DISTRIBUTED FINITE ELEMENT ANALYSIS

Dissertation

Submitted in partial fulfillment of the requirement
For the degree of
Master of Technology (CIVIL)
(Computer Aided Structural Analysis and Design)
NIRMA UNIVERSITY OF SCIENCE AND TECHNOLOGY

By
VIKAS P. SAXENA
(03MCL14)

Guide
PROF. P. V. PATEL

Civil Engineering Department
Ahmedabad - 382 481

May 2005

CERTIFICATE

This is to certify that the Major Project entitled "**Distributed Finite Element Analysis**" submitted by **Mr. Vikas P Saxena (03MCL14)**, towards the partial fulfillment of the requirements for the award of degree of **Master of Technology (CIVIL)** in field of **Computer Aided Structural Analysis and Design (CASAD)** of Nirma University of Science and Technology is the record of work carried out by him under my supervision and guidance. The work submitted has in my opinion reached a level required for being accepted for examination. The results embodied in this dissertation, to the best of my knowledge have not been submitted to any other university or institution for award of any degree or diploma.

Prof. P. V. Patel
Guide,
Asst. Professor,
Civil Engineering Department,
Institute of Technology,
Nirma University
Ahmedabad

Prof G. N. Patel
Head,
Civil Engineering Department
Institute of Technology
Nirma University
Ahmedabad

Dr. H. V. Trivedi
Director,
Institute of Technology
Nirma University
Ahmedabad

ACKNOWLEDGEMENT

It gives me great pleasure in expressing sincere thanks and profound gratitude to **Yogesh T. Vani**, Senior Consulting Engineer (Visiting Faculty at CEPT) Ahmedabad for his invaluable guidance and encouragement throughout this Major project. I heartily thank him for his suggestions and for the clarity of the concepts of the topic that helped me a lot during this study.

I extend my special thanks to **Prof. G.N. Patel**, Professor, Department of Civil Engineering and **Prof. P.H. Shah**, Head, Department of Civil Engineering, Institute of Technology, Nirma University, Ahmedabad for his continuous kind words of encouragement and motivation throughout the Major Project. I am heartily thankful to **Prof. C. H. Shah**, Structural consultant for his suggestions to improve quality of work. I am also thankful to **Dr. H. V. Trivedi**, Director, Institute of Technology for his kind support in all respect during my study.

I am thankful to **Prof. P.V. Patel**, **Prof. U.V. Dave**, **Prof. S. P. Purohit**, **S. Pandya** and all faculty members of Institute of Technology, Nirma University, Ahmedabad for their special attention and suggestions towards the project works.

I will be obliged to **Mr. Rushikesh Trivedi**, Structural engineer, VMS Consultants for his time and the help he offered during project.

The blessings of **God and my family members** makes the way for completion of major project-1. I am very much grateful to them.

I am also thankful to all my friends especially **Mr. Anil Wadile** for his suggestions and help, he gave throughout my M Tech Course.

Sonar Priyadarshini V.
Roll No.04MCL018

ABSTRACT

A lot of advancement has taken place in computer generation in last five decades. And this advancement has led to the development of computers having processing speeds of teraflop range and memory of terabyte range.

As the computing power available increased, attempt was made to solve more and more difficult problems. Some of the problems related to field such as artificial intelligence, numerical analysis, Quantum chromo dynamics, Climate modeling, Fluid turbulence, Vehicle dynamics, Ocean circulation etc, require such a high computational power and also memory requirement that present day's computers may found slower to solve these problems. These limitations have led to develop some other solutions. High performance computing is result of such attempts.

Various possible ways are available for high performance computing i.e. supercomputers, massive parallel machines, cluster computing etc. But as these options are costly from initial cost and maintenance point of view, an alternative option of Parallel Processing on network of computers, also known as distributed computing can been used. As distributed computing is carried out on network of computers, no special hardware is needed and also no special means of maintenance is needed. All these aspects make distributed computing a favorable option for high performance computing. In present study, distributed computing is carried out using WebDedip environment, developed using JAVA technology based on client server approach.

As far as structural field is concerned, problems include difficult geometry, boundary conditions and loading conditions, which makes the problem complex. Also, as the size of problem goes on increasing, requirement of computational power and storage memory goes on increasing. So, present single computers may found slower to tackle such problems. Hence analysis of such problem becomes a tedious and time consuming task. In such cases some other alternative is needed which can serve better in such conditions. High performance computing is one of the solutions for such problems.

In present study an attempt has been made to implement distributed computing in field of structural engineering.

Finite Element Analysis method has been used for analysis. Substructure analysis technique has been incorporated to implement analysis process on distributed computing. To observe the advantage of high performance computing, four problems have been solved using sequential and parallel processing. Of these problems, one is the plane stress problem, two are plate bending problems and one laminated plate problem.

First chapter includes the introduction part. Topics such as need of high performance computing, efforts made in India as well as in world for high performance computing and object of study are included in this chapter.

Second chapter includes the literature survey. Scope of work is also included in the same chapter.

Third chapter includes the issues related to parallel processing. These issues are computer architecture, parallelism in sequential as well as in multiprocessor computers, dynamic load balancing, operating system characteristics, parallel processing models, performance measurement and software tools for parallel processing.

Forth chapter deals with the aspects related to WebDedip environment. Details of various components to be installed on server as well as client machine, minimum software and hardware requirement for distributed computing and procedure of problem implementation on parallel computers are some of the topics, discussed in this chapter.

Fifth chapter includes the distributed plane stress analysis problem. The problem is a square plate having hole in the center and subjected to uniform tensile force on both sides. For analysis of plate, CST element is used. Plate is divided into 27504 number of elements. So the problem is having total 13990 number of nodes (i.e. 27980 DOFs). To observe the effect of distributed computing, plate analysis is carried out using 3, 4, 5 and 6 computers.

Sixth chapter deals with the distributed plate bending problem. This chapter includes two problems. For analysis of both problems, eight nodded isoparametric plate bending element is used. Of two problems, first problem is the annular plate subjected to uniform lateral loading. Plate is divided into 1800 elements. So problem is having total 5725 number of nodes (i.e. 17175 DOFs). Plate analysis is carried out by diving it into 3, 4, and 6 number of

substructures. The second problem is the simply supported skew plate subjected to uniform pressure. Plate is divided into 3000 number of elements. So, total number of nodes are 9221 and DOFs are 27663. To implement the problem on parallel computers, plate is divided into 3, 4, 5 and 6 number of substructures.

Seventh chapter covers the distributed laminated plate analysis. The problem is a square plate having four laminates of same thickness and subjected to sinusoidal loading. For analysis of laminated composite plate, eight nodded isoparametric element is used. The plate is divided into 1600 elements. So problem is having total 4961 number of nodes and 29766 DOFs. To observer the effect of distributed computing, the plate analysis is carried out using 3, 4 and 6 number of computers.

Chapter eight includes the summary, conclusion and further scope of work.

INDEX

INDEX OF FIGURES

INDEX OF TABLES

CHAPTER 1 INTRODUCTION

1.1 GENERAL

As the computing power available increases, the quest for more and more power also keeps increasing. There are always applications waiting for more powerful machines. Much of the effort in meeting this requirement has so far been focused on by making the existing computer architecture faster and faster. Memory access delays have been cut down using faster memories and cache architectures [21].

Analysis of large size complex structures such as multistoried buildings, long span bridges, tall towers and domes etc. involves formulation and solution of large number of equations. Either matrix stiffness method or finite element method can be used to solve such complex problems. The analysis of such large structures, involving thousands of unknowns, may be expedited by subdividing it into smaller parts known as substructures. In substructure technique, each substructure is analyzed separately and the results are combined to yield the displacements and stresses in actual structure. The use of substructure technique, depending upon the size of each substructure and number of substructures, may result in considerable saving of storage and computational time. The analysis time can further be reduced by application of distributed computing technique [27].

Finite Element Analysis now stands as the most acceptable numerical solution procedure for engineering problems in diverse areas such as solid mechanics, fluid mechanics, heat transfer, aerodynamics, manufacturing process, smart materials, and structures etc [19]. With fundamental limits existing for the performance of a processor, single processor system even with its present configuration and future prediction can never suffice to meet the computing demands of FEA. Hence high performance computing with low price tags, sustained performance, and cheap maintenance and upgradation costs is needed. Distributed computing is different but related approach to obtain faster machine [26].

1.2 NEED FOR DISTRIBUTED COMPUTING

The basic need for the distributed computing is the need of power. Computational requirements have always been on the increase. As the processor become more powerful, one is interested in getting the machine to tackle more complex problems.

One method of increasing the speed of computers is to use faster semiconductor components to build units of a computer. For instance low speed personal computers use integrated circuit based on Silicon semiconductor while Gallium Arsenide semiconductor devices which are faster, are used in supercomputer. Higher speed components cost more and normally dissipate more heat. So the increase in speed of electronic components is limited by physical constraints. Another method to increase the speed of computation is to design the computer so that the different units of the computer work simultaneously [23]. Such requirements also give rise to the development of distributed computing techniques.

Although the performance of single processor has been steadily increasing over the years, the only way to build the next generation Tflops architecture supercomputer seems to be through parallel processing technology. Even with today's workstation class high performance processors exceeding 100 Mflops, thousand of processors are required to build a Tflop architecture machine.

On the computational side, artificial intelligence, numerical analysis etc are some of the areas, which can consume any amount of computing power that is available. Some of the very complex problems require such a high-level computation power, which is beyond the computing power available today. One of such problem is Quantum Chromo Dynamics, the study of sub atomic structure. This experiment requires a Cray super computer running 24 hours a day for 1500years, in order to get the required results. Some of such other problems, requiring very high level computing power, known as grand challenges and some of such challenges are Climate modeling, Fluid turbulence, Vehicle dynamics, Ocean circulation etc [29].

Analysis of complex structures such as multistoried buildings, long span roof or bridges, tall towers and offshore platform involves formulation and solution of large number of equations

since it has to take into account either one or a combination of the following features of the problem:

1.) Irregular geometry
2.) Real boundary conditions
3.) Stress, temperature and time dependent linear/non linear behavior of constituent materials including composites
4.) Space time dependent loads including random excitations
5.) Different types of behavior-linear, nonlinear, dynamic response, stability etc.

Inclusion of such conditions makes the analysis of structure more complex and time consuming. So to arrive at the accurate solution within short period of time, latest technique is needed, which along with the faster speed, should be less expensive and also having low up gradation and maintenance cost. Such requirements give rise to the need of distributed computing technique.

In past few years a number of parallel processing machines have been developed with various processor configurations and architectures. Parallel implementation requires parallel machine consisting of multiple processors connected with each other and sharing the same memory. As parallel processing hardware designed exclusively for dedicated parallel computing applications are more expensive, an inexpensive alternative is to develop the application software, which can run on network of workstations, known as distributed processing. This arrangement employs a number of computers physically linked to permit online computer-to-computer communication. Using this approach of distributed processing, independent workstations interconnected by network and message passing for communication of data between the computers can be transformed in to cost effective parallel computing resources. Message Passing Interface (MPI) and Parallel Virtual Machine (PVM) provide libraries for such communication. But use of these libraries requires some concepts of parallel programming like data communication, process forking, process joining, data synchronization etc. Furthermore it requires special debugging techniques for testing the parallel application.

1.3 PARALLEL COMPUTING EFFORTS IN WORLD

New computing systems cover a broad spectrum of machines ranging from high performance systems to the small portable, embedded computers and microprocessors. Description of some of such machines is discussed here.

Supersystems are a class of general-purpose computers designed for extremely high performance throughput. Although there is no universally accepted definition or classification of supersystems, three classes of supersystems have been identified based on the performance, memory system used and cost, namely: large supersystems, near supers and superservers. The architectural characteristics of some of the U.S. and Japanese supersystems [10] are summarized below in Table 1.1

Table1.1: US and JAPANESE super systems

System / model	Architectural configuration	Number of processors	Maximum main memory	Peak computational rate (Gflops)
CRAY – T90	Vector SMP	32	1 GW	58
CRAY J932		32	8 GB	6.4
IBM ES/9000 9021-9X2	Multiprocessor with dedicated buffer and shared memory	10	10.2GB	5.6
Fujitsu VPP 500	Highly parallel vector multiprocessor with distributed memory	222	55GB	355
VPP 700		256	32GB	500
Hitachi S-3800/480 water cooled	Multiprocessor	4	2GB	32
Hitachi S-3800/180	Single processor	1	1GB	2

Air cooled				
Hitachi S-820/80	Single processor with multiple pipelines	1	1GB	2
SX-4/512 M16	Multiple node shared memory	512	128	1024

Massively **parallel processor** system comprises hundreds or thousands of VLSI processors integrated through high bandwidth networks. In the last few years there has been an explosion in the number of parallel processing machines developed. The architectural characteristics of some of parallel processing machines [10] are given below in Table 1.2

Table1.2: Parallel processing machines

System / Model	Architecture	Number of processors	Maximum main memory	Peak computational rate	
				Each processor node (Mflops)	Total (Gflops)
Thinking machine corp. CM-5 GWS-100	Cluster MIMD	64	32	334	21
GWS-200	Cluster MIMD	129	64	400	51.2
Intel paragon XP/S	MIMD, 2-D mesh	1872	59.9		184
XP/S MP	MIMD, 2-D mesh	2000	256	75	300
Maspar MP-2	SIMD, 2-D mesh	16384	4	0.146	2.4
nCUBE 2	MIMD	8192	282	4.1	34
nCUBE 3	Hypercube	512	32	80	46
CRAY T3E	MIMD 3D bidirectional	2048	4000	600	1200

	torus topology				
Meiko CS-2	MIMD multistage packet switch	1024	572	250	256
IBM 9076/SP2	Distributed memory	128	256	308	39.4
Convex Exemplar SPP-1600/XA	Numa shared memory system	64	16	240	15.4

1.4 PARALLEL COMPUTING EFFORTS IN INDIA

India launched a major initiative in parallel computing in 1988. This was motivated by the need for advanced computing, a vision of developing its own technology, and difficulties (political and economic) in obtaining commercial products from outside the country. The creation of the Center for Development of Advanced Computing (C-DAC) and concurrently other efforts at National Aerospace Laboratory (NAL), Bangalore, Advanced Numerical Research & Analysis Group (ANURAG), Hyderabad, Bhabha Atomic Research Center (BARC), Bombay, Center for Development of Telematics (C-DOT), Bangalore, marked the beginning of high performance computing in India. Today, India has designed its own high performance computers. Details of some of such supercomputers [9] and theirs applications are tabulated below in Table 1.3

Table1.3: Supercomputers developed in India

Developer	Computer	Year	Specification	Application
C-DAC	PARAM 8000	1988	▪ Theoretical Peak computing power 1Gflops ▪ Computing power in Actual application 100 to 200 Mflops	▪ Computation fluid dynamics ▪ Finite element analysis ▪ Seismic data processing ▪ Image processing ▪ Signal processing ▪ Molecular modeling ▪ Quantum molecular dynamics
C-DAC	PARAM 8600	1992	▪ Peak computing power 100-200 Mflops. ▪ i860 as main processor with 4 transputers acting as communication processors	
C-DAC	PARAM 10000	1998	▪ Peak computing power 100Mflops	▪ Computational structural mechanics ▪ Computational chemistry ▪ Computational fluid dynamics ▪ Evolutionary computing ▪ Seismic data processing
C-DAC	PARAM Padma	2001	▪ Peak computing power 1 Tflops ▪ POWER4 RISC processors ▪ High performance system area network PARAMNet-II provides data rates of 2.5 GB/sec ▪ Also accessible by users from remote area	
BARC	ANUPAM	1992	▪ MIMD architecture ▪ i860 processor ▪ Peak computing	▪ Parallel computing facility has been used extensively by

			power 100 Mflops	BARC scientists and engineers as well as by other users from outside Institutions
ANURAG	PACE		▪ 4 node Loosely coupled message passing parallel processing system ▪ UNIX operating system ▪ Local memory 256 MB	▪ Aerodynamic computations ▪ Neural network simulation ▪ Computational fluid dynamics codes ▪ Finite Element Method codes
NAL	FLOSOLVER	1986	▪ FLOSOLVER Mk1A - four processor system ▪ FLOSOLVER Mk1B - eight processor system ▪ FLOSOLVER Mk2 – based on Intel 80386/ 80387 processor ▪ FLOSOLVER Mk3 – based on RISC processor i860 from Intel	▪ Computational Fluid dynamics ▪ Aerodynamics ▪ Weather forecasting
C- DOTS	CHIPPS	1989	▪ Single algorithm multiple data architecture ▪ Available in 192 node machine, 64 node machine and	▪ Weather forecasting ▪ Radio astronomy applications ▪ Other scientific and engineering applications

			compact 16 node machine ▪ Peak computing power 200 Mflops	

1.5 PARALLEL PROCESSING IN FINITE ELEMENT ANALYSIS

In structural engineering applications, the Finite Element Analysis plays a prominent role. Several avenues for parallelizing this method have been proposed. They range from taking advantage of general-purpose hardware in the parallelization of the method to the design of special purpose hardware, such as the Finite Element Machine. In many finite element applications, the most costly phase of the computations is the repeated solution of large systems of linear algebraic equations. Several writers have been researching concurrent algorithms in order to optimize this operation. These methods include parallel equation solvers, as well as concurrent algorithms derived from domain decomposition methodologies, operator splitting techniques, or element-by-element strategies. Many of the above-mentioned techniques are based on the subdivision of the physical domain into a number of sub domains.

The architecture of some machines has been designed to solve efficiently specific problems. These machines are called special purpose machines. For the finite element method two types of special purpose machines have been developed. The first one, developed at NASA, is called the Finite Element Machine. In this approach, the nodal structure of the finite element mesh is mapped onto the hardware. In another development, the architecture of the parallel finite element machine developed at the University of Calgary is related to the solution technique to solve problems with geometric and material non-linearties. Several techniques for the parallelization of the finite element method have been employed. These can be divided as follows [16]:

1.5.1 Domain decomposition

This concept is based in the ‘‘divide and conquer’’ technique. Here, the task to be performed is divided into independent or loosely coupled subtasks, and thus communication among processors is reduced. The idea is one of domain decomposition, i.e., the domain is divided

into regions. The problem is then decomposed into the solution of boundary value problems in the sub domains.

1.5.2 Sub structuring

This strategy is closely related to that of domain decomposition. It was originally introduced in the 1960s to reduce the dependency in the out-of-core memory in large-scale structural engineering applications. The basic idea is to subdivide a structure into a number of substructures. The degrees of freedom of a substructure are then classified as internal DOFs and interface DOFs. The process consists of applying condensation to reduce the number of degrees of freedom in each substructure.

1.5.3 Operator splitting

This technique is a generalization of sub structuring. Operator splitting strategies can be developed in a variety of ways. One example is the method of alternating directions, whereby a multidimensional problem is reduced into a series of one-dimensional problems.

1.5.4 Element-by-element (EBE) strategies

In these strategies, advantage is taken of the fact that the element calculations can be done independently and thus can be trivially parallelized. In order to improve the efficiency of EBE strategies, they should be coupled with parallel equation solvers.

1.6 OBJECT OF STUDY

Analysis of large size problems as well as for problems having geometry and other irregularities is very tedious, time-consuming process. So to overcome this shortcoming, innovative techniques are developed. Parallel processing is one of such innovative techniques. The main object of present study is to find the cost effective solution for high performance computing.

In this analysis technique, whole structure is divided into number of substructures. Analysis of each sub structures is done concurrently on different processors, interconnected to

communicate with each other and to transfer the data. As the whole process is distributed among various processors, interconnected through LAN, it can be called as "Distributed Finite Element Analysis". As the main object is to observe the reduction in processing time after implementation of parallel processing technique, first the whole structure is descretized and analyzed as a single structure (using sequential programming) and then the structure is divided into substructure and structure is analyzed with sub structure technique (using parallel programming).

The total time elapsed in parallel processing is the time required for two activities namely, computation and communication. Out of these, time required for communication would not have significant effect on sequential as well as parallel timing. So the advantage in reduction of time can be observed only in large size problems. While for small problem the process time will be almost same for sequential and parallel implementation and so the advantage of implementation of parallel processing cannot be visualized.

1.7 ORGANIZATION OF REPORT

Whole dissertation is organized in the following manner.

Chapter 1 includes the introduction part. Some of the topics related to parallel processing and its application are also covered. These topics include need of parallel processing, parallel processing efforts in world and in India, parallel processing application in FEA, and the object of the study.

Literature survey part is included in chapter 2. It includes the literature regarding various applications of supercomputers and parallel machines in structural engineering fields such as analysis and optimization. Scope of the present study is also mentioned in the same chapter.

Various parallel processing related issues are discussed in chapter 3. These topics include, types of parallel processing, various parallel processing architectures, parallelism in sequential and multiprocessor machine, dynamic load balancing, pipelining, operating system characteristics for parallel processing, various parallel processing models, performance measurement of parallel processing and parallel processing software.

Chapter 4 includes various aspects of the WebDedip tool. These aspects include, architecture of WebDedip, installation of WebDedip, hardware as well as software requirement of WebDedip, various components to be installed on server and slave machine, configuration of various components of WebDedip and also the process of implementing the application on WebDedip.

Chapter 5 includes the first application of parallel processing. The first application is the plane stress analysis of a square plate with circular hole in middle and subjected to uniform edge force. Result and discussion about the same is also covered in the same chapter. Discussion regarding substructure analysis technique is also included in the same chapter.

Chapter 6 includes the second application of plate-bending which is analysis of a fixed circular plate, having hole in the center. The plate is subjected to transverse uniformly distributed loading. Another problem in the same chapter is the analysis of skew plate. The plate is a simply supported plate and subjected to transverse uniformly distributed loading.

Chapter 7 covers distributed finite element analysis of laminated plate analysis. The plate is a simply supported square laminated plate. Plate is subjected to sinusoidal transverse loading. Result and discussion about the same is also covered in the same chapter.

Chapter 8 includes the final summary of work along with the further scope of the work that can be carried out on the same topic.

CHAPTER 2 LITERATURE SURVEY

2.1 LITERATURE SURVEY

Literature survey is carried out to review various applications of parallel processing technique in the field of structural engineering. Such applications include analysis, optimization and arithmetical solvers to solve simultaneous equations resulting in finite element analysis. In literature survey, main emphasis is given on the published papers that describe the implementation of parallel processing fro various structural problems.

Saxena M and **Perucchio R [1]** developed parallel FEM algorithms for automatic mesh generation based on recursive spatial decomposition (RSD) and for automatic analysis via sub structuring for solid models. They also described an automatic sub-structuring scheme based on RSD of the domain that can be closely integrated with the RSD based automatic meshing procedure. The hierarchical data structure used to represent RSD based automatically derived meshes also provide a one to one mapping between spatially decomposed sub domains and analytical sub structures. Such a hierarchical organization of the sub structures and the inherent parallelism of RSD are exploited to design the sub-structuring scheme suitable for parallel processing.

Adeli H and **Kamal O [2]** have used the parallel processing concept for developing algorithms for concurrent analysis of large structures on shared memory multiprocessor computers. Algorithm was developed to parallelise each computational step of solution process. Also the algorithm was developed to maintain the workload balance through the solution process and to assure best concurrent performance and speed up.

To check the performance of the algorithm, **Adeli H** and **Kamal O [3]** have implemented the algorithm for analysis of four different structures on an Encore Multimax shared memory multiprocessor computer. Of the four structures, first was the 266-element frame structure. The second was the Geodesic dome space truss. Third was the 200 bar plane truss and the forth structure was the 760 element frame structure. Performance was measured here in terms of Speed up, workload balance and efficiency. The results for the speed up indicated that the efficiency of the parallel processing algorithm increases with the size of the structure. Overall efficiency of about 90 to 100% was achieved for the case of a 1200 DOF.

Foley C M and **Vinnakota S [4]** have implemented sub structuring method and parallel processing technique for the analysis of high-rise structural frameworks. The performance was measured on a Cray Y-MP C90 supercomputer in a multi user environment. Performance of the program was checked by implementation of this technique on two steel frameworks. Of two, one was having 175 members and 312 DOF. The other frame was having 461 members and 780 DOF.

Adeli H and **Kumar S [6]** developed algorithm for distributed finite element analysis on a loosely coupled multicomputers such as a cluster of inexpensive workstations. Solver based on the element-by-element approach is used to solve the resulting system of linear equations. To account for the slow communication speed of the ethernet network connecting workstations, techniques such as efficient data distribution were implemented. Due to the general nature of the data distribution, the algorithms are versatile and can be applied to the analysis of FE domains consisting of combination of various types of elements. The algorithm is based on and can be implemented on any distributed memory architecture.

Adeli H and **Kumar S [7]** implemented distributed algorithms for finite element analysis of structures, on network of workstations. The algorithms developed, was applied for analysis of large structure with complicated topology. A cluster of six IBM RS/6000 workstations was used for analysis of large structural models with few thousand elements. In spite of low bandwidth of the ethernet networks, an overall parallelization efficiency in the range of 75-90% was obtained. This technique was used to solve three problems. Of which first was the plane stress L shape domain, solved with four different types of elements, the linear and isoparametric quadrilateral as well as triangle element. Second application was a 100 story high-rise building with 5890 elements and 1240 nodes. The structure was modeled both as an axial load structure and as a moment resisting frame structure. The results for six workstations have shown a maximum speed up of 3.75 for moment resisting frame structure. The third application was the Taj Mahal structure. The results for third application, have shown a maximum speed up 3.25 for six workstations.

In another application **Adeli H** and **Kumar S [8]** implemented parallel processing technique for minimum weight design of large structures using genetic algorithm, on a network of workstations. Communication constructs from software library Parallel Virtual Machine (PVM) have been used for message passing between workstations. The performance was

noted for both dedicated and shared multi-user environment in a laboratory with 11 IBM RS/6000 workstations connected through the ethernet network. The cluster was heterogeneous. Six machines were of model 320H and the remaining five were of model 220H. The performance was provided based on granularity and parallelization efficiency of distributed model. The algorithm was applied for the minimum weight design of two structures. Of two one was the Dome space truss and the other was a 35-story space tower frame. For first application (i.e. medium size structure), speed up of about 5 was achieved on 11 workstations. Thus the parallel efficiency was about 50 to 60%. While for second application (i.e. large structure), speedup of 10 was achieved on 11 workstations. Thus the parallel efficiency was about 90%. Thus from results it was shown that the speed up of distributed algorithm increases with the size of structure.

Noor A K [10] reviewed recent advances in technology that are likely to impact structural analysis and design. A brief summary of the advances in microelectronics and networking technologies was described. The major features of new and projected computing systems, including parallel processing, high performance computers, and small systems were also discussed. An advance in programming environment, numerical algorithms and computational strategies for new computing system was also reviewed. A scenario for future-computing paradigm was presented and the near term needs in the computational structures areas were outlined. New computing systems were also described that could give structural analysts and designers some insight into the potential of these systems for providing cost effective solution of complex structural problems and to stimulate research and development of the necessary algorithms, firmware and software to realize this potential.

Wriggers and **Boersma [12]** implemented parallel solver for problems in solid mechanics discretized by finite elements. Performance of this solver was studied on two different MIMD computers. Of the two MIMD computers, one was the 64-node machine with T800 processor and the other was 32-node machine with a Power PC processor. The technique was implemented to two problems. First problem was solved for 64 X 64 elements on each processor for T800 system and 180 X 180 elements on each processor for Power Xplorer system. Second problem was solved for 128 X 128 elements on each processor for T800 system and 256 X 256 elements on each processor for Power Xplorer system. To rate the performance of the parallel application, the well-known principle of speed-up and scale-up were used. The parallel algorithm gave a high parallel efficiency when used for large

problems. So from paper it can be concluded that, to increase the efficiency further, a high load on each processor is essential specially when the ratio of communication to computing is low.

Soegiarso R and **Adeli H [13]** implemented the parallel-vector algorithm for optimization of large steel structures on a shared memory machine. The Cray YMP8/864 supercomputer with eight processors was used. The algorithm was used for minimum weight design of several high-rise buildings (20 story moment resisting space frame with 1920 members and 4536 DOF, 40 story moment resisting space frame with 3840 members and 8856 DOF, 60 story moment resisting space frame with 5760 members and 13176 DOF) including an 81 story building structure with 9245 members. The parallel processing speed up was measured through software called 'atexpert'. From the results for 81-story building, it can be seen that the performance of parallel processing improve with the increase in the size of structures. For 81-story building speed-up of 6.58 was achieved for the parallel processing.

Zucchini A [14] implemented parallel processing technique to develop solver for finite element problems with coarse mesh/fine mesh formulation. The algorithm was tested for two simple problems of plane stress in a linear elastic plate with boundary and loading conditions. Standard 8 node quadrilateral isoparametric element was used. To visualize the advantage of parallel processing, the method was implemented both, on a single processor IBM-RISC590 and on a Quadrics-QH1, a massive parallel SIMD machine with 128 processors.

Adeli H [15] reviewed various research papers published on parallel processing, supercomputing and distributed computing since 1994, in his paper titled "High Performance Computing For Large Scale Analysis, Optimization And Control". The main focus of the paper was on review of the journal articles published in three areas: analysis, optimization and control. The review was divided into three main sections:

1.) Parallel processing on dedicated shared memory and distributed memory parallel machines

Gummadi and Palazotto (1997) described geometrically non-linear analysis of beams and arches with large displacements and rotations on a distributed memory Intel paragon parallel computers. Watkins (1997) described fine and coarse-grained parallel methods for the

computation of real Eigen values encountered in buckling and vibration analysis of structures on a distributed memory computer, a 32 processor nCUBE2. Parallel vector Cholesky and LU decomposition methods for solving set of linear simultaneous equations were developed by Agrawal (1994) utilizing parallel processing and vectorization capabilities of shared memory multiprocessors such as CRAY Y-MP machines. Saleh and Adeli (1994) investigated various multitasking approaches for optimization of large structures with maximum speedup performance. Adeli and Cheng (1994) presented parallel augmented Lagrangian genetic algorithm for optimization of large structures utilizing the multiprocessing capabilities of high performance computers such as the Cray YMP 8/864 supercomputer. Topping and Leite (1997) also presented parallel genetic models for structural optimization. Park and Adeli (1997) presented a data parallel neural dynamics model for discrete optimization of large structures such as super high-rise building structure consisting of thousands of members.

2.) Distributed computing on a cluster of networked workstations

Hudli and Pidaparti (1995) also discussed distributed finite element analysis of simple truss structure using the client-server model and remote procedure calls on a cluster of SUN workstations. Weinert and Eschenauer (1996) described a coarse-grained decomposition strategy for parallel solution of structural optimization problems on cluster of workstations.

3.) Parallel computing and object oriented programming

Yu and Adeli (1993) presented an object oriented finite element approach for analysis of complicated structural systems using an object oriented enhanced entity relationship model. Modak (1997) described the design and implementation of an object oriented parallel matrix class library in C++ for manipulation of matrices and solution of simultaneous linear equations encountered in finite element analysis.

As per **Sotelino E D's [16]** view, parallel processing is a rapidly evolving field and it is the focus of intensive research worldwide. In past two decades, the development of algorithms for structural engineering applications has received a boost due to the advent of parallel computers. Considerable research is being done in order to rewrite the algorithms originally designed to run on sequential machines as well as to develop new methods that take advantage of the parallelism offered by the multiprocessing computers. Such algorithms

include parallel solvers (direct and iterative) for linear systems of algebraic equations, techniques for the parallelization of the finite element method, and concurrent time stepping algorithms for the solution of the equations of evolution. In structural engineering applications, a prominent role is played by finite element method. Several avenues for parallelizing this method have been proposed. They range from taking advantage of general-purpose hardware in the perallelization of the method to the design of special purpose hardware such as the finite element machine.

2.2 SCOPE OF STUDY

As the Finite Element Analysis can handle very well the geometry and material irregularities, it is the most commonly used numerical analysis method for many of the engineering problems. Also the parallel processing technique can be easily implemented to it. So in present study, Finite Element Method is used for the analysis.

In the present study three different problems all having different type of elements, are selected and analyzed using Finite Element Method. Of three problems, first is the Plane stress analysis, second one is the plate bending problem and the third problem is the laminated composite plate problem. These continuums are then discretized in to number of finite elements such as CST elements for plane stress analysis, into Quadrilateral isoparametric plate bending elements for plate bending problem and in Quadrilateral isoparametric laminated plate elements for problem of laminated plate analysis. Whole continuum is then divided into substructures in such a ways that number of elements in each sub structure remains same (to maintain the work load on each processor). This is necessary to keep the processing time of each process more or less equal. Each sub structure is then analyzed sequentially as well as on parallel computers in network of computers. Detail of four problems, selected in the present study, can be summarized as follows:

1.) First application includes the distributed plane stress analysis of rectangular plate having circular hole in the center [25], as shown in Fig 2.1. Plate is subjected to uniform tensile force on both the edges. As the analysis is plane stress analysis, Constant Stress Triangular (CST) element is used to descritize the continuum. Due to the symmetry of the plate w.r.t both x and y-axis, only the quarter plate is analyzed here. Plate is a square plate having size 30 cm X 30cm and thickness of 2.5cm. Diameter of the hole is 2cm. Modulus

of elasticity of plate material is 2 X10^4 kN/cm^2 and Poisson's ratio is 0.3. Intensity of uniform edge tension force is 10 kN/cm^2. For problem under consideration, the quarter plate is divided into 27504 number of elements. So problem is having total 13990 number of nodes i.e. 27980 DOFs (2 DOF at each node). Further the plate is divided into number of substructures to implement on parallel processing. In present problem, the plate is divided into 3, 4, 5 and 6 substructures.

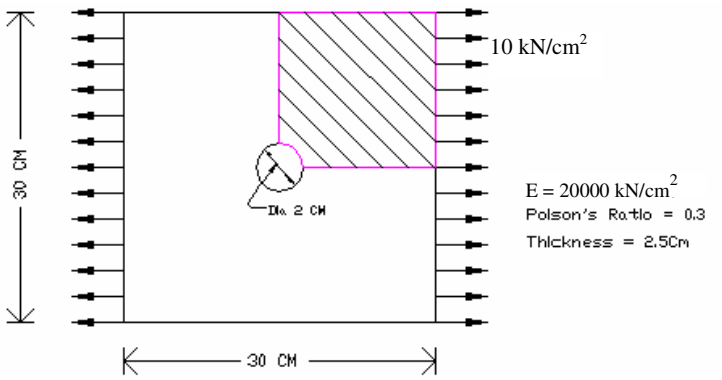


Fig 2.1: Plate having hole in the center and subjected to tension

2.) Second application includes the distributed analysis of circular plate, having hole in the center, as shown in the Fig 2.2. Plate is subjected to uniform lateral load and it is fixed at outer circumference and free at inner circumference. For descritizing the plate, eight nodded quadrilateral isoparametric plate element is used. Due to symmetry of plate w.r.t both x and y-axis, only quarter plate is analyzed. In present study, the ratio of outer diameter to inner diameter is kept 3. Inner diameter of plate is 2.5m and outer diameter is 7.5m. For plate material, modulus of elasticity is 2 X 10^8 kN/m^2 and Poisson's ratio is 0.3. Thickness of plate is 0.1m. Intensity of uniformly distributed transverse loading is 20 kN/m^2. For present problem, the quarter circular plate is descritized into 1800 elements. So problem is having total 5725 number of nodes and 17175 DOFs (3 DOF at each node). Further the plate is divided into 3, 4 and 6 substructures to implement on parallel computers. Analysis of each substructure is carried out on separate computer.

One more application using the same eight nodded quadrilateral isoparametric plate element is also included. That is distributed skew plate analysis. Geometry of the same is as shown in Fig 2.2. It is a simply supported plate supported on both inclined edges. Loading is the uniformly distributed transverse load over entire plate. Modulus of elasticity of plate material is 2 X 10^8 kN/m^2 and Poisson's ratio is 0.3. Thickness of plate

is 0.2m. Here inclination angle is kept 60 degree. Inclined length of plate in x-direction is taken as 10m and length in y direction is also taken as 10m. in present problem the skew plate is divided into 3000 number of elements. So total number of node is 9221 and total DOFs is 27663 (each node is having 3 DOFs). For parallel processing, further the plate is divided into 3, 4 and 6 number of substructures.

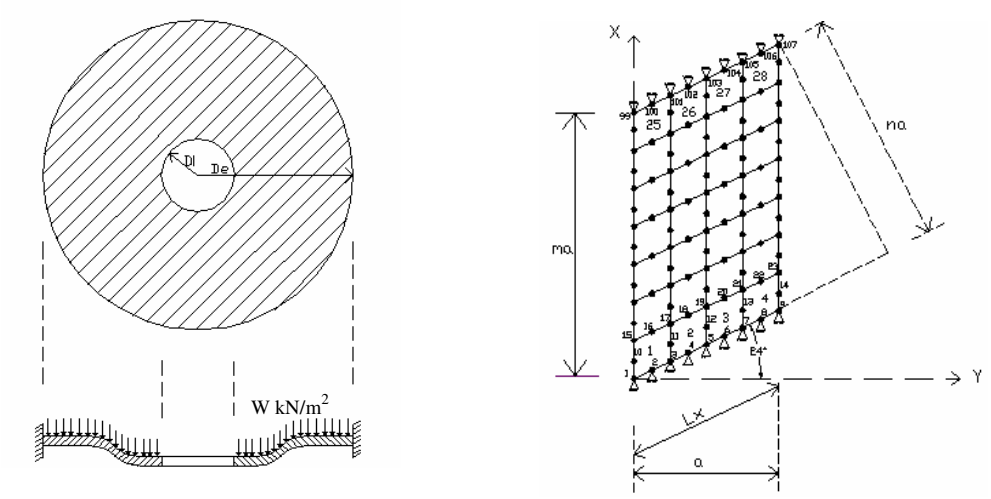


Fig 2.2: Circular plate having hole in the center and skew plate

3.) Distributed analysis of laminated composite plate, as shown in Fig 2.3, is the third application. For analysis of composite plate 8 nodded quadrilateral isoparametric element is used. Plate is simply supported at all its edges and subjected to sinusoidal transverse lading. Due to symmetry of plate w.r.t both x and y-axis, only quarter plate is analyzed. Laminate plate that is selected for the analysis is a square plate having both dimensions as 100 cm. Plate consists of four laminates, each of 6.5 cm thick, so the total thickness of plate is 25cm. Intensity of sinusoidal load is 10kN/cm^2. Various material properties for laminated plate are as follows.

$E_1 / E_2 = 25$ $E_3 / E_2 = 1$

$\nu_{12} = \nu_{23} = \nu_{13} = 0.25$

G1 / E2 = 0.2 G2 / E2 = 0.5 G3 / E3 = 0.2

For present work, laminated plate is descritized into 1600 number of elements. So total number of nodes is 4961 and total DOFs is 29766 (each node is having 6 DOFs). For

parallel implementation, the plate is further divided into 3, 4 and 6 number of substructures.

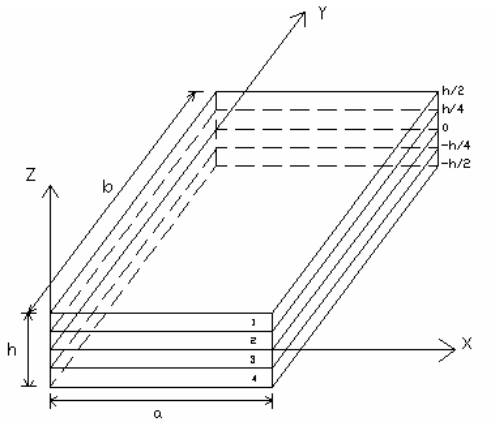


Fig 2.4: Laminated composite plate

CHAPTER 3 PARALLEL PROCESSING ISSUES

3.1 GENERAL

The art of parallel programming lies in decomposing a given problem, functionally and according to the data distribution, across processors. Over the years, work in parallel processing architectures have focused on a number of different issues and resulted in a range of different approaches for solving them. The resulting architectures have been fundamentally so disparate that the programming issues vary significantly across them. An algorithm that gives a very high speedup on one processors architecture can give a very poor performance on another. Even if only the MIMD class of parallel machines are considered, difference between the shared memory and distributed memory machines force developers to adopt different data function decomposition strategies for the same problem on these classes' machines [24].

The fundamental concept behind high performance computing is to use more number of resources to solve a given problem. But using more resources cannot speedup every task. There are restrictions and dependencies to be considered. Based on such issues, computations may be one of the following characteristics [28].

1.) Embarrassingly parallel computations

An ideal parallel computation that can be divided into a number of completely independent parts each of which can be executed by a separate processor is known as embarrassingly parallel computation or pleasantly parallel computation. Parallelizing these problems should be obvious and requires no special techniques to obtain working solution. A truly embarrassingly parallel computation suggests no communication between separate processes. Each process requires different data and produces results from its input without any need for results from other processors. This situation will give the maximum possible speedup if all the available processors can be assigned processes for the total duration of the computation. The only constructs required here are simply to distribute the data and to start the processes. The SIMD architecture is appropriate for such processes. Example of such process can be given as follows:

a = b + c d = e + f

Consider the above two processes, belonging to a single computation task. As the variables used in both the processes are altogether independent of each other, both the processes can be run parallely without having any effect on the final result. So such processes are called embarrassingly parallel computations.

2.) Nearly embarrassingly parallel computations

The computations that require results to be distributed and collected and combined in some way are known as nearly embarrassingly parallel computations. So in such computations, initially and finally a single process must be operating alone. If dynamic process creation is used, a common approach is the master-slave organization. First, a master approach will be started that will initiate identical slave processes.

3.) Inherently sequential computations

The computations, in which the parallelism of processes is not at all possible, are known as inherent sequential computations. This can be better understood by the following example:

a = b + c d = a * a

Consider the above two processes, belonging to a single computation task. In the first process, two variables (b and c) are added and that value is assigned to variable “a”. While in the second process, square of variable “a” is calculated and that value is assigned to variable “d”. As in the above processes the value of variable “d” is dependent on the value of variable ”a”, it is necessary to modify the value of “a” before using it to calculate the value of “d”. So these two processes cannot be run parallely and must be run sequentially. So this computation is the example of inherent parallel computing.

3.2 COMPUTER ARCHITECTURES

Various architectures have been proposed and recommended by the researchers in the area of parallel processing. Rather than having only a single instruction flow through the CPU processing a single stream of data, the generalized architecture is now aiming for multiprocessor executing possibly distinct streams of instructions and processing distinct data streams (MIMD). The architectures having the simplest programming interface support have become most popular [29].

Architecture for computers can be classified as follows:

1.) SISD (Single instruction single data)
2.) SIMD (Single instruction multiple data)
3.) MIMD (multiple instruction multiple data)
4.) MISD (multiple instruction single data)

Further the SIMD and MIMD architecture have both, shared and distributed memory architectures.

3.2.1 SISD

This is the traditional sequential computer consisting of one processor, memory and one communication channel as shown in Fig 3.1

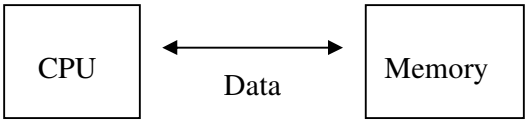


Fig 3.1: Sequential computer architecture (SISD)

As a parallel computer, this architecture has various shortcomings like memory, speed and communication channel. With the present technology providing almost unlimited memory and extremely fast processing units, the bottleneck is the communication channel.

3.2.2 SIMD

This can be viewed as the extension of the conventional sequential model. This architecture overcomes the bottleneck in the SISD architecture by implementing multiple processors and memory model working on single instruction set. However host communication may prove to be troublesome for this architecture since each processor has to share the same communication path through the pipeline.

The SIMD architecture is available in two forms, the distributed memory model and shared memory model.

SIMD with shared memory

In this architecture there is more than one processor connected to single root processor and sharing the same memory. As all processors have to share the same memory, when a processor wants to access the memory, it will try to take over the bus and proceed as before. However, the bus may not be available when one wants it because someone else is already using it. So in this architecture, along with the host communication channel, the memory bus may become a bottleneck since all processors have to share a common bus for memory.

SIMD architecture with shared memory is shown in Fig 3.2

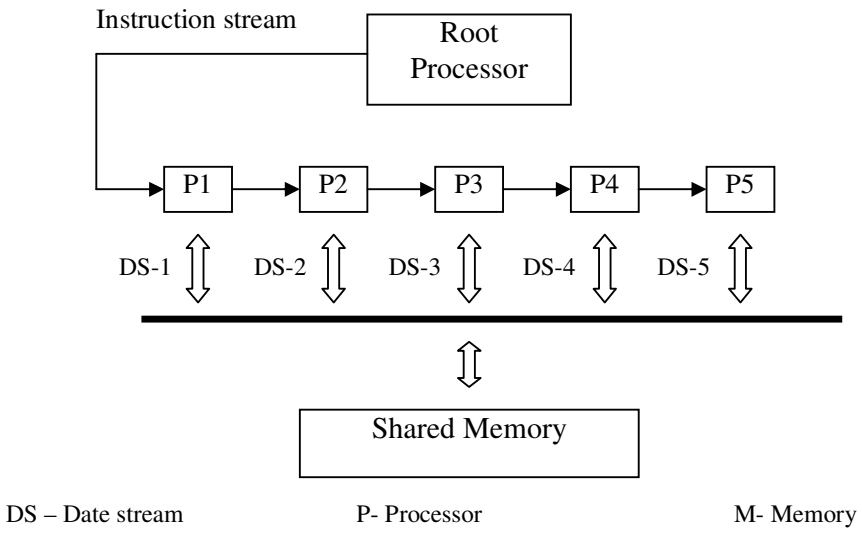


Fig 3.2: SIMD model with shared memory

It can be seen from the fig that the time taken by any processor to access memory is independent of the processor. So this model is also called known as UMA (Uniform Memory Access) model.

SIMD with distributed memory

This architecture is the modified form of the above model and shown in Fig 3.3. In this model the bottleneck of memory bus is removed by assigning separate memory for each processor. So the data needed by a particular processor can be stored in the memory of that processor so that the processor does not have to go the common bus to access data. For example, processor

P1 can access A1 memory much faster than the memory A2, A3 or any other memory. In this model as the time to access a memory location depends on whether it is attached to the invoking CPU or not, this model is also known as NUMA (Non Uniform Memory Access) model.

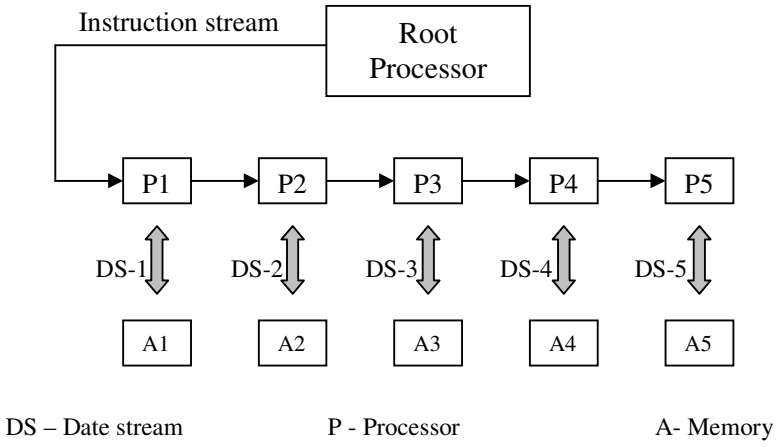


Fig 3.3: SIMD model with distributed memory

3.2.3 MIMD

This model is further the modified version of the SIMD model and overcome the host communication bottleneck of SIMD model by introducing separate instruction stream for each processor. Like SIMD the MIMD model is also available in two forms, shared memory and distributed memory model.

MIMD with shared memory

As shown in Fig 3.4, this model has separate instruction stream for each processor. However this system has one shortcoming, the common memory access bus. Thus the processor has to queue to access the same area of memory.

MIMD with distributed memory

This model overcomes the shortcoming of common memory access bus of MIMD shared memory model by assigning separate memory to each processor. In this model each processor

is provided with separate memory and connected with each other through instruction channel to communicate with each other.

MIMD model with distributed memory is shown in the following Fig 3.5

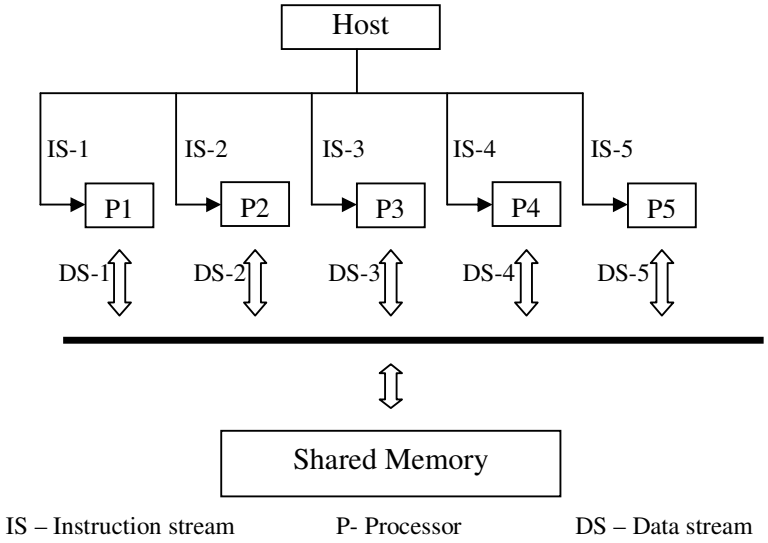


Fig 3.4: MIMD model with shared memory

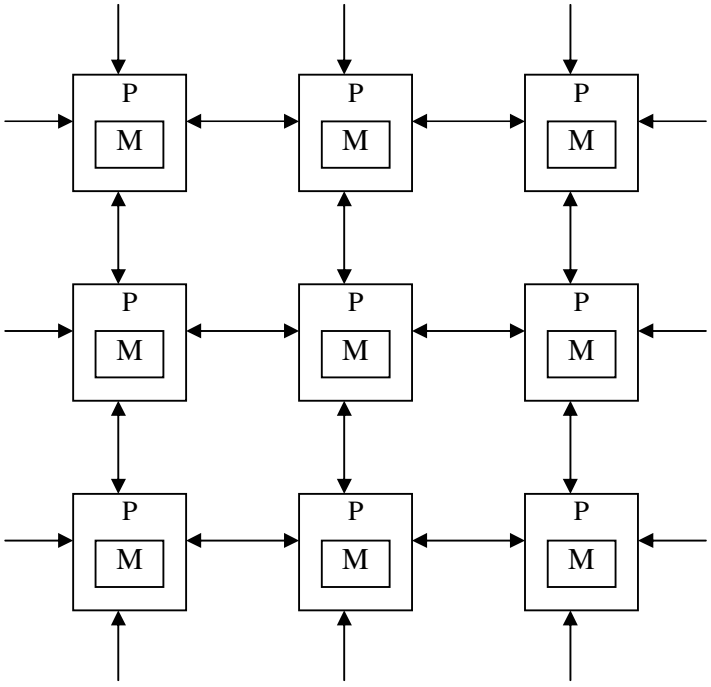


Fig 3.5: MIMD model with distributed memory

3.2.4 MISD

This architecture has yet to be implemented.

3.3 PARALLELISM IN SEQUENTIAL MACHINES

Incorporation of various ingenious schemes has made sequential machines faster. The main idea in all these schemes is to match the speeds of various components so as to utilize the resources to their peak performance. For example, speed of memory access should be matched with the speed of processor. When this is not achieved directly, use of techniques such as interleaving of successive memory locations across memory devices is resorted to. Matching of interfacing devices is needed so as to minimize the idle time of the faster device, thereby achieving higher throughput. Various aspects related with such parallelism are as follows [29]:

3.3.1 Multiplicity of functional units

The most primitive computer had only one arithmetic and logical unit in its processor. Besides, its ALU could perform only one function at a time. The practical machines in use today have multiple and specialized functional units that can operate in parallel. The CDC – 600 (designed in 1964) had 10 functional units built into its CPU. These 10 units were independent of each other and could operate simultaneously.

3.3.2 Pipelining within the CPU

Various phases of instruction execution are pipelined, including instruction fetch, decode, operand fetch, arithmetic/logical execution and storage of results. To facilitate overlapped instruction execution through pipe, instruction pre fetch and data buffering techniques have been developed. More than one instruction is in the process of execution in the processor. The execution of multiple instructions is overlapped in time- even before an instruction gets completely executed, another instruction may be in the process of being decoded, yet another instruction may be getting fetched, and so on. Pipelining of task gives us temporal parallelism.

3.3.3 Overlapped CPU and I/O Operations

Input and output operations can be performed simultaneously with the computational task by using separate I/O controllers, channels or I/O processors. Direct Memory Access (DMA) is used to provide direct information transfer between I/O devices and the primary memory.
The trickiest issue in the design of such system is that of bus contention. The advantage to be gained by overlapping devices interface and computational tasks can get undone if both these require the same bus to transfer the data. The potentially concurrent operations become serialized due to the hardware resource contention. To reduce this bus contention, some architectures employ redundant buses in system.

3.3.4 Hierarchical Memory system

Processors are generally orders of magnitude faster than the primary memory. The primary memory is capable of data transfer rates, which are many times faster than most secondary devices. A hierarchical memory system can be used to close up this speed gap. The topmost level is the register files directly addressable by ALU. The cache memory can be viewed as a buffer between processor and main memory. Parallelism in data transfer across the hierarchy is commonly exploited for an improved throughput. For example, while data is transferred from secondary memory to primary memory, some data may be simultaneously transferred from cache to the CPU. This is possible since the data paths for these transfers are independent of each other and hence there is no contention of access.

3.3.5 Multiprogramming and Time-Sharing

In a multiprogramming system, certain computationally intensive process can hog the CPU and not release it for the other processes till the job is completed. In such cases, the response time for the other process gets severely affected. To avoid this blocking of resources, the concept of time sharing operating system was introduced. The time-sharing scheduler assigns the CPU to process for fixed time slices in a round robin fashion. This way all processes that are ready to run are given a chance of computing for the CPU and other resources. Time-sharing of CPU across multiple processes gives rise to the concept of virtual processors. That is, each processor is provided an environment as if it has a virtual processor exclusively for its use.

3.4 PARALLELISM IN MULTIPROCESSOR MACHINE

While dealing with multiprocessor machine, two different terms should be clear in mind. Multiprocessor and Multicomputer. A multiprocessor can be characterized by two attributes: first, it is a single computer that includes multiple processors, second, process may communicate and cooperate at different levels in solving a given problem. The communication may occur by sending messages from one processor to other by sharing common memory. A single operating system governs activities in such computers. While a multicomputer is a collection of multiple interconnected computer system, each of which is an autonomous machine having private local memory, resources and an operating system controlling its operation, communication may take place across the computers by message passing.

Depending on the cost incurred in communicating between processes running on the processors in multicomputers and multiprocessors, these machines are also known as tightly coupled parallel machines and loosely coupled parallel machines. In the former the cost of communication is much lower than that in later. The contention of memory is much higher in tightly coupled machines compared to loosely coupled machines because there is no shared memory across processors.

3.5 CHARACTERISTICS OF MULTIPROCESSORS

Following are some desirable characteristics of a multiprocessor machine to be used as a parallel machine [29].

3.5.1 Process recoverability

If a processor fails, the process running on it should be recoverable. Another processor must be assigned to the task and the process must be continued from the state where the process failed.

3.5.2 Efficient context switching

Sometimes, the parallel programs executing on the parallel machines have processes that are more in number compared to the number of processors in machine. In such a case, there is a need of swapping the processes in and out of context. The processor must have efficient mechanism to support operating system in switching the context efficiently.

3.5.3 Large virtual and physical address space

With the increase in the size of problem being solved on the parallel machines, there is a growing need for large memory space and addressing capabilities. Problems like whether forecasting, global illumination computation and computer vision require huge amount of memory for effective and efficient solution.

3.5.4 Effective synchronization primitives

Multiple processes that constitute a parallel program need to cooperate to compute the results. This is possible by sharing data and accessing shared data while maintaining its integrity. To maintain integrity of data structures, often the access needs to be allowed only in exclusive mode. This means all other processes must wait while a process is accessing the shared data structure.

3.5.5 Inter process communication mechanism

There should be proper mechanism for communication between processes constituting parallel program. The communication between cooperating processes takes place in the form of signals, messages and interrupts. For example, a process can notify another cooperating process of an event such as data_ ready.

3.6 DYNAMIC LOAD BALANCING

While executing a program to assign the unprocessed workload to available processors, uneven distribution of work among processors and result in poor utilization of computing resources. In this context, an effective dynamic load balancing and work scheduling scheme is

needed. To manage the load balancing problem, one seeks to apply an optimal work scheduling strategy to transfer workload automatically from heavily loaded processors to lightly loaded processors or processors approaching an idle state. The primary goal of dynamic load balancing algorithms is to schedule workload among processors during program execution, to prevent the appearance of idle processors, while minimizing interprocessor communication cost and thus maximizing the utilization of the computing resources. A common load balancing strategy is the "manager-worker" scheme in which a single "manager" processor centrally conducts a group of "worker" processors to perform a task concurrently. These parallel algorithms adopt a distributed strategy that allows each processor to locally make workload placement decisions.

All distributed parallel algorithms of this type are basically composed of five phases: workload measurement, state information exchange, transfer initiation, workload placement, and global termination. *Workload Measurement, a*s the first stage in a dynamic load balancing operation, involves evaluation of the current local workload using some "work index". *State Information Exchange* makes the local information available to all other cooperating processors, through inter processor message passing, to construct a global work index vector. *Transfer Initiation,* after obtaining an overview of the workload state, first of all decides if a workload placement is necessary to maintain balance and prevent an idle state. This is done according to an initiation policy, which dictates under what conditions a workload transfer is initiated, and decides which processors will trigger the load balancing operation. *Workload Placement* is the next step of load balancing algorithm. Here the donor processor splits the local stack into two parts, sending one part to the requesting processor and retaining the other. *Global Termination* takes place when the globally optimal solution have been found, making all processors idle.

3.7 OPERATING SYSTEM CHARACTERISTICS

There is conceptually little difference between the goals of an operating system for a uniprocessor machine and that for a multiprocessor machine. In multiprocessor, there is an additional complexity of handling multiprocessor. The increased complexity in the relationships among the resources in a parallel machine results in difficulties in scheduling the resources across competing tasks in the machine. The additional requirements of a multiprocessor operating system are discussed below. An operating system that fails to

perform well in these respects tends to negate other advantages which are associated with multiprocessing.

3.7.1 Load Balancing

The operating system must utilize the resources efficiently. This is commonly expressed in terms of achieving a uniform balance of loads across the processors. The operating system should schedule the subtasks such that there are no idle resources including the processors.

3.7.2 Scheduling cooperating processes

Parallel programs, which consist of concurrently executing tasks, must be scheduled such that collectively they are able to use the resources in the machine, required to solve the given problem. Scheduling half the subtasks of one parallel program and half the subtasks of another parallel program, which cannot cooperate can be wasteful. If a parallel program needs all these subtasks to be running and cooperating, then this can be achieved by maintaining a hierarchy of processes in the sense of parent child relationship within the operating system's data structure. Scheduling of processes can then be done such that the processes belonging a single parallel program are scheduled for execution together.

3.7.3 Graceful degradation in case of failure of one of the resources

A parallel machine having multiple resources of the same kind has high degree of fault tolerance. Operating system should be such that the failure of one of the resources should not result in a catastrophic system crash and able to reschedule the task that had been running on the failed resource and continue the parallel program.

3.7.4 Communication scheme

Parallel program need to share data and intermediate results across subtasks during the process for getting solution of the problem. To achieve effective cooperation among the subtasks of a parallel program, the operating system must provide adequate facilities for communication between tasks. These facilities vary depending upon whether the machine is of shared memory type or distributed memory type.

3.7.5 Synchronization mechanism

To ensure integrity of the shared data across the subtasks of a parallel program, synchronization between tasks is required. The shared data must be accessed under mutual exclusion. The tasks may need to wait till some state is reached across all the tasks of the parallel program. To achieve such synchronization, operating system should provide signaling mechanisms.

3.8 PARALLEL PROGRAMMING MODELS

A programming model is a collection of program abstractions providing the programmer a simplified and transparent view of computer hardware and software. Parallel programming models are specifically designed for multiprocessors, multicomputers. Five such models are described here. Parallel program is collection of processes or tasks. The models described here differ in the way these processes share data, achieve synchronization and communicate.

3.8.1 Shared Memory Model

This model is generally easier to program. Multiprocessor programming is based on the use of shared variables in commonly accessible memory for communication and sharing data. Besides sharing variables in a common address space, communication also takes place through software signals. Since multiple processors may attempt to access the shared data, data memory management or locking is required to ensure the integrity of data and handle conflicts.

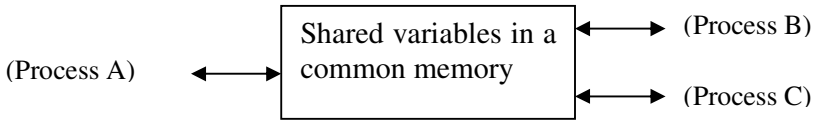


Fig 3.6: Shared memory model

This model is also available in the form of multithreading libraries. Various techniques are available for scheduling of processes across the subtasks to be carried out to solve the given instance of a problem. In such models it is the programmer's responsibility to decompose the

problem effectively and employ either static or dynamic scheduling of processes for achieving maximum parallelism.

3.8.2 Message Passing Model

Multicomputers employ message passing as the mechanism for inter process communication. Two processes residing at different nodes communicate with each other by passing messages over communication channel. The message may be instruction, data, and synchronization or interrupt signals. The communication delay in message delivery is much longer than in case of a shared memory model.

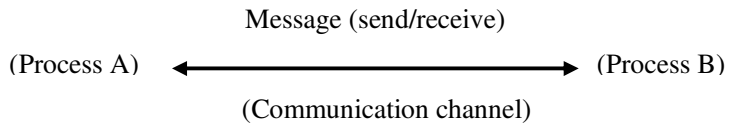


Fig 3.7: Message passing model

Two types of message passing models have been implemented.

Synchronous message passing requires both sender and the receiver to be synchronized in time for transfer of message. No buffering of message is done by the communication channel. The receiver is always blocked waiting for the message to arrive, the sender also remains blocked till the receiver receives and acknowledges the message. This mode of communication is suitable only for tightly coupled message passing multiprocessors, where communication delay overhead is sufficiently small.

Asynchronous message passing does not impose blocking on the sender. The message gets buffered by the communication channel and is delivered to the target processes when they choose to look for the message. This mode of communication is suited for multicomputers made up of networked autonomous machines.

Message passing model does not require mechanism for mutual exclusion for access to shared data because there is no way for processors to share each other's address space for data exchange. All data sharing is done through message passing.

3.8.3 Data Parallel Model

It is one of the simplest approaches to program and most appropriate for SIMD machines because the data is distributed to the processors and each executes the same set of instructions. The programming model for data parallel SIMD processors is an extension of the sequential programming. For example Fortran90 is specifically tailored for data parallelism. Data parallel programs require the use of pre-distributed data sets. Thus choice of parallel data structure plays a significant role in data parallel programming.

3.8.4 Object oriented model

Mapping of execution units to object is naturally achieved in this model. Objects are dynamically created and manipulated. Processing is performed by sending and receiving messages among the objects. Concurrent programming models are built up from low level objects such as processors, threads, queues into high level objects like monitors and program modules.

3.8.5 Functional and Logic Model

A functional programming language emphasizes the functionality of a program and should not produce any side effect during execution. There is no concept of storage, assignment and branching in functional programming. The lack of side effects opens up much more opportunity for parallelism. The evaluation of a function produces the same value regardless of the order in which its arguments are evaluated. Thus arguments in dynamically created structures of a functional program can be evaluated in parallel.

Logical programming is based on predicate logic and is suitable for knowledge processing dealing with large database. This model adopts an implicit search strategy and supports parallelism in the logic interface process. A question is answered if matching facts are found in the database. Two facts match if their predicates and associated arguments are the same. The process of matching and unification can be parallelised under certain conditions. Both functional and logical programming models are used in artificial intelligence applications where parallel processing is very much in demand.

3.9 SOFTWARE TOOLS

Many software tools have been developed and used. Newer machines are getting more and more powerful to support sophisticated tools for development of parallel programs for realistic applications. Some of such tools are discussed here.

3.9.1 Parallelising Compilers

Many years of research and development has been expended on investigation of theory for automatic parallelism of code. Techniques like dependency analysis are used to detect those segments of a code that can be executed in parallel, and then the code is generated to exploit parallelism in the architecture of the machine. The programmer is free of the responsibility of detecting and analyzing the parallelism in the solution to the given problem and then coding accordingly.

3.9.2 UNIX IPC

The inter process communication (IPC) facilities in UNIX operating system lets the programmer develop programs which, when executed run as a collection of cooperating processes. The IPC facility provides a set of machines to share data, control access to shared data, synchronize process and dynamically create and destroy processes. The process can share certain block of memory for sharing and exchanging data while cooperating. Synchronization is achieved by using signaling mechanism supported by operating system.

3.9.3 Threads Model

Conceptually, multithreading is similar to multiprocess programming using IPC. The difference is that a multithreaded program has a single process that manages multiple threads of control executing asynchronously. The thread library provides functional calls to create threads, control threads, terminate threads, control access to shared data through locking mechanism, generate events and wait for events.

The practical advantage of using threads is that threads are lightweight process. In multiprocess model multiple memory images of the program exist in the core while in

multithreaded program this overhead is avoided by sharing the same core image of the process. Also context information for each thread is also maintained.

3.9.4 PVM and MPI

PVM and MPI are two competing and functionally equivalent tools for parallel programming using message passing.

PVM

PVM is a portable message passing programming system designed to link separate Unix host machines to create a virtual machine which is a single computing resource. The PVM system is composed of two parts. The first is a daemon program, which runs on all machines that are part of the virtual machine. This permits the user to run a PVM application at a Unix prompt from any of the machines. The second part of the system is a library of PVM interface routines. This library contains user callable routines for passing messages, spawning processes, coordinating tasks and modifying the virtual machine. Application programs can be written in a mixture of C, C++ and FORTRAN but must be linked with the PVM library.

A virtual machine across a networked collection of computers is realized through a layer of PVM daemons, running on each host. These daemons provide certain functionality such as addressing hosts, addressing of tasks, mapping of tasks to hosts, routing of messages, scheduling tasks on hosts for execution etc. Fig 3.8 shows the architecture of a PVM.

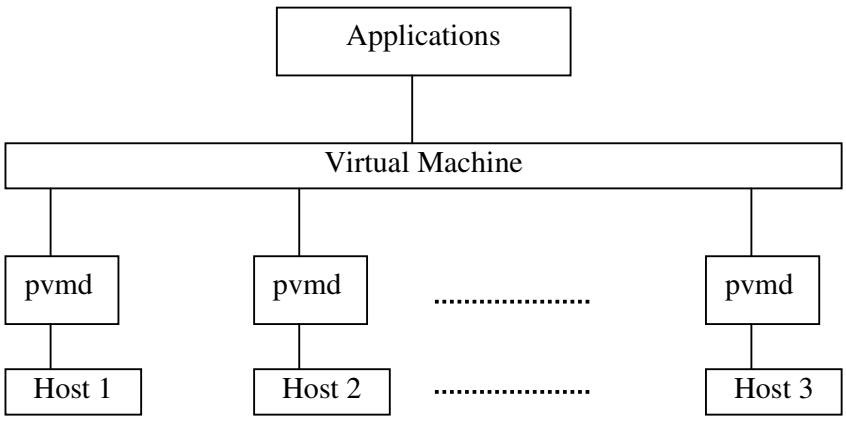


Fig 3.8: Parallel Virtual machine architecture

MPI

In past, as different hardware manufacturers developed their own distributed memory parallel computers, they also developed and implemented their own message passing libraries. So, the programmer was tied to particular computer architecture. This lack of portability lead to a number of public domain message passing libraries being developed, each with their own software implementation. The natural progression from this therefore was the definition of a standard message passing library. In 1992, an MPI forum was established, encompassing hardware and software vendors and researchers from academies, industry and U.S. government to address this issue. By 1994 an initial MPI standard was published and since that time many efficient implementations have been released for all types of parallel architectures.

MPI (Message Passing Interface) provides a standard set of definitions, which allow parallel programs to be written under distributed memory paradigm. These definitions describe a library of over 100 C and FORTRAN subprograms, which are now supported by almost all parallel computer manufacturers. In addition, numerous commercial and public domain implementations of MPI exist, which allow clusters of workstations to be used as a single parallel computing system. The main purpose behind writing such specifications is to develop some software, for the first time since the advent of parallel computers, which is truly portable between different parallel architectures.

When working with a distributed memory computer, it is necessary to ensure that each processor has its own copy of each data item that is required for each computation, performed by that processor. Often, some or all of this data will depend upon the result of a previous computation, which may have been made on one of the other processors. Therefore, some mechanism is required for the transfer of copies of data between processors. This is achieved through the message passing whereby each processor is given the ability to send and receive copies of data to and from other processors. This requires the cooperation of each processor that is involved in communication, whether as a sender or receiver or both. This system provides an API (Application Programming Interface), which permits a standard to develop parallel programs with standard message passing. A carrier such as PVM is required for distinct implementations if the operating system does not support the message passing.

3.9.5 DCE–RPC, Threads

Distributed computing environment (DCE) is a standard proposed by the open software foundation for design and implementation of distributed systems. The standard provides a specification of tools and services that implementor of DCE must provide for the development of distributed computing system. The major components of DCE are RPC (Remote Procedure Call). Time/directory/file services, access authorization features, threads package and interface definition facility. These components provide a formwork for organizing distributed resources into a client server oriented network.

RPC provides a procedural programming abstraction for design and implementation of distributed programs. The services available on geographically distributed machines can be accessed by calling a remote procedure. To bring this interface, the DCE framework requires daemons running on the distributed machines for routing and forwarding data.

3.9.6 CORBA

CORBA (Common Object Request Broker Architecture) is a set of standard mechanism for naming, locating, and defining objects in a distributed computing environment. The formwork provides an object-oriented abstraction for the design and implementation of a distributed computing system. The entire system consists of client server objects, which communicate through the object request broker (ORB). ORB is a set of distributed processes running on different machines and coordinating the communication between the objects.

3.10 PERFORMANCE MEASUREMENT OF PARALLEL PROCESSING

The objective of parallel or distributed processing must be to execute a problem at greater speed and to solve large problems with greater or more dedicated idealization. The performance measurement of parallel processing and other factors affecting its performance are as follows:

3.10.1 Granularity

To achieve improvement in speedup through the use of parallelism, it is necessary to divide the computation into tasks or processes that can be executed simultaneously. The size of a process can be described by its granularity. In coarse granularity, each process contains a large number of sequential instructions and takes a substantial time to execute. In fine granularity, a process might consist of a few or perhaps single instruction. The granularity must be increased to reduce the cost of process creation and inter process communication. For message passing, it is desirable to reduce the communication overhead because of the significant time taken by inter computer communication. This is especially true for the network of workstations.

Granularity = Computation time / Communication time

It is very important to maximize the computation / communication ratio while maintaining sufficient parallelism.

3.10.2 Speedup

A measure of relative performance between a multiprocessor system and a single processor system is the speed up and defined as:

$$S = T_{neq} / T_N$$

Where, Tneq is the time taken by the code to execute on a single processor and T_N is the time taken for the same code to execute on a parallel system having N processors.

Theoretically a problem should run N times faster on a network containing N processors than on a single processor. In reality, the speed up achieved are less than the theoretical values owing to the increase in the communication time with the increase in the number of processors. In addition any load imbalance in the problem will result in less than optimum performance.

3.10.3 Efficiency

The efficiency of the parallel system for a given algorithm is defined as:

$$E = S / N$$

Where, S is the speed up and N is the number of processors.

Efficiency is given as percentage. Efficiency gives the fraction of the time that the processors are being used on the computation. If E = 50%, the processors are being used half the time on the actual computation, on average. The maximum efficiency of 100% occurs when all the processors are being used on the computation at all times and the speed up factor would be N.

3.10.4 Overhead

There are several factors that will appear as overhead in the parallel version and limit the speed up. These factors are:

1.) Periods when not all the processors can be performing useful work and are simply idle (load imbalance).
2.) Extra computations in the parallel version not appearing in the sequential version
3.) Communication time for sending message

3.11 SUMMARY

In present chapter various aspects of parallel processing have been discussed. These aspects include the computer architecture and their suitability with respect to parallel processing, possible way of getting parallelism in sequential machine and parallelism in multiprocessor machine, characteristics of multiprocessor computer for parallel processing. Concept of load balancing has also been discussed in present chapter. Along with these issues, some other issues like operating system characteristics for parallel processing, parallel programming models and software tools for parallel programming have also been discussed. And the final part of chapter includes the method of performance measurement of parallel processing.

In parallel processing, it is necessary that each computer should have copy of required data. Also it is necessary to maintain load on each computer. So frequent data transfer is one of the most important aspect of parallel processing. In order to achieve this task of data transfer various software tools like PVM and MPI are needed. but use of these libraries require knowledge of some concepts of parallel programming like data communication, process forking, process joining, and data synchronization etc. Moreover it requires special debugging for testing the parallel application. Generally, engineers and mathematicians may not be expert in these concepts of parallel processing. In such condition, implementation and performance measurement of parallel processing becomes difficult. So it is the expectation to have some application software, which can provide easy way of implementing parallel processing over network of computers. WebDedip is one of such user friendly distributed processing environment, which helps the user to develop parallel application easily over a network of heterogeneous system. Various aspects of WebDedip environment are discussed in chapter 4.

CHAPTER 4 WebDedip ENVIRONMENT

4.1 INTRODUCTION

As in the parallel processing, processes are executed simultaneously on different computers, connected through LAN, frequent data transfer is necessary for effective performance. These data transfer takes place through messages. Also for load balancing, process starting, and process termination, effective message passing system is necessary. Therefore to achieve estimated performance, suitable message passing tool is required. PVM and MPI are two of such tools. But use of these libraries requires some concepts of parallel programming like data communication, process forking, process joining, and data synchronization etc. moreover it requires special debugging technique for testing the parallel application. Generally, engineers may not be expert in these concepts of parallel processing. In such condition, implementation and performance measurement of parallel processing becomes difficult. So it is general expectation to have some application software, which can provide easy way of implementing parallel processing over network of computers. WebDedip is one of such user friendly distributed processing environment, which helps user to develop parallel application easily over network of heterogeneous system.

4.2 WebDedip OVERVIEW

The WebDedip has three tier architecture; GUI, DedipServer and Agents, as shown in Fig 4.1 The GUI is the web enabled graphical user interface to make the entire user interaction truly system independent. It supports various Java Applets for application configuration, application building, application operation initiation, application progress monitoring, and session controlling. The user initiates the interaction by visiting a predefined site using a standard browser. The standard web server loads the required GUI on the web browser. It has a back-end DedipServer running on the web site. When the GUI submits the request to the DedipServer, it reads the application configuration information from the configuration file. The DedipServer initiates the execution of the first process in the interdependency chart. Normally, most of the applications have a single starting process. If any application has multiple starting processes, it initiates execution of all such independent processes. It informs the agents on the target node to start the execution of the process. The agent sends the status information back to the DedipServer when the process is completed. The DedipServer finds

out the dependent processes on the successful completion of a process and initiates the execution of each such process. The required files are transferred from one node to another. WebDedip has a callable library in Java to interface with the FTP server that helps in transferring files. The required process is automatically inserted in the configuration when IP designer inserts the IO dependency information between two processes. The DedipServer stores complete information about all the applications configured on the web site. The DedipServer exchanges information with the Dedip Backup Server making the model fault tolerant.

The agent accepts requests from the DedipServer, executes them and provides the status information when completed. It has process building (compilation), execution, and monitoring capabilities. It can schedule multiple processes in parallel. It does not control the synchronization among the parallel processes; instead it depends on the DedipServer for this job. It treats each process as a single independent entity. The WebDedip not only caters to the requirements of the application designer, but also addresses all the requirements of the operation manager as well as operators.

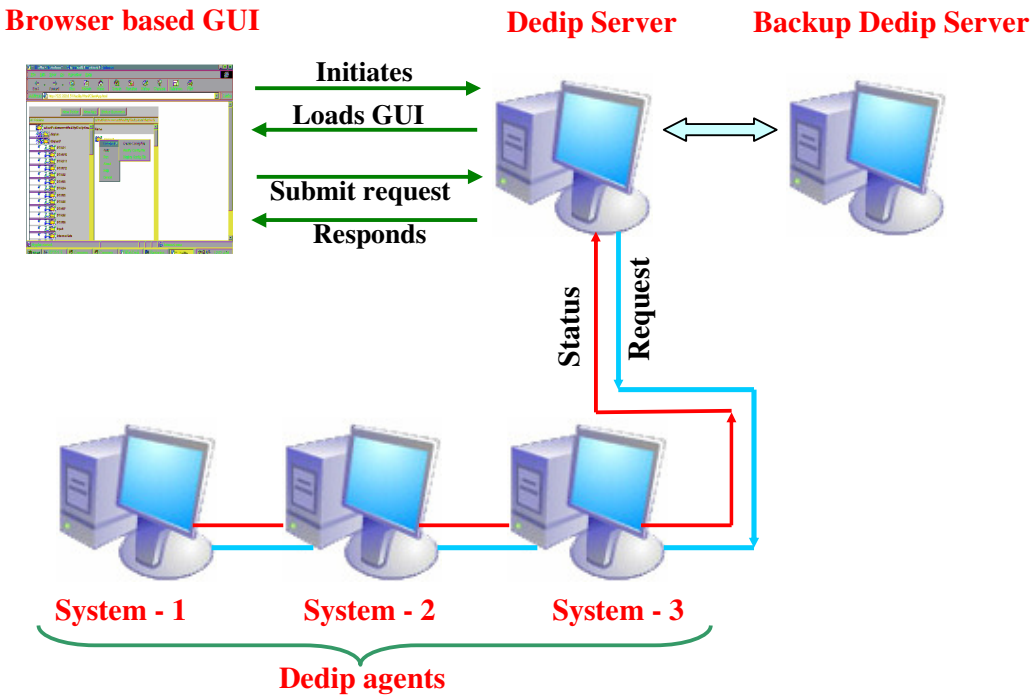


Fig 4.1: Configuration of WebDedip

The application configuration and building is a privileged task, carried out either by the application designer or operation manager. During the regular operations, the operator can initiate any required application, monitor progress, do error handling, and terminate the application, if necessary.

Object-oriented modeling (implemented in Java) is used for the design of the WebDedip. The application is modeled as an object while the process is modeled as an embedded object. The object inter linking capability is used to maintain interdependency information for an application. Java distributed object architecture is used along with the object serialization for network communication among GUI, DedipServer and agents. Hence, WebDedip can be used on a LAN or on Internet. Agents may run on any system over Internet. On start, an agent makes connection with DedipServer on a predefined port and volunteers for computation workload. The Windows-explorer is used as a metaphor in developing the navigation GUI due to its popularity and ease of use.

4.3 INSTALLATION OF WebDedip

The basic requirement for implementation of parallel application is the network of computers, which should contain sufficient computers to meet user's requirement. One computer is made DedipServer. Depending upon the user's wish one computer should be made DedipBackupServer. BackupServer is optional. In case of failure of DedipServer, all the activities allotted to DedipServer will be automatically transferred to the BackupServer and BackupServer will take care of it. Thus more safety can be assured by having DedipBackupServer. Other than DedipServer and BackupServer, Client machines are needed. Number of client machine depends upon the user's need. Various aspects related to WebDedip environment such as hardware and software requirements for Server and Slave machine, components to be installed, and various configurations that has to be made and installation of DedipServer and DedipAgent are dealt in brief in coming topics. Also the steps to be followed to implement parallel application are discussed.

4.3.1 Requirements for WebDedip

Server machine

As far as the hardware requirement of server machine is concerned, minimum Pantium-1 processor is must with minimum of 64 MB RAM. Dedip itself does not require any significant disk space however it requires disk space in the Dedip area as per the respective applications requirements. DedipServer can be installed on any operating system but Windows is preferred. Various software needed for server machine includes JRE-1.2 and later (Java Runtime Environment), JDK-1.2 and later (incase user want to compile the code), Web Server (IIS 4 is preferred), Jakarta Tomcat 4.0 or higher and FTP server.

Slave machine

Hardware requirement of Slave machine is same as the Server machine i.e. Pentium-1with minimum of 64 MB RAM. Like Server, any operating system can work on slave machine. Software requirement of slave machine include JRE-1.2 and later (Java Runtime Environment), JDK-1.2 and later (incase you want to compile the code) and FTP server.

Components

WebDedip needs the following components to be installed:
Dedip server on server
Dedip server on Backup server (optional)
Dedip Agent on slave machine

Configuration

It has following configuration requirements at SERVER and Backup SERVER.
Web Server configuration
FTP server configuration
TOMCAT server configuration

It has following configuration requirements as Slave Machine.

FTP server configuration

Web server configuration

User has to set web root path. It is preferred on drive other than OS drive. So if drive D is used for web root path then wed root folder will be located at D:\inetpub\wwwroot\.

FTP server configuration

To configure FTP server first of all it has to be installed by running eftpserver.exe file. After completion of installation process, user has to create user account. Here user name is WebDedip and password is 'haresh0'. To set home directory user has to select D:\inetpub\wwwroot\facility\dediparea folder. In main setting option user has to select various options to suit the requirements.

DedipServer installation

DedipServer is developed using Java. DedipServer has to be installed on SERVER. It has following components.

1.) Dedip packages
    Dedip
    ftp
    sharableObjects
2.) Dedip configuration files
3.) Dedip area
4.) HTML files

Dedip packages installation and configuration

Various Dedip packages are installed from class files. Following steps can configure these packages:

1.) Create "facility" folder under web root directory (D:\inetpub\wwwroot)

2.) Create “classes” folder under “facility”.

3.) Extract DedipClasses.zip to D:\. Path information is saved in the zip file.

Hence, following directory structure will be available.

D:\inetpub\wwwroot\facility\classes

SharableObjects – containing class files

ftp – containing class files

Dedip- containing class files

Icon- contains imported image files

Config-contains two important configuration files.

-dedip.inf

-dediparea.inf

4.) Create “DedipArea” folder in webroot directory\facility. So it will be at D:\inetpub\wwwroot\facility\DedipArea

5.) Extract Sample test application from DedipArea.zip to D:\. Path information is saved in the zip file. Hence, it will copy the files under home directory. If user doesn’t want to extract sample test applications, he/she has to extract at least following:
D:\inetpub\wwwroot\facility\DedipArea\SessionInformation
D:\inetpub\wwwroot\facility\DedipArea\lib\fileopen.c

6.) Open dedip.inf, located at D:\inetpub\wwwroot\facility\classes\Dedip\config

- The first entry defines the DedipArea path. The default value is set to d:\inetpub\wwwroot\facility\DedipArea
- The second entry defines the Dedip Icon directory. The default value is set as facility/classes/Dedip/icon
- The third entry is the server IP address.
- The forth entry is the Backup Server IP address.

7.) Open dediparea.inf, located at D:\inetpub\wwwroot\facility\ classes\ Dedip \config.
Each row contains three fields

- The first field is the machine IP address
- The second field is the physical location of DEDIPAREA

- The third entry is the operating system of the machine.

To configure HTML files, user has to extract DedipHTML.zip to D:\. It will creats following directory structure.
D:\inetpub\wwwroot\facility\html
D:\inetpub\wwwroot\facility\html\WebDedip

DedipAgent installation

DedipAgent is developed suing Java. Is has to be installed on slave machine. It has following components.

1.) DedipAgent packages
    Dedip
    ftp
    SharableObjects
2.) Dedip configuration files
3.) DedipArea

DedipAgent packages installation and configuration

Various DedipAgent packages are installed from class files. Configure can be done in the same way as it is for DedipServer machine. Various steps to be followed to implement an application on WebDedip can be summarized as follows:

1.) To start the WebDedip Session:
    To start the session, user needs to start DedipServer on server and DedipAgnet on all slave machines. If BackupServer is made then it also has to be start.

2.) To start DedipServer
    The DedipServer has to be run on Server machine. It can be run from DOS prompt. In DOS prompt following command has to be given to run DedipServer.
    cd D:\inetpub\wwwroot\facility\classes
    java –classpath . Dedip.DedipServer

3.) To start DedipAgent

User needs to start DedipAgent on all slave machines. Same as DedipServer, DedipAgent can also be run from DOS prompt. Following command will start the DedipAgent.

cd D:\inetpub\wwwroot\facility\classes

Java –classpath . DedipAgent.DistScheduler

4.) To start DedipBackup server

If Backup Server is made then Backup server can be run from DOS prompt by following command.

cd D:\inetpub\wwwroot\facility\classes

java –classpath . Dedip.DedipServer –Backupof "SERVER_IP"

5.) To configure new application

New application can be configure by following steps:

1.) To configure new application user has to open WebDedip home page (http://SERVER_IP/facility/html/WebDedip)
2.) Click on the ClientApp. It will open the GUI in new window.
3.) Click new application button. It will open a dialog box. In which user has to give application name and description. Select the SAMD option. After clicking OK, user can see the application folder on left hand panel of GUI window.
4.) On clicking the application icon, it will appear in the right hand panel of GUI.
5.) Right click on application icon will show a popup menu. From that menu user has to select config-> Create Config File. It will display a dialog frame with button matrix. The default size is 11X11. User can change the size of matrix by entering the values in matrix field.
6.) When a first row button is clicked on the button matrix window, a dialog box will appear, showing option for Process Detail Form and File Transfer Form. First user has to select Process Detail Form. This will open the Process Detail Form. Various process details such as process name, node on which the process will run and expected time for process has to be filled in this form. After completing process detail form, for first row, another process form has to be filled in third row of matrix. One row should be left between two processes for file transfer process. After filling details for processes, user has to give file transfer information. This information includes, the source and destination process, file name, approximate file size and type of file (i.e. input,

intermediate or output). In this way the basic structure for the application will be ready. Now application is ready to built.

6.) To build the application

Right click on the application icon will be shown the popup menu, having option build. Selection of build option will show a new window. User can build the application either from source or from executables. In case of build from executables, the Dedip will transfer the executables on the target node in appropriate directory. In case of build form source code, the Dedip will transfer the complete source code directory on target node and ask remote DedipAgent to compile code using buil.bat file. Once application is built successfully, user has to click ready button in build dialog box.

7.) To run configured application

After building application successfully, the application icon will be disappeared from right hand panel. So on clicking the application icon on left hand panel of GUI, again it will appear in the right hand panel and right click on application icon will open a popup menu containing an option for run. Selection of run option will start execution of application as per configuration.

8.) To monitor/control the progress of running application

The status of the running application can be known by opening the WebDedip home page (http://SERVER_IP/facility/html/WebDedip). From home page user has to select Operator Console. It will open an operator GUI in new window. Session tab has to be clicked to get summary report of the session. Selection of application from the table and then application tab will give the details about application status.

User can abort/ suspend/ resume/ restart an application. To do so user has to click on the session tab. Select the application and then click button of desired action.

9.) User can terminate the DedipServer using manager console. It provides three level of termination.

Immediate: It kills all processes running on different slave machines.

Process wide: It waits for all processes currently running on different slave machines. Then it terminates. It disables the DedipServer to take new application for processing during the period.

Application wide: It waits for all processes currently running and pending for initiated application. Then it terminates. It disables the DedipServer to take new application for processing during the period.

The DedipServer is capable of restarting the previous session from the point of termination so that loss of processing power is negligible. It can restore the session even in case of Server System crash.

4.3.2 WebDedip Rules

When one runs his/her application through terminal window, it is known that where are input files, intermediate files and output files. WebDedip has to handle the same. So WebDedip suggests to use input directory for input files and intermediate directory for intermediate and output files.

Furthermore, user may wish to create multiple instances of application (i.e. run his/her application more than once simultaneously on different input data). This mode is called Single Application Multiple Data (SAMD). As intermediate files are created in intermediate directory, files will be overwritten by another instance of the application. It is very difficult for user to handle such case.

WebDedip provides a simple solution for this problem. User has to follow the following steps:

1.) Insert the following code just after the include statement

```
FILE *fileopen();
```

2.) Insert the following three lines in the beginning of main function

```
Strcpy(DeipArea,argv[argc-3]);
Strcpy(Application Name,argv[argc-2]);
Counter = atoi(argv[argc-1]);
```

3.) Use “fileopen” function provided by WebDedip to open file instead of stdio.h routine. The fileopen.c is lying at DedipArea\lib on all slave machines.

For example,

f1=fileopen(“Testi.inp”,”r”,DedipArea,ApplicationName,counter,”input”);

f1=fileopen(“Test1.”,”w”,DedipArea,ApplicationName,counter,”intermediate”);

The first two arguments are the same as that of “fopen”. The next arguments are that are supplied by WebDedip to user’s process as argument to main function. The last argument mentions whether the file resides in “input” or “intermediate” directory.
The “fileopen” function does the following:

1.) Prefix the “DedipArea/Application /input” to file, if last argument is “input”.

2.) Prefix the “DedipArea/Application /intermediate” to file, if last argument is “intermediate”. It also concatenates ”_XX” as the end of file name. Where XX represents the number of execution of application. Means user can create 99 instances of his/her application simultaneously.

4.4 PARALLEL IMPLEMENTATION

Sequence of processes for parallel implementation of an application can be enumerated as follows:

Load GUI

As a very first step, user has to initiates by submitting the request for loading GUI. User can load GUI using web browser. Sequential steps for loading GUI are shown in Fig 4.2

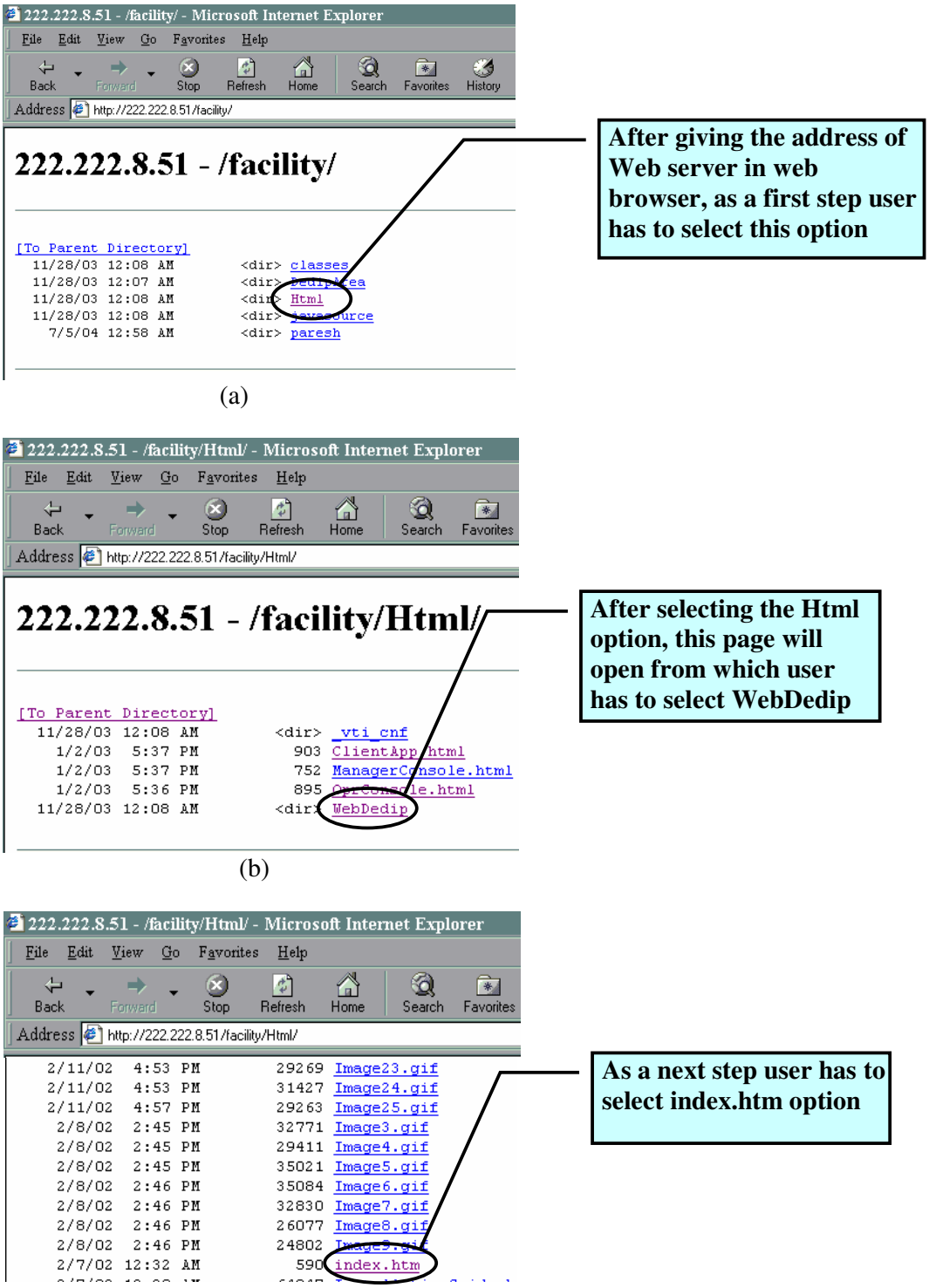


Fig 4.2: (a) (b) (c) Loading GUI for WebDedip

Creating new application

Selecting the New App option from the client app window can create new application. Details to be filled in New App form are as shown in following fig.

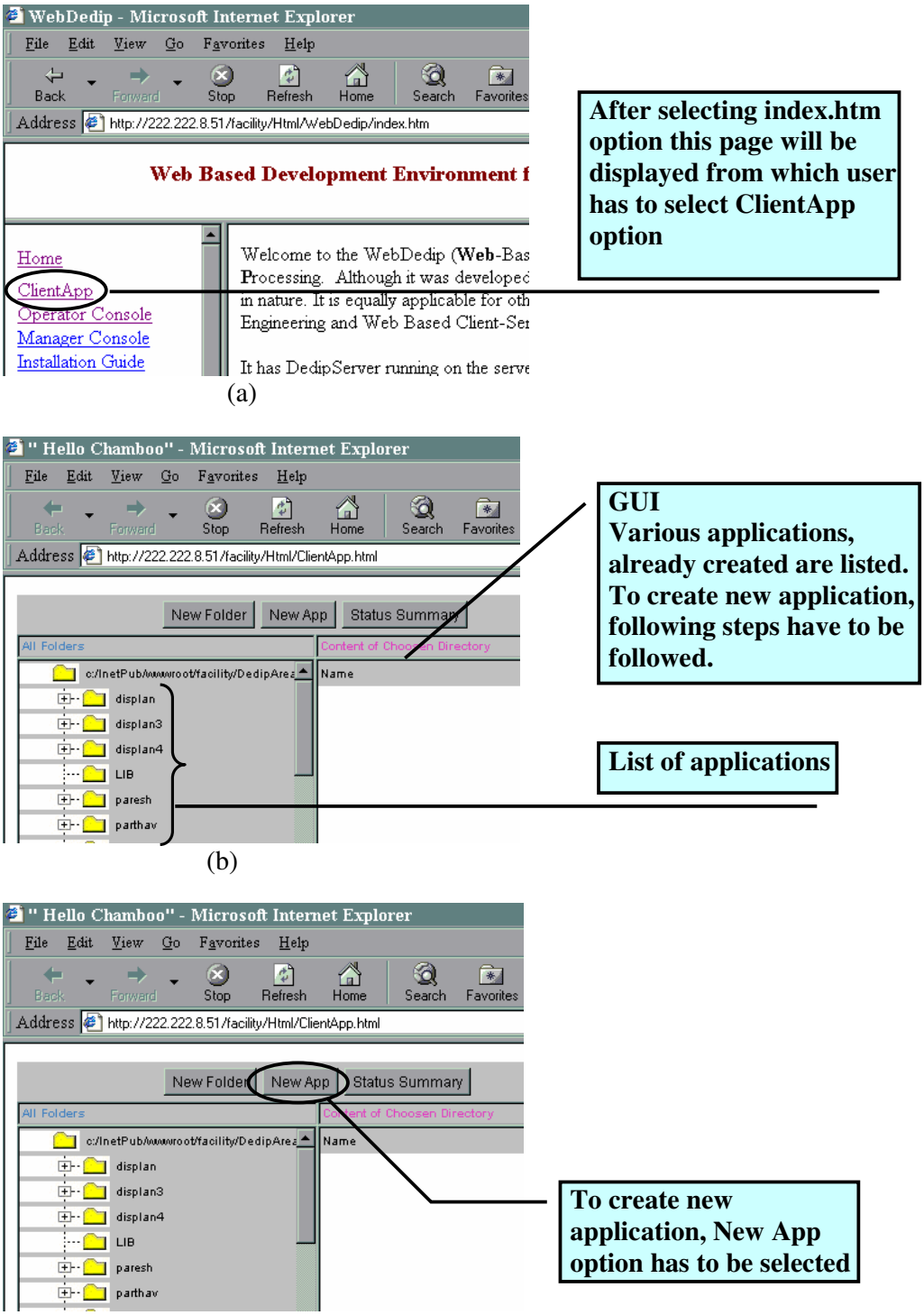


Fig 4.3: (a) (b) (c) New application creation

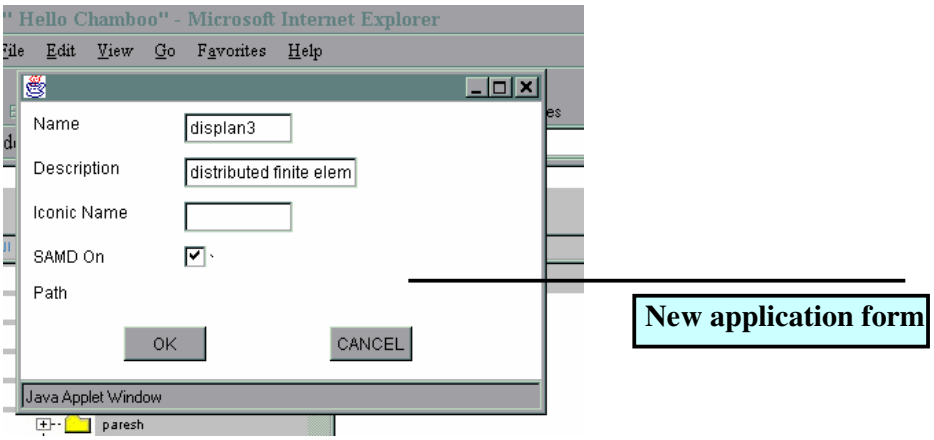


Fig 4.4: New application form

Application Configuration

The application designer has to first decide the configuration of his/her application. It depends on the distributed resource requirement, parallel processing requirement, input/output of each process, etc. The WebDedip supports a nice GUI for the same as shown in figures. Figure shows the overview of the application. The detailed information about each process is shown in figure. The detailed information about file transfer process is also shown in figure. The typical interdependency chart, generated interactively, is shown in figure. The line joining two processes shows their interdependency in top-down model. The input output dependency, if any, is a part of this interdependency and it can be easily configured. User can modify his application configuration file any time by selecting the modify configuration option as shown in fig. The effect of the modification will be applied on next execution of the application.

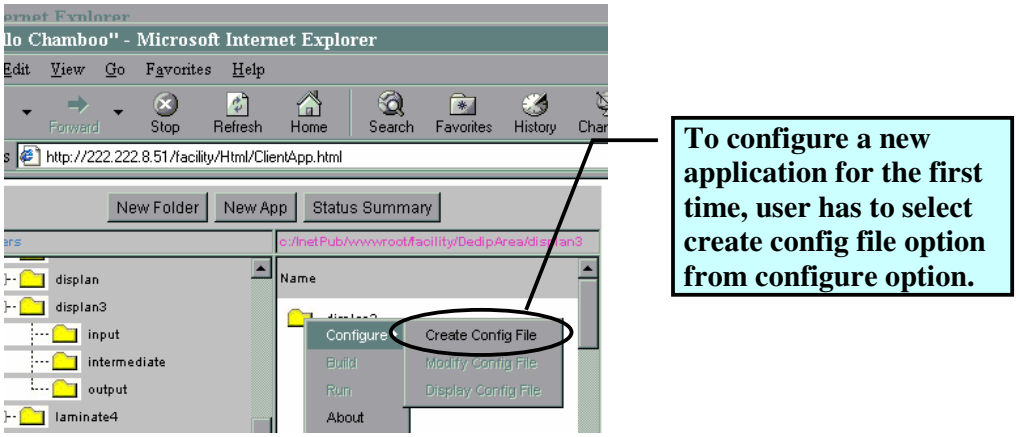


Fig 4.5: Configure new application

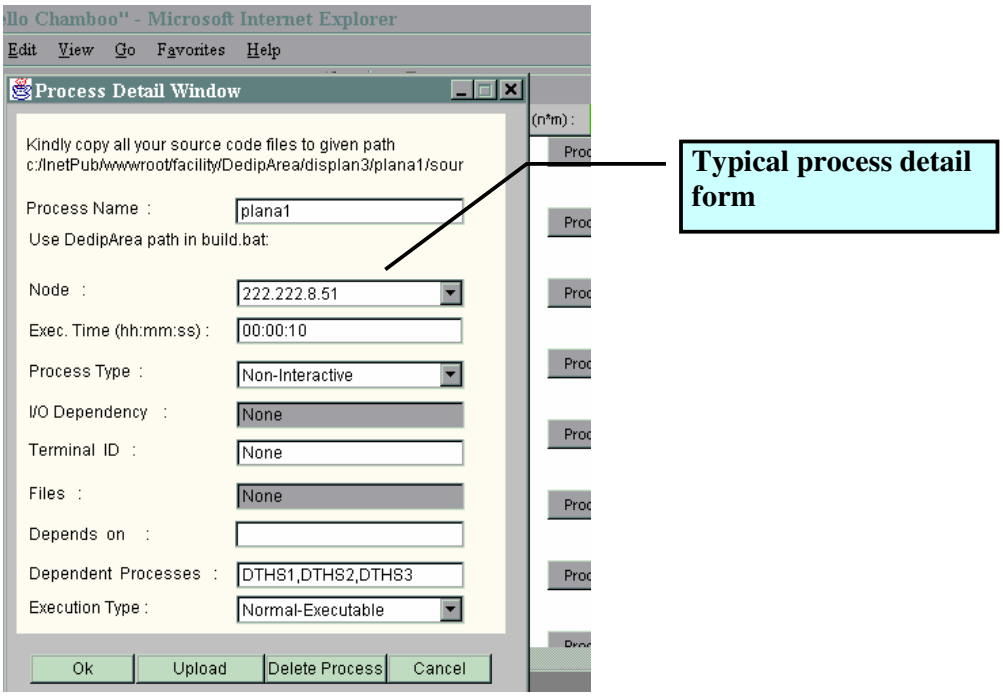


Fig 4.6: Typical process form

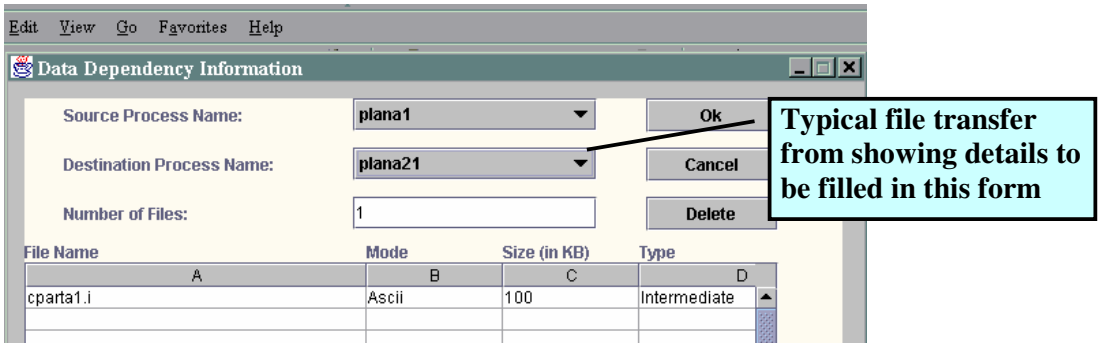


Fig 4.7: Typical file transfer form

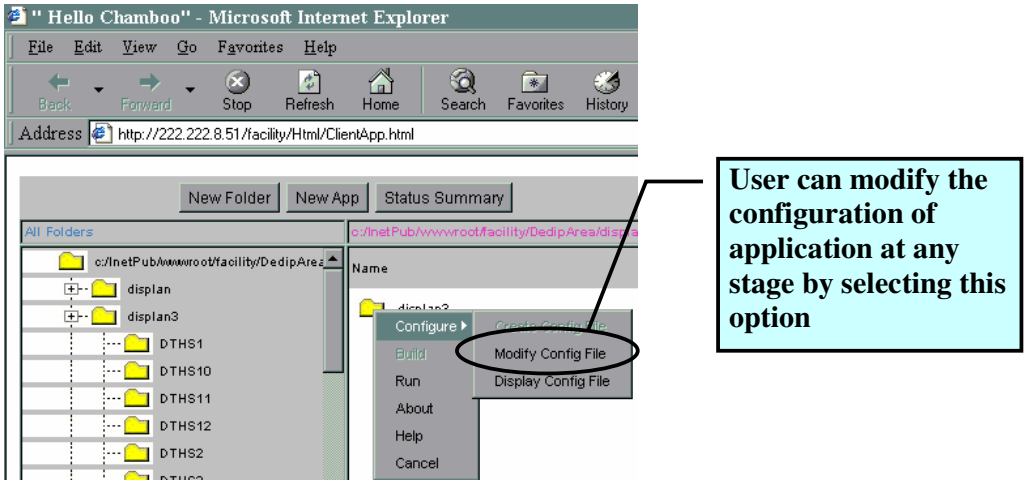


Fig 4.8: Modify configuration file

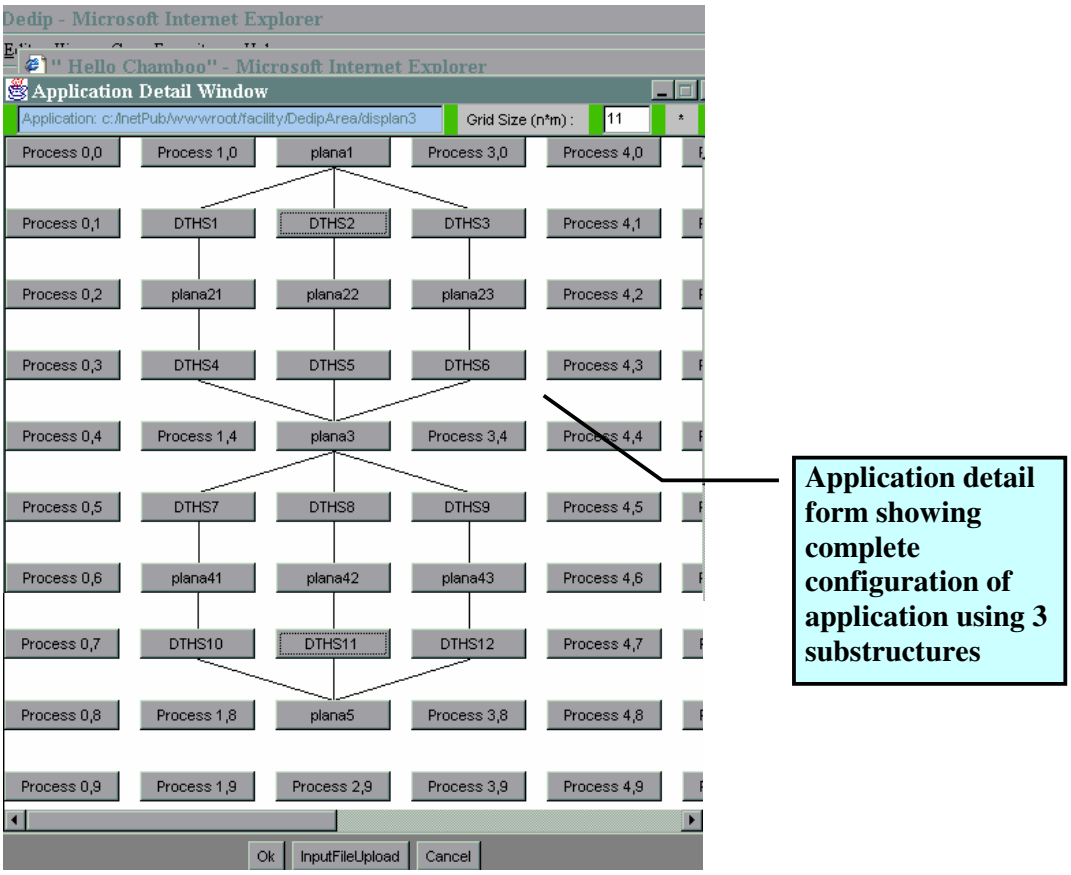


Fig 4.9: Application detail form

Application building

An application consists of many processes. All these processes need to be compiled on the target node. The WebDedip has automated all these compilation. The configuration information has all the required details about each process. The DedipServer copies the source code & make-file, required to build a process, on the target node in a predefined temporary area. It then requests the agent on the node to build the process using the make-file. It carries out this task for each process given in the configuration. The agent creates designated directory and preserves the executable in it. The application designer can build the processes externally on all systems in case he is not willing to give the code. The GUI provides necessary support for such external readiness indicator.

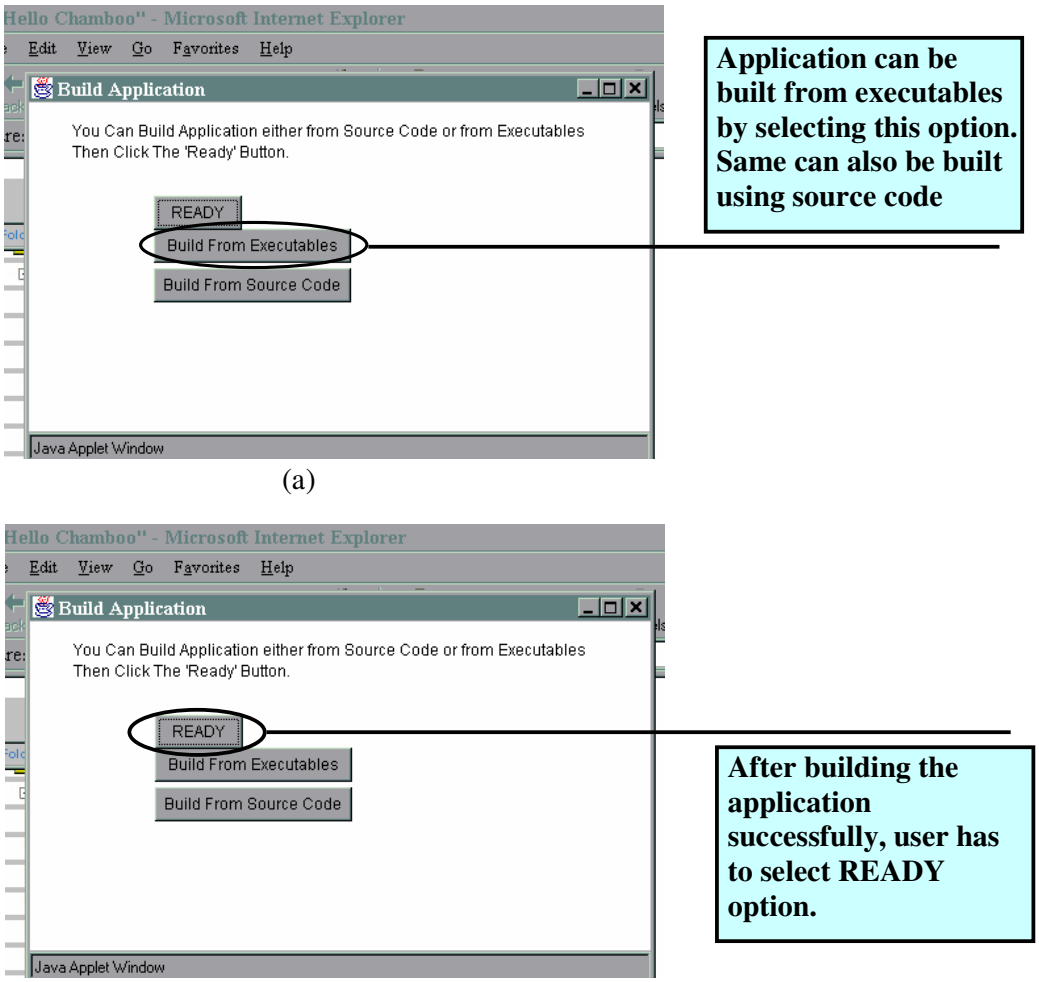


Fig 4.10: (a) and (b) Build application

Application execution and monitoring

The operator can start execution of any application from any machine on the net using the standard browser. GUI displays the configured applications to the operator for selection. Operator can start/abort/suspend/resume an application. Figures show the GUI for session and application progress information.

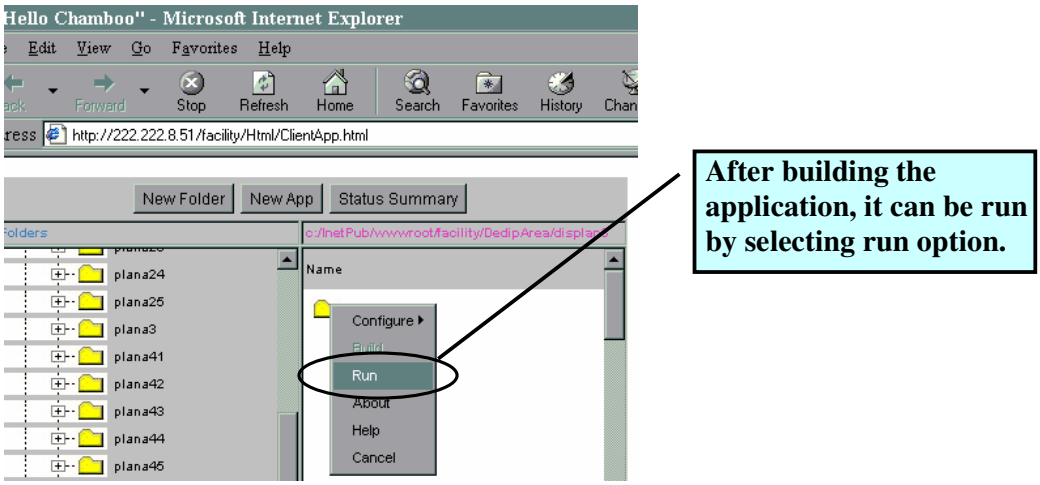


Fig 4.11: Run application

Fig 4.12: Web browser page to open operator console window

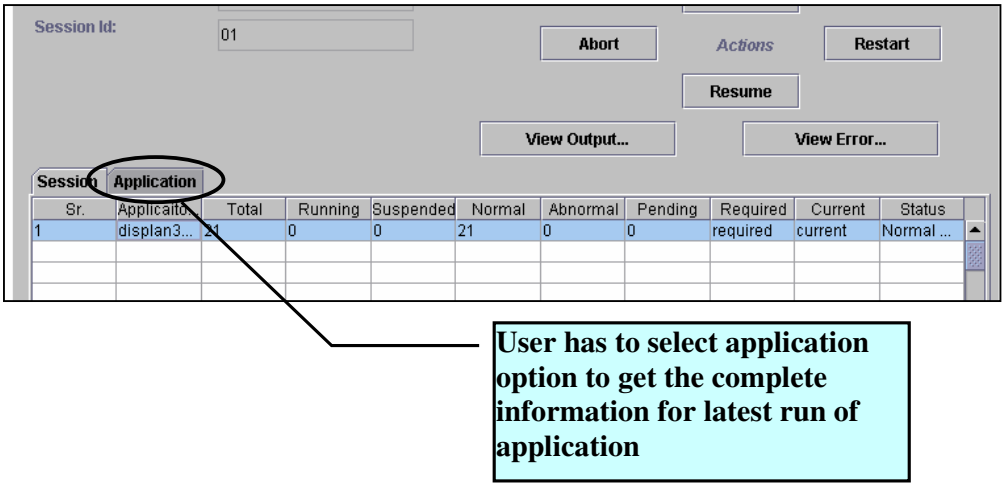


Fig 4.13: Operator console window

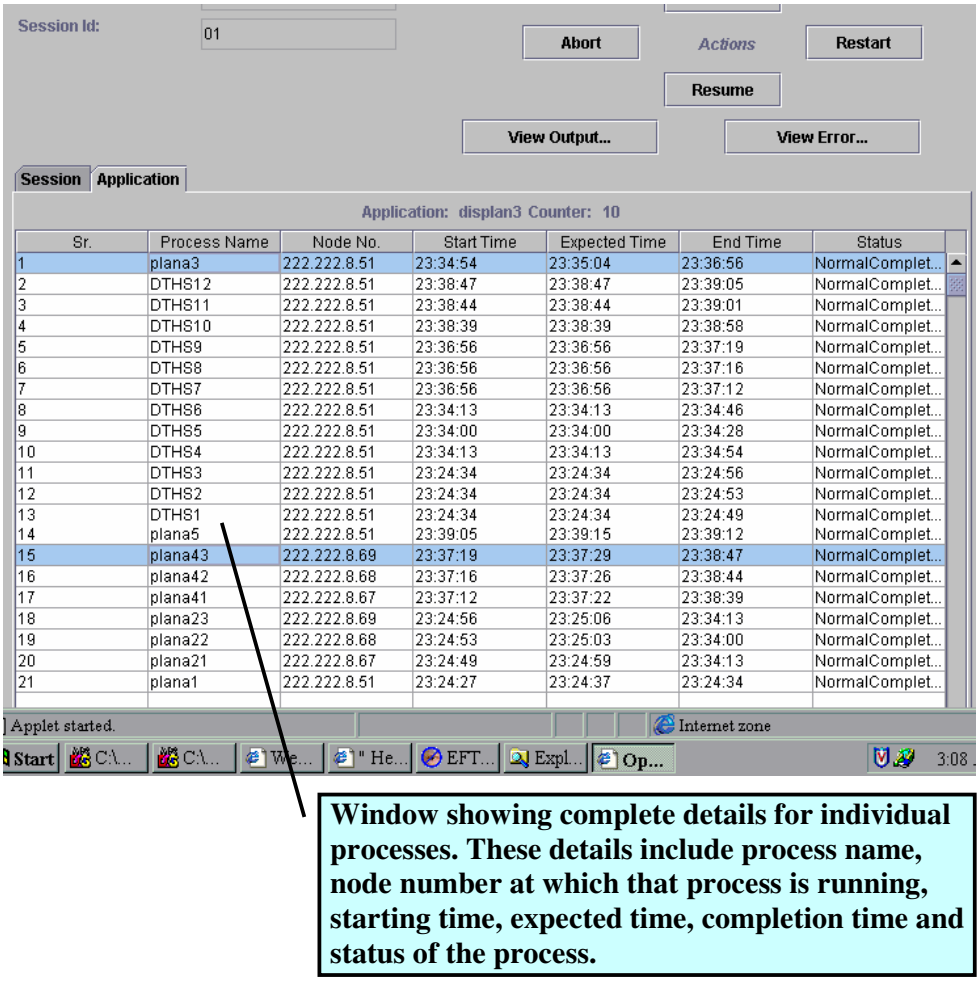


Fig 4.14: Operator console window showing complete process details

Error Handling

In case of abnormal completion, the DEDIP Server displays the error message with error code to the operator. Application designer provides these error codes and error messages. WebDedip keeps this information in the configuration file. The operator can restart the process after taking the necessary actions. In addition, the operator has the options of either restarting the entire application or aborting it.

Session management

Each time an operator logs in, DEDIP scheduler starts/restarts a session for him. Each session has a unique session identification number. It keeps all the information about the session on the server. The operator has multiple options to log out. He can close the session, terminate

the session, suspend/resume the session, or submit the session for progress in background before logging out. He can close the session only after normal completion of all the requests he has submitted. He can terminate the session immediately in case of emergency. In case of termination, the WebDedip kills all the processes of all the requests submitted by the operator irrespective of the status. The background processing is very effective in the case of non-interactive processes. The WebDedip gives the detailed status to the operator at the next logon.

WebDedip system management

The WebDedip system consists of a DedipServer and agents. The DedipServer can detect the agent termination. It displays the message on operators' console as well as operation manager console. The DedipServer is the most important process in the entire system. Its failure, for example, due to system crashing, can cause a severe problem. Dedip Backup Server is designed to handle the failure of the DedipServer. The software package Dedip backup Server runs on the machine of the backup server and duplicates the required information from the DedipServer. An agent sends a trigger to Dedip Backup Server when it fails to communicate with the DedipServer. The Dedip Backup Server validates the DedipServer failure. It takes over the complete responsibility from that moment onwards and informs the operation manager. The servers are exchanging information only in case of external events like termination of process, start of new process, initiation of an application by the operator, the start of new session, etc. The frequency of such possible events is very low. Furthermore, the volume of the information is negligible. Hence, the communication overhead for maintaining the back-up server is very low.

4.5 SUMMARY

In present chapter various aspects of WebDedip environment have been discussed. These aspects are architecture of WebDedip, installation of various components on server and slave machine and minimum software and hardware requirement of computer for installation of WebDedip. In later part of chapter, stepwise procedure of application implementation on WebDedip is discussed with the help screen shots.

5.1 GENERAL

There are many problems in the engineering, which can be treated as plane stress or plane strain cases [30]. Some of the examples of such problems are as shown in following figures.

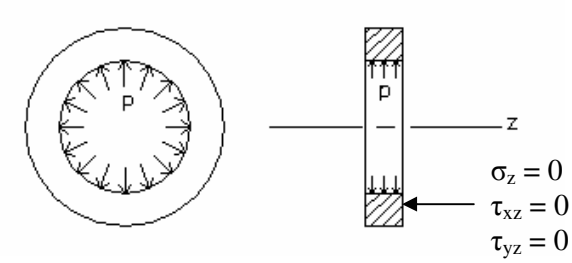


Fig 5.1: Plane stress problems

The plane stress condition is characterized by very small dimensions in one of the normal directions. Fig 5.1 shows an example of plane stress conditions. In these cases, the stress components σ_z , τ_{xz} and τ_{yz} are zero and it is assumed that no stress component varies across the thickness. The state of stress is then specified by σ_x, σ_y and T_{xy} only and is called plane stress.

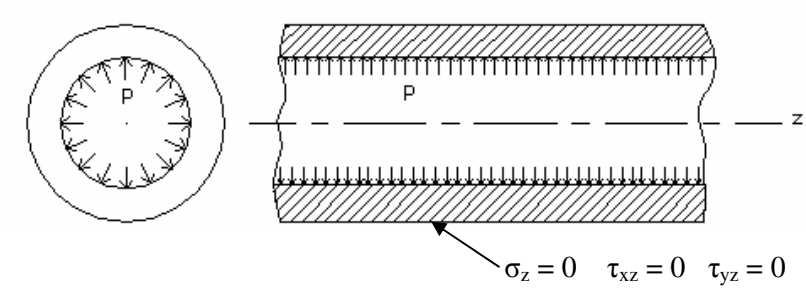


Fig 5.2: Plane strain problems

Problems involving long bodies whose geometry and loading do not vary significantly in the longitudinal direction are referred to as plane strain problems. Fig 5.2 shows some example of plane strain condition. In these situations, a constant longitudinal displacement corresponding to a rigid body translation and displacements linear in z corresponding to a rigid body rotation do not result in strain. Hence, if we consider a cross section away from the ends, it can be assumed that w = 0 and displacements u and v are functions of x and y but independent of z. So, $\varepsilon_x = \gamma_{zx} = \gamma_{yz} = 0$

5.2 FINITE ELEMENT FORMULATION

Finite Element formulation for CST element is discussed below.

5.2.1 Shape functions

For the descritization, CST elements (Fig 5.3) with linear displacement model having two degree of freedoms at each node are selected. Therefore total degree of freedom is six. The vector $q = [u_1, v_1, u_2, v_2, u_3, v_3]$ represents the displacement vector and hence the degree of freedom at each node is enumerated as [22]:

u_1 = Displacement at node 1 in X direction

v_1 = Displacement at node 1 in Y direction

Selecting displacement function for displacement in X and Y direction as follows:

$$u = c_1 + c_2x + c_3y$$

$$v = c_4 + c_5x + c_6y \qquad (5.1)$$

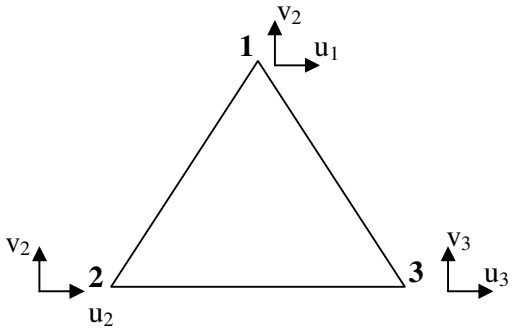


Fig 5.3: CST element having two DOF at each node

$$u = \begin{bmatrix} 1 & x & y \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} \qquad (5.2a)$$

$$v = \begin{bmatrix} 1 & x & y \end{bmatrix} \begin{bmatrix} c_4 \\ c_5 \\ c_6 \end{bmatrix} \qquad (5.2b)$$

Shape functions for displacement in X direction are,

$$N_1 = (a_1+b_1x+c_1y) / 2A$$

$$N_2 = (a_2+b_2x+c_2y) / 2A$$

$$N_3 = (a_3+b_3x+c_3y) /2A \qquad (5.3)$$

For displacement in Y direction also the shape functions will remain same.

5.2.2 Strain-Displacement relation matrix [B]

The relation between strain and displacement is expressed as,

$$[\varepsilon] = [B]\,[q] \tag{5.4}$$

Where, [ε] = strain matrix

[B] = strain displacement matrix

[q] = nodal displacement matrix

$$\begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{bmatrix} = \begin{bmatrix} \partial u / \partial x \\ \partial v / \partial y \\ \partial u / \partial y + \partial v / \partial x \end{bmatrix} = \begin{bmatrix} \partial / \partial x & 0 \\ 0 & \partial / \partial y \\ \partial / \partial y & \partial / \partial x \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \tag{5.5}$$

$$\begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{bmatrix} = \begin{bmatrix} \partial / \partial x & 0 \\ 0 & \partial / \partial y \\ \partial / \partial y & \partial / \partial x \end{bmatrix} \begin{bmatrix} N_1 & 0 & N_2 & 0 & N_3 & 0 \\ 0 & N_1 & 0 & N_2 & 0 & N_3 \end{bmatrix} \begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \end{bmatrix} \tag{5.6}$$

Equating the above equation 5.6 with equation

[ε] = [B] [q], we have

$$[B] = \begin{bmatrix} \partial N_1/\partial x & 0 & \partial N_2/\partial x & 0 & \partial N_3/\partial x & 0 \\ 0 & \partial N_1/\partial y & 0 & \partial N_2/\partial y & 0 & \partial N_3/\partial y \\ \partial N_1/\partial y & \partial N_1/\partial x & \partial N_2/\partial y & \partial N_2/\partial x & \partial N3/\partial y & \partial N_3/\partial x \end{bmatrix} \tag{5.7}$$

Therefore,

$$[B] = \frac{1}{2A} \begin{bmatrix} b_1 & 0 & b_2 & 0 & b_3 & 0 \\ 0 & c_1 & 0 & c_2 & 0 & c_3 \\ c_1 & b_1 & c_2 & b_2 & c_3 & b_3 \end{bmatrix} \tag{5.8}$$

Where,

$b_i = y_j - y_k$ $\quad c_i = x_k - x_j$

$b_j = y_k - y_i$ $\quad c_j = x_i - x_k$

$b_k = y_i - y_j$ $\quad c_k = x_j - x_i$

5.2.3 Element stiffness matrix

We know that the stiffness matrix, derived from Principle of virtual displacement and principle of stationary potential energy [18], can be given as

$$[K] = \iiint_v [B]^T[D][B]\, dv \qquad (5.9)$$

Where, [K] = Stiffness matrix [B] = Strain displacement matrix

[D] = constitutive law matrix

[B] Matrix has already been derived.

For plane stress condition, Constitutive law matrix [D] can be given as follows:

$$[D] = \frac{E}{(1-\nu^2)} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & (1-\nu)/2 \end{bmatrix} \qquad (5.10)$$

For plane strain condition, Constitutive law matrix [D] can be given as follows:

$$[D] = \frac{E}{(1+\nu)(1-2\,\nu)} \begin{bmatrix} 1-\nu & \nu & 0 \\ \nu & 1-\nu & 0 \\ 0 & 0 & (1-2\nu)/2 \end{bmatrix} \qquad (5.11)$$

Where, E = Modulus of Elasticity and ν = Poisson's ratio

$$[K] = \iiint_v [B]^T[D][B]\, dv$$

If the thickness of the element is constant value 't', the stiffness matrix is given by

$$[K] = t \iint [B]^T[D][B]\, dx\, dy$$

It can be observed that the matrix [B] is constant and the matrix [D] depends on the plane stress or plane strain condition and on material property. Hence stiffness matrix of the element can be written as follows:

$$[K] = t\, [B]^T[D][B] \iint dx\, dy$$

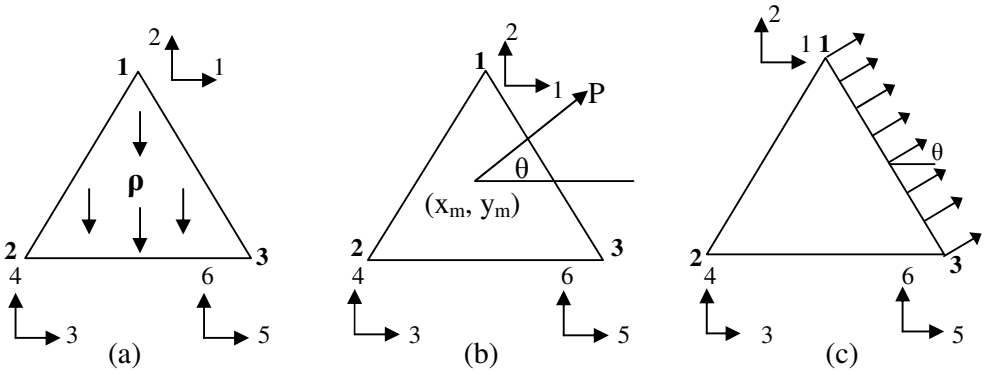
$$[K] = [B]^T[D][B]\, A\, t \qquad (5.12)$$

Where, A = area of element and can be calculated as follows

$$2A = \text{Det} \begin{pmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{pmatrix}$$

5.2.4 Consistent Load vector

The components of the consistent load vector are the equivalent load applied at the nodal points of the element due to the loads applied at the intermediate points of a finite element. The applied external forces may consist of independent or combination of the following load cases.



(a) Gravity loading
(b) Nodal loading
(c) Edge loading

Co ordinates
1 (x_1, y_1)
2 (x_2, y_2)
3 (x_3, y_3)

Fig 5.4: Various lading conditions for CST element

1.) Gravity load

Consider the CST element as shown in Fig 5.4 (a). The gravity loads, generally the self-weight of the element, always acts in the gravitational direction e.g. in the negative Y direction. This load is equally distributed at all three nodes. So the load vector for gravity load can be written as follows:

$$P = \begin{bmatrix} 0 \\ \rho At/3 \\ 0 \\ \rho At/3 \\ 0 \\ \rho At/3 \end{bmatrix}$$

Where,
ρ = Density of material
A = Area of element
t = Thickness of element

(5.13)

2.) Point load

Consider the CST element as shown in Fig 5.4 (b). As we know that the load vector is given by equation,

$$P = \int [N]^T p \, dx$$

Where,

P = load vector

p = nodal load acting at a point

[N] = shape function at point of application of nodal load (xm, ym)

Shape functions at point of application of nodal load (xm, ym) can be calculated as follows:

$$[N] = \begin{bmatrix} 1 & x_m & y_m \end{bmatrix} \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix}^{-1}$$

Load vector can be calculated as,

$$P = \begin{bmatrix} N_1 & 0 \\ 0 & N_1 \\ N_2 & 0 \\ 0 & N_2 \\ N_3 & 0 \\ 0 & N_3 \end{bmatrix}_{(x_m,\, y_m)} \begin{bmatrix} P_x \\ P_y \end{bmatrix}$$

Where,
$P_x = P\cos\theta$
$P_y = P\sin\theta$

(5.14)

3.) Edge force

The load vector is given by

$$P = \int [N]^T p \, dx$$

As the edge force is acting on a edge (Fig 5.4 (c)), it can be treated as a 2-noded bar element. Load vector for bar element is given by

$$P = p \begin{bmatrix} L/2 \\ L/2 \end{bmatrix}$$

where,
L = Length of edge on which force is acting

Therefore, final load vector due to load acting at angle θ can be given as,

$$P = \begin{bmatrix} Pl/2\cos\theta \\ Pl/2\sin\theta \\ 0 \\ 0 \\ Pl/2\cos\theta \\ Pl/2\sin\theta \end{bmatrix} \qquad (5.15)$$

5.3 SUBSTRUCTURE TECHNIQUE FOR ANALYSIS

As the problems became more and more complex, various new analysis techniques were developed for easy and fast solution of the problems. Earlier simple analytical methods were available but later as the need arise, various new analytical methods were developed which were faster and easier in programming as compared to earlier methods. Today, computers having very high computing power are available but there are various complex problems for which computational time is very high. Furthermore due to large size of problem, memory requirement for storage is also more then the storage available. In such cases it becomes difficult to carryout analysis of complex structures. So, some way should be there which can overcome such problems. Substructuring technique for analysis is one of such solution by using which above problems can be solved to large extent.

For analysis of large structural systems, substructuring technique has been found useful. This technique is also cost effective to incorporate modification in certain parts of structures and

also for nonlinear analysis situations. The method of substructuring for static structural analysis is based on subdividing the large structure into smaller parts, which are analyzed separately to obtain the relationship between forces and displacements at the common boundaries. These boundary variables are then determined and are used to obtain the unknown within each substructure. A particular structure can be modeled with different choice of substructures so some basic idea of substructuring is needed from user's part to choose the best model amongst these.

The method of sub structuring for static structural analysis is based on subdividing the large structure into smaller parts, known as substructures, to obtain the relationship between forces and displacements at common boundaries. These boundary variables are then determined and used to obtain the unknowns within each substructure.

The basic equilibrium equation, which is used to find out the displacements in the structure, can be given by

$$[K]\,[r] = [P] \tag{5.16}$$

Where, [K] is the stiffness matrix, [r] is the displacement matrix and [P] is the load vector. In substructure technique, above equilibrium equation is obtained by the assemblage of substructure equations. For each substructure, the stiffness matrix, the displacement matrix and the load vector are partitioned corresponding to internal and boundary degree of freedom $\{d_i\}$ and $\{d_b\}$ respectively as follows:

$$\left(\begin{array}{c|c} K_{ij} & K_{ib} \\ \hline K_{bi} & K_{bb} \end{array}\right) \quad \left\{\begin{array}{c} d_i \\ \hline d_b \end{array}\right\} = \left\{\begin{array}{c} Q_i \\ \hline Q_b \end{array}\right\} \tag{5.17}$$

In the above equation, a boundary node is defined as a node, which is part of more than one substructure and the degree of freedom at the boundary nodes are termed as boundary degrees of freedom.

The analysis can be performed in two stages,

1.) Considering degrees of freedom at boundaries as fixed, each substructure is analyzed on different computers in parallel. Denoting the solution obtained from this step by a superscript α.

2.) Combine the condensed stiffness of the substructures from different computers to get the global structure stiffness matrix and analyze the assemblage by releasing the boundary degree of freedom. Denoting the solution obtained in this step by a superscript β.

The displacement and load vectors can be expressed as the sum of the above two cases as,

$$\left\{\begin{array}{c} d_i \\ \hline d_b \end{array}\right\} = \left\{\begin{array}{c} d_i^{\alpha} \\ \hline d_b^{\alpha} \end{array}\right\} + \left\{\begin{array}{c} d_i^{\beta} \\ \hline d_b^{\beta} \end{array}\right\} \tag{5.18}$$

And

$$\left\{\begin{array}{c} Q_i \\ \hline Q_b \end{array}\right\} = \left\{\begin{array}{c} Q_i^{\alpha} \\ \hline Q_b^{\alpha} \end{array}\right\} + \left\{\begin{array}{c} Q_i^{\beta} \\ \hline Q_b^{\beta} \end{array}\right\} \tag{5.19}$$

Where, subscript i and b denoted the terms corresponding to the internal and boundary degree of freedom respectively. Obviously, as $\{d_b^{\alpha}\}$ is the displacement at the boundary degree of freedom, when boundaries are fixed it will be zero. Thus

$$\{d_b^{\alpha}\} = 0 \tag{5.20}$$

Also in the first stage of analysis, all the forces are applied at the internal nodes of the substructure and hence these forces do not appear at the second stage. Hence,

$$\{Q_i^{\beta}\} = \{0\} \quad \text{and} \quad \{Q_i^{\alpha}\} = \{Q_i\} \tag{5.21}$$

STAGE 1: Analysis with fixed boundaries

Substituting the vale of $\{d_b^{\alpha}\}=\{0\}$ from equation 5.20 into the equilibrium equation 5.17, the set of equations for the first stage of analysis with boundaries of substructure fixed can be written as,

$$\left[\begin{array}{c|c} K_{ii} & K_{ib} \\ \hline K_{bi} & K_{bb} \end{array}\right] \left\{\begin{array}{c} d_i^{\alpha} \\ \hline \{0\} \end{array}\right\} = \left\{\begin{array}{c} Q_i \\ \hline Q_b^{\alpha} \end{array}\right\} \tag{5.22}$$

Solving the first set of above equation,

$$\{d_i^{\alpha}\} = [K_{ii}]^{-1} \{Q_i\} \tag{5.23}$$

Substituting the value of $\{d_i^{\alpha}\}$ in the second equation

$$\{Q_b^{\alpha}\} = [K_{bi}] [K_{ii}]^{-1} \{Q_i\} \tag{5.24}$$

Here, $\{Q_b^{\alpha}\}$ is the force required to be applied at the substructure boundaries to keep the boundary displacements equal to zero. The above analysis is performed in all the substructures in parallel on different computers.

STAGE 2: Analysis with boundaries released

Again substituting the value of $\{Q_b^{\beta}\}$ in equation 5.17, the set of equations for the second stage of analysis with boundaries released can be written as,

$$\left[\begin{array}{c|c} K_{ii} & K_{ib} \\ \hline K_{bi} & K_{bb} \end{array}\right] \left\{\begin{array}{c} d_i^{\beta} \\ \hline d_b^{\beta} \end{array}\right\} = \left\{\begin{array}{c} \{0\} \\ \hline Q_b^{\beta} \end{array}\right\} \tag{5.25}$$

Solving the first set of equations, we have

$$\{d_i^{\beta}\} = - [K_{ii}]^{-1} [K_{ib}] \{d_b^{\beta}\} \tag{5.26}$$

Solving the second set of equations, we get

$$[K_{ib}] \{d_i^{\beta}\} + [K_{bb}] \{d_b^{\beta}\} = \{Q_b^{\beta}\} \tag{5.27}$$

Substituting from equation 5.26 for $\{d_i^{\beta}\}$ into equation 5.27, we get

$$-[K_{bi}] [K_{ii}]^{-1} [K_{ib}] \{d_b^{\beta}\} + [K_{bb}] \{d_b^{\beta}\} = \{Q_b^{\beta}\} \tag{5.28}$$

or,
$$[K^*] \{d_b^{\beta}\} = \{Q_b^{\beta}\} \tag{5.29}$$

where,
$$[K^*] = [K_{bb}] - [K_{bi}] [K_{ii}]^{-1} [K_{ib}] \tag{5.30}$$

The equation 5.29 is the equilibrium equation for the substructure in terms of its boundary degree of freedom and $[K^*]$ is the corresponding stiffness matrix called as condensed stiffness matrix. This analysis is carried out in parallel for all substructures on different computers and the condensed stiffness matrix for each substructure are assembled to form the global structure stiffness matrix. Thus,

$$[K] = \sum_{s=1}^{s=n} [K^*]_s \tag{5.31}$$

and

$$\{P\} = \{Q_b\} - \sum_{s=1}^{s=n} \{Q_b^{\alpha}\}_s \tag{5.32}$$

In the above equations n stands for the number of substructures, which is equal to the number of computers. The assemblage of the substructures through equation 5.31 and 5.32 leads to equation 5.16 where all the degrees of freedom are along the common boundaries of the substructures. Solution of equation 5.16 gives the global displacements along the boundaries of the substructure. Now the vector $\{d_b^{\beta}\}$ can be obtained for each substructure, which will be communicated to different computers and from that $\{d_i^{\beta}\}$ can be determined using equation 5.26. Thus all values of {d} required in equation 5.18 are known for each substructure and from that other quantities like element forces, stresses and strains can be calculated.

5.3.1 Stepwise approach for substructure technique

In present study, Finite Element Analysis method is used for the analysis and so substructuring technique is implemented for Finite Element Analysis of structures. Using basic approach of the substructuring technique analysis can be carried out in five phases. Various processes related with these phases can be summarized as follows:

1.) Data generation for individual substructure.
2.) Calculation of condensed stiffness matrix and condensed load vector, using concept of static condensation.
3.) Calculation of boundary degree of freedoms for each substructure.
4.) Calculation of internal degree of freedoms for each substructure using boundary degree of freedoms.
5.) Collection of results for each substructure and giving final result.

Out of above five processes, first process of data generation is a sequential process so carried out on single computer. After generating data for each substructure, stiffness matrices and load vectors for each substructure is calculated simultaneously on different computers. Each computer is having required data corresponding to the substructure allotted to that computer. Again the thirds process of assembly of matrices is a sequential process (i.e. carried out on single computer). So the completion of second process on all computers is necessary before beginning of third process. After calculation of DOF corresponding to boundary nodes, these DOFs are distributed again to corresponding computers. So the forth process is again the parallel process. And finally, after calculating internal DOFs, results are collected form each computer to have final results.

STEP: 1

Following fig shows a plate, which is fixed as one end and subjected to tension force on the other end. It is required to carry out the finite element analysis. For analysis CST element is used. So the Fig 5.5 shows the descritization of plate into CST elements. For time being plate is descritized into 32 no of elements and there are total 25 no of nodes.

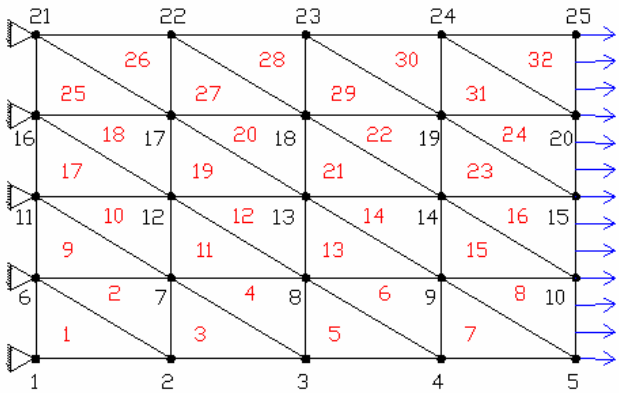


Fig 5.5: Structure descretized into CST elements

In order to carryout the analysis using substructure technique, the plate is divided into 4 no of substructures. Fig 5.6 shows the plate divided into four substructures. In order to calculate it is necessary to differentiate internal and boundary nodes. Here, each substructure is having 1 internal node and 8 boundary nodes. Node 7, 9, 17 and 19 are the internal nodes for 1st, 2nd, 3rd and 4th substructures respectively. Along with the internal and boundary nodes, other data such as the restrained conditions and the load data are also supplied individually for each substructure.

STEP: 2

As discussed in the theory of the substructure technique, first the analysis is carried out by restraining the boundary of each substructure. So following fig shows all substructures with restrained boundaries. In this step of analysis, nodes are renumbered. As shown in Fig 5.7, first all the internal nodes are numbered and then boundary nodes are numbered.

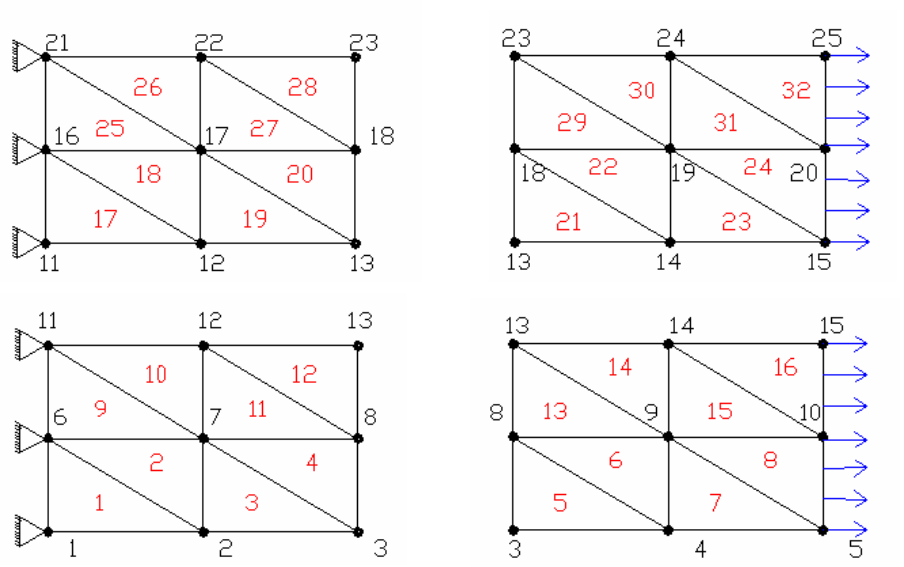


Fig 5.6: Structure divided into four substructures

After renumbering of nodes, static condensation is carried out in which internal nodes are eliminated from each substructure. Calculation of condensed stiffness matrix and condensed load vector is carried out using equations 5.15 and 5.17 respectively, as discussed in previous section. So the whole substructure will be treated as a single element. Fig 5.8 shows DOF corresponding to boundary nodes for each substructure. Here actual restrained conditions are also implemented as all substructure stiffness matrices and load vectors are to be assembled to get global stiffness matrix.

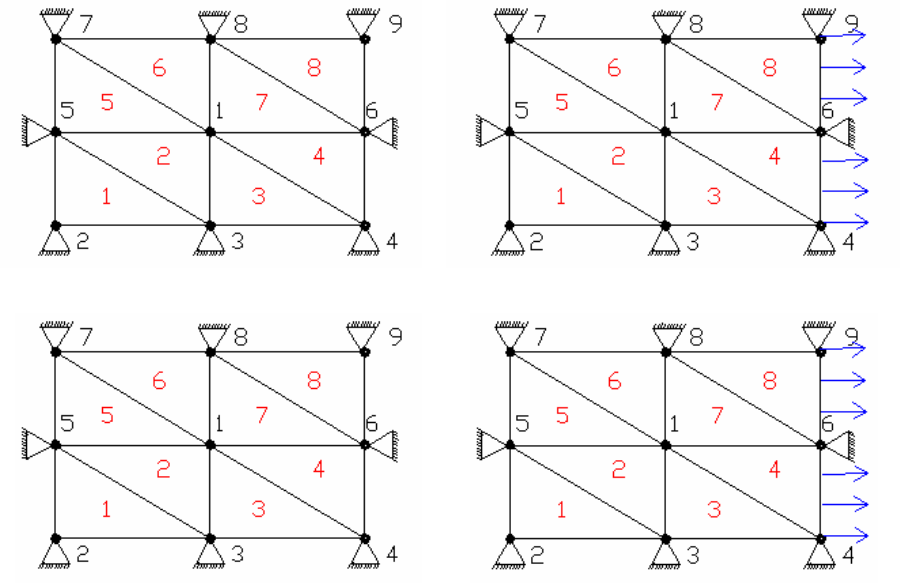


Fig 5.7: Substructures with all boundaries fixed

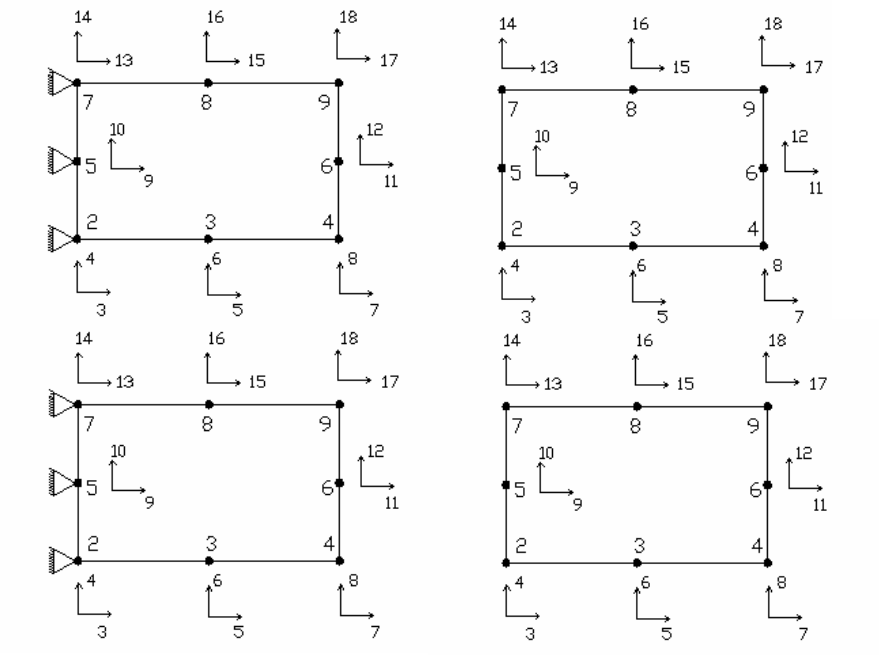


Fig 5.8: Substructures after static condensation

STEP: 3

After calculation of condensed stiffness matrices and load vectors for each substructure, these matrices are assembled according to the boundary nodes. Fig 5.9 shows the nodes and corresponding DOF, which are incorporated in the global stiffness matrix. Nodes 1, 6, 11, 16 and 21 are eliminated, as retrained and also internal nodes are not considered. After assembly of stiffness matrix and load vector, DOF corresponding to boundary nodes (shown in fig) are calculated.

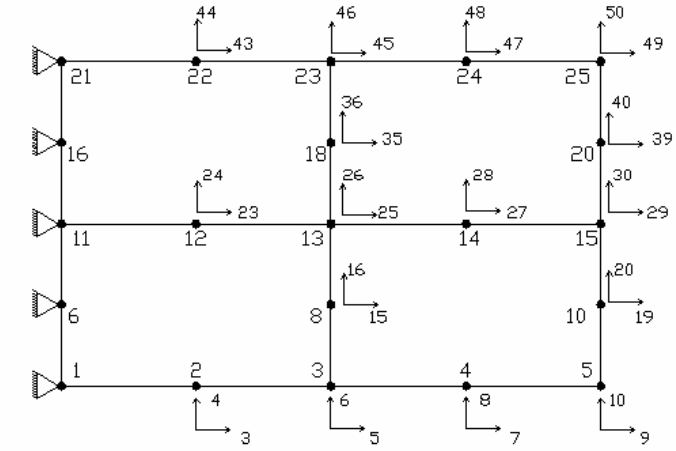


Fig 5.9: Substructures combined after static condensation

STEP: 4

After calculating DOF corresponding to boundary nodes, these DOF are again distributed to corresponding substructure to calculate internal DOF. Fig 5.10 shows all substructures with internal and boundary DOF. Nodes shown by red colors are the internal nodes. In this stage of analysis, internal DOFs are calculated using equation 11, as discussed in previous section.

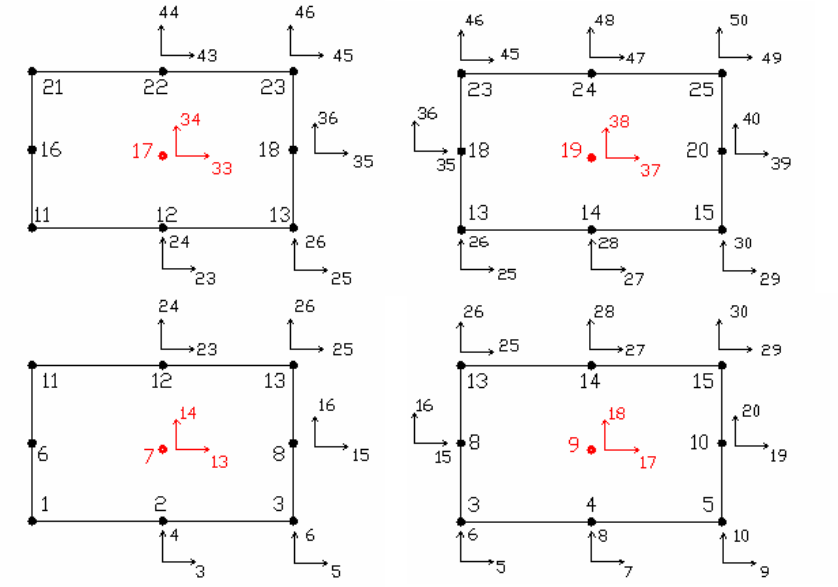


Fig 5.10: Substructures with known boundary DOF

STEP: 5

After calculating all DOFs corresponding to internal and boundary nodes, all results are assembled in order to have final results. After calculating primary unknowns, secondary unknowns such as stresses and strains are calculated. Fig 5.11 shows the whole structure with DOFs corresponding to all nodes.

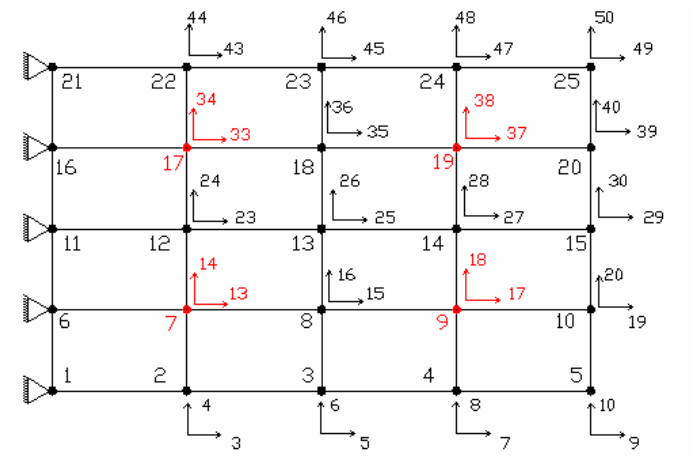


Fig 5.11: Complete structure with all DOF known

5.3.2 Advantages of substructure technique

Some of the advantages of the substructure technique can be summarized as follows:

1) In substructure technique, the stiffness matrix of each substructure is statically condensed out so that effective stiffness matrix corresponds to only the boundary degree of freedoms.

2) The number of nodes in the main structure is reduced because only the boundary nodes of the boundary nodes of substructure appear there.

3) The size of individual substructure is less than that of the main structure being analyzed. Thus at any given instant the main memory required to process the data corresponding to any one substructure is reduced.

4) Substructuring technique can be advantageous incase the structure is descritized into identical parts. In such cases, the stiffness matrix of a typical substructure can be formed and condensed only once and can be used as many times as the substructure appears in the main structure.

5) Another advantage of substructuring is the reduction in data. But this will be so, only if substructures are identical and repeatedly used.

6) The full advantage of substructure technique can be get in case of very large size problem for linear elastic analysis and in case of non linear analysis and structural optimization wherein a part of the structure only is modified before the subsequent analysis.

5.4 SOLUTION PROCEDURE AND COMPUTER PROGRAM

The finite element solution procedure is adopted for analysis. In this solution method, first of all the element stiffness matrix and consistent load vector is calculated and then theses stiffness matrices and load vectors are assembled as per the boundary conditions. From this assembled stiffness matrix and load vector, nodal displacements are calculated and finally from these nodal displacements, element stresses are calculated.

In present study, substructure analysis technique is implemented for the finite element analysis. This divides the whole analysis process into five different tasks. So for each process, different computer programs are prepared in "c" programming language. The first program "**plan1**" prepares separate data file for individual substructure. This data file includes the details such as number of elements in a particular substructure, number of nodes for that substructure, material property, loading data, number of restrained nodes, joint restraints and also the boundary nodes (needed for static condensation) for that particular substructure. After execution of the first program, in second program "**plan2**", element stiffness matrix and load vector is calculated and assembled as per corresponding degree of freedoms. Static condensation of stiffness matrix and load vector is also carried out in the same program. This condensed stiffness matrix and load vector is the input data for third program "**plan3**", in which degree of freedoms corresponding to boundary nodes are calculated. In forth program "**plan4**", internal degree of freedoms are calculated using boundary degree of freedoms. After calculating all degree of freedoms, element stresses are calculated in forth program. And in the fifth program "**plan5**", all the results of individual substructures are collected to print in final result file.

5.5 ANALYSIS PROBLEM AND RESULTS

As a first application, Plane Stress problem is selected for distributed analysis. The geometry of the plate is shown in Fig 5.12. The plate is a square plate and having a hole in the center. This plate is subjected to uniform tension on both edges. Young's modulus of elasticity of the

plate material is E = 20000 kN/cm^2 and Poisson's ratio is 0.3. Plate is having thickness of 2.5cm. As the plate is symmetrical about both x and y-axis, only the quarter plate as shown in Fig 5.12 by hatched area, is descritized into number of elements. For descritization, CST element with linear displacement models is used. Descritization of plate is shown in Fig 5.13

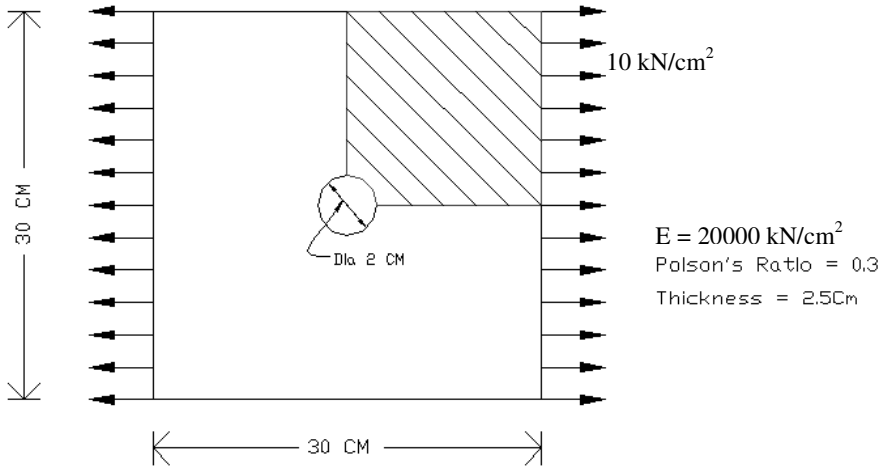


Fig 5.12: Rectangular plate with circular hole

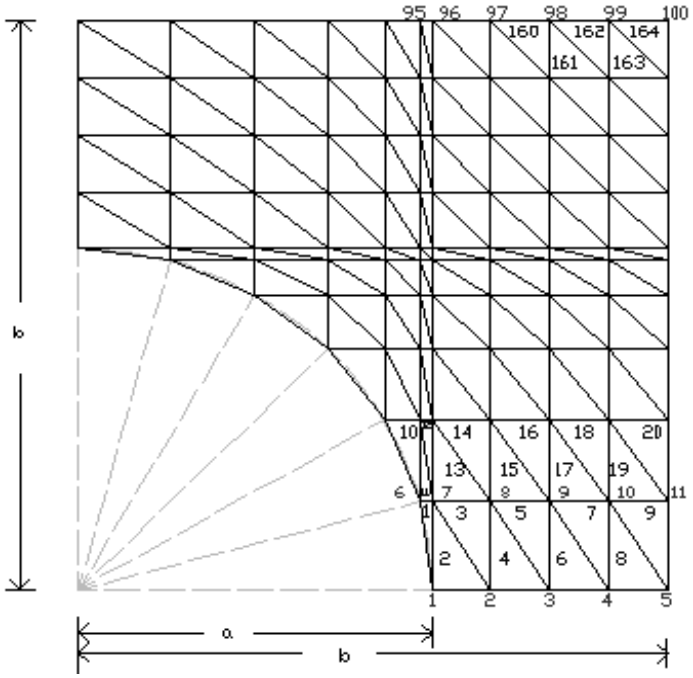


Fig 5.13: Descritization of quarter plate using CST elements

Further, this quarter substructure is divided into number of substructures, having equal number of elements. Fig 5.14 shows division of plate into five substructures.

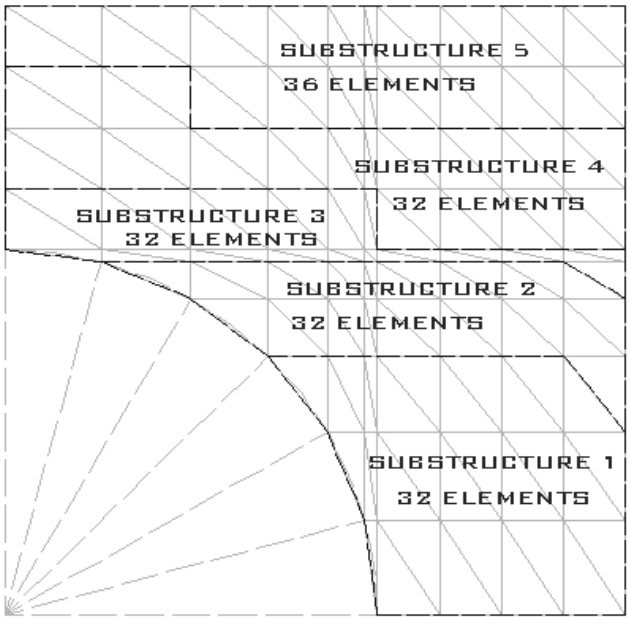


Fig 5.14: Division of quarter plate into five substructures

In present work, the quarter plate is descritized into 27504 elements. So the strucuture is having 13990 nodes. Each nodes is having 2 DOF so the problem is having total 27980 DOFs. The configuration of the application, when divided into three substructures, and the time details for each processes is shown in Fig 5.15 and Fig 5.16 respectively.

For the problem under consideration good agreement has been observed with the analytical solution for σ_θ given in reference [17]. Comparison of results of analysis with the reference results is tabulated in Table 5.1

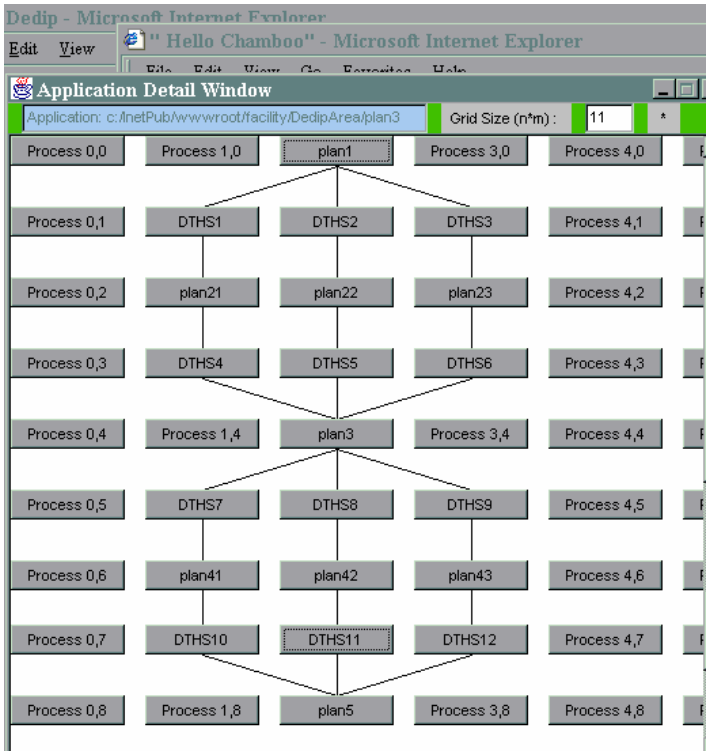
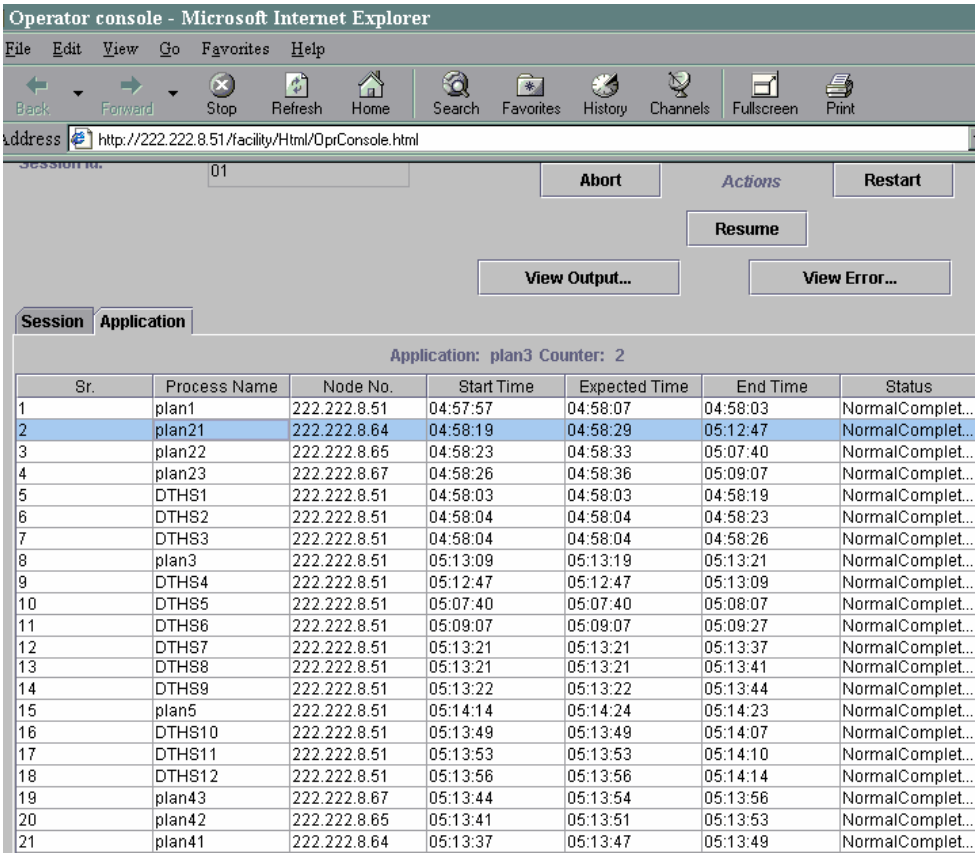


Fig 5.15: Configuration for plane stress problem for three substructures

Table5.1: Comparison of analysis and reference results

Sr. No	a (mm)	b (mm)	R (mm)	S (N/mm^2)	σ_x (N/mm^2)	σ_y (N/mm^2)	τ_{xy} (N/mm^2)	σ_θ (N/mm^2)	
								Reference	Analysis
1	100	500	110	100	221.8	23.44	-40.25	234.58	229.58
2			150		152.9	32.32	4.34	147.05	147.78
3			200		121.9	20.42	5.28	118.29	117.00
4			250		111.7	14.49	4.78	108.59	107.13
5			300		107.7	10.88	3.94	104.28	103.39
6			375		104.3	7.07	2.88	101.25	100.33
7			450		102.4	4.94	2.19	99.80	98.69

Where, R is the radial distance of point of interest and S is the intensity of tensile force.



Sr.	Process Name	Node No.	Start Time	Expected Time	End Time	Status
1	plan1	222.222.8.51	04:57:57	04:58:07	04:58:03	NormalComplet...
2	plan21	222.222.8.64	04:58:19	04:58:29	05:12:47	NormalComplet...
3	plan22	222.222.8.65	04:58:23	04:58:33	05:07:40	NormalComplet...
4	plan23	222.222.8.67	04:58:26	04:58:36	05:09:07	NormalComplet...
5	DTHS1	222.222.8.51	04:58:03	04:58:03	04:58:19	NormalComplet...
6	DTHS2	222.222.8.51	04:58:04	04:58:04	04:58:23	NormalComplet...
7	DTHS3	222.222.8.51	04:58:04	04:58:04	04:58:26	NormalComplet...
8	plan3	222.222.8.51	05:13:09	05:13:19	05:13:21	NormalComplet...
9	DTHS4	222.222.8.51	05:12:47	05:12:47	05:13:09	NormalComplet...
10	DTHS5	222.222.8.51	05:07:40	05:07:40	05:08:07	NormalComplet...
11	DTHS6	222.222.8.51	05:09:07	05:09:07	05:09:27	NormalComplet...
12	DTHS7	222.222.8.51	05:13:21	05:13:21	05:13:37	NormalComplet...
13	DTHS8	222.222.8.51	05:13:21	05:13:21	05:13:41	NormalComplet...
14	DTHS9	222.222.8.51	05:13:22	05:13:22	05:13:44	NormalComplet...
15	plan5	222.222.8.51	05:14:14	05:14:24	05:14:23	NormalComplet...
16	DTHS10	222.222.8.51	05:13:49	05:13:49	05:14:07	NormalComplet...
17	DTHS11	222.222.8.51	05:13:53	05:13:53	05:14:10	NormalComplet...
18	DTHS12	222.222.8.51	05:13:56	05:13:56	05:14:14	NormalComplet...
19	plan43	222.222.8.67	05:13:44	05:13:54	05:13:56	NormalComplet...
20	plan42	222.222.8.65	05:13:41	05:13:51	05:13:53	NormalComplet...
21	plan41	222.222.8.64	05:13:37	05:13:47	05:13:49	NormalComplet...

Fig 5.16: Time requirement for individual process for three substructures

Average time required for various processes, when entire structure divided into different number of substructure, is tabulated in Table 5.2 Also time required in parallel implementation i.e. computation time and communication time is shown. Based on the time required for various processes in sequential and parallel implementation, speedup is calculated. The comparison of ideal speedup and observed speedup is shown in Fig 5.17. Comparison of communication and computation time for various substructures is shown in Fig.5.18. From calculated speedup and ideal speedup, efficiency is also calculated as a measure of performance.

Table 5.2: Time required for sequential and parallel processing

Process	NB	NEQ	file size	time	Sequential time	Parallel time		speed up		Efficiency
						Comp	comm	Ideal	Calculated	
			(kb)	(sec)	(sec)	(sec)	(sec)			(%)
3 substructures										
plan1				6						
DTHS			377	16						
Plan2	9080	9476		689						
DTHS			2874	35						
Plan3	632	1380		12						
DTHS			11	15						
Plan4	240	9476		12						
DTHS			1035	14						
Plan5				6	2127	725	80	3	2.64	88.07
4 substructures										
Plan1				6						
DTHS			288	20						
Plan2	6808	7162		400						
DTHS			2507	38						
Plan3	590	1612		15						
DTHS			10	19						
Plan4	240	7162		10						
DTHS			778	17						
Plan5				7	1668	438	94	4	3.14	78.38
5 substructures										
Plan1				6						
DTHS			232	23						
Plan2	5442	5774		215						
DTHS			2354	48						
Plan3	572	1852		16						
DTHS			10	23						
Plan4	240	5774		9						
DTHS			624	17						
Plan5				6	1327	252	111	5	3.78	75.59
6 substructures										
Plan1				7						
DTHS			194	25						
Plan2	4534	4848		185						
DTHS			2209	56						
Plan3	554	2082		20						
DTHS			9	25						
Plan4	240	4848		7						
DTHS			520	18						
Plan5				8	1187	227	124	6	3.38	56.36

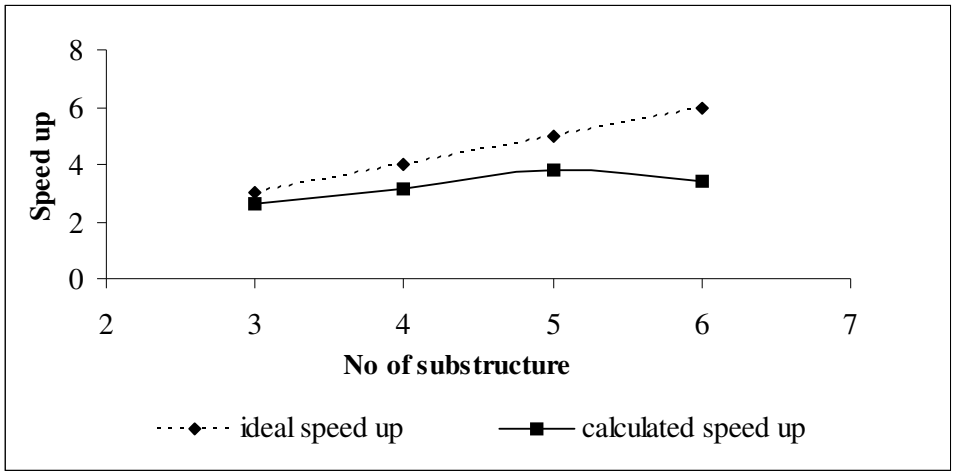


Fig 5.17: Comparison of ideal and observed Speedup

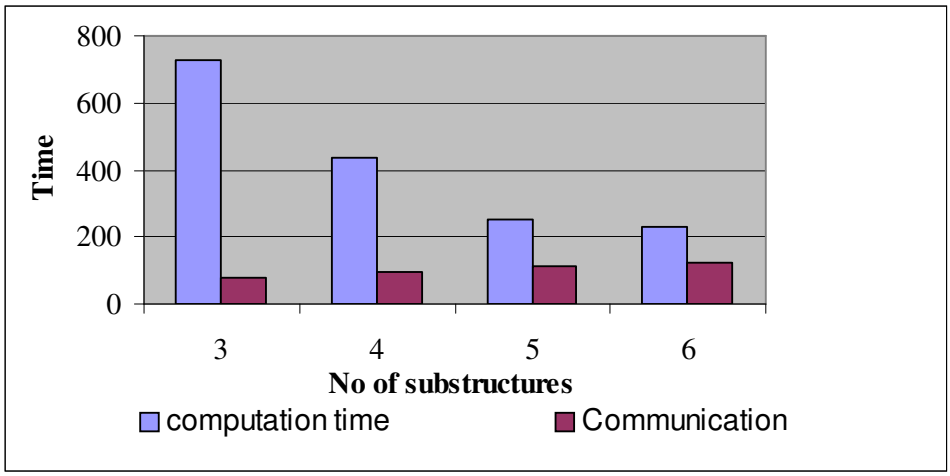


Fig 5.18: Computation and communication time

5.6 SUMMARY

In present chapter, plane stress problem has implemented on parallel computers. So the chapter includes the finite element formulation of plane stress element. Substructure technique for analysis has also been discussed. In the later part of the chapter one plane stress problem has been analyzed, which is having total 27504 elements, and 13990 nodes (i.e. 27980 DOFs). The continuum has been divided into 3, 4, 5 and 6 numbers of substructures and implemented on parallel computers. From the work carried out and the results for sequential as well as parallel processing, following points can be observed:

- For the selected problem for three substructures, ideal and calculated speed up is nearly equal and so giving around 88% efficiency.

- As the number of computers goes on increasing the difference between ideal and calculated efficiency goes on increasing and so efficiency goes on reducing.

- Figure 5.18 (computation and communication time) also shows the linear variation of computation and communication time. With the increasing in the number of computers, computation time goes on reducing and communication time goes on increasing.

CHAPTER 6 PLATE BENDING PROBLEM

6.1 INTRODUCTION

In many areas of structural design it is required to analyze plates subjected to lateral loads. As the classical solution of plate involves tedious calculations, especially when the plates are arbitrarily shaped and are anisotropic, as a numerical solution to the problem a number of finite element models have been developed.

According to the nature of stress states, the plates are classified as follows [25]:

1.) Thick plates: In thick plate tri-axial state of stress is developed. Plates for which the ratio of thickness to least dimension on plan exceeds 1/10 may be taken as belonging to this class.
2.) Thin plates with small deflection: In thin plates with small deflection the membrane stresses are very small compared to flexural stresses under deformation due to transverse loading. This class may be takes to comprise plates for which the ratio of thickness to span does not exceed 1/10 and the maximum deflection w is less than h/10 to h/5.
3.) Thin plates with large deflection: In thin plates with large deflections the flexural stresses are accompanied by relatively large tensile or compressive stresses in the middle plane. These membrane stresses significantly affect the bending moment.

6.2 MINDLIN'S THEORY

Mindlin's approximation is that straight lines originally normal to the mid surface, before deformation, remain straight but not normal to the deformed mid surface, i.e the average rotation of the section may be takes as the rotation in which normal remain perpendicular to the mid surface plus an additional rotation due to transverse shear. The three assumptions made in Mindlin's theory of plates are as follows:

A.) The deflections of the plate w are small.

B.) Normals to the plate mid surface before deformation remain straight but are not necessarily normal to it after deformation.

C.) Stresses normal to the mid surface are negligible.

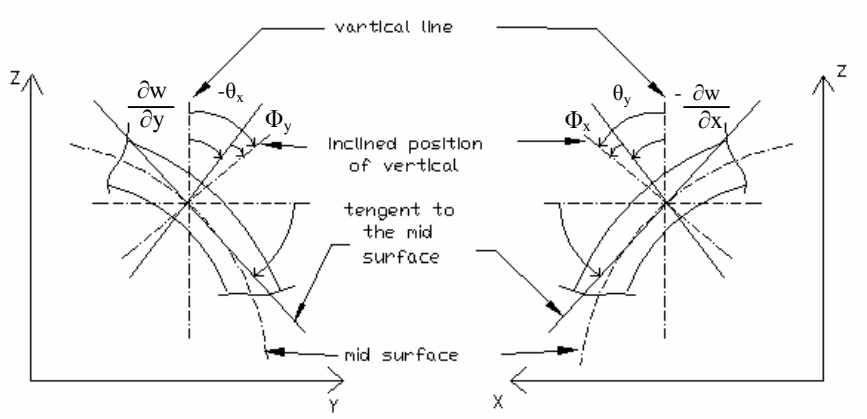


Fig 6.1: Rotation of the normal about x and y axes considering average shear deformation

Referring to Fig 6.1 in which Φ_x denoted an average transverse shear strain for a section x = constant, the total rotation θ_y can be expressed as,

$$\theta_y = -\frac{\partial w}{\partial x} + \Phi_x \qquad (6.1)$$

And similarly for section y = constant

$$-\theta_x = -\frac{\partial w}{\partial y} + \Phi_y \qquad (6.2)$$

Hence, the average shear deformations, Φ_x and Φ_y are given by

$$\Phi_x = \theta_y + \frac{\partial w}{\partial y} \qquad \Phi_x = \theta_y + \frac{\partial w}{\partial y} \qquad (6.3)$$

The basic relationships for the curvatures and stress resultants can be given as follows.

$$\{M\} = [C_f]\{k_c\} \qquad (6.4)$$

Where, $\{k_c\}^T = [\,k_x \quad k_y \quad k_{xy}]$

And,

$$[C_f] = \frac{E}{12(1-\mu^2)} \begin{pmatrix} 1 & \mu & 0 \\ \mu & 1 & 0 \\ 0 & 0 & (1-\mu)/2 \end{pmatrix}$$

The average shear deformation $\Phi_{x \text{ and}}\ \Phi_y$ are expressed in equation 6.3. The shear stresses τ_{xz} and τ_{xz} and the shear deformation are related as follows:

$$\begin{Bmatrix} \tau_{xz} \\ \tau_{xz} \end{Bmatrix} = \begin{bmatrix} C_{44} & C_{45} \\ C_{54} & C_{55} \end{bmatrix} \begin{Bmatrix} \Phi_x \\ \Phi_y \end{Bmatrix} \tag{6.5}$$

For an isotropic material the above relation can be written as

$$\begin{Bmatrix} \tau_{xz} \\ \tau_{xz} \end{Bmatrix} = \frac{E}{2(1+\upsilon)} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{Bmatrix} \Phi_x \\ \Phi_y \end{Bmatrix} \tag{6.6}$$

The stress resultants Q_x and Q_y can be computed as

$$\begin{Bmatrix} Q_x \\ Q_y \end{Bmatrix} = \frac{E\, h \propto}{2(1+\upsilon)} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{Bmatrix} \Phi_x \\ \Phi_y \end{Bmatrix} \tag{6.7}$$

The curvature K_x, K_y and the twist K_{xy} can be expressed in terms of rotations θ_x and θ_y as

$$K_x = \frac{\partial \theta_y}{\partial x} \qquad K_y = -\frac{\partial \theta_x}{\partial y} \qquad K_{xy} = \left(\frac{\partial \theta_y}{\partial y} - \frac{\partial \theta_x}{\partial x} \right) \tag{6.8}$$

The stress resultant {M} and {Q} given by Eq. (6.4) and Eq. (6.7) can be combined and for isotropic plates, the constitutive matrix, can be given as,

$$\left[\begin{array}{c} M_x \\ M_y \\ M_{xy} \\ \hdashline Q_x \\ Q_y \end{array}\right] = \left[\begin{array}{c:c} \dfrac{Eh^3}{12(1-\mu^2)} \begin{bmatrix} 1 & \mu & 0 \\ \mu & 1 & 0 \\ 0 & 0 & (1-\mu)/2 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \\ \hdashline \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \dfrac{Eh}{2(1+\mu)} \begin{bmatrix} \alpha & 0 \\ 0 & \alpha \end{bmatrix} \end{array}\right] \left[\begin{array}{c} K_x \\ K_y \\ K_{xy} \\ \hdashline \phi_x \\ \phi_y \end{array}\right] \tag{6.9}$$

The above relation can be compared with $\{\sigma\} = [C]\ \{\varepsilon\}$ for the usual stress-strain relation. Thus in case of plate bending, the stress resultants and the corresponding curvatures and shear deformations can be considered analogous to stresses and strains. Hence, for uniformity and convenience, equation 6.9 can be expressed as

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

Where, $\{\sigma\} = \{ M_x \quad M_y \quad M_{xy} \quad Q_x \quad Q_y \}$

And $\{\varepsilon\} = \{ K_x \quad K_y \quad K_{xy} \quad \Phi_x \quad \Phi_y \}$

Moments and shear in the plate, with their positive directions are shown in Fig 6.2

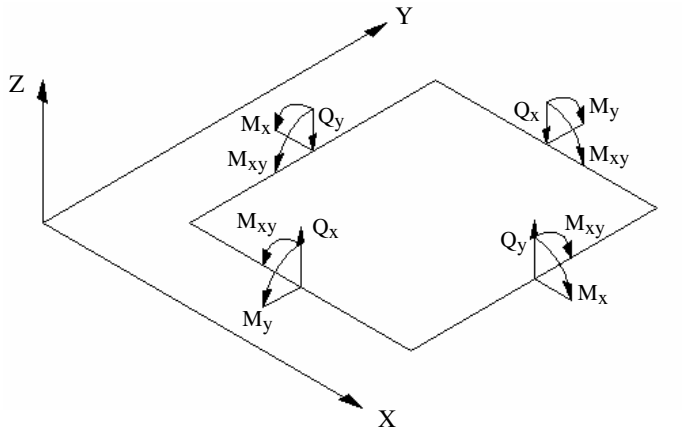


Fig 6.2 Positive moments and shear forces in plate

The curvature and shear deformation vector $\{\varepsilon\}$ can be expressed in terms of the three displacements w, θ_x and θ_y as,

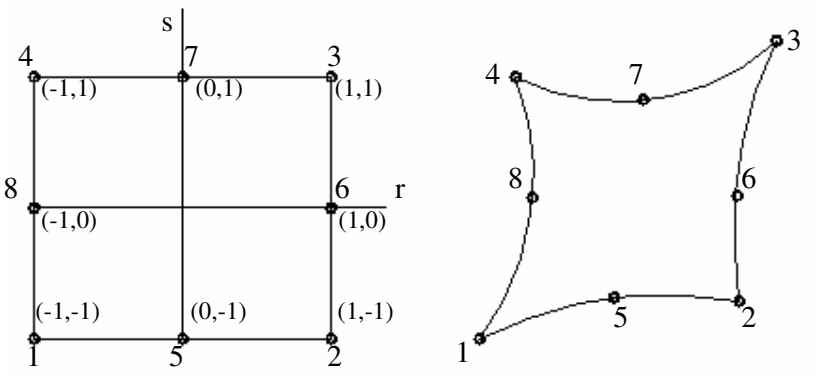
$$\{\varepsilon\} = \begin{Bmatrix} \partial\theta_y / \partial x \\ -\partial\theta_x / \partial y \\ \partial\theta_y / \partial y - \partial\theta_x / \partial x \\ \theta_y + \partial w / \partial x \\ -\theta_x + \partial w / \partial y \end{Bmatrix} \qquad (6.10)$$

6.3 FINITE ELEMENT FORMULATION

Finite element formulation for eight nodded isoparametric element is given as follows.

6.3.1 Shape functions

Consider a rectangular element as shown in Fig 6.3. The rectangle in natural coordinate is transformed into an arbitrary element with straight boundary. The shape function used for representing the variation of displacement for eight nodded isoparametric element can be given as follows [31]:



Rectangle in natural coordinate Isoparametric element

Fig 6.3: Eight nodded Isoparametric element

$N_1 = (1-r)(1-s)(-r-s-1)/4$ $N_2 = (1+r)(1-s)(r-s-1)/4$

$N_3 = (1+r)(1+s)(r+s-1)/4$ $N_4 = (1-r)(1+s)(-r+s-1)/4$

$N_5 = (1+r)(1-s)(1-r)/2$ $N_6 = (1+r)(1-s)(1+s)/2$

$N_7 = (1+r)(1+s)(1-r)/2$ $N_8 = (1-r)(1-s)(1+s)/2$

These shape functions can now be used to describe the geometry of the arbitrary rectangle in the cartesian system as follows:

$$x = \sum_{i=1}^{8} N_i x_i \qquad y = \sum_{i=1}^{8} N_i y_i \qquad (6.11)$$

The variation of displacement w, θx and θy within the element in terms of nodal values can be expressed as

$$w = \sum_{i=1}^{8} N_i w_i \qquad \theta x = \sum_{i=1}^{8} N_i \theta_{xi} \qquad \theta y = \sum_{i=1}^{8} N_i \theta_{yi} \qquad (6.12)$$

Here the shape functions are expressed in terms of natural coordinates r and s, while to calculate the displacement calculation of derivatives of shape functions w.r.t natural coordinates is necessary. So the relationship between the two coordinate systems can be computed by using the chain rule of partial differentiation and is given as

$$\begin{Bmatrix} \partial/\partial r \\ \partial/\partial s \end{Bmatrix} = \begin{bmatrix} \partial x/\partial r & \partial y/\partial r \\ \partial x/\partial s & \partial y/\partial s \end{bmatrix} \begin{Bmatrix} \partial/\partial x \\ \partial/\partial y \end{Bmatrix} = [J] \begin{Bmatrix} \partial/\partial x \\ \partial/\partial y \end{Bmatrix}$$

Jacobian matrix [J] for eight noded element can be given as

$$[J] = \begin{pmatrix} \partial N_1/\partial r & \partial N_2/\partial r & \partial N_3/\partial r \ldots\ldots \partial N_8/\partial r \\ \partial N_1/\partial s & \partial N_2/\partial s & \partial N_3/\partial s \ldots\ldots \partial N_8/\partial s \end{pmatrix} \begin{pmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ . & . \\ . & . \\ x_8 & y_8 \end{pmatrix} \quad (6.13)$$

Now the derivatives w.r.t global axis are found by inversing the Jacobian [J] and is given by

$$\begin{pmatrix} \partial N_1/\partial x & \partial N_2/\partial x \ldots\ldots \partial N_8/\partial x \\ \partial N_1/\partial y & \partial N_2/\partial y \ldots\ldots \partial N_8/\partial y \end{pmatrix} = [J]^{-1} \begin{pmatrix} \partial N_1/\partial r & \partial N_2/\partial r \ldots\ldots \partial N_8/\partial r \\ \partial N_1/\partial s & \partial N_2/\partial s \ldots\ldots \partial N_8/\partial s \end{pmatrix} \quad (6.14)$$

6.3.2 Strain Displacement matrix [B]

The element curvatures and shear deformation ε given in equation 6.10 and the nodal displacements {d} are related as

$$\varepsilon = [B] \{d\}$$

Where,

$$\{d\} = [w_1, \theta x_1, \theta y_1, w_2, \theta x_2, \theta y_2, w_3, \theta x_3, \theta y_3 \ldots\ldots\ldots\ldots w_8, \theta x_8, \theta y_8] \quad (6.15)$$

In order to compute the matrix [B], first the equation 6.12 is to be differentiated w.r.t x and y to get each of the terms of ε. Thus,

$$K_x = \sum_{i=1}^{8} \theta_{yi}\, \partial N_i/\partial x \qquad K_y = \sum_{i=1}^{8} -\theta_{xi}\, \partial N_i/\partial y$$

$$K_{xy} = \sum_{i=1}^{8} \theta_{yi}\, \partial N_i/\partial y - \sum_{i=1}^{8} \theta_{xi}\, \partial N_i/\partial x$$

$$\Phi_x = \sum_{i=1}^{8} w_i\, \partial N_i/\partial x + \sum_{i=1}^{8} \theta_{yi}\, N_i$$

$$\Phi_y = \sum_{i=1}^{8} w_i\, \partial N_i/\partial y - \sum_{i=1}^{8} \theta_{xi}\, Ni \quad (6.16)$$

Where, $\partial N_i/\partial x$ and $\partial N_i/\partial y$ are to be calculated using equation 6.14. In equation 6.16, elements of vector {ε} for curvatures and shear deformations are expressed in terms of the nodal displacements w, θ_x and θ_y. Thus

$$\{\varepsilon\} = \begin{Bmatrix} K_x \\ K_y \\ K_{xy} \\ \Phi_x \\ \Phi_y \end{Bmatrix} = [B] \begin{Bmatrix} w_1 \\ \theta_{x1} \\ \theta_{y1} \\ . \\ . \\ . \\ w_8 \\ \theta_{x8} \\ \theta_{y8} \end{Bmatrix} \quad (6.17)$$

Equation 6.17 cam be expressed as

$$\varepsilon = \sum_{i=1}^{8} [B_i] \{d_i\} \quad (6.18)$$

And,

$$[B_i] = \begin{bmatrix} 0 & 0 & \partial N_i / \partial x \\ 0 & -\partial N_i / \partial y & 0 \\ 0 & -\partial N_i / \partial x & \partial N_i / \partial y \\ \partial N_i / \partial x & 0 & N_i \\ \partial N_i / \partial y & -N_i & 0 \end{bmatrix}_{i=1,2,3.....8} \quad (6.19)$$

And,

$$\{d_i\} = \begin{Bmatrix} w_i \\ \theta_{xi} \\ \theta_{yi} \end{Bmatrix} \quad (6.20)$$

Therefore [B] matrix can be given as,

$$[B] = \begin{bmatrix} 0 & 0 & \partial N_1/\partial x & 0 & 0 & \partial N_8/\partial x \\ 0 & -\partial N_1/\partial y & 0 & 0 & -\partial N_8/\partial y & 0 \\ 0 & -\partial N_1/\partial x & \partial N_1/\partial y & 0 & -\partial N_8/\partial x & \partial N_8/\partial y \\ \partial N_1/\partial x & 0 & N_1 & \partial N_8/\partial x & 0 & N_8 \\ \partial N_1/\partial y & -N_1 & 0 \ldots\ldots\ldots\ldots & \partial N_8/\partial y & -N_8 & 0 \end{bmatrix} \quad (6.21)$$

6.3.3 Element Stiffness Matrix

As it is know that,

$$[K] = \int [B]^T [D] [B] \, dv$$

$$[K] = \int_{-1}^{1} \int_{-1}^{1} [B]^T [D] [B] \, t \det[J] \, dr \, ds$$

To evaluate the integral, two point Gauss Quadrature method is used. As per Gauss method,

$$\int_{-1}^{1}\int_{-1}^{1} f(r,s)\,dr\,ds = \sum_i \sum_j W_i W_j f(r_i,s_j)$$

Where, r_i and s_i are known as the sampling points and W_i and W_j are called the weights.

Here, for two point Gauss method $W_1 = W_2 = 1$ and natural coordinates of the sampling points are as follows:

1	+0.57735	+0.57735
2	-0.57735	+0.57735
3	-0.57735	-0.57735
4	+0.57735	-0.57735

So,

$$[K] = \sum_i^n \sum_j^n W_i W_j [B]_i^T [D] [B]_j t \det[J] \text{ ,} \quad \text{n is the number of sampling points}$$

So, to calculate the stiffness matrix first of all Jacobian matrix [J] is calculated at sampling point. Using [J] matrix strain displacement matrix [B] is calculated and using matrix [J] and [B], stiffness matrix is calculated at all sampling points and finally all stiffness matrixes are added to get the final stiffness matrix.

6.3.4 Load vector

The nodal load $\{Q_i\}$ at the node i for a uniformly distributed load q is given by

$$\{Q_i\} = \begin{Bmatrix} F_z \\ M_x \\ M_y \end{Bmatrix} = \int\int_A N_i \begin{Bmatrix} q \\ 0 \\ 0 \end{Bmatrix} \det[J]\,dr\,ds$$

Combining the nodal load vectors $\{Q_i\}$, the element load vector $\{Q\}$ can be numerically evaluated as,

$$\{Q\} = \begin{Bmatrix} Q_1 \\ Q_2 \\ Q_3 \\ . \\ . \\ . \\ Q_{22} \\ Q_{23} \\ Q_{24} \end{Bmatrix} = q \sum_{i=1}^{n} \sum_{j=1}^{n} W_i W_j \det[J] \begin{Bmatrix} N_1 \\ N_2 \\ N_3 \\ . \\ . \\ . \\ N_{22} \\ N_{23} \\ N_{24} \end{Bmatrix}$$

Gauss Quadrature with 2 x 2 sampling points is used to evaluate above equation.

6.4 SOLUTION PROCEDURE AND COMPUTER PROGRAM

The finite element method is adopted for analysis. In this solution method, first of all the element stiffness matrix and consistent load vector is calculated and then theses stiffness matrices and load vectors are assembled as per the boundary conditions. From this assembled stiffness matrix and load vector, nodal displacements are calculated and finally from these nodal displacements, element stresses are calculated.

In present study, substructure analysis technique is implemented for the finite element analysis. This divides the whole analysis process into five different tasks. So for each process, different computer programs are prepared in "c" programming language. The first program "**plana1**" prepares separate data file for individual substructure. This data file includes the details such as number of elements in a particular substructure, number of nodes for that substructure, material property, loading data, number of restrained nodes, joint restraints and also the boundary nodes (needed for static condensation) for that particular substructure. After execution of the first program, in second program "**plana2**", element stiffness matrix and load vector for is calculated and assembled as per corresponding degree of freedoms. Static condensation of stiffness matrix and load vector is also carried out in the same program. This condensed stiffness matrix and load vector is the input data for third program "**plana3**", in which degree of freedoms corresponding to boundary nodes are calculated. In forth program "**plana4**", internal degree of freedoms are calculated using boundary degree of freedoms. After calculating all degree of freedoms, element stresses are calculated in forth program. And in the fifth program "**plana5**" all the results of individual substructures are collected to print in final result file.

6.5 ANALYSIS PROBLEM AND RESULTS

6.5.1 Problem 1 (Annular plate problem)

As a first problem for plate bending analysis, a circular plate having hole in the center, as shown in Fig 6.4, is selected. Such analysis comes into picture in case of annular raft or when any circular slab is having hole in the center and subjected to uniformly distributed load. Plate is subjected to uniformly distributed pressure. Plate is fixed at the outer edge and free at the inner edge. Modulus of elasticity of plate is $2x10^8$ kN/m^2, poisson's ratio is 0.3, thickness of the plate is 0.1m and the intensity of pressure is 20kN/m^2.

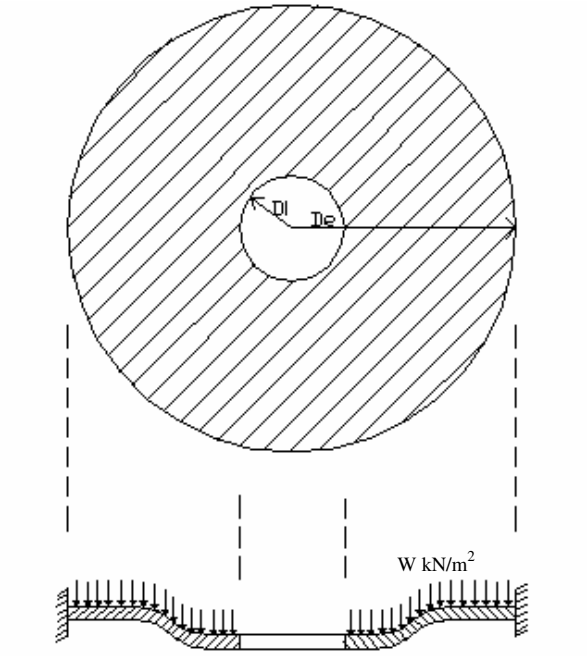


Fig 6.4: Circular Plate having whole in the center and subjected to uniformly distributed pressure

For the analysis 8 noded isoparametric plate element is selected. So the plate has been descritized into isoparametric elements. Two types of meshing is carried out. One is the uniform, in which length of each element is kept uniform and other is graded mesh in which area of each element is kept uniform. Both descritizations are shown in Fig 6.5 and Fig 6.6 Further the plate is divided into number of substructures. Load on each computer is maintained by keeping number of element same in each substructure. Fig 6.7 shows the division of quarter plate into three substructures.

Results for the deflections are tabulated for uniform as well as graded mesh in Table 6.1 These results are compared with those obtained by analytical methods, given in reference [20].

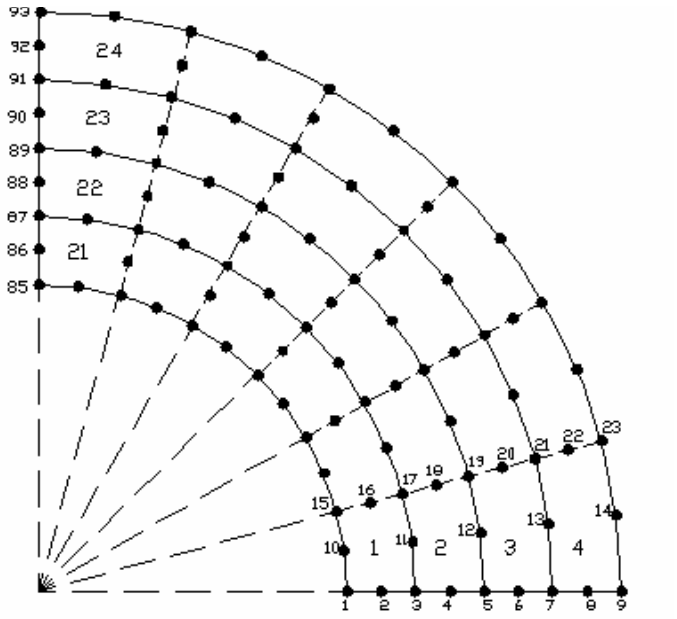


Fig 6.5: Descritization of plate in 8 noded isoparametric elements in uniform mesh

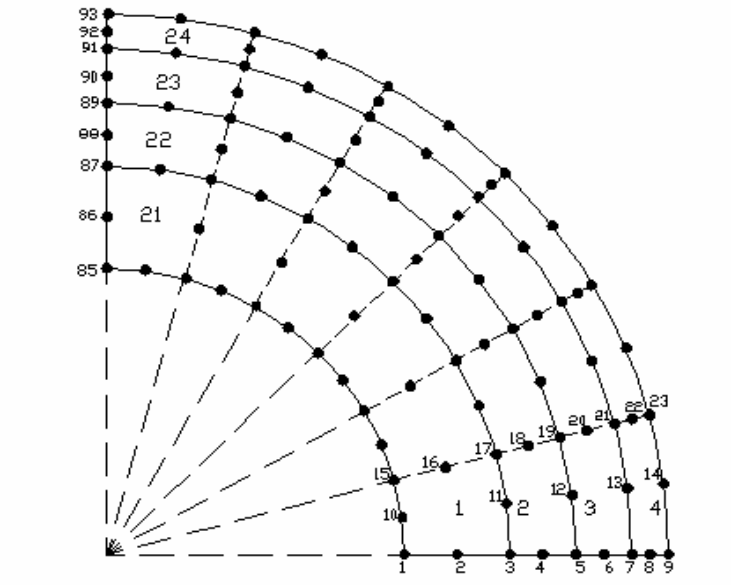


Fig 6.6: Descritization of plate in 8 noded isoparametric elements in graded mesh

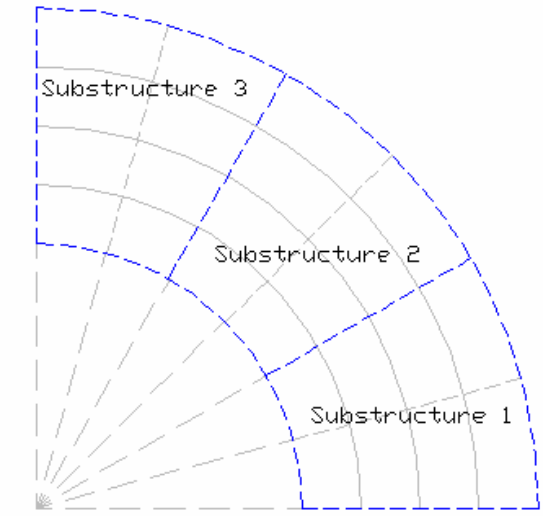


Fig 6.7: Division of quarter plate into three substructures

Table 6.1: Displacement coefficient for uniform and graded mesh

Sr No	ID =b	ED =a	TNE	TNN	k1		
					Analysis		Reference
					Uniform	Graded	
1	2	4	4	21	0.057	0.057	0.057
2	2	6	4	21	0.129	0.129	0.130
3	2	8	4	21	0.161	0.161	0.162
4	2	10	4	21	0.174	0.174	0.175
5	4	8	12	51	0.057	0.057	0.057
6	5	15	12	51	0.129	0.129	0.130
7	4	16	12	51	0.161	0.163	0.162
8	4	20	12	51	0.175	0.178	0.175
9	2	4	24	93	0.057	0.057	0.057
10	2	6	24	93	0.129	0.129	0.130
11	2	8	24	93	0.162	0.161	0.162
12	2	10	24	93	0.174	0.177	0.175
13	2	4	48	173	0.057	0.057	0.057
14	2	6	48	173	0.129	0.129	0.130
15	2	8	48	173	0.162	0.162	0.162
16	2	10	48	173	0.175	0.174	0.175
17	2	4	72	253	0.057	0.057	0.057
18	2	6	72	253	0.129	0.129	0.130
19	2	8	72	253	0.161	0.162	0.162
20	2	10	72	253	0.173	0.175	0.175
21	2	4	96	333	0.057	0.057	0.057
22	2	6	96	333	0.129	0.130	0.130
23	2	8	96	333	0.162	0.162	0.162
24	2	10	96	333	0.174	0.173	0.175

Where,

TNE = Total number of elements

TNN = Total number of nodes

k1 = $W_{max}\ E\ h^3 / q\ a^4$

W_{max} = maximum deflection of plate

q = Intensity of uniformly distributed load

For time calculation, in present work the plate is descritized into 1800 elements. So plate is having 5725 nodes. Each node is having three DOFs. So the problem has total 17175 DOFs. The configuration of the problem on WebDedip, when divided into four substructures and the time requirement for each processes is shown in Fig 6.8 and Fig 6.9 respectively.

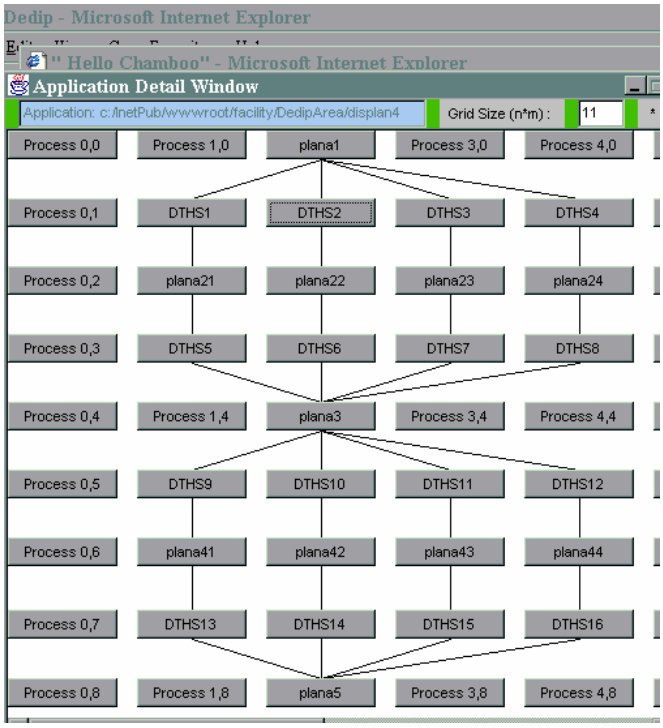
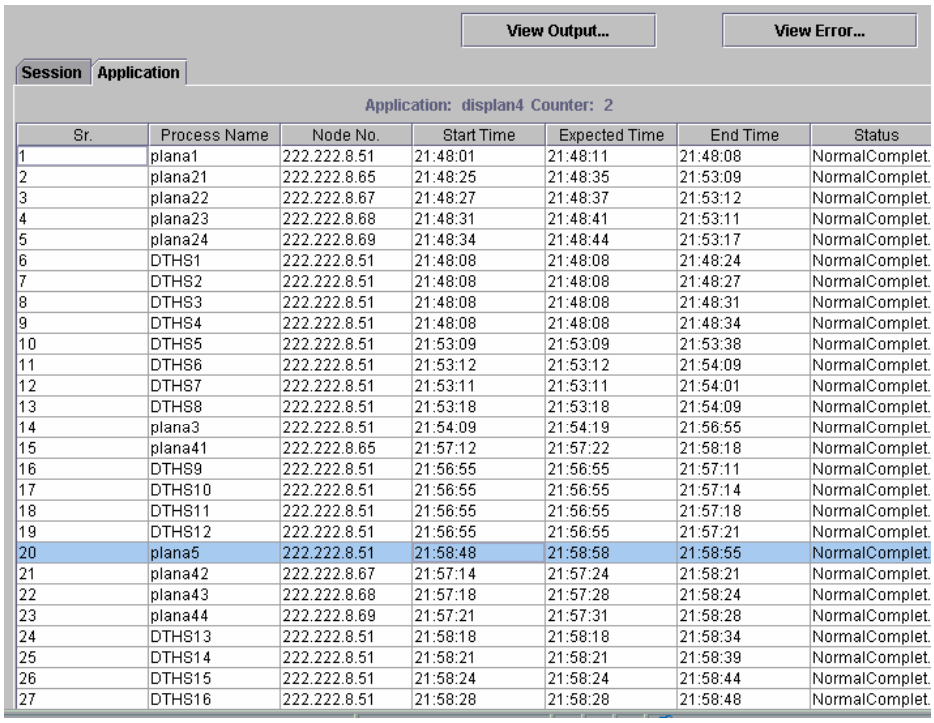


Fig 6.8: Configuration of plate bending problem, for four substructures



Sr.	Process Name	Node No.	Start Time	Expected Time	End Time	Status
1	plana1	222.222.8.51	21:48:01	21:48:11	21:48:08	NormalComplet.
2	plana21	222.222.8.65	21:48:25	21:48:35	21:53:09	NormalComplet.
3	plana22	222.222.8.67	21:48:27	21:48:37	21:53:12	NormalComplet.
4	plana23	222.222.8.68	21:48:31	21:48:41	21:53:11	NormalComplet.
5	plana24	222.222.8.69	21:48:34	21:48:44	21:53:17	NormalComplet.
6	DTHS1	222.222.8.51	21:48:08	21:48:08	21:48:24	NormalComplet.
7	DTHS2	222.222.8.51	21:48:08	21:48:08	21:48:27	NormalComplet.
8	DTHS3	222.222.8.51	21:48:08	21:48:08	21:48:31	NormalComplet.
9	DTHS4	222.222.8.51	21:48:08	21:48:08	21:48:34	NormalComplet.
10	DTHS5	222.222.8.51	21:53:09	21:53:09	21:53:38	NormalComplet.
11	DTHS6	222.222.8.51	21:53:12	21:53:12	21:54:09	NormalComplet.
12	DTHS7	222.222.8.51	21:53:11	21:53:11	21:54:01	NormalComplet.
13	DTHS8	222.222.8.51	21:53:18	21:53:18	21:54:09	NormalComplet.
14	plana3	222.222.8.51	21:54:09	21:54:19	21:56:55	NormalComplet.
15	plana41	222.222.8.65	21:57:12	21:57:22	21:58:18	NormalComplet.
16	DTHS9	222.222.8.51	21:56:55	21:56:55	21:57:11	NormalComplet.
17	DTHS10	222.222.8.51	21:56:55	21:56:55	21:57:14	NormalComplet.
18	DTHS11	222.222.8.51	21:56:55	21:56:55	21:57:18	NormalComplet.
19	DTHS12	222.222.8.51	21:56:55	21:56:55	21:57:21	NormalComplet.
20	plana5	222.222.8.51	21:58:48	21:58:58	21:58:55	NormalComplet.
21	plana42	222.222.8.67	21:57:14	21:57:24	21:58:21	NormalComplet.
22	plana43	222.222.8.68	21:57:18	21:57:28	21:58:24	NormalComplet.
23	plana44	222.222.8.69	21:57:21	21:57:31	21:58:28	NormalComplet.
24	DTHS13	222.222.8.51	21:58:18	21:58:18	21:58:34	NormalComplet.
25	DTHS14	222.222.8.51	21:58:21	21:58:21	21:58:39	NormalComplet.
26	DTHS15	222.222.8.51	21:58:24	21:58:24	21:58:44	NormalComplet.
27	DTHS16	222.222.8.51	21:58:28	21:58:28	21:58:48	NormalComplet.

Fig 6.9: Time requirement for individual process for four substructures

The average time required for various processes, when entire structure divided into different number of substructure, is tabulated in Table 6.2 Also time required in parallel implementation i.e. computation time and communication time is shown. Based on the time required for various processes in sequential and parallel implementation, speedup is calculated. The comparison of ideal speedup and observed speedup is shown in Fig 6.10. From calculated speedup and ideal speedup, efficiency is also calculated as a measure of performance. Fig 6.11 shows the comparison of computation and communication time.

Table 6.2: Time required for sequential and parallel processing

Processes	NB	NEQ	File Size	Time	Sequential Time	Parallel time		Speed up		Efficiency
						Comp	Comm.	Ideal	Calculated	
			(Kb)	(Sec)	(Sec)	(Sec)	(Sec)			(%)
3 substructures										
plana1				7						
DTHS			72	19						
plana2	5391	6327		556						
DTHS			11121	44						
plana3	1848	3738		122						
DTHS			21	19						
plana4	1365	6327		88						
DTHS			2822	18						
plana5				7	2068	780	100	3	2.35	78.33

4 substructures										
plana1				7						
DTHS			56	26						
plana2	4047	4971		358						
DTHS			10278	48						
plana3	1836	4635		165						
DTHS			20	26						
plana4	1365	4971		67						
DTHS			2123	19						
plana5				7	1879	604	119	4	2.60	64.97
6 substructures										
plana1				7						
DTHS			50	35						
plana2	2703	3615		115						
DTHS			23621	80						
plana3	1824	6429		248						
DTHS			30	35						
plana4	1365	3615		44						
DTHS			150	23						
plana5				7	1216	421	173	6	2.05	34.12

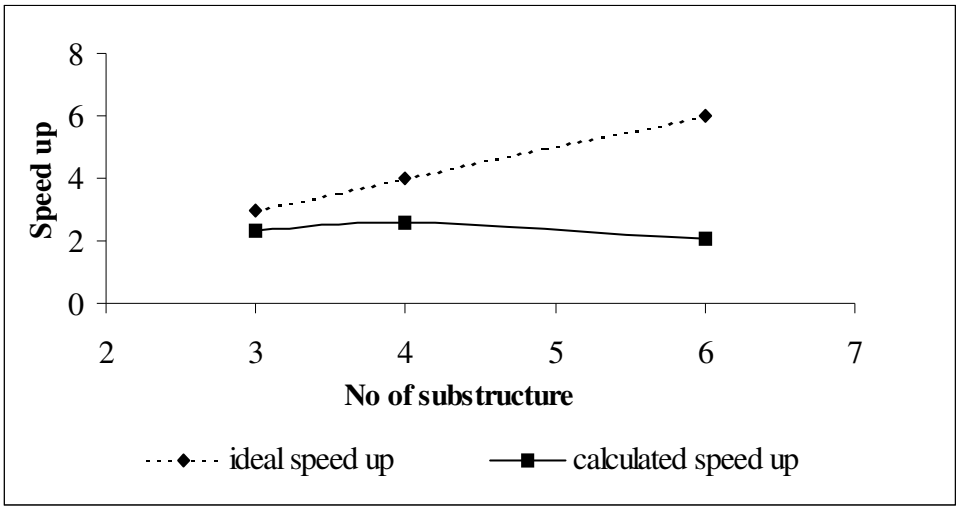


Fig 6.10: Comparison of ideal and observed Speedup

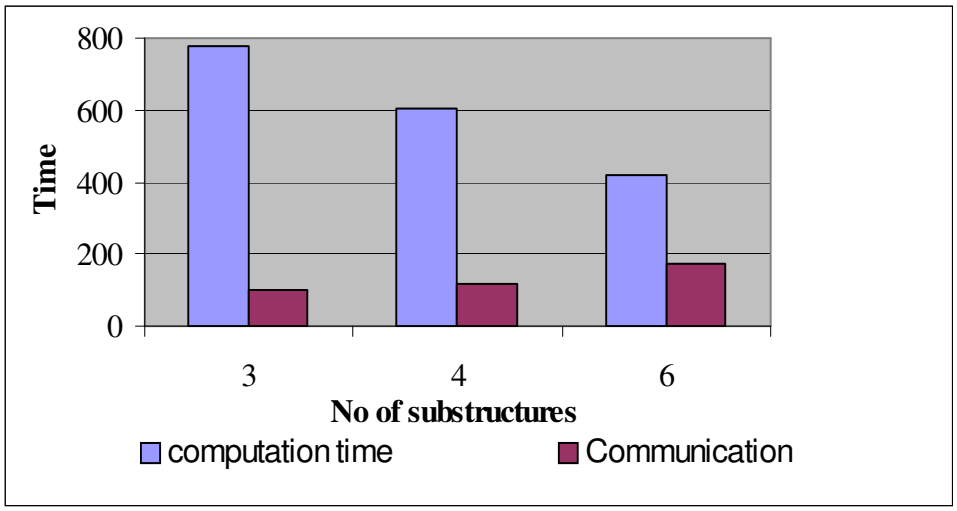


Fig 6.11: Computation and communication time

6.5.2 Problem 2 (Skew plate problem)

For second example of plate bending problem, a skew plate, as shown in Fig 6.12, subjected to uniformly distributed pressure is selected. Such analysis comes into picture when any skew bridge is to be analyzed for different skew angles. Plate is simply supported at two edges and free at the remaining two edges. Modulus of elasticity of plate is $2x10^8$ kN/m^2, Poisson's ratio is 0.2, thickness of the plate is 0.1m and the intensity of pressure is 20kN/m^2. For the analysis 8 noded isoparametric plate element is selected. So the plate has been descritized into number of isoparametric elements as shown in Fig 6.12. Division of plate into four substructures in shown in Fig 6.13

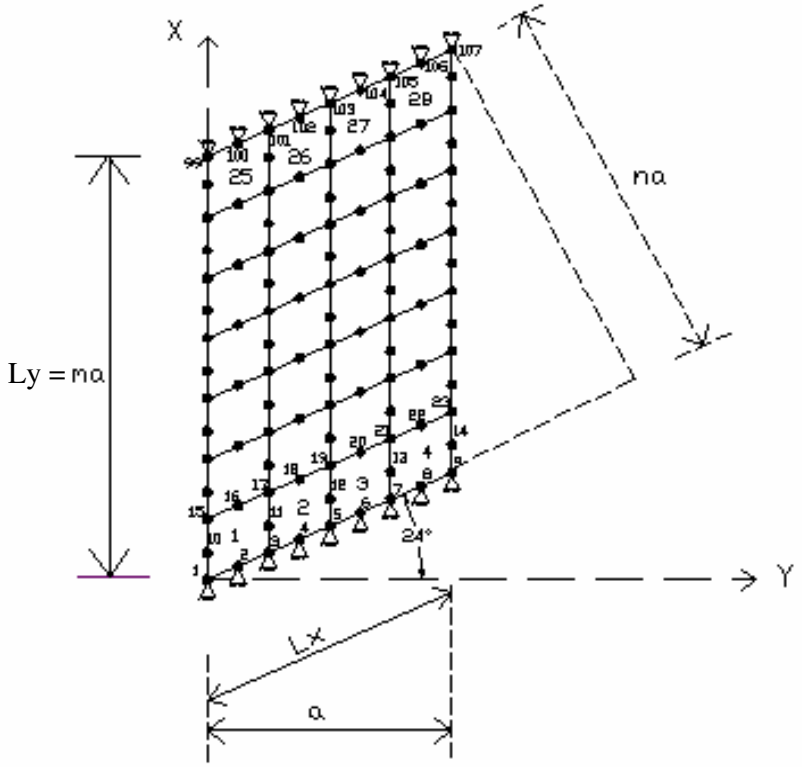


Fig 6.12: Skew plate descritization in 8noded isoparametric elements

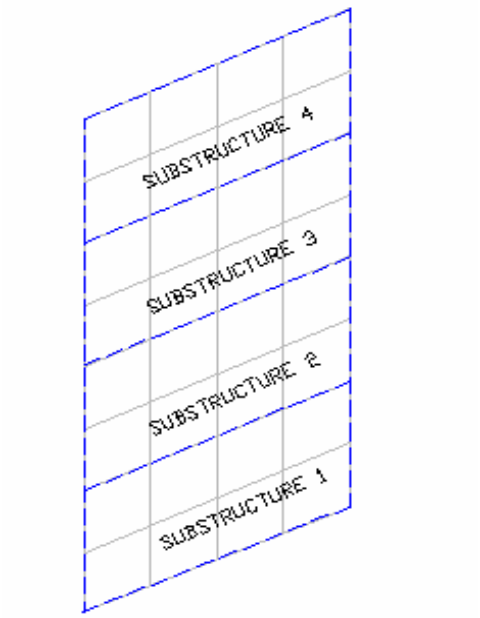


Fig 6.13: Division of plate into four substructures

Results for the deflections and moments for various skew angles are compared with those obtained by analytical methods, given in reference [20]. Variation for maximum deflection for various skew angles and for various descritization of plate is tabulated below in Table 6.3

Table 6.3: Deflection coefficient for maximum deflection for skew plate

Sr No	Angle	Lx	Ly	Nex	Ney	TNE	α_0	
							Calculated	Anlytical
1	0	5	10.00	6	12	72	0.212	0.214
2	0	5	10.00	10	20	200	0.211	0.214
3	0	10	20.00	30	40	1200	0.210	0.214
4	0	10	20.00	26	52	1352	0.198	0.214
5	30	5	8.31	6	12	72	0.115	0.118
6	30	5	8.31	10	20	200	0.114	0.118
7	30	10	16.63	30	40	1200	0.110	0.118
8	30	10	16.63	26	52	1352	0.112	0.118
9	45	5	7.07	6	12	72	0.070	0.070
10	45	5	7.07	10	20	200	0.070	0.070
11	45	10	14.14	30	40	1200	0.071	0.070
12	45	10	14.14	26	52	1352	0.068	0.070
13	60	5	5.00	6	12	72	0.018	0.018
14	60	5	5.00	10	20	200	0.018	0.018
15	60	10	10.00	30	40	1200	0.018	0.018
16	60	10	10.00	26	52	1352	0.018	0.018

Where,

Lx / Ly = length of plate in x and y-direction respectively

Nex / Ney = number of elements along x and y-direction respectively

TNE = Total number of elements

α_0 = W D / q a^4, W is maximum deflection and q is intensity of uniformly distributed load

Variation for maximum moment (My) at the center of plate for various skew angles and for various descritization of plate is tabulated below in Table 6.4

Table 6.4: Moment coefficient for maximum moment at center of plate

Sr No	Lx	Ly	Nex	Ney	TNE	Bo	
						Analysis	Reference
1	5	10	5	11	55	0.497	0.495
2	5	10	11	21	231	0.499	0.495
3	5	10	15	25	375	0.500	0.495
4	5	10	25	51	1275	0.500	0.495
5	5	10	5	11	55	0.364	0.368
6	5	10	11	21	231	0.365	0.368
7	5	10	15	25	375	0.365	0.368
8	5	10	25	51	1275	0.363	0.368
9	5	10	5	11	55	0.285	0.291
10	5	10	11	21	231	0.288	0.291
11	5	10	15	25	375	0.285	0.291
12	5	10	25	51	1275	0.283	0.291

Variation for maximum moment (My) at the center of unsupported edge of plate for various skew angles and for various descritization is tabulated below in Table 6.5

Table 6.5: Moment coefficient for maximum moment at free edge of plate

Sr No	Lx	Ly	Nex	Ney	TNE	B1	
						Analysis	Reference
1	5	10	5	11	55	0.509	0.508
2	5	10	11	21	231	0.511	0.508
3	5	10	15	25	375	0.511	0.508
4	5	10	25	51	1275	0.512	0.508
5	5	10	5	11	55	0.376	0.367
6	5	10	11	21	231	0.373	0.367
7	5	10	15	25	375	0.371	0.367
8	5	10	25	51	1275	0.367	0.367
9	5	10	5	11	55	0.298	0.296
10	5	10	11	21	231	0.296	0.296
11	5	10	15	25	375	0.293	0.296
12	5	10	25	51	1275	0.294	0.296

For present work the plate is descritized into 3000 elements. So plate is having total 9221 nodes. Each node is having three DOFs. So the problem has total 27663 DOFs. Average time required for various processes, when entire structure divided into different number of substructure, is tabulated in Table 6.6. Also time required in parallel implementation i.e. computation time and communication time is shown. Based on the time required for various processes in sequential and parallel implementation, speedup is calculated. The comparison of ideal speedup and observed speedup is shown in Fig6.14. Comparison of communication time and computation time for number of substructures is shown in Fig 6.15. From calculated speedup and ideal speedup, efficiency is also calculated as a measure of performance.

Table 6.6: Time required for sequential and parallel processing

Process	NB	NEQ	size (kb)	time	Sequential time	Parallel time		speed up		Efficiency
						Comp	comm	Ideal	Calculated	
				(sec)	(sec)	(sec)	(sec)			(%)
3 Substructure										
plana1				7						
DTHS			124	20						
plana2	8895	9423		707						
DTHS			5018	50						
plana3	840	1914		140						
DTHS			14	20						
plana4	465	9423		90						
DTHS			446	20						
plana5				7	2545	951	110	3	2.40	79.96
4 substructure										
plana1				7						
DTHS			95	26						
plana2	6675	7143		502						
DTHS			4329	52						
plana3	780	2211		170						
DTHS			13	27						
plana4	465	7143		82						
DTHS			336	22						
plana5				7	2520	768	127	4	2.82	70.39
5 substructure										
plana1				7						
DTHS			77	30						
plana2	5343	5775		315						
DTHS			3940	60						
plana3	744	2508		172						
DTHS			12	30						
plana4	465	5775		65						
DTHS			270	23						
plana5				7	2466	566	143	5	3.48	69.56

6 substructure										
plana1				7						
DTHS			65	33						
plana2	4455	4863		180						
DTHS			3691	65						
plana3	720	2805		175						
DTHS			12	37						
plana4	465	4863		62						
DTHS			226	26						
plana5				8	1642	432	161	6	2.77	46.15

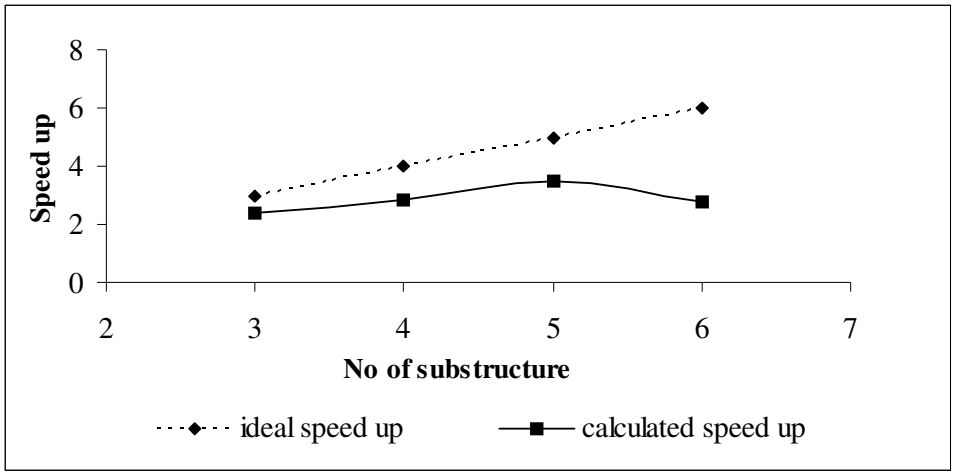


Fig 6.14: Comparison of ideal and observed Speedup

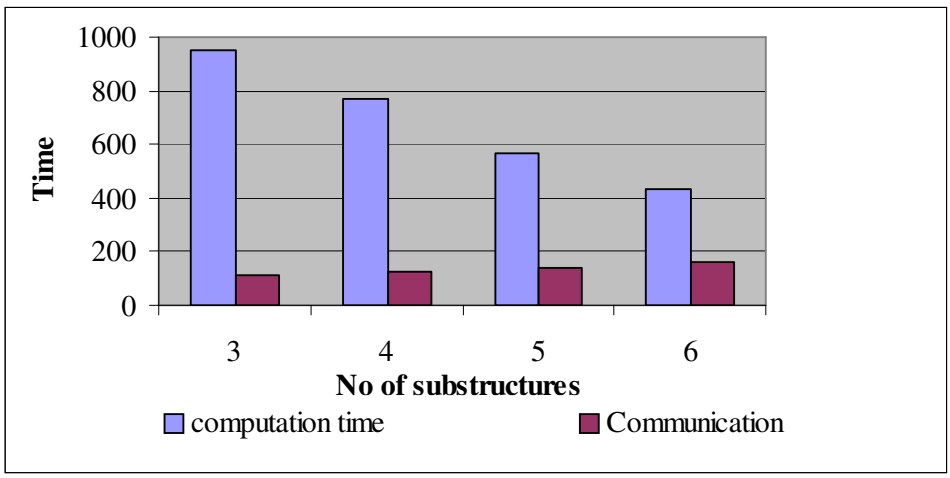


Fig 6.15: Comparison of ideal and observed Speedup

6.6 SUMMARY

Present chapter includes the plate bending problem. So the earlier part of chapter includes the finite element formulation of plate bending element. In later part two problems of plate bending have been analyzed on parallel computers. Of two problems, one is the annular plate subjected to uniformly distributed load. For analysis on parallel computer the plate is discritized into 1800 elements. So, plate has total 5725 nodes (i.e. 17175 DOFs). For implementation on parallel computers, problem is divided into 3, 4 and 6 substructures. The other problem is the skew plate problem subjected to uniformly distributed load. For the analysis, the plate is divided into 3000 elements and 9221nodes (i.e.27663 DOFs). The problem has analyzed using parallel computers that include 3, 4 and 6 computers. For first problem having 17175 DOFs, efficiency of 78.33% has observed. While for second problem, having 27663 DOFs efficiency of 79.96% has observed.

CHAPTER 7 LAMINATE PLATE ANALYSIS

7.1 GENERAL

The word composite means the material consisting by combination of more than one material. Though these materials are combined at microscopic level, individual material is easily distinguishable. The main advantage of composite material is that they exhibit the best qualities of their constituents and often some qualities that neither constituent possesses. So current developments are pointed towards combination of usually strong, fibers and organic, ceramic or metal matrices that promises to be far more efficient than any structural materials known previously. Plates made up of isotropic or orthotropic laminae are widely used in a variety of structures and machines. A multiphase or two material laminae consists of a stiff filament material embedded in a compatible matrix material. Examples of filaments are glass, boron, carbon, graphite, and steel whereas matrix materials have included polyesters, aluminum, and epoxies.

7.2 LAMINATED COMPOSITES

Laminated composites consist of layers of at least two different materials that are bonded together. Lamination is used to combine the best aspects of the constituent layers in order to achieve a more useful material. The properties that can be emphasized by lamination are strength, stiffness, low weight, and corrosion-resistance; wear resistance, beauty or attractiveness, thermal insulation, acoustical insulation etc. Examples of laminated composites are Bimetals, Clad metals, laminated glass, plastic-based laminates, and laminated fibrous composites.

7.2.1 Lamina

A lamina is a flat arrangement of unidirectional fibers as shown in Fig 7.1 in matrix. The fibers are the principal reinforcing or load-carrying agent. They are typically strong and stiff. The matrix can be organic, ceramic, or metallic. The function of matrix is to support and protect the fibers and to provide a means of distributing load among and transmitting load between the fibers.

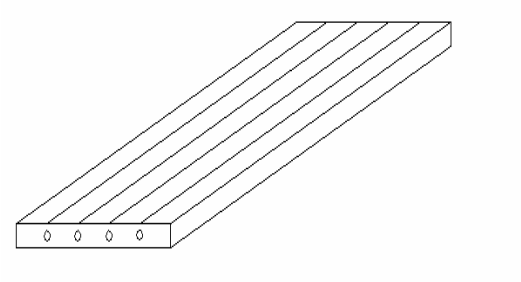


Fig 7.1: Lamina with unidirectional fibers.

7.2.2 Laminate

A laminate is a stack of laminae with various orientations of principal material directions in the laminae as in Fig 7.2. The layers of a laminate are usually bound together by the same matrix material that is used in the laminae. Laminate can be composed of plates of different materials or of laminae of the same material. A laminated circular cylindrical shell can be constructed by winding resin-coated fibers on a mandrel first with one orientation to the shell axis, then another, and so on until the desired thickness is built up.

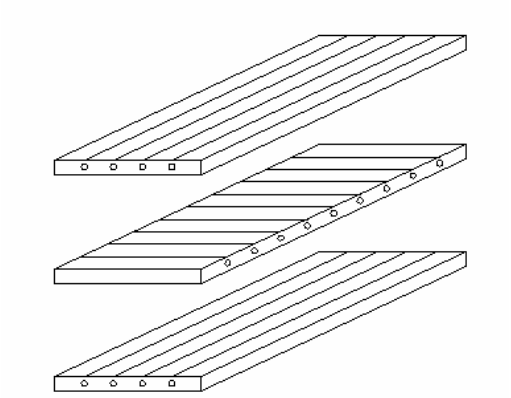


Fig 7.2: 3-ply laminate construction

A major purpose of lamination is to tailor the directional dependence of strength and stiffness of a material to match the loading environment of the structural element. Laminates are uniquely suited to this objective since the principal material directions of each layer can be oriented according to need.

7.2.3 Structural Application

In recent years, continuous fiber reinforced laminated composite plates have been extensively used as structural elements because of their desirable properties such as higher strength-to-weight ratio, higher stiffness-to-weight ratio etc. in addition, there exists the possibility of optimum structural design through the variation of fiber orientation, stacking sequence and choice of fiber and matrix materials.

As far as environmental resistance is concerned, composite materials are more efficient than traditional civil engineering materials such as steel, concrete, masonry, and plaster. Degradation in strength and stiffness for steel structures due to the corrosion problem requires frequent inspection, maintenance, and repair. Similarly, stress cracking due to the warm/cold weathering limits the service life of concrete structures. Timber is susceptible to moisture-swelling problems and paste attack.

Currently, composite materials are being used to retrofit and/or reinforce existing infrastructures. Flat composite laminates have been bonded to the exterior surface of reinforced concrete deck to increase its bending stiffness. Several pedestrian bridges have been built successfully. Composite materials are suitable for construction of highway bridges, power transmission towers, office/residential buildings, retaining walls, etc.

Some of the important structures constructed earlier using Glass-fiber reinforced Polyester (GFRP) are given below:

1) Dome structure in Benghazi in 1968.
2) Roof structures to the Dubai Airport built in 1972.
3) Covert Garden Flower Market at Nine Elms, London.
4) 37m high Chimney at Hendon, London.
5) Prestigious American Express Building in Brighton, England

7.3 DISPLACEMENT MODEL

The displacements in the x, y and z directions of the symmetrically laminated composite plates subjected to transverse loads may be taken as follows. The displacement along the x, y and z directions are expended in terms of higher order functions of thickness coordinates and mid plane variables.

$$U(x, y, z) = z\,\theta_x(x, y, 0) + z^3\,\theta_x^*(x, y, 0)$$
$$V(x, y, z) = z\,\theta_y(x, y, 0) + z^3\,\theta_y^*(x, y, 0)$$
$$W(x, y, z) = w(x, y, 0) + z^2\,w^*(x, y, 0)$$

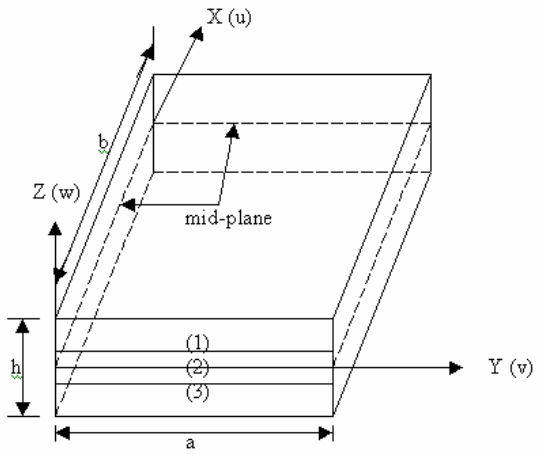


Fig 7.3: Geometry of a rectangular laminated composite plate

This can be written as,

$$u = z\,\theta_x + z^3\,\theta_x^*$$
$$v = z\,\theta_y + z^3\,\theta_y^*$$
$$w = w_0 + z^2\,w_0^* \qquad (7.1)$$

This model includes the effects of the transverse normal strain/stress also.

7.4 STRESS-STRAIN RELATIONS FOR AN ORTHOTROPIC LAMINA

For an orthotropic lamina in a 3-D state, the strain-stress relationship at a point in each of the three orthogonal planes will be given by,

$$\begin{Vmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \gamma_{12} \\ \gamma_{23} \\ \gamma_{13} \end{Vmatrix} = \begin{vmatrix} 1/E_1 & -\nu_{21}/E_2 & -\nu_{31}/E_3 & 0 & 0 & 0 \\ -\nu_{12}/E_1 & 1/E_2 & -\nu_{32}/E_3 & 0 & 0 & 0 \\ -\nu_{13}/E_1 & -\nu_{23}/E_2 & 1/E_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/G_{12} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/G_{23} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/G_{13} \end{vmatrix} \begin{Vmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \tau_{12} \\ \tau_{23} \\ \tau_{13} \end{Vmatrix}$$

$$\varepsilon = s\,\sigma \qquad (7.2)$$

The stress-strain constitutive relations can be obtained by inversion of strain-stress relations given by equation 7.2 and are written in following matrix form :

$$\begin{Vmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \tau_{12} \\ \tau_{23} \\ \tau_{13} \end{Vmatrix}^k = \frac{1}{\Delta} \begin{vmatrix} E_1(1-\nu_{23}\nu_{32}) & E_1(\nu_{21}+\nu_{31}\nu_{23}) & E_1(\nu_{31}+\nu_{21}\nu_{32}) & 0 & 0 & 0 \\ E_2(\nu_{12}+\nu_{13}\nu_{32}) & E_2(1-\nu_{13}\nu_{31}) & E_2(\nu_{32}+\nu_{12}\nu_{31}) & 0 & 0 & 0 \\ E_3(\nu_{13}+\nu_{12}\nu_{23}) & E_3(\nu_{23}+\nu_{21}\nu_{13}) & E_3(1-\nu_{12}\nu_{21}) & 0 & 0 & 0 \\ 0 & 0 & 0 & \Delta G_{12} & 0 & 0 \\ 0 & 0 & 0 & 0 & \Delta G_{23} & 0 \\ 0 & 0 & 0 & 0 & 0 & \Delta G_{13} \end{vmatrix}^k \begin{Vmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \gamma_{12} \\ \gamma_{23} \\ \gamma_{13} \end{Vmatrix}^k$$

In which, $\Delta = (1-\nu_{12}\nu_{21}-\nu_{23}\nu_{32}-\nu_{31}\nu_{13}-2\nu_{12}\nu_{23}\nu_{31})$

$$\sigma = c\,\varepsilon \qquad (7.3)$$

In the stress-strain relation equation 7.3, the subscript k is introduced to designate k^{th} layer of the laminate. The relations given by equation 7.3 are the stress-strain constitutive relations with reference to lamina axes for a homogeneous orthotropic layer in a general 3-D state of stress and these are adopted here to develop a theory based on the displacement model given by equation 7.1

As noted earlier, the relation given by equation 7.3 is the stress-strain constitutive relations for the orthotropic lamina referred to the lamina's principal axes (1,2,3). The principal material axes of a lamina may not coincide with the reference axes for the laminated plate. It is therefore necessary to transform the constitutive relation 7.3 from the lamina principal axes (1,2,3) to the reference axes of the laminate (x, y, z).

$$\sigma' = T\,\sigma \qquad \text{and} \qquad \varepsilon' = T\,\varepsilon \qquad (7.4)$$

The transformation matrix T is given by,

$$T = \begin{vmatrix} c^2 & s^2 & 0 & 2sc & 0 & 0 \\ s^2 & c^2 & 0 & -2sc & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ -sc & sc & 0 & (c^2 - s^2) & 0 & 0 \\ 0 & 0 & 0 & 0 & c & -s \\ 0 & 0 & 0 & 0 & s & c \end{vmatrix} \quad (7.5)$$

Where, $c = \cos \alpha$ and $s = \sin \alpha$.

The relation between engineering and tensor strain vectors is given by,

$$\varepsilon = R\, \varepsilon_{ts}$$

$$\varepsilon_{ts} = R^{-1}\, \varepsilon \quad (7.6)$$

R matrix is defined as,

$$R = \begin{vmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{vmatrix} \quad (7.7)$$

The stress-strain constitutive relations with reference to laminate axes are obtained in the following form by making use of relations 7.3, 7.4 and 7.7

$$\sigma = T^{-1}\, C\, R\, T\, R^{-1}\, \varepsilon \quad (7.8)$$

It can easily be proved that,

$$R\, T\, R^{-1} = T^{-1t} \quad (7.9)$$

Thus, the relation (7.8) can be rewritten as,

$$\sigma = Q\, \varepsilon \quad (7.10)$$

Where,

$$Q = T^{-1}\, C\, T^{-1t}$$

In matrix form,

$$\begin{vmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{xy} \\ \tau_{yz} \\ \tau_{xz} \end{vmatrix} = \begin{vmatrix} Q_{11} & Q_{12} & Q_{13} & Q_{14} & 0 & 0 \\ Q_{12} & Q_{22} & Q_{23} & Q_{24} & 0 & 0 \\ Q_{13} & Q_{23} & Q_{33} & Q_{34} & 0 & 0 \\ Q_{14} & Q_{24} & Q_{34} & Q_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & Q_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & Q_{66} \end{vmatrix} \begin{vmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{xz} \end{vmatrix} \qquad (7.11)$$

Q matrix coefficients are defined as,

$$\begin{aligned}
Q_{11} &= C_{11}c^4 + (2C_{12}+4C_{44})\, s^2c^2 + C_{22}\, s^4 \\
Q_{12} &= (s^4 + c^4)\, C_{12} + (C_{11}+ C_{22} - 4\, C_{44})\, s^2\, c^2 \\
Q_{13} &= c^2\, C_{13} + s^2\, C_{23} \\
Q_{14} &= (C_{11}- C_{12} - 2C_{44})\, c^3s + (C_{12} - C_{22} + 2C_{44})\, s^3c \\
Q_{22} &= C_{11}\, s^4 + C_{22}\, c^4 + (2C_{12} + 4C_{44})\, s^2\, c^2 \\
Q_{23} &= c^2\, C_{23} + s^2\, C_{13} \\
Q_{24} &= (C_{11}- C_{12} - 2C_{44})\, s^3c + (C_{12} - C_{22}+ 2C_{44})\, c^3s \\
Q_{33} &= C_{33} \\
Q_{34} &= (C_{13} - C_{23})\, sc \\
Q_{44} &= (C_{11} + C_{22} - 2C_{12} - 2C_{44})\, s^2\, c^2 + (c^4 + s^4)\, C_{44} \\
Q_{55} &= c^2\, C_{55} + s^2\, C_{66} \\
Q_{56} &= (C_{66} - C_{55})\, sc \\
Q_{66} &= s^2\, C_{55} + c^2\, C_{66}
\end{aligned} \qquad (7.12)$$

And the coefficients of C matrix in equation 7.12 are defined by equation 7.3

7.5 STRAIN-DISPLACEMENT RELATIONSHIPS

Strain expressions corresponding to model (equation 7.1) are,

$$\begin{aligned}
\varepsilon_x &= \partial u / \partial x = z\, K_x + z^3\, K_x^{*} \\
\varepsilon_y &= \partial v / \partial y = z\, K_y + z^3\, K_y^{*} \\
\varepsilon_z &= \partial w / \partial z = z\, K_z \\
\gamma_{xy} &= \partial u/\partial y + \partial v/\partial x = z\, K_{xy} + z^3\, K_{xy}^{*}
\end{aligned}$$

$$\gamma_{yz} = \partial v/\partial z + \partial w/\partial y = \phi_y + z^2 \phi_y^*$$
$$\gamma_{xz} = \partial u/\partial z + \partial w/\partial x = \phi_x + z^2 \phi_x^* \qquad (7.13)$$

Where the definitions of the various terms are as follows:

$$K_x = \partial\theta_x/\partial x, \quad K_y = \partial\theta_y/\partial y, \quad K_{xy} = \partial\theta_x/\partial y + \partial\theta_y/\partial x,$$
$$K_x^* = \partial\theta_x^*/\partial x, \quad K_y^* = \partial\theta_y^*/\partial y, \quad K_{xy}^* = \partial\theta_x^*/\partial y + \partial\theta_y^*/\partial x,$$
$$\phi_x = \theta_x + \partial w_0/\partial x, \quad \phi_y = \theta_y + \partial w_0/\partial y,$$
$$\phi_x^* = 3\theta_x^* + \partial w_0^*/\partial x, \quad \phi_y^* = 3\theta_y^* + \partial w_0^*/\partial y,$$
$$K_z = 2\, w_0^* \qquad (7.14)$$

The concise matrix form of equation 7.13 is,

$$\varepsilon_b^k = \begin{vmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{xy} \end{vmatrix} = z \begin{vmatrix} K_x \\ K_y \\ K_z \\ K_{xy} \end{vmatrix} + z^3 \begin{vmatrix} K_x^* \\ K_y^* \\ K_z^* \\ K_{xy}^* \end{vmatrix} = z\,K + z^3\,K^* \qquad (7.15a)$$

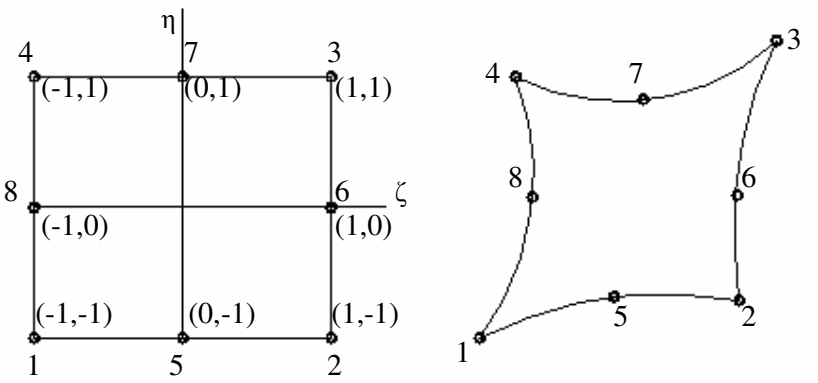
$$\varepsilon_s^k = \begin{vmatrix} \gamma_{yz} \\ \gamma_{xz} \end{vmatrix} = \begin{vmatrix} \phi_y \\ \phi_x \end{vmatrix} + z^2 \begin{vmatrix} \phi_y^* \\ \phi_y^* \end{vmatrix} = \phi + z^2\,\phi^* \qquad (7.15b)$$

The above equations 7.15a and 7.15b are the expressions for the flexure and transverse shear strains respectively, at any point in the k^{th} layer of the laminate located at a distance z from the mid-plane. It should be noted that owing to the nature of equation 7.15b, the transverse shear strains vary parabolically through the plate thickness.

7.6 FINITE ELEMENT FORMULATION

The solution of the fundamental equations of the two displacement models based on higher order shear deformation theory for laminates anisotropic plates, can conveniently be obtained by using the finite element displacement formulation. Element properties are derived by assuming a displacement function, which ensures completeness within the element and compatibility across the element boundaries. The finite element theory is developed in this section for application to linear equilibrium problems of isotropic, orthotropic and multiplayer anisotropic plates with various loading and boundary conditions. In present work, 8-noded

isoparametric quadrilateral element (Fig 7.4) is used. The finite element formulation starts with writing the shape functions, followed by the derivation of the strain-displacement matrix [B]. And in final, element stiffness matrix formulation is derived.



Rectangle in natural coordinate Isoparametric element

Fig 7.4: Eight nodded Isoparametric element

The vector,

$q = [w_{01}, \theta_{x1}, \theta_{y1}, w_0^*{}_1, \theta_x^*{}_1, \theta_y^*{}_1, w_{02}, \theta_{x2}, \theta_{y2}, w_0^*{}_2, \theta_x^*{}_2, \theta_y^*{}_2 \ldots\ldots\ldots\ldots, \theta_y^*{}_8]$ denotes the element displacement vector and hence the degree of freedom at each node is enumerated as:

w_0 = Transverse displacement at the geometrical mid-plane.

θ_x, θ_y = Rotations of the 'normal' to the geometrical mid-plane in x-z and y-z plane respectively.

w_0^* = Higher order term of transverse displacement w_0 at the geometrical mid-plane.

θ_x^*, θ_y^* = Higher order terms of rotations of the 'normal' to the geometrical mid-plane in x-z and y-z plane i.e. θ_x and θ_y respectively.

Therefore, Nodal degree of freedom for the element : 6

Number of nodes in the element : 8

Total degree of freedom for the element : 6 x 8 = 48

7.6.1 Shape Functions

The shape functions for this element in terms of the non-dimensional coordinate system can be given as:

$$N_1 = \xi (\xi - 1)\, \eta(\eta-1) / 4$$
$$N_2 = (1- \xi^2)\, \eta\, (\eta-1) / 2$$
$$N_3 = \xi (\xi + 1)\, \eta(\eta-1) / 4$$
$$N_4 = \xi (\xi + 1)\, (1-\eta^2) / 2$$
$$N_5 = \xi (\xi + 1)\, \eta(\eta+1) / 4$$
$$N_6 = (1- \xi^2)\, \eta(\eta+1) / 2$$
$$N_7 = \xi (\xi - 1)\, \eta(\eta+1) / 4$$
$$N_8 = \xi (\xi - 1)\, (1-\eta^2) / 2 \qquad (7.16)$$

Where, ξ and η are the non-dimensional coordinates (Fig7.4) of a given point on the element.

Now the displacement field is expressed in terms of the nodal values. Thus, if $d = [w_0, \theta_x, \theta_y, w_0^*, \theta_x^*, \theta_y^*]^T$ represents the displacement components of a point located at (ξ,η), and q is the element displacement vector, then

$$w_0 = N_1 w_{01} + N_2 w_{02} + \ldots.. + N_8 w_{08}$$
$$\theta_x = N_1 \theta_{x1} + N_2 \theta_{x2} + \ldots\ldots\ldots + N_8 \theta_{x8}$$
$$\theta_y = N_1 \theta_{y1} + N_2 \theta_{y2} + \ldots\ldots\ldots + N_8 \theta_{y8}$$
$$w_0^* = N_1 w_{01}^* + N_2 w_{02}^* + \ldots.. + N_8 w_{08}^*$$
$$\theta_x^* = N_1 \theta_{x1}^* + N_2 \theta_{x2}^* + \ldots\ldots\ldots + N_8 \theta_{x8}^*$$
$$\theta_y^* = N_1 \theta_{y1}^* + N_2 \theta_{y2}^* + \ldots\ldots\ldots + N_8 \theta_{y8}^* \qquad (7.17)$$

Where,

$$N_{(6x48)} = \sum_{i=1}^{NN} \begin{vmatrix} N_i & 0 & 0 & 0 & 0 & 0 \\ 0 & N_i & 0 & 0 & 0 & 0 \\ 0 & 0 & N_i & 0 & 0 & 0 \\ 0 & 0 & 0 & N_i & 0 & 0 \\ 0 & 0 & 0 & 0 & N_i & 0 \\ 0 & 0 & 0 & 0 & 0 & N_i \end{vmatrix} \quad (7.18)$$

7.6.2 Strain-Displacement relation Matrix [B]:

The strain-displacement matrix relating strain components to element nodal variables can be formed as:

$$[\varepsilon] = [B]\,[\delta] \quad (7.19)$$

Where,

$[\delta]$: vector consisting of all the nodal displacements.

$[\delta]^T = [\,\delta_1\,,\,\delta_2\,,\,\delta_3\,,\,\delta_4\,,\,\delta_5\,,\,\delta_6\,,\,\delta_7\,,\,\delta_8\,]^T$

Where each;

$[\delta]_i^T = [w_0,\, \theta_x,\, \theta_{y,}\, w_0^*,\, \theta_x^*,\, \theta_y^*]_i^T$ for i = 1 to 8.

$[\varepsilon]$: strain vector

$[B]$: strain-displacement matrix.

Now, we are considering the flexure strain terms and shear strain terms separately and from equation 7.15, writing the strain-displacement relationship in terms of the bending curvature-displacement relation $[B_b]$ and shear rotation-displacement relation $[B_s]$.

The shear rotation – displacement relations are,

$$\phi = \begin{vmatrix} \phi_x \\ \phi_y \\ \phi_x^* \\ \phi_y^* \end{vmatrix} = \begin{vmatrix} \theta_x + \partial w_0/\partial x \\ \theta_y + \partial w_0/\partial y \\ 3\theta_x^* + \partial w_0^*/\partial x \\ 3\theta_y^* + \partial w_0^*/\partial y \end{vmatrix} \quad (7.20)$$

And,

The bending curvature-displacement relations are,

$$K = \begin{Bmatrix} K_x \\ K_y \\ K_{xy} \\ K_x^* \\ K_y^* \\ K_{xy}^* \\ K_z \end{Bmatrix} = \begin{Bmatrix} \partial\theta_x/\partial x \\ \partial\theta_y/\partial y \\ \partial\theta_x/\partial y + \partial\theta_y/\partial x \\ \partial\theta_x^*/\partial x \\ \partial\theta_y^*/\partial y \\ \partial\theta_x^*/\partial y + \partial\theta_y^*/\partial x \\ 2\, w_0^* \end{Bmatrix} \qquad (7.21)$$

So, B matrix for curvature and shear can be given as,

$$B_b = \sum_{i=1}^{NN} \begin{bmatrix} 0 & \partial N_i/\partial x & 0 & 0 & 0 & 0 \\ 0 & 0 & \partial N_i/\partial y & 0 & 0 & 0 \\ 0 & \partial N_i/\partial y & \partial N_i/\partial x & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \partial N_i/\partial x & 0 \\ 0 & 0 & 0 & 0 & 0 & \partial N_i/\partial y \\ 0 & 0 & 0 & 0 & \partial N_i/\partial y & \partial N_i/\partial x \\ 0 & 0 & 0 & 2N_i & 0 & 0 \end{bmatrix} \qquad (7.22)$$

(7x48)

$$B_s = \sum_{i=1}^{NN} \begin{bmatrix} \partial N_i/\partial x & N_i & 0 & 0 & 0 & 0 \\ \partial N_i/\partial y & 0 & N_i & 0 & 0 & 0 \\ 0 & 0 & 0 & \partial N_i/\partial x & 3N_i & 0 \\ 0 & 0 & 0 & \partial N_i/\partial y & 0 & 3N_i \end{bmatrix} \qquad (7.23)$$

(4x48)

7.6.3 Element Stiffness Matrix

The virtual work principle has been used to derive the element stiffness matrix and the consistent load vector. If we apply the virtual work principle to a finite element we have,

$\delta U = \delta W$,

Now, the change in internal virtual strain energy δU and the external virtual work δW can be written in terms of the nodal displacements as:

$$\delta U = \int_v \{\delta [\varepsilon_b]^T [\sigma_b] + \delta [\varepsilon_s]^T [\sigma_s]\} dv$$

$$= \int_v \{(\delta[q_i]^T [B_b(x,y)]^T [D_b] [B_b(x,y)] [q_i]) + (\delta[q_i]^T [B_s(x,y)]^T [D_s] [B_s(x,y)] [q_i])\} dv \quad (7.24)$$

$$\delta W = \delta[q_i]^T [p_i^e] \quad (7.25)$$

Where, $[p_i^e]$ is the nodal action due to the externally applied loads.

Next, canceling $\delta[q_i]^T$ from both sides of the equation $\delta U = \delta W$ results in

$$[K^e][q_i] = [p_i^e]$$

Where, $[K^e]$ is the element stiffness matrix, and D_b and D_s vary across thickness z only. Thus, by carrying out the thickness integration through the N layers first, the element stiffness matrix can be written as:

$$[K^e] = \int_A \{[B_b(x,y)]^T [D_b] [B_b(x,y)] + [B_s(x,y)]^T [D_s] [B_s(x,y)]\} dA \quad (7.26)$$

Note that the matrices $[B_b]$ and $[B_s]$ are evaluated based on the shape functions given above. Upon evaluating matrices $[D_b]$, $[D_s]$, $[B_b]$ and $[B_s]$ the element stiffness matrix can be evaluated. However, since the shape functions and, thus, the matrices $[B_b]$ and $[B_s]$ are defined in terms of the non-dimensional coordinate system, the element stiffness matrix must be evaluated as follows:

$$[K^e] = \int_{-1}^{+1} \int_{-1}^{+1} \{[B_b(x,y)]^T [D_b] [B_b(x,y)] + [B_s(x,y)]^T [D_s] [B_s(x,y)]\} [J] \, d\xi \, d\eta \quad (7.27)$$

In this study the Gauss integration technique has been used to evaluate the integrals. In the Gauss integration technique, a polynomial of degree (2n-1) can be integrated exactly by n sampling points. In the present formulation a selective integration scheme has been used to evaluate element stiffness matrix. For the bending stiffness terms 3 x 3 integration scheme and for the shear stiffness terms 2 x 2 integration scheme has been adopted. Thus the stiffness matrix has been evaluated as follows:

$$[K^e] = \sum_{a=1}^{NG} \sum_{b=1}^{NG} \{[B_b(x,y)]^T[D_b][B_b(x,y)] + [B_s(x,y)]^T[D_s][B_s(x,y)]\} [J] W_a W_b \quad (7.28)$$

Where W_a and W_b are the weighting factors corresponding to Gauss sampling points and NG is the number of Gauss points selected for the integration schemes.

7.6.4 Load Vector

The components of the consistent load vector are the equivalent load applied at the nodal points of the element due to the loads applied at the intermediate points of a finite element. In the evaluation of the load vector the entire laminate is considered as a single layer of thickness t_i. The applied external forces may consist of independent or combination of the following load cases:

i) Gravity loads.
ii) Uniform normal surface pressure.
iii) Sinusoidal normal surface pressure.
iv) Point loads in the global z directions.

1) Gravity Load

The gravity loads, generally the self-weight of the element, always act in the global z-direction. In other words, the gravity loads will have the components along w direction only. Let 'ρ' be the uniform mass density of the element material and 'g' be the acceleration due to gravity in z-direction. The element load vector at node i is given by,

$$P_{gi} = \int_A \rho\, g\, t\, [N_i]^T\, dA \quad (7.29)$$

$$[P_g^e] = \sum_{a=1}^{NG} \sum_{b=1}^{NG} \rho\, g\, t\, [N_i]^T\, |J|\, W_a\, W_b \quad (7.30)$$

The above equation represents the element load vector for all the nodes.

2) Uniform normal surface pressure

To evaluate the nodal loads due to normal surface pressure Po, the displacement normal to the surface of the element is required. As here, there is only the transverse displacement, the transverse normal pressure acting either innermost or outermost surface is considered. The load vector at node i is given by,

$$[P_{pi}] = \int_A P_0 [N_i]^T \, dA \tag{7.31}$$

$$[P_p^e] = \sum_{a=1}^{NG} \sum_{b=1}^{NG} P_0 [N_i]^T \, |J| \, W_a \, W_b \tag{7.32}$$

The above equation represents the element load vector for all the nodes.

3) Sinusoidal normal surface pressure

The load vector at node i due to sinusoidal distributed normal pressure is obtained from equation 7.32 by replacing P_0 by,

$$P_0 \ \frac{\sin m\pi x}{a} \ \frac{\sin n\pi y}{b} \tag{7.33}$$

Where, P_0 is amplitude of loading in the z-direction and the element load vector is given by equation 7.33

4) Point load along the transverse direction

When the point of application is not coincident with nodal point and P_{pt} be the point load normal to the surface of the element, the load vector at node i is given by,

$$[P_{pti}] = P_{pt} [N_i]^T \tag{7.34}$$

7.7 SOLUTION PROCEDURE AND COMPUTER PROGRAM

The finite element method is adopted for analysis. In this solution method, first of all the element stiffness matrix and consistent load vector is calculated and then theses stiffness matrices and load vectors are assembled as per the boundary conditions. From this assembled stiffness matrix and load vector, nodal displacements are calculated and finally from these nodal displacements, element stresses are calculated.

In present study, substructure analysis technique is implemented for the finite element analysis. This divides the whole analysis process into five different tasks. So for each process, different computer programs are prepared in "c" programming language. The first program "**lamana1**" prepares separate data file for individual substructure. This data file includes the details such as number of elements in a particular substructure, number of nodes for that substructure, material property, loading data, number of restrained nodes, joint restraints and also the boundary nodes (needed for static condensation) for that particular substructure. After execution of the first program, in second program "**lamana2**", element stiffness matrix and load vector for is calculated and assembled as per corresponding degree of freedoms. Static condensation of stiffness matrix and load vector is also carried out in the same program. This condensed stiffness matrix and load vector is the input data for third program "**lamana3**", in which degree of freedoms corresponding to boundary nodes are calculated. In forth program "**lamana4**", internal degree of freedoms are calculated using these boundary degree of freedoms. After calculating all degree of freedoms, element stresses are calculated in forth program. And in the fifth program "**lamana5**", all the results of individual substructures are collected to print in final result file.

7.8 ANALYSIS PROBLEM AND RESULTS

Laminated composite plate as shown in Fig 7.5, is selected for analysis. For analysis of composite plate 8 nodded quadrilateral isoparametric element is used. Plate is simply supported at all its edges and subjected to sinusoidal transverse lading. Due to symmetry of plate w.r.t both x and y-axis, only quarter plate is analyzed. Descritization of quarter plate in 8-noddeed isoparametric elements is shown in Fig 7.6. Division of plate into four substructures is shown in Fig 7.7. Each substructure is having same number of elements I order to have same working load on each participating computer.

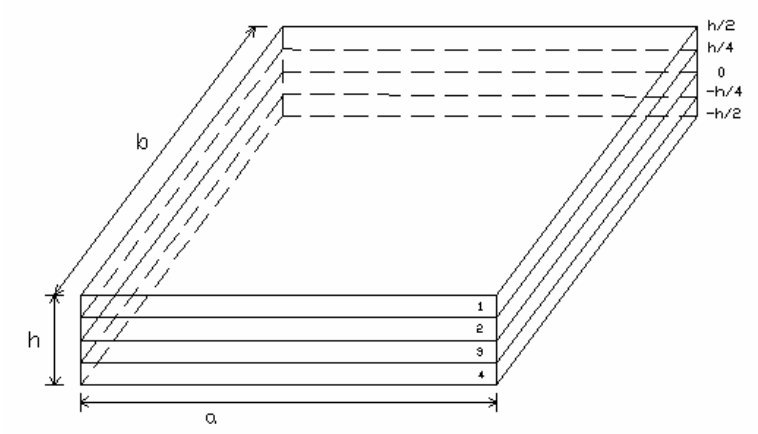


Fig 7.5: Laminated plate having four laminates

Laminate plate that is selected for the analysis is a square plate having both dimensions as 100 x 100cm. Plate consists of four laminates, each of 6.5 cm thick, so the total thickness of plate is 25cm. Intensity of sinusoidal load is 10kN/cm^2. Various material properties for laminated plate are as follows.

$E_1 / E_2 = 25$ $E_3 / E_2 = 1$

$\nu_{12} = \nu_{23} = \nu_{13} = 0.25$

G1 / E2 = 0.2 G2 / E2 = 0.5 G3 / E3 = 0.2

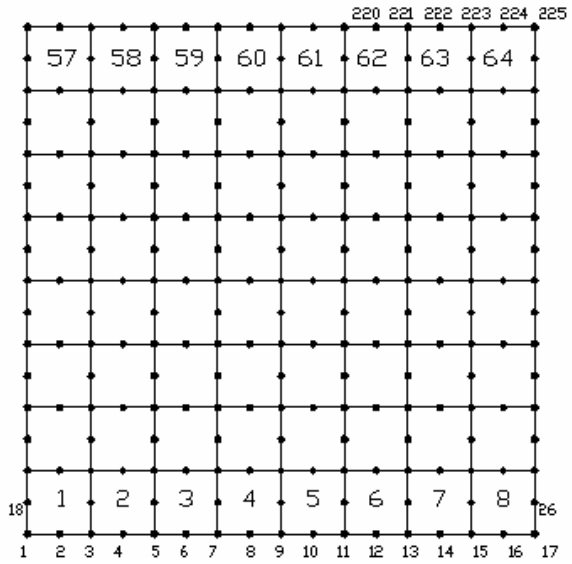


Fig 7.6: Descritization of plate into 8-nodded isoparametric elements

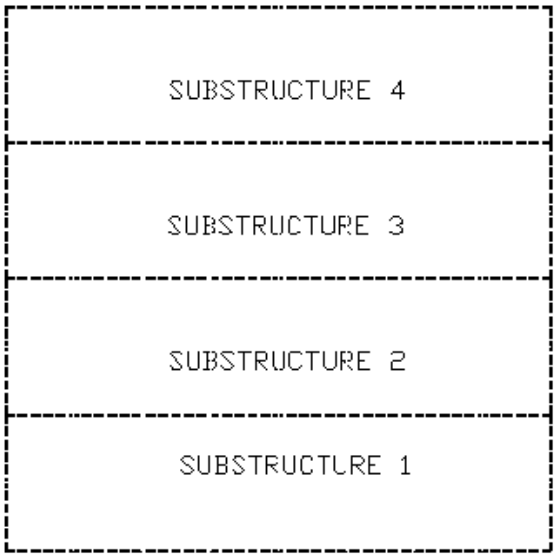


Fig 7.7: Division of plate into four substructures

For the present work, the laminated plate is divided into 1600 elements. So total number of node are 4961. Each node is having six DOFs. So problem is having total 29766 DOFs. Configuration of problem, when divided into four substructures and time requirement for individual process is shown in Fig 7.8 and Fig 7.9 respectively.

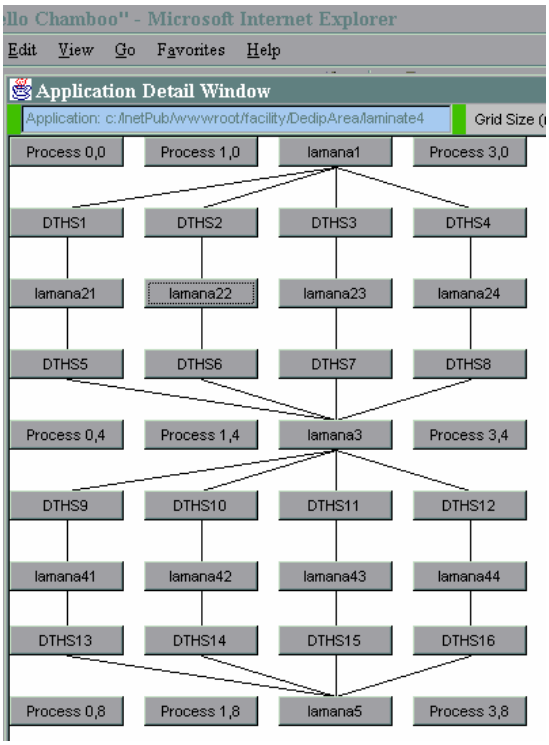
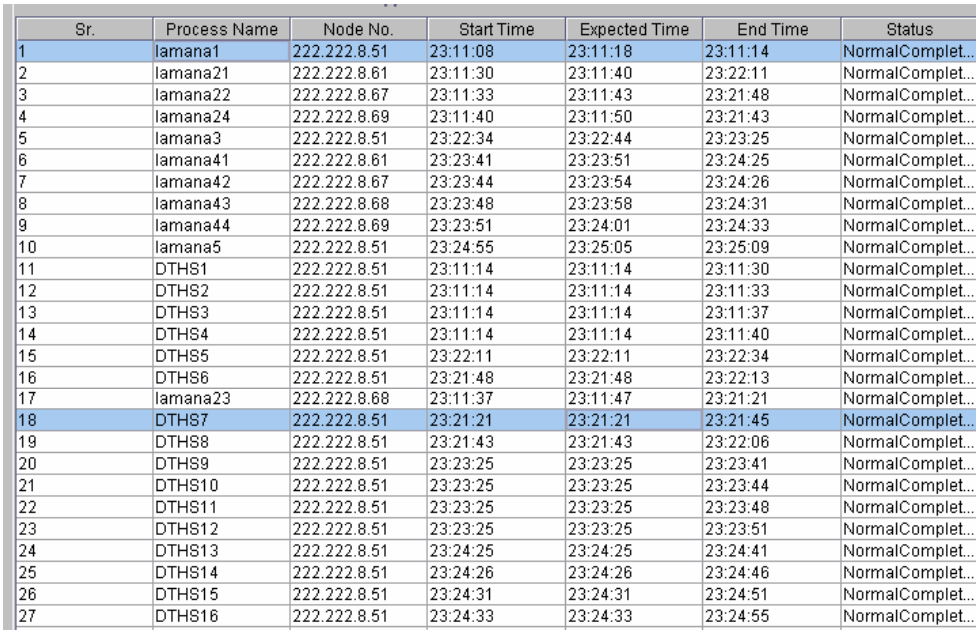


Fig 7.8: Configuration of laminate plate problem for four substructures



Sr.	Process Name	Node No.	Start Time	Expected Time	End Time	Status
1	lamana1	222.222.8.51	23:11:08	23:11:18	23:11:14	NormalComplet...
2	lamana21	222.222.8.61	23:11:30	23:11:40	23:22:11	NormalComplet...
3	lamana22	222.222.8.67	23:11:33	23:11:43	23:21:48	NormalComplet...
4	lamana24	222.222.8.69	23:11:40	23:11:50	23:21:43	NormalComplet...
5	lamana3	222.222.8.51	23:22:34	23:22:44	23:23:25	NormalComplet...
6	lamana41	222.222.8.61	23:23:41	23:23:51	23:24:25	NormalComplet...
7	lamana42	222.222.8.67	23:23:44	23:23:54	23:24:26	NormalComplet...
8	lamana43	222.222.8.68	23:23:48	23:23:58	23:24:31	NormalComplet...
9	lamana44	222.222.8.69	23:23:51	23:24:01	23:24:33	NormalComplet...
10	lamana5	222.222.8.51	23:24:55	23:25:05	23:25:09	NormalComplet...
11	DTHS1	222.222.8.51	23:11:14	23:11:14	23:11:30	NormalComplet...
12	DTHS2	222.222.8.51	23:11:14	23:11:14	23:11:33	NormalComplet...
13	DTHS3	222.222.8.51	23:11:14	23:11:14	23:11:37	NormalComplet...
14	DTHS4	222.222.8.51	23:11:14	23:11:14	23:11:40	NormalComplet...
15	DTHS5	222.222.8.51	23:22:11	23:22:11	23:22:34	NormalComplet...
16	DTHS6	222.222.8.51	23:21:48	23:21:48	23:22:13	NormalComplet...
17	lamana23	222.222.8.68	23:11:37	23:11:47	23:21:21	NormalComplet...
18	DTHS7	222.222.8.51	23:21:21	23:21:21	23:21:45	NormalComplet...
19	DTHS8	222.222.8.51	23:21:43	23:21:43	23:22:06	NormalComplet...
20	DTHS9	222.222.8.51	23:23:25	23:23:25	23:23:41	NormalComplet...
21	DTHS10	222.222.8.51	23:23:25	23:23:25	23:23:44	NormalComplet...
22	DTHS11	222.222.8.51	23:23:25	23:23:25	23:23:48	NormalComplet...
23	DTHS12	222.222.8.51	23:23:25	23:23:25	23:23:51	NormalComplet...
24	DTHS13	222.222.8.51	23:24:25	23:24:25	23:24:41	NormalComplet...
25	DTHS14	222.222.8.51	23:24:26	23:24:26	23:24:46	NormalComplet...
26	DTHS15	222.222.8.51	23:24:31	23:24:31	23:24:51	NormalComplet...
27	DTHS16	222.222.8.51	23:24:33	23:24:33	23:24:55	NormalComplet...

Fig 7.9: Time requirement for individual processes for four substructures

For the problem under consideration, good agreement has been observed with the analytical given in reference [5]. Comparison of calculated results with the reference results is tabulated in Table 7.1

Table 7.1: Comparison of calculated and reference results

Sr. No	**a/h**	**(Constant) w**	**(Constant) σ_x**	**(Constant) σ_y**	**(Constant) σ_z**
		(a/2, b/2, 0)	**(a/2, b/2, h/2)**	**(a/2, b/2, h/4)**	**(0, 0, h/2)**
1	4	1.217E-03	1.134E+02	1.005E+02	7.322E+00
	Calculated	1.90	0.71	0.63	0.05
	Reference	1.87	0.73	0.65	0.04
2	10	7.196E-03	5.608E+02	3.898E+02	2.708E+00
	Calculated	0.72	0.56	0.39	0.03
	Reference	0.70	0.57	0.40	0.03
3	20	4.057E-02	2.169E+03	1.217E+03	9.101E+01
	Calculated	0.51	0.54	0.30	0.02
	Reference	0.48	0.55	0.31	0.02
4	100	4.337E+00	5.155E+04	2.592E+04	2.116E+03
	Calculated	0.43	0.52	0.26	0.02
	Reference	0.41	0.55	0.27	0.02

The average time required for various processes, when entire structure divided into different number of substructure, is tabulated in Table 7.2. Also time required in parallel implementation i.e. computation time and communication time is shown. Based on the time required for various processes in sequential and parallel implementation, speedup is calculated. The comparison of ideal speedup and observed speedup is shown in Fig 7.10. From calculated speedup and ideal speedup, efficiency is also calculated as a measure of performance. Comparison of communication and computation time for different substructures is shown in Fig 7.11

Table 7.2: Time required for sequential and parallel processing

Processes	NB	NEQ	time	Sequential time	Parallel time		speed up		Efficiency
					Comp	comm	Ideal	Calculated	
			(sec)	(sec)	(sec)	(sec)			(%)
3 substructures									
lamana1			6						
DTHS			23						
lamana2	9438	10206	1090						
DTHS			26						
lamana3	1248	2820	44						
DTHS			22						
lamana4	714	10209	48						
DTHS			19	3471	1195	90	3	2.70	90.04
lamana5			7						
4 substructures									
lamana1			6						
DTHS			26						
lamana2	7110	7806	611						
DTHS			25						
lamana3	1200	3342	51						
DTHS			26						
lamana4	750	7806	43						
DTHS			22	2687	725	99	4	3.26	81.52
lamana5			14						
6 substructures									
lamana1			8						
DTHS			23						
lamana2	4734	5334	181						
DTHS			67						
lamana3	1080	4170	50						
DTHS			25						
lamana4	714	5334	22						
DTHS			20	1294	279	135	6	3.13	52.09
lamana5			18						

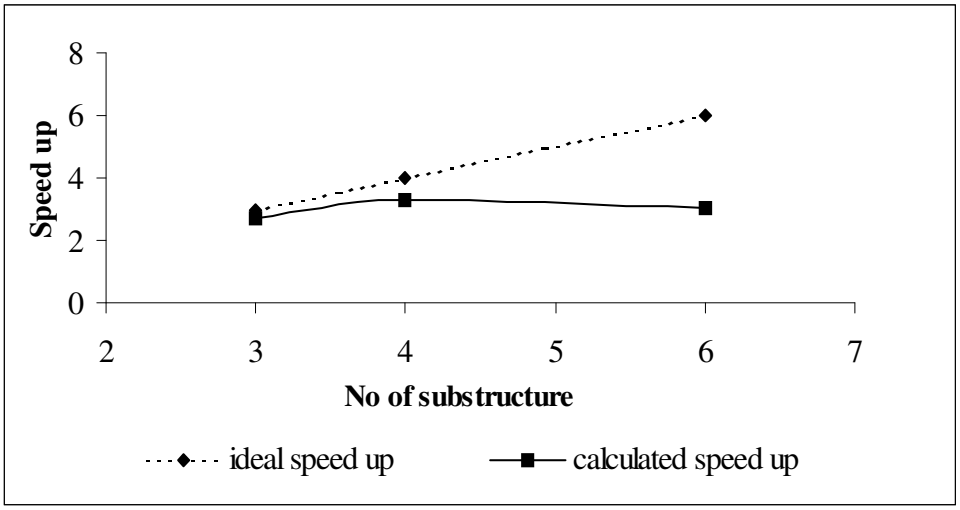


Fig 7.10: Comparison of ideal and observed Speedup

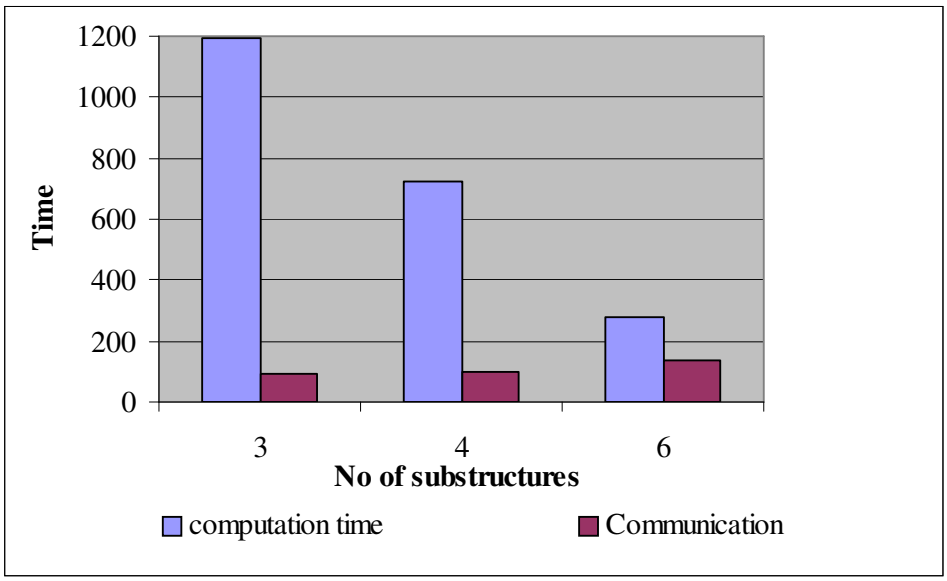


Fig 7.11: Computation and communication time

7.9 SUMMARY

Present chapter includes the laminated plate analysis. So, earlier part of the chapter includes the finite element formulation for laminate plate. In later part, one laminated plate problem has been implemented on 3, 4 and 6 parallel computers for analysis using parallel processing technique. The problem is descritized into 1600 number of elements. So there are total 4961 nodes (i.e. 29766 DOFs). For the problem under consideration having 29766 DOFs,

efficiency of 90% has observed for three computers. As number of computers goes on increasing, the communication time also goes on increasing, so speedup and hence efficiency goes on reducing.

CHAPTER 8 SUMMARY AND FURTHER SCOPE OF WORK

8.1 SUMMARY

In present study, parallel processing has been implemented in structural applications to reduce the computational time. As the parallel programming needs deep knowledge of some of the aspects of parallel programming such as special debugging techniques, data hiding, data sharing, data synchronization etc, it makes parallel programming bit difficult for us. So, in present study WebDedip environment has been used to implement parallel processing on network of computers. WebDedip environment helps user to implement parallel processing on network of computers without using special debugging technique or message passing.

For analysis Finite Element Analysis method has used and substructure technique has implemented in order to implement parallel processing. So in whole process of analysis, the structure is divided into number of substructures and analysis of each substructure is carried out concurrently on separate computer and finally these results are combined to have final results.

Using the above approach, four different problems have analyzed by sequential as well as parallel processing and reduction in computational time has been observed. These four problems are as follows:

- First problem is the plane stress analysis of rectangular plate having circular hole in the center. Plate is subjected to uniform tensile force on both the edges. Constant Stress Triangular (CST) element is used to descritize the continuum. The problem is having total 27980 DOFs and a efficiency of 88.07% is achieved when implemented on three computers.

- Second problem is the analysis of circular plate, having hole in the center. Plate is subjected to uniform lateral load. Plate is fixed at outer circumference and free at inner circumference. For descritization of the plate, eight nodded quadrilateral isoparametric plate element is used. Problem is having total 17175 DOFs and a efficiency of 78.33% has achieved when implemented on three computers.

- Third problem is the skew plate analysis. It is a simply supported plate supported on both inclined edges and subjected to uniformly distributed transverse load over entire plate. For descritization of plate, eight nodded quadrilateral isoparametric plate element is used. The problem is having total 27663 DOFs and efficiency of about 80% has achieved for three computers.

- Analysis of laminated composite plate is the third problem. For analysis of composite plate eight nodded quadrilateral isoparametric element is used. Plate is simply supported at all its edges and subjected to sinusoidal transverse lading. The continuum is descritized to have 29766 DOFs and efficiency, when implemented on three computers of 90% has been achieved.

8.2 CONCLUSION

From the work carried out following conclusions can be drawn.

- As the hardware, designed exclusively for high performance computing is expensive, the parallel processing technique, using network of computers can provide a cost effective solution for high performance computing.

- WebDedip is a user-friendly environment by which user can implement parallel application without having difficulty of special debugging and message passing techniques.

- For a problem, implemented on less number of computers, computation time is more and communication time is less and the same problem, when implemented on more number of computers, total processing time reduces but at the same time communication time becomes high so over all efficiency is reduced.

- More number of computers does not always serve the purpose. For small size of problem, when implemented on more number of computers, computational time is less as compared to communication time and hence less efficiency is achieved. So number of computers should be decided on the basis of the size of problem.

- With the implementation of parallel processing technique on network of computers using WebDedip environment, about 90% efficiency is observed. Hence it can be said that, parallel processing on network of computers serves as one of the cost effective tool for high performance computing.

8.3 FURTHER SCOPE OF WORK

The field of high performance computing is an upcoming field. So there are many fields in which the further work can be carried out. Some of such fields are as follows.

- In present study, various static applications have been solved using parallel processing technique. So in further work various dynamic and nonlinear problems such as dynamic analysis of laminated composite laminates can be analyzed using parallel processing.

- In present study, WebDedip environment is used for parallel implementation of problem. So, as further work parallel processing can be carried out by parallel programming, using PVM and MPI.

- In present work load balancing is achieved by keeping the size of each substructure same. In further work, dynamic load balancing can be implemented along with WebDedip using Message Passing Interfaces.

- Cluster computing, Meta computing and Grid computing are also some of the upcoming fields for high performance computing. So further work can be carried out in these fields also.

REFERENCES

1) Saxena M, Perucchio R, ”Parallel FEM algorithms based on recursive spatial decomposition –I. Automatic mesh generation”, Computers and structures, Vol - 45, pg.817-831, 1992

2) Adeli H, Kamal O, ”Concurrent analysis of large structures –I, Algorithms”, Computers and structures,Vol - 42, pg.413-424, 1992

3) Adeli H, Kamal O, ”Concurrent analysis of large structures –II, Applications”, Computers and structures, Vol - 42, pg.425-432, 1992

4) Foley C M, Vinnakota S, ”Parallel processing in the elastic non linear analysis of high rise frameworks”, Computers and structures, Vol - 52, pg.1169-1179, 1994

5) Sivakumaran K S, Chowdhury S H and Vajarasathira, “ Some studies on finite elements for laminated composite plates”, Computers & Structures, Vol – 52, pg. 729-741, 1994

6) Adeli H, Kumar S, ”Distributed finite element analysis on network of workstations - algorithms”, Journal of structural engineering, Vol - 121, pg.1448-1455, 1995

7) Adeli H, Kumar S, ”Distributed finite element analysis on network of workstations - implementation and applications”, Journal of structural engineering, Vol - 121, pg.1456-1462, 1995

8) Adeli H, Kumar S, ”Minimum weight design of large structures on a network of workstations”, Microcomputers in civil engineering, Vol - 10, pg. 423-432, 1995

9) Kahaner D K, “Parallel Processing Efforts in India”, Asian Technology Information Program (ATIP), ATIP96.040, 1996

10) Noor A K, ”New computing systems and future high performance computing environment and their impact on structural analysis and design”, Computers and structures, Vol - 64, pg.1-30, 1997

11) Wriggers P, Boersma A, ”A parallel algebraic multigrid solver for problems in solid mechanics descritized by finite elements”, Computers and Structures, Vol - 69, pg. 129-137, 1998

12) Soegiarso R, Adeli H, ”Parallel vector algorithm for optimization of large steel structures on a shared memory machine,” Computer aided civil and infrastructure engineering, Vol - 13, pg. 207-217, 1998

13) Zucchini A, ”A parallel preconditioned conjugate gradient solution method for finite element problem with coarse-fine mesh formulation”, Computers and structures, Vol - 78, pg. 781-787, 2000

14) Adeli H,” High performance computing for large scale analysis, optimization, and control”, Journal of aerospace engineering, Vol - 13, pg. 1-10, 2000

15) S C Patodi, P V Patel and H S Bhatt, ”Distributed Finite Element Analysis Using WebDedip Environment, Recent Developments in Structure Analysis (SEC-2001), pg.628-635, 2001

16) Sotelino E D, ”Parallel processing techniques in structural engineering applications”, Journal of structural engineering, Vol - 129, pg.1698-1703, 2003

17) Timoshenko S and J N Goodier, “Theory of Elasticity”, McGraw Hill Publishing Co. Ltd., New York, 1970

18) Desai C S and Abel John F, “Introduction to finite element method”, CBS publishers & distributors, New Delhi, 1987

19) Cook R D, Malkus D S and Plesha M E, “Concepts and application of finite element analysis”, John Wiley & Sons, 1989

20) Timoshenko S. P. and Krieger S. W, “Theory of plates and shells”, McGraw-Hill book company, 1989

21) Adeli H, “Parallel processing in computational mechanics”, Mareel Dekker Inc., New York, 1992

22) Reddy J N, “Finite element method”, McGraw Hill Publishing Company, London, 1993

23) Rajaraman V, “Supercomputers”, Wiley Eastern limited, 1993

24) Topping B H V and Khan A I, “Parallel Finite element computations”, Saxe-Coburg publications, 1996

25) Krishnamoorthy C S, “Finite Element Analysis - Theory and Programming”, Tata McGraw Hill Publishing Co. Ltd., New Delhi, 1996

26) Adeli H, Roesdiman and Soegiarso, “High performance computing in structural engineering”, CRC Press, 1998

27) Adeli H and Kumar S, “Distributed computer aided engineering for analysis, design and visualization”, CRC Press, 1999

28) Rajkumar B, “High performance cluster computing (Architecture and system)”, Prentice Hall Inc., New York, 1999

29) Sasikumar M, Shikhare D and Ravi Prakash P, “Introduction to Parallel Processing”, Prentice Hall of India Private Ltd., New Delhi, 2000

30) Chandruptla T R and Belegundu A D, “Introduction to finite elements in engineering”, Pearson education Inc., 2002

31) Zienkiewicz O C, “The finite element method”, Tata McGraw Hill Publishing Company Ltd., 2004

PAPER PUBLICATION

- **Paper published**

  Vikas Saxena, Prof Paresh V Patel, "APPLICATION OF PARALLEL PROCESSING IN STRUCTURAL ANALYSIS", National Conference on Recent Developments in Materials and Structures (REDEMET-2004) held at NIT, Calicut, December 2004, pg 263-271

- **Paper Accepted**

  P V Patel, Vikas P Saxena, Dr. S C Patodi, "SUBSTRUCTURE BASED DISTRIBUTED FINITE ELEMENT ANALYSIS OF PLATES", International Conference on Recent Advances in Concrete and Construction Technology (INCRAC & CT-2005) going to held at SRM Institute of Science and Technology, Chennai, December 2005