.12	91	19	3.18	0.91	2.34
	5.13	91	19	3.08	0.84	2.35

The value in ( ) corresponds to the first or second iteration.

of elements, as a consequence of dissimilar domains, e.g. in domains two and three a local front is located on each side of the domain, whereas in domains one and four only one local front exists. This will cause unbalancing in the processors work load because of an increased frontwidth in domains two and three. The domains can not continue execution until all domains post their event in DFRONT, therefore a domain may sit idle and cause overhead in the parallel solution. For Figure 5.7, it was found that DFRONT had a 0.4 seconds difference between the slowest (domain 2) and fastest (domain 1) domains. Since domain 2 has a larger wave front due to the local fronts on either side of the domain, the wave front will sweep across the domain slower than domain 1 which has only one local front.

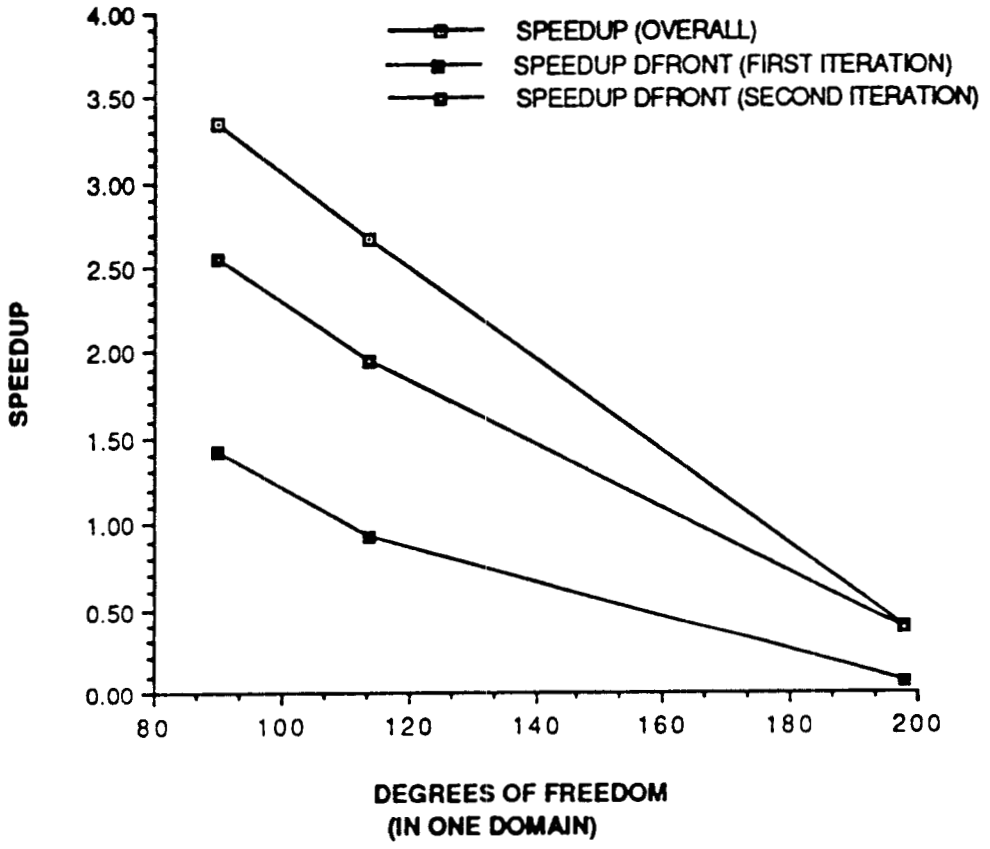

Figure 5.19: Impact of Increasing the Global Front on the Different Four Domain Models (Figures 5.7-5.9)

Chapter 6

Conclusions and Recommendations

The frontal solution [24] and modified subspace method [4] were presented with their advantages. Parallel implementation of these methods on the Cray X-MP/24 proved successful in achieving computational speedup. In addition, the use of multitasking routines [18] installed on the Cray X-MP computer has proven to be very efficient in mapping each domain to a user task.

The parallel program described in this report was found to be an accurate and effective algorithm to solve large linear finite element eigenproblems on the Cray X-MP/24 computer. The parallel eigensolver demonstrates that speedups in execution time can be achieved compared to a similar sequential algorithm (Figures 6.1 and 6.2). Utilization of the Cray's multitasking library also proved to be an efficient tool to parallelize the FORTRAN code. Multitasking subroutines were found to be of minimal impact on the total execution time.

The parallel program takes advantage of the shared and local memory on the MIMD Cray machine while successfully using a completely connected architecture to transmit information from one processor to another.

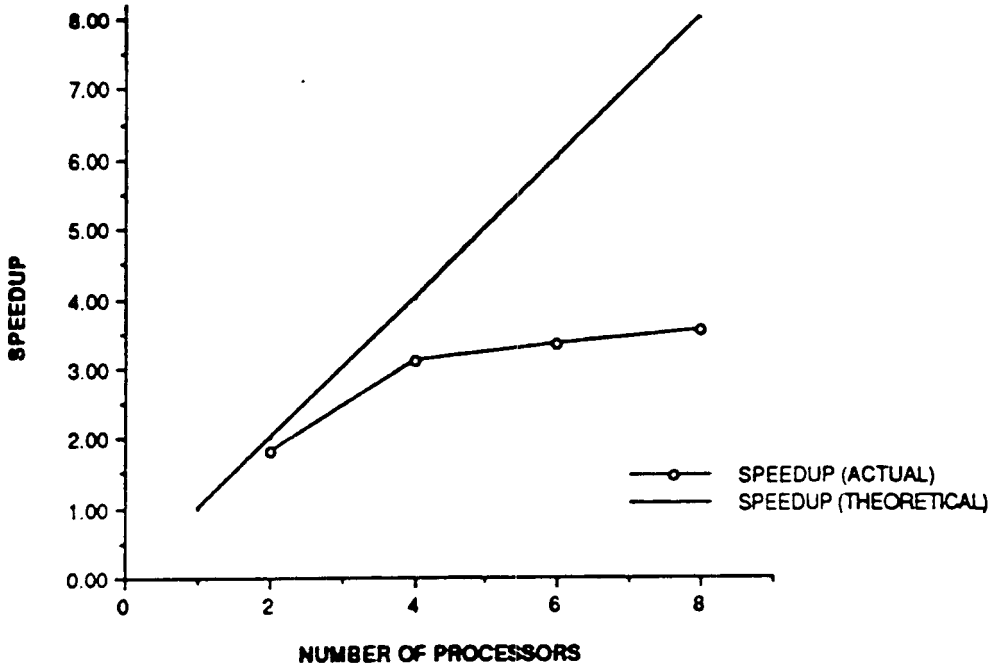

Figure 6.1: Effectiveness of *p–feda* for a 64 Element Plate with q = 6 and Six Subspace Iterations

Communication links were performed by using common blocks to store all data rather than the Cray's SSD solid-state storage device. When the problems with the SSD are resolved, i.e. the capability to perform multi-threaded I/O rather than the single-threaded I/O currently available, the authors recommends this I/O device be used so that larger and more sophisticated problems can be solved with significant improvement in performance and increased flexibility. Furthermore, the arrays used in the common blocks limited the total number of degrees–of–freedom on the global front that the program could store, especially as the number of domains increased. By utilizing the SSD, data input could be read in by each separate task, thereby lowering storage requirements and execution time in the solution process. This would eliminate waiting (overhead) by the i^{th} task for the input data from task one.

The major deficiency in this parallel algorithm was elimination of the dof on the global front, as the domains increased so did the dof on the boundary. The extra sequential calculations performed by each task to handle the

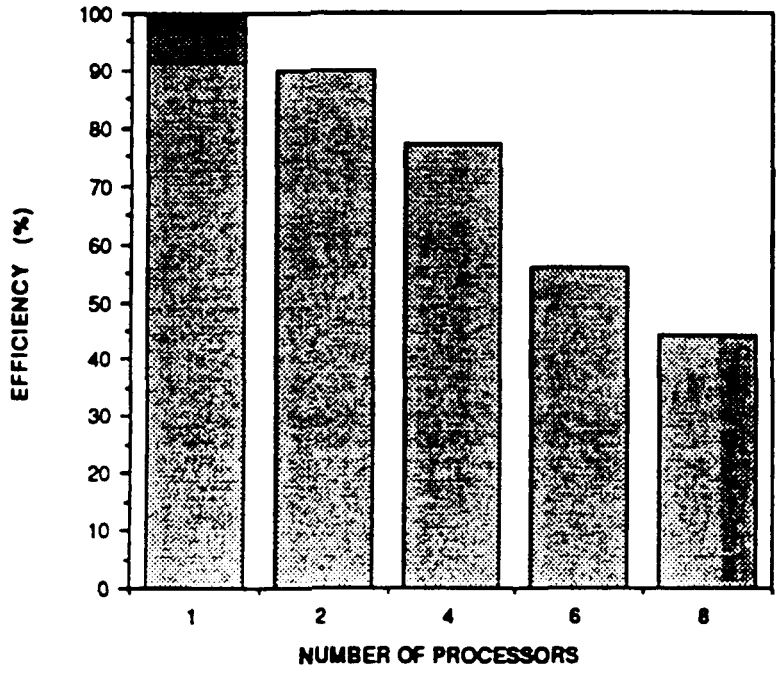

Figure 6.2: Evaluation of 64 Element Plate on *p–feda* for q = 6 and Six Subspace Iterations

global front lowered the speedup and efficiency significantly. On the other hand, the performance for the creation of the stiffness and mass matrices and the modified subspace method were extremely encouraging and indicate the effectiveness of multitasking on the Cray X-MP computer. Therefore, when calculations were performed on an element by element basis, in parallel, speedup and efficiency was very high compared to the section where extra sequential coding was required.

When subdividing a finite element model into m domains one should choose the configuration with the lowest possible dof on the global front for this will increase speedup and efficiency of the parallel algorithm. In addition, load balancing, i.e. assigning an equivalent amount of work to each task by keeping the number of elements and frontwidth equal in all domains, is very important in the performance of *p–feda*. Communication links were found to be of minimal impact (no idle domains) in parallel processing if all domains were balanced. Some overhead is accumulated due to one processor being faster than another but the overall execution time of the problem

will overshadow this deficiency. Based on the results obtained, the authors recommends that one should only further subdivide their finite element model into a greater number of domains only if the number of elements in the separate domains are large enough to overcome the overhead associated with the global front elimination. As a result of the greater number of elements per domain, the amount of work performed by the individual tasks on all subroutines will outweigh subroutine DFRONT in the total execution time. Element numbering is an important aspect in lowering the frontwidth for the frontal technique. The user should always number the elements in the domain so that the wave front converges to the local front, if at all possible. In addition, to optimize element numbering, one should follow the general rules that are applied to the optimum node numbering [9].

Reported earlier were the two types of multitasking, macrotasking and microtasking, with macrotasking being used herein. The parallelization for the solution of the linear simultaneous equations on the global front by using microtasking routines [18] could prove to be beneficial because all tasks perform the same calculations on the global front stiffness matrix. These calculations could be concurrently performed to increase speedup of *p–feda*, especially in the larger size domains, by using a parallel direct method [26,27,34,35].

Bibliography

[1] Akl, F. A. and Hackett, R., "Multi-Frontal Algorthim for Parallel Processing of Large Eigenproblems," *Proceedings of 27th SDM Conference*, Paper 86-0929, pp. 395-399, 1986.

[2] Akl, F., Dilger, W. H. and Irons, B. M., "Acceleration of Subspace Iteration", *International Journal for Numerical Methods in Engineering*, Vol. 18, 4, pp. 583-589, 1983.

[3] Akl, F. A., Dilger, W. H. and Irons, B. M., *"FEDA" A Finite Element Dynamic Analysis Program*, Department of Civil Engineering, The University of Calgary, Canada, 1979.

[4] Akl, F., Dilger, W. H. and Irons, B. M., "Over-relaxation and Subspace Iteration", *International Journal for Numerical Methods in Engineering*, Vol. 14, 4, pp. 629-630, 1979.

[5] Akl, F. A., *A Modified Subspace Algorithm for Dynamic Analysis by Finite Elements, and Vibration Control of Structures*, Ph.D. Thesis, Department of Civil Engineering, The University of Calgary, Canada, December 1978.

[6] Adams, L., and Ortega, J., "A Multi-Color SOR Method for Parallel Computation", *Proceedings of International Conference on Parallel Processing*, pp. 53-56, August 1982.

[7] Arora, J. S. and Nguyen, D. T., "Eigensolution for Large Structural Systems with Substructures", *International Journal for Numerical Methods in Engineering* 15, pp. 333-341, 1980.

[8] Baldwin, J. T., Razzaque, A. and Irons, B. M., "Shape Function Subroutine for an Isoparametric Thin Plate Element", *International Journal for Numerical Analysis in Engineering* 7, pp. 431-440, 1973.

[9] Bathe, K. J., *Finite Element Procedures in Engineering Analysis*, Prentice-Hall, New York, NY, 1982.

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

[11] Bathe, K. J., "Convergence of Subspace Iteration", *U.S.-Germany Symposium: Formulations and Computational Procedures in Finite Element Analysis*, M.I.T., MA, 1976.

[12] Baudet, G., "Asynchronous Iterative Methods for Multiprocessors", *Journal of the ACM*, Vol. 25, April 1978.

[13] Bauer, F. L., "Das Verfahren der Treppen-Iteration und Verwandte Verfahen Zur Losung Algebraischer Eigenwertprobleme", *ZAMP* 8, pp. 214-235, 1957.

[14] Bostic, S. and Fulton, R., "Implementation of the Lanzos Method for Structural Vibration Analysis on a Parallel Computer", AIAA Paper 86-0930, *AIAA/ASME/ASCE/AHS 27th Structures, Structural Dynamics and Materials Conference*, San Antonio, TX, May 1986.

[15] Bostic, S. and Fulton, R., "A Concurrent Processing Implementation for Stuctural Vibration Analysis", AIAA Paper 85-0783, *AIAA/ASME/ASCE/AHS 26th Structures, Structural Dynamics and Materials Conference*, Orlando, FL, April 1985.

[16] Clint and Jennings, A., "The Evaluation of Eigenvalues and Eigenvectors of Real Symmetric Matrices by Simulataneous Iteration", *Computer Journal* 13, pp. 76-80, 1976.

[17] Corr, R. B. and Jennings, A., "A Simultaneous Iteration Algorithm for Symmetric Eigenvalue Problems", *International Journal for Numerical Methods in Engineering*, John Wiley and Sons, Vol. 10, pp. 647-663, 1970.

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

[19] Crockett, T. W., and Knott, J. D., *System Software for the Finite Element Machine*, NASA CR 3870, February 1985.

[20] Darbhamulla, S. P., Razzaq, Z. and Storaasli, O., "Concurrent Processing in Nonlinear Structural Stability", *Proceedings of the 27th SDM Conference*, San Antonio, TX, Part 1, Paper 86-0979, pp. 545-550, May 19-21 1986.

[21] Gannon, D., "A Note on Pipelining a Mesh Connected Multiprocessor for Finite Element Problems by Nested Dissection", *Proceedings of the 1980 International Conference on Parallel Processing*, pp. 197-204, August 1980.

[22] Hwang, K. and Briggs, F. A., *Computer Architecture and Parallel Processing*, McGraw-Hill, Inc., New York, NY, 1984.

[23] (Irons, B. M. and Ahmad, S., "Techniques of Finite Elements", Ellis Horwood, England, 1979.

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

[25] Kopal, Z., *Numerical Analysis*, Chapman and Hale Ltd., London, 1961.

[26] Leuze, M. R., *Parallel Triangularization of Substructured Finite Element Problems*, NASA CR-172466, September 1984.

[27] McGregor, J. and Salama, M., "Finite Element Computation with Parallel VLSI," *Proceedings of the 8th Conference on Electronic Computation, ASCE*, Houston, TX, pp. 540-553, February 1983. 1983.

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

[29] Melosh, R. J. and Bramford, R. M., "Efficient Solution of Loading Deflection Equations", *Journal of American Society of Civil Engineers, Structural Division*, Paper No. 5610, pp. 661-676, 1969.

[30] Paz, M., *Structural Dynamics Theory and Computation*, Van Nostrand Reinhold Company, New York, NY, 1985.

[31] Quinn, M. J., *Designing Efficient Algorithms for Parallel Computers*, McGraw-Hill Book Company, Inc., New York, NY, 1987.

[32] Rheinboldt, W. C. (Panel Chairman), "Report of the Panel on Future Directions in Computational Mathematics, Algorithms, and Scientific Software", *Society of Industrial and Applied Mathematics*, Philadelphia, PA, 1985.

[33] Rutishauser, H., "Computational Aspects of F. L. Bauer's Simultaneous Iteration Method", *Numerische Mathematic* 13, pp. 4-13, 1969.

[34] Salama, S., Utku, S. and Melosh, R., "Parallel Solution of Finite Element Equations," *Proceedings of the 8th Conference on Electronic Computation, ASCE*, Houston, TX, pp. 526-539, February 1983.

[35] Sameh, A. and Brent, R. P., "Solving Triangular Systems on a Parallel Computer," *SIAM Journal on Numerical Analysis*, 14, December 1977.

[36] Schaeffer, H. G., *MSC/NASTRAN Primer Static and Normal Modes Analysis*, Wallace Press Inc., Milford, NH, 1982.

[37] Schendel, U., *Introduction to Numerical Methods for Parallel Computations*, John Wiley and Sons, New York, NY, 1984.

[38] Silbar, M. L., "The Pursuit of Parallelism MOSAIC", *National Science Foundation*, Vol. 16, No. 3, pp. 8-17, 1988.

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

[40] Storaasli, O., and Bergan, P., "A Nonlinear Substructuring Method for Concurrent Processing Computers", *Proceedings of the 27th SDM Conference*, San Antonio, TX, Part 2, Paper 86-0852, pp. 13-20, May 19-21 1986.

[41] Storaasli, O., Peebles, S., Crockett, T., Knott, J. and Adams, L., *The Finite Element Machine: An Experiment in Parallel Processing*, NASA TM-84514, July 1982.

Appendix A

The Modified Subspace Method

The solution of the eigenproblem is usually required in applications involving free vibration and dynamic response analyses of finite element models. It is typically a time consuming process if the structural system is idealized through a large number of degrees of freedom. However experience has shown that the response of structures can be adequately estimated using only a few eigenvectors, and that the contribution of higher order eigenvectors can often be neglected without significant loss in accuracy.

The subspace method is one the techniques suitable for the evaluation of the lowest order eigenpairs in large structural systems [10]. In this appendix, the modified subspace method [2] is briefly presented and its characteristics are discussed.

A.1 The Eigenproblem in Structural Dynamics

Consider the undamped free vibration of an N degree-of-freedom system:

$$[M]\{\ddot{x}\} + [K]\{x\} = \{0\} \tag{A.1}$$

Equation A.1 expresses the motion of structural system in physical coordinates $\{x\}$. However, it is more convenient to transform Equation A.1 to a new basis. Substituting

$$\{x\} = \{\phi\}\eta \tag{A.2}$$

where η is defined as a generalized displacement coordinate and ϕ is a new basis of coordinates; thus

$$[M]\{\phi\}\ddot{\eta} + [K]\{\phi\}\eta = \{0\} \tag{A.3}$$

replaces Equation A.5 in the new basis. For structural systems undergoing periodic oscillations:

$$\eta = a\sin(\omega t - \theta) \tag{A.4}$$

Substituting in Equation A.3,

$$[K]\{\phi\} = \lambda[M]\{\phi\} \tag{A.5}$$

which yields N solutions for N eigenvectors $\{\phi\}_i$ and the corresponding N eigenvalues λ_i. The complete solution to Equation A.5 can be expressed as:

$$[K][\Phi] = [M][\Phi][\Lambda] \tag{A.6}$$

where $[\Phi]$ is an n.n modal matrix containing eigevectors $\{\phi\}_i$ columnwise, and $[\Lambda]$ is the spectral diagonal matrix of λ_i.

A.2 Description of the Sequential Algorithm

Subspace iterations schemes can be traced back to the fifties when [13] proposed a "bi-iteration" method for solving the standard eigenvalue problem.

Later this method was applied to symmetric positive definite matrices by [33], and to symmetric $[K]$ matrices by [16].

In 1971 Bathe introduced the subspace method, which has the advantage of solving the generalized eigenproblem directly without transformation to the standard form. The method essentially iterates on a n-dimensional subspace rather than on n individual iteration vectors, where $n \leq N$.

The complete set of eigenvectors $[\Phi]$ of Equation A.6 spans the N-dimensional space describing the system. These vectors are linearly independent, and satisfy the following conditions:

1. M-orthonormality:
$$[\Phi]^T[M][\Phi] = [I] \tag{A.7}$$

2. K-orthogonality:
$$[\Phi]^T[K][\Phi] = [\Omega] \tag{A.8}$$

They are also unique if the system does not have multiple eigenvalues, otherwise they are not unique within the subspace of eigenvectors with multiple eigenvalues; but this subspace itself is unique, and satisfies Equations A.7 and A.8.

To demonstrate the procedure of the modified subspace method, let us assume that we wish to calculate the n lowest eigenpairs of Equation A.6. The n eigenvectors are said to span the least-dominant subspace of the operator $[K]$ and $[M]$ which is called E_∞. The modified subspace iteration technique can be stated as follows:

1. Establish n starting vectors $[V]_1$ which are said to span subspace E_1, where $n \leq N$. Random numbers may be used for this purpose.

2. For each subspace E_l, iterate from E_l to E_{l+1}, where $l = 1, 2, 3, 4, \ldots$
$$\begin{aligned} [B]_l &= [M][V]_l \\ [V]^*_{l+1} &= [K]^{-1}[B]_l - \beta[V]_l \end{aligned} \tag{A.9}$$

3. Find the projections of the operators $[K]$ and $[M]$ onto E_{l+1} :

$$[K]^*_{l+1} = [V]^{*T}_{l+1}[K][V]^*_{l+1} \quad \text{(A.10)}$$
$$[M]^*_{l+1} = [V]^{*T}_{l+1}[M][V]^*_{l+1} \quad \text{(A.11)}$$

4. Solve for the eigensystem of the projected operators using an auxiliary eigen analysis routine:

$$[K]^*_{l+1}[Q]_{l+1} = [M]^*_{l+1}[Q]_{l+1}[\Lambda]_{l+1} \quad \text{(A.12)}$$

5. Find an improved approximation to the required eigenvectors:

$$[V]_{l+1} = [V]^*_{l+1}[Q]_{l+1} \quad \text{(A.13)}$$

6. Repeat from step (2) to step (5) above until the required accuracy is achieved. Finally:

$$[\Lambda]_{l+1} \longrightarrow [\Lambda], \text{ and}$$
$$[V]_{l+1} \longrightarrow [\Phi] \text{ as } l \rightarrow \infty.$$

There are a number of important observations concerning the subspace procedure:

1. $[K]^*_{l+1}$ tend toward diagonal forms, and $[Q]_{l+1}$ approaches $[I]$ as l increases.

2. The number of iterations required depends on how close E_1 is to E_∞, or in other words on how close $[V]_1$ is to $[\Phi]$.

3. $[\Lambda]_{l+1}$ is an upper bound to $[\Lambda]$

4. It is generally recommended to iterate on a number of eigenpairs large than n, e.g. $\min\{2n, (n+8)\}$ [10]. The reason for this will be appreciated when we discuss the convergence of the modified subspace method in the next section.

A.3 Behavior of the Subspace Method

In general the starting vectors $[V]_1$ are linear combination of all the n eigenvectors $[\Phi]$. However if we assume that $[V]_1$ are a combination of only the n required eigenvectors, then the subspace method converges in one step [10]. The i^{th} starting vector can be expressed as:

$$\{v\}_1^i = \sum_{j=1}^{N} a_{ji}\{\phi\}_j, \quad \text{where } i = 1, 2, \ldots, n. \tag{A.14}$$

Those vectors of order higher than n are said to be "polluting" the subspace E_l in each iteration, and basically the subspace method attempts through the set of vectors $[Q]_{l+1}$ to get the best combination of the vectors $[V]_{l+1}$ in subspace E_l to converge to subspace E_∞. The number of subspace iterations required depends directly on the "noise" due to vectors of order higher than n.

Reference [10] augmented the basic subspace approach with an elaborate procedure to establish n starting vectors $[V]_1$ as close as possible to the required subspace E_∞ in order to minimize the number of subspace iterations. In addition, the basic subspace method calls for a Sturm Sequence check on the eigensystem to verify that all the required eigenpairs have indeed been calculated.

However the authors share the view of researchers [17,33] that allocating random numbers to vectors $[V]_1$ is equally satisfactory in practice. Moreover it is very improbable that any eigenpair will be absent at the end of the iteration, particulary when random numbers are used for $[V]_1$.

A.4 Convergence of The Modified Subspace Method

Bathe and Wilson [10] reported the use of over relaxation, shifting, and Aitken's Formula to improve the convergence rate of the basic algorithm. However, no theoretical basis has been established for their performance.

Referring to the analysis presented in the previous section, the starting subspace E_1 is always polluted with eigenvectors of order higher than n. This requires the subspace procedure to iterate in order to convergence toward E_∞, or in other words to eliminate $\{\phi\}_{n+1}, \ldots, \{\phi\}_N$. Writing a typical vector in E_l

$$\begin{aligned} \{v\}_l^i &= \sum_{s=1}^{N} (a_{si}/\ \lambda_s^{l-1})\{\phi\}_s \\ &= \sum_{j=1}^{n} (a_{ji}/\ \lambda_j^{l-1})\{\phi\}_j + \sum_{q=n+1}^{N} (a_{qi}/\ \lambda_q^{l-1})\{\phi\}_q \end{aligned} \tag{A.15}$$

where the first part of the right-hand-side represents the subspace E_l that we wish to isolate, and the second part represents a complementary subspace E_l^c as a "noise" polluting subspace E_l.

The fundamental idea in accelerating the convergence of the subspace method is based on eliminating the complementary subspace E_l^c at an increased rate.This task is exactly equivalent to that of persuading the coefficients a_{qi} in Equation A.15 to convergence to zero as rapidly as possible.

Proceeding with the modified algorithm of the subspace method, inverse iteration on $\{v\}_i$ gives

$$\{v\}_{l+1}^{*i} = [K]^{-1}[M]\{v\}_l^i - \beta\{v\}_l^i \tag{A.16}$$

Substituting Equation A.1 into Equation A.16 and using Equation A.6,

$$\{v\}_{l+1}^{*i} = \sum_{j=1}^{n} (a_{ji}/\lambda_j^l)\{\phi\}_j + \sum_{q=n+1}^{N} (a_{qi}/\lambda_q^l)\{\phi\}_q - \beta \sum_{r=1}^{N} \{\phi\}_r \tag{A.17}$$

$$\{v\}_{l+1}^{*i} \longleftarrow \{v\}_{l+1}^{*i} - \beta\{v\}_l^{*i} \tag{A.18}$$

where: $0 < \beta < 1/\lambda_n$

If factor β is chosen to be equal to $1/\lambda_{n+1}$, then the contribution of vector $\{\phi\}_{n+1}$ to the complementary subspace E_{l+1}^c is entirely eliminated. In general if we choose β equal to $1/\lambda_q$, vector $\{\phi\}_q$ (where $q = n+1, \ldots, N$) does not contribute noise to the eigensystem. But the important question

now is how can we select the factor β when we do not know the eigenvalues $\lambda_{n+1}, \lambda_{n+2}, \ldots, \lambda_N$.

The following strategy proved successful in reducing the number of iterations in some cases than half that required with basic subspace method using the same starting vectors $[V]_1$:

Step 1: After one iteration, a poor estimate of λ_n is known. The lowest root r_1 of the 11^{th} order Gaussian Quadrature formula of the closed type for integrating over the range 0 to $1/\lambda_n$ [25] was used for factor β in the second iteration. Table A.1 gives the roots of the Gaussian Quadrature formula and shows how to transfer these roots from the range $(-1, 1)$ as given by [25] to $(0, 1/\lambda_n)$. This value of β is not optimal, but being small it is less dependent on the initial value of λ_n, than a higher root would be.

Step 2: After the second iteration is completed, a more accurate value for λ_n is known, the second root of the Gaussian formula was used for calculating β.

Step 3: The process is repeated with every iteration using the current value for λ_n to get the $(j-1)_{th}$ root of the Gaussian formula in the j^{th} iteration until the 11^{th} iteration is reached. If more subspace iteration are still needed, then proceed with $\beta = 0$ until the required accuracy is achieved, Experience has shown that the solution is sometimes sensitive to the eleventh root of Gaussian formula, therefore beyond the 11^{th} iteration a value of zero is assigned to factor β, i.e. a return to the basic subspace approach.

The rate of convergence of the modified subspace method can be expressed as follows:

$$\text{Rate of convergence} \leq \frac{\lambda_i(1-\beta\lambda_{n+1})}{\lambda_{n+1}(1-\beta\lambda_i)} \tag{A.19}$$

which is faster than the rate of convergence of the basic subspace algorithm of $(\lambda_i/\lambda_{n+1})$ [10]. By choosing factor β sufficiently small to be as close as possible to $1/\lambda_n$ (root r_1), the q^{th} element of noise in the complementary subspace E^c_{l+1}, and probably other elements of order less than n, are either

Table A.1: Roots of the Gaussian Quadrature Formula of the Closed Type

Root	Value	Root	Value
r_1	-0.9533098466	r_6	0.00000000000
r_2	-0.8462475646	r_7	+0.2492869301
r_3	-0.6861884690	r_8	+0.4829098210
r_4	-0.4829098210	r_9	+0.6861884690
r_5	-0.2492869301	r_{10}	+0.8463475646
Transformation Formula: $\beta = 0.5(1 + r_i)/\lambda_n$			

eliminated or minimized. As factor β grows in value from root r_2 to r_{10}, the contribution to the noise of vectors $\{\phi\}$ of progressively decreasing order up to $(n + 1)$ in the complementary subspace E^c_{l+1} are swept away in turn, collectively. This "sweeping" strategy, from root r_1 to r_{10}, ensures that no noise due to eigenvectors of higher order is reintroduced into the subspace as we proceed to eliminate the noise due to the lower eigenvectors.

Appendix B

Variable List

Variable Name	Description
EIGEN2	element eigenvectors.
EIGENSHP	profile of the modes of vibration of the system at some prescribed nodes listed in NODEIGN.
ELKSTR	projection of an element stiffness matrix onto the current subspace.
ELV	element initial displacements in modal analysis.
ELVDOT	element initial velocities in model analysis.
FULMASS	element mass matrix.
FULSTIF	element stiffness matrix.
INITIAL	maximum number of nodes with initial displacement or velocities.
INTEG	number of integer words per floating point.
IPROP	number of properties, e.g. thickness, density, etc.

Variable Name	Description
JPROP	number of sets of properties.
KODFIX	code number listing which degree of freedom is prescribe at each node.
KODLOAD	code number for loading in dynamic response analysis = 0 no uniform dynamic loads = 1 uniform dynamic loads
KODSOL	solution code = 0 static solution. = 1 dynamic response analysis by the mode superposition method = 2 dynamic response analysis by the direct integration method = 3 subspace eigen analysis
LCOEF	number of coefficients in the lower triangle of $[K]$ or $[M]$.
LDEST	element destination vector.
LENVEC	length of vector VEC/NVEC.
LENVEC1	length of vector VEC1/NVEC1.
LIMFRO	length of the limiting front width.
LNODES	element node numbers.
LPROP	the set of properties application to each element.
LVAB	maximum number of variables per element.

Variable Name	Description
MATRICES	LVAB*(LVAB+1)/2
MEQ	identifies the degree of freedom of a node for which the current equation is to be solved.
NAME	identifies the prescribed variables and their locations in VFIX (with negative sign), otherwise it gives the node number (with positive sign).
NBUFZ	maximum number of equations existing at one time in the core storage.
NCASE	maximum number of iterations in substance eigen analysis.
NDF	number of degrees of freedom per node.
NDFRO	width of the limiting front.
NDIM	dimension of the physical coordinates of the system.
NEGIEN	number of required eigenpairs.
NEIGNT	NEIGEN = NRHS
NEL	total number of elements.
NEXTIF	number of nodes with additional stiffness.
NFIX	number of nodes with some fixed values.
NGAUS	order of the gauss rule used in numerical intergration.
NLOAD	number of nodes with additional concentrated loads in static analysis.
NLOAD1	number of nodes with additional concentrated loads in dynamic response analysis (NLOAD must be equal to zero).

Variable Name	Description
NMODE	number of vibration modes used in modal analysis $\leq$ NEIGEN.
NODEIGN	list of nodes at which the profile of the modes of vibration is required.
NODEL	maximum degrees of freedom per node.
NODFIX	list of nodes with prescribed displacements.
NODINIT	list of nodes where initial displacements or velocities are given.
NODLOD	list of nodes with additional loads.
NOPTION	printing option = 0 print element stiffness, mass and load arrays. = 1 suppress printing.
NOSTIF	list of nodes with additional stiffness.
NPDFRO	node numbers in the current front.
NPIVOT	location of the pivot in each equation in eq.
NPOIN1	number of nodes in plotting vibration mode profiles.
NPOINT	total number of nodes.
NRESOL	iteration number in eigen analysis.
NRHS	number of right hand sides.
NRHS1	number of right hand sides in model analysis $\leq$ NEIGEN.

Variable Name	Description
NSTEADY	code number for steady state response under harmonic loading using modal analysis = 0 no steady state response is required. = 1 steady state response for sine loading function. = 11 steady state response for cosine loading function.
NSTIF	LIMFRO (LIMFRO + 1) /2.
NSTOP	= 0 if tolerance level TOL2 in subspace eigen analysis is not achieved. = 10 if TOL2 is achieved.
NSTRES	number of the stress components = size of matrix $[D]$.
NTYPE	type of the problem, e.g. plane stress, plane strain.
OLIGNVL	eigen values from a previous iteration.
POTEGY	potential energy for each right hand side.
PSTR	modal loads, i.e. projection of the system loads onto the system eigenvectors.
REACTN	reactions at nodes with prescribed displacements.
SIGDIG	the sum of the squares of the diagonal stiffness.
SKSTR	projection of the system $[K]$ matrix onto the current subspace.

Variable Name	Description
SMSTR	projection of the system $[M]$ matrix onto the current subspace.
STREGY	strain energy for each right hand side.
SUBLOD	in assembly, it is a subset of the right hand sides corresponding to sustif. In back substitution, it is used to hold the vector of running variables.
SUSTIF	grandpa matrix into which elements are assembled.
TOL2	tolerance level in subspace eigen analysis.
TOTLOD	total load at active nodes.
VFIX	prescribed displacement at the nodes listed in NODFIX.
VLAD	additional loads at the nodes listed in NODLOD.
VPROP	element properties.
VSTIF	additional stiffness at the nodes listed in NOSTIF.

Appendix C

Management of Files

File Type	File Number	Number of Records	Content of Records
Element	1	NEL	Each record contains ELSTIF + ELOAD of the first frontal solution
	3	NEL	ELOAD for re-solutions
	7	NEL	EIGEN1 in eigen-analysis X, XDOT, XDDOT in the dynamic response analysis
	8	NEL	ELDISP in backward order, Files i.e. first record is ELDISP of the first element.
	9	NEL	ELMASS
Solution, Re-Solution	2	NBUFZ	EQ, EQR, EQRTOT, EQSIG, NPIVOT, NAME, MDEQ in first frontal solution
	4	NBUFZ	EQR, EQRTOT in resolutions.

Appendix D

Error Messages

The diagnostics provided in program *p–feda* are listed below. It should be noted that if the dimensions of vectors VEC/NVEC and/or VEC1/NVEC1 are changed, LENVEC and/or LENVEC1, respectively, must also be changed to the same amount. Otherwise the diagnostics will be worse than useless.

Error #	Interpertation
Fatal Diagnostics in DNURSE: (in the first card)	
1	NPOIN = 0 or -ve
2	NEL = 0 or -ve
3	NCASE = 0 or -ve
4	NSTRESS = 0 or -ve
5	NTYPE = 0 or -ve
6	NGAUS $\neq$ 2 or 3 or 4
7	NODEL = 0 or -ve
8	NDFMAX = 0 or -ve
9	IPROP = 0 or -ve
10	JPROP = 0 or -ve

Error #	Interpertation
11	NFIX = 0 or -ve, or NFIX > NPOIN
12	NEXTIF = -ve, or NEXTIF > NPOIN
13	NLOAD = -ve, or NLOAD > NPOIN
14	INTEG $\neq$ 1 or 2
15	LENVEC or LENVEC1 is not adequate to accommodate the given problem.
Fatal Diagnostics in INPUT (reading of the card is interrupted)	
16	The last card read asked for an element or a node number or an element property that lay outside the permitted range.
Fatal Diagnostics in DMATRON	
17	An entry in NDF > NDFMAX, or = 0 or -ve
18	An entry in NDF like 3, 2, 0, 3 is not allowed. 3, 2, 3, 0 would be allowed.
19	All the entries in NDF of a given node are zeros.
20	The summation of entries in NDF of a given node > LVAB.
21	An entry in JPROP is outside the range permitted in the first card, or is negative or zero.
22	A node number in an element is -ve, or it exceeds NPOIN
23	An element has all the node numbers zero.
24	Node numbers of an element are given as e.g. (21, 13, 0, 19) instead of (21, 13, 19, 0).
25	Length of vector VEC is not enough even to do all the checks. Increase the dimension of VEC and LENVEC to the "associated number" given in the error message.

Error #	Interpertation
26	A node is repeated.
27	An entry in array NODFIX = 0 or -ve, or > NPOIN.
28	An entry in array KODFIX = 0.
29	You have fixed the same node twice or more.
30	An entry in array NOSTIF = 0 or -ve, or > NPION.
31	You have added a stiffness to the same node twice.
32	An entry in array NODLOD = 0 or -ve, or > NLOAD.
33	You have applied a point load to the same node twice.
34	Dimension of VEC is not enough to run the program. Increase the length of VEC and LENVEC to the "associated number" printed in the message.
Fatal Diagnostics in DFRONT (reading of the card is interrupted)	
35	Diagonal stiffness ÷ pivot is large enough to suggest that serious roundoff damage has occurred, or the pivot is negative.
36	The calculation has terminated, but a subsequent investigation suggests that the results are almost certainly nonsense because of roundoff errors.
41	NODFL*NEL $\leq$ NPOIN, or 20*NEL $\leq$ NPOIN
42	NODEL $\leq$ NDIM, or NODEL > 20
Non-Fatal Diagnostics in DNURSE	
43	NDFMAX > 6
44	NDIM $\neq$ 2 or 3
45	NSTRES > 2.*DIM
46	IPROP > 10
47	JPROP > 50
48	NFIX < NDIM
49	LENVEC $\leq$ 300, or $\geq$ 100,000

Error #	Interpertation
Non-Fatal Diagnostics in DMATRON	
50	Two nodes have identical coordinates.
51	An element has repeated node numbers.
52	A particular node number does not appear at all.
53	You have specified the coordinate of this node, then you have not use it.
54	You have not use a node, and yet you have fixed it.
55	Or, you have not used a node, and yet you have added a stiffness there.
56	Or, you have not used a node, and yet you have added a point load there.
Non-Fatal Diagnostics in DFRONT	
57	Diagonal stiffness ÷ pivot is large, suggesting that some roundoff has been done, probably not much.
58	Frontwidth is prematurely zero. You have not put two independent structures in one run. Did you mean to do this?
59	Your energy is zero. Did you intend to put in an unloaded structure.
60	My guess is that you have a heated but unloaded structure. I trust you know what you are doing.
61	The final level of roundoff leaves me a little suspicious of the value of your results.

æ

Appendix E

Data Input for p–feda

DESCRIPTION	VARIABLE	COLUMN
1st line (Format 20I4)		
Total number of nodes.	NPOIN	1-4
Total number of nodes along global front.	NGLOBE	5-8
Total number of elements.	NEL	9-12
Maximum number of nodes per element.	NODEL	13-16
Maximum degrees of freedom per node.	NDFMAX	17-20
Maximun number of solutions or iterations in eigen analysis.	NCASE	21-24
Number of dimensions (2 or 3).	NDIM	25-28
Number of stresses = size of matrix D.	NSTRES	29-32
Number of element types.	NTYPE	33-36
Gauss rule used.	NGAUS	37-40
Number of properties, i.e., E, A, etc.	IPROP	41-44
Number of sets of properties.	JPROP	45-48

DESCRIPTION	VARIABLE	COLUMN
Number of nodes with prescribed values.	NFIX	49-52
Number of nodes with additional stiffnesses.	NEXTIF	53-56
Number of nodes with additional loads.	NLOAD	57-60
Maximum number of variables per element.	LVAB	61-64
Number of right hand sides or eigenpairs.	NRHS	65-68
Number of nodes in vibration mode problem.	NPOIN1	69-72
Printing option.	NPTION	73-76
Solution option.	KODSOL	77-80

2nd line (Format 20I4)

Number of sets of element loads.	JLOAD	1-4
Number of fronts/domains.	NFRONT	5-8
Maximum number of nodes per global front.	GNODEL	9-12

3rd line (Format E10.2)

Tolerance level in eigen analysis.	TOL2	1-10

4th set of lines (Format 20I4)

[Enter values for each set of properties]

Number of degrees of freedom per node.	NDF(N,J)	1-4, etc.

where N = 1,NODEL

J = 1,JPROP

- Total number of lines = JPROP

DESCRIPTION	VARIABLE	COLUMN
5th set of lines (Format 20I4)		
[Enter values for each element]		
Element number.	NE	1-4
Type of element.	LTYPE(NE)	5-8
Property type of element.	LPROP(NE)	9-12
Load type.	LLOAD(NE)	13-16
Domain/Front the element is located on.	LDMAIN(NE)	17-20
Element node numbers.	LNODS(N,NE)	21-24, etc.
where N = 1,NODEL		
NE = 1,NEL		
- Total number of lines = NEL		
6th set of lines (Format I5,3F11.4)		
[Enter values for each node]		
Node number.	NODES	1-5
Coordinates of node (X,Y,Z).	COORD(N,I)	6-16, etc.
where I = 1,NDIM		
N = 1,NPOIN		
- Total number of lines = NPOIN		

DESCRIPTION	VARIABLE	COLUMN
7th set of lines (Format 2I6,6F11.7)		
[Enter values for each node that is fixed]		
Node number that is fixed.	NODFIX(N)	1-6
Code number listing which degree of freedom is prescribed at each node.	KODFIX(N)	7-12
Prescribed displacements at the nodes listed in NODFIX.	VFIX(N,I)	13-23, etc.
where I = 1,NDFMAX		
N = 1,NFIX		
- Total number of lines = NFIX		
8th set of lines (Format I5,6E12.5)		
[Enter a value for each node with additional stiffness]		
Node number with additional stiffness.	NOSTIF(N)	1-5
Additional stiffnesses at the nodes.	VSTIF(N,I)	6-17, etc.
where I = 1,NDFMAX		
N = 1,NEXTIF		
- Total number of lines = NEXTIF		
9th set of lines (Format I5,6E12.5)		
[Enter values for each node with additional loads]		
Node number with additional loads.	NODLOD(N)	1-5
Additional loads at the nodes.	VLOAD(N,I,J)	6-17, etc.
where I = 1,NDFMAX		
N = 1,NLOAD		
J = 1,NRHS		
- Total number of lines = NLOAD Repeated NRHS times.		

DESCRIPTION	VARIABLE	COLUMN
10th set of lines (Format I5,6E12.5)		
[Enter values for each set of properties]		
Property set number.	N	1-5
Enter property values for the specific element you are using.	VPROP(I,N)	6-16, etc.
where I = 1,IPROP		
J = 1,JPROP		
- Total number of values = IPROP		
- Total number of lines = JPROP		
11th set of lines (Format I5,6E12.5)		
[Enter values for each node]		
Node number.	N	1-5
Load set.	SLOAD(I,N)	6-17, etc.
where NR = 1,NRHS		
I = 1,JLOAD		
- Total number of lines = JLOAD		
12th set of lines (Format 20I4)		
[Enter each node on global front]		
Node number on global front.	NDMAIN(I)	1-4, etc.
where I = 1,NGLOBE		
- Total number of values = NGLOBE		
13th set of lines (Format 20I4)		
[Enter each node on global front]		

Appendix F

Output of p–feda

An output file conntaining the input data and eigenpairs is presented for a two domain 64 element plate as shown in Figure F.1

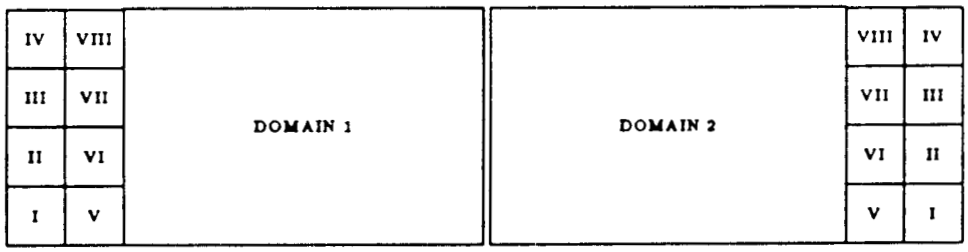

Figure F.1: Two Domain 64 Element Plate with 13 dofs on the Global Front

```
**********************************************
*                                            *
*        //L  F E D A   V. 88-3              *
*                  BY                        *
*      F. FRED AKL  AND  M. MOREL            *
**********************************************
```

P R O B L E M D E S C R I P T I O N

```
MAXIMUM NODE NUMBER = NPOIN =                                  233
NUMBER OF NODES ALONG BOUNDRY = NGLOBE =                         9
NUMBER OF ELEMENTS = NEL =                                      64
MAXIMUM NODES PER ELEMENT = NODEL =                              8
MAXIMUM DEGREES OF FREEDOM PER NODE = NDFMAX =                   3
NUMBER OF SOLUTIONS REQUIRED = NCASE =                         100
NUMBER OF DIMENSIONS, 2 OR 3, =                                  3
NUMBER OF STRESSES = SIZE OF MATRIX D = NSTRES                   3
NUMBER OF ELEMENT TYPES = NTYPE =                                1
GAUSS RULE USED = NGAUS =                                        2 POINTS
NUMBER OF PROPERTIES: E , A , I , RHO , THICK = IPROP =          5
NUMBER OF SETS OF PROPERTIES AVAILABLE = JPROP =                 1
NUMBER OF NODES WITH SOME FIXED VALUES = NFIX =                 80
NUMBER OF NODES WITH ADDITIONAL STIFFNESSES = NEXTIF =           0
NUMBER OF NODES WITH ADDITIONAL LOADS = NLOAD =                  0
MAXIMUM VARIABLES PER ELEMENT = LVAB =                          16
NUMBER OF RIGHT HAND SIDES = NRHS =                              8
NUMBER OF INTEGER WORDS PER FLOATING WORD = INTEG =              1
LENGTH OF VECTOR OF FLOATING WORDS = LENVEC =               200000
LENGTH OF VECTOR OF FLOATING WORDS =  LNVEC1 =              100000
NUMBER OF SETS OF ELEMENT LOADS = JLOAD =                        0
NUMBER OF NODES IN VIBRATION MODE PROFILES = NPOIN1 =            0
PRINTING OPTION =  NPTION =                                      1
SOLUTION CODE = KODSOL =                                        -3
KODSOL = 0 IE. STATIC SOLUTION
KODSOL = -1 IE. MODAL DYNAMIC ANALYSIS
KODSOL = -2 IE. DIRECT DYNAMIC ANALYSIS
KODSOL = -3 IE. SUBSPACE EIGEN ANALYSIS
TOLERENCE LEVEL IN EIGEN ANALYSIS = TOL2 =                  0.10000E-06
NUMBER OF FRONTS = NFRONT =                                      2
MAXIMUM NODES PER GLOBAL FRONT = GNODEL =                        9
DEGREES OF FREEDOM AT NODES OF ELEMENT OF TYPE  1 = NDF = 3 1 3 1 3 1 3 1
ELEMENT TYPE PROPERTY LOAD DOMAINNODE NUMBERS = LNODS
NUMBER
```

NUMBER												
1	3	1	0	1	1	10	15	16	17	11	3	2
2	3	1	0	1	3	11	17	18	19	12	5	4
3	3	1	0	1	5	12	19	20	21	13	7	6
4	3	1	0	1	7	13	21	22	23	14	9	8
5	3	1	0	1	15	24	29	30	31	25	17	16
6	3	1	0	1	17	25	31	32	33	26	19	18
7	3	1	0	1	19	26	33	34	35	27	21	20
8	3	1	0	1	21	27	35	36	37	28	23	22
9	3	1	0	1	29	38	43	44	45	39	31	30
10	3	1	0	1	31	39	45	46	47	40	33	32
11	3	1	0	1	33	40	47	48	49	41	35	34
12	3	1	0	1	35	41	49	50	51	42	37	36
13	3	1	0	1	43	52	57	58	59	53	45	44
14	3	1	0	1	45	53	59	60	61	54	47	46

15	3	1	0	1	47	54	61	62	63	55	49	48
16	3	1	0	1	49	55	63	64	65	56	51	50
17	3	1	0	1	57	66	71	72	73	67	59	58
18	3	1	0	1	59	67	73	74	75	68	61	60
19	3	1	0	1	61	68	75	76	77	69	63	62
20	3	1	0	1	63	69	77	78	79	70	65	64
21	3	1	0	1	71	80	85	86	87	81	73	72
22	3	1	0	1	73	81	87	88	89	82	75	74
23	3	1	0	1	75	82	89	90	91	83	77	76
24	3	1	0	1	77	83	91	92	93	84	79	78
25	3	1	0	1	85	94	99	100	101	95	87	86
26	3	1	0	1	87	95	101	102	103	96	89	88
27	3	1	0	1	89	96	103	104	105	97	91	90
28	3	1	0	1	91	97	105	106	107	98	93	92
29	3	1	0	1	99	108	113	114	115	109	101	100
30	3	1	0	1	101	109	115	116	117	110	103	102
31	3	1	0	1	103	110	117	118	119	111	105	104
32	3	1	0	1	105	111	119	120	121	112	107	106
33	3	1	0	2	211	220	225	226	227	221	213	212
34	3	1	0	2	213	221	227	228	229	222	215	214
35	3	1	0	2	215	222	229	230	231	223	217	216
36	3	1	0	2	217	223	231	232	233	224	219	218
37	3	1	0	2	197	206	211	212	213	207	199	198
38	3	1	0	2	199	207	213	214	215	208	201	200
39	3	1	0	2	201	208	215	216	217	209	203	202
40	3	1	0	2	203	209	217	218	219	210	205	204
41	3	1	0	2	183	192	197	198	199	193	185	184
42	3	1	0	2	185	193	199	200	201	194	187	186
43	3	1	0	2	187	194	201	202	203	195	189	188
44	3	1	0	2	189	195	203	204	205	196	191	190
45	3	1	0	2	169	178	183	184	185	179	171	170
46	3	1	0	2	171	179	185	186	187	180	173	172
47	3	1	0	2	173	180	187	188	189	181	175	174
48	3	1	0	2	175	181	189	190	191	182	177	176
49	3	1	0	2	155	164	169	170	171	165	157	156
50	3	1	0	2	157	165	171	172	173	166	159	158
51	3	1	0	2	159	166	173	174	175	167	161	160
52	3	1	0	2	161	167	175	176	177	168	163	162
53	3	1	0	2	141	150	155	156	157	151	143	142
54	3	1	0	2	143	151	157	158	159	152	145	144
55	3	1	0	2	145	152	159	160	161	153	147	146
56	3	1	0	2	147	153	161	162	163	154	149	148
57	3	1	0	2	127	136	141	142	143	137	129	128
58	3	1	0	2	129	137	143	144	145	138	131	130
59	3	1	0	2	131	138	145	146	147	139	133	132
60	3	1	0	2	133	139	147	148	149	140	135	134
61	3	1	0	2	113	122	127	128	129	123	115	114
62	3	1	0	2	115	123	129	130	131	124	117	116
63	3	1	0	2	117	124	131	132	133	125	119	118
64	3	1	0	2	119	125	133	134	135	126	121	120

NODE	X	Y	Z
1	0.0000	0.0000	1.0000
2	0.0000	1.0000	1.0000
3	0.0000	2.0000	1.0000
4	0.0000	3.0000	1.0000
5	0.0000	4.0000	1.0000
6	0.0000	5.0000	1.0000
7	0.0000	6.0000	1.0000
8	0.0000	7.0000	1.0000
9	0.0000	8.0000	1.0000

10	1.0000	0.0000	1.0000
11	1.0000	2.0000	1.0000
12	1.0000	4.0000	1.0000
13	1.0000	6.0000	1.0000
14	1.0000	8.0000	1.0000
15	2.0000	0.0000	1.0000
16	2.0000	1.0000	1.0000
17	2.0000	2.0000	1.0000
18	2.0000	3.0000	1.0000
19	2.0000	4.0000	1.0000
20	2.0000	5.0000	1.0000
21	2.0000	6.0000	1.0000
22	2.0000	7.0000	1.0000
23	2.0000	8.0000	1.0000
24	3.0000	0.0000	1.0000
25	3.0000	2.0000	1.0000
26	3.0000	4.0000	1.0000
27	3.0000	6.0000	1.0000
28	3.0000	8.0000	1.0000
29	4.0000	0.0000	1.0000
30	4.0000	1.0000	1.0000
31	4.0000	2.0000	1.0000
32	4.0000	3.0000	1.0000
33	4.0000	4.0000	1.0000
34	4.0000	5.0000	1.0000
35	4.0000	6.0000	1.0000
36	4.0000	7.0000	1.0000
37	4.0000	8.0000	1.0000
38	5.0000	0.0000	1.0000
39	5.0000	2.0000	1.0000
40	5.0000	4.0000	1.0000
41	5.0000	6.0000	1.0000
42	5.0000	8.0000	1.0000
43	6.0000	0.0000	1.0000
44	6.0000	1.0000	1.0000
45	6.0000	2.0000	1.0000
46	6.0000	3.0000	1.0000
47	6.0000	4.0000	1.0000
48	6.0000	5.0000	1.0000
49	6.0000	6.0000	1.0000
50	6.0000	7.0000	1.0000
51	6.0000	8.0000	1.0000
52	7.0000	0.0000	1.0000
53	7.0000	2.0000	1.0000
54	7.0000	4.0000	1.0000
55	7.0000	6.0000	1.0000
56	7.0000	8.0000	1.0000
57	8.0000	0.0000	1.0000
58	8.0000	1.0000	1.0000
59	8.0000	2.0000	1.0000
60	8.0000	3.0000	1.0000
61	8.0000	4.0000	1.0000
62	8.0000	5.0000	1.0000
63	8.0000	6.0000	1.0000
64	8.0000	7.0000	1.0000
65	8.0000	8.0000	1.0000
66	9.0000	0.0000	1.0000
67	9.0000	2.0000	1.0000
68	9.0000	4.0000	1.0000
69	9.0000	6.0000	1.0000

70	9.0000	8.0000	1.0000
71	10.0000	0.0000	1.0000
72	10.0000	1.0000	1.0000
73	10.0000	2.0000	1.0000
74	10.0000	3.0000	1.0000
75	10.0000	4.0000	1.0000
76	10.0000	5.0000	1.0000
77	10.0000	6.0000	1.0000
78	10.0000	7.0000	1.0000
79	10.0000	8.0000	1.0000
80	11.0000	0.0000	1.0000
81	11.0000	2.0000	1.0000
82	11.0000	4.0000	1.0000
83	11.0000	6.0000	1.0000
84	11.0000	8.0000	1.0000
85	12.0000	0.0000	1.0000
86	12.0000	1.0000	1.0000
87	12.0000	2.0000	1.0000
88	12.0000	3.0000	1.0000
89	12.0000	4.0000	1.0000
90	12.0000	5.0000	1.0000
91	12.0000	6.0000	1.0000
92	12.0000	7.0000	1.0000
93	12.0000	8.0000	1.0000
94	13.0000	0.0000	1.0000
95	13.0000	2.0000	1.0000
96	13.0000	4.0000	1.0000
97	13.0000	6.0000	1.0000
98	13.0000	8.0000	1.0000
99	14.0000	0.0000	1.0000
100	14.0000	1.0000	1.0000
101	14.0000	2.0000	1.0000
102	14.0000	3.0000	1.0000
103	14.0000	4.0000	1.0000
104	14.0000	5.0000	1.0000
105	14.0000	6.0000	1.0000
106	14.0000	7.0000	1.0000
107	14.0000	8.0000	1.0000
108	15.0000	0.0000	1.0000
109	15.0000	2.0000	1.0000
110	15.0000	4.0000	1.0000
111	15.0000	6.0000	1.0000
112	15.0000	8.0000	1.0000
113	16.0000	0.0000	1.0000
114	16.0000	1.0000	1.0000
115	16.0000	2.0000	1.0000
116	16.0000	3.0000	1.0000
117	16.0000	4.0000	1.0000
118	16.0000	5.0000	1.0000
119	16.0000	6.0000	1.0000
120	16.0000	7.0000	1.0000
121	16.0000	8.0000	1.0000
122	17.0000	0.0000	1.0000
123	17.0000	2.0000	1.0000
124	17.0000	4.0000	1.0000
125	17.0000	6.0000	1.0000
126	17.0000	8.0000	1.0000
127	18.0000	0.0000	1.0000
128	18.0000	1.0000	1.0000
129	18.0000	2.0000	1.0000

130	18.0000	3.0000	1.0000
131	18.0000	4.0000	1.0000
132	18.0000	5.0000	1.0000
133	18.0000	6.0000	1.0000
134	18.0000	7.0000	1.0000
135	18.0000	8.0000	1.0000
136	19.0000	0.0000	1.0000
137	19.0000	2.0000	1.0000
138	19.0000	4.0000	1.0000
139	19.0000	6.0000	1.0000
140	19.0000	8.0000	1.0000
141	20.0000	0.0000	1.0000
142	20.0000	1.0000	1.0000
143	20.0000	2.0000	1.0000
144	20.0000	3.0000	1.0000
145	20.0000	4.0000	1.0000
146	20.0000	5.0000	1.0000
147	20.0000	6.0000	1.0000
148	20.0000	7.0000	1.0000
149	20.0000	8.0000	1.0000
150	21.0000	0.0000	1.0000
151	21.0000	2.0000	1.0000
152	21.0000	4.0000	1.0000
153	21.0000	6.0000	1.0000
154	21.0000	8.0000	1.0000
155	22.0000	0.0000	1.0000
156	22.0000	1.0000	1.0000
157	22.0000	2.0000	1.0000
158	22.0000	3.0000	1.0000
159	22.0000	4.0000	1.0000
160	22.0000	5.0000	1.0000
161	22.0000	6.0000	1.0000
162	22.0000	7.0000	1.0000
163	22.0000	8.0000	1.0000
164	23.0000	0.0000	1.0000
165	23.0000	2.0000	1.0000
166	23.0000	4.0000	1.0000
167	23.0000	6.0000	1.0000
168	23.0000	8.0000	1.0000
169	24.0000	0.0000	1.0000
170	24.0000	1.0000	1.0000
171	24.0000	2.0000	1.0000
172	24.0000	3.0000	1.0000
173	24.0000	4.0000	1.0000
174	24.0000	5.0000	1.0000
175	24.0000	6.0000	1.0000
176	24.0000	7.0000	1.0000
177	24.0000	8.0000	1.0000
178	25.0000	0.0000	1.0000
179	25.0000	2.0000	1.0000
180	25.0000	4.0000	1.0000
181	25.0000	6.0000	1.0000
182	25.0000	8.0000	1.0000
183	26.0000	0.0000	1.0000
184	26.0000	1.0000	1.0000
185	26.0000	2.0000	1.0000
186	26.0000	3.0000	1.0000
187	26.0000	4.0000	1.0000
188	26.0000	5.0000	1.0000
189	26.0000	6.0000	1.0000

190	26.0000	7.0000	1.0000
191	26.0000	8.0000	1.0000
192	27.0000	0.0000	1.0000
193	27.0000	2.0000	1.0000
194	27.0000	4.0000	1.0000
195	27.0000	6.0000	1.0000
196	27.0000	8.0000	1.0000
197	28.0000	0.0000	1.0000
198	28.0000	1.0000	1.0000
199	28.0000	2.0000	1.0000
200	28.0000	3.0000	1.0000
201	28.0000	4.0000	1.0000
202	28.0000	5.0000	1.0000
203	28.0000	6.0000	1.0000
204	28.0000	7.0000	1.0000
205	28.0000	8.0000	1.0000
206	29.0000	0.0000	1.0000
207	29.0000	2.0000	1.0000
208	29.0000	4.0000	1.0000
209	29.0000	6.0000	1.0000
210	29.0000	8.0000	1.0000
211	30.0000	0.0000	1.0000
212	30.0000	1.0000	1.0000
213	30.0000	2.0000	1.0000
214	30.0000	3.0000	1.0000
215	30.0000	4.0000	1.0000
216	30.0000	5.0000	1.0000
217	30.0000	6.0000	1.0000
218	30.0000	7.0000	1.0000
219	30.0000	8.0000	1.0000
220	31.0000	0.0000	1.0000
221	31.0000	2.0000	1.0000
222	31.0000	4.0000	1.0000
223	31.0000	6.0000	1.0000
224	31.0000	8.0000	1.0000
225	32.0000	0.0000	1.0000
226	32.0000	1.0000	1.0000
227	32.0000	2.0000	1.0000
228	32.0000	3.0000	1.0000
229	32.0000	4.0000	1.0000
230	32.0000	5.0000	1.0000
231	32.0000	6.0000	1.0000
232	32.0000	7.0000	1.0000
233	32.0000	8.0000	1.0000

NODE = NODFIX	FIXING CODE = KODFIX	FIXED VALUES = VFIX		
1	111	0.0000000	0.0000000	0.0000000
2	1	0.0000000	0.0000000	0.0000000
3	111	0.0000000	0.0000000	0.0000000
4	1	0.0000000	0.0000000	0.0000000
5	111	0.0000000	0.0000000	0.0000000
6	1	0.0000000	0.0000000	0.0000000
7	111	0.0000000	0.0000000	0.0000000
8	1	0.0000000	0.0000000	0.0000000
9	111	0.0000000	0.0000000	0.0000000
225	111	0.0000000	0.0000000	0.0000000
226	1	0.0000000	0.0000000	0.0000000
227	111	0.0000000	0.0000000	0.0000000
228	1	0.0000000	0.0000000	0.0000000
229	111	0.0000000	0.0000000	0.0000000

230	1	0.0000000	0.0000000	0.0000000
231	111	0.0000000	0.0000000	0.0000000
232	1	0.0000000	0.0000000	0.0000000
233	111	0.0000000	0.0000000	0.0000000
10	1	0.0000000	0.0000000	0.0000000
15	111	0.0000000	0.0000000	0.0000000
24	1	0.0000000	0.0000000	0.0000000
29	111	0.0000000	0.0000000	0.0000000
38	1	0.0000000	0.0000000	0.0000000
43	111	0.0000000	0.0000000	0.0000000
52	1	0.0000000	0.0000000	0.0000000
57	111	0.0000000	0.0000000	0.0000000
66	1	0.0000000	0.0000000	0.0000000
71	111	0.0000000	0.0000000	0.0000000
80	1	0.0000000	0.0000000	0.0000000
85	111	0.0000000	0.0000000	0.0000000
94	1	0.0000000	0.0000000	0.0000000
99	111	0.0000000	0.0000000	0.0000000
108	1	0.0000000	0.0000000	0.0000000
113	111	0.0000000	0.0000000	0.0000000
122	1	0.0000000	0.0000000	0.0000000
127	111	0.0000000	0.0000000	0.0000000
136	1	0.0000000	0.0000000	0.0000000
141	111	0.0000000	0.0000000	0.0000000
150	1	0.0000000	0.0000000	0.0000000
155	111	0.0000000	0.0000000	0.0000000
164	1	0.0000000	0.0000000	0.0000000
169	111	0.0000000	0.0000000	0.0000000
178	1	0.0000000	0.0000000	0.0000000
183	111	0.0000000	0.0000000	0.0000000
192	1	0.0000000	0.0000000	0.0000000
197	111	0.0000000	0.0000000	0.0000000
206	1	0.0000000	0.0000000	0.0000000
211	111	0.0000000	0.0000000	0.0000000
220	1	0.0000000	0.0000000	0.0000000
14	1	0.0000000	0.0000000	0.0000000
23	111	0.0000000	0.0000000	0.0000000
28	1	0.0000000	0.0000000	0.0000000
37	111	0.0000000	0.0000000	0.0000000
42	1	0.0000000	0.0000000	0.0000000
51	111	0.0000000	0.0000000	0.0000000
56	1	0.0000000	0.0000000	0.0000000
65	111	0.0000000	0.0000000	0.0000000
70	1	0.0000000	0.0000000	0.0000000
79	111	0.0000000	0.0000000	0.0000000
84	1	0.0000000	0.0000000	0.0000000
93	111	0.0000000	0.0000000	0.0000000
98	1	0.0000000	0.0000000	0.0000000
107	111	0.0000000	0.0000000	0.0000000
112	1	0.0000000	0.0000000	0.0000000
121	111	0.0000000	0.0000000	0.0000000
126	1	0.0000000	0.0000000	0.0000000
135	111	0.0000000	0.0000000	0.0000000
140	1	0.0000000	0.0000000	0.0000000
149	111	0.0000000	0.0000000	0.0000000
154	1	0.0000000	0.0000000	0.0000000
163	111	0.0000000	0.0000000	0.0000000
168	1	0.0000000	0.0000000	0.0000000
177	111	0.0000000	0.0000000	0.0000000
182	1	0.0000000	0.0000000	0.0000000

```
 191        111         0.0000000  0.0000000  0.0000000
 196          1         0.0000000  0.0000000  0.0000000
 205        111         0.0000000  0.0000000  0.0000000
 210          1         0.0000000  0.0000000  0.0000000
 219        111         0.0000000  0.0000000  0.0000000
 224          1         0.0000000  0.0000000  0.0000000
NUMBER      ELEMENT PROPERTIES = VPROP
   1       0.100000E+01 0.100000E+01 0.300000E+00 0.800000E+01 0.100000E+01


LIST OF NODES LOCATED ALONG GLOBAL FRONT
 113 114 115 116 117 118 119 120 121
   3   1   3   1   3   1   3   1   3
TOL2 HAS BEEN ACHIEVED
CURRENT TOLERENCE LEVELS
 -0.4739E-14  0.0000E+00  0.0000E+00 -0.1651E-13
  0.1626E-13 -0.1164E-13  0.4111E-14  0.7575E-07

NUMBER OF SUBSPACE ITERATIONS =    23

EIGENVALUES ARE
  0.1171E-01  0.1306E-01  0.1569E-01  0.2017E-01
  0.2731E-01  0.3814E-01  0.5401E-01  0.7662E-01
```

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR-185166		

4. Title and Subtitle	5. Report Date
Parallel Eigenanalysis of Finite Element Models in a Completely Connected Architecture	November 1989
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
F.A. Akl and M.R. Morel	None
	10. Work Unit No.
	505-63-1B

9. Performing Organization Name and Address	11. Contract or Grant No.
Ohio University Department of Civil Engineering Athens, Ohio	
	13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address	Contractor Report Final
National Aeronautics and Space Administration Lewis Research Center Cleveland, Ohio 44135-3191	14. Sponsoring Agency Code

15. Supplementary Notes

Project Manager, Louis J. Kiraly, Structures Division, NASA Lewis Research Center.

16. Abstract

This report presents a parallel algorithm for the solution of the generalized eigenproblem in linear elastic finite element analysis, $[K][\Phi] = [M][\Phi][\Omega]$, where: $[K]$ and $[M]$ are of order N, and $[\Omega]$ is order of q. The concurrent solution of the eigenproblem is based on the multifrontal/modified subspace method and is achieved in a completely connected parallel architecture in which each processor is allowed to communicate with all other processors. The algorithm has been successfully implemented on a tightly coupled multiple-instruction multiple-data parallel processing machine, Cray X-MP. A finite element model is divided into m domains each of which is assumed to process n elements. Each domain is then assigned to a processor or to a logical processor (task) if the number of domains exceeds the number of physical processors. The macrotasking library routines are used in mapping each domain to a user task. Computational speed-up and efficiency are used to determine the effectiveness of the algorithm. The effect of the number of domains, the number of degrees-of-freedom located along the global fronts and the dimension of the subspace on the performance of the algorithm are investigated. A parallel finite element dynamic analysis program, "p-feda", is documented and the performance of its subroutines in parallel environment is analyzed.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Parallel computers; Eigenvalues; Eigenvectors; Subspace; Multifront; Domains; Multitasking; Finite elements	Unclassified – Unlimited Subject Category 39

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No of pages	22. Price*
Unclassified	Unclassified	112	A06

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