Application of Parallel Computing in Finite Element Analysis of Two-Dimensional Small and Large Deformations

THESIS

Submitted in partial fulfilment
of the requirements for the degree of
DOCTOR OF PHILOSOPHY

by

Rajendra Narayan Khapre

Under the Supervision of

Dr. Pramod Kumar Gupta

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
PILANI (RAJASTHAN) INDIA
2006

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE

PILANI (RAJASTHAN)

CERTIFICATE

This is to certify that the thesis entitled **"Application of Parallel Computing in Finite Element Analysis of Two-Dimensional Small and Large Deformations"** submitted by **Rajendra Narayan Khapre** ID. No. **1999PH02409** for award of Ph. D. Degree of the Institute, embodies the original work done by him under my supervision.

	Signature in full of the Supervisor	[illegible]
	Name in capital block letters	**PRAMOD KUMAR GUPTA**
Date: 27/01/06	**Designation:**	Assistant Professor of Civil Engineering

ACKNOWLEDGEMENTS

I wish to express deepest gratitude and heartfelt indebtedness to my thesis supervisor Dr. Pramod Kumar Gupta, Assistant Professor, Civil Engineering Group for his valuable and continuous guidance, encouragement, suggestions, and moral support throughout the period of this research work. It has been a privilege for me to work under his valuable guidance.

Gratitude is also accorded to BITS, Pilani for providing all the necessary facilities to complete the research work. My sincere thanks to Prof. S. Venkateswaran, Vice-Chancellor of the Institute for allowing me to pursue my research work successfully. My special thanks to Prof. L. K. Maheshwari – Pro. Vice-Chancellor and Director (Pilani), Prof. K. E. Raman – Deputy Director (Administration), Prof. V. S. Rao – Deputy Director (Off-Campus Programmes), Prof. A. K. Sarkar – Dean Faculty Division I and Instruction Division, Prof. H. S. Moondra – Chief, Maintenance Unit, for providing the necessary infrastructure and other facilities.

Sincere thanks to Prof. J. P. Misra – Unit chief (Information Processing Centre) for providing excellent computing facilities for the present research work. I am also thankful to Dr. M. Ishwara Bhat, Librarian, BITS for providing library facility.

I also express my gratitude for the kind and affectionate enquiries about the work and the encouragement given by Prof. Ravi Prakash - Dean (Research and Consultancy Division) and Dr. S. D. Pohekar of the same Division.

Much appreciation is expressed to Dr. Manoj Kumar, Dr. K. S. Raju, Prof. (Dr.) Rajiv Gupta, Prof. (Dr) S. P. Gupta and Dr. D. K. Maharaj who were the members of Doctoral Advisory Committee (DAC) for their kind suggestions, moral support, and assistance from time to time.

I am also thankful to Center for Development of Advanced Computing, Pune (C-DAC) for their support given to this research work through research project *Computer Simulation of Large Deformations Process.*

Special thanks and appreciation is extended to Dr. A. P. Singh, Dr. Anshuman, Dr. Rahul Ralegaonkar, Dr. Arun C., Dr. A Vasan, Mr. Manish Kewalramani and other members of Civil Engineering Group for their valuable time and support extended to me in hour of need.

I am thankful to Mr. Vikas Singh, Administrator, PARAM 10000 for his continuous support throughout the research work. I am also thankful to Mr. Balbir Nanglia, Mr. Ashish Chauhan and Mr. Soji George for providing excellent networking facilities and hardware support.

Sincere thanks to Mr. Amit Gaikwad, Research Scholar (IIT, Delhi) for providing research literature throughout the study. I am also thankful to Mr. Pravin S. Talan for his support and constant encouragement. I would like to express gratitude to my close friends Mr. Shrikant Charde and Mr. Anup Deshpande who supported and encouraged all the way for the research work. I also thank my co-researchers Mr. K Venugopal, Mr. K. C. Sati and other research scholars at BITS, Pilani for their constant cooperation and encouragement in completing my research work. I also thank Mr. Parul Jain and Mr. Avinash Sinhal, ex BITS students for their contribution in this work.

I express my thanks to non-teaching staff of Civil Engineering Group and Information Processing Center for their cooperation. I would like to thank one and all who have helped me in myriad ways throughout the course of this work.

Finally a very special expression of appreciation is extended to my father Mr. Narayan S Khapre, mother Mrs. Indu N. Khapre and beloved sisters Mrs. Rajani Bhiwapurkar and Mrs. Ragini Dharmik. Without their encouragement, patience, and understanding, this endeavor would not have been possible.

[illegible]

Rajendra N. Khapre

ABSTRACT

The thesis presents an implementation of parallel computing technique in linear and non-linear finite element codes. For proper implementation of parallel computing technique in finite element codes, computational time expenditure in different steps of finite element solution procedure is presented. It was found that major portion of computational time in computational finite element analysis is consumed in getting solution of generated system of linear equations. Hence, to reduce this computational time, parallel solvers are developed and employed.

Three parallel solvers using Gauss-Seidel Method, Gauss Elimination Method and Matrix Inversion Method are developed in C programming language. Using these developed solvers, several data sets taken from finite element problems, were analyzed on supercomputer PARAM 10000. It was observed that the variation of Total time measured in terms of Real time was not consistent and hence a timer was developed to measure computational time in terms of User time for a particular code segment. With the help of computational time results (measured in terms of Real time and User time), comparison between these developed solvers is carried out. Based on the comparison of performance shown of these solvers, it was found that Matrix Inversion Method parallel solver is better as compared to other the solvers.

Thereafter Matrix Inversion Method parallel solver is developed using FORTRAN77 programming language. After comparing the performance of solvers developed in C and FORTRAN77 languages, it was found that solver developed in C language is faster than the solver developed in FORTRAN77 language. Blocking and Non-blocking communication mechanisms were used in this solver and it was found that both these mechanisms are equally effective. After this, the solver was modified to speedy solution of system of linear equations especially generated in finite element analysis. After comparing the modified solver with original solver, it was found that modified solver is considerably faster than the original solver and hence it was adopted in finite element analysis.

Incorporating the Modified Matrix Inversion Method parallel solver, a code for linear elastic finite element analysis is developed. The code has three noded constant strain triangular elements to discretize the problem domain. It is capable of solving axisymmetric, plane strain and plane stress problems on supercomputer PARAM 10000. With the help of this code, analysis of anchorage zone in prestressed post-tensioned concrete beam was carried out. This analysis was carried out into two parts. In the first part, stress variation in anchorage zone was studied for concentric prestressing loading condition. The effect of Poisson's ratio and load area ratio on stress variation is studied and compared with the available literature. It was found that stress variation obtained from present investigation match well with referred literature. In the second part, effect of eccentricity of prestressing force on anchorage zone stresses is studied. The effect of eccentricity on bursting tensile force is also studied and it was found that bursting tensile force reduces with increase in eccentricity of prestressing force. In both the parts, it was found that spalling zone exists near free corners of the beam. An expression for computing magnitude of bursting tensile force has also been developed with incorporating the parameter Poisson's ratio. The analysis was carried out on supercomputer PARAM 10000 using its multiple processors to save the computational time involved in the analysis.

Second finite element code (FEMLD) was developed based on the flow formulation of the finite element analysis. It is having two types of elements namely three noded constant strain triangular element and four noded rectangular elements to discretize the problem domain. This code was developed to analyze axisymmetric as well as plane strain large deformation problems on supercomputer PARAM 10000. One problem from each category was analyzed and its results were presented in detail by studying contours of various strain-rates, stresses, and nodal velocities. In order to validate the developed code, similar problems were also analyzed using commercial softwares FORGE2 and ANSYS. After comparing various results obtained from developed code with the results obtained from commercial software, it was found that the results match fairly well. The performance of developed code on supercomputer PARAM 10000 was also measured and it was found that the developed code performed well. Capability of the developed solver for its implementation in three-dimensional problems was also tested by solving bigger data sizes. It was found that the developed code performed better for bigger size problems, which proves its suitability in solving three-dimensional problems also.

The developed code FEMLD was further improved to simulate compression process of various shapes between two flat rigid dies. Using this modified code. four problems namely axial compression of solid metallic cylinder, axial compression of hollow metallic tube. lateral compression of rectangular metallic tube and lateral compression of round tube between concentrated loads, were simulated. These four problems were also analyzed using commercial software FORGE2 to validate the developed code. These problems were analyzed using multiple processors of supercomputer PARAM 10000, so various components of computational time were measured. After computing the performance of the developed code, it was found that the developed code performed well.

In order to propose an economical alternative to supercomputer, two Clusters were developed on Windows NT operating system using Local Area Network. The first Cluster consisted of eight PC's having similar configurations connected through Switch whereas second Cluster consisted of eight PC's having different configurations connected through HUB. On these Clusters, four earlier developed solvers were redeveloped and tested. After comparing their performance on Windows NT Cluster having eight PC's with different configurations, it was found that the Matrix Inversion Method parallel solver gave maximum Speedup whereas Modified Matrix Inversion Method parallel solver required minimum computational time. When these results were compared with the results obtained using supercomputer PARAM 10000, it was found that developed solvers performed better on supercomputer PARAM 10000 as compared to the developed Cluster. Finite element codes for linear and non-linear finite element analysis were developed on these Clusters and tested by solving one problem from each category. It was found that computational time reduces with increase in number of computers, but the observed reduction is very less as compared to the reduction observed in case of supercomputer PARAM 10000.

KEYWORDS

Parallel Computing, Finite Element Analysis, Parallel Solver, Anchorage Zone, Large Deformation, Cluster Computing.

TABLE OF CONTENTS

List of Tables

List of Figures

List of Notations

$[A]$	Matrix representing global stiffness matrix
$[A]^{-1}$	Matrix representing inverse of matrix $[A]$
$\mathbf{B}$	strain-rate matrix, strain displacement matrix
$\{B\}$	Vector representing known vector
BW	band width
Cal	Calculation Time
Comm	Communication Time
CPU	CPU time
C&S	Communication and Synchronization time
$\mathbf{D}$	effective strain-rate coefficient matrix
E	Youngs Modulus
$E(\dot{\varepsilon}_{ij})$	work function
F77	FORTRAN77
F_{bst}	bursting tensile force
F_i	surface tractions
$[I]$	Identity matrix
$\mathbf{J}$	Jacobian matrix
$\mathbf{K}$	stiffness matrix
K	penalty constant
M	moment
P	number of processors
P_k	magnitude of prestressing force
$\{Q\}$	nodal load vectors
R	real
RT	Real time
S	Speedup
Sys	System Time
$\mathbf{T}$	coordinate transform matrix.
Total	Total Time
U	user

U_D	tool velocity
UT	User Time
V	volume
X	X component of body force
$\{X\}$	Vector representing unknown vector
$X^{(0)}$	initial guess solution vector
Y	Y component of body force
a	depth of anchor plate
$\{\mathbf{d}\}$	nodal displacement vector
d	depth of beam (2b)
d	size of global stiffness matrix
e	eccentricity of prestressing force
$\mathbf{f}$	residual nodal point force vector.
f_s	frictional stress
i	iteration number
k	load area ratio
k	shear yield stress
l	unit vector in the opposite direction of relative sliding,
m	constant shear friction factor
$\mathbf{n}$	unit normal to the interface surface
n	data size
p	number of processors
t	User time per iteration (in seconds),
t_{calc}	calculations time
t_{comm}	communication time
t_p	Parallel time
t_{parr}	parallel computational time
t_s	Sequential time
u	component of displacement taken parallel to the x axis
u_0	small positive number compared to u_s.
u_s	sliding velocity of a material relative to the die velocity
$\mathbf{v}$	velocity vector

v	component of displacement taken parallel to the y axis
v_0	assumed velocity
v_i^P	adjusted velocity of the node P
y_i^P	Y-coordinate of the node P at (i)th increment
$y_{(i+1)}^{die}$	Y-coordinate of rigid die at (i+1)th increment
y_0	depth of beam
y_{p0}	depth of anchor plate
Δv	velocity correction term
Δt	time increment
α	deceleration coefficient
β	ratio of loaded depth and actual depth of beam
$\{\varepsilon\}$	strain vector
$\bar{\varepsilon}$	effective strain
$\dot{\bar{\varepsilon}}$	effective strain-rate
$\dot{\bar{\varepsilon}}_0$	limiting strain-rate
$\dot{\varepsilon}_v$	volumetric strain-rate
λ	Lagrange multiplier
ν	Poisson's ratio
$\{\sigma\}$	stress vector
$\bar{\sigma}$	effective stress
$\bar{\sigma}_0$	effective stress corresponding to $\dot{\bar{\varepsilon}}_0$
σ_t	transverse tensile stress
$\sigma_{t(\max)}$	maximum transverse tensile stress
$\sigma_{t(zero)}$	zero transverse tensile stress
σ_x	longitudinal stress
$\sigma_{x(avg)}$	average longitudinal stress
σ_y	transverse stress
τ_{xy}	shear stress

Note: All the components of computational time are measured in seconds.

CHAPTER 1

INTRODUCTION

1.1 GENERAL

In past few years, advancement in the field of computer technology resulted in faster and inexpensive computing. Now it is possible to analyze time taking problems within no time on conventional computers. But still in few fields where the computations are so excessive that the conventional computer sometimes takes days to complete the assigned task. With the help of parallel computing technique, now it is possible to solve such problems within reduced time frame on multiple processor computers or supercomputers. Parallel computing technique has shown a great success in the areas of computational atmospheric sciences, computational chemistry, computational fluid dynamics, evolutionary computing, computational structural mechanics, bioinformatics activities, seismic data processing, and many more. In this technique, the total computational job is distributed among several processors. Every processor operates simultaneously that results in saving in computational time without interfering the accuracy of the required results.

1.2 NEED FOR PARALLEL COMPUTING IN FINITE ELEMENT ANALYSIS

In past few decades, Finite Element Method (FEM) has grownup and emerged as one of the most efficient method for structural analysis. This method is very systematic to implement in form of computer code and therefore several commercial softwares are now available based on this approximate approach. Most of these softwares are written for a Personal Computer (PC) having single processor. In the finite element method, finer mesh generally gives relatively more accurate and detailed solution but as the mesh becomes finer, the problem size increases. This results in tremendous increase in number of computations, correspondingly computational time also increases. For non-linear finite element analysis, the scenario is even worse as it involves iterative solution procedure. There are few research papers available that show the implementation of parallel computing technique for analyzing such problems using finite element method on supercomputers. These papers discuss various approaches used for implementation of parallel computing technique on different types of supercomputers. These papers also discuss different techniques of reduction in computational time in Finite Element Analysis (FEA) by incorporating parallelism.

1.3 NEED FOR CLUSTER COMPUTING

Supercomputers are expensive as compared to the conventional computers. Therefore Cluster computing could be used as economical alternative to supercomputers. Cluster computing may be defined as group computing. Here, a group of conventional computers is formed, which is called as a Cluster. The computational task is distributed among these computers using parallel computing technique. This Cluster acts as a supercomputer and hence the distributed-computations save computational time. Very few research papers are available that show finite element analysis using Cluster.

1.4 OBJECTIVES

The main objective of the present study is to implement parallel computing technique in finite element applications. Although the area of parallel computing is old but its application in finite element analysis is not much reported, especially for non-linear finite element analysis. There are number of commercial softwares available that allow us to carry out structural analysis using finite element method but almost all these softwares operate on conventional computers. Therefore it is required to develop finite element analysis codes on supercomputers to obtain the solution of structural analysis problems in relatively lesser time. Cluster computing is also one of the emerging areas of computational analysis. Cluster can easily replace supercomputer because of its simplified implementation and lower cost. For effective implementation of parallel computing technique in finite element analysis, present research work is subdivided into following tasks.

1. Study of parallel computing technique.
2. Development of parallel solver to solve system of linear equations using parallel computing technique.
3. Implementation of the developed parallel solver in linear elastic finite element computer code to produce parallelized finite element software for linear elastic finite element analysis on platform of supercomputer PARAM 10000.
4. Development of a generalized finite element computer code to simulate non-linear plastic large deformation process on supercomputer PARAM 10000.

5. Development of a finite element code on supercomputer PARAM 10000 for simulation of compression process of metallic tubes of different cross sections between two flat rigid dies.
6. Development of Windows NT Cluster and implementation of parallelized linear elastic and non-linear plastic finite element computer codes on this developed Cluster.

1.5 SCOPE OF PRESENT WORK

Present work covers implementation of parallel computing technique for analyzing linear elastic as well as non-linear plastic problems on supercomputer PARAM 10000 and Windows NT Cluster in development of finite element computer codes. The emphasis is mainly given to non-linear finite element analysis categorized under the area of metal forming. The outcomes of the present research work are the two computer programs those are capable of analyzing elastic and plastic deformation in structural mechanics. With the use of these two programs one can analyze elastic and plastic two-dimensional axisymmetric and plane strain problems on supercomputer PARAM 10000 as well as Windows NT Cluster. The specialty of these programs is that they are developed using parallel computing technique, therefore one can analyze problems within reduced time frame.

1.6 REPORT ORGANIZATION

In the first chapter, introduction of work is presented. In the second chapter of this report, a brief discussion on referred literature is presented. It mainly includes the basics of parallel computing technique and hardware details of the supercomputer PARAM 10000. It also describes various parallel solvers developed by several researchers for solving system of linear equations. Further this chapter discusses anchorage zone in prestressed post tensioned concrete beam and presents extracts of few research papers presenting analysis of anchorage zone in prestressed post-tensioned concrete beam. Details of some commercial softwares and their limitations, which are also used in finite element analysis of tubular cross sections as energy absorption purpose, are also described. It also covers survey of available literature on use of metallic tubes as energy absorbing devices and their analytical, experimental and computational studies carried out by previous researchers.

In the third chapter, development of parallel solvers for solving system of linear equations is covered. Three parallel algorithms are developed using Gauss Seidel Method, Gauss Elimination Method and Matrix Inversion Method. Three solvers are developed on supercomputer PARAM 10000 based on these algorithms and their comparison is also presented. Based on the comparison, suitability of Matrix Inversion Method parallel solver was highlighted. Hence how it is further modified and redeveloped to analyze system of linear equations especially for finite element analysis is discussed.

Chapter four presents parallel implementation of finite element analysis code development for analysis of small deformation problems. It includes implementation of Matrix Inversion Method parallel solver for linear elastic finite element analysis. Initially, the basic formulation of linear elastic finite element analysis is presented in brief. Based on this formulation, parallelized code for two-dimensional plane stress, plane strain and axisymmetric linear elastic finite element analysis is developed and presented. A case study problem of anchorage zone in prestressed post-tensioned concrete beam is analyzed using the developed software. A study of effects of parameters like Poisson's ratio and eccentricity of prestressing forces on anchorage zone stresses is presented. An expression to compute the magnitude of bursting tensile force in anchorage zone is developed and compared with the expression available in literature and Indian Standard Code IS: 1343-1980.

Fifth chapter present the formulation adopted for the code development along with the two case study problems. Flow formulation in finite element analysis to solve metal forming problems is explained in the fifth chapter. Based on this formulation, a generalized computer program FEMLD is developed using parallel computing technique. Two case study problems are presented in detail based on the output obtained from the developed software. The same case study problems are analyzed using commercial softwares FORGE2 and ANSYS. To validate the results of developed software, various results in form of contours of velocity, strain-rate components, stress components, effective strain-rate, effective stress and effective strain are plotted and compared with the corresponding results obtained from commercial softwares. The performance of the developed code is also tested on supercomputer PARAM 10000 and described.

In chapter six, simulation of metallic tubes as energy absorbing element on supercomputer PARAM 10000 is presented. To analyze such problems using finite element method, code FEMLD4 is modified to deal with contact problems. Mainly two contact conditions are incorporated in code FEMLD4 by simulating two flat die surfaces. With the help of this modified code, four problems categorized under axisymmetric problems and plane strain problems are analyzed. These problems are axisymmetric compression of solid cylinder, lateral compression of rectangular metallic tubes, fold formation in axisymmetric compression of hollow round metallic tubes and lateral compression of round tube between two concentrated loads. The obtained results of these four problems are described in detail and also compared with the corresponding results obtained by using commercial software FOREGE2. Reduction in computational time with increasing number of processors of supercomputer PARAM 10000 was obtained. The performance of modified FEMLD code was also reported.

In the seventh chapter, Cluster computing technique is described. Two Windows NT Clusters are developed and presented in this chapter. The parallel solvers developed on the platform of supercomputer PARAM 10000 (described in chapter 3) are redeveloped and implemented on these developed Windows NT Clusters. The performance of these parallel solvers on Windows NT Cluster is measured and described in detail in this chapter. Based on the computational time results and the performance of the parallel solvers, comparison of parallel solvers are carried out and discussed in detail. Two computer codes for analysis of small and large deformation problems, which were developed earlier for supercomputer PARAM 10000, were redeveloped on these Clusters. One sample problem in each category was analyzed and variation in computational time components was obtained.

In the last chapter, summery of the present research work is presented. The advantages, limitations and the future scope of the present work are also described in this chapter.

In the third chapter, development of parallel solvers for solving system of linear equations is covered. Three parallel algorithms are developed using Gauss Seidel Method, Gauss Elimination Method and Matrix Inversion Method. Three solvers are developed on supercomputer PARAM 10000 based on these algorithms and their comparison is also presented. Based on the comparison, suitability of Matrix Inversion Method parallel solver was highlighted. Hence how it is further modified and redeveloped to analyze system of linear equations especially for finite element analysis is discussed.

Chapter four presents parallel implementation of finite element analysis code development for analysis of small deformation problems. It includes implementation of Matrix Inversion Method parallel solver for linear elastic finite element analysis. Initially, the basic formulation of linear elastic finite element analysis is presented in brief. Based on this formulation, parallelized code for two-dimensional plane stress, plane strain and axisymmetric linear elastic finite element analysis is developed and presented. A case study problem of anchorage zone in prestressed post-tensioned concrete beam is analyzed using the developed software. A study of effects of parameters like Poisson's ratio and eccentricity of prestressing forces on anchorage zone stresses is presented. An expression to compute the magnitude of bursting tensile force in anchorage zone is developed and compared with the expression available in literature and Indian Standard Code IS: 1343-1980.

Fifth chapter present the formulation adopted for the code development along with the two case study problems. Flow formulation in finite element analysis to solve metal forming problems is explained in the fifth chapter. Based on this formulation, a generalized computer program FEMLD is developed using parallel computing technique. Two case study problems are presented in detail based on the output obtained from the developed software. The same case study problems are analyzed using commercial softwares FORGE2 and ANSYS. To validate the results of developed software, various results in form of contours of velocity, strain-rate components, stress components, effective strain-rate, effective stress and effective strain are plotted and compared with the corresponding results obtained from commercial softwares. The performance of the developed code is also tested on supercomputer PARAM 10000 and described.

CHAPTER 2

LITERATURE REVIEW

2.1 INTRODUCTION

This chapter covers review of the literature referred in the attempted study. Since present work deals with the application of parallel computing techniques in linear as well as non-linear finite element analysis, so literature review presented in different sections highlights the different studies covered. In the first part, introduction to parallel computing technique is covered. The present work is carried out on supercomputer PARAM 10000, hence its hardware is also discussed in detail. It is followed by the review of few research papers covering the development of parallel solvers for finite element analysis. Further, review of research papers showing analysis of anchorage zone in prestressed post-tensioned concrete beam is also presented since it is considered as a case study problem in linear elastic finite element application. For non-linear finite element application, few problems related to analysis of metallic tubes as energy absorbing devices are considered, hence related literature is also discussed in this chapter.

2.2 INTRODUCTION TO PARALLEL COMPUTING

Solving many scientific problems requires high speed computing, which is difficult to achieve by single processor computer. Hence parallel computing is introduced to achieve high speed in solving various scientific problems [1, 2]. Parallel computing mainly require computer having multiple processors called supercomputer. The processors are organized in different patterns namely mesh networks, binary tree networks, hypertree networks, pyramid networks, butterfly networks, hypercube networks, cube-connected cycles networks, shuffle-exchange networks and de Bruijn networks. Parallel computer architectures are classified as per Flynn's taxonomy scheme, which is based on duel concepts of instruction stream and data stream. The four classes of parallel computers based on multiplicity of data and instruction streams are as follows

- Single Instruction stream, Single Data stream (SISD)
- Single Instruction stream, Multiple Data stream (SIMD)
- Multiple Instruction stream, Single Data stream (MISD)
- Multiple Instruction stream, Multiple Data stream (MIMD)

The computer programs executable on multiple processors computer are based on Parallel Random Access Machine (PRAM) algorithms. PRAM consists of a control unit, global memory and a set of processors with its own memory. PRAM has following models.

- EREW (Exclusive Read Exclusive Write)
- CREW (Concurrent Read Exclusive Write)
- CRCW (Concurrent Read Concurrent Write)

With the help of Message Passing Interface (MPI) [3. 4] and programming languages like C [5], C++ [6], FORTRAN [7] and recently introduced Java, parallel programs are written. MPI provides wide range of inbuilt functions that enable user efficient data transfer among the processors. MPI also provides standard timer functions that are used in measurement of computational time for particular code segments. The performance of the parallel codes is measured by computing Speedup, Efficiency and MFLOPS.

2.3 PARAM 10000 ARCHITECTURE OVERVIEW

PARAM 10000 has a MIMD distributed memory machine architecture, developed by Center for Development of Advanced Computing (C-DAC) (see Fig. 2.1). The hardware configuration of supercomputer PARAM 10000 consists of following major components [8].

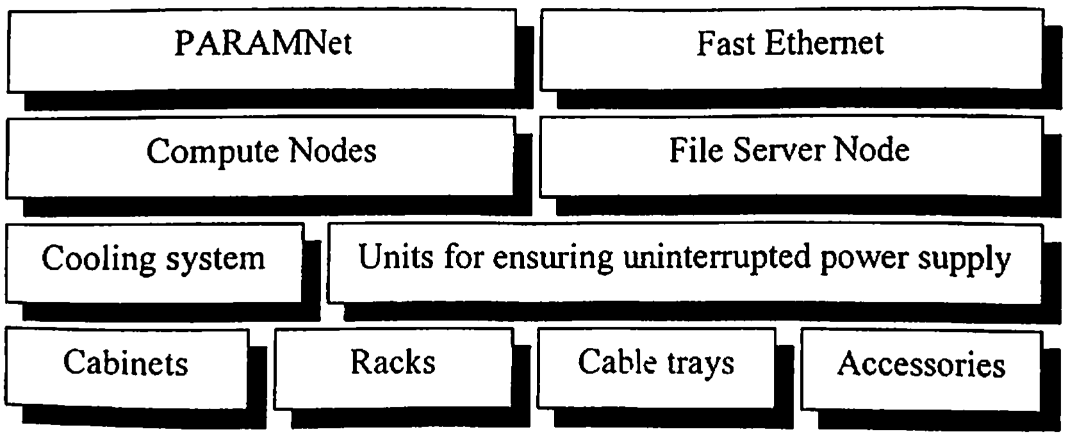


Fig. 2.1 Major hardware components of the PARAM 10000 system

2.3.1 Nodes

Supercomputer PARAM 10000 has altogether four nodes out of which three nodes are categorized as compute nodes and the fourth one is categorized as a file server node. The roles and the configurations of these two categories of nodes are given below.

2.3.1.1 Compute Nodes

These nodes are mainly used for computation. Each of these nodes has the following configuration:

- Two UltraSparc II 64-bit RISC CPUs of 400 MHz each, with 2 MB external cache per CPU
- 512 MB main memory expandable to 2 GB
- Two Ultra SCSI HDD of 9.1 GB each
- One internal 32x CD-ROM drive
- One 1.44 MB floppy disk drive
- One PGx 32 graphics card
- Two 360 W hot swap power supplies
- One PARAMNet CCP2 Card
- One 10/100 Fast Ethernet Card
- Solaris 2.6 (upgradable to Solaris 7) with Server License (unlimited user license)

2.3.1.2 File Server Node

This node is used as the main storage server for all the programs and data of the users of the system. That is, it acts as a host to the compute nodes for the I/O requirements of the users. The file server node has the following configuration:

- Two UltraSparc II 64-bit RISC CPUs of 400 MHz each, with 2 MB external cache per CPU
- 1 GB main memory expandable to 2 GB
- Four Ultra SCSI HDD of 9.1 GB each

- One internal 32x CD-ROM drive
- One 1.44 MB floppy disk drive
- One PGx 32 graphics card
- One internal 4 mm 12/24 GB Tape Drive
- Two 360 W hot swap power supplies
- One PARAMNet CCP2 card
- One 10/100 Fast Eathernet card
- One 21" Color Monitor
- Solaris 2.6 (upgradable to Solaris 7) with Server License (unlimited user license)

2.3.2 Interconnection Networks

PARAM 10000 has the two interconnection networks namely PARAMNnet and Fast Ethernet.

2.3.2.1 PARAMNet

C-DAC's PARAMNet is conceived as a high-speed switched network for Cluster computing. PARAMNet is based on the technology of packet communication and switching with PARAM 10000. The flexibility of PARAM architecture enables user to unbundle the communication network to form a geographically distributed high-speed LAN. PARAMNet LAN is centered around C-DAC's PARAMNet Switch. This is an 8-port bi-directional switch with 4 links and aggregate throughput of 400 Mbps. This network is based on wormhole packet switch from INMOS and can be used as general purpose network, as well as high speed, low latency network using light weight protocols.

2.3.2.2 Fast Ethernet

Ethernet provides well-established solution for a high-speed reliable communication network to interconnect powerful nodes/workstations for building PARAM 10000. This 100 Mbps full duplex network is mainly used as an administrative network. All the standard services such as NFS, NIS+, etc. are served over this network. This is a standby network.

2.4 FINITE ELEMENT METHOD

Finite element method is one of the efficient tools for numeric solution of several engineering problems. In the past few decades this method has developed enough and now various complex problems can be easily analyzed using finite element method [9-11]. In finite element method, problem domain is discretized into simple geometric shapes called as elements. To achieve detailed and more accurate solution of the desired problem, more numbers of elements are normally preferred. It is quite natural that as the number of elements increases the problem size also increases. Hence this method is generally implemented in form of computer codes and they are widely used for solving complex problems. Literature [12] shows that some problems are so large that their solution on conventional computer becomes very time consuming. According to literature, major portion of the time in computational finite element analysis is consumed in the process of solving system of linear equations generated in finite element solution procedure. Hence parallel solvers are developed and employed in finite element codes.

2.5 PARALLEL SOLVERS

The main aim of using parallel solver is to trim down the computational time in the finite element analysis. The computational time in any computer code can be measured using timers [13]. Wadleigh and Crawford [14] discussed the computational time measurement using timers. Authors also discussed different types of timers. Depending on their properties, like accuracy, overheads, resets, etc., the suitability of these timers was also presented. Authors also discussed the different components of computational time (Real time, User time, CPU time, and System time). Authors also gave few examples explaining the factors those affect these components of computational time.

There are various methods available to solve system of linear equations [15, 16] but very few of them are used in development of parallel solvers. Thiagarajan and Aravamuthan [17] had discussed the implementation of High-performance FORTRAN on 32-node Pentium II 350 MHz Linux Clusters. Authors used two different parallelization strategies on preconditioned conjugate gradient solver for linear elastic finite element analysis. Authors discussed various components of computational time like CPU time, Real time, Communication and Synchronization time (C&S). Authors showed that, variation of Real time with increasing number of processors is not smooth. Authors found that initially Real time starts reducing with increase in number of processors, but after certain number of processors, the Real time starts increasing. The number of processors at which the measured Real time is minimum was called as optimal number of processors. Authors concluded that this optimal number of processors depends on the problem size. After comparing CPU and C&S time, authors concluded that there is even balanced in the time spent in CPU and C&S time at optimal number of processors. Authors also discussed the performance of developed code by measuring the Speedup based on CPU time and Real time. Authors found that the CPU time Speedup constantly increases and shows linear variation that keeps up with ideal Speedup, whereas the Real time Speedup increases curvilinearly. Authors also measured performance of their code by calculating the MFLOPS based on CPU time. Authors found that MFLOPS increases with increase in number of processors.

Khan and Topping [18] presented a modified parallel Jacobi-conditioned conjugate gradient method for solution of linear elastic finite element system of equations. Authors discussed and implemented element-by-element and diagonally conditioned approaches on distributed memory MIMD architectures. Authors analyzed two finite element domains discretized using constant strain triangular elements (CST) resulted in 934 and 1294 degrees of freedom. Authors solved these problems over a pipeline of 14 transputers by changing the number of processors from 3 to 14 and the obtained time variation was presented. Authors also presented results of various components of computational time like parallel computational time (t_{parr}), average time spent by each processor for performing calculation (t_{cal}), average time spent by each processor for performing communication (t_{comm}) and the ratio of t_{cal} / t_{comm}. Authors found that t_{calc} reduces with increase in number of processors and t_{comm} increases with increase in number of processors. Authors also found that ratio t_{calc} / t_{comm} reduces with increase in number of processors. Authors also measured the Speedup and Efficiency of the code and found that Speedup increases and Efficiency decreases with the increase in number of processors. Authors also plotted a curve between ratio t_{calc} / t_{comm} and Efficiency and found that the ratio t_{calc} / t_{comm} increases with increase in Efficiency.

Mahinthakumar and Saied [19] presented a hybrid MPI-OpenMP model of an implicit finite element application using FORTRAN as programming language. Authors also compared this model with pure MPI, OpenMP model on four parallel architectures. Authors found that execution time reduces with increase in number of processors for all four parallel architectures. Authors also measured MFLOPS for all four cases and found that MFLOPS increases with increase in number of processors for pure MPI model as well as hybrid MPI-OpenMP model.

Danielson and Namburu [20] presented non-linear dynamic finite element analysis on three supercomputers namely CRAY T3E, IBM SP, and SGI ORIGIN 2000. Authors used FORTRAN90-MPI combination for code development. Authors used non-blocking communication mechanism for data transfer among the processors. Authors carried out their analysis on few to hundreds of processors and found that CPU time reduces with increase in number of processors for all three supercomputers. Authors achieved good Speedup (close to Ideal Speedup) for all three supercomputers.

Sziveri and Topping [21] presented finite element analysis of transient dynamic problems using MPI on MIMD computer architectures. Authors adopted C programming language and non-blocking communication mechanism to carry out their analysis on different machines running on SUN's Solaris operating system with the 10 Mbps connection network. Authors discussed how the user activities affect the computational time. In this article, authors presented the results based on Real time. To minimize the effect of user activities, authors carried out their analysis when there were fewer user activities. In spite of taking all the possible measures, authors got the discrepancy in the computational time results. Therefore repeatedly measurements were taken and the extreme results were omitted. The averages of sensible computational time results were taken for the further course of action. Authors further discussed that disturbance created by users could be prevented by booting the machine into a different state. In such machine state, the general system activities could be restricted as well as network and user processes also totally excluded.

King and Sonnad [22] presented a element-by-element approach along with preconditioned conjugate gradient solver for solving system of equations arising in finite element analysis. Authors used loosely coupled array of processors (LCAP) parallel computer. Authors found 90% efficiency for 20 processors. Authors also found that as the number of processors increases, the efficiency of the solver decreases. Authors also discussed the advantage of element-by-element approach over the direct solution algorithm on sequential computer.

Chu et al. [23] presented parallel matrix inversion on hypercube multiprocessors. Authors employed Gauss-Jordan inversion technique with column exchanges. Authors found that the computational time reduces with increase in the number of processors. Authors used double precision FORTRAN77 language for programming and analyzed few data sets on 8 x 8 sub cube grid consist of 64 Intel iPSC/860 machines having 40 MHz speed. Authors also compared sequential Gauss elimination technique with sequential Gauss-Jordan technique.

2.6 FINITE ELEMENT ANALYSIS ON PARAM 10000

In the area of finite element analysis, very little work has been carried out using supercomputer PARAM 10000. Most of the work includes static linear elastic finite element analysis only.

Kant and his associates [24-26] presented finite element analysis of composite materials using supercomputer PARAM 10000. Authors used parallel Cholesky solver to determine the solution of linear equations using Master-Slave approach. Ramesh and Shah [27] developed a parallel solver using a Preconditioned Conjugate Gradient technique for finite element analysis. Authors also used Master-Slave approach. Kant et al. [28] developed a parallel solver using Conjugate Gradient technique and the results obtained were compared with the results obtained by parallel Cholesky solver. Authors found that the Cholesky solver is faster than Conjugate Gradient solver in context of computational time. According to Kant et al. [28], Conjugate Gradient method, which is an iterative technique, was found to be more useful for the large size problems. Shah and Ramesh [29] presented fracture analysis software FRACT2D to determine the Stress Intensity Factor (SIF) of the cracked structures developed using finite element method on parallel supercomputer PARAM having MIMD architecture that uses transputers T805 as a processor. Authors used Cholesky method for analysis of linear equations generated in finite element analysis. Authors found that computational time reduces with increase in number of transputers employed for the analysis.

Rao [30] presented MPI based non-linear implicit transient dynamic analysis on supercomputer PARAM 10000. Combination of three formulations for domain decomposition in finite element method and linear preconditioned conjugate gradient technique was adopted to solve large-scale problems in structural mechanics. Rao et al. [31] developed Software for PArallel Non-linear Dynamic ANalysis (SPANDAN) on supercomputer PARAM 10000 using MPI. Authors used parallel overlapped domain decomposition approach in their code and compared with conventional non-overlapped domain decomposition approach and found that their algorithm is superior.

Most of the parallel solvers mentioned above were developed for direct method of solution of system of linear equations and using FORTRAN77 as programming language.

2.7 PRESTRESSED CONCRETE

In prestressed concrete members, stresses are induced during the construction in such a way that they can resist stresses caused by externally applied loads. Prestressed concrete structural members are widely used to achieve high strength at lower self-weight. Prestressed concrete is most suitable for long span structural elements like beams and girders, where larger bending moment results in greater depth of beam or girder [32, 33]. Broadly there are two methods of prestressing namely Pre-tensioning and Post-tensioning. In Pre-tensioning, the prestressing tendons are tensioned before the concrete is placed while in Post-tensioning hardened concrete is stressed by applying external forces.

In the post-tensioned concrete beams, a duct is formed inside the beam and prestressing cable is kept inside this duct. Once the concrete gets harden, prestressing cable is stressed and anchored at the end of beam that induces internal stresses in the concrete beam. The stress distribution inside the post-tensioned concrete beam is very complex, especially near the end of beam where prestressing cable is anchored. This zone is called as Anchorage Zone [32].

In the past, few researchers attempted to analyze stress distribution in anchorage zone in post tensioned concrete beam using different techniques, which include analytical techniques [34-37], experimental methods [38-41] and numerical methods [42, 43].

Guyon [34] analyzed the anchorage zone using elasticity equations assuming the beam as end-loaded-semi-infinite strip as two-dimensional problem. The length of anchorage zone was considered equal to the depth of the beam. Som and Ghosh [35] made a similar attempt by treating it as a two-dimensional plane stress boundary value problem. Authors used the Airy's function for their analysis and the stress function was expressed in the form of Fourier series. The obtained results were quite similar to the findings of the Guyon [34]. Iyengar and his other associates [36, 37] analyzed the problem of anchorage zone using the equations of elasticity considering problem as two-dimensional and three-dimensional. Authors carried out the analysis for concentric as well as eccentric prestressing forces and compared the results with the available literature.

Some researchers also carried out experimental investigation of the anchorage zone. Christodoulides [38, 39] conducted actual tests on the concrete block along with two-dimensional and three-dimensional experimental studies using photoelastic bench. Zielinski and Rowe [40] presented results of surface strains measured on the concrete end block subjected to concentric prestressing forces. On the basis of their results, authors gave an expression to calculate the magnitude of the bursting tensile force (F_{bst}) for different values of k (ratio of loaded area and cross-sectional area of the beam). The modified version of this expression, by introducing factor of safety, was adopted in the Indian Standard Code IS: 1343-1980 [41]. The effect of Poisson's ratio (ν) and eccentricity (e) of prestressing forces (P_k) over F_{bst} was not included in the given expression.

Yettram and Robbins [42] investigated anchorage zone stresses considering it as a three-dimensional problem. Authors used finite element analysis to determine the anchorage zone stresses. Their investigation did not prove the occurrence of spalling zone. Recently, Byung-Wan Jo et al. [43] investigated the anchorage zone stresses by considering effects of various parameters namely cable inclination, position of anchor plate, and the modeling methods. Authors also carried out their analysis using finite element method considering the problem as two-dimensional as well as three-dimensional and found that the three-dimensional analysis gives slightly smaller values of stresses as compared to their two-dimensional analysis. Authors suggested to adopt the results of two-dimensional analysis to ensure the safety in the design.

2.8 ENERGY ABSORBING DEVICES

Energy absorbers have an important role to play in the improvement of aircraft crashworthiness. In crashes of light fixed wing and rotary wing aircraft and transport aircraft impact, the loads are usually low enough to be survivable by the occupants if some measures are taken to provide improved protection through the use of energy absorbing devices. Areas in which these devices may be applied include the landing gear, the bottom of the fuselage, the seats and the mountings for massive structures such as helicopter transmissions [44, 45].

The important characteristics of energy absorber are the specific energy absorption capacity per unit weight of device or system, the efficiency of the stroke, the stroke to length ratio, the reliability, the repeatability, the ability to sustain rebound loads, and the cost. In specific application it is desirable to optimize the design in the sense that some desired combination of low cost, low weight, small size and high performance is achieved. The understanding of the characteristics of a particular energy absorbing device is required to accomplish this. Energy absorbing devices are classified into three general categories according to their primary mechanism used for the absorption of energy. These are material deformation, extrusion and friction. These classifications made on the basis of the primary energy absorbing mechanism. In many devices, there are more than one energy absorbing mechanism operative but, in general, one is dominant.

Deforming tubes are deformable elements, which lend themselves to a wide variety of uses as energy absorber. The tubes can be flattened, made to turn inside out, made to expand, made to contract, made to change in cross-sectional shape, made to fold or spilt and curl up. The mechanism of energy absorption is plastic deformation of tubes, therefore to understand their deformation process both geometric and material nonlinearities should be considered. Experimental and computational study of deformation of metallic tubes was carried out in detail especially by Gupta and his associates in last few years [47-58] and by other researchers [59-62].

An investigation into the energy absorbing characteristics of the metallic circular, square and rectangular cross-section hollow tubes of aluminium and mild steel was carried out by Gupta [46]. Gupta carried out experiments to study the large deformation process of several tube specimens oriented and loaded in axial as well as lateral directions. The deformation process was also analyzed using commercial software FORGE2 to study their collapse mechanisms. Various results obtained from computational study were compared with the corresponding results obtained from experimental study.

Study of collapse of rectangular and square metallic tubes between flat platen was carried out by Gupta et al. [47-49]. Authors used aluminum as well as mild steel tubes to study their energy absorbing capacity. Compression process of several tubular cross sections was analyzed experimentally as well as computationally. Experimental results obtained were compared with their computational counterparts obtained using commercial software FORGE2. A finite element model was presented and collapse mechanism of these tubes was studied. Effects of parameters like friction factor, wall thickness and shape of tube on collapse mechanism was also studied.

A detailed study on lateral compression of aluminium and mild steel round tubes between two rigid flat platens was presented by Gupta et al. [50-52]. Experiments were carried out on several tubular specimens with different diameter to thickness ratios and deformation histories along with load-compression curves were presented. A finite element model for computational study was proposed and used in computational study carried out using commercial software FORGE2. Experimental and computational results were compared and discussed. Deformation mechanism of round tubes and effects of process parameters on deformation mechanism was also presented and discussed.

A study on fold formation in axisymmetric deformation of round tubes was presented by Gupta et al. [53-55]. Experimental and computational study was carried out on aluminium and mild steel tubular specimens with different diameter to thickness ratios. The process of fold formation was studied and described with the help of computational model analyzed using commercial software FORGE2. Various results like history of deformation, load-compression curves, energy absorbing capacity and other were presented in details. Effects of process parameters on process of fold formation were also presented. Recently Gupta and Nagesh [56] presented experimental and numerical studies of collapse of thin aluminum tubes having circular cross section under axial compression. Author used commercial software ANSYS for their study. Authors carried out parametric study for analyzing collapse mechanism by changing various geometrical properties and material properties.

Sekhon et al. [57, 58] presented a study on external inversion of round tubes over a circular die. Three aluminium specimens were studied experimentally as well as computationally. Process of tube inversion was presented in details with the help of deformed shapes and contours of nodal velocity, strain-rate and strain at various stages of deformation. Load-compression curves and energy-compression curves were also presented and discussed in details. The effect of friction between die and tube interface on energy absorbing capacity was also discussed and presented.

Sun and Yang [59] presented a finite element code for simulation of inversion of thin tubes subjected to axial loading. Authors considered material as rigid plastic and flow formulation was adopted for the code development. Authors also carried out experiments on two specimens to obtain the deformed shapes and load-compression relationship. The experimental load-compression relationship was compared with the corresponding load compression relationship obtained from the code.

Reid [60] presented deformation mechanism of different circular and square metal tubes used as energy absorbers. Progressive buckling, tube inversion and splitting phenomena in axial compression of circular tubes are also discussed in this literature.

Guillow et al. [61] presented axial compression of thin walled aluminium tubes. Authors carried out experiments on 70 tubular specimens with different diameter to thickness ratios. Authors studied collapse modes for all specimens and found that axisymmetric and nonsymmetric modes lie on a single curve. The authors also discussed the effect of density of polyurethane foam filling in aluminium tubes on crushing force.

Hosseinipour and Daneshi [62] presented analysis of axial compression of thin walled steel grooved tubes in context of energy absorption and mean crushing load. Authors performed experiments and obtained load displacement curves. Authors also presented theoretical formulations for predicting the energy absorbing capacity and the mean crushing load. Authors found good agreement between theoretical results and experimental findings.

2.9 FEM IN LARGE DEFORMATIONS

Large deformations occur at many places and have wide area of applications. It mainly covers metal forming, crashworthiness analysis, industrial manufacturing, impact analysis, etc. There are several mathematical methods available to analyze such processes. They include the slab method, the slip-line field method, the viscoplasticity method, upper bound and lower bound technique, Hill's general method, and the most recent method is the finite element method [63]. All these methods are approximate methods of analysis and still research is going on to develop a more accurate method to analyze such a complex phenomena.

Analysis of large deformation process is a complex task. It is mainly because the strains developed are large in magnitude and the corresponding stresses are in the plastic region and both are dependent upon many process parameters. Finite element method is quite effective to carry out general structural analysis as well as the large deformation analysis. This method can be easily implemented on computers, which is one of the major advantage of this method. In the application of FEM to large deformation process, there are two formulations available, namely flow formulation and solid formulation. Flow formulation in FEM is widely used in analyzing the large deformation processes. The solution procedure is stepwise. The entire deformation is divided into sub-steps of certain step size. In each step, the geometry of the problem domain changes. Therefore such type of analysis becomes very complex. Numerous calculations are involved in obtaining the solutions of each sub-step. Therefore, the entire solution process also becomes very time consuming.

Lee and Kobayashi [64] presented matrix method for analysis of rigid plastic large deformation problems. Using this method Authors also solved two problems of simple compression of cylinders as well as bore expanding and flange drawing categorized under axisymmetric problem and plane strain problem respectively. Lee and Kobayashi [65] presented detailed studies of the deformation of a solid cylinder. The cylinder was axially compressed till 33% reduction in height was achieved. Authors used finite element method for their analysis. Various obtained results having load displacement curves and contours of strains and stresses were presented. Authors also presented compression of cylindrical specimen between flat parallel dies categorized under plane strain conditions.

Kobayashi [66] presented experimental study on compression of steel solid cylinders and rings. The relationship between friction at the interface, deformation characteristics and fracture of specimen was discussed. The deformation histories of the cylinder and ring specimens were also presented.

Petersen et al. [67] presented friction in bulk metal forming by comparing general friction model and constant friction law. Authors carried out experimental study of upsetting of a semi-tapered specimen and ring compression test to validate their numerical investigation. Axisymmetric large deformation behavior of aluminium and low carbon steel short cylinder of various diameters and heights was experimentally carried out by Gupta and Shah [68]. Authors studied histories of deformation of cylindrical surface and found that the profile of a deforming specimen can be approximated by an arc of a circle.

The finite element formulation given by Kobayashi et al. [63-65] could be used for analyzing two-dimensional as well as three-dimensional large deformation problems. It also allows us to carry out the thermo-viscoplastic analysis. Authors also developed a computer program Simple Plastic Incremental Deformation (SPID) for analyzing the two-dimensional plane strain and axisymmetric problems. SPID was written in FORTRAN77 language and capable of handling a finite element model with maximum 100 nodes. This program is executable on any conventional computer.

There are some other softwares existing that can be used for carrying out analysis of large deformation process. FORM2D is also one of the software capable of analyzing the large deformation problems. This software is developed by Singh [69] and is based on the formulation given by Kobayashi et al. [63]. In addition to these, some commercial software are also available, namely FORGE2 [70], ANSYS [71] and others. These commercial softwares operate on different Operating Systems (OS) compatible on conventional computer.

2.10 LARGE DEFORMATIONS AND PARALLEL COMPUTING

Very little work has been done in the area of application of parallel computing technique to analyze large deformation process. Kim and Im [72] and Cheon et al. [73] presented modified block Jacobi preconditioning technique for analyzing three-dimensional metal forming problems. Authors also used domain decomposition approach in their solver. Authors carried out their analysis on supercomputer CRAY T3E 900. Authors compared their computational time results with the results obtained from conventional conjugate gradient method and Jacobi preconditioned conjugate gradient method. Authors found that the modified block Jacobi preconditioning technique is more efficient than the other two techniques. Authors also achieved reduced number of Newton-Raphson iterations during solution convergence when modified block Jacobi preconditioning technique was used as compared to the other two methods. Authors also presented a numerical application by analyzing a problem of simple upsetting of cube type work piece of aluminum AL6061-T4 where material behavior was expressed by expression $\bar{\sigma} = 35.16\bar{\varepsilon}^{0.08}$ MPa. The work piece was compressed with speed of 1.0 mm/s and the friction factor was considered as 0.1. In their finite element computer code, the material behavior was considered as rigid-viscoplastic and finite element formulation presented by Kobayashi et al. [52] was used.

2.11 CLUSTER COMPUTING TECHNIQUE

In past few years Clusters have been emerged as affordable platform for high performance computing. Advantages of Clusters over the expensive supercomputers are discussed in details by Sterling [74]. Author discussed the hardware configuration of a Cluster node, which includes processor, memory, secondary storage and external interface. Author also discussed Cluster network hardware in form on Local Area Network (LAN) and System Area Network (SAN). Application programming environment and software components are also discussed in details. Application of Linux Cluster for direct numerical simulation of fluid turbulence code was presented by Chun et al. [75]. Authors used Cluster of 64 PC's of 2.8 GHz processors and using this setup 40% reduction in CPU time was achieved by doubling the number of processors.

Cheon et al. [76] developed a PC Cluster (1.6 GHz and 1.0GHz) using LAN (100 Mbps) each having Linux 2.1 operating system. Authors carried out the same analysis (discussed above) on this Cluster. Authors used parallel LDU factorization as well as domain decomposition approach during the analysis. Authors found that the computational time reduces with the increase in number of PC's employed for the analysis as well as with the increase in the number of sub-domains used to decompose the whole problem domain. Authors presented same numerical application as presented in literature [72, 73]

2.12 SUMMARY

In area of structural analysis using finite element method. use of parallel computing technique is recent and literature shows that parallel solver is needed for effective implementation of parallel computing technique. From literature it is also observed that there is not much work reported related to application of parallel computing technique in FEA. Some research papers are available that show application of parallel computing technique in linear elastic finite element analysis. Few research papers are also available that show non-linear and/or dynamic analysis on multiple processor computers. In case of non-linear finite element analysis, particularly in large deformation problems, use of parallel computing is scarce. Hence emphasis should be given in development of generalized finite element code for analyzing large deformation problems on supercomputers. In order to accomplish this, it is required to develop an efficient parallel solver that can be used in linear as well as non-linear structural analysis using FEM on supercomputer. Cluster computing is also one of the emerging areas in the field of computer sciences. Significant work is needed in this area so that Cluster can become an efficient and inexpensive alternative to the supercomputers.

CHAPTER 3

DEVELOPMENT OF PARALLEL SOLVERS FOR SOLVING SYSTEM OF LINEAR EQUATIONS

3.1 INTRODUCTION

This chapter mainly deals with the development of parallel solver for finite element analysis to achieve reduction in computational time during the analysis. The chapter starts with the definitions of the different components of the computational time along with the developed timer used to measure these. Certain terms, which indicate the performance of parallel codes, are also summarized in brief. The chapter further discusses three techniques namely Gauss-Seidel method, Gauss Elimination method, and Matrix Inversion method used for development of parallel solvers. It also highlights the variation in computational time and performance of these methods on supercomputer PARAM 10000. A brief study on two communication mechanisms namely Blocking and Non-Blocking is also presented and discussed. The chapter also presents the comparison of C and FORTRAN77 programming languages based on their performance for application in parallel solver development. At the end of the chapter, a modified efficient parallel solver is introduced and presented.

3.2 COMPONENTS OF COMPUTATIONAL TIME

The main aim of using parallel computing technique is to save computational time. Therefore it is essential to study different components of computational time. There are different components of computational time [13, 14], which are used in the present investigation. The definitions and the possible factors that affect these components are discussed in the following text.

Real time **(RT)**: Real time can be defined as the wall clock elapsed time for a particular process. This time differs on multi-user systems where several programs may be running concurrently. Therefore if one executes some program for some number of data for several times, then every time the Real time will be different.

User Time **(UT)**: User time is the time spend by the program in executing itself (can be measured only in UNIX operating system). This time will not change with other system activities or user activities. Negligible change may be observed for program running when processor is heavily loaded or the program is dealing with huge data.

Total Time **(Total)**: It is the time that is necessary to execute overall process on machine with multiple processors. In other words, it is time measured from the invocation of command till its termination. For parallel processes, it is sum of Communication time and the Calculation time. This component can be measured in terms of Real time (RT) as well as User time (UT).

Communication Time **(Comm)**: Communication time may be defined as the time required to transfer the data from one processor to the other processor or processors. The processor mapping mainly affects the Communication time. This time component also depends on the data transfer rate between the processors and the machine type. This component can be measured in terms of Real time as well as User time.

Calculation Time **(Cal)**: Calculation time may be defined as the time required for the processor in performing the calculation exclusively. In other word it may be defined as the difference of Total time and Communication time. This time component is purely dependent on the data size to be handled by processors. This component can be measured in terms of Real time as well as User time.

System Time **(Sys)**: System time may be defined as the time used by the system in doing work on behalf of the user (can be measured only in UNIX operating system). In other words, when user gives command for any execution process, some amount of time is spent by the operating system in supporting that execution process which is called as System time.

CPU Time **(CPU)**: It is the summation of User time and System time (can be measured only in UNIX operating system). It is not necessary that CPU time is equal to Real time, since little time is also consumed by processor in doing other works in the system or assigned by different users connected to it.

The timers those are available in standard library return only Real time. To measure User time, it is required to develop timer that returns the User time. Following example in C language presents a subroutine of a timer and shows that how it is used to measure the time involved in certain code segment. With the help of this timer, one can also measure different components of time during the execution process.

Example:

```
#include<time.h>
#include<sys/times.h>
double zero = 0.0, t0, t1;
double timer(double t);
main()
{
 t0 = timer(zero);
          < code segment being timed >
 t1 = timer(t0);
 printf(" The User Time = %f\n", (t1 - t0) );
}
double timer(double t)
{
 double time_user, time_sys, time_cpu;
 static double recip_cpu, recip_sys;
 struct tms buffer;
 static long base_sec_cpu = 0, base_sec_sys = 0;
 (void) times(&buffer);
 if ( base_sec_cpu == 0 )
          {
          recip_cpu = 1.0 / (double) CLK_TCK;
          base_sec_cpu = buffer.tms_utime + buffer.tms_stime;
          }
 if ( base_sec_sys == 0 )
          {
          recip_sys = 1.0 / (double) CLK_TCK;
          base_sec_sys = buffer.tms_stime;
          }
 time_sys = ((double)(buffer.tms_stime - base_sec_sys)) * recip_sys - t;
 time_cpu = ((double)(buffer.tms_utime + buffer.tms_stime - base_sec_cpu)) * recip_cpu - t;
 time_user = time_cpu - time_sys;
 return(time_user);
}
```

The above discussed subroutine will return only the User time. It basically measures the CPU time (time_cpu) and the System time (time_sys), then the User time (time_user) is calculated by subtracting the System time from CPU time. This is done only to demonstrate that the other time components like System time or CPU time can also be measured by using the same subroutine, which could be done by changing few lines of it. The wall clock elapsed time (Real time) can be measured in similar way by using the readymade subroutines MPI_Wtime available in standard library of MPI. There are some other subroutines, like clock and time available in standard C ++ library, which can also be used for the same purpose, but these standard subroutines will give only the wall clock elapsed time or Real time.

There are few terms that are used for measuring the performance of the parallel programs [1, 2]. The definitions of these are as follows

Speedup **(S)**: Speedup may be defined as the ratio between the time needed for the most efficient sequential algorithm to perform a computation (t_s) and the time needed to perform the same computation on a machine incorporating parallelism (t_p).

$$S = \frac{t_s}{t_p}$$

Speedup can be categorized under three categories namely Linear Speedup, Ideal Speedup and Superlinear Speedup.

Efficiency (η): Efficiency can be defined as the percentage ratio of time needed for sequential algorithm to the product of number of processors *P* and time needed for *P* number of processors for parallel algorithm.

$$\eta = \frac{t_s}{t_p \times P} \times 100$$

MFLOPS: It is the number of floating points operations per second. This has been evaluated by calculating the number of floating point operations performed and dividing it by time required to carry out these operations.

All these components involve a term time, which can be Real time or User time.

3.3 NECESSITY OF PARALLEL COMPUTING

To employ the parallel computing technique efficiently in finite element codes, it is very much essential to study the time elapsed in different processes in a typical FEA. Typical computer aided FEA can be divided into five distinct processes.

1. Reading input files.

2. Memory allocations for various variables involved in analysis.
3. Generation of stiffness equation.
4. Solution of stiffness equation.
5. Post processing calculations involving computations of strains, stresses and others.

A generalized finite element code was written and the computational time elapsed in the above mentioned processes was measured by solving a typical problem. Figure 3.1 shows a time distribution obtained in various processes involved in FEA. It was found that major portion of the computational time gets consumed in the process of solving stiffness equation. The third process consumes nearly 1% of Total time, whereas the forth process takes almost 99% of the Total time. The remaining processes have insignificant contributions toward the overall computational time. It clearly expresses the need of parallel solver to get quicker solution of stiffness equation.

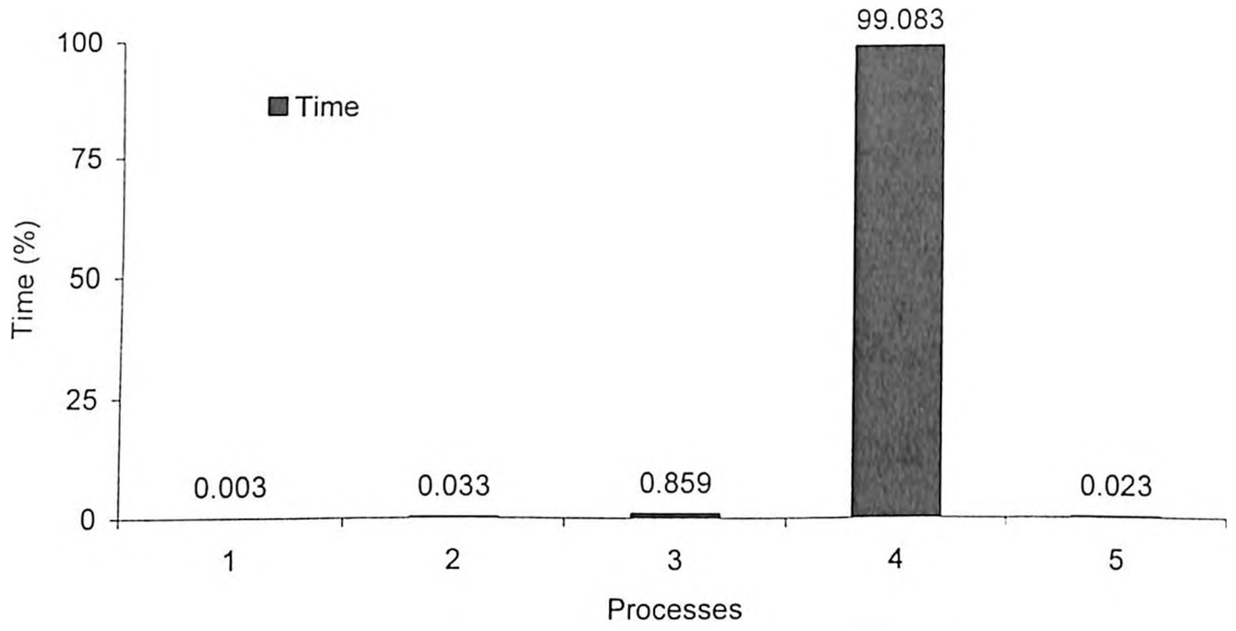


Fig. 3.1 Computation time expenditure by different processes involved in FEA

3.4 DEVELOPMENT OF PARALLEL SOLVERS

From Fig. 3.1 it is clear and advisable to use parallel computing technique in solving the stiffness equation. Generally the stiffness equation is expressed as a system of linear equations $[A]\{X\} = \{B\}$. Matrix $[A]$ is nothing but the global stiffness matrix, whereas vectors $\{X\}$ and $\{B\}$ represent global displacement matrix and global force matrix respectively. There are several methods available to solve such system of linear equations. These methods are categorized under two categories namely Direct methods and Iterative methods. In Direct methods, the number of computations are always known and fixed, whereas in Iterative methods, the number of computations are unknown and one can always find the results with desired accuracy. Methods those are employed for getting the solution of system of linear equations and development of parallel solvers are described in following sections. To test the performance of the developed solver, several data sets taken from linear elastic finite element analysis problem (discussed in chapter 4) and non-linear finite element analysis problems (discussed in chapter 5) were analyzed. The global stiffness matrices of these problem are of size 870 × 870, 882 × 882, 1226 × 1226, 1352 × 1352, 2312 × 2312, 3362 × 3362, and 4232 × 4232.

3.4.1 Gauss-Seidel Method

Gauss-Seidel Method (GSM) is an Iterative method so the solution of system of linear equations can be found of the desired accuracy. In this method, the iterations are carried out until the desired accuracy is achieved. It is quite natural that as the desired accuracy increases, the number of iterations also increases correspondingly. Let us consider a system of linear equations

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n &= b_2 \\
a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \cdots + a_{3n}x_n &= b_3 \\
\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots& \\
a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \cdots + a_{nn}x_n &= b_n
\end{aligned} \tag{3.1}$$

Initially some guess values of solution vector ($X^{(0)}$) are assumed. After dividing every equation by its diagonal element, equation 3.1 can be rewritten as

$$\begin{aligned}
x_1^{(1)} &= \frac{b_1}{a_{11}} - \frac{a_{12}}{a_{11}}x_2^{(0)} - \frac{a_{13}}{a_{11}}x_3^{(0)} - \cdots - \frac{a_{1n}}{a_{11}}x_n^{(0)} \\
x_2^{(1)} &= \frac{b_2}{a_{22}} - \frac{a_{21}}{a_{22}}x_1^{(1)} - \frac{a_{23}}{a_{22}}x_3^{(0)} - \cdots - \frac{a_{2n}}{a_{22}}x_n^{(0)} \\
x_3^{(1)} &= \frac{b_3}{a_{33}} - \frac{a_{31}}{a_{33}}x_1^{(1)} - \frac{a_{32}}{a_{33}}x_2^{(1)} - \cdots - \frac{a_{3n}}{a_{33}}x_n^{(0)} \\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
x_n^{(1)} &= \frac{b_n}{a_{nn}} - \frac{a_{n1}}{a_{nn}}x_1^{(1)} - \frac{a_{n2}}{a_{nn}}x_2^{(1)} - \cdots - \frac{a_{n,n-1}}{a_{nn}}x_{n-1}^{(0)}
\end{aligned} \tag{3.2}$$

Here $x_1^{(1)},\cdots,x_n^{(1)}$ represent the solution of system of linear equations after first iteration. These values will be used for the next iteration. The solution for (i+1)th iteration can be found by expression

$$\begin{aligned}
x_1^{(i+1)} &= \frac{b_1}{a_{11}} - \frac{a_{12}}{a_{11}}x_2^{(i)} - \frac{a_{13}}{a_{11}}x_3^{(i)} - \cdots - \frac{a_{1n}}{a_{11}}x_n^{(i)} \\
x_2^{(i+1)} &= \frac{b_2}{a_{22}} - \frac{a_{21}}{a_{22}}x_1^{(i)} - \frac{a_{23}}{a_{22}}x_3^{(i)} - \cdots - \frac{a_{2n}}{a_{22}}x_n^{(i)} \\
x_3^{(i+1)} &= \frac{b_3}{a_{33}} - \frac{a_{31}}{a_{33}}x_1^{(i)} - \frac{a_{32}}{a_{33}}x_2^{(i)} - \cdots - \frac{a_{3n}}{a_{33}}x_n^{(i)} \\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
x_n^{(i+1)} &= \frac{b_n}{a_{nn}} - \frac{a_{n1}}{a_{nn}}x_1^{(i)} - \frac{a_{n2}}{a_{nn}}x_2^{(i)} - \cdots - \frac{a_{n,n-1}}{a_{nn}}x_{n-1}^{(i)}
\end{aligned} \tag{3.3}$$

The generalized iteration formula can be written as

$$X^{(i+1)} = BX^{(i)} + C \tag{3.4}$$

The iterations are carried out until the desired accuracy is achieved. The accuracy is checked by comparing two vectors $X^{(i)}$ and $X^{(i+1)}$. The numbers of computations carried out in entire process can be approximately estimated by the following expression

$$\text{Number of computations} = (3n^2)i \tag{3.5}$$

3.4.1.1 Parallel Implementation

In the development of parallel solver using Gauss-Seidel Method on supercomputer, column wise data distribution was carried out. Initially all processors were given ranks starting from zero. Then the data was uniformly distributed to all processors. If uniform data distribution was not possible, few processors with lower ranks were overloaded with few columns. A typical uneven data distribution is shown in Fig. 3.2, where the number of equations and the number of processors are twenty and eight respectively. It can be observed that four processors with lowermost ranks are overloaded by one additional equation. Therefore they have to perform more computations as compared to the rest of the processors. Because of this overloading of data on lower rank processors, the Total time may get affected to a little extent. The reason for this is the remaining processors shall remain idle for small duration when the overloaded processors handles the additional data supplied to them.

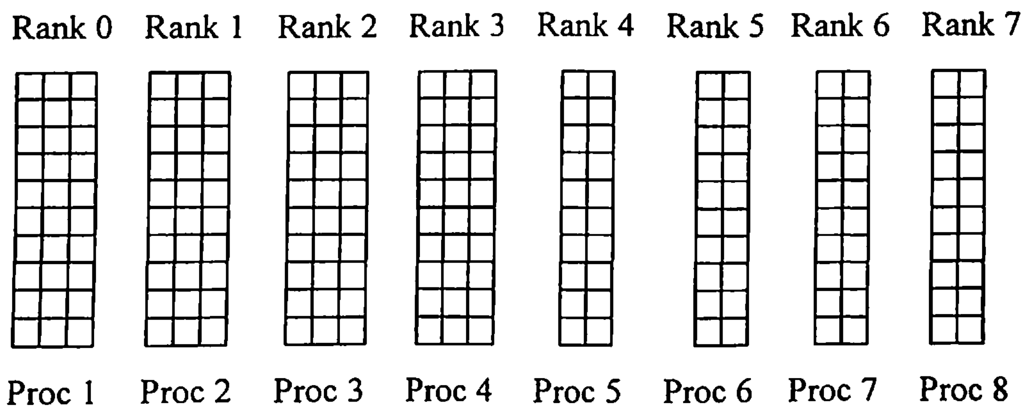
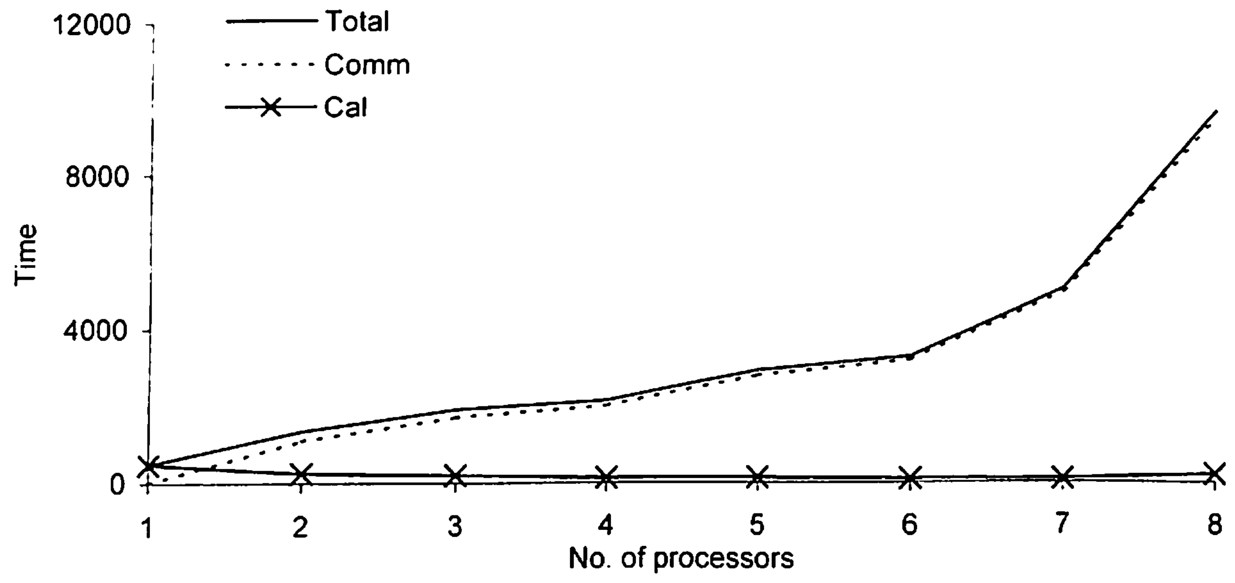


Fig. 3.2 Uneven data distribution (column wise) among the processors

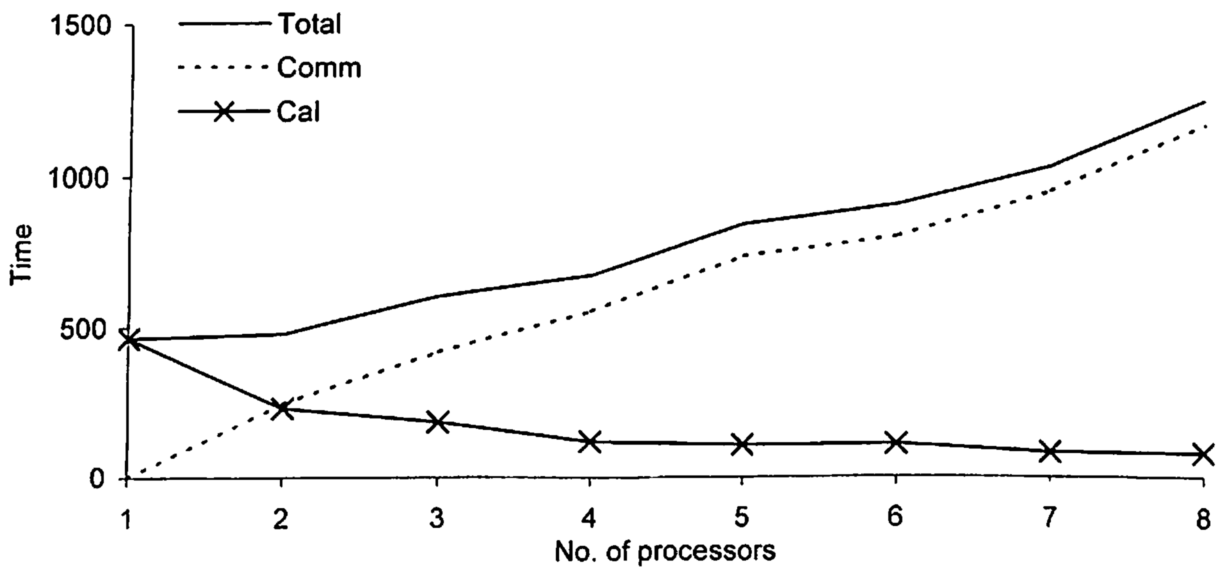
Each processor operated columns of its share and evaluated sum that was necessary to compute the numerical value of each unknown. Then each processor sent this sum to the master processor and master processor added them to get the new value unknown vector $\{X\}^i$. After this master processor broadcasted new value of unknown vector $\{X\}^i$ to all the processors which were used for further calculation. This process continued till current iteration got completed. After every iteration, all processors checked the accuracy of the obtained solution by comparing the old values with the new values of the unknown vector $\{X\}$. If all values of unknown vector $\{X\}$ have achieved the desired accuracy, then the master processor printed the results and the program got terminated. The parallel algorithm is presented in Fig. 3.3.

```
Global  P       {Number of Processors}
        n       {Number of Equations}
        MyRank  {Rank of the Processor}
        Rank    {Rank of processor holding current row}
        start   {Flag indicating starting row number for each processor}
        end     {Flag indicating ending row number for each processor }
        i,j,k   {Control Variables}
        itr     {Maximum No. of Iterations}
for all Pi where $0 \le i \le P$ do
        Set start
        Set end
end for
for i = 0 to itr-1
        for j = 0 to n-1
                for k = start to end
                        sum = sum – $[A]_{jk} / [A]_{jj} \times [X]_k$
                end for
                for all Pi where $1 \le i \le P$ do
                        Send sum to Master
                end for
                if MyRank == Master
                        Broadcast sum
                $[X]_j$ = sum
                end if
        endfor
        if MyRank == Master
                Calculate $[X]_j$
                Check accuracy of $[X]_j$
        else
                if
                        desired accuracy achieved
                then
                        Terminate process
                end if
        end if
endfor
```

Fig. 3.3 Parallel algorithm for Gauss-Seidel Method



(a) Variation in computational time components measured in terms of Real time



(b) Variation in computational time components measured in terms of User time

Fig. 3.4 Variation in computational time components for GSM solver for data set of size 1226 × 1226

3.4.1.2 Computational Time Results

Figure 3.4 shows variation in different components of computational time obtained for data set of size 1226 × 1226. Figure 3.4(a) shows that Total time (RT) increases with increasing number of processors. It can also be observed that Communication time (RT) is marginally smaller than the Total time. This indicates that the increase in Communication time is only responsible for increase in Total time with increase in number of processors. It can also be observed that the Calculation time (RT) reduces with increase in number of processors. From Fig. 3.4(b), it can be observed that Communication time (UT) is slightly less than the Total time for all number of processor (UT). Both the time components increase with the increase in number of processors. Reduction in Calculation time (UT) can be seen from this figure. Great difference between Total time (RT) and Total time (UT) as well as Communication time (RT) and Communication time (UT) can be observed from Fig. 3.4(a) and Fig. 3.4(b) both. Considerable amount of time is wasted in establishing Real time communication between the processors. The major part of data communication was performed by MPI_SEND subroutine. This communication was carried out in sequential way therefore large amount of time is consumed in this process.

Figure 3.5 shows variation in the Speedup measured in terms of Real time as well as User time with number of processors. It can be observed that the obtained Speedup is not encouraging. It is observed that Speedup reduces with increase in processors. Sudden reduction in Real time Speedup can be observed whereas gradual reduction in User time Speedup can be seen in Fig. 3.5.

3.4.1.3 Solver Performance

From Tables 3.1(a) to 3.6(a), it can be observed that the Total time increases with the increase in number of processors for all data sets. The variation in Total time (RT) is very abrupt and increasing with number of processors. One can find that the contribution of Communication time towards the Total time is very significant. The variation in Communication time is also abrupt and continuously increasing with number of processors. It can be observed that the variation in Calculation time (RT) is abrupt

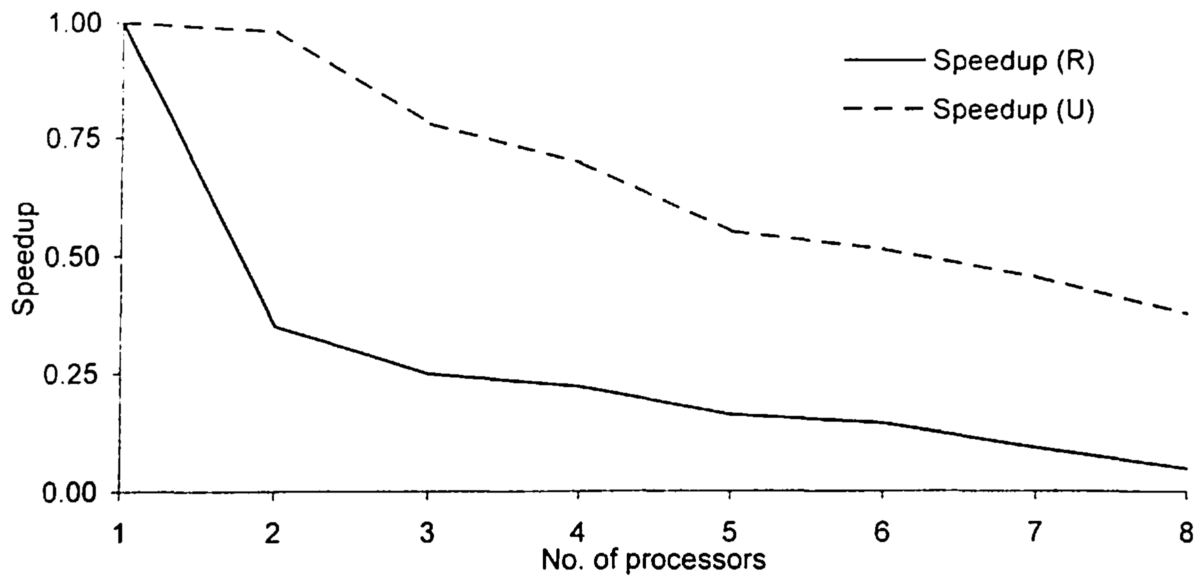


Fig. 3.5 Speedup achieved by GSM solver for data set of size 1226 × 1226

Table 3.1 Computational time variation and performance of GSM solver for data set of size 870 × 870

(a) Computational time variation

No. of processors	Total		Comm		Cal	
	Real	User	Real	User	Real	User
1	917.13	151.42	0.00	0.00	917.13	151.42
2	3636.32	211.26	3290.44	135.01	345.88	76.25
3	1372.61	241.06	1254.86	185.77	117.75	55.29
4	1668.10	278.55	1573.40	238.69	94.70	39.86
5	2433.74	359.31	2356.21	327.60	77.53	31.71
6	2805.05	384.87	2742.56	356.08	62.49	28.79
7	4121.00	484.16	4061.42	456.19	59.58	27.97
8	4786.46	537.93	4690.07	510.79	96.39	27.14

(b) Performance

No. of processors	Speedup		Efficiency		MFLOPS	
	Real	User	Real	User	Real	User
1	1.00	1.00	100.00	100.00	2.68	16.24
2	0.11	0.73	5.44	36.47	0.68	11.64
3	0.07	0.58	2.47	19.20	1.79	10.20
4	0.06	0.50	1.43	12.43	1.47	8.83
5	0.08	0.32	1.66	6.39	1.01	6.84
6	0.04	0.25	0.59	4.15	0.88	6.39
7	0.03	0.24	0.49	3.36	0.60	5.08
8	0.06	0.22	0.77	2.78	0.51	4.57

Table 3.2 Computational time variation and performance of GSM solver for data set of size 882 × 882

(a) Computational time variation

No. of processors	Total		Comm		Cal	
	Real	User	Real	User	Real	User
1	1543.85	590.24	0.00	0.00	1543.85	590.24
2	14199.89	809.27	8274.49	497.25	5925.40	312.02
3	20844.14	1024.52	19312.89	796.84	1531.25	227.68
4	27034.03	1186.78	23874.15	1004.16	3159.88	182.62
5	18631.41	1848.62	18308.66	1720.70	322.75	127.92
6	43383.03	2371.09	39410.62	2257.79	3972.41	113.30
7	44825.91	2507.37	44525.85	2403.06	300.06	104.31
8	24910.76	2657.27	24604.34	2547.59	306.42	109.68

(b) Performance

No. of processors	Speedup		Efficiency		MFLOPS	
	Real	User	Real	User	Real	User
1	1.00	1.00	100.00	100.00	6.67	17.44
2	0.11	0.73	5.44	36.47	0.72	12.72
3	0.07	0.58	2.47	19.20	0.49	10.05
4	0.06	0.50	1.43	12.43	0.38	8.67
5	0.08	0.32	1.66	6.39	0.55	5.57
6	0.04	0.25	0.59	4.15	0.24	4.34
7	0.03	0.24	0.49	3.36	0.23	4.11
8	0.06	0.22	0.77	2.78	0.41	3.87

Table 3.3 Computational time variation and performance of GSM solver for data set of size 1352 × 1352

(a) Computational time variation

No. of processors	Total		Comm		Cal	
	Real	User	Real	User	Real	User
1	2137.52	2096.68	0.00	0.00	2079.51	2054.72
2	5851.95	2091.95	4699.57	1040.07	1152.38	1051.88
3	8773.02	2542.75	7592.37	1791.86	1180.65	750.89
4	9695.634	2554.486	8085.04	1856.22	1610.59	698.27
5	10720.47	2685.823	10297.43	2036.45	423.03	649.37
6	13899.23	2714.16	12355.89	2378.21	1543.34	335.95
7	18486.69	2887.77	17262.76	2452.52	1223.93	435.25
8	20302.83	2960.82	19126.73	2507.44	1176.10	453.38

(b) Performance

No. of processors	Speedup		Efficiency		MFLOPS	
	Real	User	Real	User	Real	User
1	1.00	1.00	100.00	100.00	4.63	4.72
2	0.37	1.00	18.26	50.11	1.69	4.73
3	0.24	0.82	8.12	27.49	1.13	3.89
4	0.22	0.82	5.51	20.52	1.02	3.87
5	0.20	0.78	3.99	15.61	0.92	3.68
6	0.15	0.77	2.56	12.87	0.71	3.64
7	0.12	0.73	1.65	10.37	0.53	3.42
8	0.11	0.71	1.32	8.85	0.49	3.34

Table 3.4 Computational time variation and performance of GSM solver for data set of size 2312 × 2312

(a) Computational time variation

No. of processors	Total		Comm		Cal	
	Real	User	Real	User	Real	User
1	10594.34	10398.68	0.00	0.00	10594.34	10398.68
2	20674.31	8408.07	14929.01	3101.45	5745.30	5306.62
3	24075.27	9084.48	20292.17	5063.53	3783.10	4020.95
4	27706.09	9097.57	24739.52	6392.13	2966.57	2705.44
5	41956.26	11239.07	35752.82	9197.85	6203.44	2041.22
6	61758.57	12787.51	51080.07	10881.29	10678.50	1906.22
7	70063.76	13818.01	64693.65	11881.42	5370.11	1936.59
8	78307.23	14848.51	71567.31	12881.55	6739.92	1966.96

(b) Performance

No. of processors	Speedup		Efficiency		MFLOPS	
	Real	User	Real	User	Real	User
1	1.00	1.00	100.00	100.00	17.50	17.83
2	0.51	1.24	25.62	61.84	8.97	22.05
3	0.44	1.14	14.67	38.16	7.70	20.41
4	0.38	1.14	9.56	28.58	6.69	20.38
5	0.25	0.93	5.05	18.50	4.42	16.49
6	0.17	0.81	2.86	13.55	3.00	14.50
7	0.15	0.75	2.16	10.75	2.65	13.42
8	0.14	0.70	1.69	8.75	2.37	12.48

Table 3.5 Computational time variation and performance of GSM solver for data set of size 3362 × 3362

(a) Computational time variation

No. of processors	Total		Comm		Cal	
	Real	User	Real	User	Real	User
1	22683.61	22444.38	0.00	0.00	22683.61	22444.38
2	29930.18	16299.98	17278.04	3713.16	12652.14	12586.82
3	33455.98	17639.91	24361.94	5560.41	9094.04	12079.50
4	65716.35	15449.73	58399.71	8403.15	7316.64	7046.58
5	85234.62	17216.30	77902.00	11472.83	7332.62	5743.47
6	69838.11	18031.75	64209.11	12554.55	5629.00	5477.20
7	73974.97	19033.29	68851.43	14132.50	5123.55	4900.79
8	169263.09	21901.75	161449.97	17133.09	7813.12	4768.66

(b) Performance

No. of processors	Speedup		Efficiency		MFLOPS	
	Real	User	Real	User	Real	User
1	1.00	1.00	100.00	100.00	15.08	15.24
2	0.76	1.38	37.89	68.85	11.43	20.98
3	0.68	1.27	22.60	42.41	10.22	19.39
4	0.35	1.45	8.63	36.32	5.20	22.14
5	0.27	1.30	5.32	26.07	4.01	19.87
6	0.32	1.24	5.41	20.75	4.90	18.97
7	0.31	1.18	4.38	16.85	4.62	17.97
8	0.13	1.02	1.68	12.81	2.02	15.62

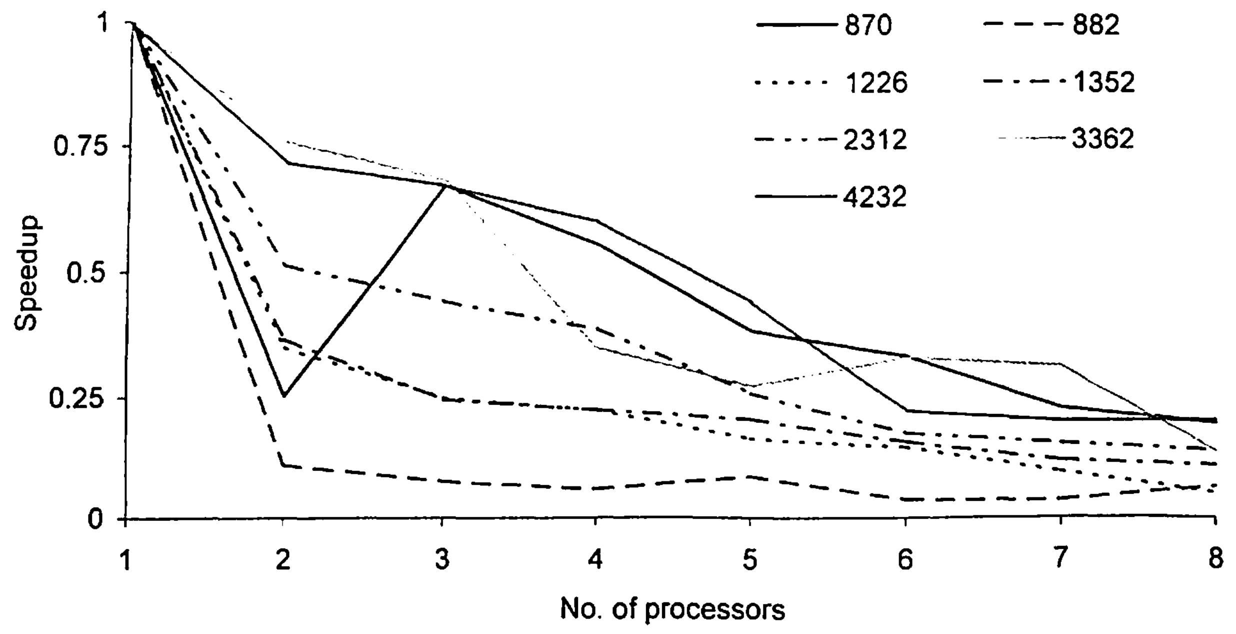
Table 3.6 Computational time variation and performance of GSM solver for data set of size 4232 × 4232

(a) Computational time variation

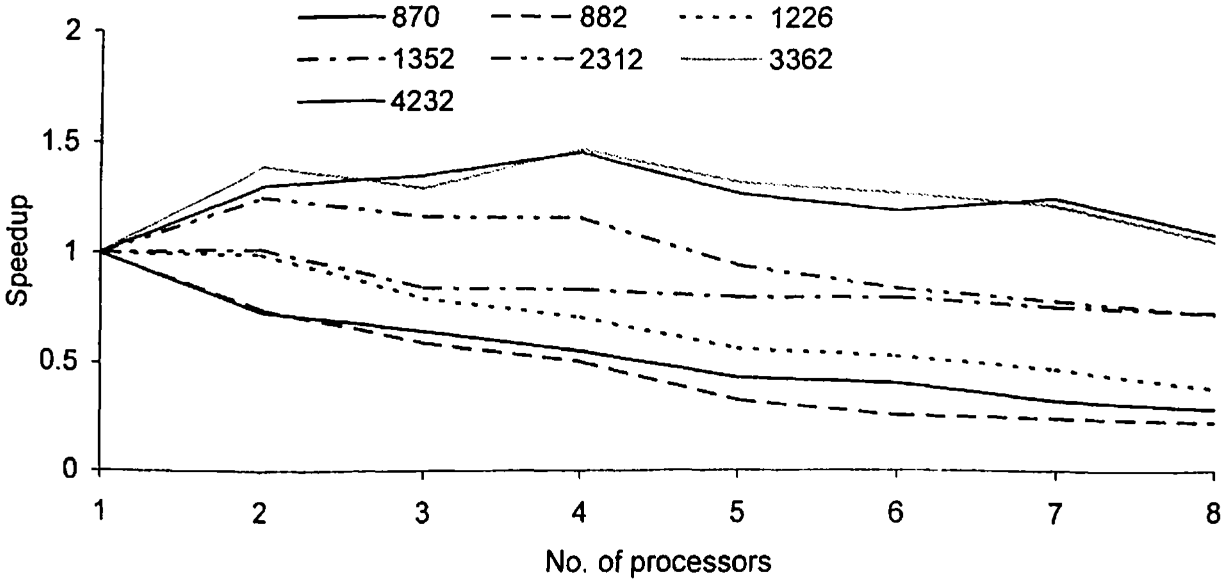
No. of processors	Total		Comm		Cal	
	Real	User	Real	User	Real	User
1	38397.66	38178.03	0.00	0.00	38397.66	38178.03
2	53737.73	29647.70	28136.16	6383.55	25601.57	23264.15
3	57460.65	28692.96	39038.58	10919.19	18422.06	17773.77
4	64550.05	26564.48	50012.16	13134.65	14537.89	13429.83
5	88564.14	30487.54	75100.08	18464.43	13464.06	12023.11
6	176748.32	32763.93	165223.59	22032.00	11524.73	10731.93
7	195356.06	31491.16	182366.97	22141.19	12989.09	9349.97
8	194341.54	36157.57	179727.38	26873.62	14614.16	9283.95

(b) Performance

No. of processors	Speedup		Efficiency		MFLOPS	
	Real	User	Real	User	Real	User
1	1.00	1.00	100.00	100.00	11.84	11.91
2	0.71	1.29	35.73	64.39	8.46	15.34
3	0.67	1.33	22.27	44.35	7.91	15.85
4	0.59	1.44	14.87	35.93	7.05	17.12
5	0.43	1.25	8.67	25.05	5.13	14.92
6	0.22	1.17	3.62	19.42	2.57	13.88
7	0.20	1.21	2.81	17.32	2.33	14.44
8	0.20	1.06	2.47	13.20	2.34	12.58



(a) Variation in Real time Speedup



(b) Variation in User time Speedup

Fig. 3.6 Variation in Speedup for GSM solver for various data sets

whereas variation in Calculation time (UT) is smooth and reducing with increase in number of processors. Reduction in Calculation time and increase in Communication time highlights the requirement of significant enhancement of the communication speed between the processors. It is immaterial to mention that this Total time increases with increase in size of data set that makes performance of GSM inadequate.

Reduction in Speedup, Efficiency and MFLOPS with increasing number of processors can also be seen in Tables 3.1(b) to 3.6(b). This is mainly because of increase in Total time (RT and UT) with increase in number of processors. Figure 3.6 (a) and (b) shows variation in the Real time Speedup and the User time Speedup respectively for various data sets under consideration. It can be seen from Fig. 3.6 (a) that variation in the Real time Speedup is abrupt and constantly reducing with increase in number of processors for all data sets. One can also observe that for smaller data set the performance of GSM solver is very poor as compared to its performance for higher data sets. In short, the performance of GSM solver improves with increase in data size. For biggest data size also, its performance is unacceptable, because Speedup remains less than one. Variation in User time Speedup for various data sets is shown in Fig. 3.6 (b). It can be observed that the User time Speedup also reduces with increasing number of processors for data sets of size 870 × 870, 882 × 882, 1226 × 1226 and 1352 × 1352. For higher data sets of size 2312 × 2312, 3362 × 3362, and 4232 × 4232 the User time Speedup increase initially. After reaching to a peak value, this User time Speedup starts reducing. It can be observed that this peak value of User time Speedup is higher for highest data set. It can be concluded that the User time Speedup would improve with further increase in data size.

Gauss-Seidel Method is an Iterative method therefore required computational time is directly dependent on number of iterations carried out for getting the solution of desired accuracy. Table 3.7 shows number of iterations carried out in finding the solution of various data sets under consideration. It can be observed that solution of few data sets namely 870 × 870, 1226 × 1226 and 1352 × 1352, were found. Number of iteration required to achieve 90% accuracy in solution was less than 1.5 times of size of individual data set. For data set of size 1352 × 1352, the number of iterations are even less than the actual data size itself. For rest of the data sets, solutions with 90% accuracy are not found even after carrying out iterations three times (for data set of size 3362 × 3362 and 4232 ×

4232) and five time (for data set of size 882 × 882 and 2312 × 2312) of size of data sets. Therefore, one can conclude that there is no direct relationship between number of iteration and size of data. Number of iterations purely depends on the type of data to be handled and the initial guess.

Table 3.7 Number of iteration for various data sizes carried out by GSM solver

Data size	No. of iterations	Status
870 × 870	1083	Solution found
882 × 882	4411	Incomplete
1226 × 1226	1790	Solution found
1352 × 1352	1803	Solution found
2312 × 2312	11560	Incomplete
3362 × 3362	10087	Incomplete
4232 × 4232	12697	Incomplete

3.4.2 Gauss Elimination Method

Gauss Elimination Method (GEM) falls under the category of Direct method of solving system of linear equations. It reduces the original matrix of system of linear equations to an equivalent upper triangular matrix, which can be solved by method of back substitution. Let us consider a system of linear equations

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n &= b_2 \\
a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \cdots + a_{3n}x_n &= b_3 \\
\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots& \\
a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \cdots + a_{nn}x_n &= b_n
\end{aligned} \tag{3.6}$$

Equation 3.6 can be rewritten in the form of augmented matrix

$$\begin{bmatrix}
a_{11} & a_{12} & a_{13} & \cdots & a_{1n} & b_1 \\
a_{21} & a_{22} & a_{23} & \cdots & a_{2n} & b_2 \\
a_{31} & a_{32} & a_{33} & \cdots & a_{3n} & b_3 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} & b_n
\end{bmatrix} \tag{3.7}$$

To eliminate x_1 term from second equation, first equation was multiplied by factor $-a_{21}/a_{11}$ and then added to the second equation. Similar procedure is followed on rest of the elements in the lower triangle of given matrix. Equation 3.7 will become

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} & b_1 \\ 0 & a'_{22} & a'_{23} & \cdots & a'_{2n} & b'_2 \\ 0 & 0 & a'_{33} & \cdots & a'_{3n} & b'_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a'_{nn} & b'_n \end{bmatrix} \tag{3.8}$$

The values of unknown vector $\{X\}$ can be found out by method of back substitution. The number of computations carried out in Gauss Elimination Method can be roughly estimated by expression

$$\text{Number of computations} = n^2\frac{(n+1)}{2} \tag{3.9}$$

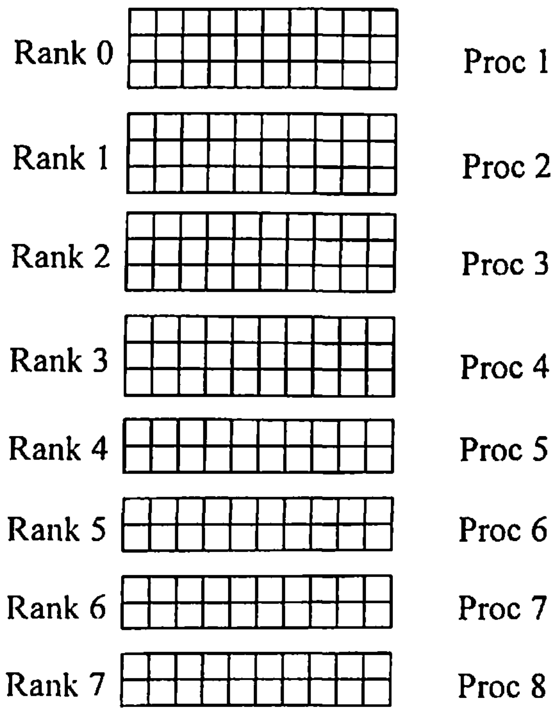


Fig. 3.7 Uneven data distribution (row wise) among the processors

3.4.2.1 Parallel Implementation

For parallel implementation of GEM on supercomputer, row wise data distribution was carried out. Initially the range of data to be handled by each processor was decided. If data distribution was not even, then the remaining data was distributed to the processors with lower ranks. Figure 3.7 shows a typical uneven data distribution among the processors. Here the number of equations and the number of processors are twenty and eight respectively. It can be observed that the four processors with lowermost ranks will be overloaded by one additional equation. Therefore those four processors have to perform more computations as compared to the rest of the processors.

```
Global  P        {Number of Processors}
        n        {Number of Equations}
        MyRank   {Rank of the Processor}
        Rank     {Rank of processor holding current row}
        start    {Flag indicating starting row number for each processor}
        end      {Flag indicating ending row number for each processor }
        i        {Variable indicating current row}
for all Pi where $0 \le i \le P$ do
        Set start
        Set end
end for
for i = 0 to n-1
        Set diagonal element of $[A]_i = 1.0$
        $[B]_i = [B]_i / [A]_{ii}$
        for all Pi where $0 \le i \le P$ do
                Find the Rank of current row
                If MyRank = Rank
                        Broadcast current row
                        Broadcast $[B]_i$
                end if
        end for
        for j = i to end
                Change non-diagonal element of $[A]_i = 0.0$
                Change elements of matrix $[B]_i$
        end for
end for
for i = end to start
        Compute $[x]_i$
end for
```

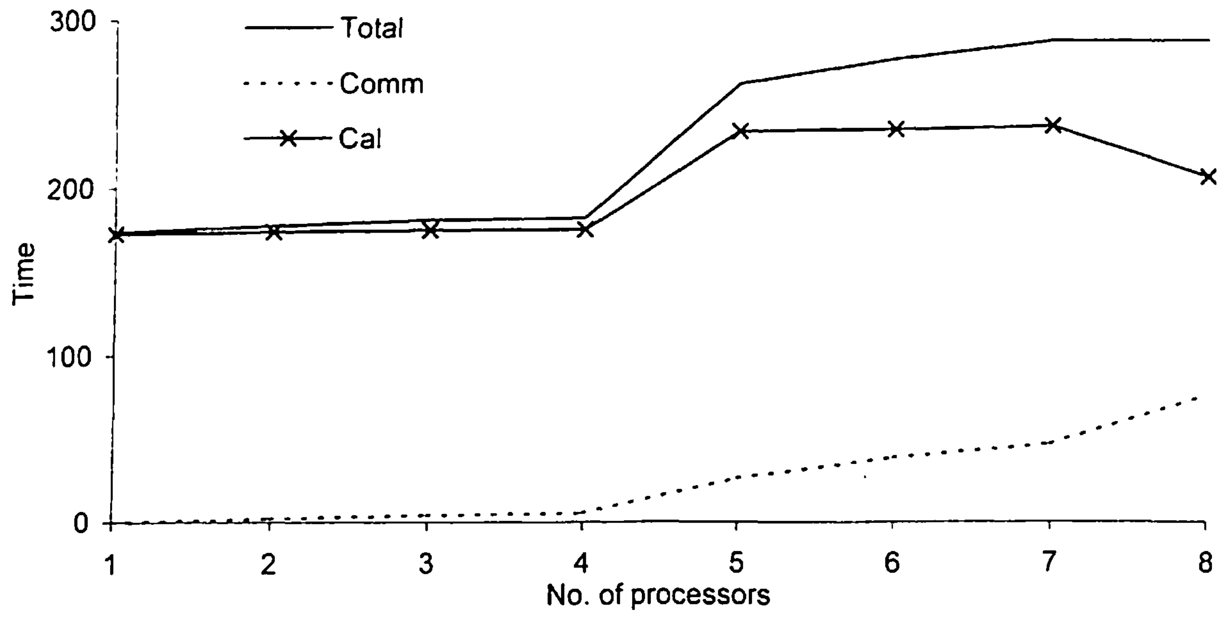
Fig. 3.8 Parallel algorithm for Gauss Elimination Method

The operations were started from first row and stopped at last row. Initially the diagonal element of current row was converted to unity, then the elements of lower triangle were converted to zero by every processor simultaneously. When the complete matrix reduced to upper triangular matrix, then the unknowns were evaluated by method of back substitution. This process is data dependent and the entire unknowns cannot be evaluated simultaneously. Therefore the processor with highest rank started this process. It evaluated all the unknowns of its share; thereafter broadcasted them to all the processors, which were utilized by the other processors to evaluate the unknowns of their share. Figure 3.8 shows the parallel algorithm for this method.

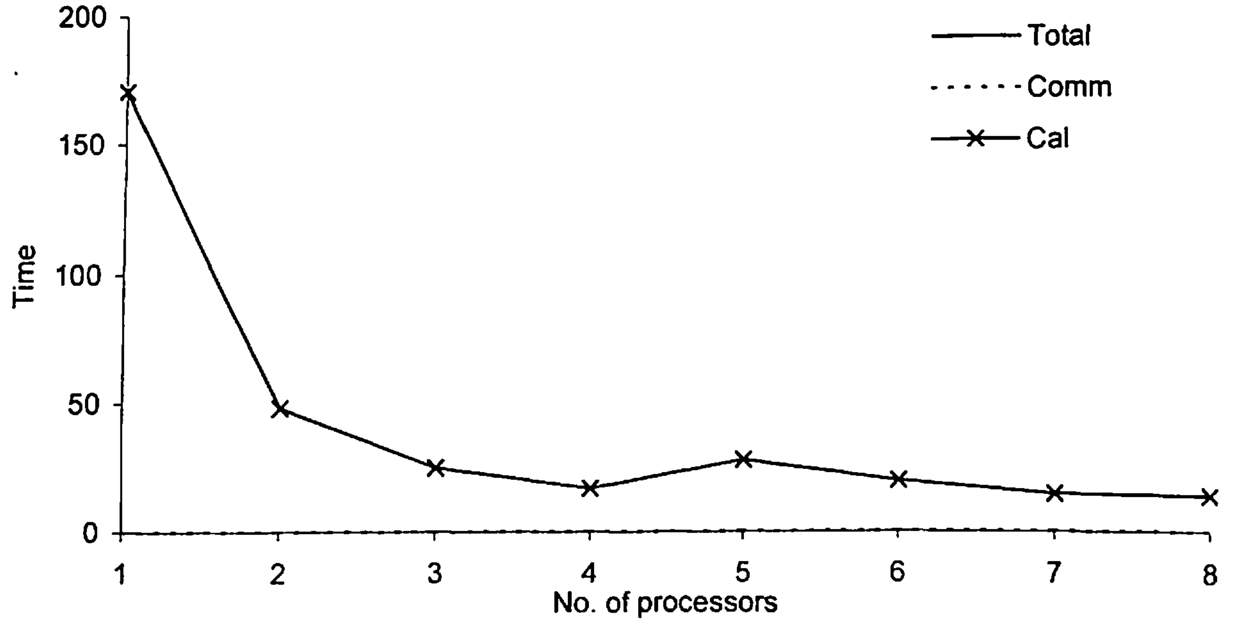
3.4.2.2 Computational Time Results

Figure 3.9 shows variation in different components of computational time obtained for data set of size 1226 × 1226. Figure 3.9(a) shows variation in different components of computational time measured in term of Real time. It can be observed that the Total time remains nearly same when one to four processors were employed to obtain the solution. Sudden increase in the Total time can be observed between the four and five processors. Thereafter the marginal increase in Total time can be seen. Communication time curve shows gradual increase in Communication time with increase in number of processors. Calculation time variation follows the similar pattern that of Total time. Calculation time curve lies just below the Total time curve for all processors. It can be observed that the subroutines MPI_SEND and MPI_BCAST are equally used for the data communication. In the last stage of this method, the unknowns are calculated by method of back substitution. At this point of time, only one processor remains active, while other processors remain idle because at this juncture they have insufficient data to evaluate unknowns of their share.

Figure 3.9(b) shows the variation in different components of computational time measured in terms of User time. It can be observed that Total time reduces with increase in number of processors. Drastic reduction can be observed from one processors to four processors. After four processors, the reduction in Total time is slow and insignificant. Communication time (UT) is negligible at every number of processors. Therefore Calculation time and Total time are almost equal.



(a) Variation in computational time components measured in terms of Real time



(b) Variation in computational time components measured in terms of User time

Fig. 3.9 Variation in computational time components for GEM solver for data set of size 1226 × 1226

Figure 3.10 shows variation in the Speedup calculated using Real time as well as User time. It can be seen that as number of processors increases the Real time Speedup reduces. Sudden reduction in Real time Speedup can be observed between four and five number of processors whereas User time Speedup increases with increase in number of processors up to four processors. There is sudden drop in User time Speedup between four and five processors. Maximum Speedup of approximately 12 can be seen at eight number of processors which is greater than Ideal Speedup.

3.4.2.3 Solver Performance

From Tables 3.8(a) to 3.13(a), it can be observed that the variation in Total time (RT) is very abrupt. It remains almost constant from one processor to four processors for all the data sets under consideration. There after it suddenly increases. After four processors, Total time (RT) starts increasing with increase in number of processors. It can be observed that Total time measured in term of User time reduces with increase in number of processors. The variation in Communication time is quite similar to the variation in Total time. Communication time measured in term of Real time as well as User time increases with increase in number of processors. It can be observed that Calculation time reduces with increase in number of processors. Overall performance of GEM is good and can be improved further by adopting higher communication speed.

Figure 3.11(a) shows variation in the Real time Speedup achieved by GEM for different data sets. It can be observed that Real time Speedup is almost constant from one to four number of processors for all the data sets under consideration. After four number of processors, the Real time Speedup variation is abrupt and its values is less than one. Variation in User time Speedup achieved by GEM for all data sets under consideration is shown in Fig. 3.11(b). It can be observed that the User time Speedup increases linearly from one to four number of processors. Thereafter, the increase is User time Speedup is abrupt. It can also be observed that for higher data size, User time Speedup is high as compared to the User time Speedup of low size data sets. It can be concluded that the performance of GEM increases with increase in data size. Maximum User time Speedup obtained is 35.63 at eight number of processor for the highest data set under consideration.

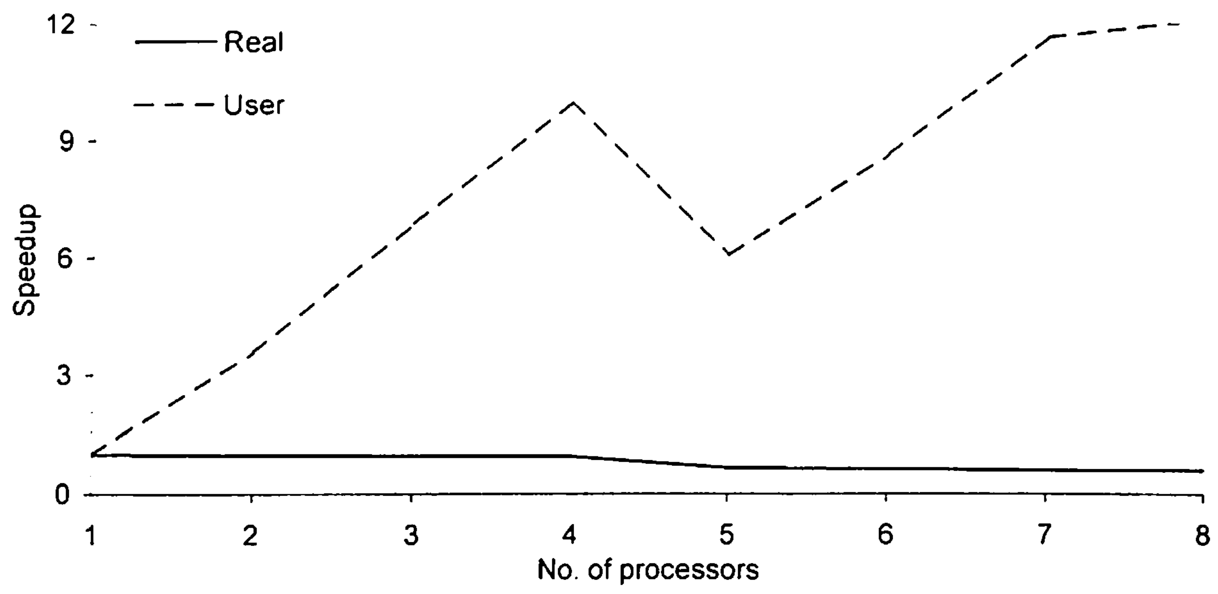


Fig. 3.10 Speedup achieved by GEM solver for data set of size 1226 × 1226

Table 3.8 Computational time variation and performance of GEM solver for data set of size 870 × 870

(a) Computational time variation

No. of processors	Total		Comm		Cal	
	Real	User	Real	User	Real	User
1	60.15	59.28	0.02	0.00	60.13	59.28
2	63.96	16.55	1.37	0.33	62.59	16.22
3	64.73	9.46	2.15	0.28	62.58	9.18
4	65.78	6.55	2.54	0.37	63.24	6.18
5	69.79	6.8	4.79	1.09	65.00	5.71
6	71.15	6.03	4.96	0.74	66.19	5.29
7	117.05	11.42	23.45	2.73	93.60	8.69
8	105	7.04	27.02	1.90	77.98	5.14

(b) Performance

No. of processors	Speedup		Efficiency		MFLOPS	
	Real	User	Real	User	Real	User
1	1.00	1.00	100.00	100.00	5.48	5.56
2	0.94	3.58	47.02	179.09	5.15	19.92
3	0.93	6.27	30.97	208.88	5.09	34.84
4	0.91	9.05	22.86	226.26	5.01	50.33
5	0.86	8.72	17.24	174.35	4.72	48.47
6	0.85	9.83	14.09	163.85	4.63	54.67
7	0.51	5.19	7.34	74.16	2.82	28.86
8	0.57	8.42	7.16	105.26	3.14	46.82

Table 3.9 Computational time variation and performance of GEM solver for data set of size 882 × 882

(a) Computational time variation

No. of processors	Total		Comm		Cal	
	Real	User	Real	User	Real	User
1	219.34	216.86	0.00	0.00	219.34	216.86
2	116.88	110.17	1.11	0.03	115.77	110.14
3	84.12	75.30	1.96	0.04	82.16	75.26
4	66.71	57.27	2.09	0.12	64.62	57.15
5	97.55	47.77	11.68	0.21	85.87	47.56
6	99.29	40.86	8.46	0.25	90.83	40.61
7	86.49	35.44	7.29	0.32	79.20	35.12
8	82.51	31.55	10.49	0.44	72.02	31.11

(b) Performance

No. of processors	Speedup		Efficiency		MFLOPS	
	Real	User	Real	User	Real	User
1	1.00	1.00	100.00	100.00	5.46	5.55
2	1.88	1.97	93.83	98.42	5.13	19.27
3	2.61	2.88	86.92	96.00	5.06	35.33
4	3.29	3.79	82.20	94.67	4.95	49.92
5	2.25	4.54	44.97	90.79	4.67	50.21
6	2.21	5.31	36.82	88.46	4.61	54.34
7	2.54	6.12	36.23	87.42	4.59	66.95
8	2.66	6.87	33.23	85.92	4.49	64.44

Table 3.10 Computational time variation and performance of GEM solver for data set of size 1352 × 1352

(a) Computational time variation

No. of processors	Total		Comm		Cal	
	Real	User	Real	User	Real	User
1	240.52	233.96	0.00	0.00	240.52	233.96
2	276.84	63.61	4.09	0.50	272.75	63.11
3	260.12	33.47	7.76	0.83	252.36	32.64
4	259.93	22.22	7.43	0.99	252.50	21.23
5	833.06	53.7	405.19	6.83	427.87	46.87
6	659.54	24.56	107.33	3.32	552.21	21.24
7	536.04	23.99	112.46	4.53	423.58	19.46
8	469.2	17.24	143.79	3.27	325.41	13.97

(b) Performance

No. of processors	Speedup		Efficiency		MFLOPS	
	Real	User	Real	User	Real	User
1	1.00	1.00	100.00	100.00	5.14	5.29
2	0.87	3.68	43.44	183.90	4.47	19.44
3	0.92	6.99	30.82	233.00	4.75	36.95
4	0.93	10.53	23.13	263.23	4.76	55.65
5	0.29	4.36	5.77	87.14	1.48	23.03
6	0.36	9.53	6.08	158.77	1.87	50.35
7	0.45	9.75	6.41	139.32	2.31	51.55
8	0.51	13.57	6.41	169.63	2.64	71.73

Table 3.11 Computational time variation and performance of GEM solver for data set of size 2312 × 2312

(a) Computational time variation

No. of processors	Total		Comm		Cal	
	Real	User	Real	User	Real	User
1	1685.81	1178.95	0.00	0.00	1685.81	1178.95
2	1172.75	301.04	16.70	0.07	1156.05	300.97
3	1177.77	144.06	21.72	0.09	1156.05	143.97
4	1214.47	93.82	33.28	0.13	1181.19	93.69
5	2390.98	171.94	154.59	0.08	2236.39	171.86
6	2098.02	115.74	187.16	0.11	1910.85	115.63
7	1816.85	92.86	113.33	0.17	1703.53	92.69
8	1566.45	65.26	171.18	0.17	1395.27	65.09

(b) Performance

No. of processors	Speedup		Efficiency		MFLOPS	
	Real	User	Real	User	Real	User
1	1.00	1.00	100.00	100.00	3.67	5.24
2	1.44	3.92	71.87	195.81	5.27	20.54
3	1.43	8.18	47.71	272.79	5.25	42.91
4	1.39	12.57	34.70	314.15	5.09	65.89
5	0.71	6.86	14.10	137.14	2.59	35.95
6	0.80	10.19	13.39	169.77	2.95	53.41
7	0.93	12.70	13.26	181.37	3.40	66.57
8	1.08	18.07	13.45	225.82	3.95	94.73

Table 3.12 Computational time variation and performance of GEM solver for data set of size 3362 × 3362

(a) Computational time variation

No. of processors	Total		Comm		Cal	
	Real	User	Real	User	Real	User
1	3629.89	3528.92	0.00	0.00	3629.89	3528.92
2	3554.37	923.32	33.73	0.11	3520.64	923.21
3	3570.68	436.01	40.11	0.10	3530.57	435.91
4	3594.91	265.91	64.16	0.15	3530.75	265.76
5	4049.39	275.57	138.46	0.19	3910.92	275.38
6	3766.95	201.24	113.29	0.23	3653.66	201.01
7	3687.99	200.03	93.71	0.20	3594.28	199.83
8	3651.06	107.10	81.13	0.21	3569.93	106.89

(b) Performance

No. of processors	Speedup		Efficiency		MFLOPS	
	Real	User	Real	User	Real	User
1	1.00	1.00	100.00	100.00	5.24	5.39
2	1.02	3.82	51.06	191.10	5.35	20.58
3	1.02	8.09	33.89	269.79	5.32	43.59
4	1.01	13.27	25.24	331.78	5.29	71.48
5	0.90	12.81	17.93	256.12	4.69	68.97
6	0.96	17.54	16.06	292.26	5.05	94.44
7	0.98	17.64	14.06	252.03	5.15	95.02
8	0.99	32.95	12.43	411.87	5.21	177.46

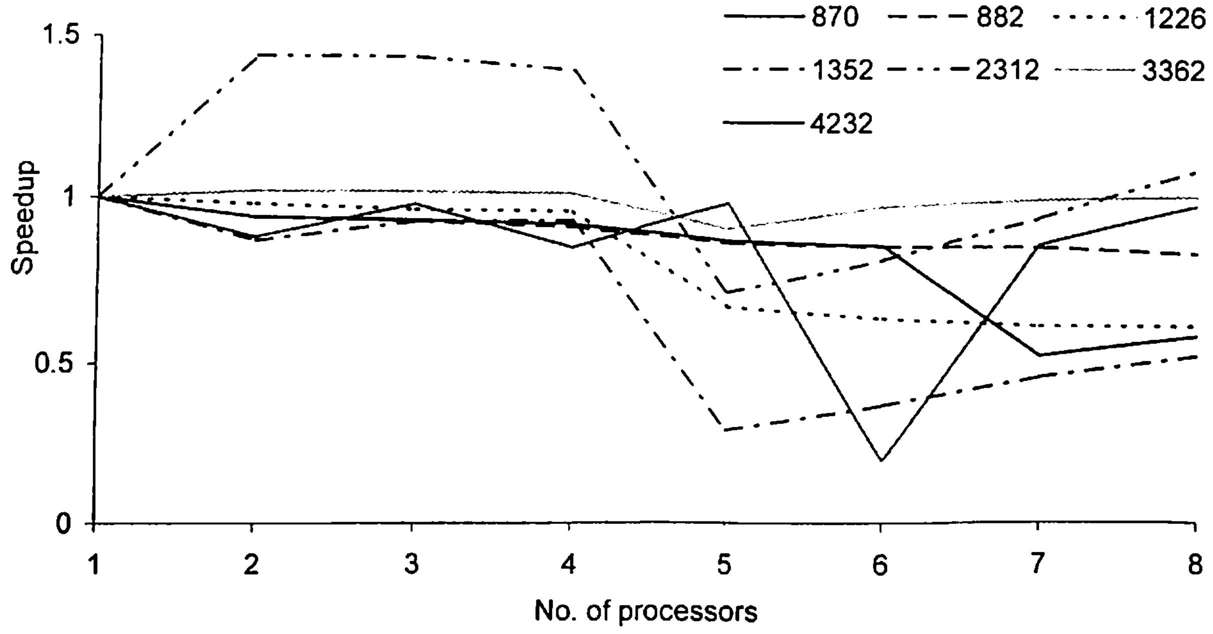
Table 3.13 Computational time variation and performance of GEM solver for data set of size 4232 × 4232

(a) Computational time variation

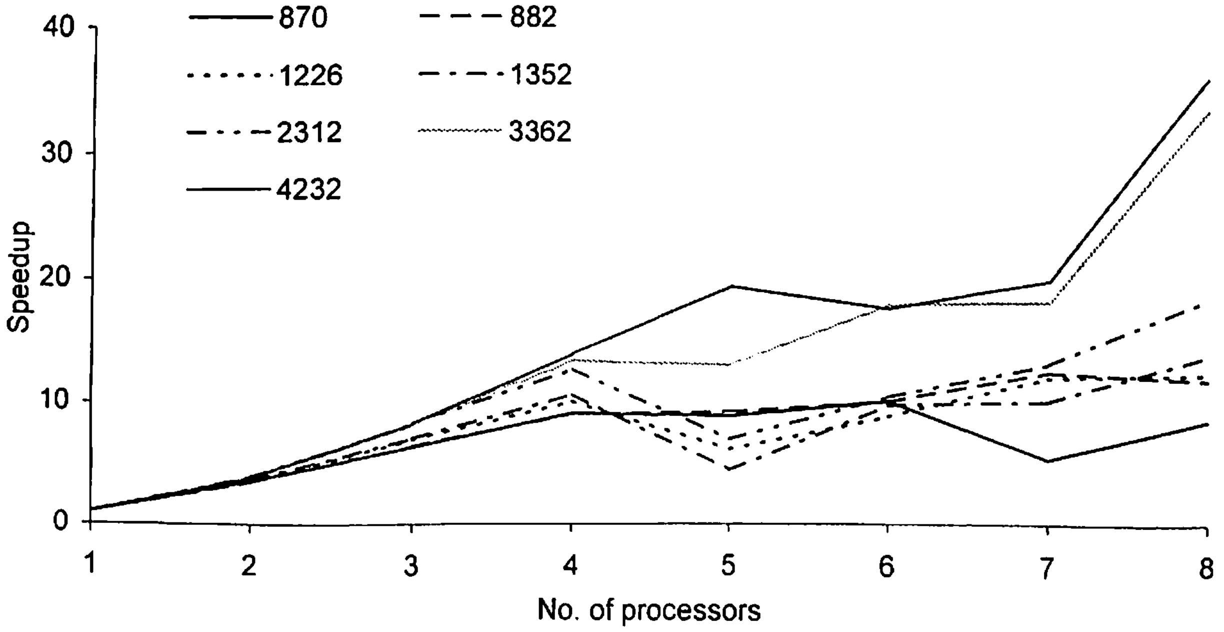
No. of processors	Total		Comm		Cal	
	Real	User	Real	User	Real	User
1	7057.91	7006.88	0.00	0.00	7057.91	7006.88
2	8028.15	1813.39	40.43	0.15	7987.72	1813.24
3	7210.35	858.16	59.59	0.15	7150.76	858.01
4	8368.35	507.39	168.98	0.24	8199.37	507.15
5	7236.71	365.61	73.82	0.20	7162.89	365.41
6	35742.16	407.48	1282.34	0.30	34459.82	407.18
7	8329.01	362.33	250.67	17.16	8078.34	345.17
8	7297.62	196.68	135.13	22.32	7162.49	174.36

(b) Performance

No. of processors	Speedup		Efficiency		MFLOPS	
	Real	User	Real	User	Real	User
1	1.00	1.00	100.00	100.00	5.37	5.41
2	0.88	3.86	43.96	193.20	4.72	20.90
3	0.98	8.17	32.63	272.17	5.26	44.17
4	0.84	13.81	21.09	345.24	4.53	74.71
5	0.98	19.16	19.51	383.30	5.24	103.68
6	0.20	17.20	3.29	286.59	1.06	93.03
7	0.85	19.34	12.11	276.26	4.55	104.62
8	0.97	35.63	12.09	445.32	5.19	192.73



(a) Variation in Real time Speedup



(b) Variation in User time Speedup

Fig. 3.11 Variation in Speedup for GEM solver for various data sets

3.4.3 Matrix Inversion Method

Matrix Inversion Method (MIM) is one of the common method for obtaining the solution of the system of linear equations $[A]\{X\} = \{B\}$. In this method inverse of the matrix $[A]$ is calculated. Then the solution is computed using expression $\{X\} = [A]^{-1}\{B\}$. To compute $[A]^{-1}$ let us consider an expression

$$[A][A]^{-1} = [I] \tag{3.10}$$

where $[I]$ is a unit matrix same of order same as matrix $[A]$. Matrix $[A]$ is operated and converted to unit matrix. All the necessary operation carried out on matrix $[A]$, are also carried out on matrix $[I]$. Finally, matrix $[A]$ gets converted to a unit matrix whereas matrix $[I]$ takes form of $[A]^{-1}$. Let us consider a system of linear equations

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n &= b_2 \\
a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \cdots + a_{3n}x_n &= b_3 \\
\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots & \\
a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \cdots + a_{nn}x_n &= b_n
\end{aligned} \tag{3.11}$$

Equation 3.11 can be rewritten in the augmented matrix form

$$\begin{bmatrix}
a_{11} & a_{12} & a_{13} & \cdots & a_{1n} & \cdot & 1 & 0 & 0 & \cdots & 0 \\
a_{21} & a_{22} & a_{23} & \cdots & a_{2n} & \cdot & 0 & 1 & 0 & \cdots & 0 \\
a_{31} & a_{32} & a_{33} & \cdots & a_{3n} & \cdot & 0 & 0 & 1 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \cdot & \vdots & \vdots & \vdots & \ddots & \vdots \\
a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} & \cdot & 0 & 0 & 0 & \cdots & 1
\end{bmatrix} \tag{3.12}$$

where left hand side represents the matrix $[A]$ and right hand side represents unit matrix $[I]$. After row operations Eq. 3.12 takes final form as

$$\left[\begin{array}{ccccc:ccccc} 1 & 0 & 0 & \cdots & 0 & \alpha_{11} & \alpha_{12} & \alpha_{13} & \cdots & \alpha_{1n} \\ 0 & 0 & 0 & \cdots & 0 & \alpha_{21} & \alpha_{22} & \alpha_{23} & \cdots & \alpha_{2n} \\ 0 & 0 & 0 & \cdots & 0 & \alpha_{31} & \alpha_{32} & \alpha_{33} & \cdots & \alpha_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & \alpha_{n1} & \alpha_{n2} & \alpha_{n3} & \cdots & \alpha_{nn} \end{array}\right] \tag{3.13}$$

where right hand side represents the inverse of matrix $[A]$. The values of unknown vector $\{X\}$ can be found out by expression

$$\begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{Bmatrix} = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \cdots & \alpha_{1n} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \cdots & \alpha_{2n} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \cdots & \alpha_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha_{n1} & \alpha_{n2} & \alpha_{3n} & \cdots & \alpha_{nn} \end{bmatrix} \begin{Bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{Bmatrix} \tag{3.14}$$

The number of computations carried out in Matrix Inversion Method can be expressed as

$$\text{Number of computations} = 4n^2 + 2n^3 \tag{3.15}$$

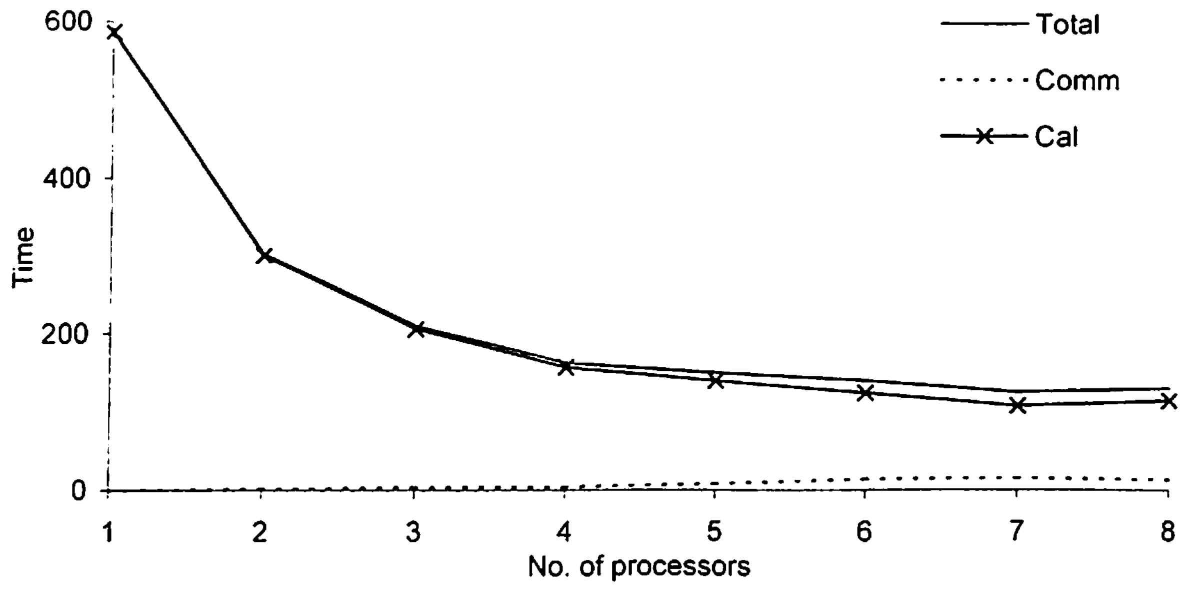
3.4.3.1 Parallel Implementation

Initially all processors were given their ranks. Then the range of data to be handled by each processor was decided. Row wise data distribution was carried out as described in Gauss Elimination Method. After proper data distribution among the processors, an Identity matrix $[I]$ of size $[A]$ was created by all processors. In the process of matrix inversion, column wise operations were carried out. Every non-diagonal element of matrix $[A]$ was converted to zero and every diagonal element of matrix $[A]$ was made unity.

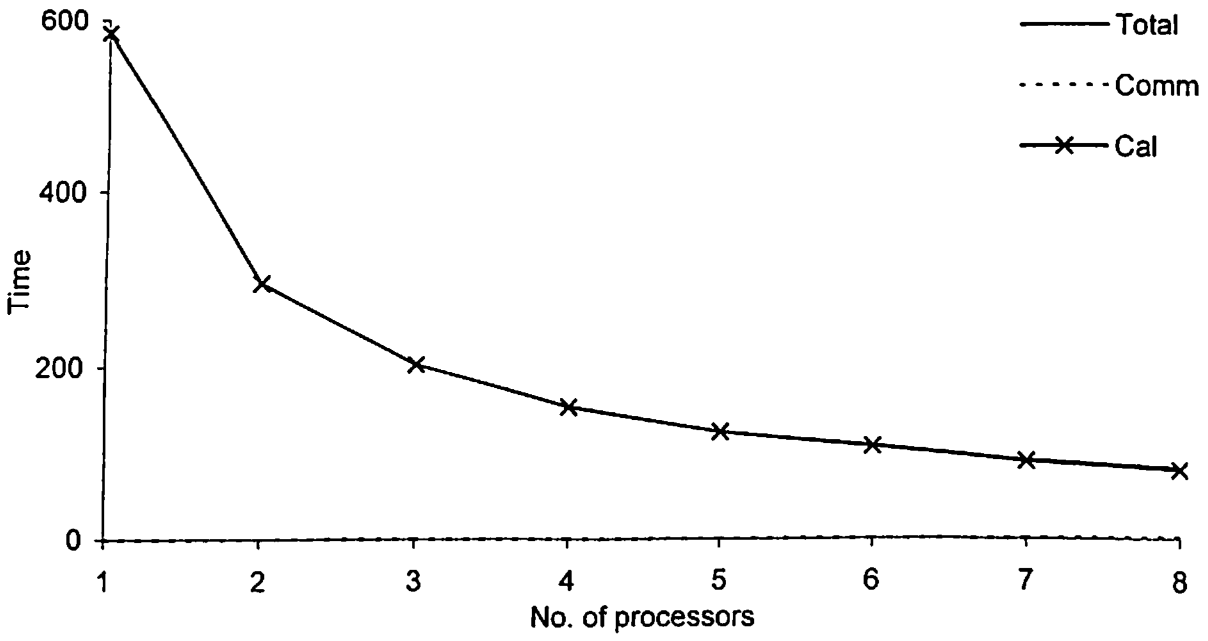
Whatever operations were carried out on matrix $[A]$, same operations were also carried out on matrix $[I]$ simultaneously. Each processor operated only those rows, which were designated to it to spend less computational time. After finding the inverse of matrix $[A]$, the unknown vector $\{X\}$ was calculated by multiplying $[A]^{-1}$ with $\{B\}$. At this juncture, each processor was having elements of vector $\{X\}$ those belong to its share. Then each processor broadcasted these elements of vector $\{X\}$ to the all other processors so that every processor should have complete vector $\{X\}$. Figure 3.12 shows the parallel algorithm for Matrix Inversion Method.

```
Global  P        {Number of Processors}
        n        {Number of Equations}
        MyRank   {Rank of the Processor}
        Rank     {Rank of processor holding current row}
        start    {Flag indicating starting row number for each processor}
        end      {Flag indicating ending row number for each processor }
        i        {Variable indicating current row}
        [I]      {Matrix indicating inverse of matrix [A]}
for all Pi where 0 < i < P do
        Set start
        Set end
for i = 0 to n-1
        Set diagonal element of [A]_i = 1.0
        Change elements of matrix [I]_i
        for all Pi where 0 < i < P do
                Find the Rank of current row
                If MyRank = Rank
                        Broadcast current row
                endif
        endfor
        for j = start to end
                if [A]_ij ≠ 0.0
                        Change non-diagonal element of [A]_ij = 0.0
                        Change elements of matrix  [I]_ij
                endif
        endfor
endfor
for i = start to end
        Compute {x}_i
endfor
for all Pi where 0 < i < P do
        Broadcast {x}_i to All Processor
endfor
```

Fig. 3.12 Parallel algorithm for Matrix Inversion Method



(a) Variation in computational time components measured in terms of Real time



(b) Variation in computational time components measured in terms of User time

Fig. 3.13 Variation in computational time components for MIM solver for data set of size

1226 × 1226

3.4.3.2 Computational Time Results

Figure 3.13 shows variation in different components of computational time obtained for data set of size 1226 × 1226. Figure 3.13(a) shows the variation in different components of computational time measured in terms of Real time. One can see that as the number of processor increases the Total time reduces considerably. The Calculation time also reduces with increase in number of processors. Communication time increases with increase in number of processors. This increase is very slow and insignificant.

Figure 3.13(b) shows variation in different components of computational time measured in terms of User time. It can be observed that Total time as well as Calculation time reduces with increase in number of processors. Sudden reduction can be seen from one to four processors but thereafter the reduction is slow. The Communication time (UT) is insignificant for Matrix Inversion Method. One can observe that the Total time and Calculation time is nearly same at every number of processors. It can also be observed that Total time measured in term of Real time as well as User time is nearly same at every number of processors

Figure 3.14 shows the variation in Speedup measured in terms of Real time as well as User time. It can be observed that maximum Real time Speedup of 5 (approximately) is achieved at seven number of processors and maximum User time Speedup of 7.2 is achieved at eight number of processors. The User time Speedup is very close to Ideal Speedup at every number of processors.

3.4.3.3 Solver Performance

From Tables 3.14 to 3.19, it can be observed that Matrix Inversion Method gives excellent performance for all data sets under consideration. Total time (RT and UT) reduces considerably with increase in number of processors. The contribution of Communication time is very less as compared to the Calculation time toward the Total time.

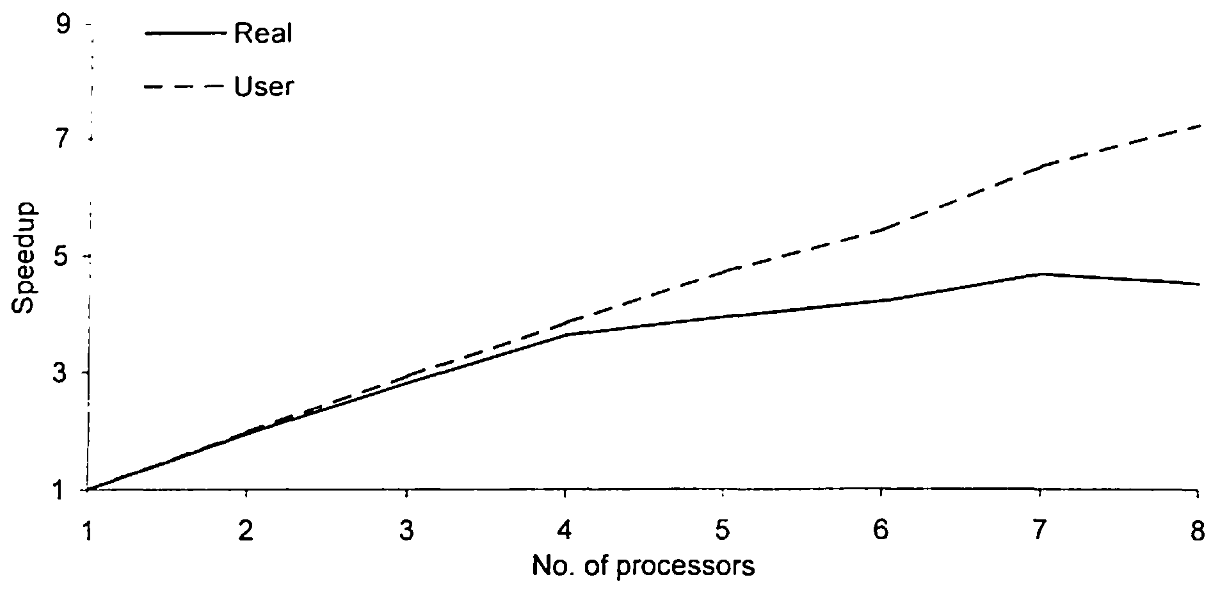


Fig. 3.14 Speedup achieved by MIM solver for data set of size 1226 × 1226

Table 3.14 Computational time variation and performance of MIM solver for data set of size 870 × 870

(a) Computational time variation

No. of processors	Total		Comm		Cal	
	Real	User	Real	User	Real	User
1	203.82	202.29	0.00	0.00	203.82	202.29
2	107.45	103.42	0.75	0.07	106.70	103.35
3	75.41	69.75	1.68	0.05	73.73	69.70
4	63.91	53.53	2.74	0.06	61.17	53.47
5	93.16	45.29	14.68	0.19	78.48	45.10
6	92.16	39.87	12.11	1.49	80.05	38.38
7	78.83	34.15	9.55	0.31	69.28	33.84
8	84.69	30.3	14.96	0.44	69.73	29.86

(b) Performance

No. of processors	Speedup		Efficiency		MFLOPS	
	Real	User	Real	User	Real	User
1	1.00	1.00	100.00	100.00	6.48	6.53
2	1.90	1.96	94.84	97.80	12.29	12.76
3	2.70	2.90	90.09	96.67	17.50	18.93
4	3.19	3.78	79.73	94.48	20.65	24.66
5	2.19	4.47	43.76	89.33	14.17	29.15
6	2.21	5.07	36.86	84.56	14.32	33.11
7	2.59	5.92	36.94	84.62	16.75	38.65
8	2.41	6.68	30.08	83.45	15.59	43.57

Table 3.15 Computational time variation and performance of MIM solver for data set of size 882 × 882

(a) Computational time variation

No. of processors	Total		Comm		Cal	
	Real	User	Real	User	Real	User
1	219.34	216.86	0.00	0.00	219.34	216.86
2	116.88	110.17	1.11	0.03	115.77	110.14
3	84.12	75.30	1.96	0.04	82.16	75.26
4	66.71	57.27	2.09	0.12	64.62	57.15
5	97.55	47.77	11.68	0.21	85.87	47.56
6	99.29	40.86	8.46	0.25	90.83	40.61
7	86.49	35.44	7.29	0.32	79.20	35.12
8	82.51	31.55	10.49	0.44	72.02	31.11

(b) Performance

No. of processors	Speedup		Efficiency		MFLOPS	
	Real	User	Real	User	Real	User
1	1.00	1.00	100.00	100.00	6.27	6.34
2	1.88	1.97	93.83	98.42	11.77	12.48
3	2.61	2.88	86.92	96.00	16.35	18.27
4	3.29	3.79	82.20	94.67	20.62	24.02
5	2.25	4.54	44.97	90.79	14.10	28.79
6	2.21	5.31	36.82	88.46	13.85	33.66
7	2.54	6.12	36.23	87.42	15.90	38.81
8	2.66	6.87	33.23	85.92	16.67	43.59

Table 3.16 Computational time variation and performance of MIM solver for data set of size 1352 × 1352

(a) Computational time variation

No. of processors	Total		Comm		Cal	
	Real	User	Real	User	Real	User
1	790.48	787.98	0.00	0.00	790.48	787.98
2	409.22	399.79	1.72	0.02	407.50	399.77
3	284.11	269.44	3.94	0.06	280.17	269.38
4	220.19	205.41	4.16	0.14	216.03	205.27
5	201.09	167.43	14.04	0.35	187.05	167.08
6	179.76	140.45	18.66	0.46	161.10	139.99
7	167.04	121.69	19.57	0.80	147.47	120.89
8	171.92	108.82	17.76	2.51	154.16	106.31

(b) Performance

No. of processors	Speedup		Efficiency		MFLOPS	
	Real	User	Real	User	Real	User
1	1.00	1.00	100.00	100.00	6.26	6.28
2	1.93	1.97	96.58	98.55	12.10	12.38
3	2.78	2.92	92.74	97.48	17.42	18.37
4	3.59	3.84	89.75	95.90	22.48	24.10
5	3.93	4.71	78.62	94.13	24.62	29.56
6	4.40	5.61	73.29	93.51	27.54	35.24
7	4.73	6.48	67.60	92.50	29.63	40.68
8	4.60	7.24	57.47	90.51	28.79	45.49

Table 3.17 Computational time variation and performance of MIM solver for data set of size 2312 × 2312

(a) Computational time variation

No. of processors	Total		Comm		Cal	
	Real	User	Real	User	Real	User
1	5524.44	3997.63	0.00	0.00	5524.44	3997.63
2	3114.43	2007.07	42.19	0.03	3072.24	2007.04
3	2164.64	1348.66	53.23	0.21	2111.41	1348.45
4	1634.28	1012.74	59.91	0.44	1574.38	1012.30
5	2126.91	827.82	82.14	0.86	2044.77	826.96
6	1687.16	697.32	73.21	1.29	1613.96	696.03
7	1462.59	602.21	72.71	1.77	1389.88	600.44
8	1289.62	530.21	80.04	2.46	1209.58	527.75

(b) Performance

No. of processors	Speedup		Efficiency		MFLOPS	
	Real	User	Real	User	Real	User
1	1.00	1.00	100.00	100.00	4.48	6.19
2	1.77	1.99	27.06	99.59	7.94	12.33
3	2.55	2.96	25.96	98.80	11.43	18.34
4	3.38	3.95	25.79	98.68	15.14	24.43
5	2.60	4.83	15.85	96.58	11.63	29.88
6	3.27	5.73	16.65	95.55	14.66	35.48
7	3.78	6.64	16.47	94.83	16.91	41.08
8	4.28	7.54	16.34	94.25	19.18	46.66

Table 3.18 Computational time variation and performance of MIM solver for data set of size 3362 × 3362

(a) Computational time variation

No. of processors	Total		Comm		Cal	
	Real	User	Real	User	Real	User
1	12245.14	12092.82	0.00	0.00	12245.14	12092.82
2	6131.13	6065.80	13.24	0.12	6117.89	6065.68
3	4130.53	4054.96	29.01	0.29	4101.52	4054.67
4	3176.20	3069.75	69.24	0.93	3106.96	3068.82
5	3968.67	2503.82	115.67	1.70	3853.00	2502.12
6	3418.15	2084.11	138.58	3.17	3279.57	2080.94
7	3016.27	1798.20	196.98	3.44	2819.29	1794.76
8	2156.61	1595.05	90.36	3.89	2066.25	1591.16

(b) Performance

No. of processors	Speedup		Efficiency		MFLOPS	
	Real	User	Real	User	Real	User
1	1.00	1.00	100.00	100.00	6.21	6.29
2	2.00	1.99	99.86	99.68	12.40	12.54
3	2.96	2.98	98.82	99.41	18.41	18.75
4	3.86	3.94	96.38	98.48	23.94	24.77
5	3.09	4.83	61.71	96.59	19.16	30.37
6	3.58	5.80	59.71	96.71	22.25	36.49
7	4.06	6.72	58.00	96.07	25.21	42.29
8	5.68	7.58	70.97	94.77	35.26	47.68

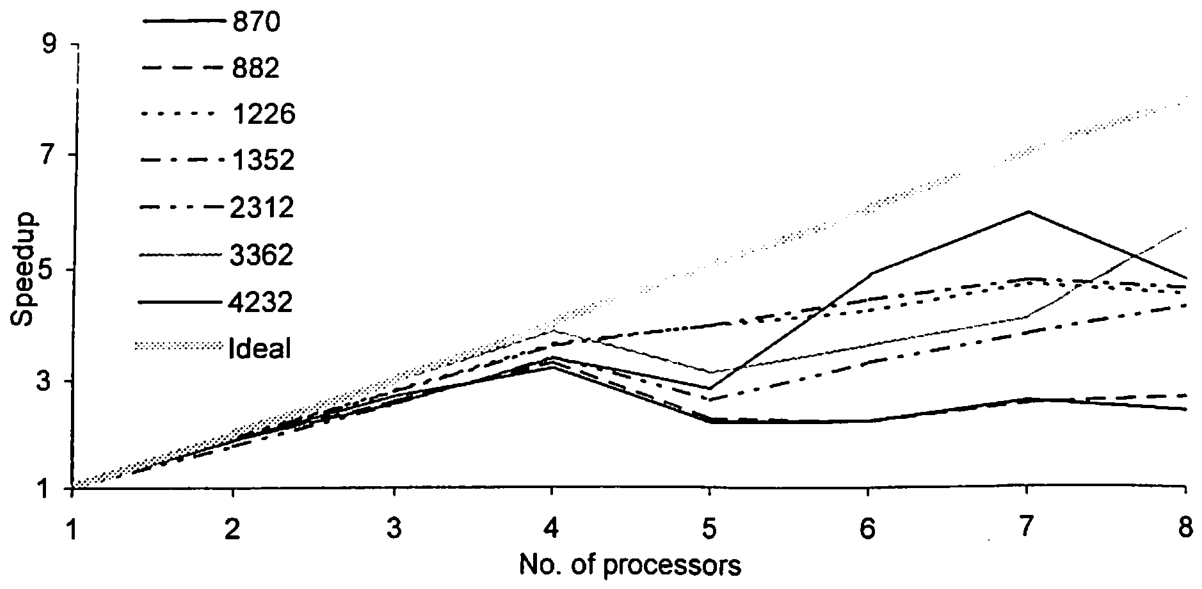
Table 3.19 Computational time variation and performance of MIM solver for data set of size 4232 × 4232

(a) Computational time variation

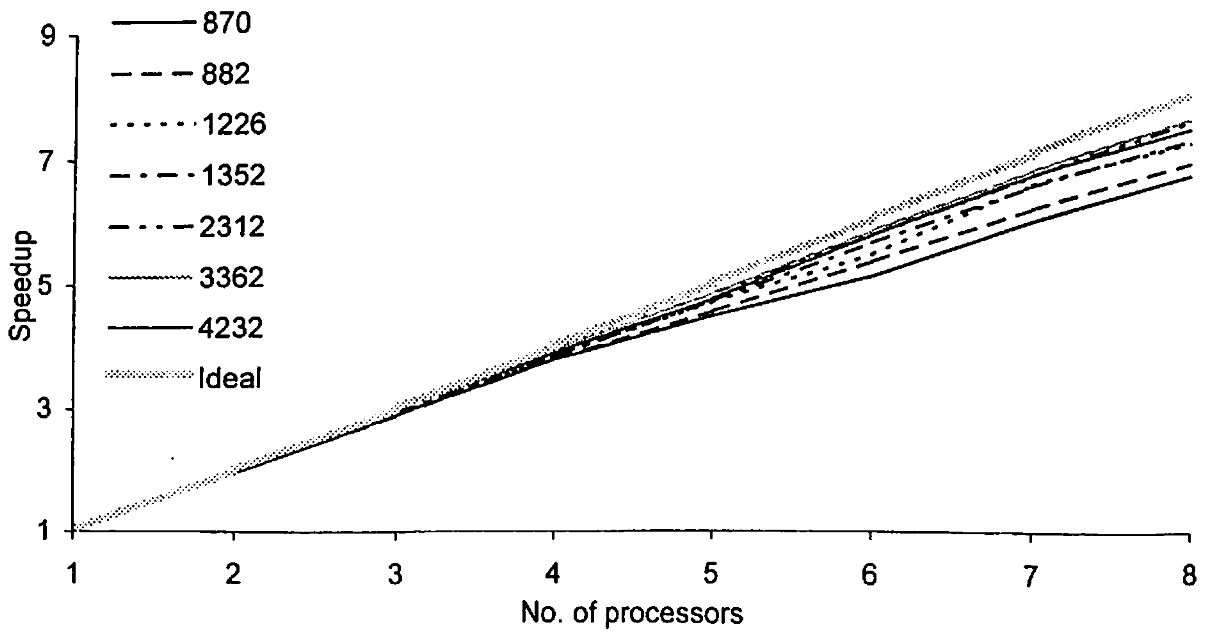
No. of processors	Total		Comm		Cal	
	Real	User	Real	User	Real	User
1	24039.08	23984.18	0.00	0.00	24039.08	23984.18
2	12781.28	12013.02	21.28	0.14	12760.00	12012.88
3	9370.52	8034.54	47.26	0.37	9323.26	8034.17
4	7143.15	6161.49	54.12	1.28	7089.03	6160.21
5	8593.55	5055.69	208.53	2.59	8385.02	5053.10
6	4971.59	4180.78	179.68	5.17	4791.91	4175.61
7	4058.18	3602.92	271.61	5.19	3786.57	3597.73
8	5060.32	3229.96	301.42	6.60	4758.90	3223.36

(b) Performance

No. of processors	Speedup		Efficiency		MFLOPS	
	Real	User	Real	User	Real	User
1	1.00	1.00	100.00	100.00	6.31	6.32
2	1.88	2.00	94.04	99.83	11.87	12.62
3	2.57	2.99	85.51	99.50	16.18	18.88
4	3.37	3.89	84.13	97.31	21.23	24.61
5	2.80	4.74	55.95	94.88	17.65	30.00
6	4.84	5.74	80.59	95.61	30.51	36.28
7	5.92	6.66	84.62	95.10	37.37	42.09
8	4.75	7.43	59.38	92.82	29.97	46.95



(a) Variation in Real time Speedup



(b) Variation in User time Speedup

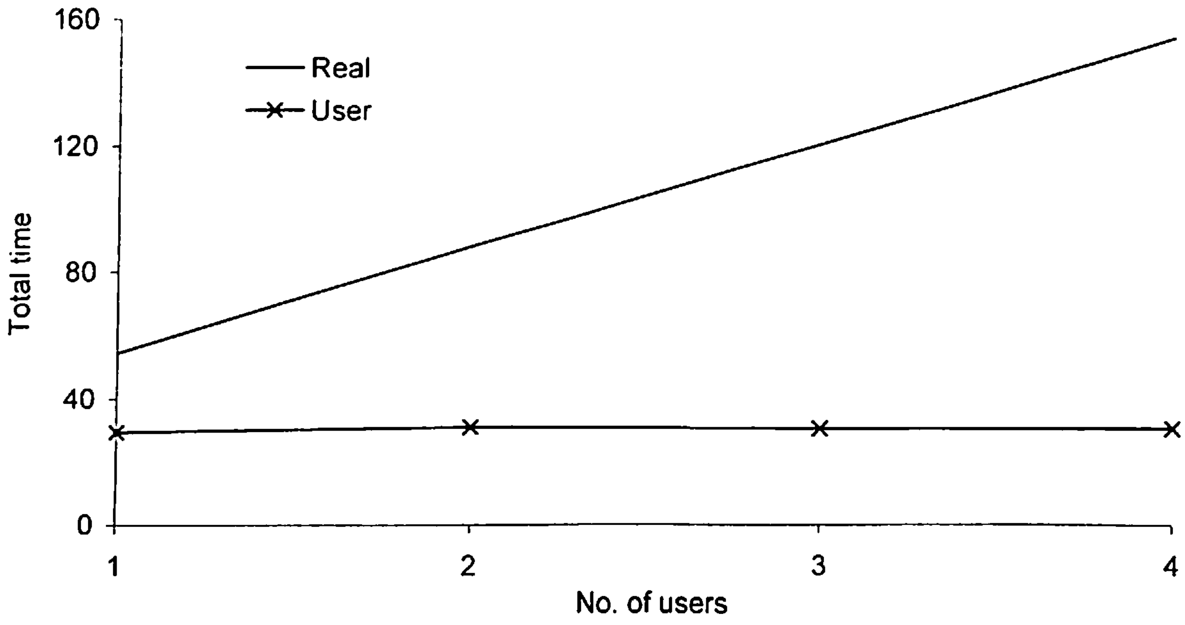
Fig. 3.15 Variation in Speedup for MIM solver for various data sets

In Fig. 3.15 (a) variation in Real time Speedup for various data sets under consideration is shown. It can be observed that this variation is very abrupt whereas User time Speedup variation for various data sets is smooth and constantly increasing (see Fig. 3.15 (b)). One important observation can be made that is, as the data size increases the corresponding User time Speedup for a particular number of processor also increases. This indicates that solver performance improves with increasing data size. The User time Speedup of 7.5 (approximately) was achieved at eight number of processors, which is very close to the Ideal Speedup. After observing Tables 3.15 to 3.20, it can be seen that the efficiency of the solver reduces by small amount with increase in number of processors. A speed of nearly 50 (approximately) millions of floating point operations per second was achieved at eight number of processors. In general, this method performed well as compared to the other two methods discussed earlier.

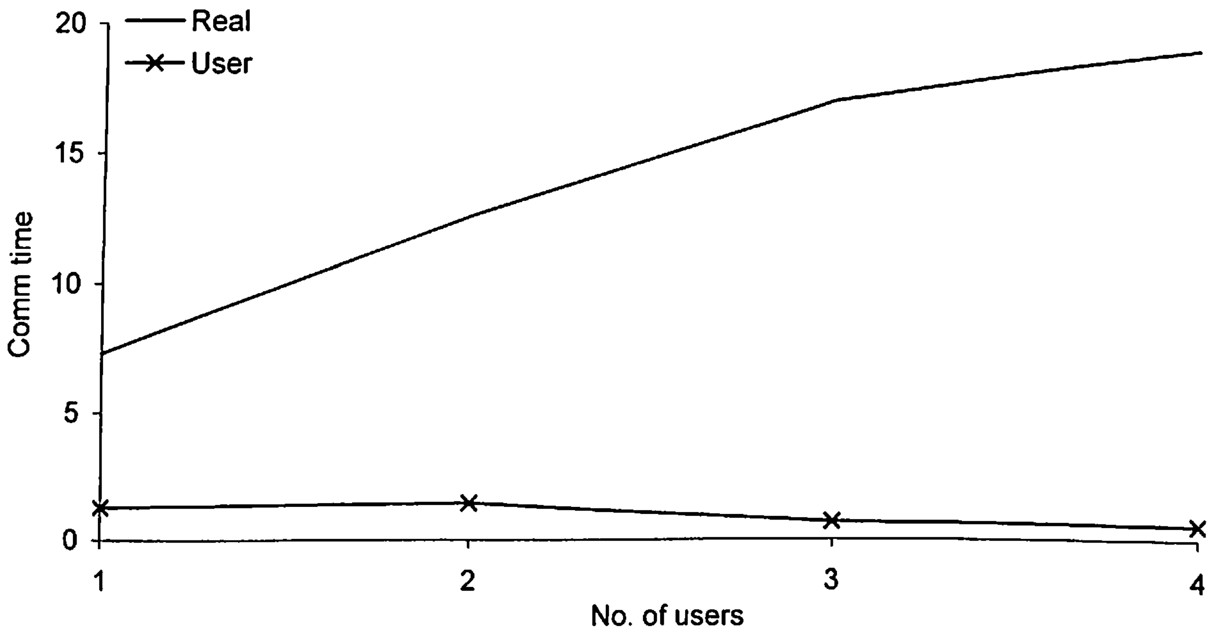
3.5 EFFECT OF USER ACTIVITIES ON COMPUTATIONAL TIME

From the computational time results presented in the above section, it can be observed that the variation in components of computational time measured in terms of Real time is very abrupt. It was also observed that computational time components measured in terms of Real time during the study were not same for the same code is executed several times for the same data. This may be because of activities carried out by several users connected to PARAM 10000. To observe this effect, a small study was carried out. In this study same data (set of 870 equations) was analyzed by different users at the same time with the same number of processors by Matrix Inversion Method parallel solver. For all these users, different components of computational time were measured with increasing number of users.

Fig. 3.16(a) shows that, as the number of users increases, the average of Total time (RT), measured by all the users using subroutine MPI_Wtime. also increases. It can also be observed that the average of Total time (UT), measured by all the users using developed subroutine (see section 3.1), remains almost constant and does not get affected by increase in number of users connected. This is quite obvious because the number of computations carried out by processors do not change with increase in number of users. But, the average of Total time (RT), measured by all the users, increases with increase in



(a) Typical variation in Total time with number of users



(b) Typical variation in Communication time with number of users

Fig. 3.16 Typical variation in Total time and Communication time with number of users

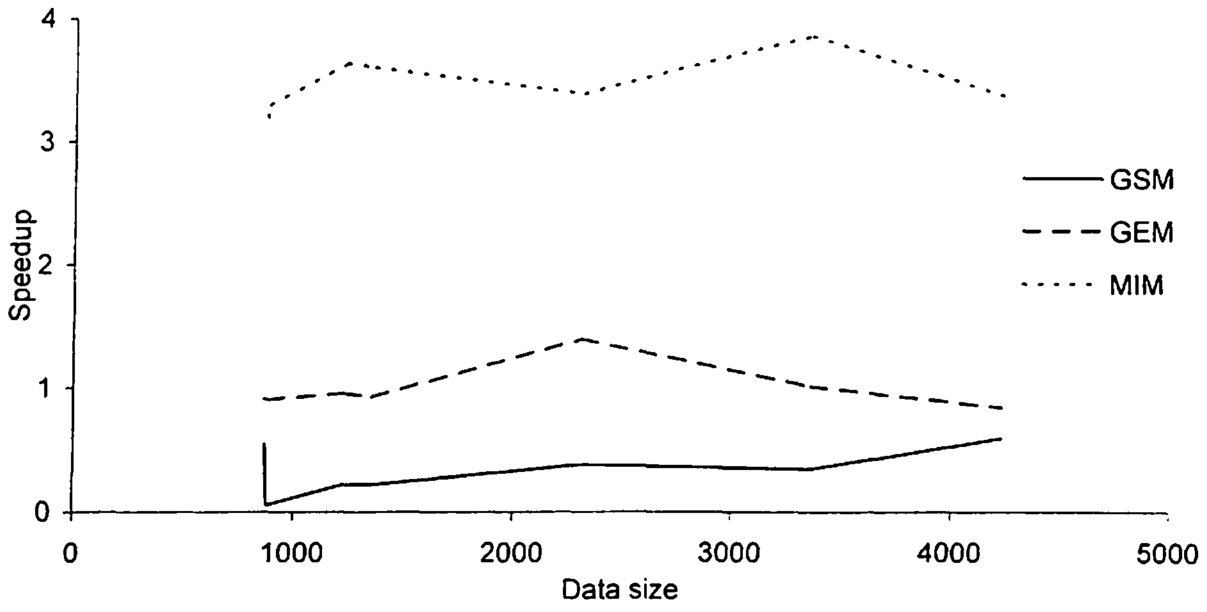
number of users because of increase in the computational load assigned to each of the processors.

From Fig. 3.16(b), it can be seen that as the number of users increases, the average of Communication time (RT), measured by all the users using subroutine MPI_Wtime, also increases. Whereas the average of Communication time (UT), measured by all the users using developed subroutine, remains almost constant. Very little variation (in milliseconds) may be observed, which is very small as compared to overall process that consumes several seconds and hence can be neglected.

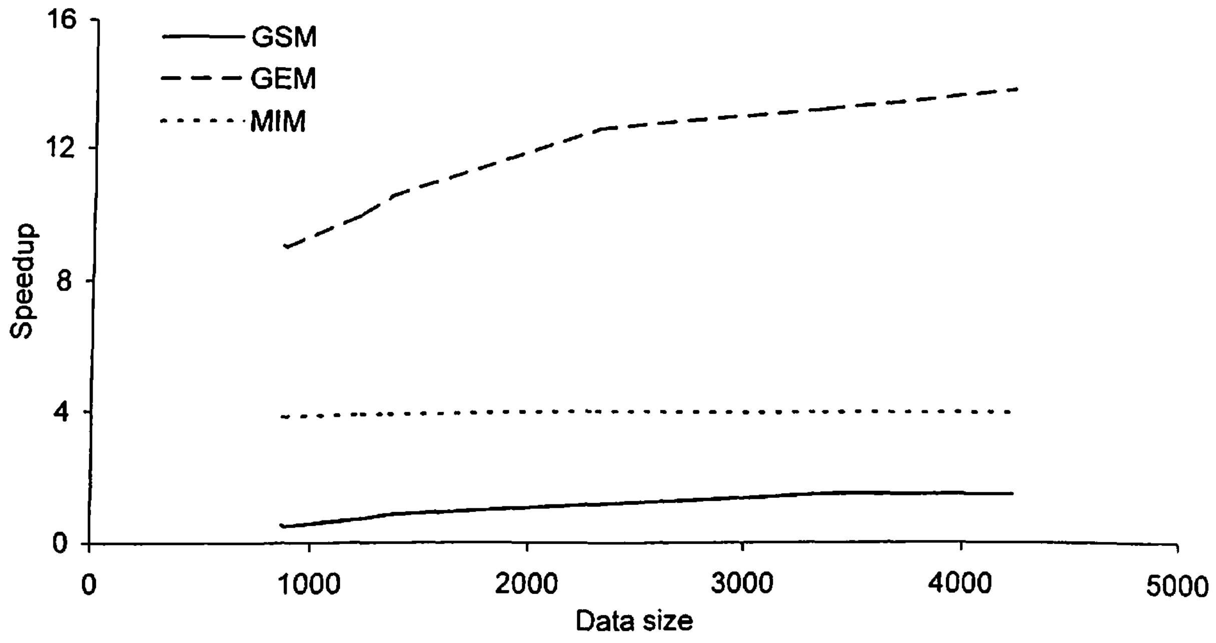
3.6 COMPARISON OF PARALLEL SOLVERS

Three parallel solvers discussed in previous sections performed in different manner and the comparison of these solvers in context of their performance should be scrutinized to select the best parallel solver for its implementation in finite element analysis. The main parameters for comparing parallel solvers are computational time and Speedup. In Gauss Seidel Method, Total time (RT and UT) increases with increase in number of processors. Real time Communication is mainly responsible for such increase in Total time. Since this method is Iterative so the number of iterations fully dependent on data type and initial guess. Therefore this method cannot be used for faster processing.

The numerical procedure of Gauss Elimination Method and Matrix Inversion Method is quite identical. In GEM, only lower triangle elements of matrix are converted into zero whereas in MIM except diagonal elements, all are converted into zero. This shows that GEM requires lesser computations as compared to the MIM. This is also reflected when Total time (UT) is observed for both the method for all data sets under consideration (see Tables 3.8 to 3.19). The only problem with GEM is that the sequential process of calculation of unknowns. In this process, only one processor remains active and rest all processors remain idle. Because of such sequential procedure, Total time (RT) increases with increase in number of processors. One can also observe that the Communication time (RT) is more for GEM as compared to MIM. One of the major drawbacks of GEM and MIM is that both fail to give solution, if any of the diagonal elements turns into zero.

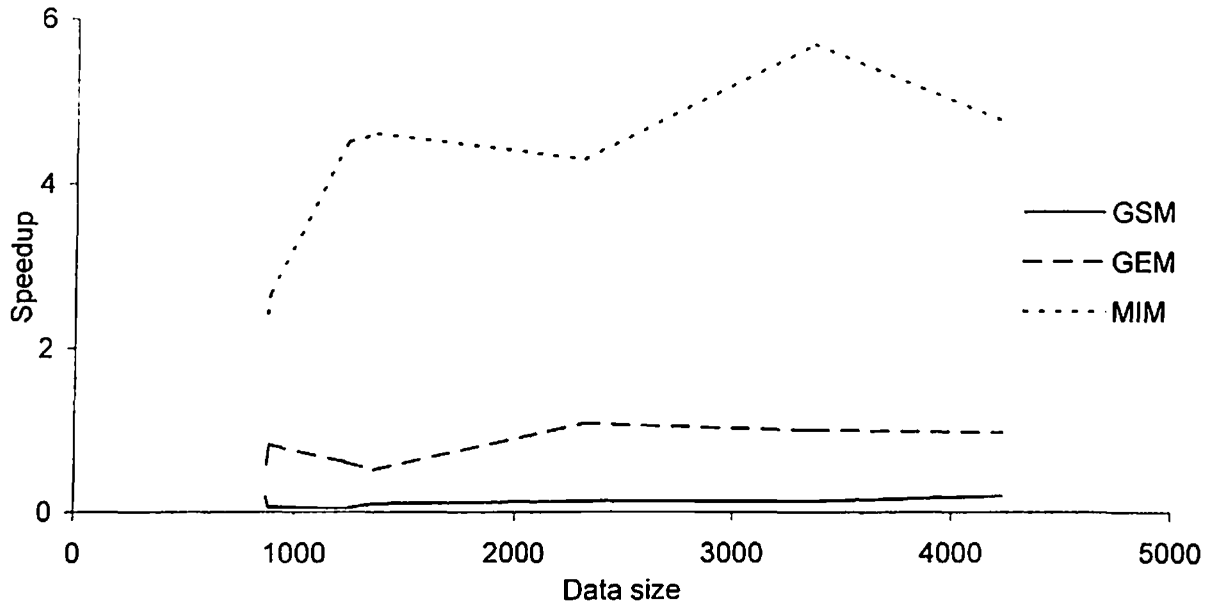


(a) Performance comparison of three solvers based on Real time Speedup

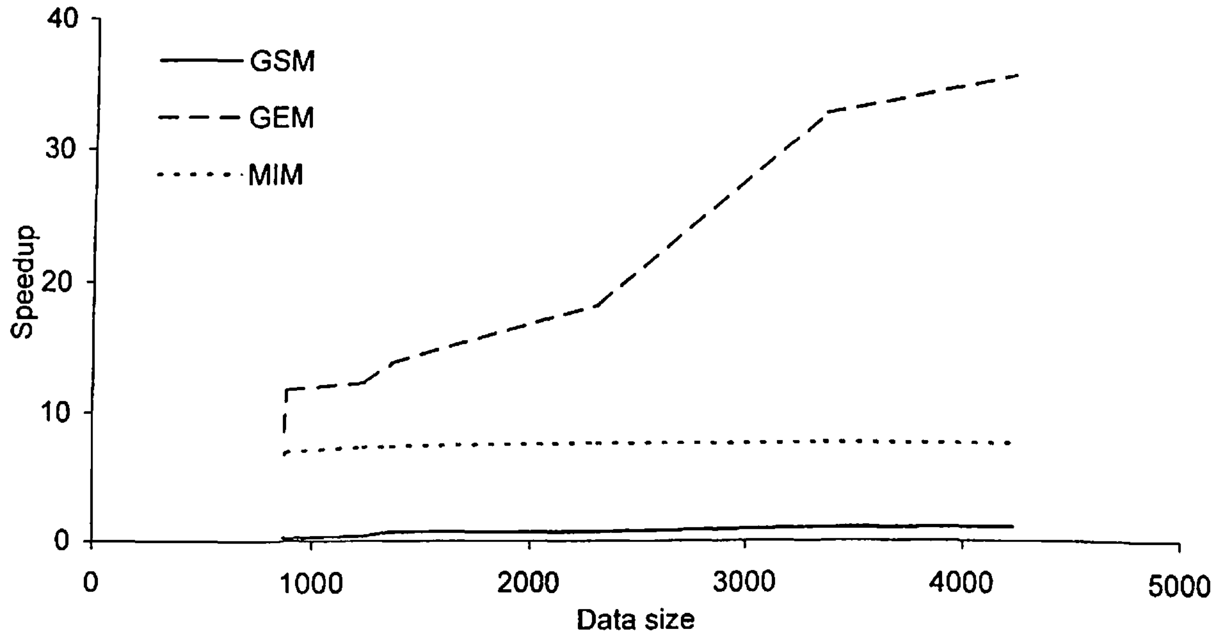


(b) Performance comparison of three solvers based on User time Speedup

Fig. 3.17 Performance comparison of three solvers for different data sizes when four processors are used



(a) Performance comparison of three solvers based on Real time Speedup



(b) Performance comparison of three solvers based on User time Speedup

Fig. 3.18 Performance comparison of three solvers for different data sizes when eight processors are used

Figure 3.17 and 3.18 show performance of three methods for different data sets when four and eight processors are used. Figure 3.17 (a) and 3.18 (a) show the performance of all these solvers based on Real time whereas Figure 3.17 (b) and 3.18 (b) show the performance based on User time. From Fig. 3.17 (a) and 3.18 (a) it can be observed that the performance of these solvers is uneven. User activities are mainly responsible for such uneven variation. It can be observed that Real time Speedup of MIM is highest for all data sets. It is followed by Real time Speedup of GEM. Similar variation can also be observed for all number of processors for all data sets under consideration. For User time Speedup, it is found that GEM shows highest Speedup (see Fig. 3.17 (b) and 3.18 (b)). It can also be observed for GEM that User time Speedup improves with increase in data size. It is followed by User time Speedup achieved by MIM. This Speedup remains constant irrespective of data size.

3.7 COMPARISON OF C AND FORTRAN77

For programming in MPI on parallel computers, C and FORTRAN77 (F77) are very commonly used as programming languages on supercomputer PARAM 10000. Therefore it is essential to study the performance of both the languages. From section 3.5 it is very clear that Matrix Inversion Method parallel solver is the most efficient solver among all the developed solvers. Therefore this solver is also developed in FORTRAN77 language. Three data sets containing 870, 1226 and 1722 linear equations taken from linear elastic finite element analysis (see Chapter 4) were solved with the help of this solver and the computational time variation is obtained. The same data sets were also analyzed using same solver developed in C language and the results are compared. According to section 3.4, users activities affect Real time; therefore User time is also measured and considered as a main parameter for comparison. Table 3.20 to 3.22 shows the computational time results obtained by both the solvers for the data sets under consideration. It contains the Total time (RT and UT) obtained by parallel solvers developed using C and FORTRAN77 programming languages.

From Tables 3.20 to 3.22, it can be observed that the Total time (RT and UT) reduces with increase in number of processors. Such variation is observed for both the solvers developed using C and FORTRAN77 languages. One can observe that, Total time (RT)

Table 3.20 Computational time variation and performance of C and FORTRAN77 codes for data set of size 870 × 870

(a) Computational time variation

No. of processors	Total (RT)			Total (UT)		
	C	F-77	F-77 / C	C	F-77	F-77 / C
1	203.74	312.77	1.54	202.57	311.38	1.54
2	111.30	171.70	1.54	105.58	165.84	1.57
3	83.57	118.78	1.42	71.81	111.25	1.55
4	62.93	92.24	1.47	54.12	83.77	1.55
5	93.07	139.08	1.49	44.28	69.40	1.57
6	86.18	132.45	1.54	38.38	60.30	1.57
7	76.36	113.70	1.49	33.34	51.38	1.54
8	73.99	101.30	1.37	29.32	45.68	1.56

(b) Performance of C code

No. of processors	Speedup		Efficiency		MFLOPS	
	Real	User	Real	User	Real	User
1	1.00	1.00	100.00	100.00	6.48	6.52
2	1.83	1.92	91.53	95.93	11.86	12.50
3	2.44	2.82	81.26	94.03	15.80	18.38
4	3.24	3.74	80.94	93.57	20.98	24.39
5	2.19	4.57	43.78	91.50	14.18	29.81
6	2.36	5.28	39.40	87.97	15.32	34.39
7	2.67	6.08	38.11	86.80	17.29	39.59
8	2.75	6.91	34.42	86.36	17.84	45.02

(c) Performance of FORTRAN77 code

No. of processors	Speedup		Efficiency		MFLOPS	
	Real	User	Real	User	Real	User
1	1.00	1.00	100.00	100.00	4.22	4.24
2	1.82	1.88	91.08	93.88	7.69	7.96
3	2.63	2.80	87.77	93.30	11.11	11.87
4	3.39	3.72	84.77	92.93	14.31	15.76
5	2.25	4.49	44.98	89.73	9.49	19.02
6	2.36	5.16	39.36	86.06	9.97	21.89
7	2.75	6.06	39.30	86.58	11.61	25.69
8	3.09	6.82	38.60	85.21	13.03	28.90

Table 3.21 Computational time variation and performance of C and FORTRAN77 codes for data set of size 1226 × 1226

(a) Computational time variation

No. of processors	Total (RT)			Total (UT)		
	C	F-77	F-77 / C	C	F-77	F-77 / C
1	593.21	793.84	1.34	589.80	789.03	1.34
2	535.73	406.31	0.76	297.37	401.38	1.35
3	500.38	524.83	1.05	202.97	270.47	1.33
4	336.48	219.44	0.65	154.38	205.21	1.33
5	287.62	290.34	1.01	126.57	168.10	1.33
6	265.88	178.08	0.67	108.55	137.72	1.27
7	233.18	243.42	1.04	92.19	123.10	1.34
8	217.44	155.59	0.72	81.41	107.03	1.31

(b) Performance of C code

No. of processors	Speedup		Efficiency		MFLOPS	
	Real	User	Real	User	Real	User
1	1.00	1.00	100.00	100.00	6.38	6.41
2	1.11	1.98	55.36	99.17	7.06	12.72
3	1.19	2.91	39.52	96.86	7.56	18.64
4	1.76	3.82	44.07	95.51	11.24	24.50
5	2.06	4.66	41.25	93.20	13.15	29.89
6	2.23	5.43	37.19	90.56	14.23	34.85
7	2.54	6.40	36.34	91.40	16.22	41.03
8	2.73	7.24	34.10	90.56	17.40	46.46

(c) Performance of FORTRAN77 code

No. of processors	Speedup		Efficiency		MFLOPS	
	Real	User	Real	User	Real	User
1	1.00	1.00	100.00	100.00	4.76	4.79
2	1.95	1.97	97.69	98.29	9.31	9.42
3	1.51	2.92	50.42	97.24	7.21	13.99
4	3.62	3.84	90.44	96.12	17.24	18.43
5	2.73	4.69	54.68	93.88	13.03	22.50
6	4.46	5.73	74.30	95.49	21.24	27.47
7	3.26	6.41	46.59	91.57	15.54	30.73
8	5.10	7.37	63.78	92.15	24.31	35.34

Table 3.22 Computational time variation and performance of C and FORTRAN77 codes for data set of size 1722 × 1722

(a) Computational time variation

No. of processors	Total (RT)			Total (UT)		
	C	F-77	F-77 / C	C	F-77	F-77 / C
1	1668.21	2652.37	1.59	1660.28	2460.32	1.48
2	827.53	1463.42	1.77	813.31	1451.27	1.78
3	566.68	965.51	1.70	547.75	855.86	1.56
4	434.30	708.08	1.63	414.77	647.16	1.56
5	384.58	866.44	2.25	342.44	530.40	1.55
6	344.83	806.95	2.34	286.45	445.00	1.55
7	306.18	453.79	1.48	247.44	378.54	1.53
8	307.26	422.55	1.38	219.90	338.11	1.54

(b) Performance of C code

No. of processors	Speedup		Efficiency		MFLOPS	
	Real	User	Real	User	Real	User
1	1.00	1.00	100.00	100.00	6.13	6.16
2	2.02	2.04	100.79	102.07	12.36	12.57
3	2.94	3.03	98.13	101.04	18.04	18.67
4	3.84	4.00	96.03	100.07	23.54	24.65
5	4.34	4.85	86.76	96.97	26.59	29.86
6	4.84	5.80	80.63	96.60	29.65	35.69
7	5.45	6.71	77.83	95.85	33.39	41.32
8	5.43	7.55	67.87	94.38	33.28	46.50

(c) Performance of FORTRAN77 code

No. of processors	Speedup		Efficiency		MFLOPS	
	Real	User	Real	User	Real	User
1	1.00	1.00	100.00	100.00	3.85	4.16
2	1.81	1.70	90.62	84.76	6.99	7.05
3	2.75	2.87	91.57	95.82	10.59	11.95
4	3.75	3.80	93.65	95.04	14.44	15.80
5	3.06	4.64	61.22	92.77	11.80	19.28
6	3.29	5.53	54.78	92.15	12.67	22.98
7	5.84	6.50	83.50	92.85	22.53	27.01
8	6.28	7.28	78.46	90.96	24.20	30.24

for C code is significantly less at all number of processors for all the data sets under consideration as compared to FORTRAN77 code. Similar behavior can also be observed when Total time measured in term of User time is considered. The range of ratio of computational time measured by FORTRAN77 code and C code was observed to be from 1.27 to 1.78 which clearly indicates that C code is faster than the FORTRAN77 code. This comparison is made on the basis of User time due to its independency with the users activities.

It can be observed that the performance of both the solvers is quite well. Both the solvers achieved a User time Speedup of 7.5 (approximately) for all data sets under consideration at eight number of processors. The reduction in User time efficiency is very slow with the increase in number of processors for both the solvers. Millions of floating point operations per second measured for both the solvers increases with the increase in number of processors. One can observe that MFLOPS measured by C code is on higher side as compared to the MFLOPS measured by FORTRAN77 code at every number of processors.

3.8 COMMUNICATION MECHANISMS

In the parallel computing technique, the Communication time plays an important role. The communication between the processors can be established by calling few readymade subroutines available in MPI library. These subroutines are based on two communication mechanisms i.e. Blocking Communication and Non-blocking Communication [3]. Blocking communication mechanism is used in all the parallel codes explained earlier. To study these mechanisms and their effect on communication time, both the communication mechanisms are incorporated in the Matrix Inversion Method parallel solver. The parallelization strategy for both the communication mechanisms was kept same. To incorporate both communication mechanisms, few subroutines MPI_SEND and MPI_RECV were changed to MPI_ISEND and MPI_IRECV. Solution of set of linear equation of size 870 was obtained by both the codes written in C language. The Communication time was measured in term of Real time and User time.

Table 3.23 shows values of Communication time (RT and UT) for data sets under consideration. It can be observed that as the number of processors increases, the

Communication time also increases correspondingly. It can also be observed from this table that both communication mechanisms have similar performance. The effect of communication mechanisms on Communication time is not significant. In the presented algorithm of matrix inversion process, the major part of communication was carried out by subroutine called MPI_BCAST. This may be one of the reasons of finding the same communication pattern by both the communication mechanisms. The study highlights that any of the communication mechanism can be adopted for data communication for the Matrix Inversion Method parallel solver.

Table 3.23 Comparison of two communication mechanisms

No. of processors	Communication (RT)		Communication (UT)	
	Blocking	Non-Blocking	Blocking	Non-Blocking
1	0.00	0.00	0.00	0.00
2	0.74	1.49	0.01	0.01
3	1.68	2.75	0.05	0.04
4	1.86	3.17	0.17	0.14
5	4.68	3.80	0.20	0.18
6	7.04	4.53	0.33	0.27
7	7.56	4.65	0.40	0.37
8	8.11	6.11	0.37	0.38

Table 3.24 Comparison of C and FORTRAN77 for two communication mechanisms

No. of processors	Communication (RT) in C		Communication (RT) in FORTRAN77	
	Blocking	Non-Blocking	Blocking	Non-Blocking
1	0.00	0.00	0.00	0.00
2	2.04	2.04	0.10	0.64
3	30.14	30.14	0.38	0.84
4	8.33	8.33	0.60	0.83
5	15.25	15.25	2.61	3.03
6	14.22	14.22	3.50	7.97
7	12.85	12.85	4.68	5.25
8	13.43	6.16	7.99	4.46

Both the communication mechanisms were also incorporated in FORTRAN77 code. A data set of size 1226 was once again analyzed using solvers developed in C and FORTRAN77 languages with both the communication mechanisms. Table 3.24 shows the Communication time (RT) obtained by both the codes. It can be observed that very little variation exists in Communication time obtained using both the mechanisms for both the languages. The Communication time is lower for FORTRAN77 solver as compared to the Communication time measured for C solver.

3.9 MODIFIED MATRIX INVERSION SOLVER

Now it is clear that the Matrix Inversion Method is the most suitable method among all the methods discussed. Therefore Matrix Inversion Method parallel solver was once again developed using C language with blocking type of communication. Modifications were incorporated in such a way in the previously developed Matrix Inversion Method parallel solver, so that it can give the inversion of the matrix generated exclusively in finite element analysis in less time. It was observed that, stiffness matrix developed in finite element analysis contains large number of zero elements. All the elements in lower and upper triangles were zero. In addition to this few elements in the bandwidth of the global stiffness matrix were also zero (see Fig. 3.19). One can observe that, major part of stiffness matrix is identical to an Identity matrix. Therefore, operations in this region can be skipped to save huge amount of computations.

In Matrix Inversion Method, it was observed that the bandwidth does not affect the number of computations if the computations at the elements having zero values are skipped. For the discretized finite element domain containing large number of elements, it is very difficult to control the bandwidth. It was also observed that, for problems of large bandwidth, the number of elements having zero value inside the bandwidth were more as compared to elements having non-zero value. Therefore by using Matrix Inversion Method, significant number of computations can be saved. This shows the suitability of the Matrix Inversion Method for solving stiffness equation in finite element analysis.

3.9.1 Parallel Implementation

Initially all processors were given their ranks (starting from zero to seven). After that the range of data to be handled by each processor was decided. If data distribution was not even, then the additional data was distributed to the processors with lower ranks. After proper data distribution among the processors, an Identity matrix $[I]$ of size $[A]$ was created by all processors. In the process of matrix inversion, column wise operations were

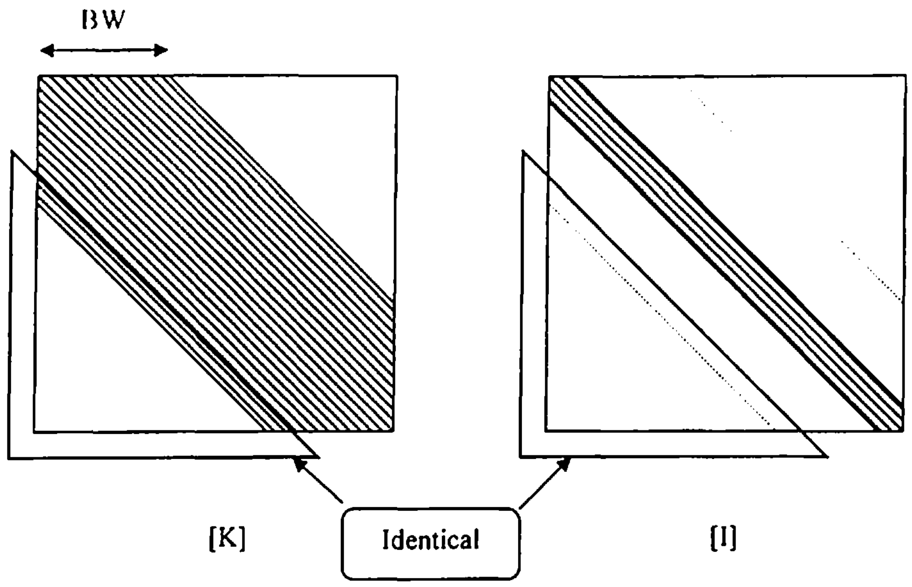


Fig. 3.19 Stiffness matrix and Identity matrix

```
Global  P          {Number of Processors}
        n          {Number of Equations}
        MyRank     {Rank of the Processor}
        Rank       {Rank of processor holding current row}
        start      {Flag indicating starting row number for each processor}
        end        {Flag indicating ending row number for each processor }
        i          {Variable indicating current row}
        [I]        {Matrix indicating inverse of matrix [A]}
for all Pi where 0 < i < P do
        Set start
        Set end
for i = 0 to n-1 step 1
        if diagonal of [A]_i = 1.0
                continue
        else
                Set diagonal element of [A]_i = 1.0
                Change elements of matrix [I]_i
        endif
        for all Pi where 0 < i < P do
                Find the Rank of current row
                If MyRank = Rank
                        Broadcast current row
                endif
        endfor
        for j = start to end step 1
                if [A]_ij ≠ 0.0
                        Change non-diagonal element of [A]_ij = 0.0
                        Change elements of matrix [I]_ij
                endif
        endfor
endfor
for i = start to end step 1
        Compute {x}_i
endfor
for all Pi where 0 < i < P do
        Broadcast {x}_i to All Processor
endfor
```

Fig. 3.20 Parallel algorithm for Modified Matrix Inversion Method

carried out. Every non-diagonal element of matrix $[A]$ was converted to zero and every diagonal element of matrix $[A]$ was made unity. While doing this, the operations were skipped at locations where non-diagonal elements have zero value and diagonal element have unit value. This helped in reducing the number of computations.

Whatever operations were carried out on matrix $[A]$, same operations were also carried out on matrix $[I]$ simultaneously. Each processor operated only those rows, which were designated to it to achieve less computational time. After finding the inverse of matrix $[A]$, the unknown vector $\{X\}$ was calculated by multiplying $[A]^{-1}$ with $\{B\}$. At this juncture, each processor was having elements of vector $\{X\}$ those belong to its share. Then each processor broadcasted these elements of vector $\{X\}$ to the all other processors so that every processor should have complete vector $\{X\}$. Figure 3.20 shows the parallel algorithm for Modified Matrix Inversion Method.

3.9.2 Computational Time Results

Based on the above-discussed algorithm, a parallel solver is developed. Data of size 1226 × 1226 generated from finite element analysis is solved by this developed solver. Computational time results were generated and compared with the results of the original solver developed earlier in the Table 3.25. One can observe from Table 3.25(a) that Total time (RT and UT) reduces drastically when modified solver was used. One can also observe that for single processor nearly 67% of User time as well as Real time can be saved by using the modified solver. As the number of processors increases the percentage saving in both time components reduces. The percentage saving in Total time (RT and UT) reduces up to 29% and 43% respectively when eight processors were employed.

Sudden reduction in Total time (RT) can be observed from one processor to four processors. It can also be observed that after four processors, reduction in Total time (RT) is gradual but insignificant. It can also be observed that variation of reduction in percentage saving in User time with increase in number of processor is continuous, whereas in case of Real time the variation is abrupt (sudden fall of percentage saving at four processors). The user activities are mainly responsible for such variations.

Table 3.25 Comparison of Modified and Original solver based on time results and Speedup for data size 1226 × 1226

(a) Comparison based on time results

No. of processors	Modified Solver		Original Solver		% Saving	
	Total (RT)	Total (UT)	Total (RT)	Total (UT)	Total (RT)	Total (UT)
1	189.86	188.51	578.16	575.42	67.16	67.24
2	173.76	109.30	299.72	294.14	42.03	62.84
3	136.06	90.94	208.52	200.37	34.75	54.61
4	128.05	76.19	163.36	153.00	21.61	50.20
5	110.73	65.25	155.09	127.17	28.60	48.69
6	105.03	56.98	141.84	108.41	25.95	47.44
7	92.35	50.65	125.84	92.24	26.61	45.09
8	94.88	47.11	134.43	82.95	29.42	43.21

(b) Comparison based Speedup

No. of processors	Modified Solver		Original Solver	
	Real	User	Real	User
1	1.00	1.00	1.00	1.00
2	1.09	1.72	1.93	1.96
3	1.40	2.07	2.77	2.87
4	1.48	2.47	3.54	3.76
5	1.71	2.89	3.73	4.52
6	1.81	3.31	4.08	5.31
7	2.06	3.72	4.59	6.24
8	2.00	4.00	4.30	6.94

From Table 3.25 (b), it can be observed that the modified solver recorded less Speedup as compared to the original solver. Maximum User time Speedup of 4.0 can be observed at eight number of processors for modified solver, whereas maximum User time Speedup of 6.94 can be observed at eight number of processors for original solver. One important observation can be made that at eight number of processors, User time saving is nearly 43%, which is very significant even if Speedup is less.

Table 3.26 Computational time variation and performance of MMIM solver for data set of size 870 × 870

(a) Computational time variation

No. of processors	Total		Comm		Cal	
	Real	User	Real	User	Real	User
1	74.43	65.43	0.00	0.00	74.43	65.43
2	72.63	37.82	1.27	0.27	71.36	37.55
3	57.65	31.97	2.21	0.42	55.44	31.55
4	53.64	26.71	2.50	0.47	51.14	26.24
5	74.25	24.63	8.01	1.60	66.24	23.03
6	91.93	24.65	17.11	3.04	74.82	21.61
7	89.60	22.04	12.32	0.80	77.28	21.24
8	82.86	20.25	16.30	2.19	66.56	18.06

(b) Performance

No. of processors	Speedup		Efficiency		MFLOPS	
	Real	User	Real	User	Real	User
1	1.00	1.00	100.00	100.00	17.74	20.17
2	1.02	1.73	51.24	86.50	18.17	34.90
3	1.29	2.05	43.04	68.22	22.90	41.29
4	1.39	2.45	34.69	61.24	24.61	49.42
5	1.00	2.66	20.05	53.13	17.78	53.59
6	0.81	2.65	13.49	44.24	14.36	53.55
7	0.83	2.97	11.87	42.41	14.73	59.89
8	0.90	3.23	11.23	40.39	15.93	65.19

Table 3.27 Computational time variation and performance of MMIM solver for data set of size 882 × 882

(a) Computational time variation

No. of processors	Total		Comm		Cal	
	Real	User	Real	User	Real	User
1	137.81	119.31	0.00	0.00	137.81	119.31
2	90.31	74.64	1.25	0.26	89.06	74.38
3	71.77	50.89	2.06	0.38	69.71	50.51
4	60.20	37.06	3.08	0.44	57.12	36.62
5	74.76	30.39	9.38	0.89	65.38	29.50
6	92.44	25.33	17.23	2.11	75.21	23.22
7	93.41	20.10	12.96	0.83	80.45	19.27
8	85.42	17.33	12.19	0.89	73.23	16.44

(b) Performance

No. of processors	Speedup		Efficiency		MFLOPS	
	Real	User	Real	User	Real	User
1	1.00	1.00	100.00	100.00	9.98	11.53
2	1.53	1.60	76.30	79.92	15.23	18.43
3	1.92	2.34	64.01	78.15	19.16	27.03
4	2.29	3.22	57.23	80.48	22.85	37.11
5	1.84	3.93	36.87	78.52	18.40	45.26
6	1.49	4.71	24.85	78.50	14.88	54.30
7	1.48	5.94	21.08	84.80	14.72	68.43
8	1.61	6.88	20.17	86.06	16.10	79.36

Table 3.28 Computational time variation and performance of MMIM solver for data set of size 1352 × 1352

(a) Computational time variation

No. of processors	Total		Comm		Cal	
	Real	User	Real	User	Real	User
1	485.53	427.38	0.00	0.00	485.53	427.38
2	325.57	277.72	2.72	0.54	322.85	277.18
3	315.42	199.85	11.84	1.04	303.58	198.81
4	202.45	142.68	5.92	0.97	196.52	141.71
5	280.53	114.40	29.68	2.24	250.84	112.16
6	278.77	92.14	37.50	2.68	241.27	89.46
7	282.56	75.55	43.06	2.22	239.50	73.33
8	257.83	62.29	39.15	3.23	218.68	59.06

(b) Performance

No. of processors	Speedup		Efficiency		MFLOPS	
	Real	User	Real	User	Real	User
1	1.00	1.00	100.00	100.00	10.19	11.58
2	1.49	1.54	74.57	76.94	15.20	17.82
3	1.54	2.14	51.31	71.28	15.69	24.77
4	2.40	3.00	59.96	74.88	24.45	34.69
5	1.73	3.74	34.62	74.72	17.65	43.27
6	1.74	4.64	29.03	77.31	17.76	53.72
7	1.72	5.66	24.55	80.81	17.52	65.52
8	1.88	6.86	23.54	85.76	19.20	79.47

Table 3.29 Computational time variation and performance of MMIM solver for data set of size 2312 × 2312

(a) Computational time variation

No. of processors	Total		Comm		Cal	
	Real	User	Real	User	Real	User
1	2099.93	2091.61	0.00	0.00	2099.93	2091.61
2	1447.35	1422.24	7.65	1.46	1439.70	1420.78
3	1122.43	980.66	16.13	3.90	1106.30	976.76
4	941.80	733.01	21.57	4.23	920.23	728.78
5	736.37	576.02	42.51	4.02	693.86	572.00
6	1105.94	482.83	82.78	6.55	1023.16	476.28
7	1000.86	404.33	94.01	6.31	906.85	398.02
8	947.08	341.62	86.42	9.87	860.66	331.75

(b) Performance

No. of processors	Speedup		Efficiency		MFLOPS	
	Real	User	Real	User	Real	User
1	1.00	1.00	100.00	100.00	11.78	11.83
2	1.45	1.47	72.54	73.53	17.09	17.39
3	1.87	2.13	62.36	71.10	22.04	25.23
4	2.23	2.85	55.74	71.34	26.27	33.75
5	2.85	3.63	57.03	72.62	33.59	42.95
6	1.90	4.33	31.65	72.20	22.37	51.24
7	2.10	5.17	29.97	73.90	24.72	61.18
8	2.22	6.12	27.72	76.53	26.12	72.41

Table 3.30 Computational time variation and performance of MMIM solver for data set of size 3362 × 3362

(a) Computational time variation

No. of processors	Total		Comm		Cal	
	Real	User	Real	User	Real	User
1	7608.38	6450.98	0.00	0.00	7608.38	6450.98
2	6499.82	4741.81	69.59	3.57	6430.23	4738.24
3	3412.81	3113.75	61.64	17.36	3351.17	3096.39
4	2659.27	2313.98	65.92	21.91	2593.35	2292.07
5	2290.78	1830.45	103.87	6.10	2186.91	1824.35
6	1936.09	1496.80	81.48	11.57	1854.61	1485.23
7	1734.28	1253.43	90.09	12.81	1644.19	1240.62
8	1659.43	1063.37	97.60	16.50	1561.83	1046.87

(b) Performance

No. of processors	Speedup		Efficiency		MFLOPS	
	Real	User	Real	User	Real	User
1	1.00	1.00	100.00	100.00	10.00	11.79
2	1.17	1.36	58.53	68.02	11.70	16.04
3	2.23	2.07	74.31	69.06	22.28	24.42
4	2.86	2.79	71.53	69.70	28.60	32.86
5	3.32	3.52	66.43	70.49	33.20	41.55
6	3.93	4.31	65.50	71.83	39.28	50.81
7	4.39	5.15	62.67	73.52	43.85	60.67
8	4.58	6.07	57.31	75.83	45.83	71.51

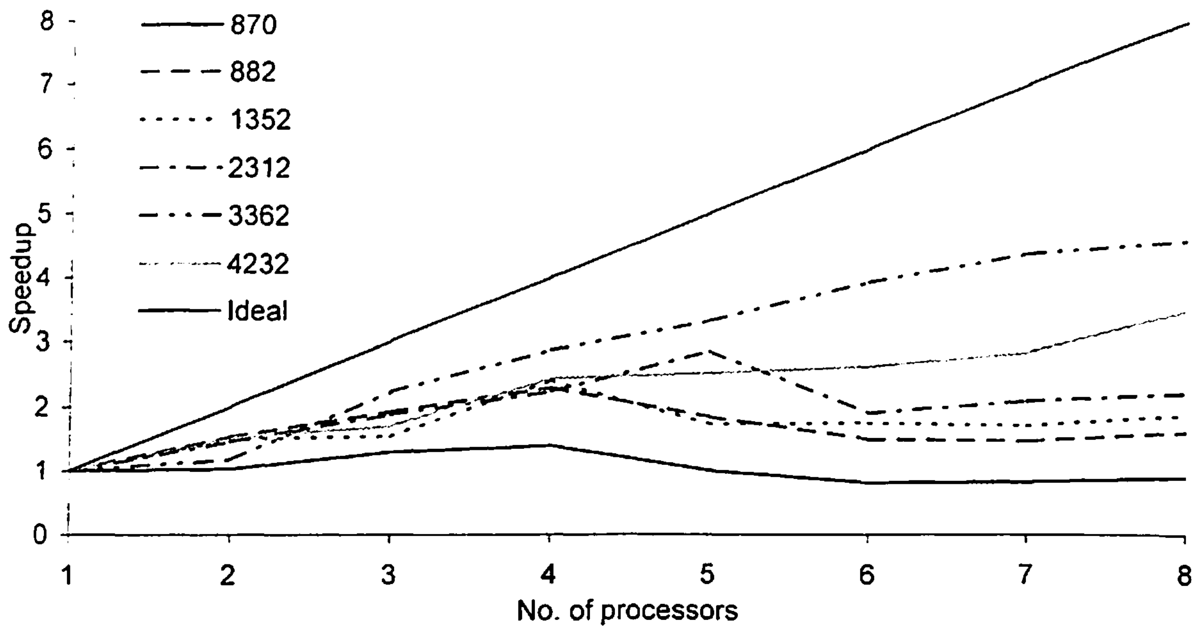
Table 3.31 Computational time variation and performance of MMIM solver for data set of size 4232 × 4232

(a) Computational time variation

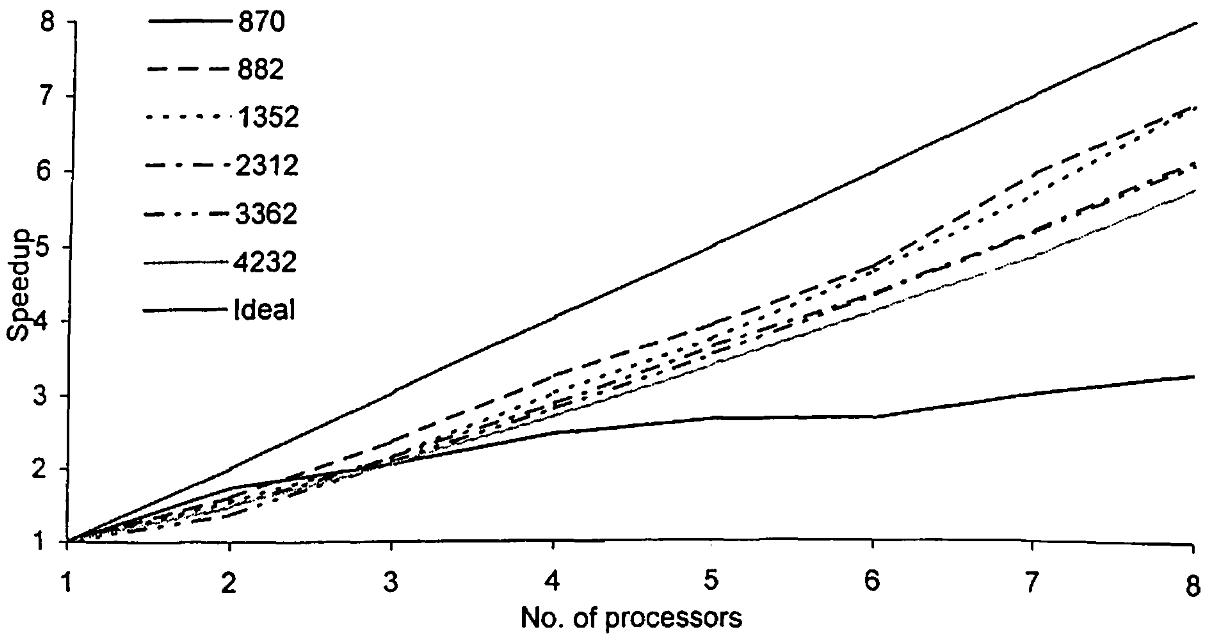
No. of processors	Total		Comm		Cal	
	Real	User	Real	User	Real	User
1	14695.15	12883.19	0.00	0.00	14695.15	12883.19
2	9716.18	8750.64	34.96	8.71	9681.22	8741.93
3	8676.87	6282.33	89.55	19.88	8587.33	6262.45
4	6042.83	4796.81	63.98	13.27	5978.85	4783.54
5	5839.82	3807.58	185.27	16.33	5654.54	3791.25
6	5624.08	3149.25	181.61	19.28	5442.47	3129.97
7	5182.14	2669.59	253.45	32.05	4928.69	2637.54
8	4197.98	2246.67	197.73	30.96	4000.24	2215.71

(b) Performance

No. of processors	Speedup		Efficiency		MFLOPS	
	Real	User	Real	User	Real	User
1	1.00	1.00	100.00	100.00	10.32	11.77
2	1.51	1.47	75.62	73.61	15.61	17.33
3	1.69	2.05	56.45	68.36	17.48	24.14
4	2.43	2.69	60.80	67.14	25.10	31.62
5	2.52	3.38	50.33	67.67	25.97	39.83
6	2.61	4.09	43.55	68.18	26.97	48.16
7	2.84	4.83	40.51	68.94	29.27	56.81
8	3.50	5.73	43.76	71.68	36.13	67.50



(a) Variation in User time Speedup



(b) Variation in User time Speedup

Fig. 3.21 Variation in Speedup for MMIM solver for various data sets

3.9.3 Solver Performance

To study the performance of the modified solver, several data sets taken from linear and non-linear finite element analysis problems were analyzed (see Chapter 4 and Chapter 5). Tables 3.26 to 3.31 show the different components of computational time measured in terms of Real time as well as User time. One can observe from these tables that Total time (RT and UT) reduces with increase in number of processors. Modified solver take significantly less time at every number of processors as compared to the original matrix inversion parallel solver (see Table 3.26 to 3.31). Figure 3.21 (a) shows Real time Speedup with number of processors for different data sets of various sizes. It can be observed that maximum Real time Speedup of 5 (approximately) was obtained. User time Speedup variation with number of processors for different data sets is shown in Fig. 3.21 (b). It can be observed that User time Speedup remains close to Ideal Speedup all nearly all data sets of various sizes.

3.10 SUMMARY

In this chapter, implementation of parallelization techniques in finite element analysis is presented. Initially the chapter discusses various components of computational time and the techniques to measure them by implementing timers in the computer codes. It also highlights the necessity of parallel computing technique in finite element analysis. It shows that the process of solving linear equations generated in finite element analysis requires major part of computational time. Hence parallel solver is necessary to reduce the computational time in finite element analysis.

The chapter also presents three different parallel solvers developed using three mathematical techniques namely Gauss-Seidel Method, Gauss Elimination Method and Matrix Inversion Method, ported on a platform of supercomputer PARAM 10000. It has been found that Matrix Inversion Method is the most suitable method. The effect of users activities on computational time is also presented and it was found that user activities significantly affect the computational time components measured in terms of Real time whereas components of computational time measured in terms of User time insignificantly affected by user activities.

The comparison of C and FORTRAN77 languages in parallel solvers is also presented in this chapter. It was found that code written in FORTRAN77 language is slower than the similar code developed in C language. Two communication mechanisms namely Blocking and Non-Blocking are also compared and it was found that both communication mechanisms are equally effective for establishing the communication between the processors.

At the end in this chapter, Modified Matrix Inversion Method parallel solver is presented for analyzing linear equations generated in finite element analysis exclusively. Comparison of Modified Matrix Inversion Method parallel solver with the original Matrix Inversion Method parallel solver is discussed. It was found that the modified solver is significantly faster than the original solver. The suitability of the Matrix Inversion Method for finite element analysis is also discussed in this chapter.

CHAPTER 4

ANALYSIS OF ANCHORAGE ZONE

4.1 INTRODUCTION

This chapter presents a detailed finite element analysis of anchorage zone in prestressed post-tensioned concrete beam. Initially basics of two-dimensional linear elastic finite element analysis are discussed in brief. Using constant strain triangular elements and parallel solver discussed in section 3.7, a finite element computer code is developed on the platform of a supercomputer PARAM10000. This code is employed for the analysis of anchorage zone stresses. The analysis of anchorage zone is carried out in two parts. In the beginning, stress distribution in anchorage zone is studied when concentric prestressing forces are applied. Distribution of transverse stresses, longitudinal stresses and shear stresses are obtained for different values of Poisson's ratio (0.0-0.3) and ratio of loaded area and cross-section area of the beam (0.1-0.6). Effect of Poisson's ratio on transverse tensile stress and bursting tensile force is studied. Effect of eccentricity of prestressing forces on transverse tensile stress and bursting tensile force is also studied. An expression to compute magnitude of bursting tensile force is developed and also compared with the existing results in the literature. Since the analysis was carried out on supercomputer PARAM 10000, so variation different components of computational time is also obtained, presented and discussed. The performance of the developed code is also studied by computing Speedup and Efficiency obtained on supercomputer PARAM 10000.

4.2 PRESTRESSED POST-TENSIONED CONCRETE BEAM

The anchorage zone is defined as the end portion of prestressed post-tensioned concrete beam starting from the loaded face to the section at a distance of d (depth of the beam) measured along the axis of the beam. This zone can be subdivided into three zones namely local zone, bursting zone and spalling zone (See Fig. 4.1). Local zone is the zone that is located just ahead of the loaded plate along the axis of loading. In this zone the concrete is subjected to large compressive stresses. The local zone is instantaneously followed by the bursting zone. In this zone the concrete is subjected to higher tensile stresses, which may cause busting of concrete mainly along the axis of loading in this region. Generally, special reinforcement is provided in this zone to resist the bursting tensile force caused by these tensile stresses. Spalling zone is found near the free corners of the beam where also tensile stresses are developed but of lower magnitudes as compared to the stresses developed in the bursting zone. Magnitude of highest tensile stress in the spalling zone is usually smaller than the permissible tensile stress in the concrete; therefore no reinforcement is specially needed to resist these stresses.

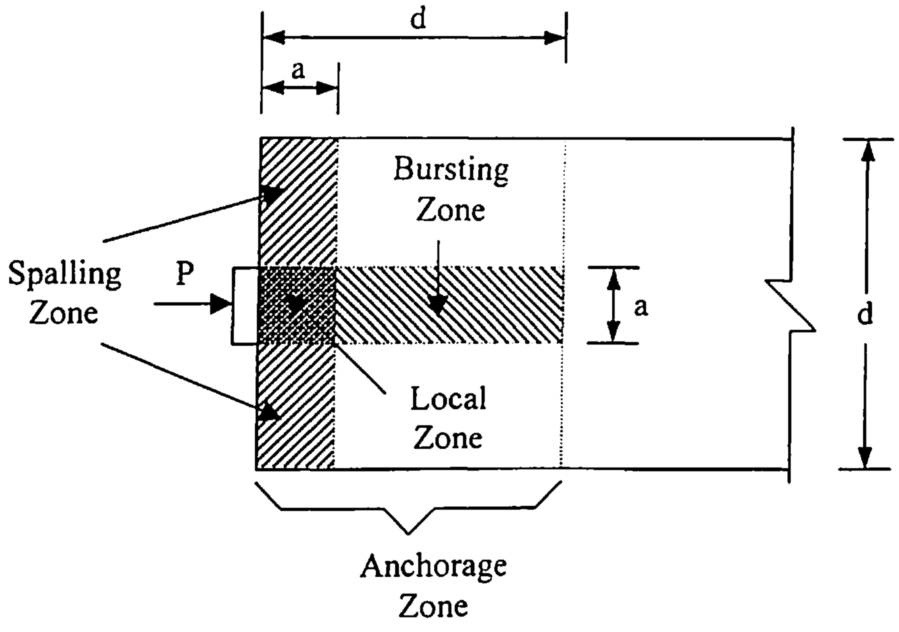


Fig. 4.1 Anchorage zone in prestressed post-tensioned concrete beam

4.3 REVIEW OF FINITE ELEMENT PROCEDURE

It is already seen that [32, 33] the stress variation in the anchorage zone is very complex, so to study this stress development, finite element method is used here. Following text covers the basics of linear elastic finite element analysis.

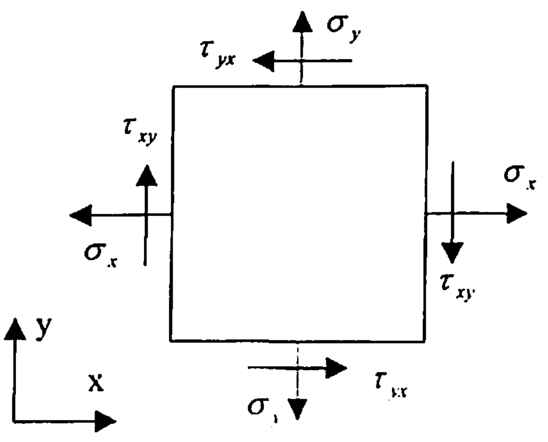


Fig. 4.2 Two-dimensional state of stress under equilibrium conditions

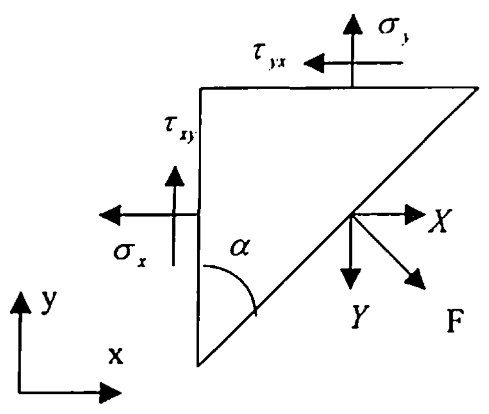


Fig. 4.3 Forces acting on two-dimensional element on the boundary

4.3.1 Basic Equations of Structural Mechanics

Consider a deformable body in the state of equilibrium subjected to external forces and internal forces developed due to small deformations (see Fig. 4.2). It is assumed that the stresses are acting in their positive directions (as shown in the Fig. 4.2) It is also assumed that the stress variation is uniform throughout the body. Let X and Y denote body force components per unit volume of body along x and y directions. Solving forces in x and y directions we get

$$\begin{aligned} &\frac{\partial \sigma_x}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + X = 0 \\ &\frac{\partial \tau_{xy}}{\partial y} + \frac{\partial \sigma_y}{\partial y} + Y = 0 \end{aligned} \tag{4.1}$$

Taking moment about z axis and $\sum M_z = 0$, we get

$$\tau_{xy} = \tau_{yx} \tag{4.2}$$

Equations 4.1 and 4.2 are equilibrium equations for two-dimensional stress distributions. These equations must be satisfied at all the points throughout the volume of the body. At the boundary, stress components must be in equilibrium with the external forces. From Fig. 4.3, solving forces in x and y directions we get

$$
\begin{aligned}
X &= l\sigma_x + m\tau_{yx} \\
Y &= l\tau_{xy} + m\sigma_y
\end{aligned}
\tag{4.3}
$$

where $l = \cos\alpha$, and $m = \sin\alpha$

Equation 4.3 represents the force boundary conditions in two-dimensional body in equilibrium condition.

The displacements at any given point in a deformable body can be described by its components u and v taken parallel to the cartesian coordinate axes x and y. The strains in the deformed body can be represented as partial derivatives of the displacement components u and v as follows

$$
\varepsilon_x = \frac{\partial u}{\partial x} \qquad \varepsilon_y = \frac{\partial v}{\partial y} \qquad \gamma_{xy} = \frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}
\tag{4.4}
$$

Equation 4.4 can be expressed in a matrix form as

$$
\begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix} = \begin{bmatrix} \partial/\partial x & 0 & 0 \\ 0 & \partial/\partial y & 0 \\ \partial/\partial y & \partial/\partial x & 0 \end{bmatrix} \begin{Bmatrix} u \\ v \end{Bmatrix}
\tag{4.5}
$$

It is assumed that the material obeys Hook's law so stress components can be expressed as linear functions of strain components. For a linear elastic, isotropic and homogeneous material, the stress-strain relationship is given as

$$
\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{Bmatrix} = \frac{E}{(1+v)(1-2v)} \begin{bmatrix} (1-v) & v & 0 \\ v & (1-v) & 0 \\ 0 & 0 & (1-2v)/2 \end{bmatrix} \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix}
\tag{4.6}
$$

where E is Youngs Modulus and v is the Poisson's Ratio.

Expression 4.6 can be rewritten in a reduced form as

$$\{\sigma\} = [D]\{\varepsilon\} \tag{4.7}$$

where $[D]$ is called as the material constitutive matrix.

4.3.2 Classification of Two-Dimensional Problems

Structural analysis using finite element analysis can be carried out by idealizing three-dimensional problems as two-dimensional problems based on their different geometric and stress conditions. These two-dimensional problems are classified as follows;

4.3.2.1 Plane Stress Problem

In plane stress problem, one of the dimensions is very small as compared to the other two dimensions normal to it. Figure 4.4 shows the example of plane stress condition where a thin plate is subjected to external loading. The thickness of plate is very small as compared to the other two dimensions. In such cases, stress components σ_z, τ_{xz} and τ_{yz} are zero that means there is no stress variation across the thickness of the plate. Here σ_x, σ_y and τ_{xy} are functions of x and y only. In plane stress problems, expression 4.6 can be expressed as

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{Bmatrix} = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & (1-\nu)/2 \end{bmatrix} \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix} \tag{4.8}$$

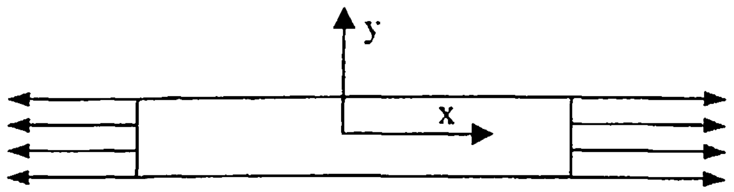


Fig. 4.4 Plane stress example: thin plate subjected to external loading

4.3.2.2 Plane Strain Problem

Plane strain problem includes a long deformable body whose geometry and loading is almost constant in longitudinal direction. Figure 4.5 shows an example of plane strain problem where a cross section of dam is subjected to external water pressure. It is assumed that rigid body movement in z direction does not cause strain in z direction. Therefore strain components ε_z, γ_{xz} and γ_{zy} are zero that indicates displacement u and v are the functions of x and y only and independent of z. In plane strain problems, expression 4.6 can be expressed as

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{Bmatrix} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} (1-\nu) & \nu & 0 \\ \nu & (1-\nu) & 0 \\ 0 & 0 & (1-2\nu)/2 \end{bmatrix} \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix} \tag{4.9}$$

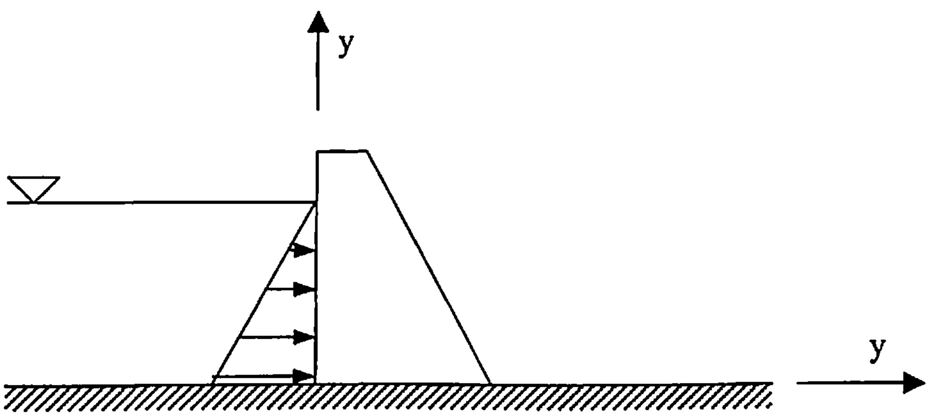


Fig. 4.5 Plane strain example: cross section of dam subjected to external water pressure

4.3.2.3 Axisymmetric Problem

When a solid is subjected to axially symmetric loadings, the deformation process could be idealized as axisymmetric. Solid circular column subjected to external pressure is an example of axisymmetric problem as shown in Fig. 4.6. In axisymmetric stress condition, an additional stress component σ_θ along the circumferential direction is included. The strain displacement relationship is expressed as

$$\varepsilon_r = \frac{\partial u}{\partial r} \qquad \varepsilon_z = \frac{\partial v}{\partial z} \qquad \varepsilon_\theta = \frac{u}{r} \qquad \gamma_{rz} = \frac{\partial v}{\partial r} + \frac{\partial u}{\partial z} \tag{4.10}$$

The stress-strain relationship in axisymmetric problems can be expressed as

$$\begin{Bmatrix} \sigma_r \\ \sigma_z \\ \sigma_\theta \\ \tau_{rz} \end{Bmatrix} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} (1-\nu) & \nu & \nu & 0 \\ \nu & (1-\nu) & \nu & 0 \\ \nu & \nu & (1-\nu) & 0 \\ 0 & 0 & 0 & (1-2\nu)/2 \end{bmatrix} \begin{Bmatrix} \varepsilon_r \\ \varepsilon_z \\ \varepsilon_\theta \\ \gamma_{rz} \end{Bmatrix} \tag{4.11}$$

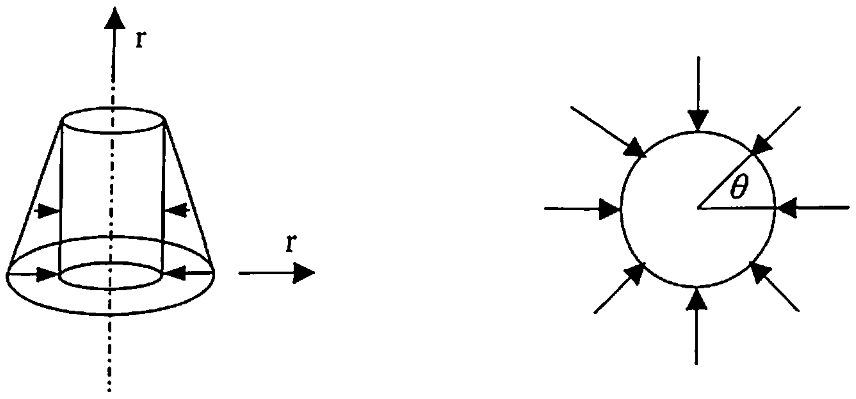


Fig. 4.6 Axisymmetric example: circular column subjected to pressure

4.3.3 Element Stiffness Matrix

By applying principle of virtual displacement, the equilibrium equations can be rewritten in the reduced form as

$$[\mathbf{k}]\{\mathbf{d}\} = \{\mathbf{Q}\} \tag{4.12}$$

where $\{\mathbf{d}\}$ and $\{\mathbf{Q}\}$ are nodal displacement and nodal load vectors. Matrix $[\mathbf{k}]$ is called as stiffness matrix of element and can be written in following form

$$[\mathbf{k}] = \iiint_v [\mathbf{B}]^T [\mathbf{D}][\mathbf{B}] dV \tag{4.13}$$

where $[\mathbf{B}]$ is called as the strain displacement matrix that depends on type of element and degree of displacement at nodes.

4.3.4 Constant Strain Triangle (CST) Element

As the name indicates, it is a triangular element in which the strain distribution is uniform throughout the element. It is one of the basic element used in finite element analysis (see Fig. 4.7). The strain displacement matrix for CST can be derived as

$$[\mathbf{B}] = \begin{bmatrix} X_1 & 0 & X_2 & 0 & X_3 & 0 \\ 0 & Y_1 & 0 & Y_2 & 0 & Y_3 \\ Y_1 & X_1 & Y_3 & X_2 & Y_3 & X_3 \end{bmatrix} \tag{4.14}$$

where

$$\begin{aligned} X_1 &= \frac{1}{|\mathbf{J}|}(y_2 - y_3), \quad X_2 = \frac{-1}{|\mathbf{J}|}(y_1 - y_3), \quad X_3 = -X_1 - X_2 \\ Y_1 &= \frac{-1}{|\mathbf{J}|}(x_2 - x_3), \quad Y_2 = \frac{1}{|\mathbf{J}|}(x_1 - x_3), \quad Y_3 = -Y_1 - Y_2 \end{aligned} \tag{4.15}$$

and

$$|\mathbf{J}| = (x_1 - x_3)(y_2 - y_3) - (x_2 - x_3)(y_1 - y_3) \tag{4.16}$$

where $|\mathbf{J}|$ is the Jacobian matrix that is twice of the area of the triangle.

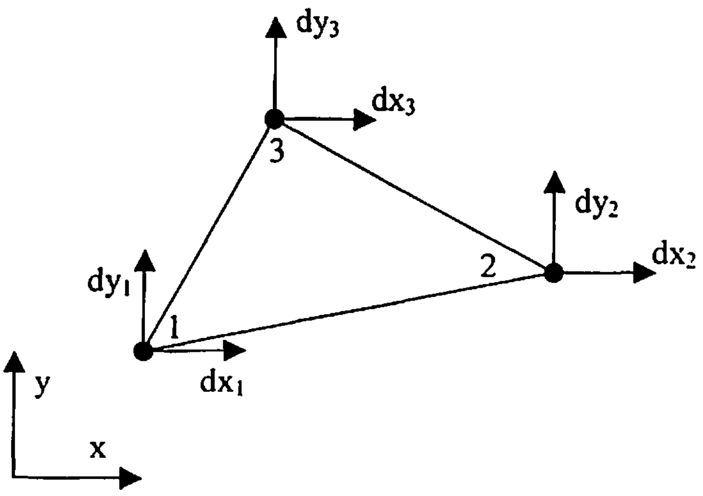


Fig. 4.7 Constant strain triangular element

4.3.5 Elemental Stress and Strain Computations

Stresses can be computed by using Eq. 4.7. Elemental strains can be computed by multiplying strain displacement matrix of a particular element by nodal displacement. Mathematically it can be written as

$$\{\epsilon\} = [B]\{d\} \tag{4.17}$$

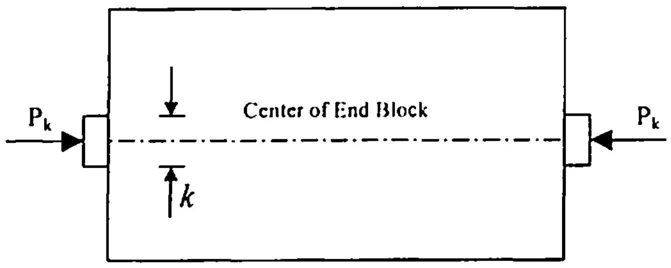
4.3.6 Solution Procedure

Structural analysis using finite element method can be carried out using following six steps.

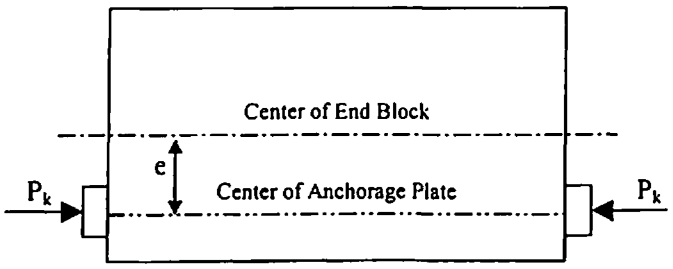
1. Discretization of the problem domain.
2. Derivation of stiffness matrix for element used for problem discretization.
3. Assembling all local stiffness matrices into global stiffness matrix and generation of global nodal load vector by assembling local nodal load vectors.
4. Putting of boundary conditions in global equation.
5. Solving global equation to get the values of unknowns.
6. Computation of strains and stresses

4.4 METHODOLOGY

In the present study, the problem of anchorage zone in prestressed post-tensioned concrete beam is idealized as two-dimensional plane stress problem. A rectangular beam of unit thickness is considered. Length of beam is taken as twice of its depth. Finite element method is used to analyze this problem. The analysis of beam is carried out considering two cases: Case I – concentric prestressing and Case II – eccentric prestressing. Large number of constant strain triangular elements were used for the problem discretization so that proper stress variation can be achieved (These stresses are normalized by dividing them by average longitudinal stress ($\sigma_{x(avg)}$)). This resulted in large computational time when single processor was used. Therefore a finite element computer code is developed incorporating parallel solver discussed in section 3.7. Figure 4.8 (a) and 4.8 (b) show Case I and Case II problems respectively.



(a) concentric prestressing forces



(b) eccentric prestressing forces

Fig. 4.8 Idealized prestressed concrete beam

4.5 CASE STUDIES

The case studies include two types of prestressing conditions. In the first case study concentric prestressing force is considered while in second eccentric prestressing force is considered. The values of ratio of loaded area and cross-sectional area of the beam (k) and Poisson's ratio (v) are changed and their effect on bursting tensile force is studied. In the second case, effect of eccentricity on spalling zone stresses is studied.

4.5.1 Case I: Concentric Prestressing

why constant strains

Figure 4.8(a) shows the idealized prestressed concrete beam subjected to concentric prestressing forces. The beam is discretized three times using 1600, 1136 and 784 constant strain triangular elements with 861, 613 and 435 nodes resulting in 1722, 1226 and 870 unknown displacements respectively. Figure 4.9 shows the discretized beam with 1136 elements and 613 nodes. The grade of concrete is taken as M45. To get correct results, the mesh is kept finer along the axis of loading and along the loaded face. The

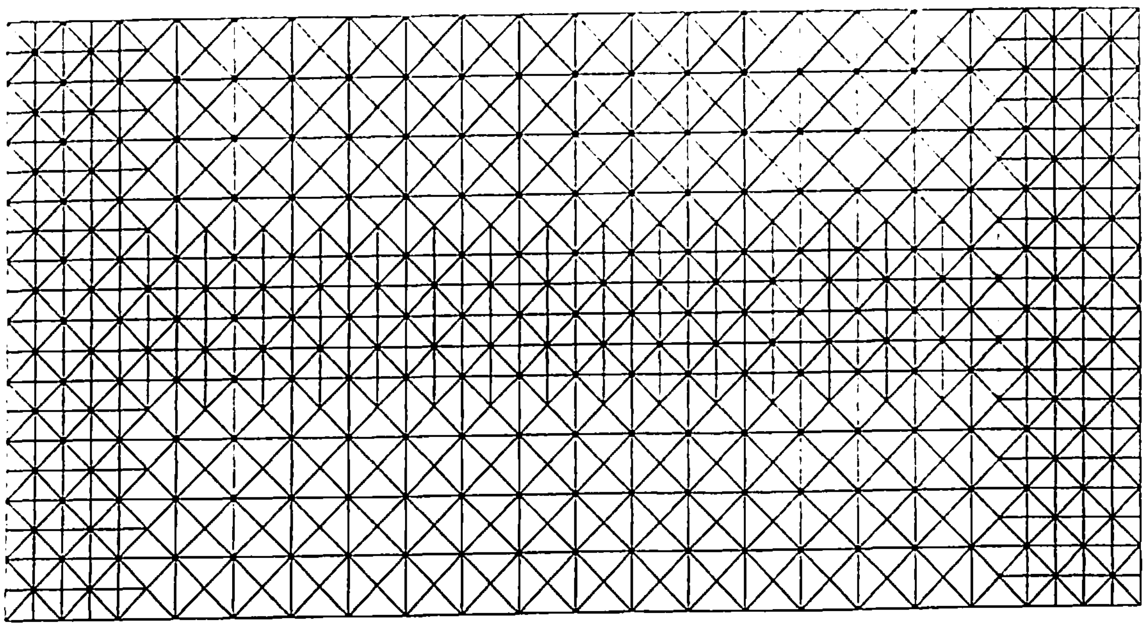


Fig. 4.9 Discretized prestressed concrete beam with 1136 elements and 613 nodes

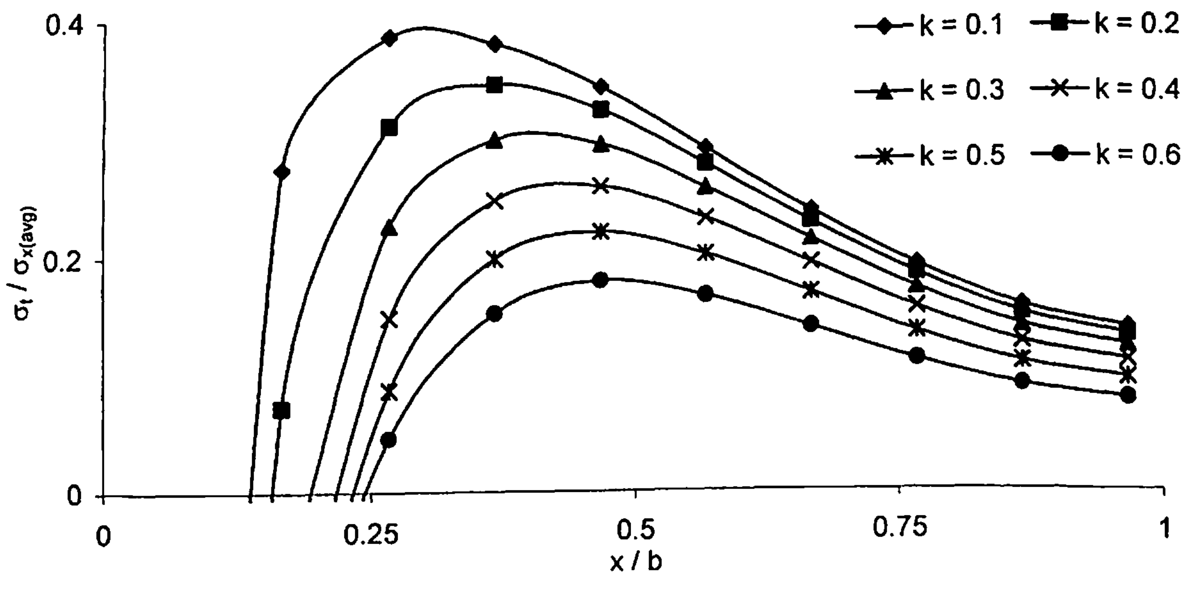


Fig. 4.10 Distribution of σ_t along axis of loading for $\nu = 0.0$

analysis of beam is carried out for different values of k and ν. The value of k is varied between 0.1 to 0.6. As the value of ν for M45 grade concrete is in the range of 0.0 to 0.3, so the value of ν is considered in the same range during the analysis. The problem is then analyzed by changing values of ν and k with the help of developed finite element computer code on the platform of PARAM 10000.

4.5.1.1 Transverse Tensile Stress Variation

Figure 4.10 shows the distribution of transverse tensile stress (σ_t) for the different values of k for ν = 0.0. One can observe that as the value of k increases the magnitude of maximum transverse tensile stress reduces. The position of maximum transverse tensile stress shifts along the axis of beam away from the loaded face as the value of k increases. Here it is clear that the position of the zero transverse tensile stress $\sigma_{t(zero)}$ shifts along the axis of beam and away from the loaded face as the values of k increase.

Figure 4.11 shows the comparison of distribution of transverse tensile stress (σ_t) for value of k = 0.1. It can be observed that the magnitude of maximum transverse tensile stress ($\sigma_{t(max)}$) calculated by Guyon [34] is on extremely higher side. The stress distribution obtained by present investigation fairly matches with the distribution obtained by Iyengar (two-dimensional as well as three-dimensional) [36, 37].

For the higher values of k = 0.5 (see Fig. 4.12), the stress distribution obtained by present investigation matches well with the stress distribution obtained by Iyengar (two-dimensional) [36], whereas the stress distribution in three-dimensional analysis given by Iyengar [37] does not match with the stress distribution obtained by present investigation. The magnitude of maximum transverse tensile stress obtained by Iyengar (three-dimensional) [37] is nearly half of the maximum transverse tensile stress obtained by present investigation for the same value of ν = 0.15 and k = 0.5.

Figure 4.13 shows the distribution of transverse tensile stress (σ_t) for various values of β = 0.2 (where β is the ratio of loaded depth and actual depth of beam, in present investigation k = β). The graph shows that for value of β = 0.2, the stress distribution obtained from present investigation match well with the stress distribution given by

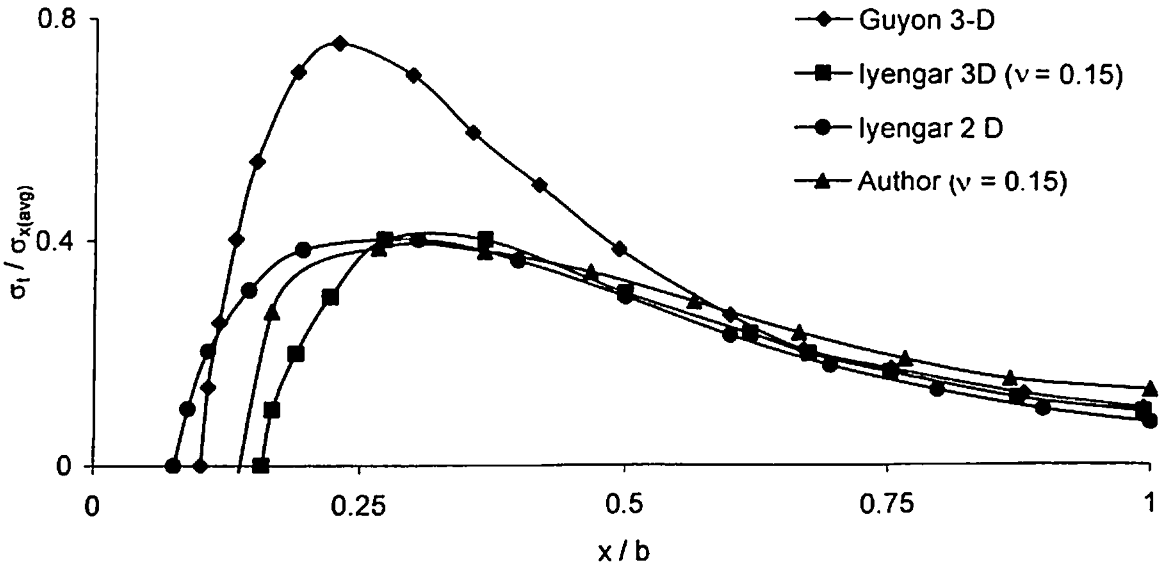


Fig. 4.11 Comparison of transverse tensile stress distribution along the axis of loading for k = 0.1

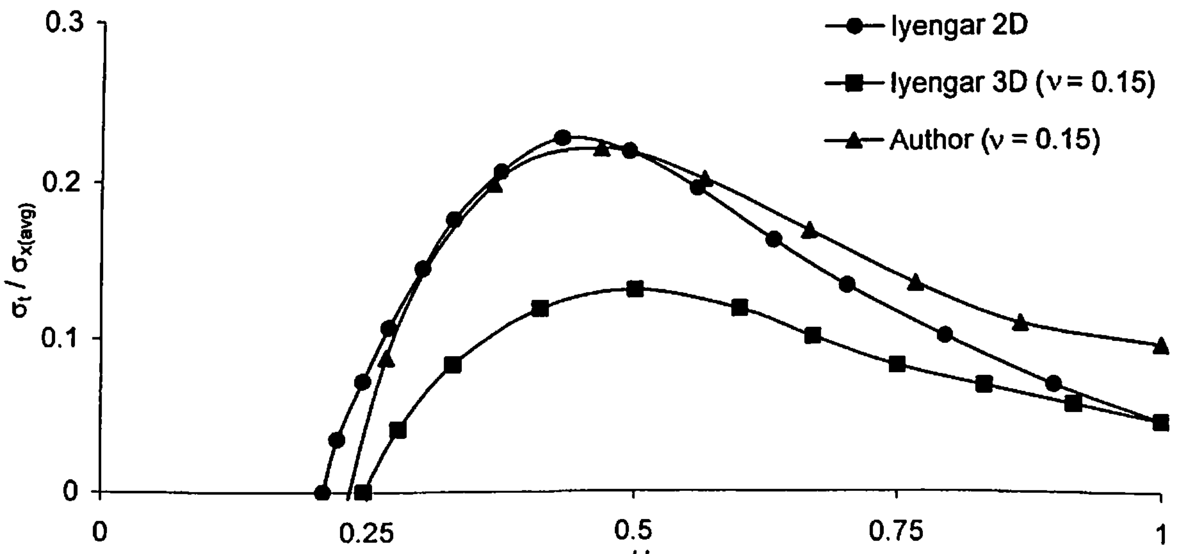


Fig. 4.12 Comparison of transverse tensile stress distribution along the axis of loading for k = 0.5

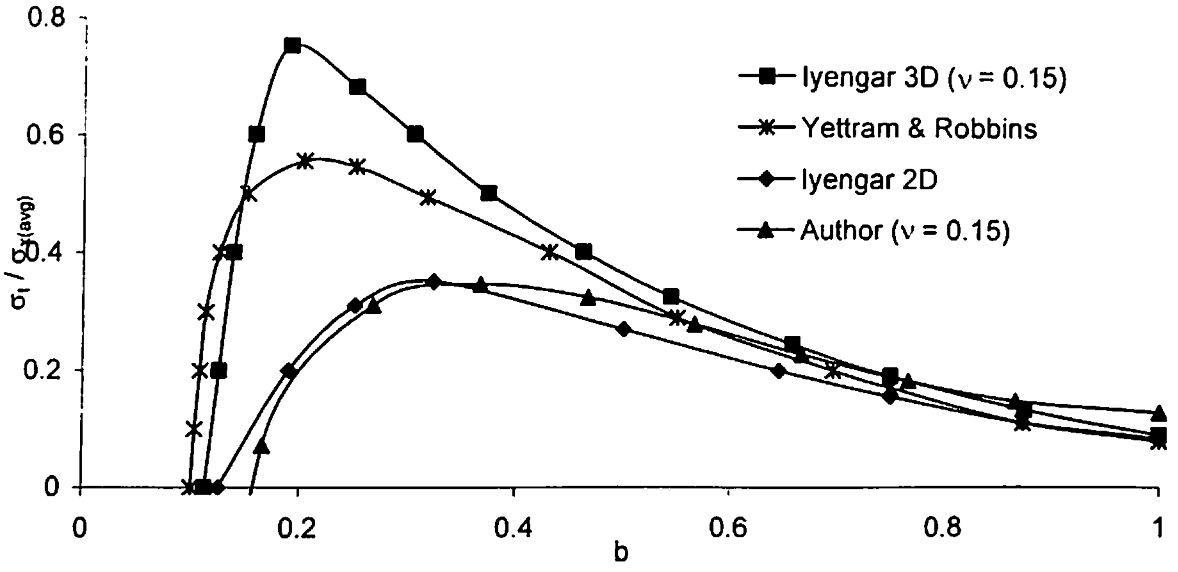


Fig. 4.13 Comparison of transverse tensile stress distribution along the axis of loading for $\beta = 0.2$

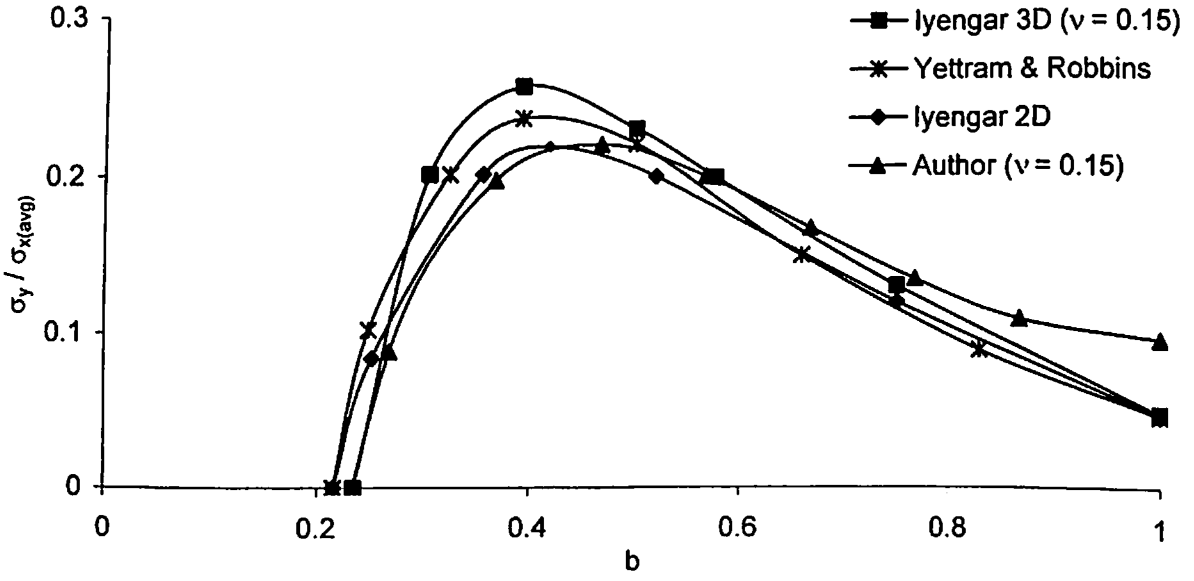


Fig. 4.14 Comparison of transverse tensile stress distribution along the axis of loading for $\beta = 0.5$

Iyengar (two-dimensional) [36]. The stress distribution given by Yettram and Robbins [42] and Iyengar (three-dimensional) [37] do not match with the stress distribution obtained from present investigation. The magnitude of maximum transverse tensile stress given by Yettram and Robbins [42] is nearly 1.5 times and given by Iyengar (three-dimensional) [37] is nearly 2.5 time of the magnitude of maximum transverse tensile stress obtained by present investigation. One can also observe that the location of maximum transverse tensile stress given by Yettram and Robbins [42] and Iyengar (three-dimensional) [37] is closer to the loaded face as compared the location of maximum transverse tensile stress given by Iyengar (two-dimensional) [36] and by present investigation.

For value of $\beta = 0.5$, the stress distribution obtained by present investigation match well with the distribution given by Iyengar (two-dimensional) [36], Yettram and Robbins [42] and Iyengar (three-dimensional) [37] (see Fig. 4.14).

Figure 4.15 shows the position of zero transverse stress $\sigma_{t(zero)}$ for different values of k. The results of Iyengar (two-dimensional) show that the position of the zero transverse stress $\sigma_{t(zero)}$ is nearest to the loaded surface for all possible values of k. The present analysis shows that the location of zero transverse stress $\sigma_{t(zero)}$ falls slightly ahead than the Iyengar's (two-dimensional) analysis, whereas three-dimensional investigations by Yettram and Robbins as well as Iyengar (three-dimensional) indicate the location of zero transverse stress $\sigma_{t(zero)}$ further ahead as compared to the two-dimensional analysis. Experimental investigation by Zielinski and Rowe [40] shows that there is very little variation in position of zero transverse stress.

Figure 4.16 shows the variation of maximum transverse tensile stress $\sigma_{t(max)}$ for different values of k. The results show that the location of maximum transverse tensile stress $\sigma_{t(max)}$ obtained by present investigation match well with the three-dimensional analysis of Yettram and Robbins. Among all the results the location of maximum transverse tensile stress $\sigma_{t(max)}$ presented by Iyengar's (two-dimensional) is nearest and Iyengar (three-dimensional) is farthest from the loaded face. The experimental results of Zielinski and Rowe show very little variation in position of maximum transverse tensile stress $\sigma_{t(max)}$.

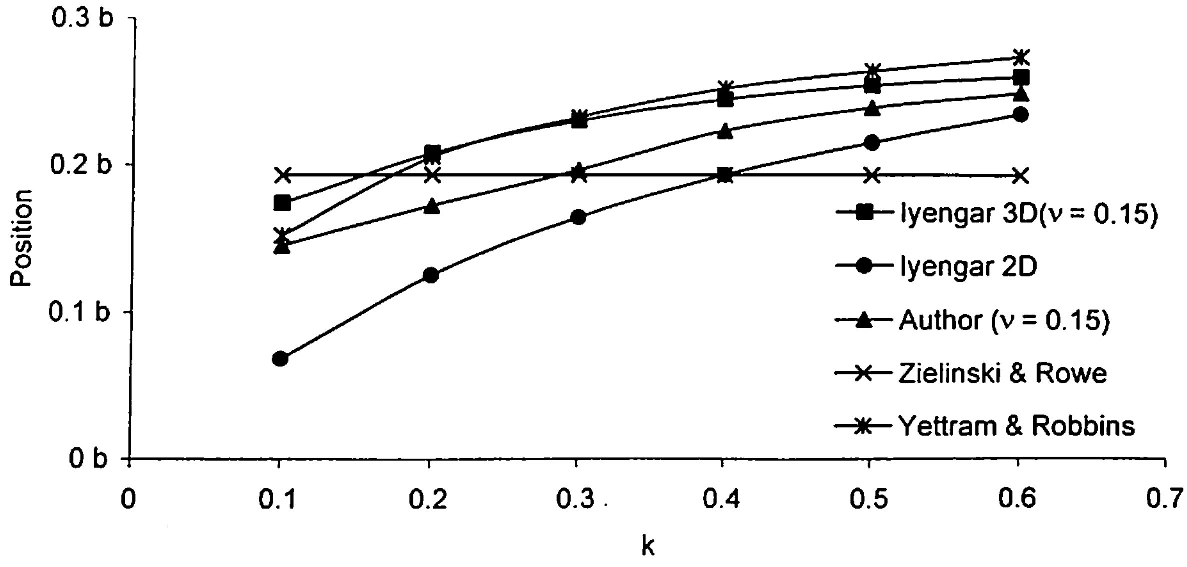


Fig. 4.15 Position of zero transverse stress $\sigma_{t(zero)}$ along the axis of loading

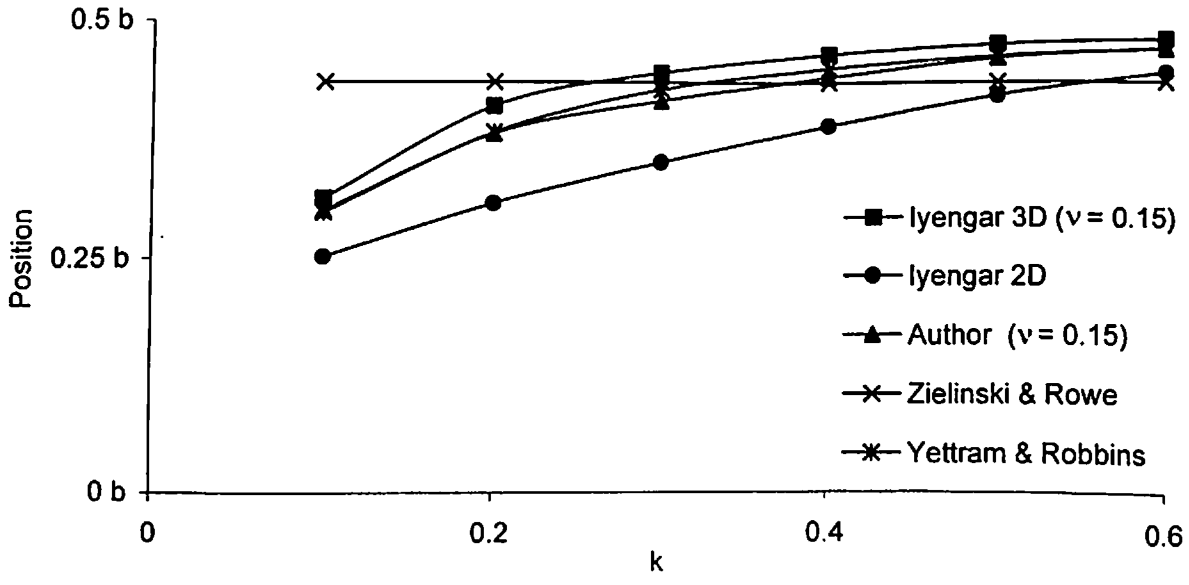


Fig. 4.16 Position of maximum $\sigma_{t(max)}$ along the axis of loading

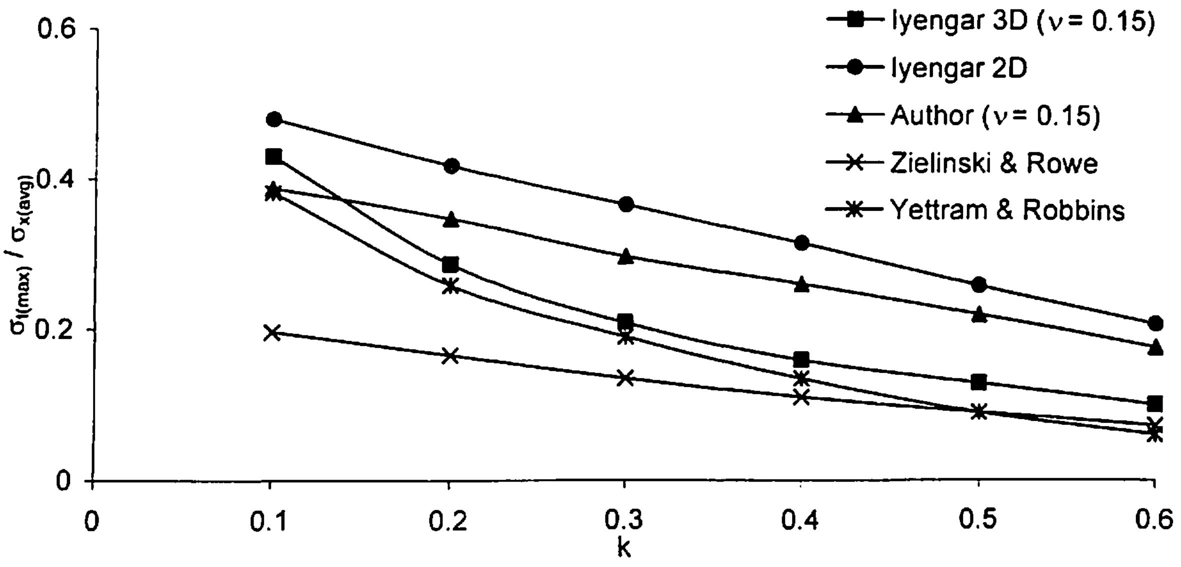


Fig. 4.17 Magnitude of maximum transverse tensile stress $\sigma_{t(max)}$ along the axis of loading

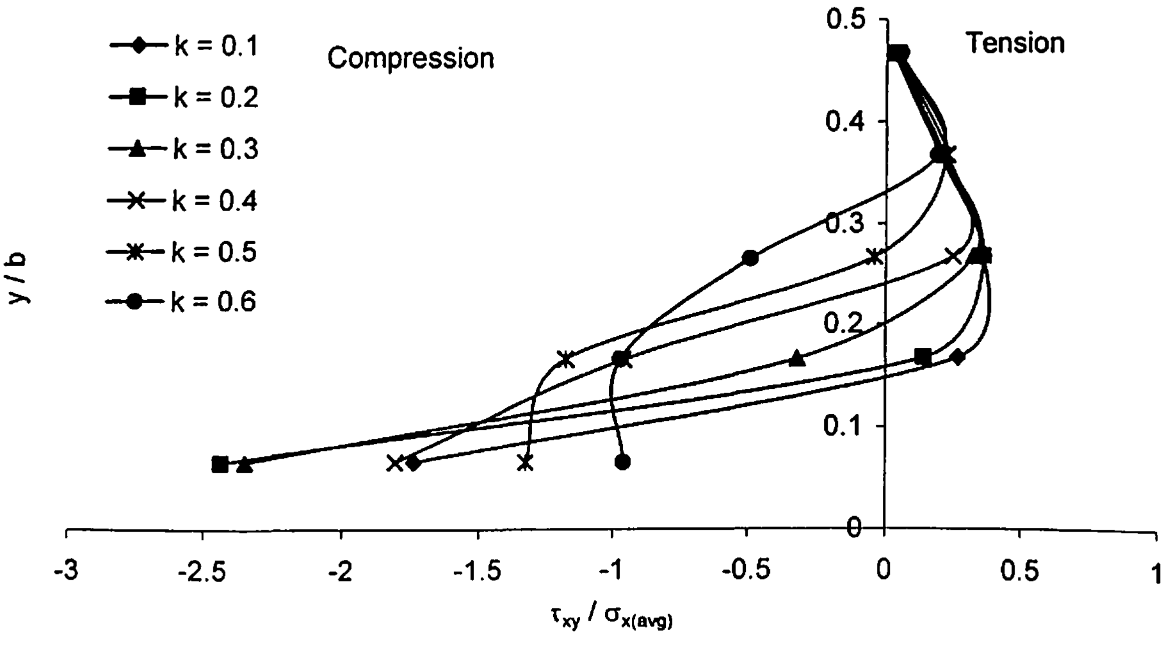


Fig. 4.18 Transverse stress distribution on loaded face (ν = 0.15)

Figure 4.17 shows the variation in magnitude of maximum transverse tensile stress $\sigma_{t(max)}$ for different values of k. This figure shows that the magnitude of maximum transverse tensile stress by present investigation is slightly less than magnitude of maximum transverse tensile stress of Iyengar's (two-dimensional) analysis. On the other hand three-dimensional investigations by Iyengar, Yettram and Robbins give lower values of magnitude of maximum transverse tensile stress $\sigma_{t(max)}$. Experimental investigations by Zielinski and Rowe shows that the variation in the magnitude of maximum transverse stress $\sigma_{t(max)}$ is linear. The numerical values of maximum transverse tensile stress are also lowest among all the investigations.

Figure 4.18 shows the distribution of transverse stress along the loaded face for different values of k and ν = 0.00. One can observe that the nature of transverse stress is compressive along the axis of loading. The nature of stress changes from compression to tension as we move towards top/bottom surface of beam from center of beam. Finally transverse stress reaches to zero at the top/bottom corner. This indicates existence of spalling zone in prestressed post-tensioned concrete beams. One can observe that area under tension reduces with the increase k. One can also observe that the magnitude of maximum transverse tensile stress along loaded face also decreases with increase in k.

4.5.1.2 Longitudinal Stress Variation

Figure 4.19 shows the variation in the longitudinal stress (σ_x) along the axis of loading for different values of k and ν = 0.15. The results indicate that for smaller values of k the longitudinal stress (σ_x) is very high near the loaded end. This stress then slowly reduces and finally becomes almost constant after the anchorage zone, which satisfies the Saint Venant's principle.

Figure 4.20 shows the variation in the longitudinal stress (σ_x) along the top/bottom surface of the beam for different values of k. It can be observed that the stress variation obtained from present investigation match well with the variation presented by Iyengar (two-dimensional). Figure 4.20 also indicates that the variation obtained by Iyengar (three-dimensional) do not match with variation obtained by two-dimensional analysis carried out by Iyengar [36]. One can also observe that, in Iyengar's three-dimensional

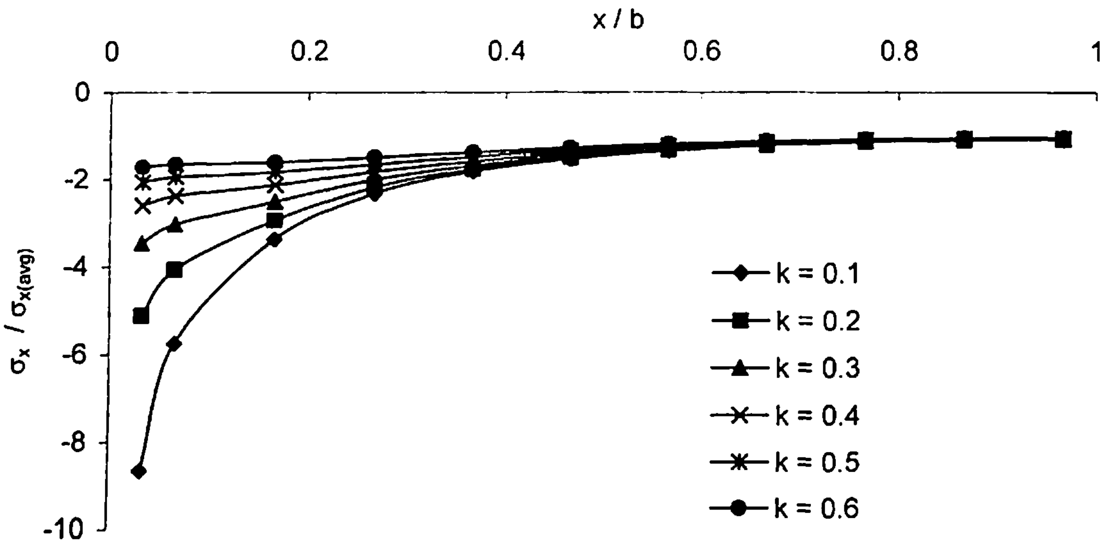


Fig. 4.19 Distribution of longitudinal stress σ_x along axis of loading for different values of k and $\nu = 0.15$

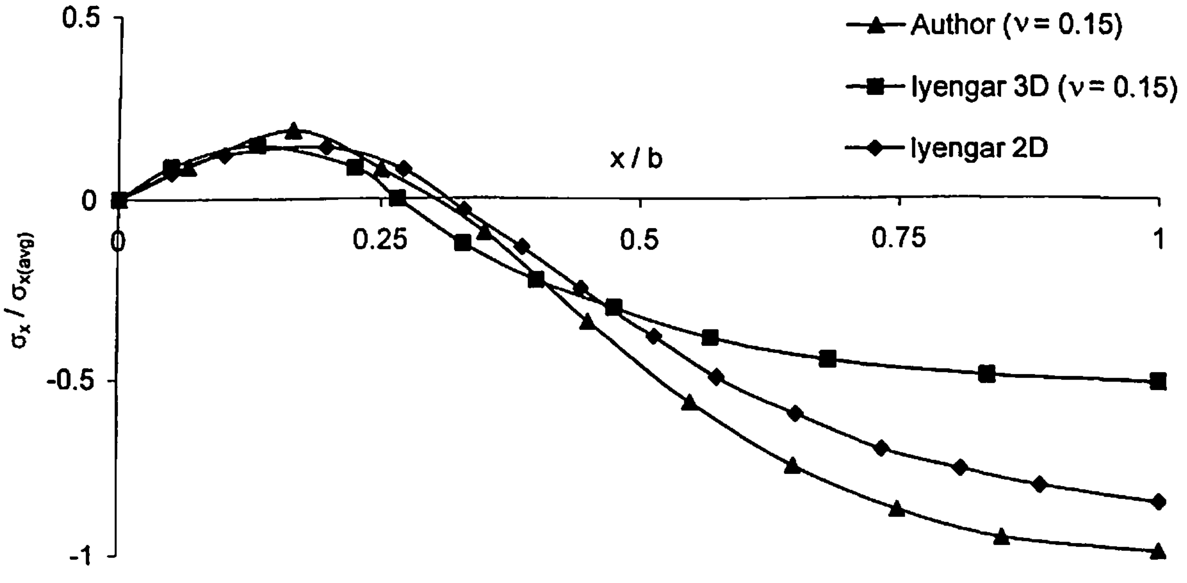


Fig. 4.20 Distribution of longitudinal stress σ_x along the top/bottom surface for different values of k

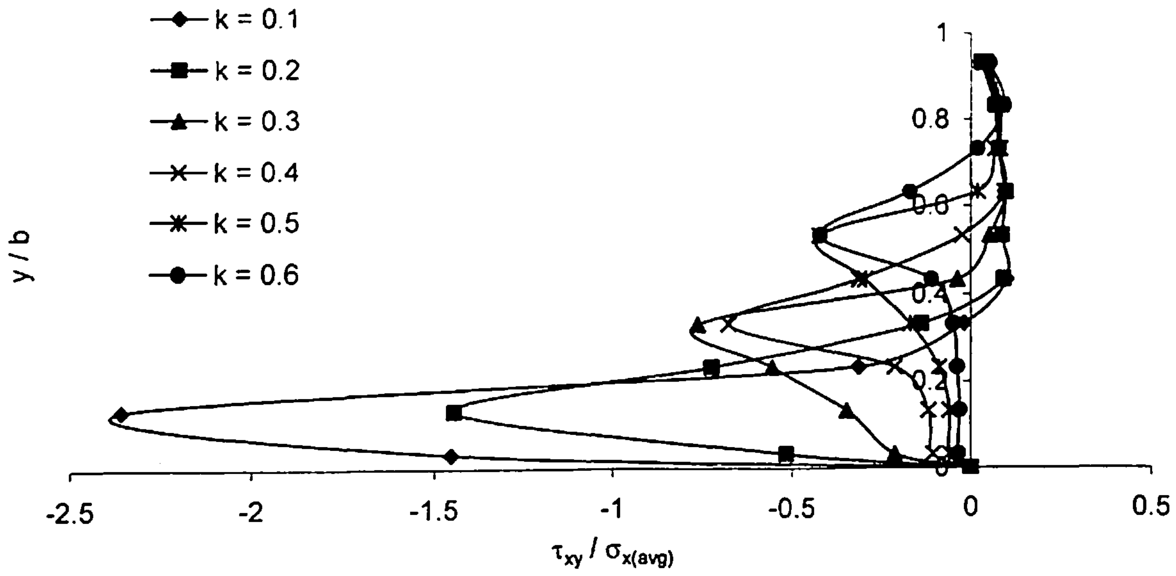


Fig. 4.21 Distribution of shear stress along the loaded face for different values of k

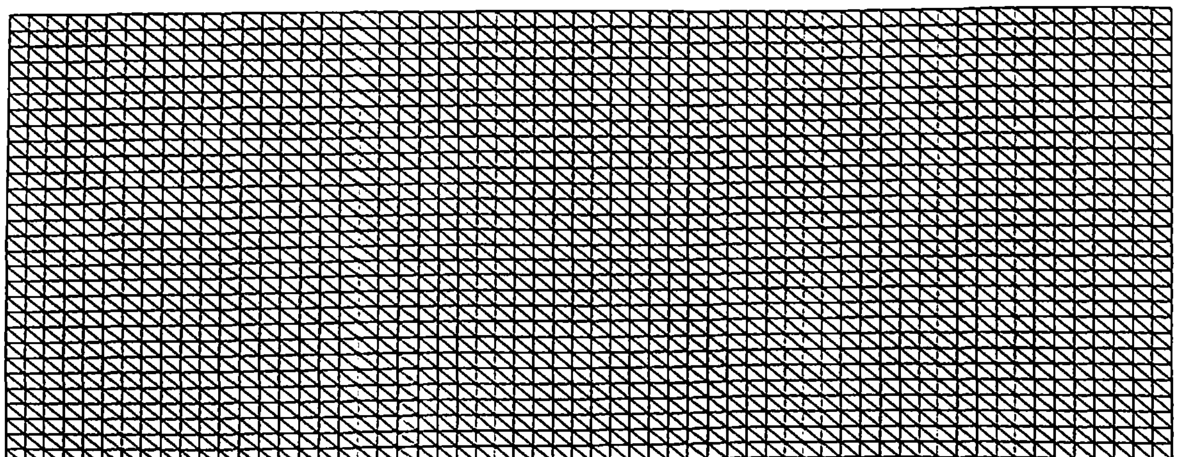


Fig. 4.22 Discretized prestressed concrete beam with 4800 elements and 2501 nodes

analysis ratio of longitudinal stress to average stress is nearly 0.5 at the distance of d from loaded face that does not satisfies Saint Venant's principle.

4.5.1.3 Shear Stress Variation

Figure 4.21 shows the variation in shear stress along the loaded face for different values of k. One can observe that shear stress is zero at the center of loaded face. Then magnitude of shear stress increases and reaches to peak value. After that it starts reducing and its nature reverses. After reaching to the peak value, finally it becomes zero near the top/bottom corner of beam. It can also be observed that the magnitude of maximum shear stress reduces with increase in value of k. The point of contraflexture moves away from center of loaded face toward the top/bottom of beam as the value of k increases. Such type of variation confirms the existence of spalling zone.

4.5.2 Case II: Eccentric Prestressing

Figure 4.8(b) shows the idealized prestressed concrete beam subjected to eccentric prestressing forces. As the stress variation in eccentrically loaded prestressed concrete beam is very complex, the beam is discretized using 4800 constant strain triangular elements with 2501 nodes resulting in 5002 unknown displacements (see Fig. 4.22). The numerical value of Poisson's ratio is considered as 0.15. The problem is then analyzed with changing values of eccentricity keeping value of k = 0.1 with the help of finite element computer code developed on the platform of PARAM 10000.

4.5.2.1 Transverse Tensile Stress Variation

Figure 4.23 shows the variation in σ_t along the axis of loading for different values of eccentricity for $k = 0.1$ and $\nu = 0.15$. It can be observed from this figure that as the eccentricity of prestressing force increases the magnitude of $\sigma_{t\,(max)}$ inside the anchorage zone also increases. One can observe that the location of maximum transverse tensile stress moves towards the loading face as the eccentricity of forces increases.

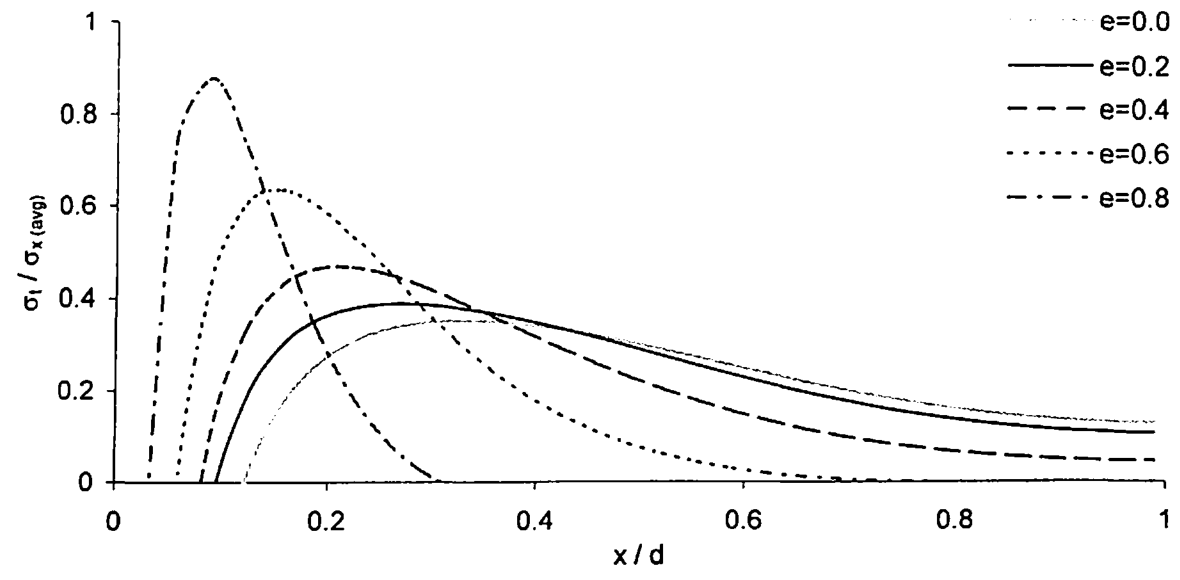


Fig. 4.23 Variation in transverse tensile stress along the axis of loading for different values of eccentricity and k = 0.1

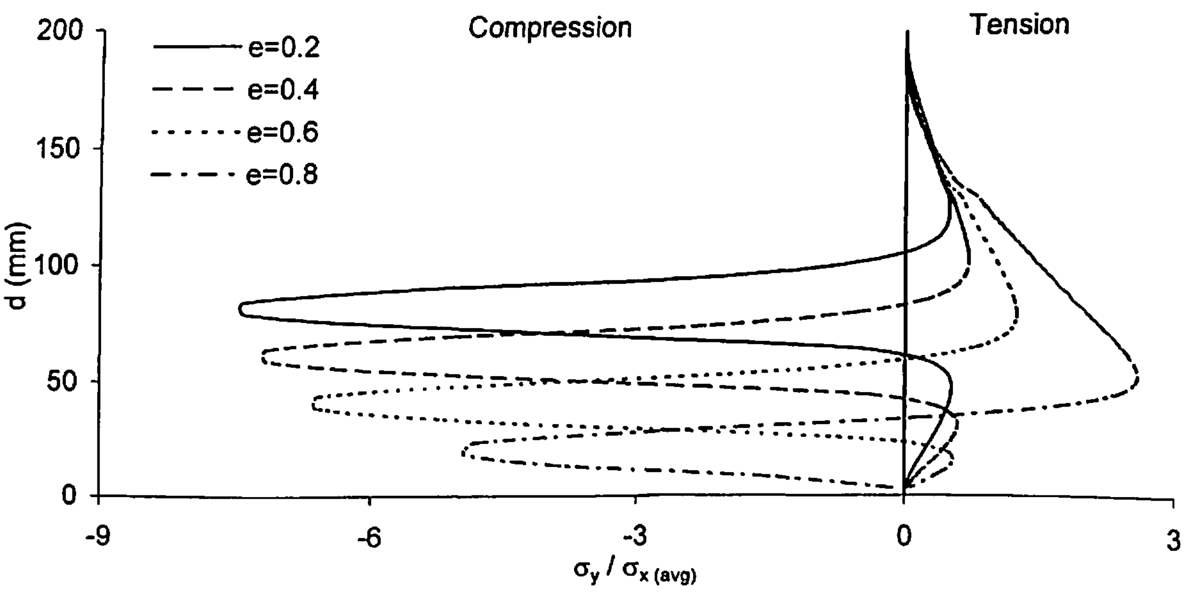


Fig. 4.24 Variation in transverse tensile stress along the loaded face for different values of eccentricity and k = 0.1

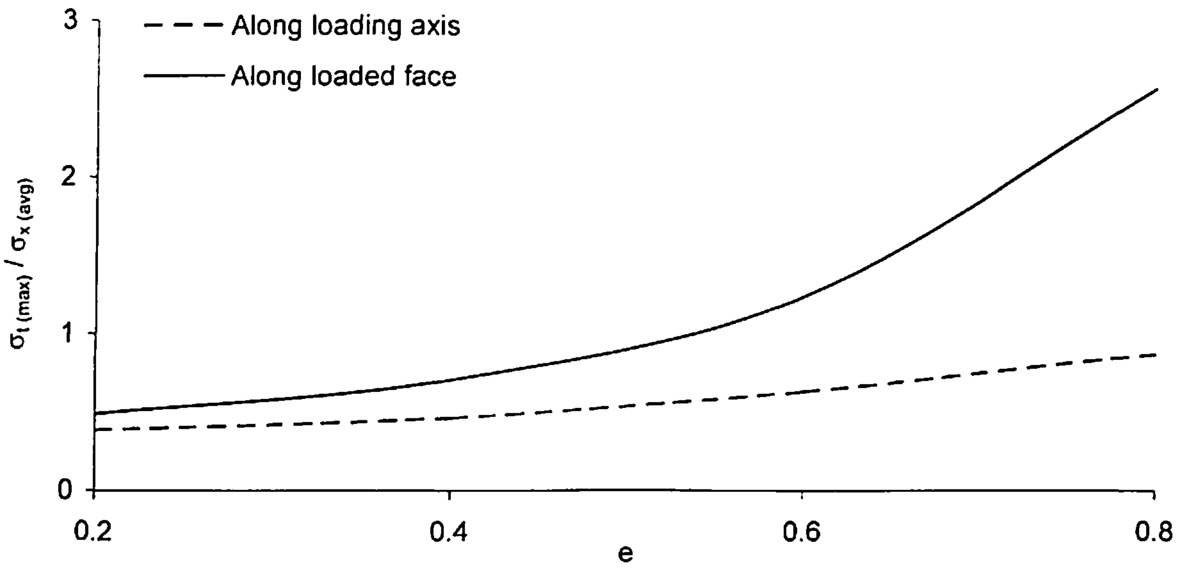


Fig. 4.25 Comparison of $\sigma_{t\,(max)}$ along loaded face and axis of loading for different values of eccentricity

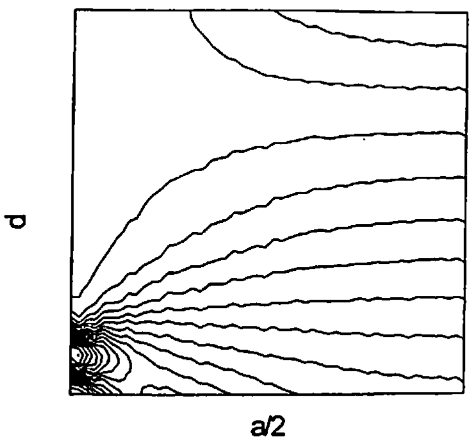


Fig. 4.26 Contours of longitudinal stress at e = 0.8 and k = 0.1

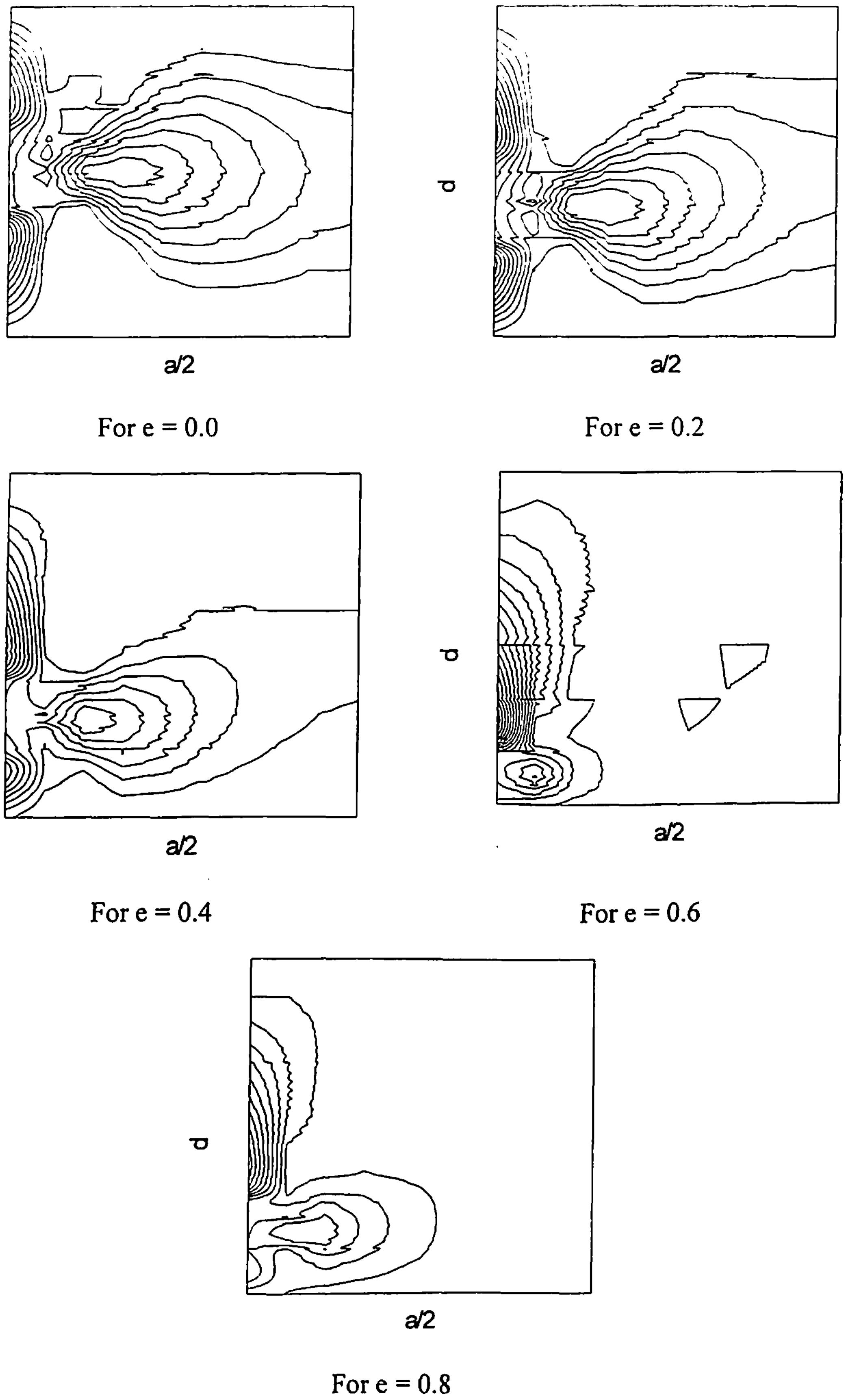


Fig. 4.27 Contours of transverse tensile stress for different values of eccentricity

In Fig. 4.24, the variation of σ_y along the loaded face for different values of eccentricity is shown. It can be observed that as the value of eccentricity increases the magnitude of $\sigma_{t(max)}$ also increases. On the other hand, the magnitude of maximum transverse compressive stress ($\sigma_{c(max)}$) reduces with the increase in value of eccentricity. One can observe that considerable portion along the loaded face is subjected tensile stress, which indicates the region of spalling zone.

One very important observation can be made that the magnitude of $\sigma_{t(max)}$ along loaded face is very much higher than the magnitude of $\sigma_{t(max)}$ along the axis of loading for eccentric prestressing forces (See Fig. 4.25). For example, at $e = 0.8$ the magnitude of $\sigma_{t(max)}$ along loaded face is 2.57 which is almost three times if the magnitude of $\sigma_{t(max)}$ along the axis of loading ($\sigma_{t(max)} = 0.88$). It can also be observed that, $\sigma_{t(max)}$ along loaded face increases rapidly with the increase in the value of e.

4.5.2.2 Stress Contours

Figure 4.26 shows that higher longitudinal stress concentration exists in the section just next to the area subjected to prestressing force (local zone). These stresses slowly spread in the whole beam cross-section as we move away from the loaded face and at the end of anchorage zone, it gets converted into uniform distribution.

Figure 4.27 shows the stress contours of transverse tensile stress for different values of eccentricity. It can be observed from this figure that the anchorage zone area under tensile stress is wider for lower eccentricity values, while for higher eccentricity values; the spalling zone area subjected to tensile stress is wider.

4.5.3 BURSTING TENSILE FORCE

The Bursting tensile force (F_{bst}) can be found by calculating the total area under the transverse tensile stress curves (See Fig. 4.10 and 4.23). For this purpose, commercial software AutoCAD is used. Transverse tensile stress distribution curves were plotted for different values of eccentricity, k and v. The area under these curves is measured by using

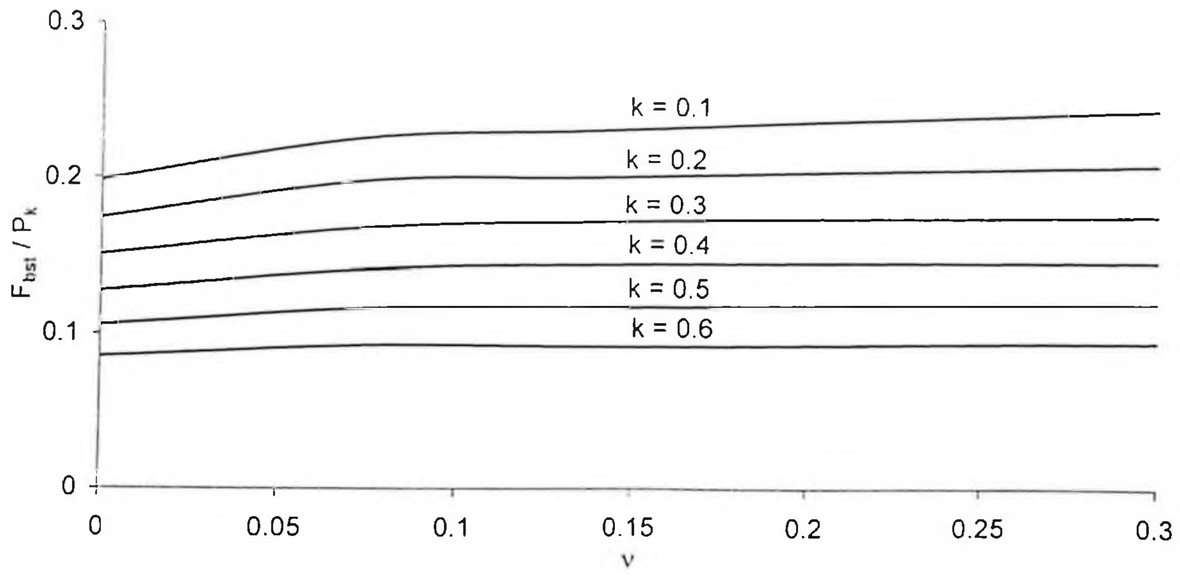


Fig. 4.28 Variation of F_{bst} with ν different values of k

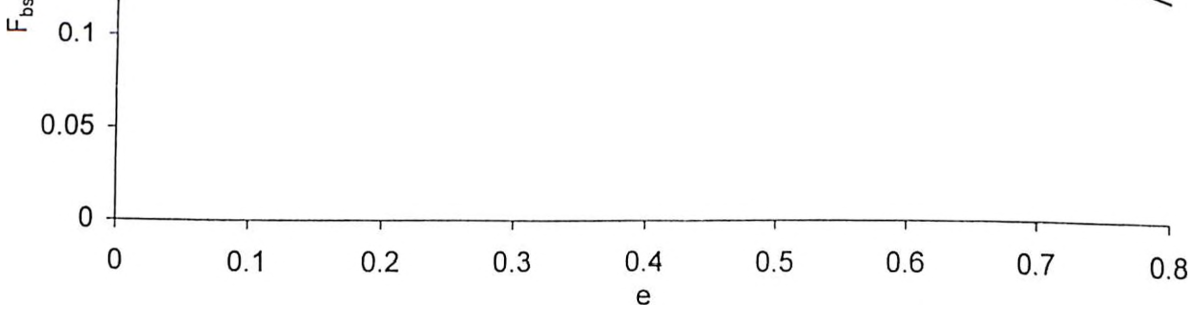


Fig. 4.29 Variation of F_{bst} with eccentricity for k = 0.1 and ν = 0.15

area command in AutoCAD and the variation of bursting tensile force is plotted in Fig. 4.28 and 4.29.

It can be seen from Fig. 4.28 that bursting tensile force is higher for lower values of k. It can also be observed that bursting tensile force also varies with the Poisson's ratio ν. This variation is insignificant for higher values of k. For lower values of k, significant variation can be seen. It can be seen from Fig. 4.29 that F_{bst} decreases with the increase in value of eccentricity. The magnitude of maximum F_{bst} can be observed at concentric loading conditions (e = 0.0). It can be seen from Fig. 4.23 that the area under the transverse tensile stress curve reduces with increase in value of e, which ultimately results in reduction in bursting tensile force. Hence to insure the safety of the prestressed concrete beam, the effect of eccentricity can be ignored while computing the magnitude of bursting tensile force.

Multiple regression analysis was carried out to obtain the correlation between F_{bst} and ν ignoring the effect of e (see table 4.1). The following two expressions were obtained

$$F_{bst} = P_k\,(0.239 - 0.267\,k + 0.075\,\nu) \tag{4.18}$$

and

$$F_{bst} = P_k\,(0.229 - 0.238\,k + 0.152\,\nu - 0.22\,k\,\nu) \tag{4.19}$$

The R^2 values obtained for equation 4.18 and 4.19 were 0.98 and 0.988 respectively. It can be noted that these equations also include the effect of Poisson's ratio in calculation of F_{bst} which is ignored in the equation given in Indian Standard Code IS:1343-1980 (Eq. 4.20).

$$\frac{F_{bst}}{P_k} = 0.32 - 0.3\frac{y_{po}}{y_o} \tag{4.20}$$

where,

$$\frac{y_{po}}{y_o} = k$$

Table 4.1 Table showing magnitude of bursting tensile force for different values of k and ν

k \ ν	0.10	0.20	0.30	0.40	0.50	0.60
0.00	0.20	0.17	0.15	0.13	0.11	0.09
0.08	0.23	0.20	0.17	0.14	0.12	0.09
0.15	0.23	0.20	0.17	0.14	0.12	0.09
0.30	0.24	0.21	0.18	0.15	0.12	0.10

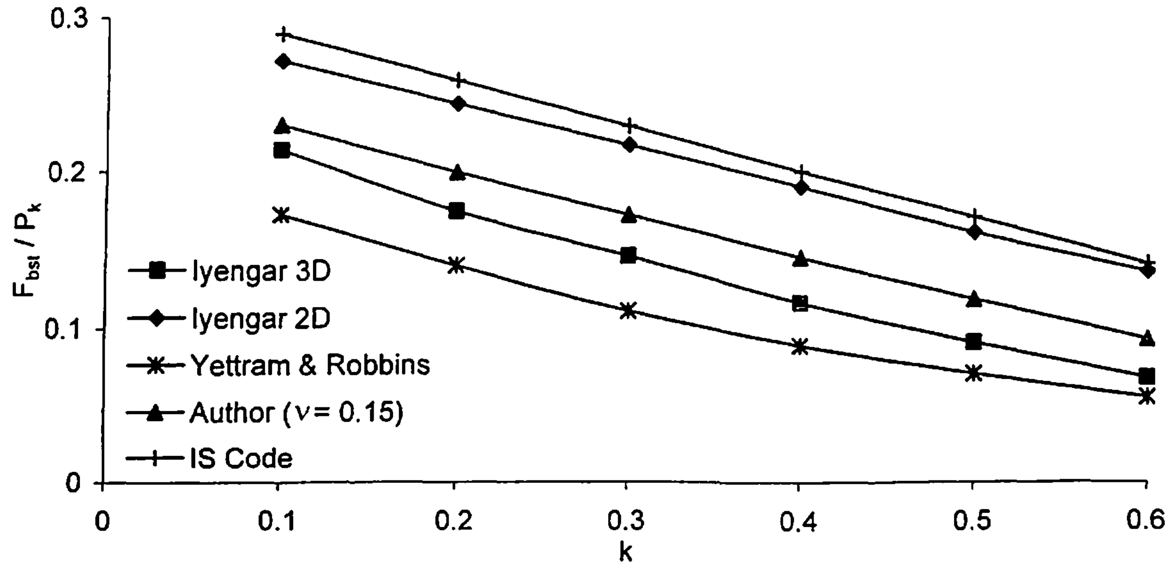


Fig. 4.30 Comparison of variation in bursting tensile force for different values of k

It can be seen that Eq. 4.19 is more accurate as compared to the Eq. 4.18. Hence Eq. 4.19 is followed to compare the results with the available literature. The comparison of variation in F_{bst} for different values of k for a constant value of ν = 0.15 is shown in Fig 4.30. The magnitude of F_{bst} obtained by present investigation will always be lower than the magnitude of F_{bst} obtained using the equation given in the Indian Standard Code IS: 1343-1980 for all values of k. It is clear from this figure that the magnitude of F_{bst} obtained by Iyengar (two-dimensional) is slightly higher and Iyengar (three-dimensional) is slightly lower than the magnitude of F_{bst} obtained by present investigation. Among all the investigations, the magnitude of F_{bst} obtained by Yettram and Robbins is the lowest and shows curvilinear variation.

Table 4.2 Computational time variation for problem having size of global stiffness matrix 870 × 870

No. of processors	Total		Comm		Cal	
	Real	User	Real	User	Real	User
1	88.83	62.56	0.00	0.00	88.83	62.56
2	78.08	35.26	0.82	0.19	77.26	35.07
3	65.76	29.50	5.20	0.52	60.57	28.98
4	53.18	23.87	2.23	0.44	50.96	23.43
5	70.02	20.74	4.30	0.67	65.72	20.07
6	73.72	19.41	4.08	1.09	69.64	18.32
7	66.12	17.18	4.96	0.71	61.16	16.47
8	39.57	15.21	3.21	0.82	36.36	14.39

Table 4.3 Performance of parallelized FEM code problem having size of global stiffness matrix 870 × 870

No. of processors	Speedup		Efficiency	
	Real	User	Real	User
1	1.00	1.00	100.00	100.00
2	1.14	1.77	56.88	88.71
3	1.35	2.12	45.02	70.69
4	1.67	2.62	41.76	65.52
5	1.27	3.02	25.37	60.33
6	1.20	3.22	20.08	53.72
7	1.34	3.64	19.19	52.02
8	2.24	4.11	28.06	51.41

Table 4.4 Computational time variation for problem having size of global stiffness matrix 1226 × 1226

No. of processors	Total		Comm		Cal	
	Real	User	Real	User	Real	User
1	205.07	181.61	0.00	0.00	205.07	181.61
2	165.21	103.64	1.53	0.58	163.68	103.06
3	127.76	85.31	3.54	0.77	124.23	84.54
4	120.22	70.03	3.97	0.94	116.25	69.09
5	130.43	61.13	14.20	1.98	116.23	59.15
6	139.55	55.47	10.86	1.89	128.69	53.58
7	121.91	48.00	8.78	1.40	113.13	46.60
8	135.81	43.36	4.75	1.78	131.07	41.58

Table 4.5 Performance of parallelized FEM code problem having size of global stiffness matrix 1226 × 1226

No. of processors	Speedup		Efficiency	
	Real	User	Real	User
1	1.00	1.00	100.00	100.00
2	1.24	1.75	62.06	87.62
3	1.61	2.13	53.50	70.96
4	1.71	2.59	42.65	64.83
5	1.57	2.97	31.44	59.42
6	1.47	3.27	24.49	54.57
7	1.68	3.78	24.03	54.05
8	1.51	4.19	18.87	52.36

Table 4.6 Computational time variation for problem having size of global stiffness matrix 1722 × 1722

No. of processors	Total		Comm		Cal	
	Real	User	Real	User	Real	User
1	845.08	841.90	0.00	0.00	845.08	841.90
2	630.55	623.85	3.31	0.97	627.25	622.88
3	472.25	459.60	10.28	1.67	461.97	457.93
4	374.47	363.43	8.38	2.28	366.09	361.15
5	320.40	301.00	10.54	2.10	309.85	298.90
6	283.94	259.53	11.76	3.64	272.18	255.89
7	252.17	225.90	14.39	2.49	237.79	223.41
8	467.54	203.46	28.65	4.06	438.89	199.40

Table 4.7 Performance of parallelized FEM code problem having size of global stiffness matrix 1722 × 1722

No. of processors	Speedup		Efficiency	
	Real	User	Real	User
1	1.00	1.00	100.00	100.00
2	1.34	1.35	67.01	67.48
3	1.79	1.83	59.65	61.06
4	2.26	2.32	56.42	57.91
5	2.64	2.80	52.75	55.94
6	2.98	3.24	49.61	54.07
7	3.35	3.73	47.87	53.24
8	1.81	4.14	22.59	51.72

Table 4.8 Computational time variation for problem having size of global stiffness matrix 5002 × 5002

No. of processors	Total		Comm		Cal	
	Real	User	Real	User	Real	User
1	22850.71	20821.39	0.00	0.00	22850.71	20821.39
2	15399.26	15305.83	89.59	32.75	15309.67	15273.08
3	11467.03	11316.55	138.44	45.12	11328.59	11271.43
4	9027.31	8884.88	151.80	48.76	8875.50	8836.12
5	7538.28	7328.22	133.22	28.35	7405.06	7299.87

Table 4.9 Performance of parallelized FEM code problem having size of global stiffness matrix 5002 × 5002

No. of processors	Speedup		Efficiency	
	Real	User	Real	User
1	1.00	1.00	100.00	100.00
2	1.48	1.36	74.19	68.02
3	1.99	1.84	66.42	61.33
4	2.53	2.34	63.28	58.59
5	3.03	2.84	60.63	56.83

4.6 COMPUTATIONAL TIME RESULTS

In the present investigation several analyses were carried out. Due to the numerous calculations in each analysis of the each analysis, the computational time consumption was very high. To save the computational time, the present analysis is carried out on supercomputer PARAM 10000. Present study is carried out on four different types of finite element meshes. These four meshes resulted in global stiffness matrices of size 5002, 1722, 1226 and 870. Table 4.2 to 4.9 shows the computational time variation for these problems for different number of processors.

It can be observed that Total time (measured in terms of Real time as well as User time) reduces with increase in number of processors for all four different types of problems having different mesh sizes. One can observe that reduction in Total time measured in term of Real time is abrupt whereas smooth variation can be observed for Total time measured in terms of User time. It can also be observed that the contribution of Communication time (measured in terms of Real time as well as User time) is very insignificant as compared to the Total time. Maximum User time Speedup of nearly 4.00 and 50% reduction in User time can be seen at eight number of processors for all finite element meshes under consideration.

4.7 SUMMARY

An application of parallel computing technique for linear elastic finite element analysis is presented though this chapter. A problem of anchorage zone in prestressed post-tensioned concrete beam is analyzed on supercomputer PARAM 10000. Transverse tensile stress distribution is obtained for different values of k and ν for concentric prestressing forces. This distribution is compared with the literature and found that obtained results match well with the literature. Longitudinal stress distribution and shear stress distribution is also obtained and discussed. Effect of eccentricity on transverse tensile stress is studied and it was found that magnitude of maximum transverse tensile stress in anchorage zone reduces with increase in eccentricity. Bursting tensile force variation is studied for concentric and eccentric prestressing forces and it was found that highest magnitude of bursting tensile force shall be obtained for concentric case only. Considering various parameters, an equation for computation of bursting tensile force is developed. This equation includes effect of load area ratio and Poisson's ratio together. The results obtained from developed equation are compared with the literature and it was found that predicted results match well with the literature. Computational time results are also obtained for four different types of meshes used in present investigation. Reduction in computational time is achieved by employing multiple number of processor in present investigation. The performance of parallelized finite element code is also measured by calculating Speedup and Efficiency. Maximum User time Speedup of 4.00 with 50% Efficiency was achieved at eight number of processors for all mesh sizes under consideration.

CHAPTER 5

FINITE ELEMENT FORMULATION FOR LARGE DEFORMATIONS AND CODE DEVELOPMENT

5.1 INTRODUCTION

In this chapter a non-linear parallel finite element code is developed using Modified Matrix Inversion Method parallel solver on the platform of supercomputer PARAM 10000. The code is capable to carryout analysis of large deformation problems those could be idealized as two-dimenstional axisymmetric problem and plane strain problem. Flow formulation used in development of finite element code is described in detail. Two types of elements namely three noded triangular element and four noded rectangular element are used for discretizing the problem domain. Two case study problems, one from each category i.e. axisymmetric and plane strain, are presented and their results are compared with the commercial finite element softwares ANSYS and FORGE2. Computational time results of both these case studies are analyzed and discussed. The developed software is also tested for bigger data size by analyzing one of the case study problem discretized using different mesh sizes to ascertain its applicability in three-dimenstional finite element analysis. Based on the variation in computational time results, performance of the developed software is evaluated and presented.

5.2 FINITE ELEMENT FORMULATION

Structural analysis mainly includes two types of problems, which may be categorized as small deformation (strain) and large deformation (strain) problems. In metal forming deformations in plastic stage are very high as compared to their elastic counterparts, so fall under the category of large deformation problems. Various methods of analysis are available for the analysis of large deformation processes. The finite element method is one of the recent and quite efficient technique to analyze such problems. In finite element method two formulations are available, namely flow formulation and solid formulation [63]. In flow formulation, elasticity of the material is neglected during the analysis as plastic strains are on very higher side as compared to the elastic strain. The material behavior is considered as rigid-plastic or rigid-viscoplastic during the analysis. Whereas, in solid formulation, the elasticity of the material is also considered in the analysis so the material is considered as elasto-plastic or elasto-viscoplastic.

In present study, the flow formulation described in literature [63] is used for the finite element solution of the larger deformation processes. This chapter highlights the flow formulation in brief and also describes the finite element procedure to solve two-dimensional plane strain and axisymmetric problems. There are four approaches generally used for the derivation of the basic equations in the finite element analysis. These are the direct approach, the variational method, the method of weighted residuals, and the energy balance approach. Here, the variational method is used.

5.2.1 Basis of Finite Element Formulation

Variational approach requires that among admissible velocities u_i that satisfy the conditions of compatibility and incompressibility, as well as the velocity boundary conditions, the actual solution give the following functional (function of functions) a stationary value

$$\pi = \int_{v} \bar{\sigma}\, \dot{\bar{\varepsilon}} dV - \int_{S_F} F_i u_i dS \qquad \text{for rigid plastic materials}$$

and

$$\pi = \int_{v} E(\dot{\varepsilon}_{ij}) dV - \int_{S_F} F_i u_i dS \qquad \text{for rigid viscoplastic materials} \tag{5.1}$$

where $\bar{\sigma}$ is the effective stress, $\dot{\bar{\varepsilon}}$ is the effective strain-rate, F_i represents surface tractions, and $E(\dot{\varepsilon}_{ij})$ is the work function. The solution of the original boundary value problem is then obtained from the solution of the dual variational problem, where the first order variation of the functional vanishes, namely,

$$\delta\pi = \int_{v} \bar{\sigma}\delta\dot{\bar{\varepsilon}} dV - \int_{S_F} F_i \delta u_i dS = 0 \tag{5.2}$$

where $\bar{\sigma} = \bar{\sigma}(\bar{\varepsilon})$ and $\bar{\sigma} = \bar{\sigma}(\bar{\varepsilon}, \dot{\bar{\varepsilon}})$ for rigid plastic and rigid viscoplastic materials, respectively. The incompressibility constraint on admissible velocity fields in Eq. 5.2 may be removed by introducing a Lagrange multiplier λ [67, 77] and modifying the functional by adding the term $\int \lambda \dot{\varepsilon}_v dV$, where $\dot{\varepsilon}_v = \dot{\varepsilon}_{ii}$, is the volumetric strain-rate. Then,

$$\delta\pi = \int_V \bar{\sigma}\delta\dot{\bar{\varepsilon}}dV + \int_V \lambda\delta\dot{\varepsilon}_v dV + \int_V \dot{\varepsilon}_v \delta\lambda dV - \int_{S_F} F_i \delta u_i dS = 0 \tag{5.3}$$

Another way of removing the constraint is to use the penalized form of the incompressibility as

$$\delta\pi = \int_V \bar{\sigma}\delta\dot{\bar{\varepsilon}}dV + K\int_V \dot{\varepsilon}_v \delta\dot{\varepsilon}_v dV - \int_{S_F} F_i \delta u_i dS = 0 \tag{5.4}$$

where K is a penalty constant, is a very large positive constant.

In Eq. 5.3 and 5.4, δu_i and $\delta\lambda$ are arbitrary variations and $\delta\dot{\bar{\varepsilon}}$ and $\delta\dot{\varepsilon}_v$ are the variations in strain-rate derived from δu_i. Equation 5.3 or 5.4 is the basic equation for the finite element formulation.

5.2.2 Treatment of a Rigid Region

The rigid zones are characterized by a very small value of effective strain-rate in comparison with effective strain-rate in the deforming body. If these portions are included within the control volume V, the value of the first term of the basic equation 5.3 or 5.4 cannot be uniquely determined because the undefined value of the effective stress when the effective strain-rate approaches zero. This is done by assuming that the stress strain-rate relationship approximated by

$$\dot{\varepsilon}_{ij} = \frac{3}{2}\frac{\dot{\bar{\varepsilon}}_0}{\bar{\sigma}_0}\sigma_{ij}' \qquad \text{with } \bar{\sigma}_0 = \bar{\sigma}(\bar{\varepsilon}, \dot{\bar{\varepsilon}}_0) \quad \text{for } \dot{\bar{\varepsilon}} \le \dot{\bar{\varepsilon}}_0$$

where $\dot{\bar{\varepsilon}}_0$ takes an assigned limiting value, say 10^{-3} [78]. This presumed stress strain-rate relationship is equivalent to the assumption of a Newtonian fluid like material behavior for nearly rigid regions. For these regions, the first term of the basic equation, $\int_V \bar{\sigma}\delta\dot{\bar{\varepsilon}}dV$, is then replaced by

$$\int\left(\frac{\bar{\sigma}_0}{\dot{\bar{\varepsilon}}_0}\right)\dot{\varepsilon}\delta\dot{\varepsilon}dV \qquad \text{for} \qquad \dot{\varepsilon} \le \dot{\varepsilon}_0 \tag{5.5}$$

Thus, the finite element discretization process is based on Eq. 5.3 or 5.4 with Eq. 5.5 for the regions are considered to be nearly rigid.

5.2.3 Finite Element Procedure

The procedure to the solution of problem formulated in finite element form is as follows. Discretization of a problem domain is the first step of finite element solution, which includes: (1) describing the element, (2) setting up the element equations and (3) assembling the element equations. Numerical analysis techniques are then applied for obtaining the solution of the global equations. The basis of the element equations and the assembling into global equations is described in Eq. 5.3 or 5.4

5.2.3.1 Governing Equations

The solution satisfying Eq. 5.2 is obtained from the admissible velocity fields that are constructed by introducing the shape functions in such a way that a continuous velocity field over each element can be defined uniquely in terms of velocities of the associated nodal points. In the deformation process, the workpiece is divided into elements without gaps or overlaps between elements. In order to ensure continuity of the velocities over the whole workpiece, the shape functions are defined such that the velocities along any shared element side are expressed in terms of velocity values at the same shared set of nodes (compatibility requirements). Then a continuous velocity field over the whole workpiece can be uniquely defined in terms of velocity values at nodal points specified globally.

The nodal point velocities are defined in a vector form as

$$\mathbf{v}^{\mathrm{T}} = \{v_1, v_2, v_3, \ldots, v_N\}$$

where the superscript T denotes transposition and N = (total number of nodes) × (degree of freedom per node).

An admissibility requirement for the velocity field is that the velocity boundary conditions prescribed in surface S_u (essential boundary conditions) must be satisfied. This condition can be imposed at nodes on S_u by assigning known values to the corresponding values to the corresponding variables. It is to be noted that the incompressibility condition is not required for defining a velocity field in the formulation Eq. 5.3 or 5.4

Equations 5.2 and 5.3 or 5.4 are now expressed in terms of nodal velocities v and their variation δv. From arbitrariness of δv_I a set of algebraic equations (stiffness equation) are obtained as

$$\frac{\partial \pi}{\partial v_I} = \sum_j \left(\frac{\partial \pi}{\partial v_I} \right)_{(j)} = 0 \qquad (5.6)$$

where (j) indicates the quantity at the j^{th} element. The capital letter suffix I signifies that it refers to the node number.

Equation 5.6 is obtained by evaluating the term $(\partial \pi / \partial v_I)$ at the elemental level and assembling them into the global equation under appropriate constraints.

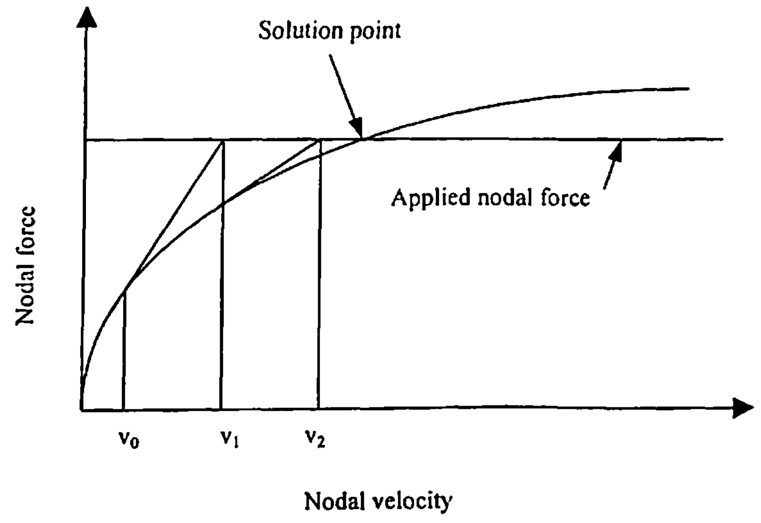
In large deformation problems, the stiffness equation is nonlinear and the solution is obtained iteratively using Newton-Raphson method. The method consists of linearization and application of convergence criteria to obtained the final solution. Linearization is achieved by Taylor expansion near an assumed solution point $v = v_0$ (initial guess), namely

$$\left[\frac{\partial \pi}{\partial v_I} \right]_{v=v_0} + \left[\frac{\partial^2 \pi}{\partial v_I \partial v_J} \right]_{v=v_0} \Delta v_J = 0 \qquad (5.7)$$

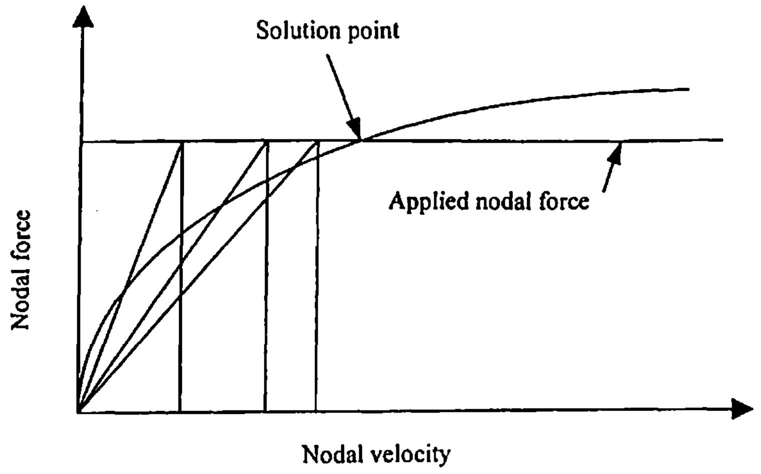
where Δv_j is the first order correction of the velocity v_0. Equation 5.7 can be written in the form

$$\mathbf{K}\,\Delta \mathrm{v} = \mathbf{f} \tag{5.8}$$

where **K** is called the stiffness matrix and **f** is the residual of the nodal point force vector.



(a) Schematic representation of Newton-Raphson method



(b) Schematic representation of direct iteration method

Fig. 5.1 Newton-Raphson method and direct iteration method for solution of non-linear equations

5.2.3.2 Newton-Raphson Method

Once the solution of Eq. 5.8 for the velocity correction term Δv is obtained, the assumed velocity v_0 is updated according to $v_0 + \alpha \Delta v$, where α is a constant between 0 and 1 called the deceleration coefficient. Iterations are continued until the velocity correction term becomes negligibly small. In the Newton-Raphson iterations [79], the initial guess velocity should be close to the actual solution for the convergence. When a deformation process is relatively simple, the initial guess velocity can be provided. However, if the process is complex and obtaining a good initial guess solution is difficult then the use of direct iteration method may be appropriate. Figure 5.1 (a) shows schematic representation of Newton-Raphson method.

5.2.3.3 Direct Iteration Method

Another technique for solving a nonlinear equation is the direct iteration method [80, 81]. In the direct iteration method, it is assumed that the constitutive equation is linear during each iteration and $\bar{\sigma}/\dot{\bar{\varepsilon}}$ is assumed to be constant during each iteration. The nonlinear friction term is also approximated by a linear relationship between the frictional stress and the relative sliding velocity. Then the stiffness equation resulting from $\delta\pi = 0$ becomes linear. Figure 5.1 (b) shows schematic representation of direct iteration method.

The computational process of the direct method is as follows:

1. Assign an assumed strain-rate $\dot{\bar{\varepsilon}}$ for each element. If a previous solution or iteration is not available, assign a constant average strain-rate to each element. If a previous solution or iteration is available, then use the strain-rate obtained previously for each element.
2. Assign an assumed sliding velocity to each element side that is in contact with a die. If a pervious solution or iteration is not available, assign a constant average sliding velocity to all relevant element sides. If it is available, use a sliding velocity that is obtained from previous solution or iteration.
3. Calculate $\bar{\sigma}/\dot{\bar{\varepsilon}}$ at each integration point of the element, where $\bar{\sigma}$ is evaluated for $\dot{\bar{\varepsilon}}$ assigned in step 1.

4. Calculate the viscous friction coefficient for each die contact side from the linear relationship between frictional stress and the relative sliding velocity.
5. Evaluate the stiffness matrix and obtain a velocity solution.
6. Calculate the strain-rate for each element by using the velocity solution of step 5.
7. Calculate the sliding velocity for each die contact element side.
8. Check whether solution converges, using convergence criteria.
9. If the solution does not converge, go to step 3.

It can be seen that for steps 1 and 2, the direct iteration method does not require any initial guess velocity. For large deformation problems, the direct iteration method converges fast towards the solution during the earlier stages of iteration. However, as the solution point is approached, the convergence becomes very slow. It seem that the best computational efficiency can be obtained by using (1) the direct iteration method for generating the initial guess and for case where the Newton-Raphson method does not converge, and (2) the Newton-Raphson for all other cases [63].

5.2.3.4 Convergence Criteria

Two convergence criteria are used in the developed code. One measures the error norm of the velocities, $\|\Delta v\|/\|v\|$, where the Euclidean vector is defined as $\|v\| = (v^T v)^{1/2}$, and requires such an error norm to decrease from iteration to iteration. The other criterion requires the norm of residual equations, $\|\partial\pi / \partial v\|$, to decrease.

In general terms, the first criterion is most useful in the early stages of iteration, when velocity field is still far from the solution. The second test is most useful when slightly ill conditioned systems reach the final stages of iterations. The final solution is considered to be achieved when the error norm reaches a specified small value, say 5×10^{-5}.

5.2.3.5 Solution Procedure

The finite element procedures outlined above are implemented in following manner.

1. Generate an assumed solution velocity.

2. Evaluate the element stiffness matrix for the velocity correction term Δv in Eq. 5.8.
3. Impose velocity conditions to the elemental stiffness matrix and repeat step 2 over all elements defined in the workpiece.
4. Assemble elemental stiffness matrix to form a global stiffness equation.
5. Obtain the velocity correction terms by solving the global stiffness equation.
6. Update the assumed solution velocity by adding the correctional term to the assumed velocity. Repeat steps 2 through 6 until the velocity solution converges.
7. When the converged velocity solution is obtained, update the geometry of workpiece using the velocity of nodes during a time increment. Steps 2 through 7 are repeated until the desired degree of deformation is achieved.

5.2.4 Elements and Shape Functions

In the developed code, two types of elements namely three noded triangular element and four noded rectangular element, are used to discretize the problem domain. The geometry of an element is uniquely defined by a finite number of nodal points or nodes. The nodes are located on the boundary of the elements and the shape functions define an admissible velocity field locally in terms of velocities of associated nodes. Thus, elements are characterized by the shape and order of shape functions.

In the finite element method, interpolation of a scalar function $f(x,y)$ defined over an element is introduced in a form

$$f(x,y) = \sum q_\alpha(x,y) \cdot f_\alpha \qquad (5.9)$$

where f_α is a function value associated with α^{th} node, and $q_\alpha(x,y)$ is the shape function. Generally, it is a polynomial function of x and y defined over the element in such a way that

$$q_\alpha(x_\beta, y_\beta) = \delta_{\alpha\beta} \qquad (5.10)$$

where (x_β, y_β) is the coordinates of β^{th} node and $\delta_{\alpha\beta}$ is the Kronecker delta. Owing the property of the shape function given by Eq. 5.10, f_α in Eq. 5.9 has the value of the function f at (x_β, y_β) and the f_α are independent of each other.

5.2.4.1 Triangular Element

In the three noded triangular element, it is convenient to define shape functions in the area coordinate system L_1, L_2, L_3. The area coordinates for a triangle, as shown in the Fig. 5.2 (a), are defined by the following linear relations

$$\begin{aligned} x &= L_1 x_1 + L_2 x_2 + L_3 x_3 \\ y &= L_1 y_1 + L_2 y_2 + L_3 y_3 \\ L_1 &+ L_2 + L_3 = 1 \end{aligned} \tag{5.11}$$

where (x_α, y_β) are the coordinates of a corner of the triangle. It can be readily shown that an alternative definition of the coordinate of point P can be given by the ratio of the shaded triangle to that of the total triangle as

$$L_1 = \frac{\text{area } P23}{\text{area } 123}, \qquad L_2 = \frac{\text{area } P13}{\text{area } 123}, \qquad \text{and} \qquad L_3 = \frac{\text{area } P12}{\text{area } 123}$$

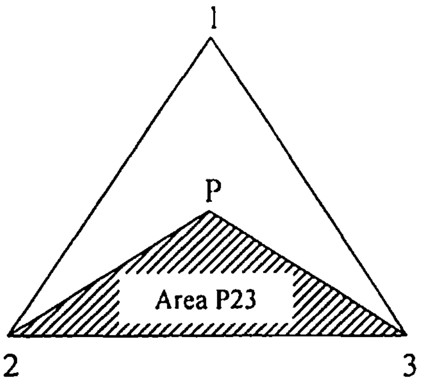
Solving Eq. 5.11 for L_1, L_2, and L_3 gives

$$L_1 = (a_1 + b_1 x + c_1 y)/2\Delta, \qquad L_2 = (a_2 + b_2 x + c_2 y)/2\Delta, \qquad L_3 = (a_3 + b_3 x + c_3 y)/2\Delta$$

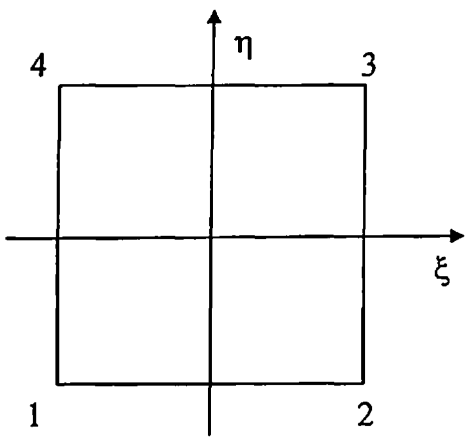
where $\Delta =$ area of (123)

and

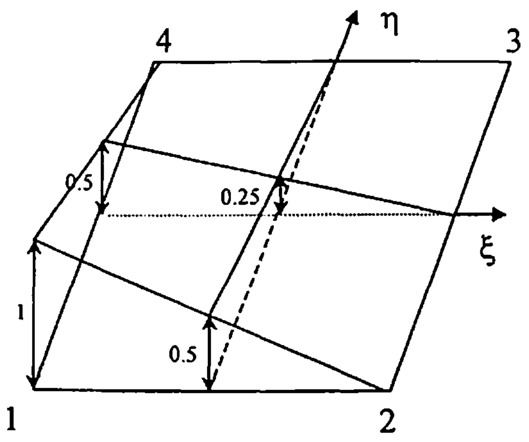
$$\begin{aligned} a_1 &= x_2 y_3 - x_3 y_2 & a_2 &= x_3 y_1 - x_1 y_3 & a_3 &= x_1 y_2 - x_2 y_1 \\ b_1 &= y_2 - y_3 & b_2 &= y_3 - y_1 & b_3 &= y_1 - y_2 \\ c_1 &= x_3 - x_2 & c_2 &= x_1 - x_3 & c_3 &= x_2 - x_1 \end{aligned} \tag{5.12}$$



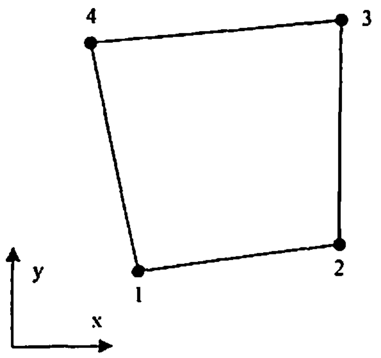
(a) Area coordinate system of the triangular element



(b) Natural coordinate system for rectangular element



(c) Shape functions for rectangular element



(d) Cartesian coordinate system for rectangular element

Fig. 5.2 Coordinate systems for triangular and rectangular elements

5.2.4.2 Rectangular Element

The shape functions of rectangular elements are defined in a parametric form over a domain $-1 \le \xi \le 1$, $-1 \le \eta \le 1$ in a natural coordinate system (ξ,η). The element defined in the natural coordinate system is called as parent element. The simplest of the rectangular elements is the used four-node linear element shown in Fig. 5.2(b). The shape function, q_α, which are bilinear in ξ and η, are defined as

$$q_\alpha(\xi,\eta) = \tfrac{1}{4}(1+\xi_\alpha\xi)(1+\eta_\alpha\eta) \tag{5.13}$$

where (ξ_α,η_α) are the natural coordinates of a node at one of its corners. The value of the shape functions, given by Eq. 5.13 are shown schematically in Fig. 5.2(c). Admissible velocity fields can be defined uniquely over the rectangular element by the nodal velocity components as

$$\begin{aligned} u_x(\xi,\eta) &= \sum_\alpha q_\alpha(\xi,\eta)u_x^{(\alpha)} \\ u_y(\xi,\eta) &= \sum_\alpha q_\alpha(\xi,\eta)u_y^{(\alpha)} \end{aligned} \tag{5.14}$$

where $(u_x^{(\alpha)},u_y^{(\alpha)})$ is the velocity at the α th node and summation is over all four nodes.

Coordinate transformation from the natural coordinate (ξ, η) to the global coordinate (x,y) is defined by

$$\begin{aligned} x(\xi.\eta) &= \sum_\alpha q_\alpha(\xi,\eta)x_\alpha \\ y(\xi.\eta) &= \sum_\alpha q_\alpha(\xi,\eta)y_\alpha \end{aligned} \tag{5.15}$$

where (x_α,y_α) are the global coordinates of the α th node. Since the coordinate transformation (Eq. 5.15) uses the same shape function (Eq. 5.14), the linear element is isoparametric and takes quadrilateral shape in the cartesian map, as shown in the Fig. 5.2(d).

5.2.5 Element Strain-Rate Matrix

The strain-rate components in cartesian coordinate system are defined by

$$\dot{\varepsilon}_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial u_i}\right) \tag{5.16}$$

The admissible velocity for all type of elements can be express by

$$u_i = \sum_{\alpha} q_{\alpha} u_i^{(\alpha)} \tag{5.17}$$

substituting Eq. 5.17 into Eq. 5.16 we have

$$\dot{\varepsilon}_{ij} = \frac{1}{2}\sum_{\alpha}\left(\frac{\partial q_{\alpha}}{\partial x_j} u_i^{(\alpha)} + \frac{\partial q_{\alpha}}{\partial x_i} u_j^{(\alpha)}\right) \tag{5.18}$$

It is seen from Eq. 5.18 that strain-rate components can be evaluated if $\partial q_{\alpha} / \partial x_i$ is known.

For the cartesian coordinate system, we denote the coordinate x_i by (x, y, z) for three-dimensional deformation, by (r, z, θ) for axisymmetric deformation and by (x, y) for two-dimensional deformation.
Let X_{α}, Y_{α} and Z_{α} be defined as

$$X_{\alpha} = \frac{\partial q_{\alpha}}{\partial x}, \quad Y_{\alpha} = \frac{\partial q_{\alpha}}{\partial y}, \quad Z_{\alpha} = \frac{\partial q_{\alpha}}{\partial z} \tag{5.19}$$

Then the strain-rate components given by Eq. 5.18 are expressed by

$$\begin{aligned}
\dot{\varepsilon}_x &= \sum X_\alpha u_x^{(\alpha)}, & \dot{\varepsilon}_{xy} &= \frac{1}{2}\sum\left(Y_\alpha u_x^{(\alpha)} + X_\alpha u_y^{(\alpha)}\right) \\
\dot{\varepsilon}_y &= \sum Y_\alpha u_y^{(\alpha)}, & \dot{\varepsilon}_{yz} &= \frac{1}{2}\sum\left(Z_\alpha u_y^{(\alpha)} + Y_\alpha u_z^{(\alpha)}\right) \qquad (5.20) \\
\dot{\varepsilon}_z &= \sum Z_\alpha u_z^{(\alpha)}, & \dot{\varepsilon}_{zx} &= \frac{1}{2}\sum\left(X_\alpha u_z^{(\alpha)} + Z_\alpha u_x^{(\alpha)}\right)
\end{aligned}$$

It is convenient to arrange the strain-rate components in a vector form. For two-dimensional elements the strain-rate components can be written as

$$\dot{\varepsilon} = \begin{Bmatrix} \dot{\varepsilon}_x \\ \dot{\varepsilon}_y \\ \dot{\varepsilon}_z \\ \dot{\gamma}_{xy} \end{Bmatrix} = \begin{Bmatrix} \dfrac{\partial u_x}{\partial x} \\ \dfrac{\partial u_y}{\partial y} \\ 0 \\ \dfrac{\partial u_y}{\partial x} + \dfrac{\partial u_x}{\partial y} \end{Bmatrix} \quad \text{for plane strain problems}$$

and (5.21)

$$\dot{\varepsilon} = \begin{Bmatrix} \dot{\varepsilon}_r \\ \dot{\varepsilon}_z \\ \dot{\varepsilon}_\theta \\ \dot{\gamma}_{rz} \end{Bmatrix} = \begin{Bmatrix} \dfrac{\partial u_r}{\partial r} \\ \dfrac{\partial u_z}{\partial z} \\ \dfrac{u_r}{r} \\ \dfrac{\partial u_z}{\partial r} + \dfrac{\partial u_r}{\partial z} \end{Bmatrix} \quad \text{for axisymmetric problems}$$

Substituting Eq. 5.20 into Eq. 5.21 the strain-rate vectors are represented in a unified form as

$$\dot{\varepsilon} = \begin{Bmatrix} \dot{\varepsilon}_1 \\ \dot{\varepsilon}_2 \\ \dot{\varepsilon}_3 \\ \dot{\varepsilon}_4 \end{Bmatrix} = \begin{Bmatrix} \sum X_\alpha u_1^{(\alpha)} \\ \sum Y_\alpha u_2^{(\alpha)} \\ \sum P_\alpha u_1^{(\alpha)} \\ \sum X_\alpha u_2^{(\alpha)} + Y_\alpha u_1^{(\alpha)} \end{Bmatrix} \qquad (5.22)$$

In Eq. 5.22, $\dot{\varepsilon}_1$, $\dot{\varepsilon}_2$, $\dot{\varepsilon}_3$, $\dot{\varepsilon}_4$ are $\dot{\varepsilon}_x$, $\dot{\varepsilon}_y$, $\dot{\varepsilon}_z$, $\dot{\gamma}_{xy}$ and $\dot{\varepsilon}_r$, $\dot{\varepsilon}_z$, $\dot{\varepsilon}_\theta$, $\dot{\gamma}_{rz}$ for plane strain and axisymmetric problem respectively. Velocity components u_1 and u_2 correspond to u_x and u_y respectively, for two-dimensional deformation and P_a is zero for plane strain problem. For the axisymmetric case, u_1 and u_2 represents u_r and u_z respectively and P_a becomes q_a / r.

Equation 5.22 can be written in the matrix form as

$$\dot{\boldsymbol{\varepsilon}} = \mathbf{B}\,\mathbf{v} \tag{5.23}$$

where **B** is called the strain-rate matrix and written as

$$\mathbf{B} = \begin{bmatrix} X_1 & 0 & X_2 & 0 & X_3 & 0 & X_4 & 0 & \cdots \\ 0 & Y_1 & 0 & Y_2 & 0 & Y_3 & 0 & Y_4 & \cdots \\ P_1 & 0 & P_2 & 0 & P_3 & 0 & P_4 & 0 & \cdots \\ Y_1 & X_1 & Y_2 & X_2 & Y_3 & X_3 & Y_4 & X_4 & \cdots \end{bmatrix} \tag{5.24}$$

The number of columns of the matrix **B** is determined by the number of degrees of freedom allowed to the element.

The evaluation of the strain-rate matrix requires the differentiation of shape functions with respect to the global coordinate. Since the shape functions are expressed in the natural coordinate system, it is necessary to express the global derivatives in terms of the derivatives with respect to the natural coordinate. Consider a coordinate transformation where shape functions are defined in the natural coordinate system, then the derivatives of the shape functions with respect to the natural coordinate system can be expressed as

$$\begin{Bmatrix} \partial q_a / \partial \xi \\ \partial q_a / \partial \eta \\ \partial q_a / \partial \zeta \end{Bmatrix} = \mathbf{J} \begin{Bmatrix} \partial q_a / \partial x \\ \partial q_a / \partial y \\ \partial q_a / \partial z \end{Bmatrix} \tag{5.25}$$

where **J** is the Jacobian matrix of the coordinate transformation, given by

$$\mathbf{J} = \begin{bmatrix} \partial x/\partial\xi & \partial y/\partial\xi & \partial z/\partial\xi \\ \partial x/\partial\eta & \partial y/\partial\eta & \partial z/\partial\eta \\ \partial x/\partial\zeta & \partial y/\partial\zeta & \partial z/\partial\zeta \end{bmatrix} \tag{5.26}$$

Then the derivatives in Eq. 5.19 can be obtained as

$$\begin{Bmatrix} X_\alpha \\ Y_\alpha \\ Z_\alpha \end{Bmatrix} = \begin{Bmatrix} \partial q_\alpha/\partial x \\ \partial q_\alpha/\partial y \\ \partial q_\alpha/\partial z \end{Bmatrix} = \mathbf{J}^{-1} \begin{Bmatrix} \partial q_\alpha/\partial\xi \\ \partial q_\alpha/\partial\eta \\ \partial q_\alpha/\partial\zeta \end{Bmatrix} \tag{5.27}$$

where $\mathbf{J}^{-1}$ is the inverse of **J**

It may be mentioned that in plane strain deformation, the strain-rate $\dot{\varepsilon}_3$ is not necessary, since it is always zero. However, it is convenient to include $\dot{\varepsilon}_3$, in Eq. 5.23 so that the strain-rate matrix **B** of the plane strain deformation has the same form as that of the axisymmetric deformation as shown in Eq. 5.24.

5.2.5.1 Triangular Element

From the Fig. 5.2 (a), the shape functions q_α for a linear triangular element are given by

$$q_1 = L_1, \qquad q_2 = L_2, \qquad q_3 = L_3 \tag{5.28}$$

The strain-rate matrix of the triangular element can be derived by applying Eq. 5.27 to the shape functions given in Eq. 5.28. Since the area coordinates are not independent of each other, we can eliminate L_3 from the expression of q_α by using $L_3 = 1 - L_1 - L_2$. Equation 5.27 can be written for triangular element as

$$\begin{Bmatrix} \dfrac{\partial q_\alpha}{\partial x} \\ \dfrac{\partial q_\alpha}{\partial y} \end{Bmatrix} = \frac{1}{|\mathbf{J}|} \begin{bmatrix} \dfrac{\partial y}{\partial L_2} & -\dfrac{\partial y}{\partial L_1} \\ -\dfrac{\partial x}{\partial L_1} & \dfrac{\partial x}{\partial L_1} \end{bmatrix} \begin{Bmatrix} \dfrac{\partial q_\alpha}{\partial L_1} \\ \dfrac{\partial q_\alpha}{\partial L_2} \end{Bmatrix} \tag{5.29}$$

where $|\mathbf{J}|$ is the determinant of the Jacobian matrix and expressed by

$$|\mathbf{J}| = \frac{\partial x}{\partial L_1}\frac{\partial y}{\partial L_2} - \frac{\partial x}{\partial L_2}\frac{\partial y}{\partial L_1} \tag{5.30}$$

The strain-rate matrix of a linear triangular element, shown in Fig. 5.2 (a) can be obtained in a closed form by substituting $q_1 = L_1$, $q_2 = L_2$, $q_3 = 1 - L_1 - L_2$ and is written as follows

$$\begin{aligned} X_1 &= \frac{1}{|\mathbf{J}|}(y_2 - y_3), \quad X_2 = \frac{-1}{|\mathbf{J}|}(y_1 - y_3), \quad X_3 = -X_1 - X_2 \\ Y_1 &= \frac{-1}{|\mathbf{J}|}(x_2 - x_3), \quad Y_2 = \frac{1}{|\mathbf{J}|}(x_1 - x_3), \quad Y_3 = -Y_1 - Y_2 \end{aligned} \tag{5.31}$$

where

$$|\mathbf{J}| = (x_1 - x_3)(y_2 - y_3) - (x_2 - x_3)(y_1 - y_3) \tag{5.32}$$

Note that $|\mathbf{J}|$ is twice of the area of the triangle.

5.2.5.2 Rectangular Element

For the rectangular element X_α and Y_α in Eq. 5.22 can be written as

$$\begin{Bmatrix} X_\alpha \\ Y_\alpha \end{Bmatrix} = \frac{1}{|\mathbf{J}|}\begin{bmatrix} \frac{\partial y}{\partial \eta} & -\frac{\partial y}{\partial \xi} \\ -\frac{\partial x}{\partial \eta} & \frac{\partial x}{\partial \xi} \end{bmatrix}\begin{Bmatrix} \frac{\partial q_\alpha}{\partial \xi} \\ \frac{\partial q_\alpha}{\partial \eta} \end{Bmatrix} \tag{5.33}$$

where $|\mathbf{J}|$ is the determinant of Jacobian matrix of Eq. 5.15 and is expressed by

$$|\mathbf{J}| = \frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta} - \frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \xi} \tag{5.34}$$

For a quadrilateral element with the number of nodes shown in Fig. 5.2(c) X_α, Y_α and $|\mathbf{J}|$ can be expressed in the closed form as

$$\begin{Bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{Bmatrix} = \frac{1}{8|\mathbf{J}|} \begin{Bmatrix} +y_{24} - y_{34}\xi - y_{23}\eta \\ -y_{13} + y_{34}\xi + y_{14}\eta \\ -y_{24} + y_{12}\xi - y_{14}\eta \\ +y_{13} - y_{12}\xi + y_{23}\eta \end{Bmatrix}$$

and (5.35)

$$\begin{Bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{Bmatrix} = \frac{1}{8|\mathbf{J}|} \begin{Bmatrix} -x_{24} + x_{34}\xi + x_{23}\eta \\ +x_{13} - x_{34}\xi - x_{14}\eta \\ +x_{24} - x_{12}\xi + x_{14}\eta \\ -x_{13} + x_{12}\xi - x_{23}\eta \end{Bmatrix}$$

and $|\mathbf{J}|$ is expressed as

$$|\mathbf{J}| = 1/8\left[(x_{13} \cdot y_{24} - x_{24} \cdot y_{13}) + (x_{34} \cdot y_{12} - x_{12} \cdot y_{34})\xi + (x_{23} \cdot y_{14} - x_{14} \cdot y_{23})\eta\right] \tag{5.36}$$

where $x_{ij} = x_i - x_j$ and $y_{ij} = y_i - y_j$

5.2.6 Matrices of Effective Strain-Rate and Volume Strain-Rate

In the finite element formulation for the analysis of large deformation process, the effective strain-rate $\dot{\bar{\varepsilon}}$ and the volumetric strain-rate $\dot{\varepsilon}_v$ are frequently used. Therefore, it is necessary to express the effective strain-rate and volumetric strain-rate in terms of the strain-rate matrix.

The effective strain-rate is defined in terms of strain-rate components as

$$\dot{\bar{\varepsilon}} = \sqrt{2/3}\left\{\dot{\varepsilon}_{ij}\dot{\varepsilon}_{ij}\right\}^{1/2} \tag{5.37}$$

or, in the matrix form

$$\left(\dot{\bar{\varepsilon}}\right)^2 = \dot{\varepsilon}^{\mathrm{T}} \mathbf{D} \dot{\varepsilon} \tag{5.38}$$

The diagonal matrix **D** has 2/3 and 1/3 as its components; corresponding to normal strain-rate and engineering shear strain-rate, respectively. Substitution of Eq. 5.23 into Eq. 5.38 gives

$$\left(\dot{\bar{\varepsilon}}\right)^2 = \mathrm{v}^{\mathrm{T}}\, \mathbf{B}^{\mathrm{T}}\, \mathbf{D}\,\mathbf{B}\, \mathrm{v} = \mathrm{v}^{\mathrm{T}}\, \mathbf{P}\, \mathrm{v} \tag{5.39}$$

where $\mathbf{P} = \mathbf{B}^{\mathrm{T}}\, \mathbf{D}\,\mathbf{B}$

The matrix **D** in Eq. 5.38 takes different forms depending upon the expression of the effective strain-rate, in terms of strain-rate components. The expression of the effective strain-rate also depends on the yield criteria.

The volumetric strain-rate $\dot{\varepsilon}_v$ is given by

$$\dot{\varepsilon}_v = \dot{\varepsilon}_x + \dot{\varepsilon}_y + \dot{\varepsilon}_z \tag{5.40}$$

and expressed by

$$\dot{\varepsilon}_v = \mathrm{C}^{\mathrm{T}}\, \mathrm{v} = C_I v_I \tag{5.41}$$

with $C_I = B_{1I} + B_{2I} + B_{3I}$ where B_{IJ} is an element of the strain-rate matrix **B**.

5.2.7 Elemental Stiffness Equation

It can easily be seen from the way in which the element was introduced that the global integrals over the whole workpiece stem from the assembly of integrals over the local domain of disjoint finite elements. Therefore, it is convenient to evaluate the stiffness matrix given by Eq. 5.7 at the element level, and to assemble into a global stiffness matrix.

In the penalty function method, (Eq. 5.4) denote the first, second and third term (including signs) of Eq. 5.4 with $\delta\pi_D$, $\delta\pi_P$, $\delta\pi_{S_F}$ respectively. In large deformation process, the boundary conditions along the die-workpiece interface are mixed. Therefore, along the interface S_C the treatment of traction depends on the friction representation.

Using the discrete representation of the quantities involved in $\delta\pi$ that are developed in sections 5.2.4 and 5.2.5, we can express the integral of $\delta\pi$ in terms of nodal-point velocities. Eq. 5.7 becomes

$$\frac{\partial\pi}{\partial v_I}=\frac{\partial\pi_D}{\partial v_I}+\frac{\partial\pi_P}{\partial v_I}+\frac{\partial\pi_{S_F}}{\partial v_I}$$

where,

$$\frac{\partial\pi_D}{\partial v_I}=\int_v\frac{\bar{\sigma}}{\dot{\bar{\varepsilon}}}P_{IJ}v_J\,dV$$
$$\frac{\partial\pi_P}{\partial v_I}=\int_v KC_Jv_JC_I\,dV \tag{5.42}$$
$$\frac{\partial\pi_{S_F}}{\partial v_I}=-\int_{S_F}F_jN_{jI}\,dS$$

It should be noted that the term $(-\partial\pi_{S_F}/\partial v_I)$ is the applied nodal point force and that $\partial\pi_D/\partial v_I+\partial\pi_P/\partial v_I$ is the reaction nodal point force.

The second derivatives of π are expressed as

$$\frac{\partial^2\pi}{\partial v_Iv_J}=\int_v\frac{\bar{\sigma}}{\dot{\bar{\varepsilon}}}P_{IJ}\,dV+\int_v\left(\frac{1}{\dot{\bar{\varepsilon}}}\frac{\partial\bar{\sigma}}{\partial\dot{\bar{\varepsilon}}}-\frac{\bar{\sigma}}{\dot{\bar{\varepsilon}}^2}\right)\frac{1}{\dot{\bar{\varepsilon}}}P_{IK}v_Kv_MP_{MJ}\,dV+\int_v KC_JC_I\,dV \tag{5.43}$$

Evaluating stiffness matrices at the elemental level from Eq. 5.42 and Eq. 5.43 assembling them for the whole workpiece, we obtain a set of simultaneous linear equations (Eq. 5.8).

When effective strain-rate $\dot{\bar{\varepsilon}}$ approaches zero or becomes less than a preassigned value $\dot{\bar{\varepsilon}}_0$, we have

$$\delta\pi_D = \int_v \bar{\sigma}\delta\dot{\bar{\varepsilon}}dV = \int_v \frac{\bar{\sigma}_0}{\dot{\bar{\varepsilon}}_0}\dot{\bar{\varepsilon}}\delta\dot{\bar{\varepsilon}}dV \tag{5.44}$$

where $\bar{\sigma}_0 / \dot{\bar{\varepsilon}}_0$ = constant. The derivatives of π_D can be expresses by

$$\begin{aligned} \frac{\partial\pi_D}{\partial v_I} &= \int_v \frac{\bar{\sigma}_0}{\dot{\bar{\varepsilon}}_0} P_{IJ} v_J dV \\ \frac{\partial^2\pi_D}{\partial v_I \partial v_J} &= \int_v \frac{\bar{\sigma}_0}{\dot{\bar{\varepsilon}}_0} P_{IJ} dV \end{aligned} \tag{5.45}$$

The penalty constant K and the limiting strain-rate $\dot{\bar{\varepsilon}}_0$ are introduced rather arbitrarily for computational convenience. However, proper choices of these two constants are important in successful simulation of large deformation processes. According to literature [63] a large value of K is preferred to keep the volumetric strain-rate close to zero. However, too large value of K may cause difficulties in convergence, while too small values of K results in unacceptable large volumetric strain. Numerical tests show that an appropriate K value can be estimated by restricting volumetric strain-rate is 0.0001-0.001 times the average effective strain-rate.

The limiting strain-rate, $\dot{\bar{\varepsilon}}_0$ under which the material is considered to be rigid, has been introduced to improve the numerical behavior of the rigid-plastic formulation [78]. Too large value of limiting strain-rate result in a solution with a rigid zone of unacceptably large strain-rate. On the other hand, if we choose too small value of limiting strain-rate, then the convergence of the Newton-Raphson method deteriorates considerably. Numerical tests show that an optimal result can be obtained by choosing the limiting strain-rate 1/100 of the average effective strain-rate [63].

5.2.8 Boundary Conditions

Since the boundary condition along the die-workpiece interface S_C is mixed, it is convenient to write the boundary surface S in three distinct parts.

$$S = S_u + S_F + S_C$$

The traction boundary condition on S_F is either zero traction or ordinarily at most a uniform hydrostatic pressure. However the boundary conditions along the other interface are mixed. Generally, neither velocity nor force can be prescribed completely along this interface, because the direction of the frictional stress is opposite to the direction of the relative velocity is not known a priori. Situations exist in which the direction of deformation in the deforming zone relative to the undeformed portion is known. This class of problem can be solved if the magnitude of the frictional stress f_s is given according to the well known Coulomb law, $f_s = \mu p$, or the friction law of constant factor m, expressed by $f_s = mk$ (where $k = Y/\sqrt{3}$) here, p is the die pressure and k is the shear yield stress.

It is difficult to handle the boundary conditions in a straightforward manner in problems in which the direction of the relative velocity between the compressing die and deforming material interfaces is unknown. In order to deal with these situations, a velocity dependent frictional stress is used as an approximation to the conditions to constant frictional stress. At the interface S_C the velocity boundary conditions are given in the direction normal to the interface by the die velocity, and the traction boundary condition is expressed by

$$\mathrm{f_s} = mkl \cong mk\left\{\frac{2}{\pi}\tan^{-1}\left(\frac{u_s}{u_0}\right)\right\}l \qquad (5.46)$$

where $\mathrm{f_s}$ is the frictional stress, l is the unit vector in the opposite direction of relative sliding, u_s is the sliding velocity of a material relative to the die velocity, and u_0 is a small positive number compared to u_s. The approximate expression (Eq. 5.46) for a constant frictional stress has been used for the smooth transition of the frictional stress in the range near neutral point [82].

Imposition of the traction boundary conditions on S_F is straightforward. Recalling the boundary integral $\delta\pi_{S_F}$ with respect to a node velocity component, the traction boundary conditions is imposed in the form of nodal point forces. It should be mentioned that the same nodal point force could be obtained with different traction distribution.

The velocity boundary conditions on S_u are essential boundary conditions. In the finite element discretization, the velocity boundary condition is enforced only at nodes on S_u, and the velocity along the element side is determined automatically in terms of velocities of the nodes and element shape functions. For the node at which the velocity is defined, the velocity correction Δv_M is zero. Consequently, the corresponding stiffness equation should be removed. The simplest way to implement this procedure is to replace the corresponding rows and columns by zeros and to set the diagonal terms to 1 as shown below

$$\begin{bmatrix} K_{11} & K_{12} & \cdots & 0 & \cdots & K_{1n} \\ K_{21} & K_{22} & \cdots & 0 & \cdots & K_{2n} \\ \cdot & \cdot & \cdots & 0 & \cdots & \cdot \\ 0 & 0 & \cdots & 1 & \cdots & 0 \\ \cdot & \cdot & \cdots & 0 & \cdots & \cdot \\ K_{n1} & K_{n2} & \cdots & 0 & \cdots & K_{nn} \end{bmatrix} \begin{Bmatrix} \Delta v_1 \\ \Delta v_2 \\ \\ \Delta v_M \\ \\ \Delta v_n \end{Bmatrix} = \begin{Bmatrix} f_1 \\ f_2 \\ \cdot \\ 0 \\ \cdot \\ f_n \end{Bmatrix}$$

On the surface S_C, the traction is prescribed in the tangential direction and the velocity is prescribed in the normal direction to the interface. When the interface direction is inclined with respect to the global coordinate axis, the coordinate transformation of the stiffness matrix upon the inclined direction is necessary in order to impose mixed boundary conditions.

Consider a velocity vector **v** in the global coordinate system and the corresponding vector **v'** in the inclined boundary coordinates. Then the vector is transformed from the global to the local coordinate system by

$$\mathbf{v}' = \mathbf{T}\,\mathbf{v} \tag{5.47}$$

where **T** is the coordinate transform matrix.

Similarly, the nodal point force vector **f** is transformed to **f'** according to

$$\mathbf{f}' = \mathbf{T}\,\mathbf{f} \tag{5.48}$$

in the two-dimensional coordinate system, the transformation matrix of node *I* is written as

$$\mathbf{T}_I = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

where θ is measured from the *x*-axis in the global coordinate system to the x' axis of the local coordinate system in counterclockwise direction. The stiffness equation (Eq. 5.8) is transformed to

$$\mathbf{TKT}^{\mathrm{T}}\Delta\mathbf{v}' = \mathbf{f}' \tag{5.49}$$

The velocity boundary condition at the tool-workpiece interface is given by

$$\mathbf{u}_n = \mathbf{U}_{\mathrm{D}}^{\mathrm{T}}\,\mathbf{n}$$

where $\mathbf{U}_{\mathrm{D}}$ is the tool velocity and **n** is the unit normal to the interface surface

In the direction of the relative sliding between the die and the workpiece, the frictional stress f_s is prescribed as the traction boundary condition. The frictional stress is usually represented according to the Coulomb law or as a constant frictional stress. The friction represented by a constant friction factor m is approximated by Eq. 5.46 in order to deal with neutral-point problems in large deformation process.

The equation (Eq. 5.46) expresses that the magnitude of the frictional stress is dependent on the magnitude of the relative sliding and that their directions are opposite to each other. Then, the relationship can be written as

$$f_s = -mk\frac{u_s}{|u_s|} \cong -mk\left(\frac{2}{\pi}\tan^{-1}\left[\frac{u_s}{u_0}\right]\right) \tag{5.50}$$

The approximation of the frictional stress by the arctangent function of the relative sliding velocity eliminates the sudden change of direction of the frictional stress mk at the neutral point. The literature [63] shows that the frictional stress approaches mk asymptotically as the relative sliding velocity u_s increases. However, the frictional stress f_s approximated by Eq. 5.50 deviates considerably from the value of mk as u_s approaches zero. It may be noted that the value of u_0 was introduced arbitrarily for performing numerical calculations and that the choice of u_0 could have a significant influence on the reliability of the solution. It is seen from the literature [63] that the ratio u_s / u_0 should be equal to or larger than 10 in order to attain the friction value within 9% of the one originally intended. On the other hand, if we choose the ratio too large, then the sudden change of the frictional stress near the neutral point can cause difficulties in numerical calculations. Since the order of magnitude of u_s is 0.1 with the unit die velocity, a recommended value of u_0 is 10^{-3} to 10^{-4} [82].

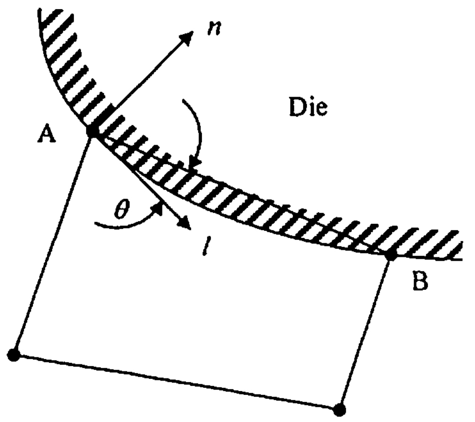


Fig. 5.3 An element in contact with die

For the discretization, consider a die and an element that is in contact with the die, as shown in Fig. 5.3. The boundary condition normal to the contact surface is enforced at the contact nodes. Also, the relative sliding velocity at the nodes v_s can be evaluated. It should be noted that the element-side cannot be made to conform the die surface. However, it may be assumed that the relative sliding velocity u_s can be approximated in terms of nodal-point values $v_{s\alpha}$ by using a shape function of the element as

$$u_s = \sum_{\alpha} q_\alpha v_{s\alpha} \tag{5.51}$$

where the subscript α denotes the value at α^{th} node.

In deriving the stiffness equation, $\delta\pi$, include the term $\delta\pi_{S_C}$, and the final form of the stiffness equation should contain the terms

$$\frac{\delta\pi_{S_C}}{\partial v_\alpha} = \int_{S_C} mk\frac{2}{\pi} q_\alpha \tan^{-1}\left[q_\beta v_{S_\beta} / u_0\right] dS \tag{5.52}$$

and

$$\frac{\partial^2\pi_{S_C}}{\partial v_\alpha \partial v_\beta} = \int_{S_C} mk\frac{2}{\pi} q_\alpha q_\beta \left(\frac{u_0}{u_0^2 + (q_k v_{sk})^2}\right) dS \tag{5.53}$$

The finite element method approximation of the boundary conditions introduces errors to the solution of the boundary value problem. Note that the surface integration in Eq. 5.52 and 5.53 is carried out over the element surface rather than the actual die surface. When linear elements are used with a curved die, the interface area represented by element is always smaller than the actual interface area, and the effect of friction in the analysis is always smaller than the actual. For deformation processes that are sensitive to friction, this type of error could be quite serious.

Also, the velocity boundary condition imposed by the finite element model can be considerably different from that of the actual problem. In expressing the sliding velocity by Eq. 5.51 it was assumed that the mismatch angle between the element side and the tangent direction of the die at contact node is small. When this angle of mismatch is large, the deformation mode is not modeled correctly. The errors resulting from the boundary conditions imposed by the finite element method can be minimized by increasing the number of elements at the boundary.

5.2.9 Time Increment and Geometry Updating

When the solution of velocity is obtained, then the deformed geometry of the workpiece, in the case of two-dimensions can be obtained by updating the coordinates of the nodes.

$$
\begin{aligned}
x_I(t_0+\Delta t) &= x_I(t_0) + u_x^{(I)}\Delta t \\
y_I(t_0+\Delta t) &= y_I(t_0) + u_y^{(I)}\Delta t
\end{aligned}
\tag{5.54}
$$

where (x_I, y_I) are the coordinates of I th node, t_0 is the time at current configuration, and Δt is the time increment. The strain is updated in a similar manner from the strain-rate solution. In general the time increment Δt can be determined by considering several factors, such as the time (Δt_{die}) necessary for a next free node to contact the die surface, a desired maximum strain increment (Δt_{strain}), and a maximum allowable time increment (Δt_a). The actual time increment is determined by taking the minimal of (Δt_{die}), (Δt_{strain}) and (Δt_a). The time necessary for a next free node to contact the die can be determined by calculating these time increments for all free nodes and choosing the minimal time increment. The time increment required to limit the maximum strain increment cab be readily obtained from strain-rate solutions. The maximum allowable time increment is given rather arbitrarily. However, consideration of the error in the volume constancy is a factor for determining its magnitude.

5.2.10 Stress-Strain Computations

After computing the solution of every step, the strain-rate components are calculated as

$$
\dot{\varepsilon} = \mathbf{B}\,\mathbf{v}
\tag{5.55}
$$

where **B** is the strain-rate matrix, and the **v** is the solution velocity vector.

The effective strain-rate is computed using following expression

$$\dot{\bar{\varepsilon}} = \left(\dot{\boldsymbol{\varepsilon}}^{T}\mathbf{D}\dot{\boldsymbol{\varepsilon}}\right)^{1/2} \qquad (5.56)$$

where **D** is effective strain-rate coefficient matrix and is defined as

$$\mathbf{D} = \begin{bmatrix} \frac{2}{3} & 0 & 0 & 0 \\ 0 & \frac{2}{3} & 0 & 0 \\ 0 & 0 & \frac{2}{3} & 0 \\ 0 & 0 & 0 & \frac{1}{3} \end{bmatrix}$$

The effective stress is calculated as

$$\bar{\sigma} = \bar{\sigma}\left(\bar{\varepsilon}, \dot{\bar{\varepsilon}}\right) \qquad (5.57)$$

and the stress components are expressed as

$$\sigma = \frac{2}{3}\frac{\bar{\sigma}}{\dot{\bar{\varepsilon}}_0}\left(\dot{\varepsilon} - \dot{\varepsilon}_m\right) + 3K\dot{\varepsilon}_m \qquad (5.58)$$

5.3 CASE STUDIES

Based on the finite element formulation discussed above, two computer codes are developed; one uses three noded triangular element (FEMLD3) whereas other four noded rectangular element (FEMLD4). With the help of these codes, one can analyze two-dimensional plane strain and axisymmetric problems. These codes were parallelized by incorporating the Modified Matrix Inversion Method parallel solver (MMIM) discussed in section 3.7. A typical problem in each category is analyzed using both the codes on supercomputer PARAM 10000. The following text describes the obtained results of the both case studies.

5.3.1 Axisymmetric Problem

A problem of simple compression of solid cylinder having dimensions, 1 inch radius and 1 inch height was considered (see Fig. 5.4). The cylinder was compressed with a velocity of 1 inch/s till 30% reduction in height was achieved. The reduction was occurred in 15 steps. The bottom surface was considered as frictionless and for top surface, friction factor of magnitude 0.5 was considered. The error norm was considered as 0.001 and limiting strain-rate value was considered as 0.01 to define the rigid portion of cylinder. The material behavior was expressed by the equation $\bar{\sigma} = k\dot{\bar{\varepsilon}}^m$, where the values of k and m were taken as 10 Ksi and 0.1 respectively. The cylinder was discretized using 800 three noded triangular elements with 441 nodes and 400 four noded rectangular elements with 441 nodes, resulting in global stiffness matrix of size 882 × 882 in both the cases. The problem was analyzed by increasing the number of processors from one to eight using codes FEMLD3 and FEMLD4. Each processor required approximately 14 MB of memory for every execution for both the codes on supercomputer PARAM 10000.

Deflected profiles of the specimen at different stages of deformation process were recorded. Variation of contours of nodal velocity, different components of stress tensor and strain tensor were also recorded. Load required for the compression during the compression process was also studied. To compare the results obtained using developed codes, the same problem was analyzed using the computer code (SPID) given in book by Kobayashi et al. [63] where the problem was discretized using 16 four nodded rectangular element with 25 nodes. The rest of the problem details were same.

5.3.1.1 Iterations

The solution procedure is iterative therefore, 59 iterations were carried out by FEMLD3 and 84 iterations were carried out by FEMLD4, to analyze the problem in 15 steps. Figure 5.5 shows the graph between number of iterations and the step numbers for FEMLD3 and FEMLD4.

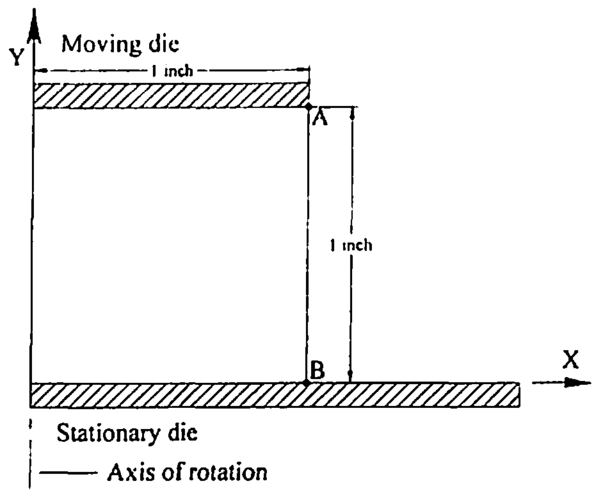
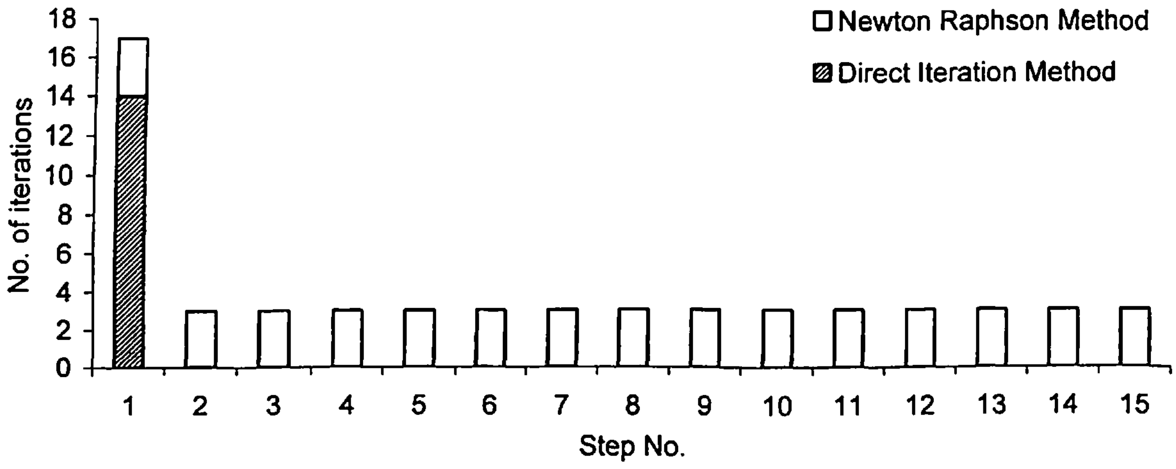
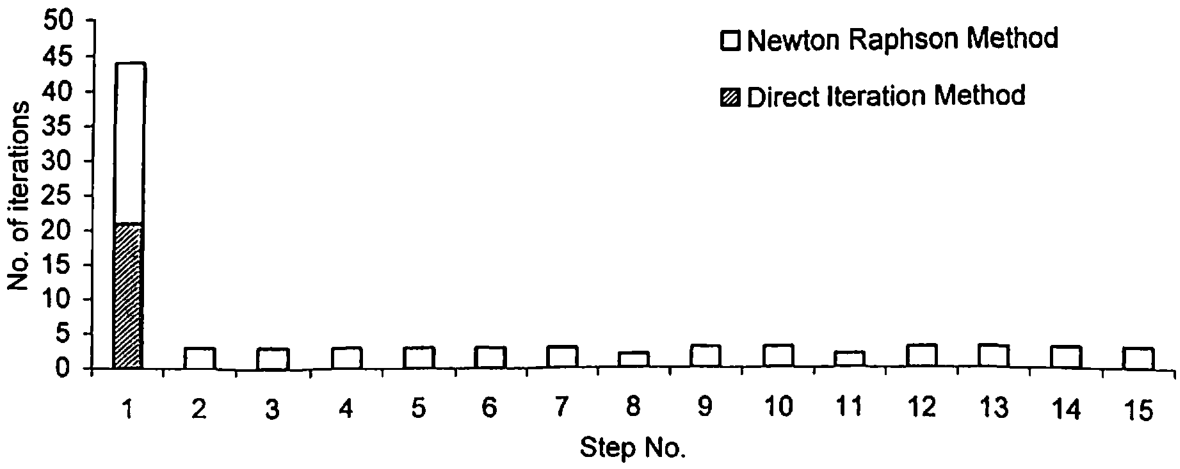


Fig. 5.4 Axisymmetric compression of solid cylinder

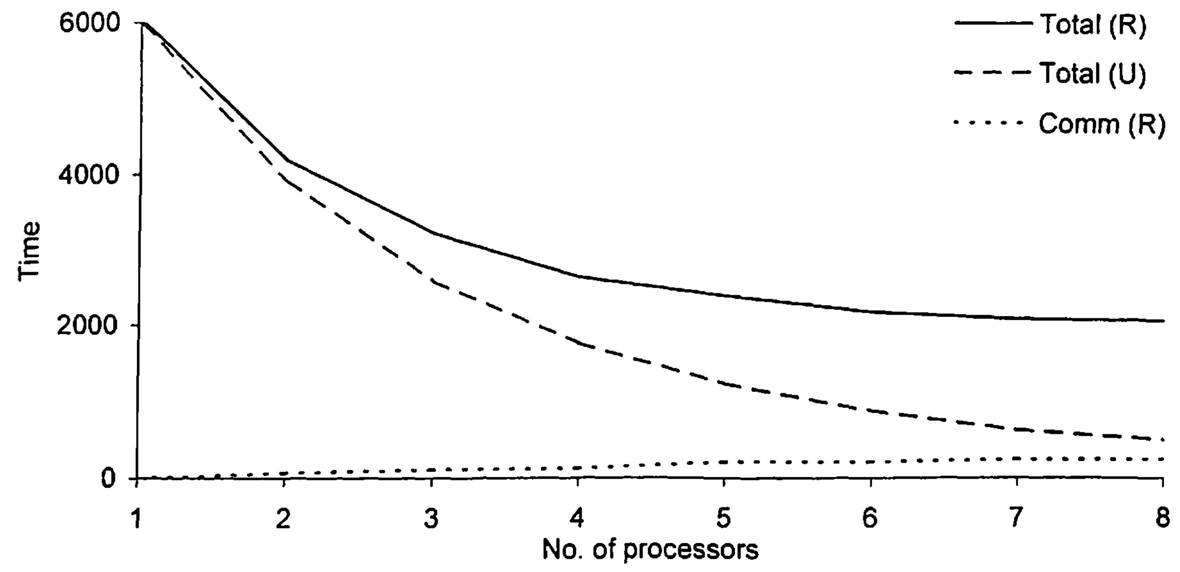


(a) Variation in number of iterations for code FEMLD3

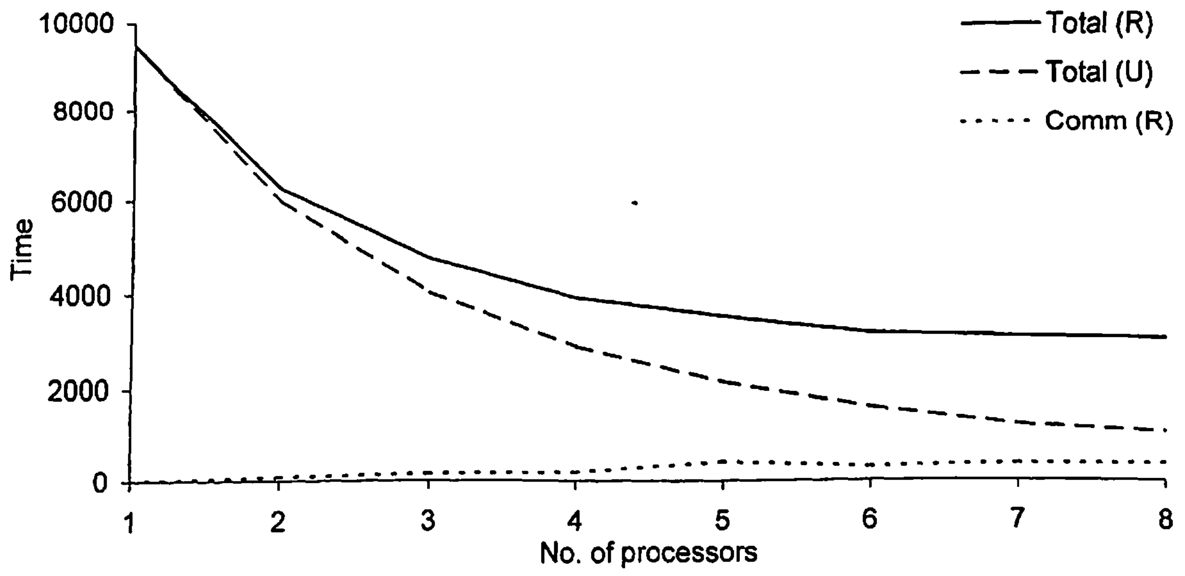


(b) Variation in number of iterations for code FEMLD4

Fig. 5.5 Variation in number of iterations with number of steps obtained from codes FEMLD3 and FEMLD4

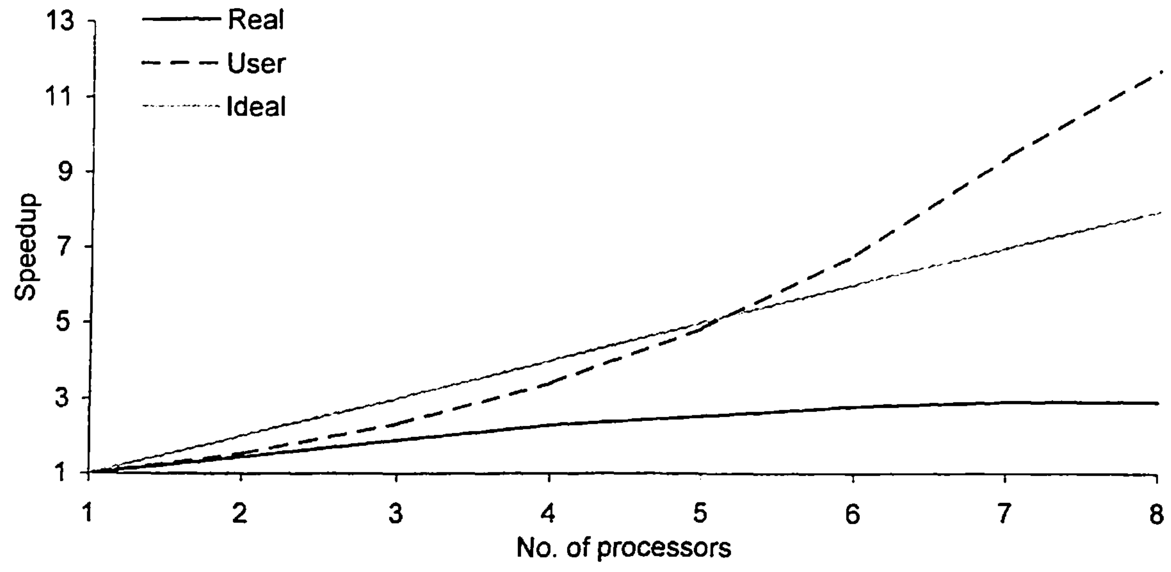


(a) Variation in components of computational time for code FEMLD3

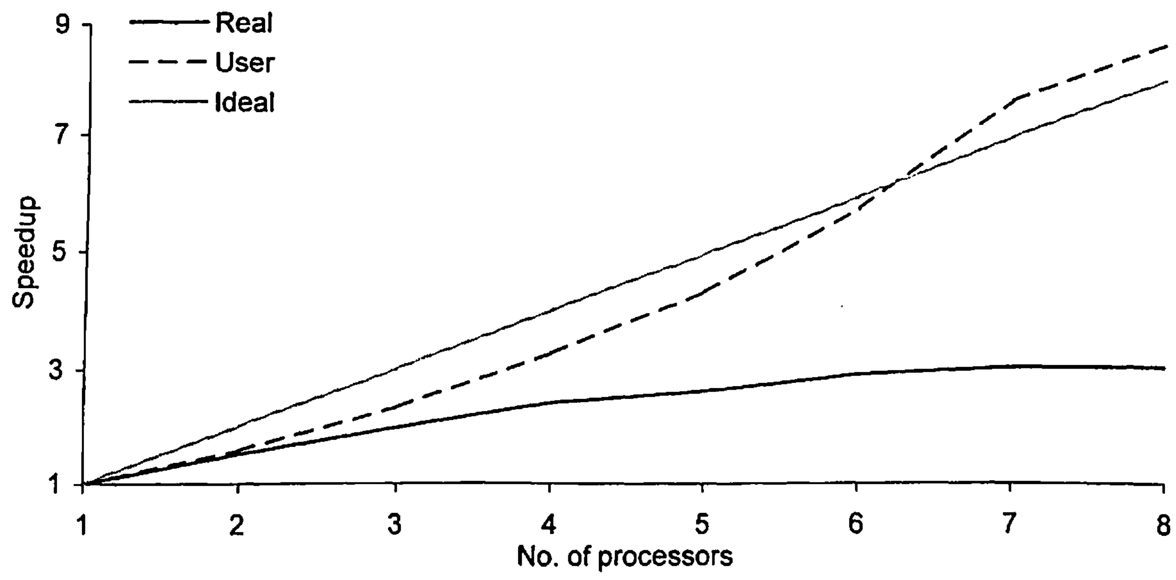


(b) Variation in components of computational time for code FEMLD4

Fig. 5.6 Variation in components of computational time obtained from codes FEMDL3 and FEMLD4



(a) Variation in Speedup for code FEMLD3



(b) Variation in Speedup for code FEMLD4

Fig. 5.7 Variation in Speedup obtained from codes FEMLD3 and FEMLD4

It can be seen from Fig. 5.5 that, the number of iterations required to achieve the solution velocities are more in the first step. The initial guess velocities are automatically generated in the first step. These velocities are not nearer to the solution velocities. Therefore, more number of iterations is required in the first step.

5.3.1.2 Performance On PARAM 10000

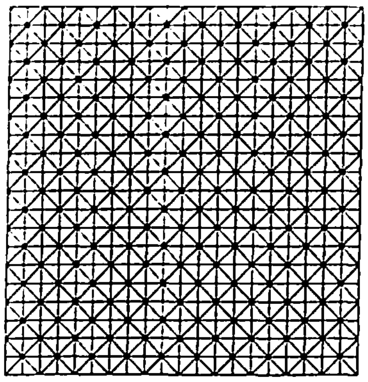
Figure 5.6 shows the variation in components of computational time with the number of processors achieved by both the codes. It can be seen from these graphs that the Total time measured in terms of Real time as well as User time reduces by considerable amount by increasing the number of processors, whereas the Communication time measured in terms of Real time increases with increase in number of processors.

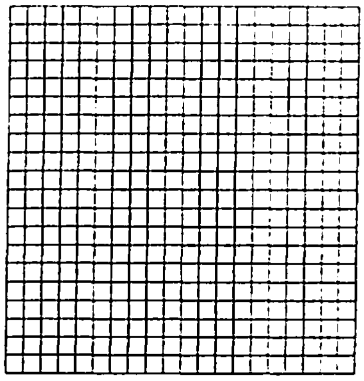
Figure 5.7 (a) and (b) shows the Speedup achieved using FEMLD3 and FEMLD4 codes respectively. The maximum Real time Speedup and User time Speedup achieved by FEMLD3 is 2.92 and 11.76 respectively, whereas the maximum Real time Speedup and User time Speedup achieved by FEMLD4 is 3.04 and 8.64 respectively. The User time Speedup is higher than the Ideal Speedup for both the codes at seven and eight number of processors. From this figure it can be observed that the performance of FEMLD3 is better than FEMLD4.

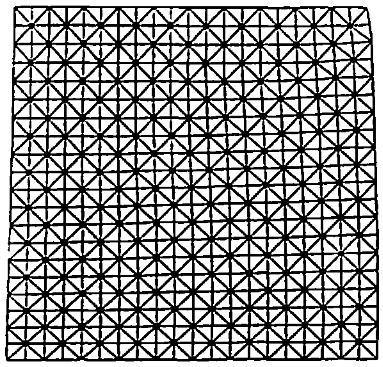
5.3.1.3 Results

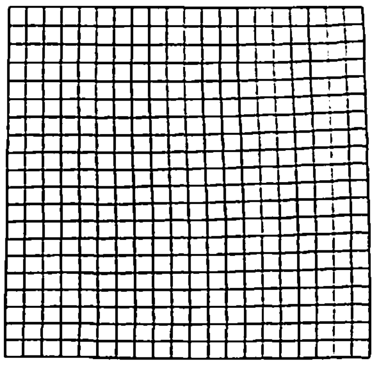
Figure 5.8 and 5.9 shows deformation at selected number of steps. It can be seen from these figures that, the deformed shape obtained by both the codes at various stage of deformation are nearly same.

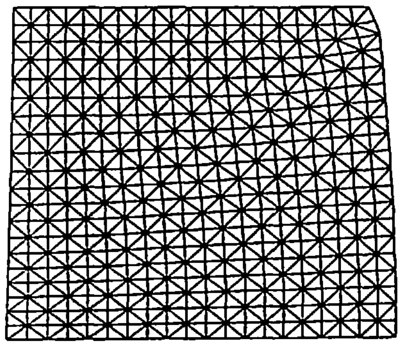
Figure 5.10 shows the undeformed-deformed shapes obtained by SPID. It can be seen from this figure that, the deformed shapes obtained by FEMLD3 and FEMLD4 at step No. 15 are identical to the deformed shape obtained by SPID.

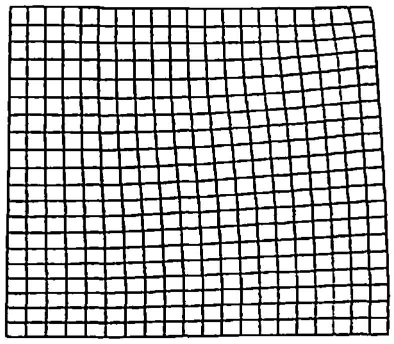

(a) Undeformed

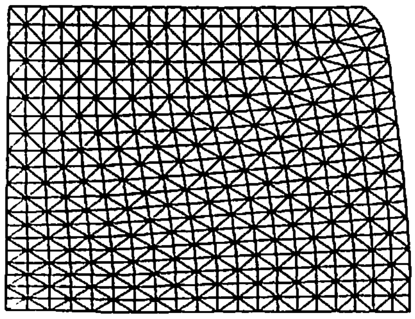

(a) Undeformed

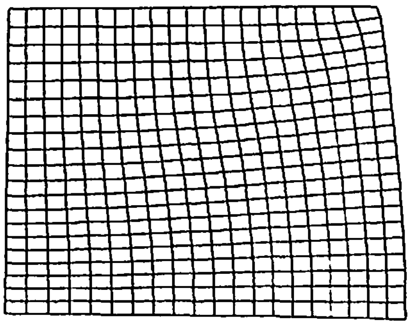

(b) Step No. 3 (6% compression)

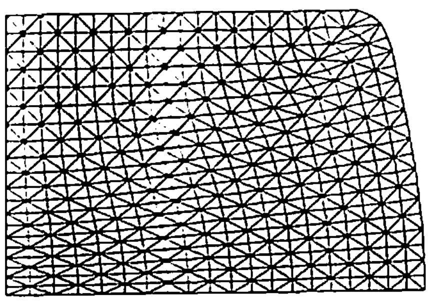

(b) Step No. 3 (6% compression)


(c) Step No. 6 (12% compression)

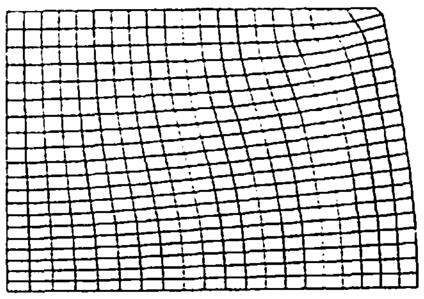

(c) Step No. 6 (12% compression)


(d) Step No. 9 (18% compression)

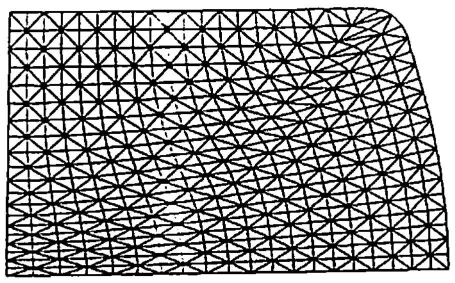

(d) Step No. 9 (18% compression)



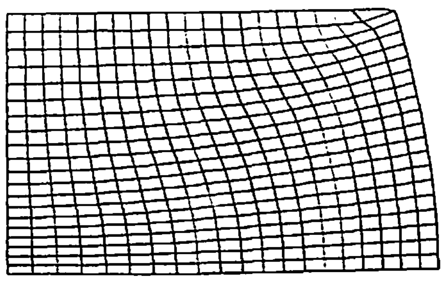
(e) Step No. 12 (24% compression)



(e) Step No. 12 (24% compression)



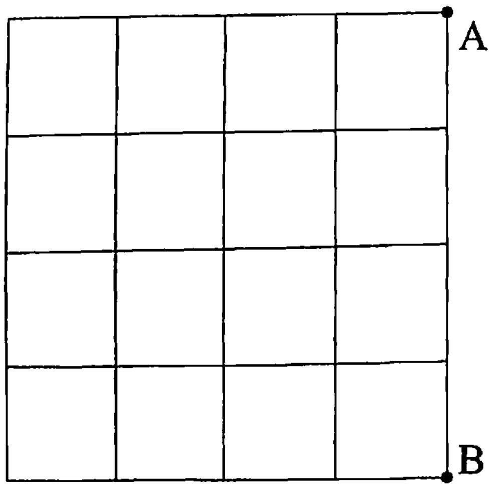
(f) Step No. 15 (30% compression)



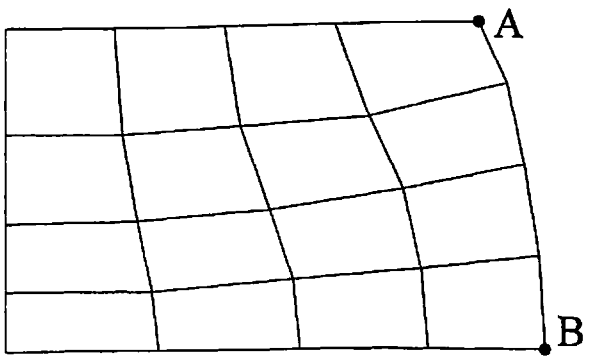
(f) Step No. 15 (30% compression)

Fig. 5.8 Deformed shaped obtained by FEMLD3

Fig. 5.9 Deformed shapes obtained by FEMLD4



(a) Undeformed shape



(b) Deformed shape at step No. 15 (30% compression)

Fig. 5.10 Undeformed and deformed shapes obtained by SPID

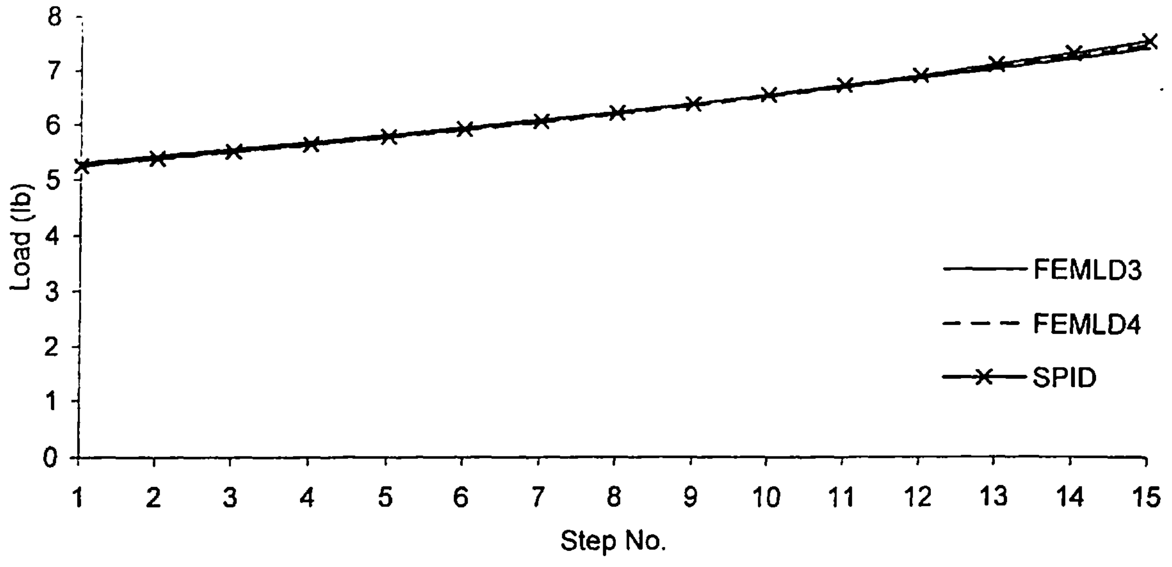
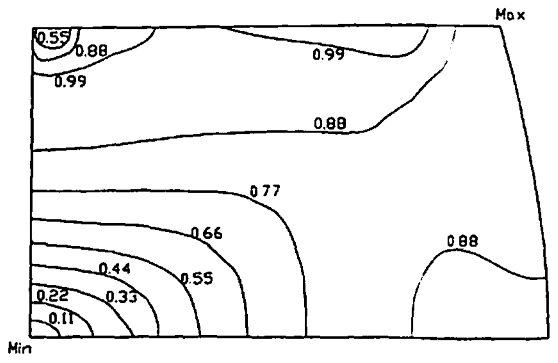
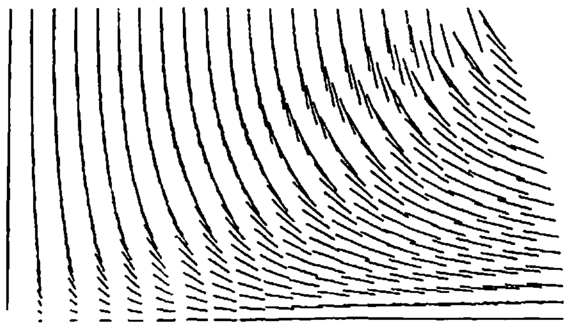


Fig. 5.11 Variation in forming load with number of steps



(a) Contour of nodal velocity



(b) Nodal velocity directions

Fig. 5.12 Contours of nodal velocity (inch/s) and their direction at step No.15 (30% compression) obtained by FEMLD4

When the forces obtained by both the codes at various nodes are compared, it was found that the magnitude of force and their directions are almost same at every step and has good agreement with the forces obtained by SPID. Figure 5.11 shows the variation in forming load, which is calculated by adding forces in y-direction at nodes of which the nodal velocities are specified.

Figure 5.12 (a) shows the contours of velocity at step 15. It also shows the location of maximum velocity as well as minimum velocity points. Figure 5.12 (b) shows the velocity directions at step 15. This denotes the direction of metal flow during the deformation process.

Figure 5.13 shows the contours of strain-rate components $\dot{\varepsilon}_r$, $\dot{\varepsilon}_z$, $\dot{\varepsilon}_\theta$ and $\dot{\gamma}_{rz}$ at step 15. After comparing the strain-rate results at different location, large variation in their magnitude was found near the top right corner of the sample (point A as per Fig. 5.4). The strain-rate components obtained by both the codes are found nearly same at any location under consideration. It was found that the volumetric strain of the sample is near to zero in both the cases. But in case of triangular elements, the volumetric strain-rate is greater than the volumetric strain-rate of rectangular elements.

Figure 5.14 shows the contours of stress components σ_r, σ_z, σ_θ, and τ_{rz} at step 15. From these contours, high stress concentration can be observed at the top right corner of the specimen (point A as per Fig. 5.4). When the stress results obtained by both the codes are compared, it was found that the results of FEMLD3 do not match with the results obtained by FEMLD4 and SPID. The main reason for this is the increased volumetric strain-rate in triangular element as compared to the rectangular element. This increase in volumetric strain-rate results in improper stress computation. Therefore the results of stress components, namely σ_r, σ_z, σ_θ and τ_{rz}, computed by FEMLD3 do not aggress with the results of stress components obtained by FEMLD4 and SPID.

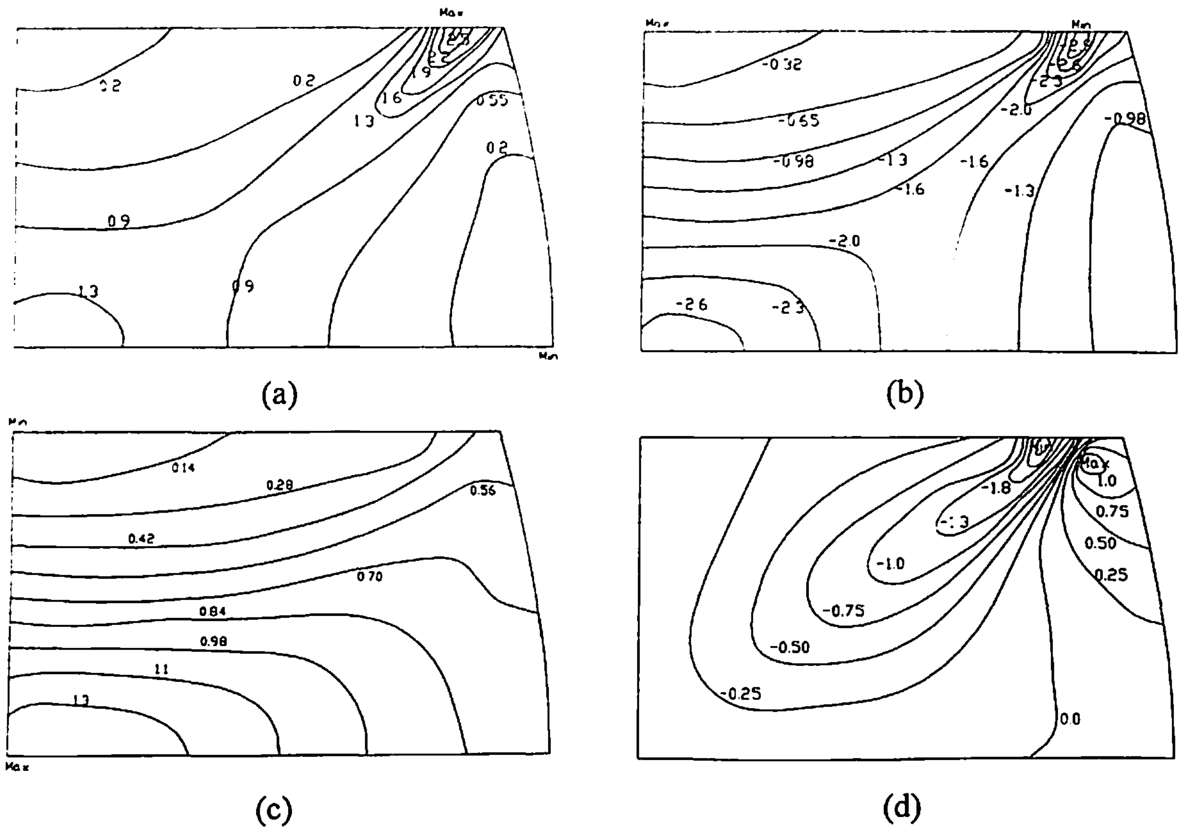


Fig. 5.13 Contours of (a) $\dot{\varepsilon}_r$ (b) $\dot{\varepsilon}_z$ (c) $\dot{\varepsilon}_\theta$ and (d) $\dot{\gamma}_{rz}$ at step No. 15 (30% compression) obtained by FEMLD4

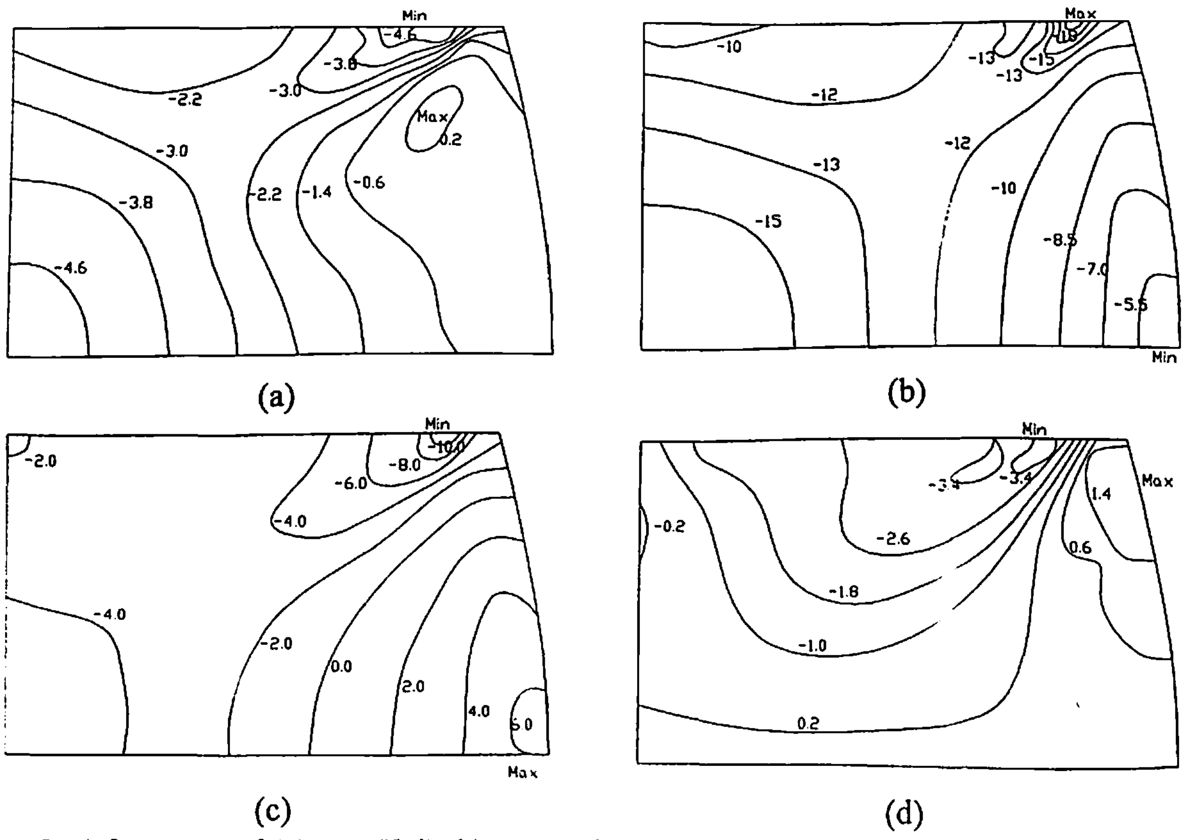


Fig. 5.14 Contours of (a) σ_r (Ksi) (b) σ_z (Ksi) (c) σ_θ (Ksi) and (d) τ_{rz} (Ksi) at step No. 15 (30% compression) obtained by FEMLD4

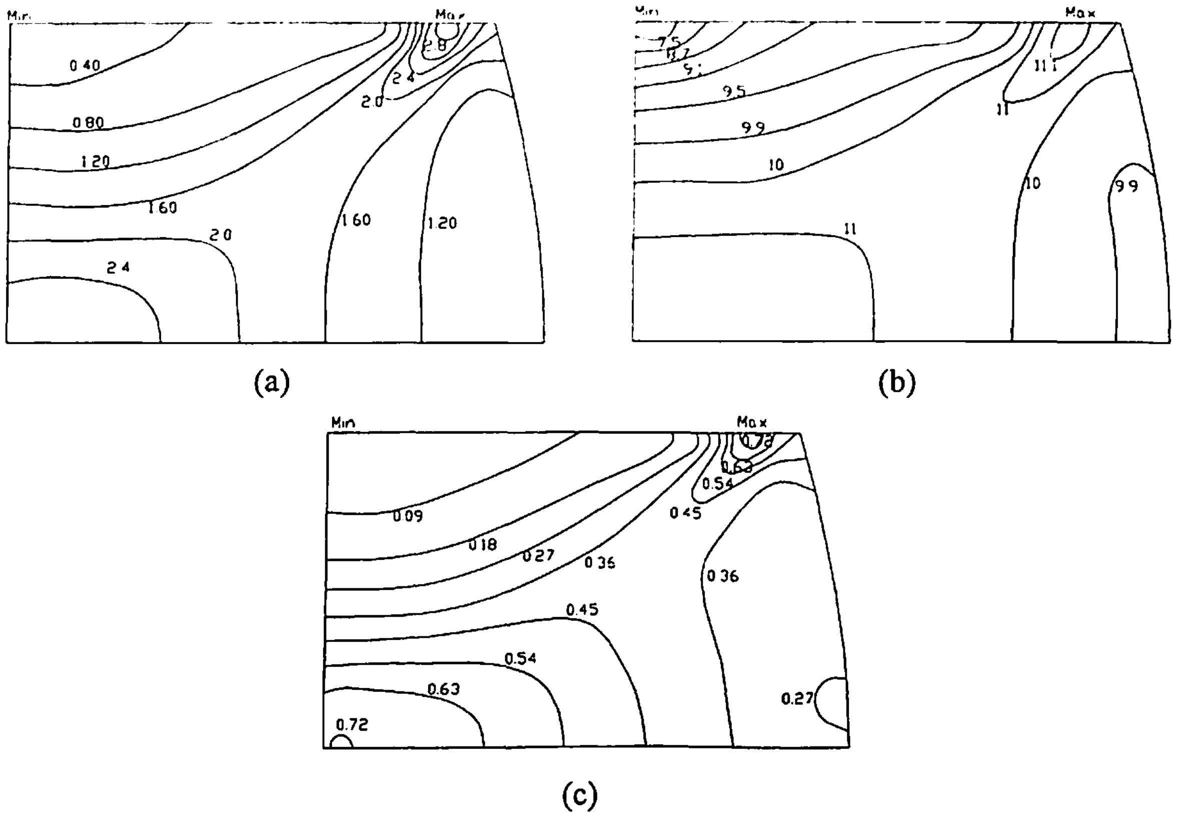


Fig. 5.15 Contours of (a) $\dot{\bar{\varepsilon}}$ (b) $\bar{\sigma}$ (Ksi) and (c) $\bar{\varepsilon}$ at step No. 15 (30% compression) obtained by FEMLD4

Table 5.1 Comparison of X-coordinates (mm) obtained by present investigation (FEMLD4) and FORGE2 at different stages of compression

Compression	Point	FORGE2	Authors	SPID
10 %	A	25.84	25.94	25.94
	B	26.78	27.20	27.23
20 %	A	26.73	26.67	26.71
	B	28.80	29.16	29.22
30 %	A	28.00	27.56	27.73
	B	30.90	31.33	31.45

Figure 5.15 shows the contours of effective strain-rate ($\dot{\bar{\varepsilon}}$), effective stress ($\bar{\sigma}$) and total effective strain ($\bar{\varepsilon}$) at step 15. It was found that the above-mentioned results obtained by FEMLD3 and FEMLD4 has excellent agreement with the results obtained by SPID.

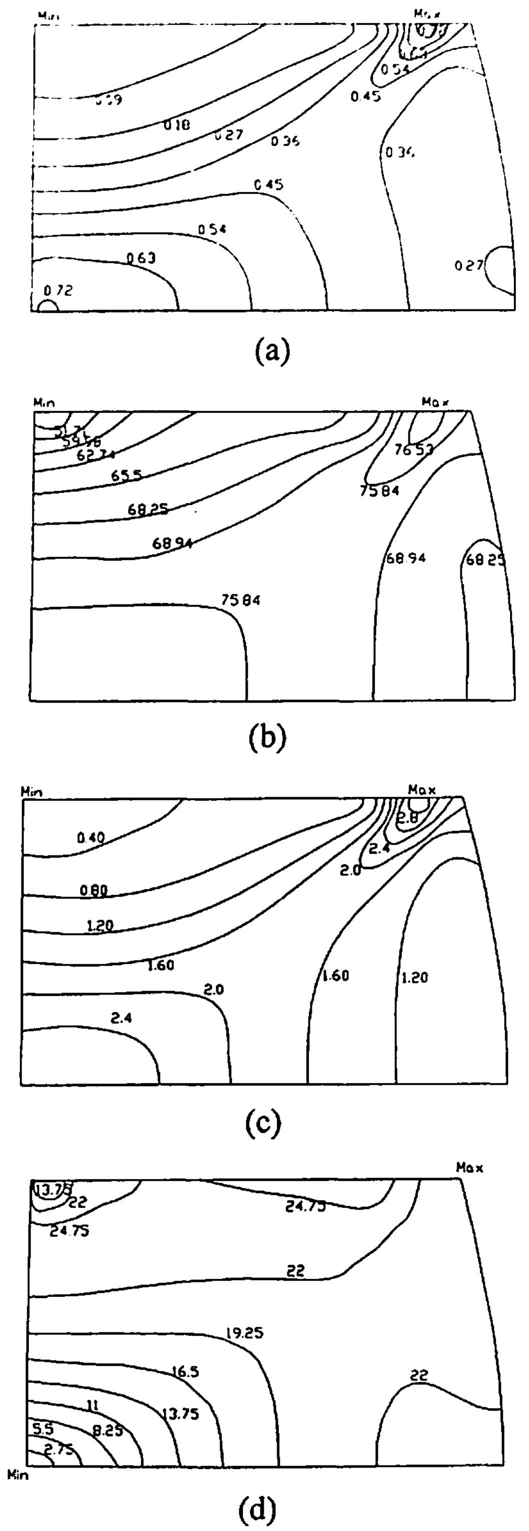


Fig. 5.16 Contours showing (a) effective strain (b) effective stress (MPa) (c) effective strain rate (d) nodal velocity (mm/s) distribution obtained by present investigation

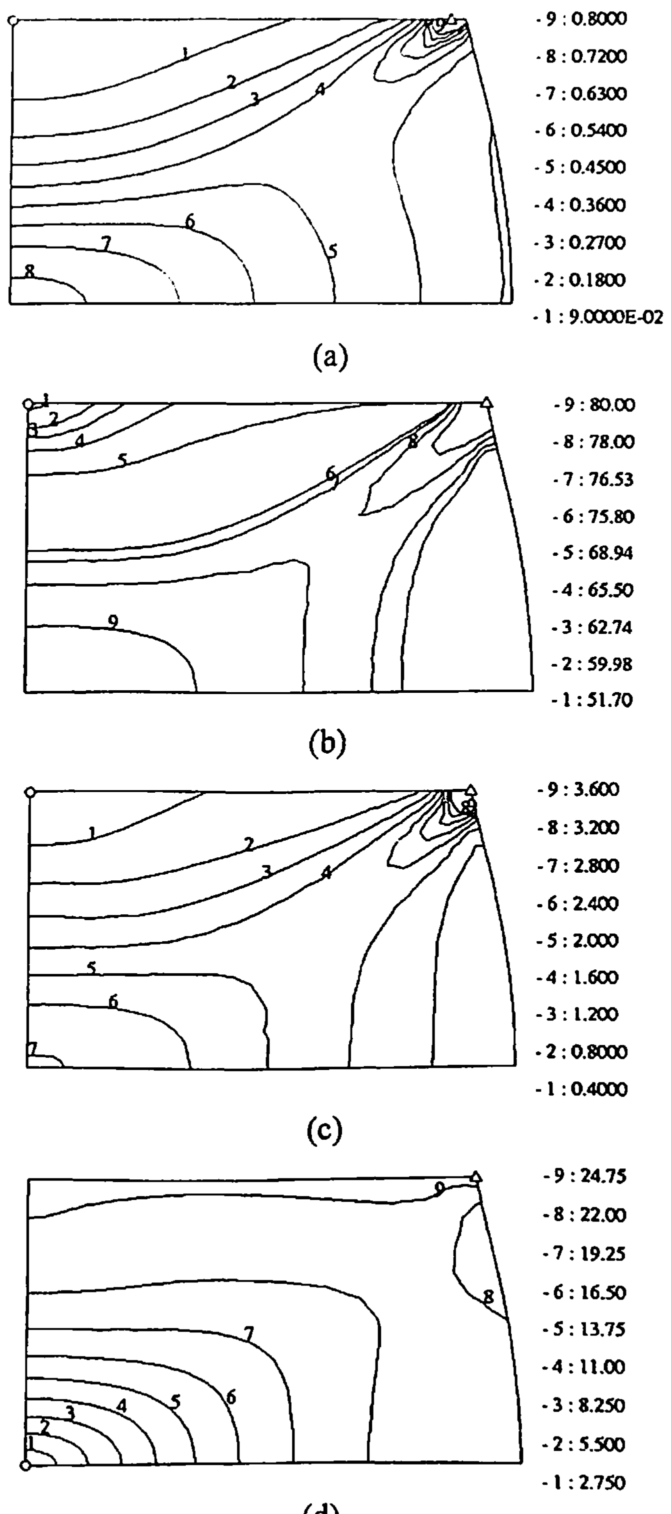


Fig. 5.17 Contours showing (a) effective strain (b) effective stress (MPa) (c) effective strain rate (d) nodal velocity (mm/s) distribution obtained by FORGE2 (Δ and O represent maximum and minimum respectively)

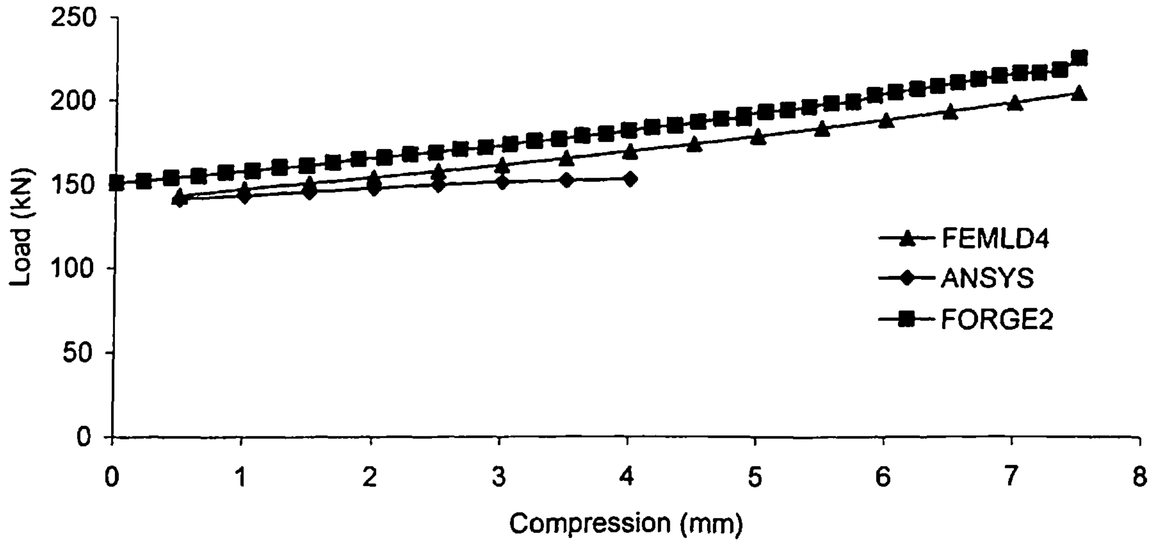


Fig. 5.18 Load compression relationship

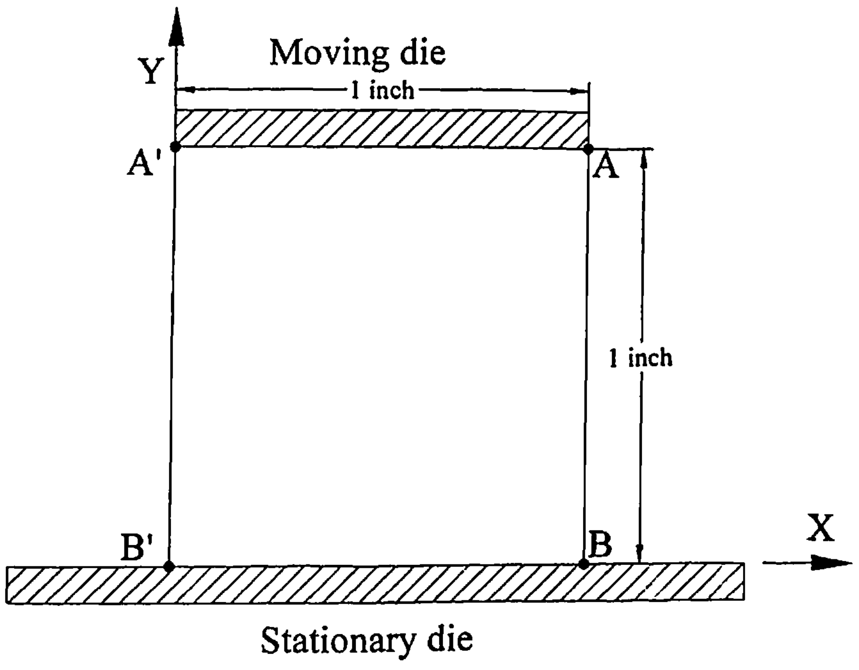


Fig. 5.19 Compression of solid square bar

5.3.1.4 Comparison with Commercial Softwares

The same problem was also analyzed using commercial software FORGE2 and ANSYS. To compare the results obtained by present investigation and FORGE2, two points A and B are considered (shown in Fig. 5.4). Table 5.1 shows the comparison of X-coordinates of points A and B at different stages of compression. It can be seen from this table that at every stage of deformation, X-coordinates of point A and B obtained by present investigation and FORGE2 are nearly same.

Figures 5.16 and 5.17 show the variation of total effective strain ($\bar{\varepsilon}$), effective stress ($\bar{\sigma}$), effective strain-rate ($\dot{\bar{\varepsilon}}$) and nodal velocity obtained by present investigation and FORGE2 respectively at 30% compression (step 15). After comparing both the figures one can conclude that the variations obtained using FORGE2 and present investigation match well. The location of minimum/maximum values of effective strain, effective stress, effective strain-rate and nodal velocity is also almost at the same points.

Figure 5.18 shows the load compression curve. It can be seen from this figure that the forming load obtained by developed software matches fairly well with the forming load obtained by commercial software FORGE2 and ANSYS.

5.3.2 Plane Strain Problem

A problem of simple compression of solid square bar of dimension 1 inch with unit thickness was taken for study (see Fig. 5.19). The bar was compressed with a velocity of 1 inch/s till 30% reduction in height was achieved. The reduction was occurred in 15 steps. The bottom surface was considered as frictionless while for top surface, friction factor of magnitude 0.5 was considered. The velocity error norm was taken equal to 0.001 and limiting strain-rate value was considered as 0.01 to define the rigid portion of square bar. The material behavior was expressed by the equation $\bar{\sigma} = k\dot{\bar{\varepsilon}}^m$, where the values of k and m were taken as 10 Ksi and 0.1 respectively. The specimen was discretized using 800 three noded triangular elements with 441nodes and 400 four noded rectangular elements with 441 nodes, resulting in global stiffness matrix of size 882 × 882 in both the cases. The problem was analyzed by increasing the number of processors from one to eight using FEMLD3 and FEMLD4. Each processor required approximately 14 MB of memory for every execution for both the codes.

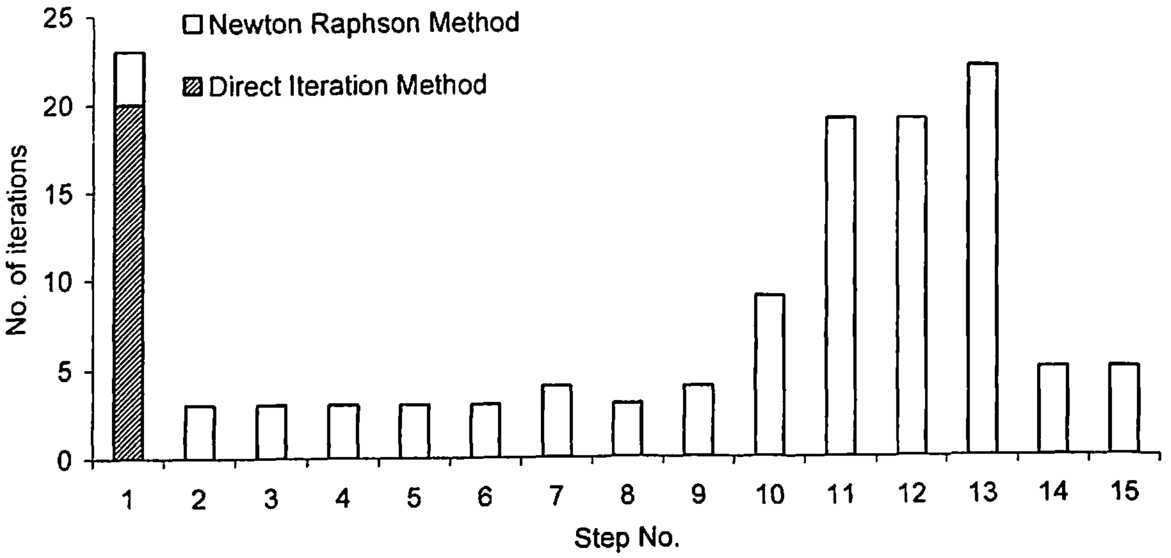
Deflected profiles of the specimen at different stages of deformation process were recorded. Variation of contours of nodal velocity, different components of stress tensor and strain tensor were also recorded. Load required for the compression during the compression process was also studied. To compare the results obtained using developed codes, the same problem was analyzed using the computer code (SPID) given in book by Kobayashi et al. [63] where the problem was discretized using 16 four nodded rectangular element with 25 nodes. The rest of the problem details were same.

5.3.2.1 Iterations

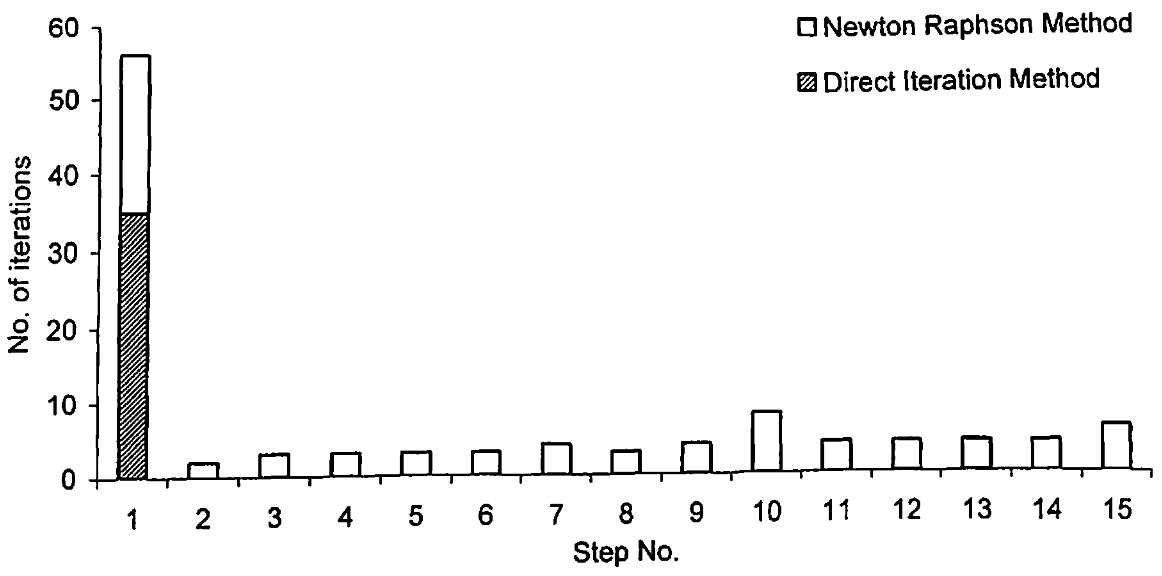
The solution procedure is iterative therefore, 128 iterations were carried out in FEMLD3 and 111 iterations were carried out in FEMLD4, to analyze the problem in 15 steps. Figure 5.20 shows the graph between number of iterations and the step numbers for FEMLD3 and FEMLD4.

5.3.2.2 Performance on PARAM 10000

The above analysis was carried out on supercomputer PARAM 10000 by changing the number of processors from one to eight. Figure 5.21 shows the variation in components of computational time with the number of processors achieved by both the codes. It can be seen from these graphs that the Total time measured in terms of Real time as well as User time reduces by considerable amount by increasing the number of processors, whereas the Communication time measured in terms of Real time increases with increase in number of processors. Figure 5.22 (a) and (b) shows the Speedup achieved by FEMLD3 and FEMLD4 respectively. The maximum Real time Speedup and User time Speedup achieved by FEMLD3 is 3.01 and 8.47 respectively, whereas the maximum Real time Speedup and User time Speedup achieved by FEMLD4 is 3.15 and 7.13 respectively. From this figure it can be observed that the performance of FEMLD3 is better than FEMLD4.

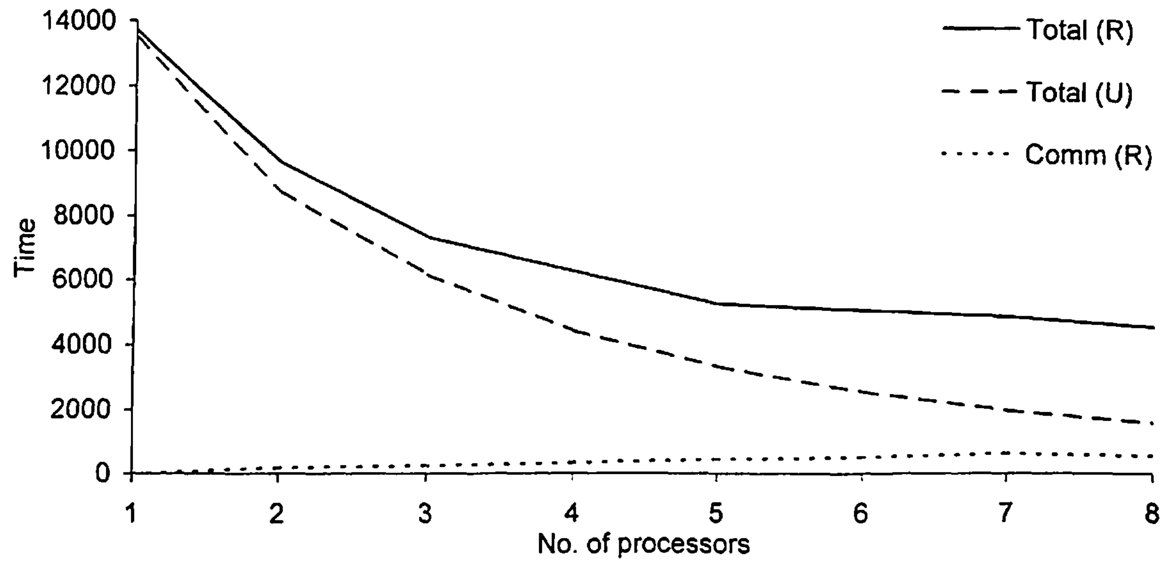


(a) Variation in number of iterations for code FEMLD3

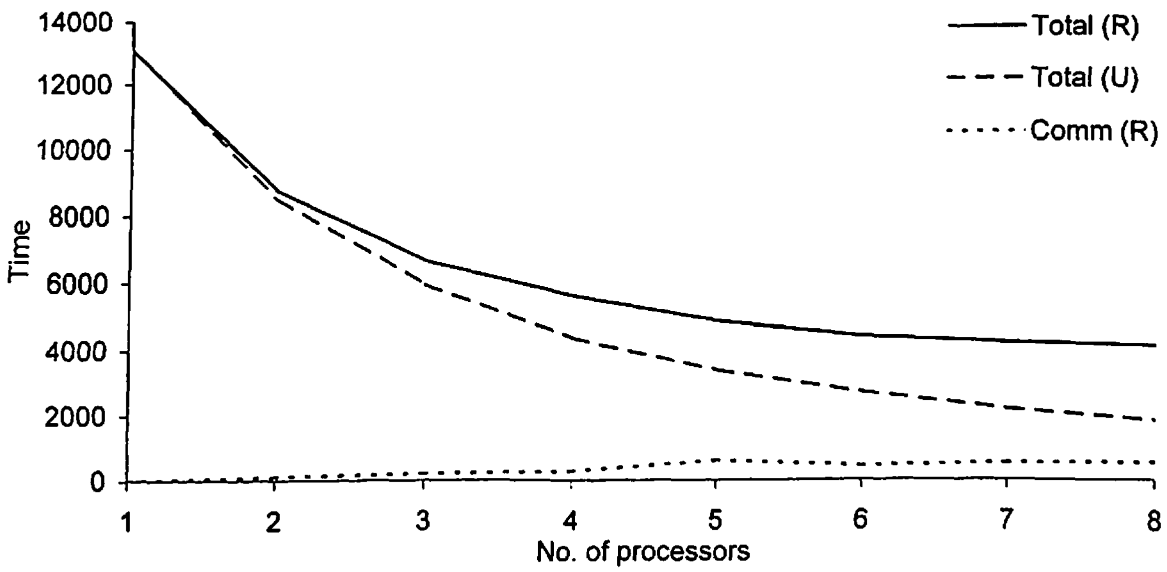


(b) Variation in number of iterations for code FEMLD4

Fig. 5.20 Variation in number of iterations with number of steps obtained from codes FEMLD3 and FEMLD4

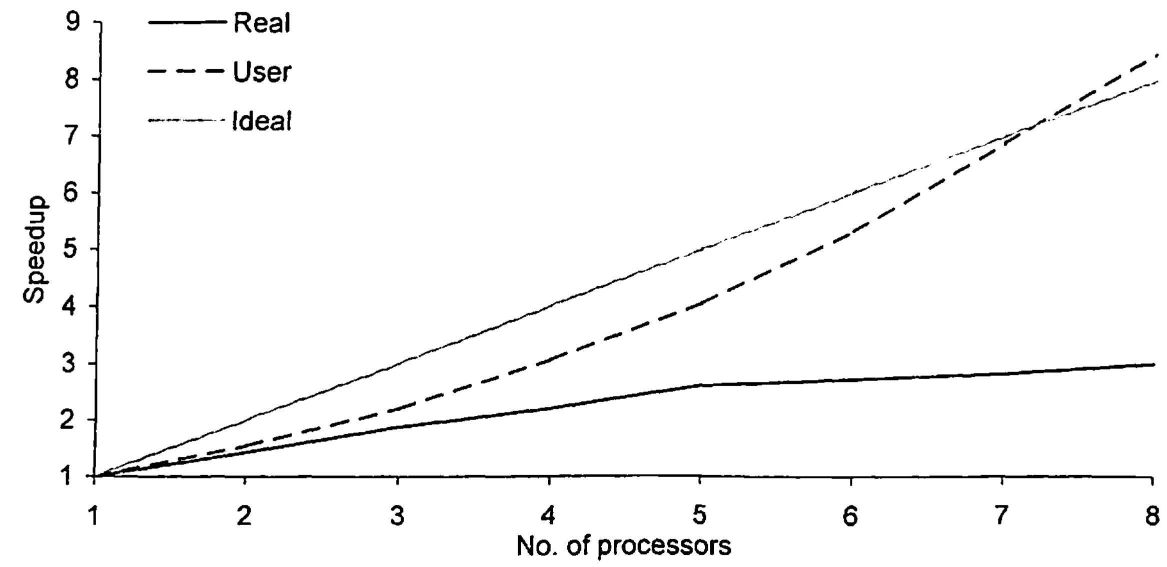


(a) Variation in components of computational time for code FEMLD3

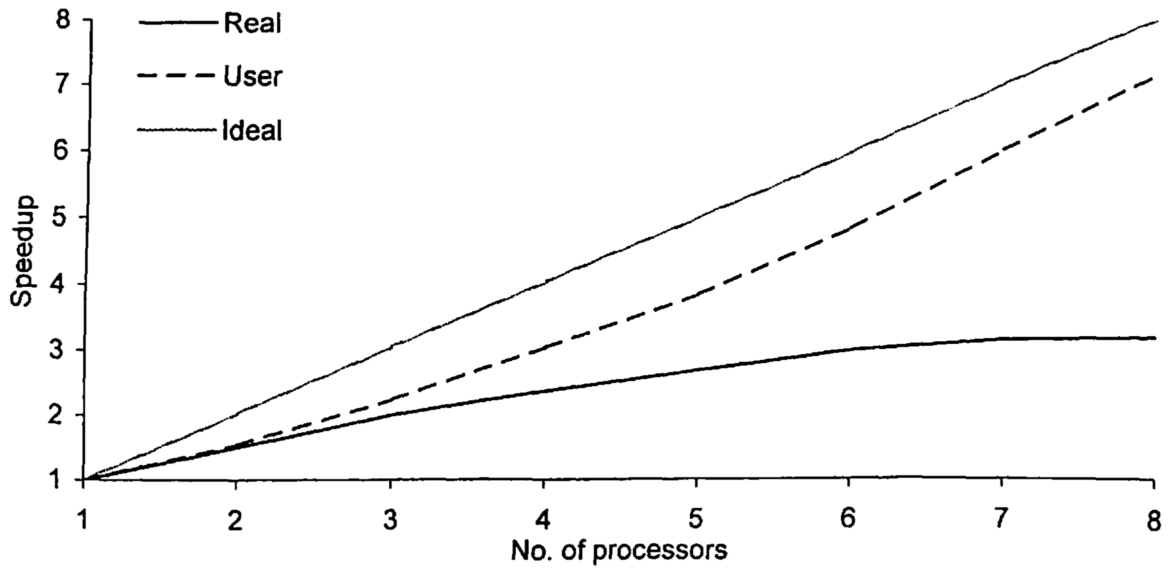


(b) Variation in components of computational time for code FEMLD4

Fig. 5.21 Variation in components of computational time obtained from codes FEMDL3 and FEMLD4

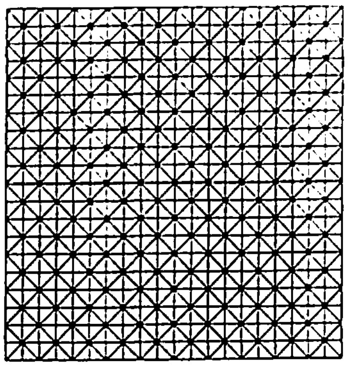


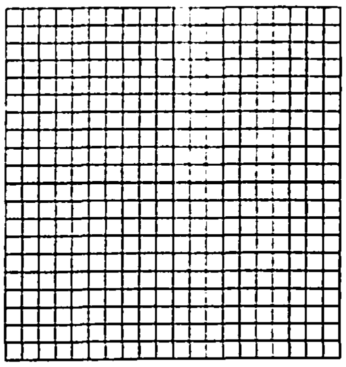
(a) Variation in Speedup for code FEMLD3

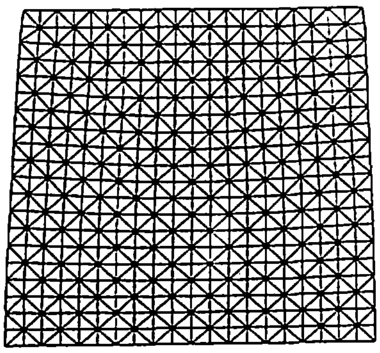


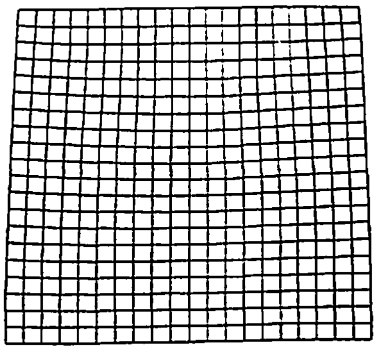
(b) Variation in Speedup for code FEMLD4

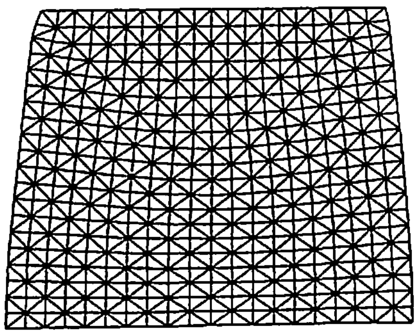
Fig. 5.22 Variation in Speedup obtained from codes FEMLD3 and FEMLD4

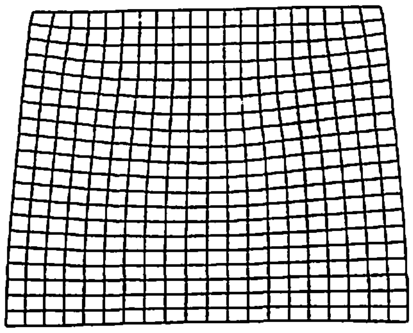

(a) Undeformed

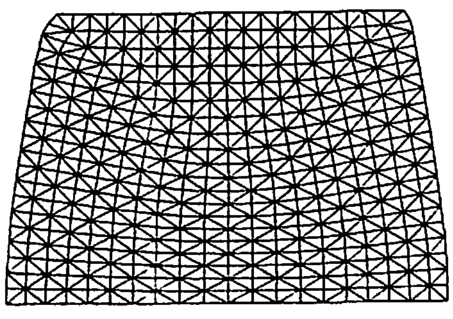

(a) Undeformed

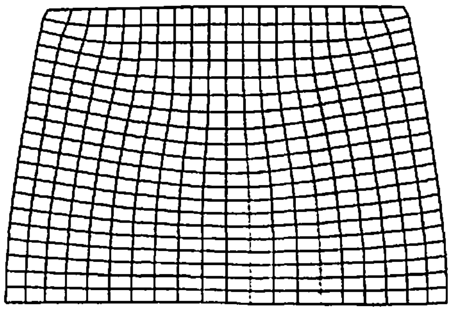

(b) Step No. 3 (6% compression)

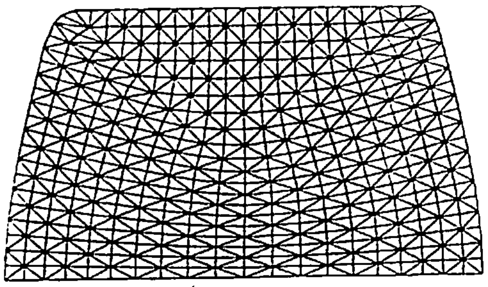

(b) Step No. 3 (6% compression)


(c) Step No. 6 (12% compression)

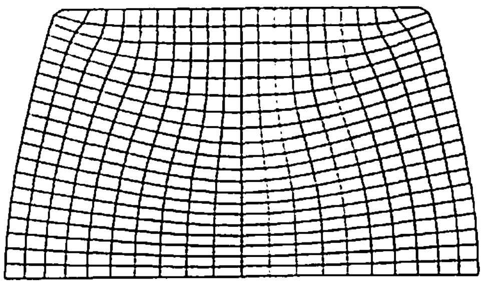

(c) Step No. 6 (12% compression)


(d) Step No. 9 (18% compression)

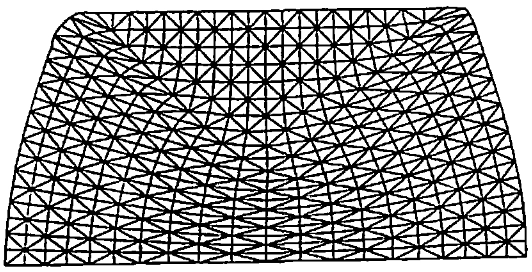

(d) Step No. 9 (18% compression)



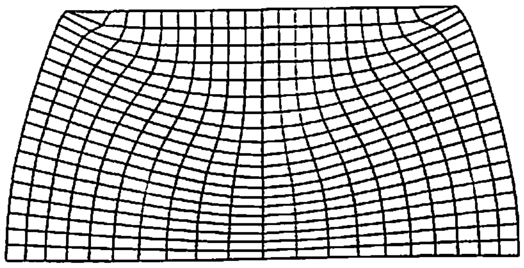
(e) Step No. 12 (24% compression)



(e) Step No. 12 (24% compression)



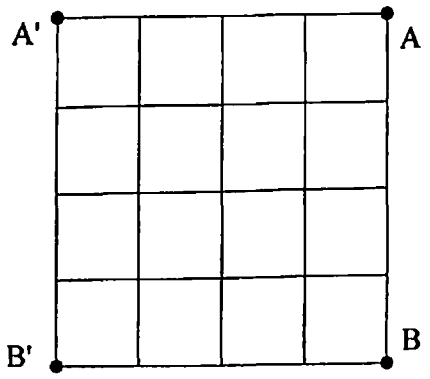
(f) Step No. 15 (30% compression)



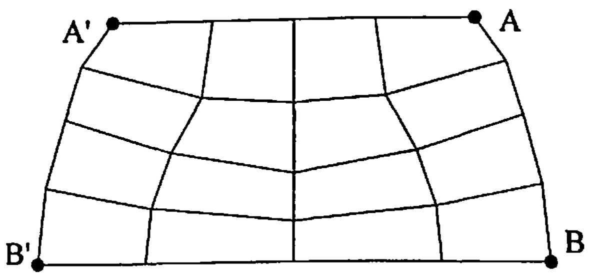
(f) Step No. 15 (30% compression)

Fig. 5.23 Deformed shaped obtained by FEMLD3

Fig. 5.24 Deformed shapes obtained by FEMLD4



(a) Undeformed shape



(b) Deformed shape at step No. 15 (30% compression)

Fig. 5.25 Undeformed and deformed shapes obtained by SPID

5.3.2.3 Results

Figure 5.23 and 5.24 shows deformation at selected number of steps. It can be seen from these figures that, the deformed shape obtained by both the codes at various stage of deformation are nearly same.

Figure 5.25 shows the undeformed-deformed shapes obtained by SPID. It can be seen from this figure that, the deformed shapes obtained by FEMLD3 and FEMLD4 at step No. 15 is identical to the deformed shape obtained by SPID.

When the compressive force obtained using both the codes at various nodes are compared, it was found that the magnitude of compressive force and their directions are almost same at every step and has excellent agreement with the forces obtained by SPID. Figure 5.26 shows the variation in forming load computed by adding forces in y-direction at nodes on which the nodal velocities were specified.

Figure 5.27 (a) shows the contours of velocity at step 15. It also shows the location of maximum as well as minimum velocity points. Figure 5.27 (b) shows the velocity directions at step 15. This roughly depicts the direction of metal flow during the deformation process.

Figure 5.28 shows the contours of strain-rate components $\dot{\varepsilon}_x$, $\dot{\varepsilon}_y$, and $\dot{\gamma}_{xy}$ at step 15. After comparing the strain-rate results at different location, large variation in their magnitude was found near the top corners of the sample (point A and A' as per Fig. 5.19). The strain-rate component in z-direction ($\dot{\varepsilon}_z$) is zero for plane strain problems. The strain-rate components obtained by both the codes are found nearly same at any location under consideration. It was found that the volumetric strain of the sample is very much near to zero in both the cases. But in case of triangular elements, the volumetric strain-rate is greater than the volumetric strain-rate of rectangular elements.

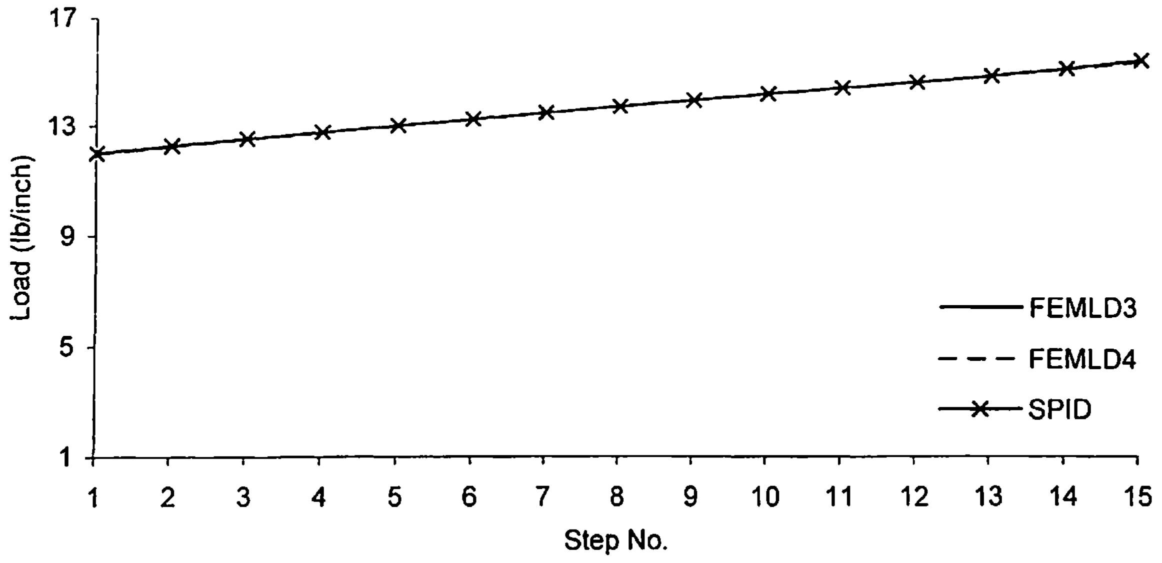
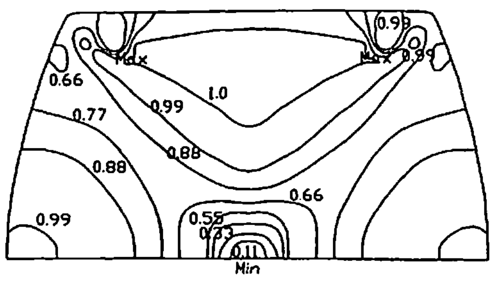
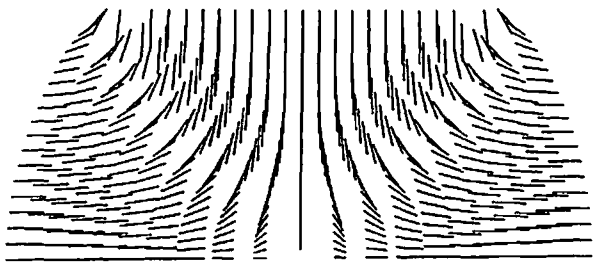


Fig. 5.26 Variation in forming load with number of steps



(a) Contour of nodal velocity



(b) Nodal velocity directions

Fig. 5.27 Contours of nodal velocity (inch/s) and their direction at step No.15 (30% compression) obtained by FEMLD4

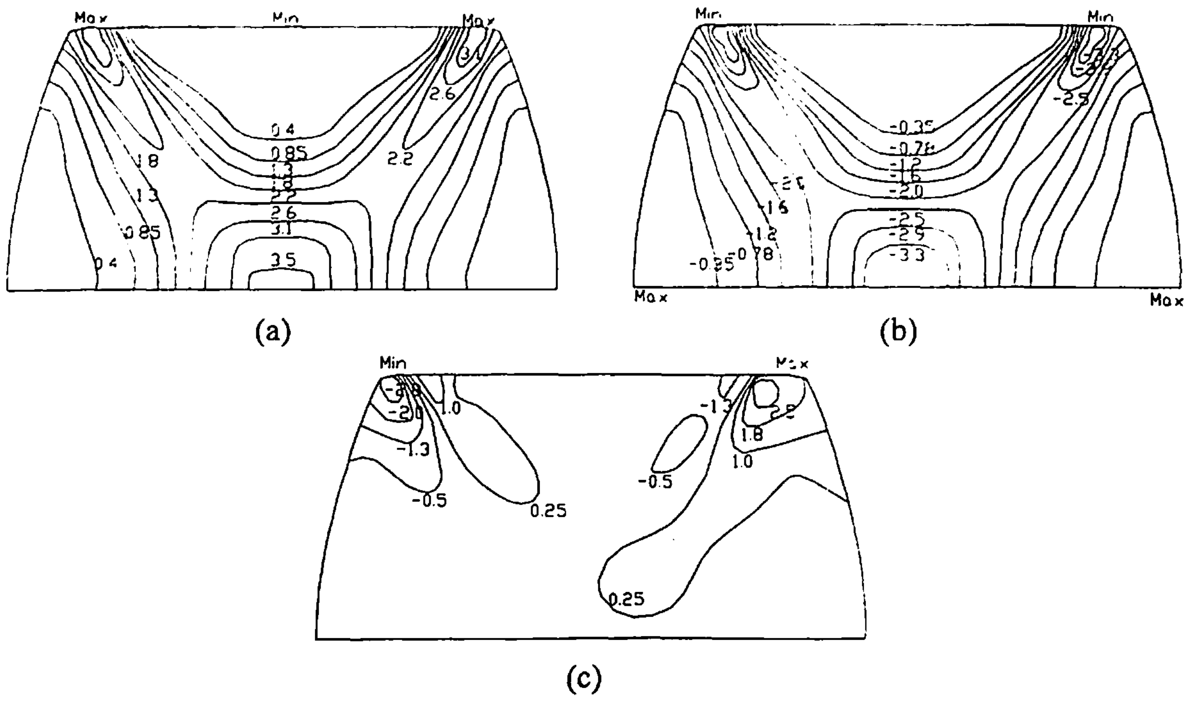


Fig. 5.28 Contours of (a) $\dot{\varepsilon}_x$ (b) $\dot{\varepsilon}_y$ and (c) $\dot{\gamma}_{xy}$ at step No. 15 (30% compression) obtained by FEMLD4

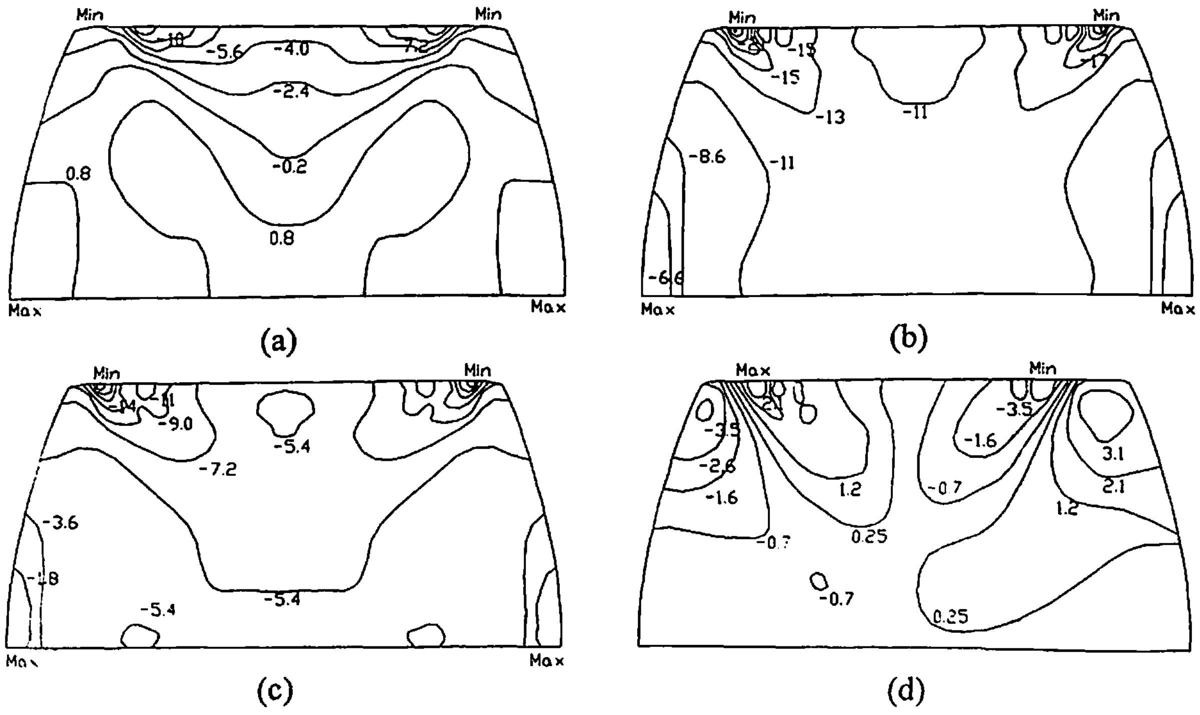


Fig. 5.29 Contours of (a) σ_x (Ksi) (b) σ_y (Ksi) (c) σ_z (Ksi) and (d) τ_{xy} (Ksi) at step No. 15 (30% compression) obtained by FEMLD4

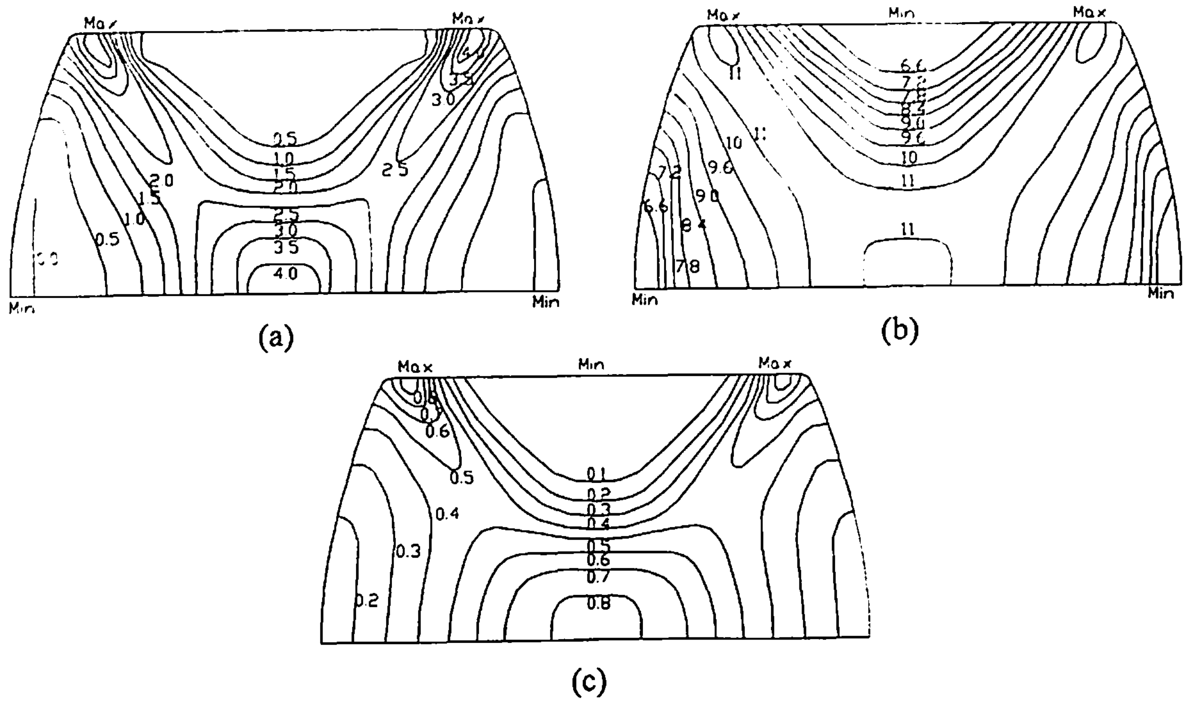


Fig. 5.30 Contours of (a) $\dot{\bar{\varepsilon}}$ (b) $\bar{\sigma}$ (Ksi) and (c) $\bar{\varepsilon}$ at step No. 15 (30% compression) obtained by FEMLD4

Table 5.2 Comparison of X-coordinates (mm) obtained by present investigation (FEMLD4) and FORGE2 at different stages of compression

Compression	Point	FORGE2	Authors	SIPD
10 %	A	25.36	25.75	25.78
	B	26.98	27.40	27.44
20 %	A	26.24	26.16	26.25
	B	29.38	29.75	29.88
30 %	A	27.27	26.56	26.82
	B	31.82	32.23	32.65

Figure 5.29 shows the contours of stress components σ_x, σ_y, σ_z and τ_{xy} at step 15. From these contours, high stress concentration can be observed at the top corners of the specimen (point A and A' as per Fig. 5.19). When the stress results obtained by both the codes were compared, it was found that the result of FEMLD3 does not matches with the result obtained by FEMLD4 and SPID. The main reason for this is the increased volumetric strain-rate in triangular element as compared to the rectangular element. This increase in volumetric strain-rate results in improper stress computation. Therefore the results of stress components, namely σ_x, σ_y, σ_z and τ_{xy}, computed by FEMLD3 do not match with the results of stress components obtained by FEMLD4 and SPID.

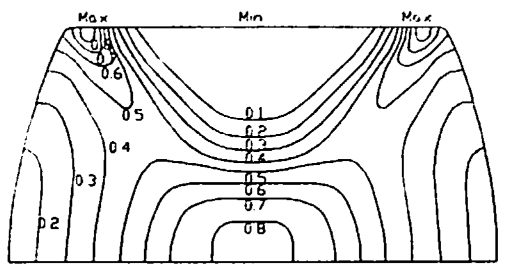
Figure 5.30 shows the contours of effective strain-rate ($\dot{\bar{\varepsilon}}$), effective stress ($\bar{\sigma}$) and total effective strain ($\bar{\varepsilon}$) at step 15 (30% compression). It was also found that the above-mentioned results obtained by FEMLD3 and FEMLD4 have excellent agreement with the results obtained using SPID.

5.3.2.4 Comparison with Commercial Softwares

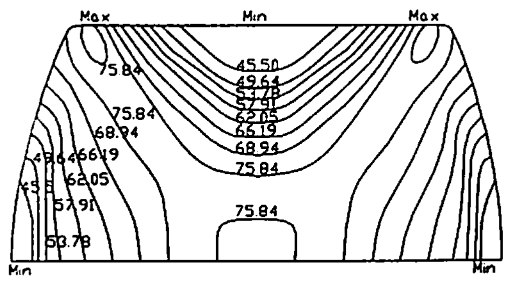
The problem was also analyzed by FORGE2 and Table 5.2 shows the comparison of X-coordinate of points A and B (as per Fig. 5.19) at different stages of compression. It can be seen from this table that at every stage of compression, X-coordinates of points A and B obtained by present investigation and FORGE2 are nearly same.

Figures 5.31 and 5.32 show the variation of total effective strain ($\bar{\varepsilon}$), effective stress ($\bar{\sigma}$), effective strain-rate ($\dot{\bar{\varepsilon}}$) and nodal velocity obtained by present investigation and FORGE2 respectively at 30% compression (step 15). After comparing both the figures one can conclude that the variations obtained using FORGE2 and present investigation match well. The location of minimum/maximum values of effective strain, effective stress, effective strain-rate and nodal velocity is also almost at the same points.

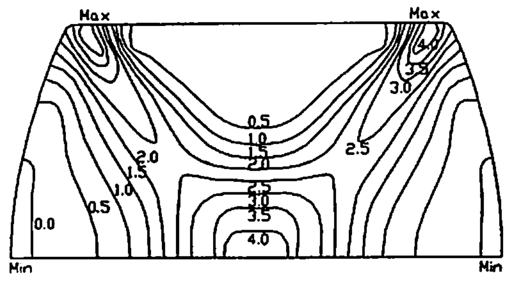
Figure 5.33 shows the load compression curve during the deformation process. It can be seen from this figure that the forming load obtained by developed software matches quite well with the forming load obtained by commercial softwares FORGE2 and ANSYS.



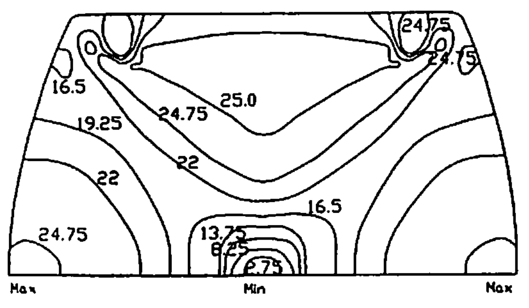
(a)



(b)

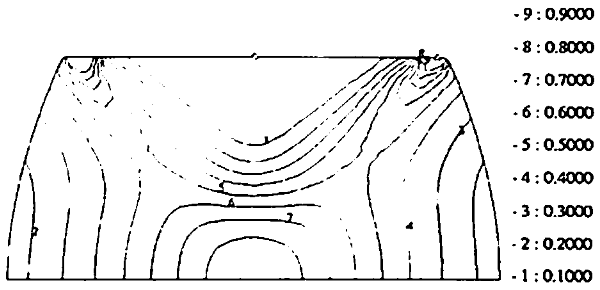


(c)

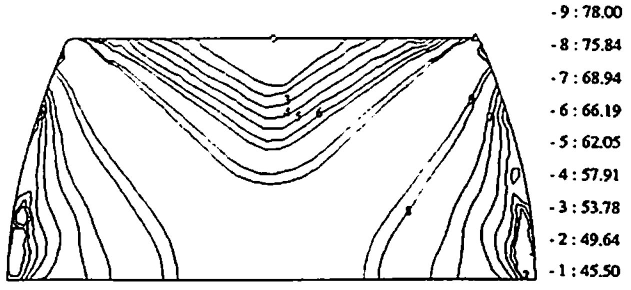


(d)

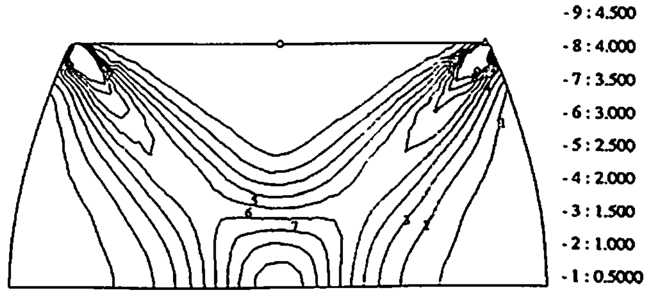
Fig. 5.31 Contours showing (a) effective strain (b) effective stress (MPa) (c) effective strain rate (d) nodal velocity (mm/s) distribution obtained by present investigation (FEMLD4)



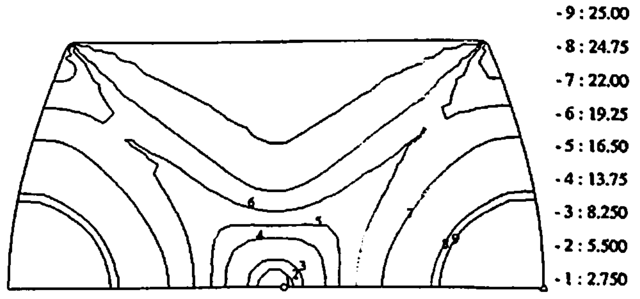
(a)



(b)



(c)



(d)

Fig. 5.32 Contours showing (a) effective strain (b) effective stress (MPa) (c) effective strain rate (d) nodal velocity (mm/s) distribution obtained by FORGE2 (Δ and O represent maximum and minimum respectively)

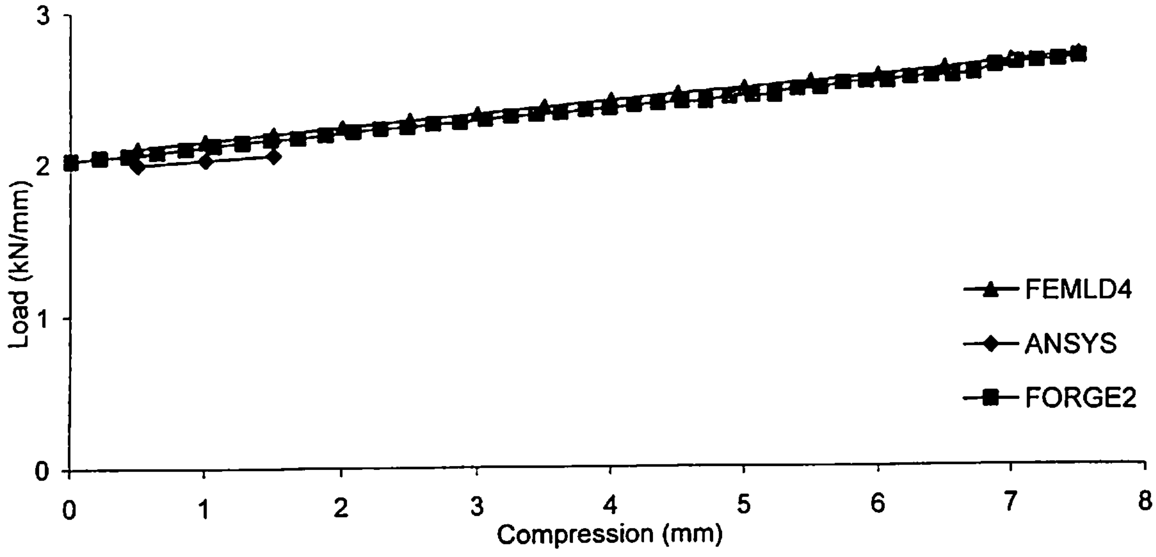


Fig. 5.33 Load compression relationship

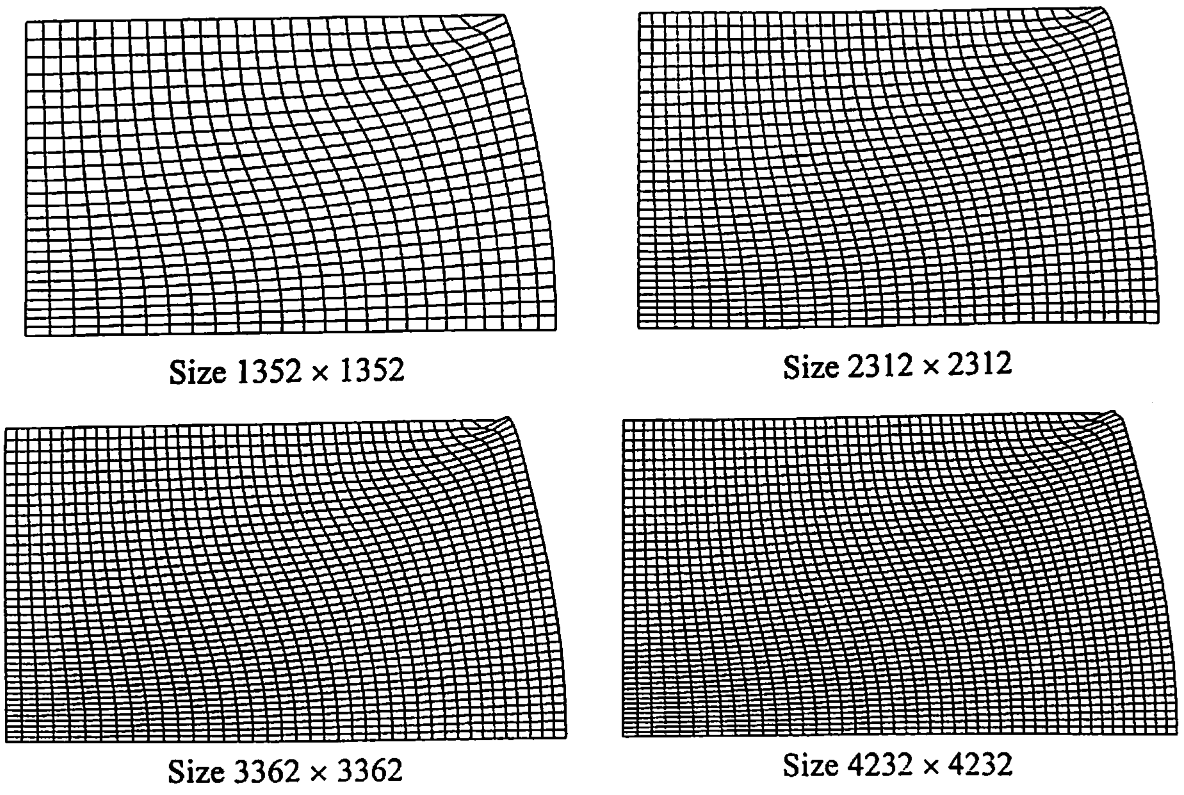


Fig. 5.34 Deformed shapes of axisymmetric problem with different mesh sizes obtained by FEMLD4

5.4 PERFORMANCE OF FEMLD

The main aim of using parallel computing technique in large deformation problems is to achieve more accurate results by adopting finer mesh and also to save computational time during the analysis. Therefore the developed software was tested for huge size large deformation problems. The axisymmetric problem (discussed in section 5.3.1) was once again analyzed by adopting different mesh sizes. The problem was discretized using rectangular elements in such a way that the global stiffness matrices of sizes 882 × 882, 1352 × 1352, 2312 × 2312, 3362 × 3362, and 4232 × 4232. The other problem details like number of steps, deformation, material law, and others were same as described earlier (section 5.3.1). These problems were analyzed by using different number of processors. Figure 5.34 shows the deformed shape of axisymmetric problem under consideration with different mesh sizes at the last step.

Table 5.3 to 5.7 shows the computational time measured in terms of Real time with number of processors for different mesh sizes. One can observe that computational time reduces considerably when higher number of processors were employed for the computation. Abrupt variation in Total time measured in terms of Real time can be observed which is mainly due to the users activities occurred during the execution processes. Similar behavior can also be observed for Communication time measured in terms of Real time. Figure 5.35 shows the variation in Total time per iteration measured in terms of User time (which do not get affected by user activities) with different data sizes for one to eight number of processors. It can be seen from this figure that as the data size increases, User time per iteration also increases. One can also observe that for higher data sizes the saving in computational time is on higher side as compared to the small data sizes. After multiple linear regression analysis, it was found that the curves shown in Fig. 5.35 could be expressed in the form of an equation

$$t = 2.67\times10^{-7} e^{(-0.6586p)} d^{(2.9598+0.0481p)} \tag{5.59}$$

where, t is the User time per iteration (in seconds), d is the size of global stiffness matrix and p is the number of processors involved during the analysis. With the help of this expression, one can roughly estimate the Total time measured in terms of User time per iteration for any data size and any number of processors.

Table 5.3 Computational time variation (RT) for problem of size 882 × 882

No. of processors	Total time	Comm	Cal	Speedup
1	9463.71	0.00	9463.71	1.00
2	6270.69	83.98	6186.71	1.51
3	4806.65	157.18	4649.47	1.97
4	3929.77	184.11	3745.66	2.41
5	3588.98	419.81	3169.17	2.64
6	3235.34	305.46	2929.88	2.93
7	3124.00	368.87	2755.13	3.03
8	3112.25	377.01	2735.24	3.04

Table 5.4 Computational time variation (RT) for problem of size 1352 × 1352

No. of processors	Total	Comm	Cal	Speedup
1	14607.72	0.00	14607.72	1.00
2	13288.26	243.53	13044.73	1.10
3	10759.79	322.58	10437.21	1.36
4	9578.46	237.65	9340.81	1.53
5	5226.60	522.19	4704.41	2.79
6	4636.89	286.09	4350.80	3.15
7	4225.23	380.99	3844.24	3.46
8	4136.61	395.24	3741.37	3.53

Table 5.5 Computational time variation (RT) for problem of size 2312 × 2312

No. of processors	Total	Comm	Cal	Speedup
1	92881.80	0.00	92881.80	1.00
2	61952.31	368.35	61583.96	1.50
3	85759.33	22095.40	63663.93	1.08
4	37474.09	948.99	36525.10	2.48
5	66298.74	8543.91	57754.83	1.40
6	45200.84	2638.59	42562.25	2.05
7	23730.11	1441.97	22288.13	3.91
8	21781.41	1628.93	20152.48	4.26

Table 5.6 Computational time variation (RT) for problem of size 3362 × 3362

No. of processors	Total	Comm	Cal	Speedup
1	228302.28	0.00	228302.28	1.00
2	157580.71	611.56	156969.15	1.45
3	117520.37	1184.18	116336.19	1.94
4	93380.70	1307.67	92073.03	2.44
5	81737.60	3859.53	77878.07	2.79
6	70164.69	2451.50	67713.19	3.25
7	93517.85	9096.67	84421.18	2.44
8	55338.32	2827.07	52511.25	4.13

Table 5.7 Computational time variation (RT) for problem of size 4432 × 4432

No. of processors	Total	Comm	Cal	Speedup
1	440082.40	0.00	440082.40	1.00
2	613710.00	5046.31	608663.69	0.72
3	513941.76	4847.03	509094.73	0.86
4	481525.04	7030.57	474494.47	0.91
5	451882.24	8695.81	443186.43	0.97
6	137742.00	11495.03	126246.97	3.19
7	410271.44	33544.78	376726.66	1.07
8	159397.28	10485.13	148912.15	2.76

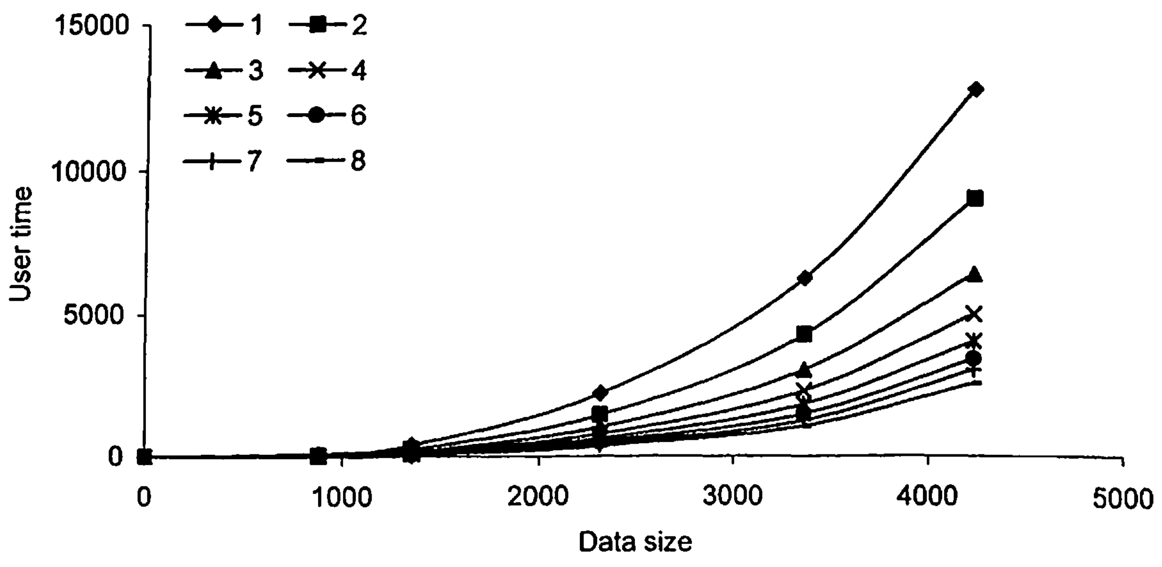


Fig. 5.35 Variation in User time per iteration with data size for various number of processors (1 to 8 shows No. of processors)

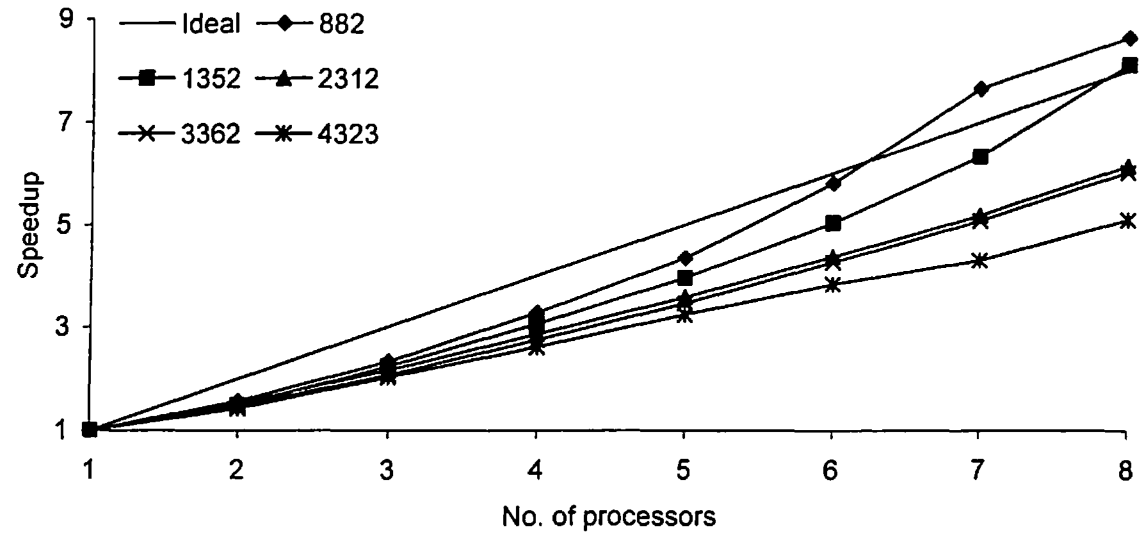


Fig. 5.36 Variation in User time Speedup for different data sizes

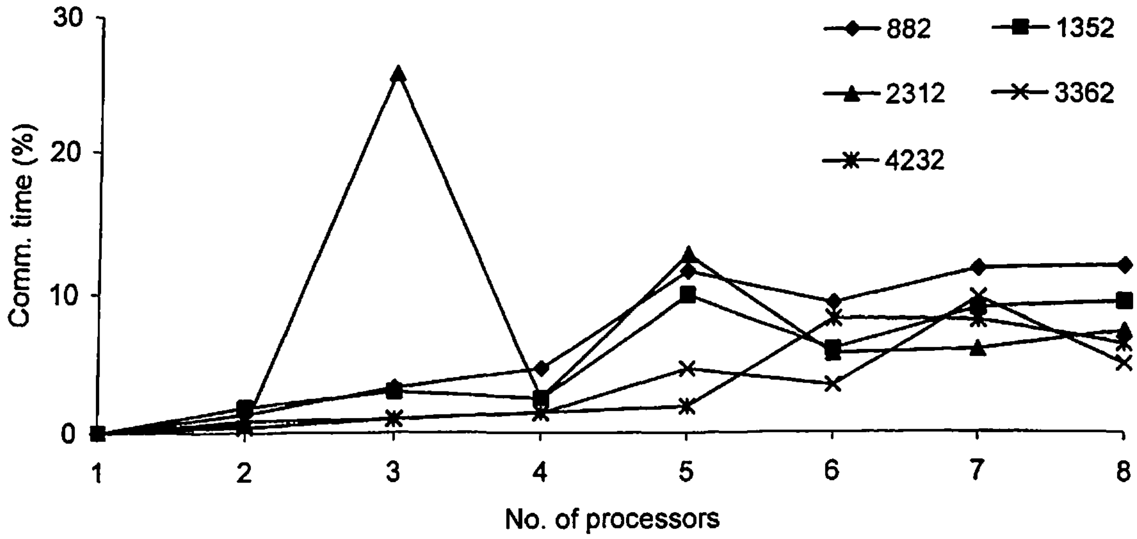


Fig. 5.37 Variation in percentage Communication time with number of processors for different data sizes

Figure 5.36 shows the variation in User time Speedup for different data sizes. It can be observed that, for smaller data sizes 882 × 882 and 1352 × 1352, the Speedup is even higher than the Ideal Speedup. This figure also shows that as the data size increases the Speedup reduces. This may be because of higher contribution of Communication time toward the Total time during the execution process. Figure 5.37 shows the variation of percentage Communication time with increase in number of processors. It can be seen that the percentage contribution of Communication time towards the Total time increases as the number of processors increases. It can also be seen that this contribution is also higher for higher data sizes. The variation in percentage Communication time is abrupt. This is mainly because of uneven user activities during the execution process (See Table 5.3 to 5.7).

5.5 SUMMARY

In general, various results obtained using codes FEMLD3, FEMLD4 and SPID were identical. It was observed that stress components obtained by FEMLD3 do not match with the corresponding results obtained from SPID. It was observed that when triangular elements are used the volumetric strain-rate is not very much closer to zero as compared to the rectangular elements. This makes triangular element a weaker element. The only advantage of using triangular element is that, it requires lesser computation in generating local stiffness matrix as compared to the rectangular element. The results obtained by FEMLD4 are also compared with the commercial softwares FORGE2 and ANSYS ported on computers with single processors. It was found that various results obtained by FEMLD4 match well with corresponding results obtained from commercial softwares.

It can be observed that the load compression curve obtained by commercial software ANSYS is only up to displacement of 4 mm (Fig. 5.18) and 1.5 mm (Fig. 5.33) for axisymmetric and plane strain problems respectively. It is mainly due to error occurred during the analysis carried out using ANSYS. As the deformation progresses, the few elements distort and after certain stage, these element do not satisfy the distortion parameters predefined in the ANSYS software. Therefore the complete solution of the problem could not found. Whereas, remeshing facility is available in FORGE2 code. Therefore during the compression process whenever any element gets distorted and does not satisfy the distortion parameters, automatic remeshing is carried out for the entire problem domain. It was observed that, remeshing was carried out three times in axisymmetric problem and seven times in plane strain problem discussed in section 5.3.1 and 5.3.2 respectively.

As far as performance of FEMLD3 and FEMLD4 is concerned, they showed good performance on supercomputer PARAM 10000. The Total time measured in terms of Real time reduces considerably after using higher number of processors of supercomputer PARAM 10000. The developed software is also tested by analyzing huge size problems. It was found that software performed well for higher data sizes also.

CHAPTER 6

COMPUTER SIMULATION OF METALLIC TUBES AS ENERGY ABSORBING ELEMENTS

6.1 INTRODUCTION

Metal forming process is a phenomenon of plastic deformation of metal piece (billet) between rigid tools (dies). Metallic parts of complex shape can be made by pressing billet between dies of desired shape. As the deformation progresses, metal flows between dies and attains final shape after appropriate degree of deformation. Hence emphasis should be given to simulate die surfaces in code development to solve the metal forming problems. In the previous chapter, compression of cylinder was carried out by prescribing downward nodal velocities on the boundary nodes. Due to this, it was observed that the mesh distorted severely (see Fig. 5.34) near point A (as per Fig. 5.4). Moreover, specifically for discretized mesh having global stiffness matrix of size 3362 × 3362 and 4232 × 4232, some part of the mesh around the point A moved in the upward direction, which cannot occur actually. Therefore to rectify this error, two flat die surfaces are simulated in this chapter to achieve proper deformed shapes of billet throughout the deformation process. This would not only predict correct deformed shapes during simulation but also allow user to simulate deformation of billet of any geometric shape between two rigid flat dies.

6.2 CONTACT PROBLEM

In the whole deformation process, some part of deforming body always remains in contact with rigid dies. At a typical stage of deformation process, some new part of deforming body comes in contact with the die and/or some part of deforming body that was already in contact with the die may loose the contact from the rigid die. This depends on the boundaries of deforming body and the rigid dies. At this instance, frictional forces present between the deforming body and the rigid die interfaces also varies. Collectively frictional force and die shape control the deformation of the deforming body. In order to simulate the flat die, contact problem is subdivided into two cases namely Case I: Initiate new contact nodes, and Case II: Terminate old contact nodes.

6.2.1 Case I: Initiate New Contact Nodes

For the computer simulation of flat die, a flat die with user-defined velocity is created instead of specifying nodal velocities in the beginning. Thereafter Y-coordinates of all the boundary nodes are compared with the Y-coordinate of flat die. At this juncture, if Y-coordinate of any node along the boundary matches with the Y-coordinate of the die then velocity of die was transferred to that node. This initiated the deformation process.

In the compression process after every increment of deformation, Y-coordinates of all the nodes are compared with the present Y-coordinate of the rigid die. At this juncture, if Y-coordinate of any node crosses the boundary of flat die then its velocity is adjusted. Figure 6.1(a) shows a typical finite element mesh at (i)th increment. It can be seen that node P is not in contact with upper die at (i)th increment but it crosses the boundary of upper die in the next increment (see Fig. 6.1(b)). Hence its velocity in Y-direction is adjusted so that its Y-coordinate remains same as Y-coordinate of moving upper die after (i+1)th increment. The expression for the modified velocity component will be as follows,

$$v_y^P = \left(y_{(i+1)}^{die} - y_{(i)}^{P}\right) / \Delta t$$

where v_y^P is the adjusted velocity of the node P, $y_{(i+1)}^{die}$ is the Y-coordinate of rigid die at (i+1)th increment, y_i^P is the Y-coordinate of the node P at (i)th increment and Δt is the time increment.

After this nodal velocity adjustment, the iterations are once again carried out till the correct shape is achieved (see Fig. 6.1(c)). During this procedure, velocity components of other nodes are automatically adjusted to meet the convergence criteria. Once the solution of that particular increment is obtained, the velocity conditions of new nodes those recently came in contact with the die are changed. The Y-component of velocity of such nodes is changed and made equal to the die velocity so that these nodes travel along with die in further iterations.

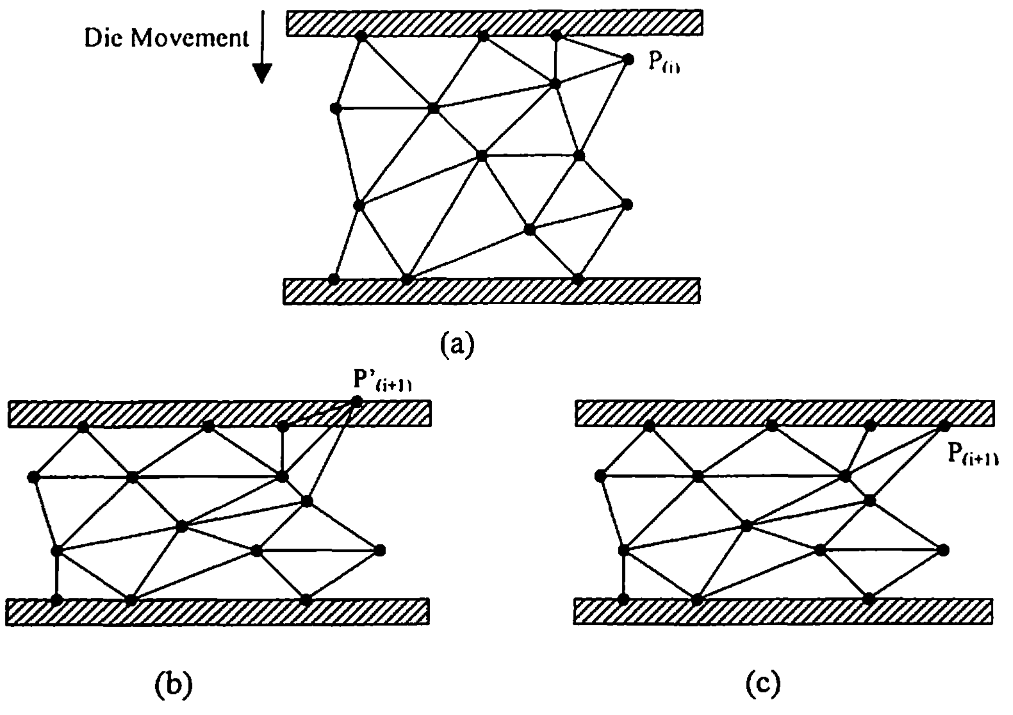


Fig. 6.1 Initiating new contact nodes: Typical deformed mesh at (a) (i)th iteration, (b) intermediate (i+1)th iteration and (c) final (i+1)th iteration

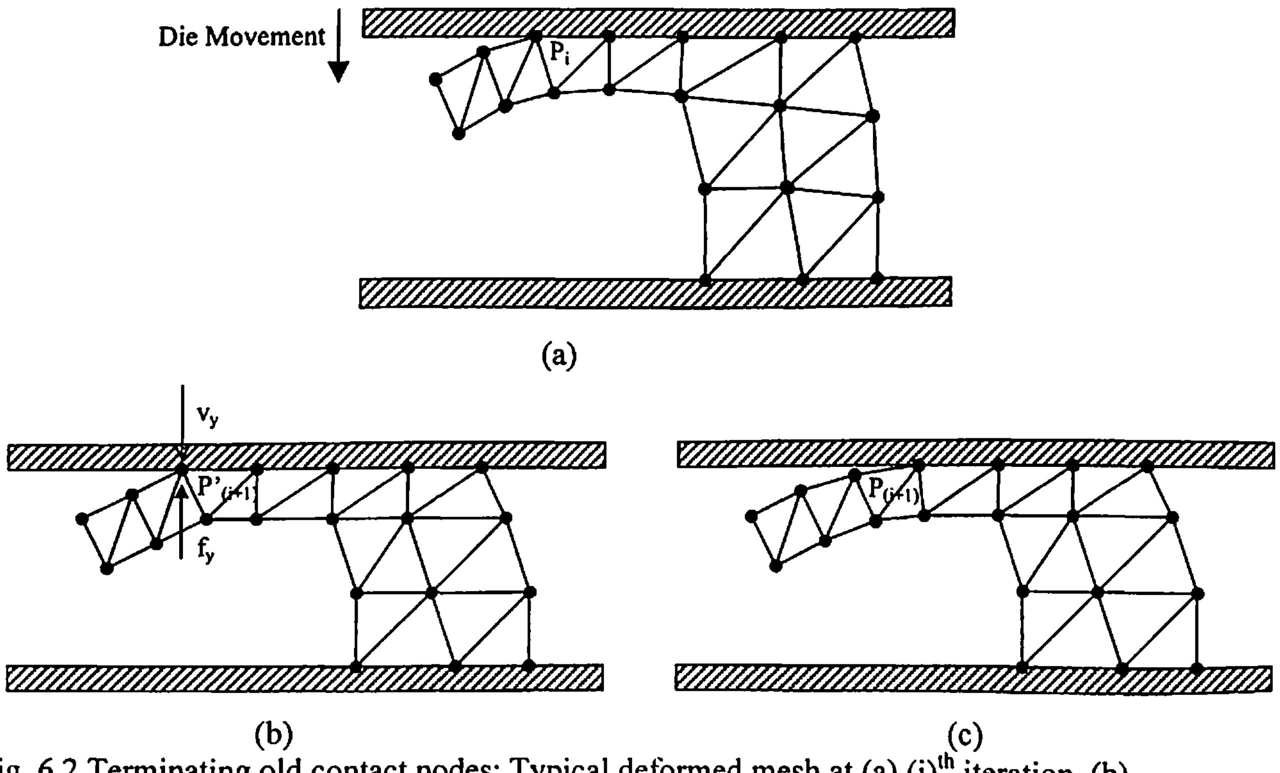


Fig. 6.2 Terminating old contact nodes: Typical deformed mesh at (a) (i)th iteration, (b) intermediate (i+1)th iteration and (c) final (i+1)th iteration

6.2.2 Case II: Terminate Old Contact Nodes

During the deformation process, velocity in Y-direction of the contact nodes and the compressing die were having same magnitude (see node P in Fig. 6.2(a)) till the direction of force (in Y-direction) and the velocity in Y-direction of the contact nodes were same. As soon as this condition for any contact node was not satisfied it was assumed that the node has been detached from the die (see node P in Fig. 6.2(b)). So at this juncture to incorporate this, such nodes were removed from contact with die by changing their boundary conditions (see node P in Fig. 6.2(c)). This condition was checked for all contact nodes after each increment to achieve true deformed shape after every increment.

Both the cases of contact conditions are incorporated in the developed parallelized computer code FEMLD. With the help of these modifications one can simulate deformation process of different materials with several shapes between the boundary of two flat dies. This is an ideal condition of simple compression within a two rigid flat moving dies of compression machine.

6.3 CASE STUDIES

After incorporating various contact conditions in the developed code, four case study problems are simulated using this code. Among these four, two are axisymmetric deformation problems and remaining two are plane strain deformation problems.

6.3.1 Axisymmetric Compression of Solid Cylinder

As mentioned in the literature [68], Gupta and Shah experimentally carried out axial compression of round solid cylinders made up of aluminium and low carbon steel. Several specimens with different geometric properties were compressed and their deformation processes were examined. From these specimens, a typical specimen of aluminium having 30 mm outer diameter (D) and 60 mm initial height (H_0) is taken for the present investigation. This specimen was compressed between flat rigid dies till 83.33% reduction in height was achieved (final height 10 mm). This specimen was compressed in 80 increments. Magnitude of frication factor ranging from 0.1 to 0.5 was taken for the study. The material behavior of the deforming cylinder was expressed by expression $\bar{\sigma} = k\dot{\bar{\varepsilon}}^m$, where numerical values of k and m were taken as 69.94 MPa and 0.1 respectively. Error norms (velocity error norm and force error norm) were considered as 0.001 and limiting strain-rate value was considered as 0.01 to define the rigid portion. The deforming cylinder was discretized using 450 four noded rectangular elements with 496 nodes, resulting in global stiffness matrix of size 992 × 992. Figure 6.3 shows the finite element model used for the present investigation.

In the compression process of the round cylinder, metal flows as the deformation progresses and the vertical face AB (see Fig. 6.3) attains curvilinear shape due to friction between the die and the cylinder interface. An arc of a circle with radius R can be approximately fitted to this deformed profile.

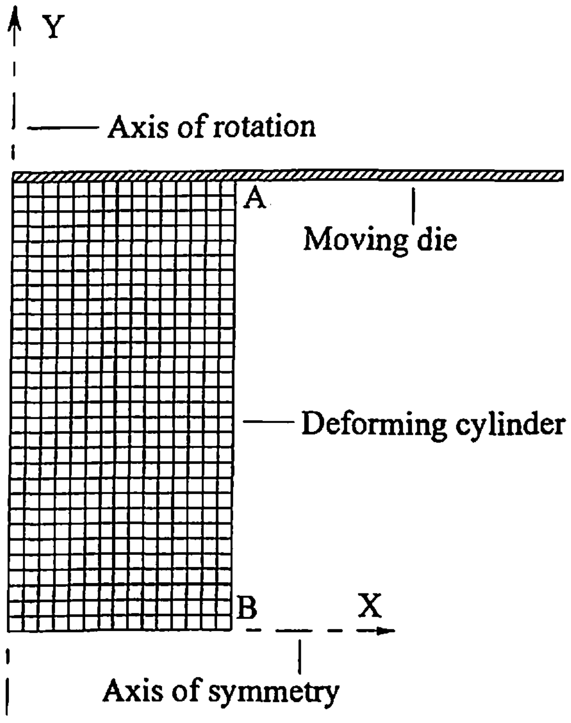


Fig. 6.3 Discretized finite element mesh for solid cylinder

Table 6.1 Table showing X-coordinate of point P for different values of friction factors

Friction factor	X-coordinate (mm)
0.1	30.55
0.2	30.43
0.3	30.20
0.4	28.37
0.5	28.19

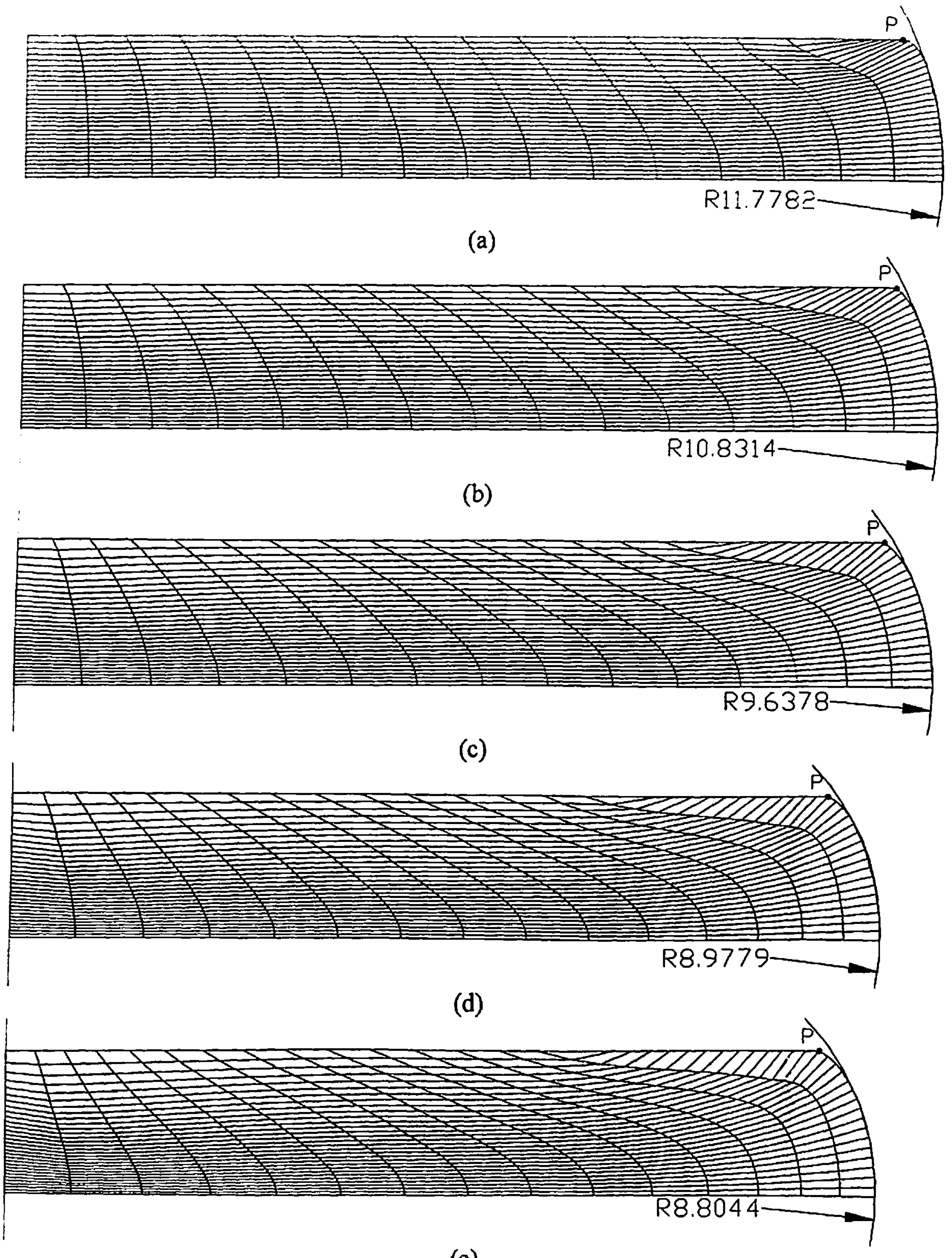


Fig. 6.4 Deformed shapes of solid cylinder after 83.33% compression with friction factor between die-cylinder interface equal to (a) 0.1 (b) 0.2 (c) 0.3 (d) 0.4 and (e) 0.5

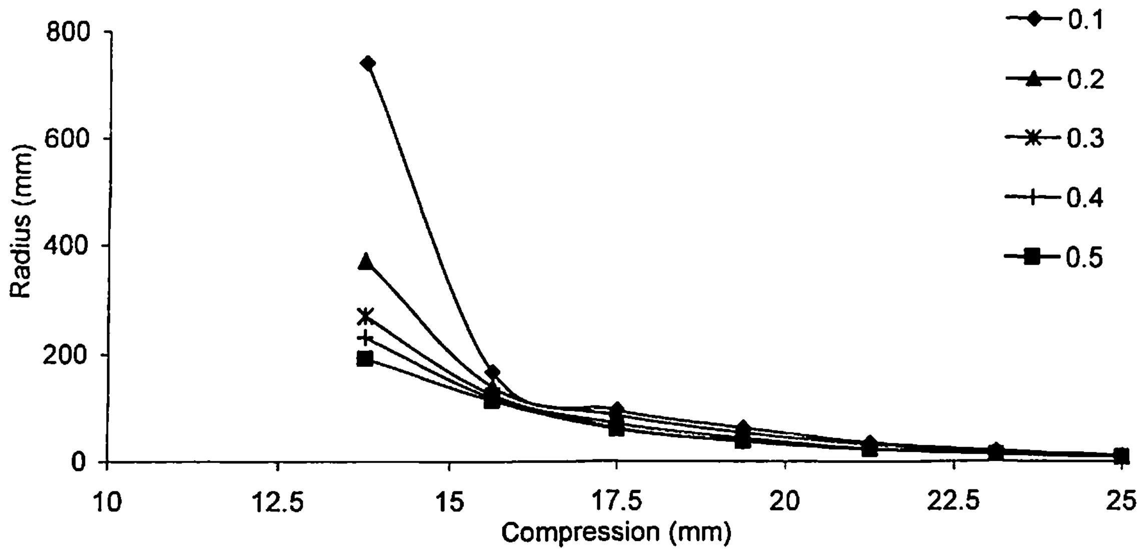


Fig. 6.5 Variation in radiuses of circles with compression for various values of friction factor between die-cylinder interface

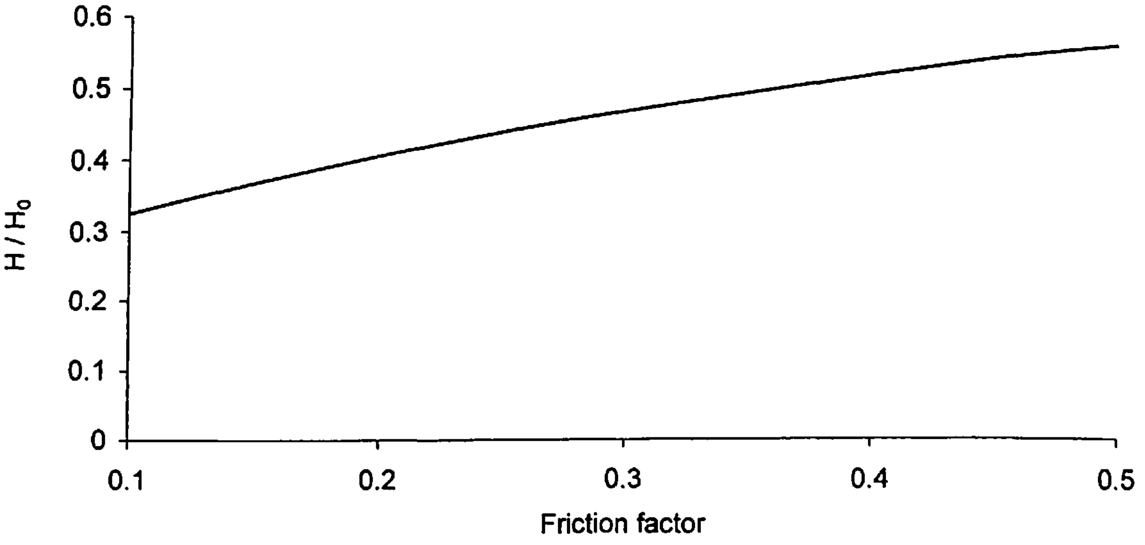


Fig. 6.6 Curve showing H/H_0 ratio with friction factor at which folding process begins

Figure 6.4 shows deformed shapes of solid cylinder obtained by considering various friction factor values. It can be seen from the deformed profiles that as the friction factor increases the radius of the circle reduces. X-coordinates of point P at 83.33% compression for different friction factor values are shown in Table 6.1 (where P is a point on vertical face AB that shows the final radius of the cylinder along loaded face after 83.33% compression). Figure 6.5 shows variation in these radii of circles with the progress of compression for various values of friction factors. Significant variation in the radii of circles can be observed in the initial stages of compression for all the values of friction factors. On the other hand, in the later stages of deformation, especially after the beginning of process of folding, insignificant variation in these radii can be seen. The process of folding begins when the upper portion of the vertical surface AB (near point A) comes in contact with the upper die. Figure 6.6 shows variation in H/H_0 ratio with friction factors at which folding process begins (where H is the height of the specimen at which folding process begins). It can be observed from this figure that, for higher friction factor values, folding process begins early (higher H/H_0 ratio) as compared to the lower friction factor values. The H/H_0 ratio increases with increase in the friction factor values. Load compression relationship for this compression process is shown in the Fig. 6.7. It can be seen that variation in the magnitude of load with different friction factors is insignificant in the early stages of deformation while marginal variation can be seen in the last stage of deformation. The deformations at which folding process begins are also illustrated in this figure by different markers (♦, ✳, •, ▲, ■ represents 0.1, 0.2, 0.3, 0.4, 0.5 respectively).

As present investigation includes parametric study, several executions on finite element mesh shown in Fig. 6.3 are involved; hence supercomputer PARAM 10000 is employed to save computational time. Figure 6.8 shows the variation in components of computational time per iteration with number of processors of supercomputer PARAM 10000. It can be seen that Total time measured in terms of Real time as well as User time reduces with increase in number of processors. Calculation time also reduces with increase in number of processors. Increase in Communication time with increase in number of processors can also be seen in this figure. Variation in Speedup with number of processors is shown in Fig. 6.9. It can be observed from this figure that Real time Speedup as well as User time Speedup increases with increase in number of processors.

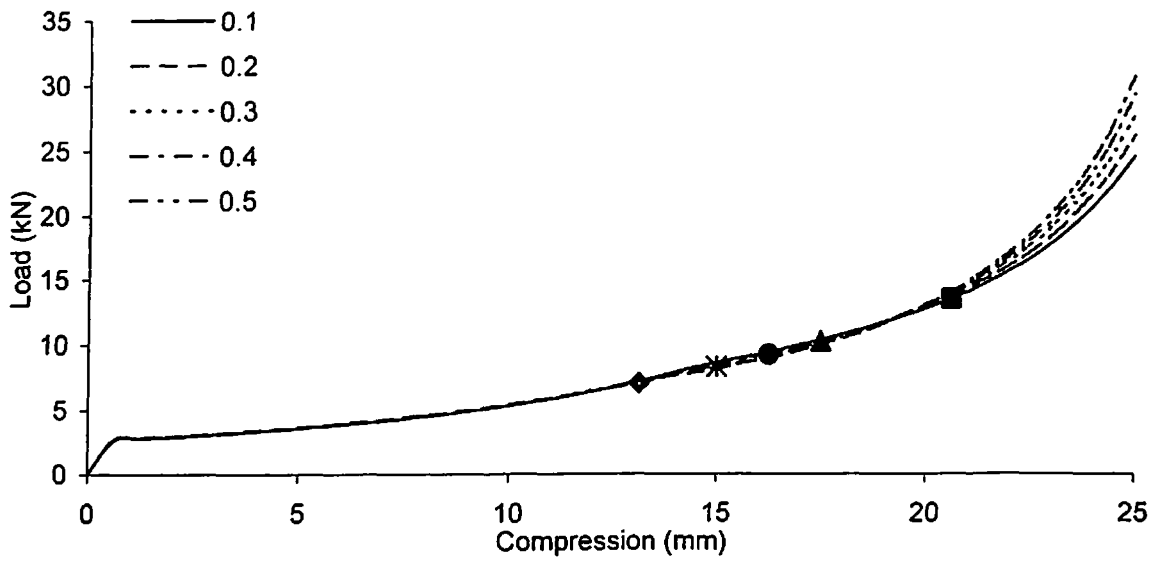


Fig. 6.7 Force deformation relationship for axial compression of solid cylinder for various friction factors

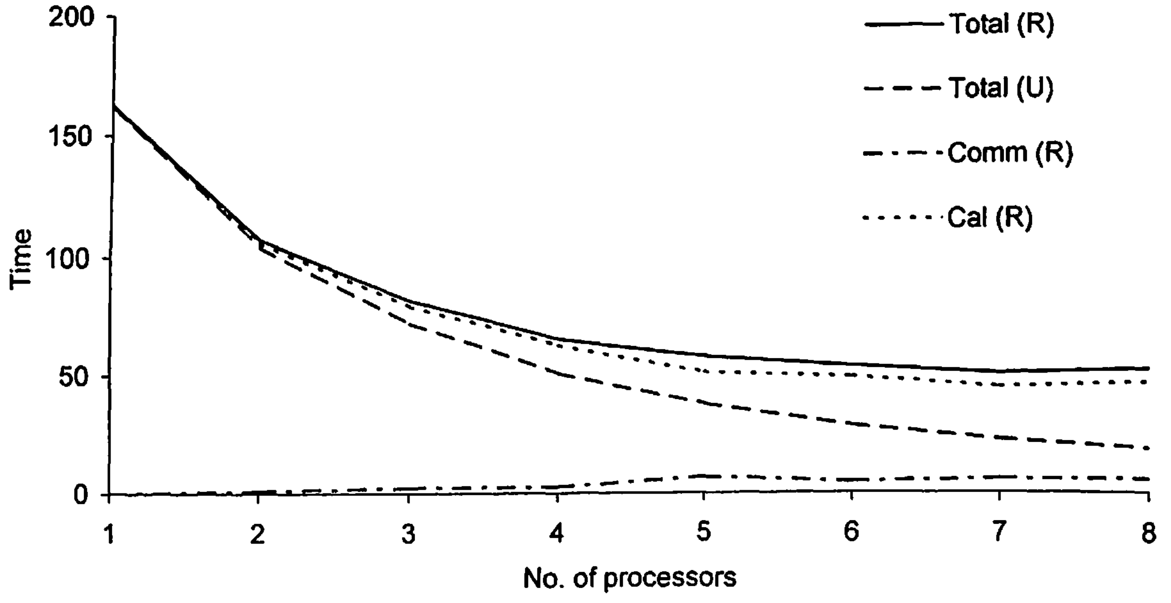


Fig. 6.8 Variation in the components of computational time per iteration with number of processors for axisymmetric compression of solid cylinder

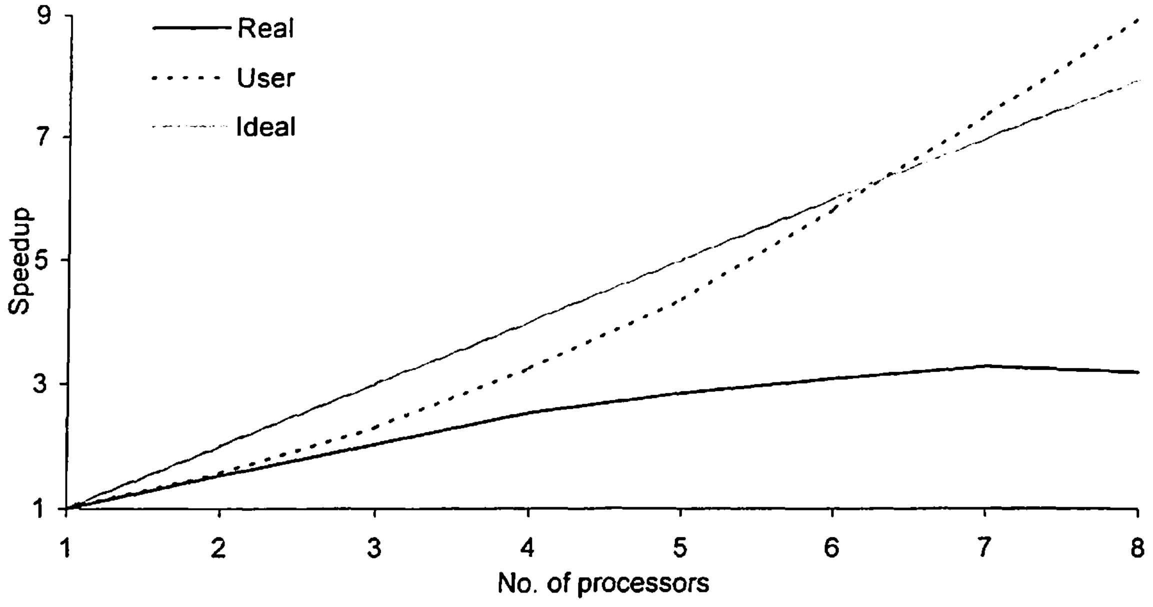


Fig. 6.9 Variation in speedup with number of processors for axisymmetric compression of solid cylinder

Table 6.2 Tube specifications

Tube No.	Tube Dimensions (mm)		Thickness (mm)			Material properties		
	Width	Height	Top	Bottom	Vertical	K_{ot}	m	a
AS26261	24.08	24.08	0.72	1.12	0.94	105.5	0.02853	4.17
AS26262	25.44	25.44	1.74	2.2	2.18	108.5	4.16	0.02853
AS26263	25.36	25.36	2.34	2.34	2.2	108.5	4.11	0.02953
AS26264	26.82	26.82	3.52	3.75	3.63	108.5	4.16	0.02853
AS26265	26.93	26.96	4.37	4.63	4.44	108.5	4.16	0.02853
AS26266	27.12	27.12	5.42	5.50	5.38	108.5	4.16	0.02853

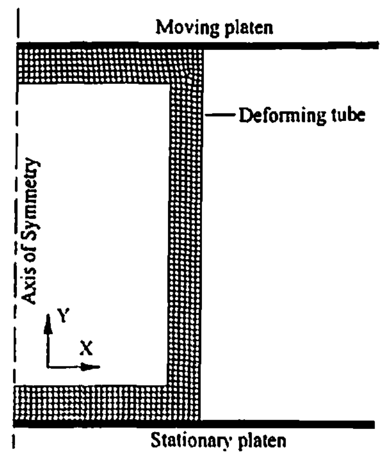


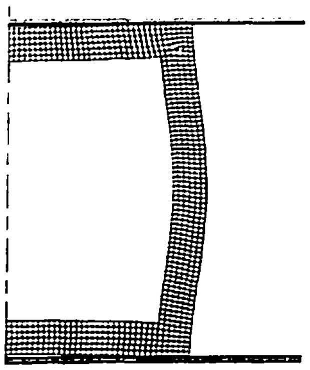
Fig. 6.10 Discretized finite element mesh for AS26263

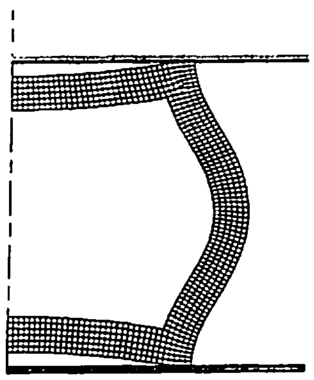
Real time Speedup follows linear Speedup pattern and User time Speedup follows superlinear Speedup pattern that is even greater than Ideal Speedup. Maximum Real time Speedup and User time Speedup of 3.29 and 9.01 are obtained at seven and eight number of processors respectively. In general significant saving in computational time and good performance of the parallel code is obtained on supercomputer PARAM 10000.

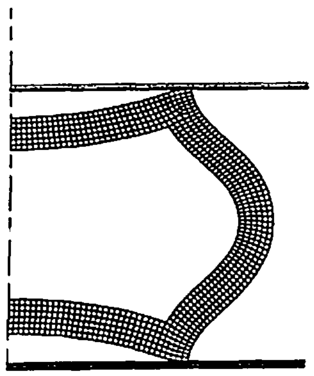
6.3.2 Lateral Compression of Rectangular Tubes

Experimental study on lateral compression of square and rectangular metallic tube was carried out by Gupta et al. [47-49] in past. These tubes were compressed between two flat rigid dies with stationary bottom die and moving top die with the downward speed of 10 mm/min. They also carried out computational study to simulate the experimental process of flattening of rectangular tubes using commercial software FORGE2. They developed a finite element model to study the collapse mechanism of these metallic tubes. Here, to test the functioning of the developed code on supercomputer PARAM 10000, simulation of lateral compression of few rectangular tubes are carried out. Finite element model presented by Gupta et al. [49] is used for the present investigation. Table 6.2 shows the geometric and material properties of the six mild steel tubes used for study. Figure 6.10 shows the discretized mesh of the specimen tube AS26263 using the developed code. Mesh consists of 891 nodes with 758 rectangular elements resulting in global stiffness matrix of size 1782 × 1782. This tube is compressed till tube final height of 15.06 mm is achieved in 200 increments. Figure 6.11 shows the deformed shapes at various stages of deformation obtained by present investigation.

To validate the results obtained from present investigation, specimen tube AS26263 is also analyzed using the commercial software FORGE2. Figure 6.12 shows the deformed shapes obtained using FORGE2 at various stages of compression. It can be seen from Fig. 6.11 and 6.12 that the deformed shapes obtained by present investigation and using commercial software FORGE2 match fairly well at various stages of deformation.


(a) 2.11 mm comp.


(b) 4.43 mm comp.


(c) 6.54 mm comp.

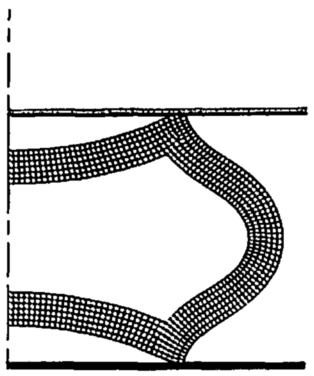
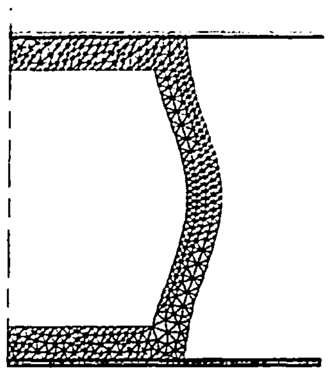
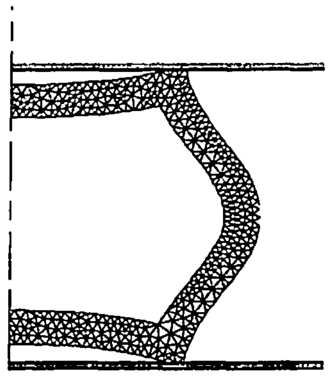
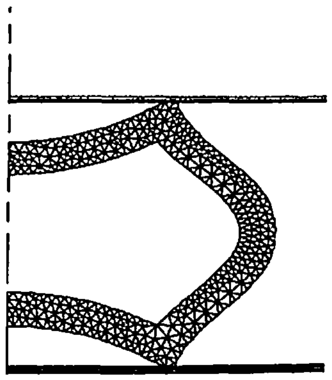

(d) 8.08 mm comp.

Fig. 6.11 Deformed mesh at various stages of compression obtained by present investigation.


(a) 2.12 mm comp.


(b) 4.44 mm comp.


(c) 6.52 mm comp.

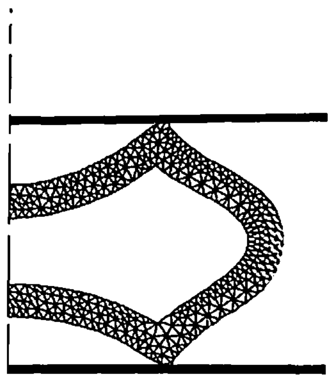
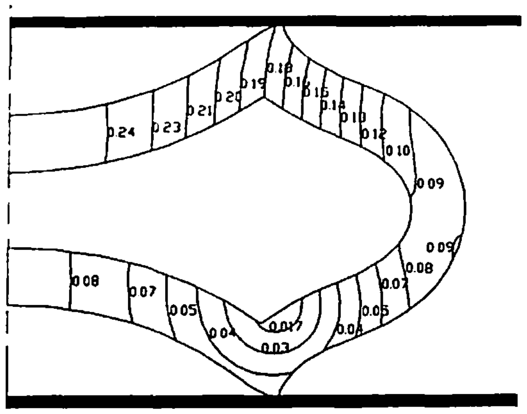
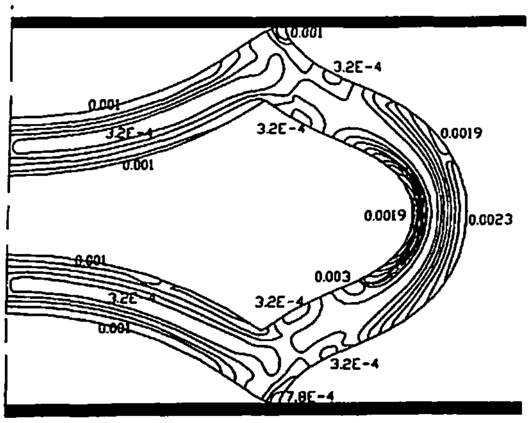

(d) 8.10 mm comp.

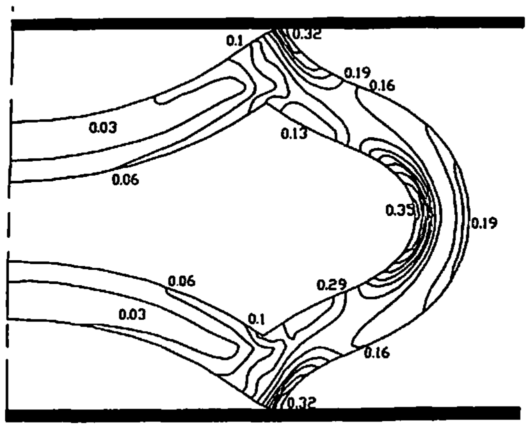
Fig. 6.12 Deformed mesh at various stages of compression obtained by commercial software FORGE2



(a)

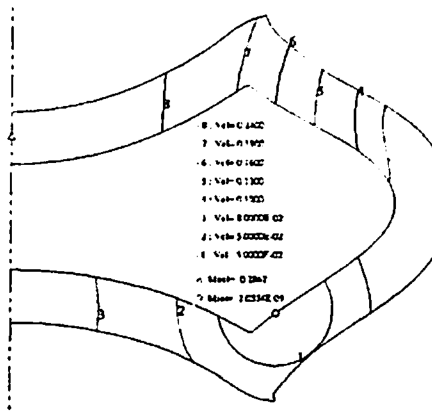


(b)

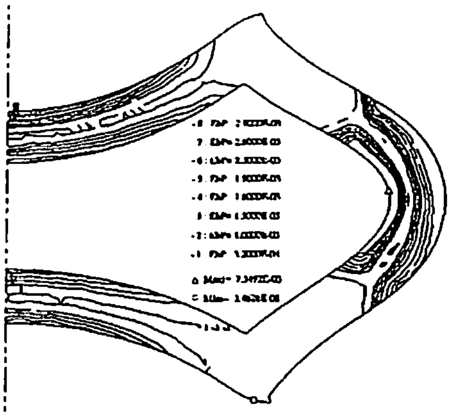


(c)

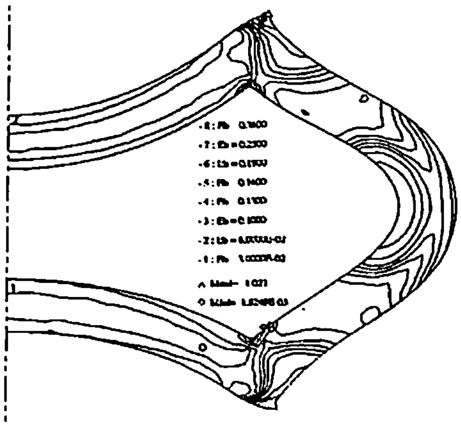
Fig. 6.13 Contours of (a) velocity (mm/s) (b) effective strain rate and (c) equivalent strain at 10.3 mm compression for specimen AS26263 obtained by present investigation



(a)



(b)



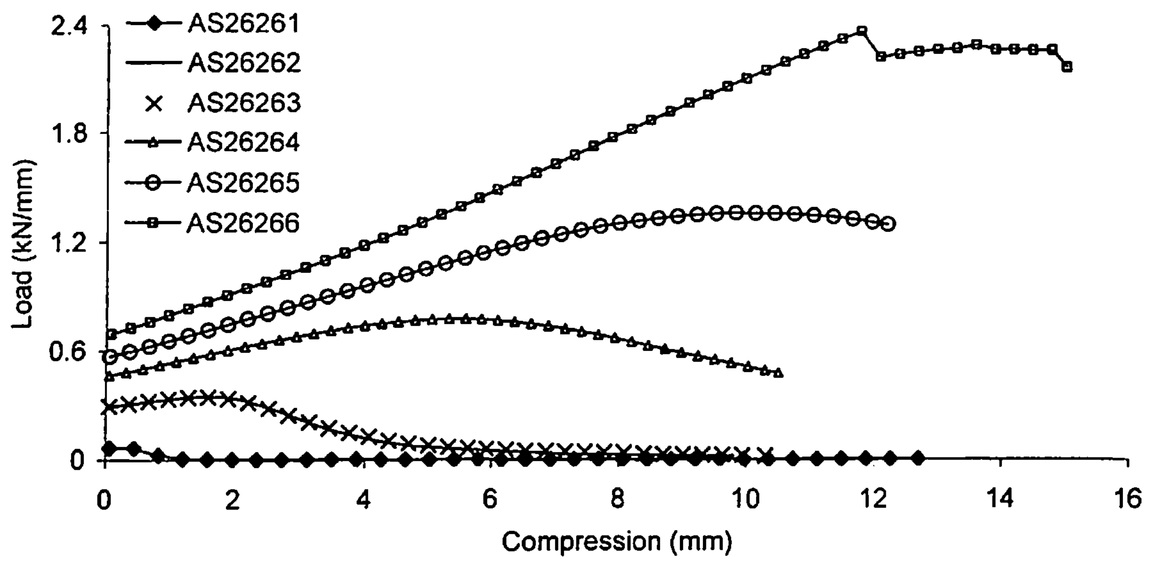
(c)

Fig. 6.14 Contours of (a) velocity (mm/s) (b) effective strain rate and (c) equivalent strain at 8.10 mm compression for specimen AS26263 obtained by FORGE2

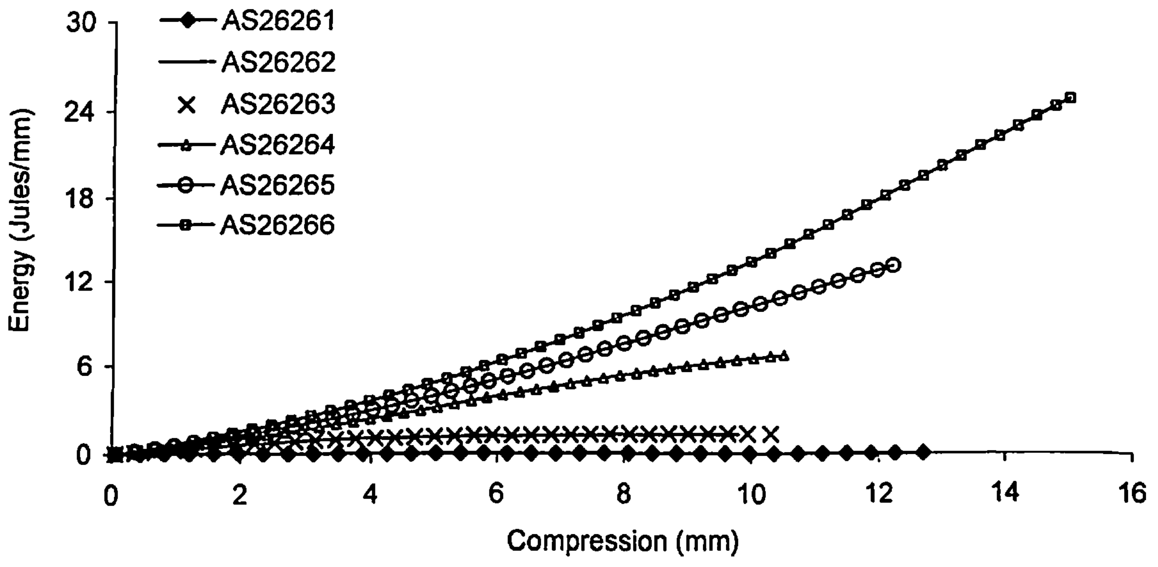
Figure 6.13 (a), (b) and (c) shows the contours of velocity, effective strain-rate and effective strain respectively after 10.3 mm compression for specimen AS26263 obtained by present investigation. Figure 6.14 (a), (b) and (c) shows the contours of velocity, effective strain-rate and effective strain respectively after 8.10 mm compression of the same specimen obtained using FORGE2. It can be seen in Fig. 6.13 and 6.14 that the comparison of distribution of contours obtained using developed code and FORGE2 are presented at different stages of compression (compression values are different). This is because the deformed profiles are almost identical at these stages of compression. After comparison it was found that the contours of various entities obtained by present investigation match fairly well with the contours of same entities obtained by FORGE2.

The load compression relationship for all six tubes under consideration obtained by present investigation is shown in Fig. 6.15 (a). It can be seen that the deforming load initially increases with the increase in compression. After reaching its peak value, the force starts decreasing till the end of compression process. It can be observed that the variation of load displacement curves for all tube specimens is identical. It can also be seen that as the wall thickness of the tube increases the load carrying capacity of the tube also increases. The load compression curves are drawn up to the displacement at which the top portion and the bottom portion of the tubes nearly touch each other. It can be observed that top and bottom portion of thickest tube specimen AS26266 touches each other before load reaches its peak value. Figure 6.15 (b) shows the variation in absorbed energy with compression. It can be seen from this figure that as the wall thickness of rectangular tube specimen increases the absorbed energy also increases.

Above discussed compression process of six tubes was simulated on supercomputer PARAM 10000 by increasing number of processors. Table 6.3 shows the details of the simulation study carried out on supercomputer PARAM 10000. Table shows deformation, size of stiffness matrix and number of iterations required to obtain the complete solutions for all six tubes under consideration.



(a) Load compression relationship



(b) Energy compression relationship

Fig. 6.15 Load compression and energy compression relationship for the rectangular tubes

Table 6.3 Details of simulation study on supercomputer PARAM 10000

Tube No.	Deformation (mm)	Matrix size	Iterations
AS26261	11.05	1980	496
AS26262	9.8	2010	421
AS26263	10.3	1782	420
AS26264	10.5	1946	418
AS26265	12.2	1896	417
AS26266	15.0	1930	507

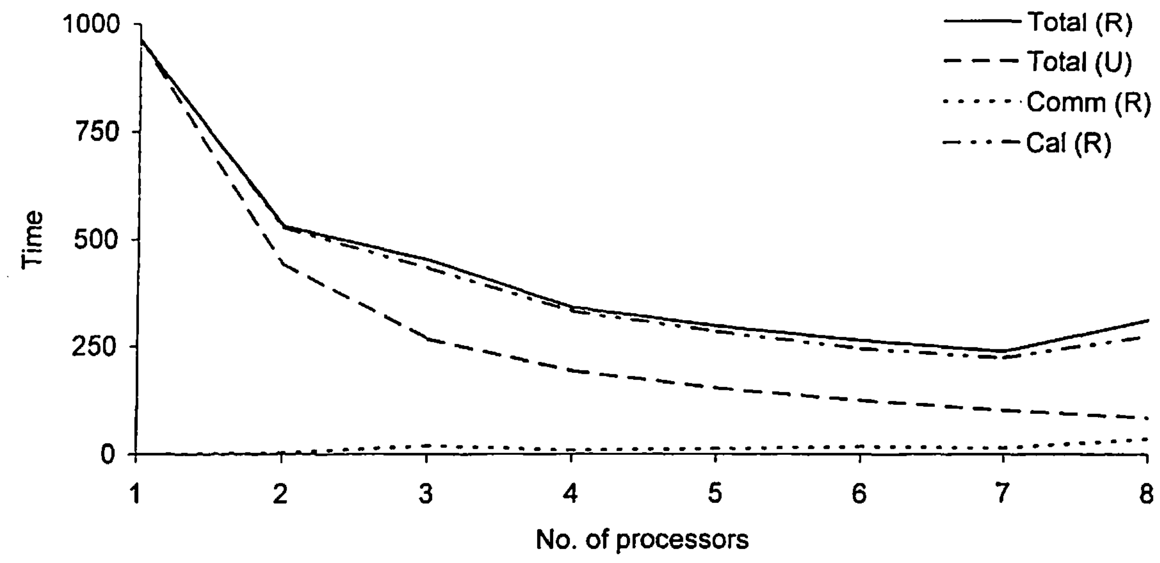
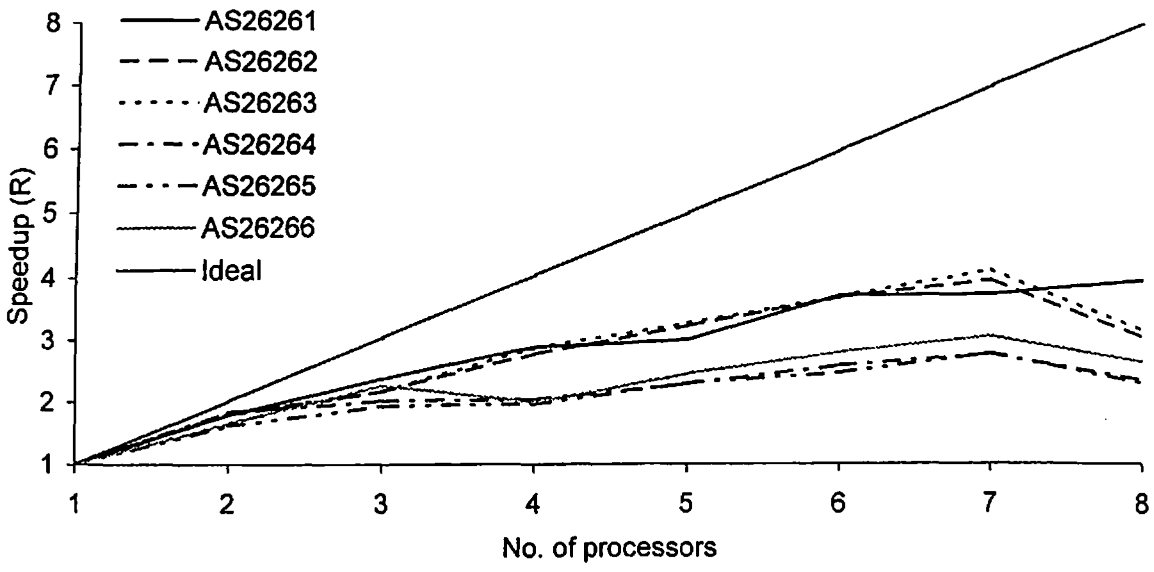
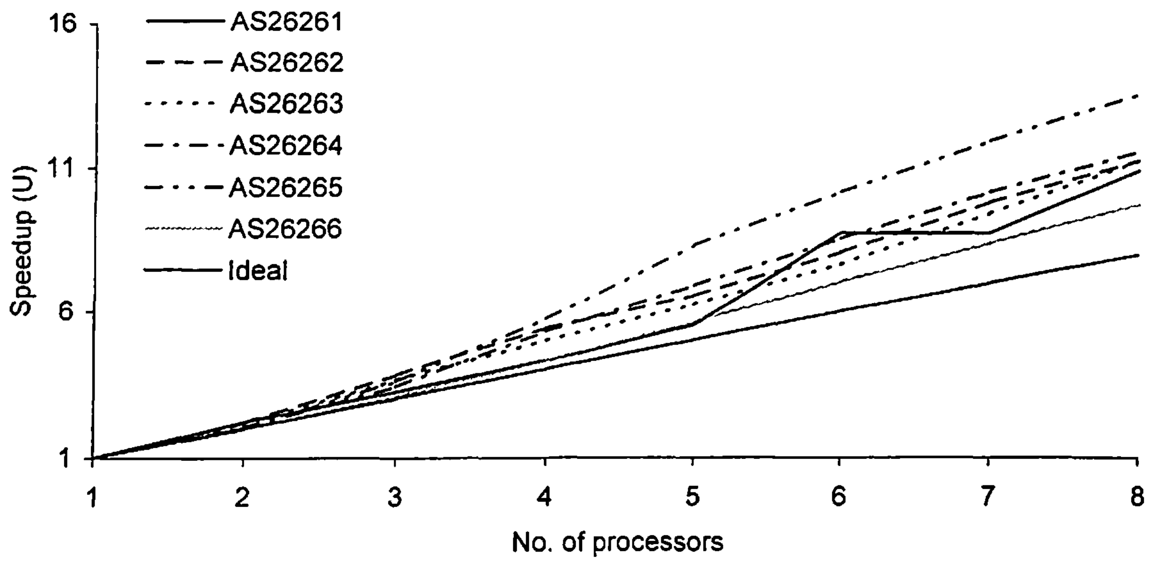


Fig. 6.16 Computational time variation for specimen AS26263



(a) Variation in Real time Speedup



(b) Variation in User time Speedup

Fig. 6.17 Variation in speedup with number of processors for six specimens under consideration

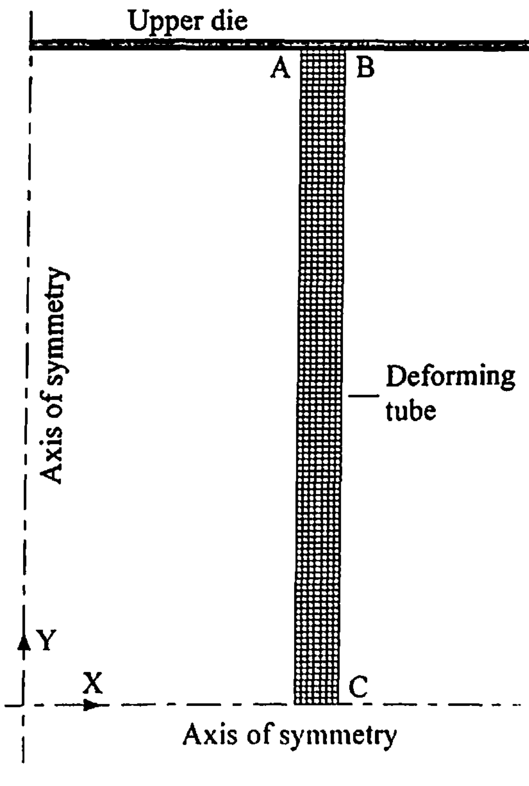


Fig. 6.18 Discretized finite element mesh for aluminum round tube AAC503

Figure 6.16 shows a typical variation in computational time per iteration obtained for tube specimen AS26263. It can be observed that Total time measured in terms of Real time as well as User time reduces with the increase in number of processors of supercomputer PARAM 10000. Calculation time also decreases with the increase in number of processors of supercomputer PARAM 10000. Increase in Communication time with increase in number of processor can also be observed in this figure. Similar type of variation was also observed for all the simulations for other tube specimens on supercomputer PARAM 10000.

Variation in Real time Speedup and User time Speedup is shown in Fig. 6.17 (a) and (b) respectively. It can be observed that Real time Speedup as well as User time Speedup increases with increase in number of processors. For every specimen, peak Real time Speedup was obtained at seven number of processors. In case of User time Speedup, it continuously increases with increase in number of processors employed for the simulation. Hence maximum User time Speedup was obtained at eight number of processors in each case. The highest Real time Speedup obtained was 4.07 at seven numbers of processors for specimen AS26263. On the other hand highest User time Speedup obtained was 13.54 at eight numbers of processors for specimen AS26265 that is even greater than the Ideal Speedup.

6.3.3 Axial Compression of Round Tube

A study on fold formation in axisymmetric axial compression of round metallic tube was carried out experimentally as well as computationally by Gupta et al. [53-55]. An attempt has been made to simulate the same compression process using the developed code. Finite element model used by Gupta et al. [53] is used in the present investigation. Aluminium round tube specimen AAC503 having average diameter and thickness of 47.66 mm and 3.44 mm respectively is modeled using the developed code. The height of circular tube is considered as twice the outer diameter of the tube as taken by Gupta et al. [53]. Figure 6.18 shows the discretized finite element mesh for the specimen tube AAC503. Mesh consists of 880 nodes with 763 rectangular elements resulting in global stiffness matrix of size 1760 × 1760. The material property of the deforming tube is expressed by expression $\bar{\sigma} = k(1 + a\bar{\varepsilon})\dot{\bar{\varepsilon}}^{m}$, where numerical values of k, a and m were taken as 239.9 MPa, 1.19

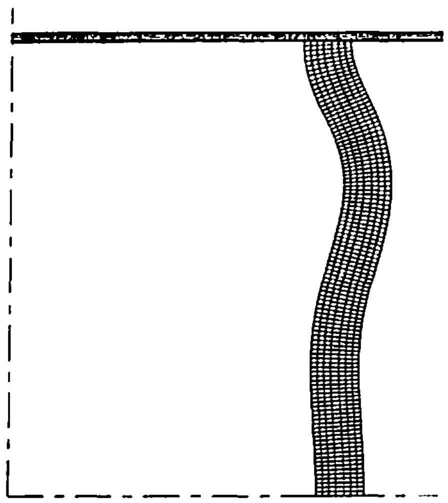
and 0.0034 respectively. The friction factor between the deforming material and rigid die interface is considered as 0.45. This tube is compressed between rigid dies with stationary bottom platen and moving top platen with the downward velocity of 10 mm/min. Compression is carried out till 66.8% reduction in height is achieved (final height of specimen is 68.28 mm) in 116 increments.

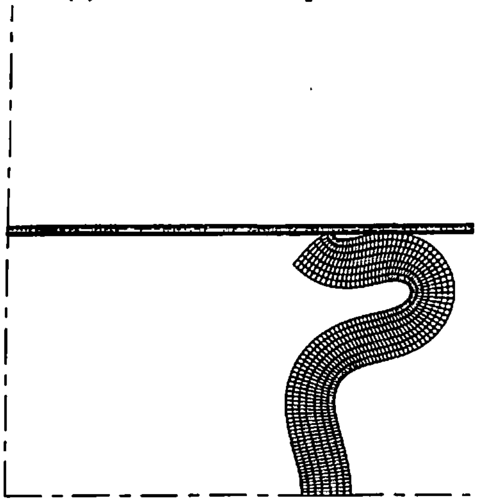
Figure 6.19 shows the deformed shapes at various stages of deformation. Figure 6.19 (a) shows the deformed shape when full loaded face AB is in contact with upper die. Figure 6.19 (b) shows the deformed shape when some portion of vertical face BC comes in contact with upper die. In Fig. 6.19 (c), deformed shape is shown when fold is fully formed.

Similar simulation is also carried out using commercial software FORGE2. Figure 6.20 shows the deformed shapes obtained using FORGE2 at various stages of deformation depicted in Fig. 6.19. It can be observed that deformed shapes at various stages of deformation obtained from developed code match fairly well with the deformed shapes obtained by FORGE2.

The contours of velocity, effective strain-rate and equivalent strain at 34.14 mm and 29.31 mm compression obtained by developed code and FORGE2 are shown in Fig. 6.21 and 6.22 respectively. It can be observed that contours obtained by developed code match well with the corresponding contours obtained by FORGE2 that validates the developed code.

The load compression relationship is shown in the Fig. 6.23 (a). It can be observed that the load initially increases with the increase in displacement. After reaching the peak value, load starts decreasing with the increase in compression. Sudden increase in the load can be seen at 30.6 mm compression. This is because some portion of the vertical face BC comes in contact with the moving die, that resulted in the decrease in lever arm for the applied load. Further increase in the compression resulted in the continuous reduction in the load. The relationship between absorbed energy and compression is shown in the Fig. 6.23 (b). It can be seen that the absorbed energy continuously increases with increase in the compression.


(a) 16.48 mm compression


(b) 30.61 mm compression

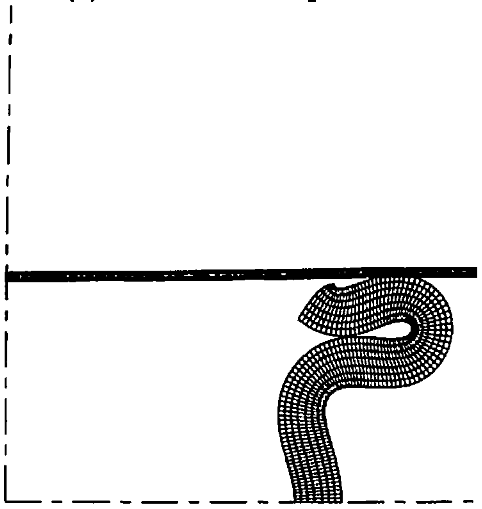
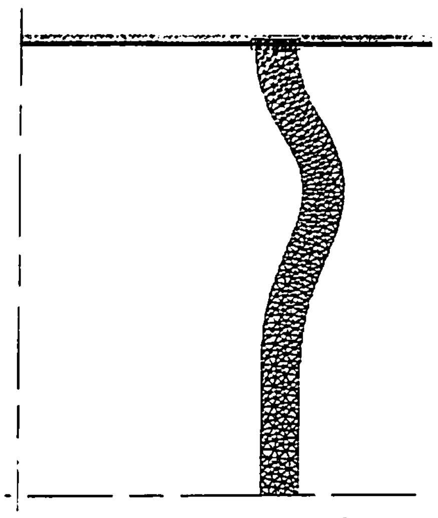
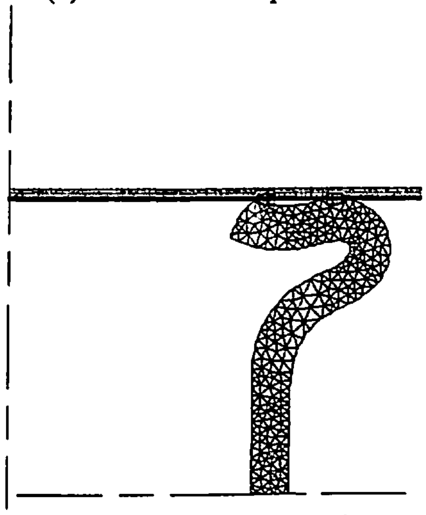

(c) 34.14 mm compression

Fig. 6.19 Deformed mesh at various stages of compression obtained by developed code


(a) 8.53 mm compression


(b) 23.54 mm compression

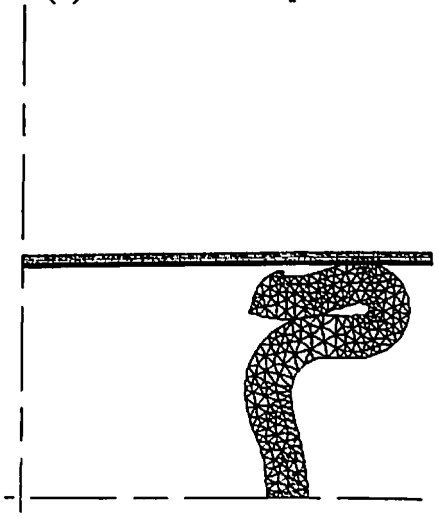
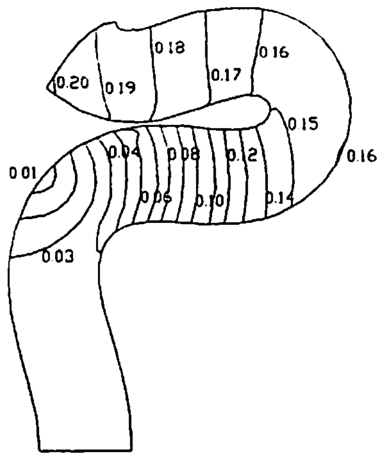
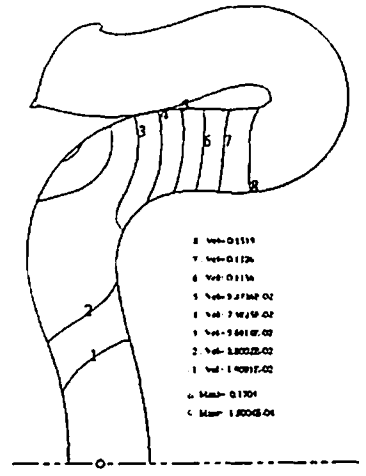

(c) 29.31 mm compression

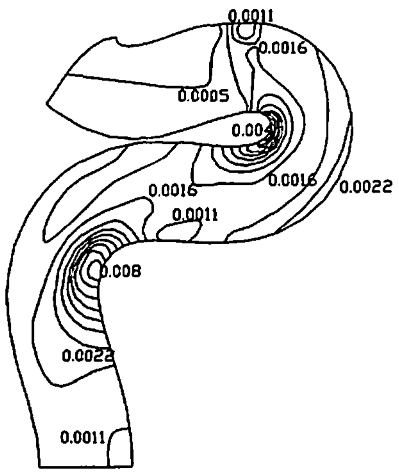
Fig. 6.20 Deformed mesh at various stages of compression obtained by commercial software FORGE2



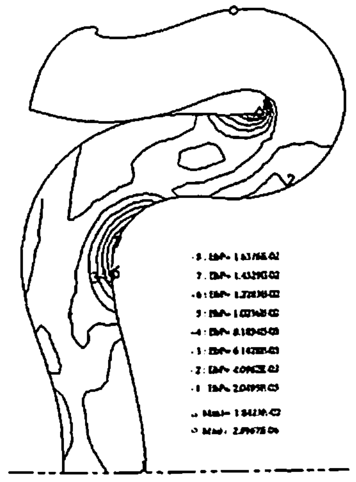
(a)



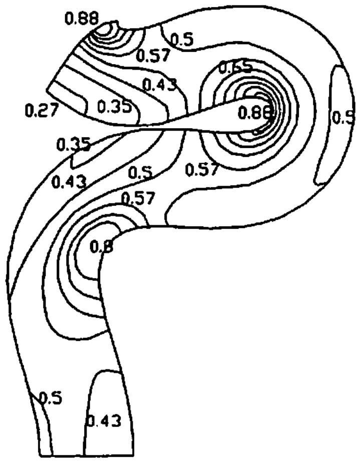
(a)



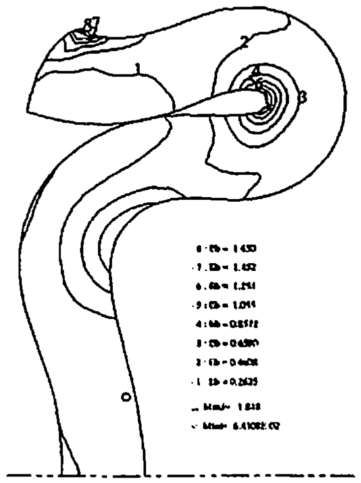
(b)



(b)



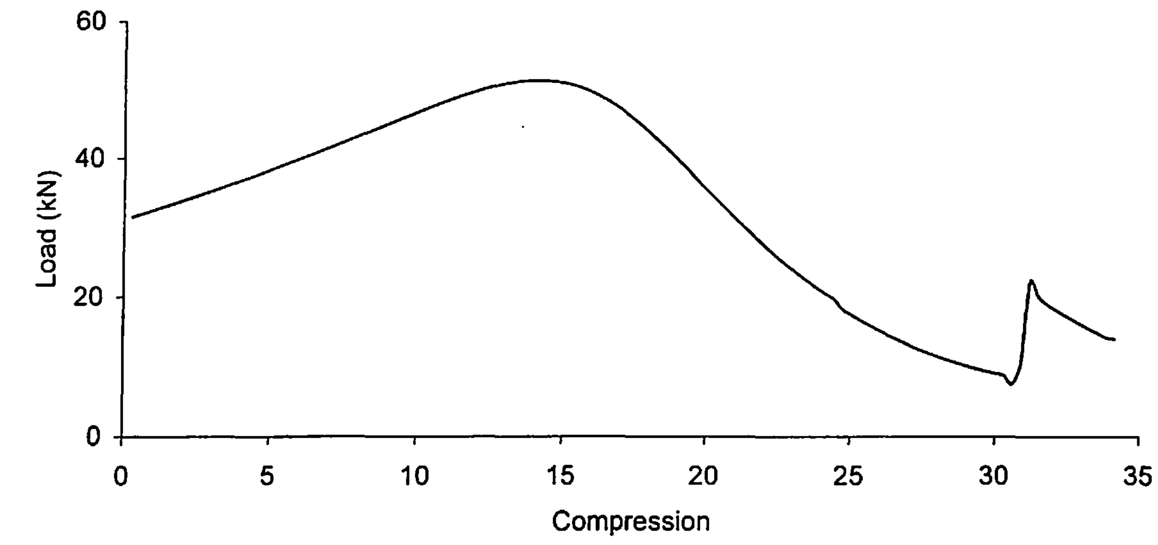
(c)



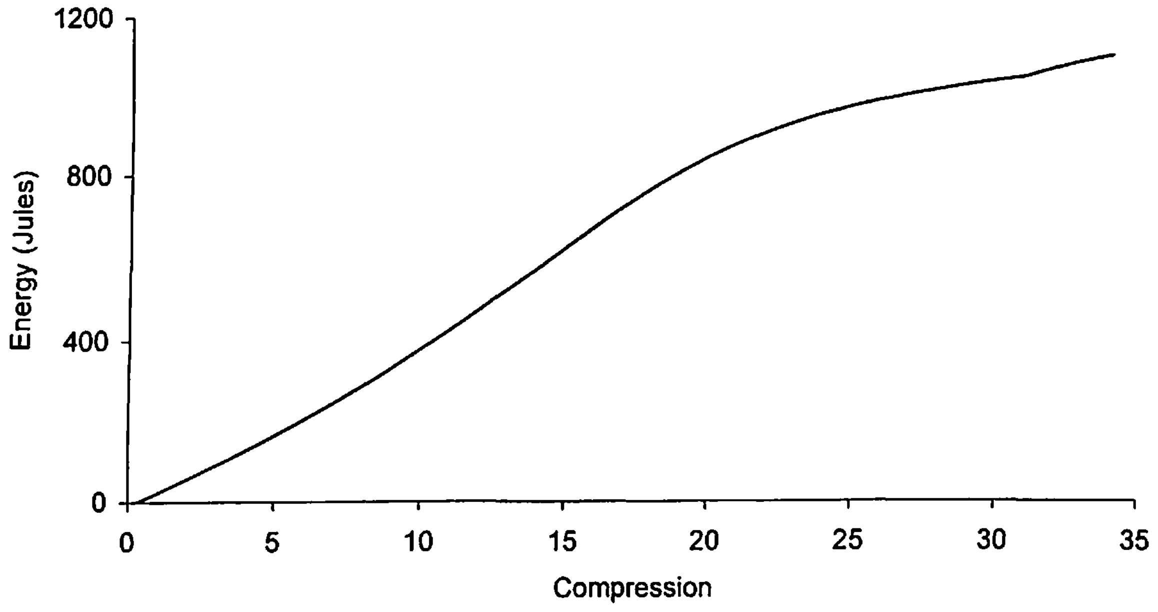
(c)

Fig. 6.21 Contours of (a) velocity (mm/s) (b) effective strain rate and (c) equivalent strain at 34.14 mm compression obtained by present investigation.

Fig. 6.22 Contours of (a) velocity (mm/s) (b) effective strain rate and (c) equivalent strain at 29.31 mm compression obtained by FOREG2.

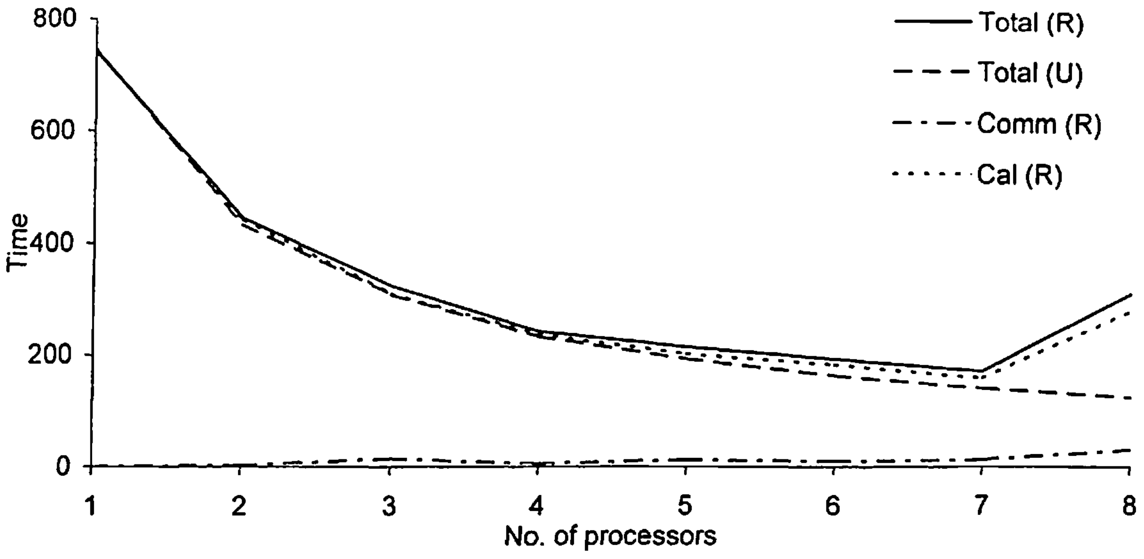


(a) Load compression relationship

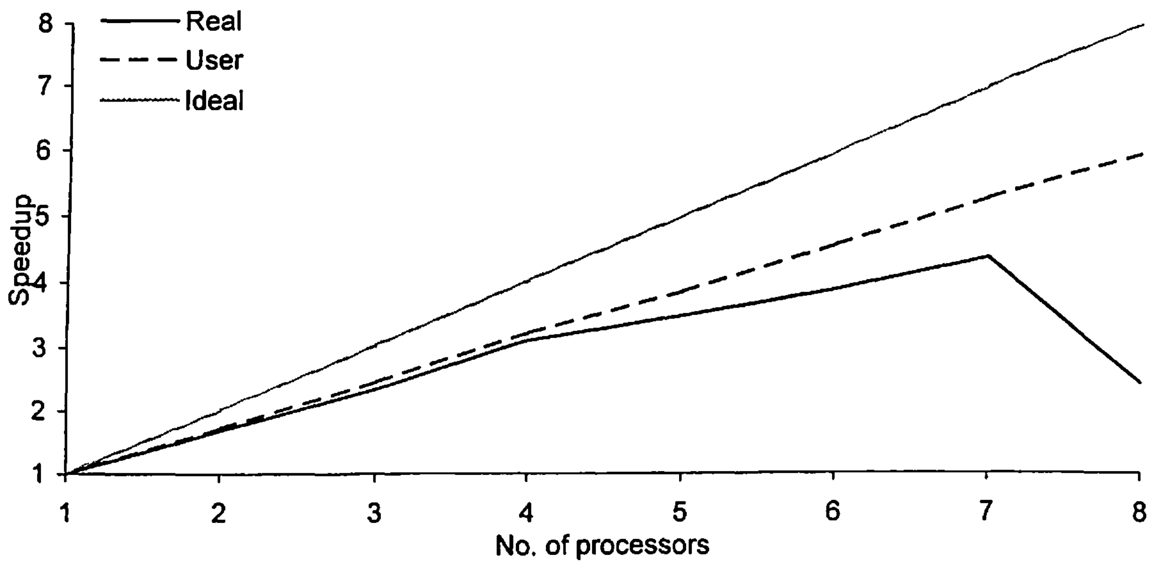


(b) Energy compression relationship

Fig. 6.23 Load compression relationship and energy compression relationship for the circular tube AAC503 under axial compression



(a) Variation in components of computational time



(b) Variation in Speedup

Fig. 6.24 Variation in components of computational time and Speedup with number of processors for specimen tube AAC503

After comparing the obtained technical results of the compression process using the developed code, the described problem is solved on supercomputer PARAM 10000 using multiple processors from one to eight to obtain the variation of computational time. To obtain complete solution of attempted problem, 1387 iterations were required. Figure 6.24 (a) shows the variation in components of computational time per iteration with number of processors. It can be observed that significant amount of Total time was saved by employing multiple processors. It was also found that Total time measured in terms of Real time is minimum at seven processors. Variation in Real time Speedup and User time Speedup is shown in Fig. 6.24 (b). From this figure it can be observed that maximum Real time Speedup of 4.35 and User time Speedup of 5.95 was achieved at seven and eight number of processors respectively. This signifies that developed code is capable of saving computational time considerably.

6.3.4 Lateral Compression of Round Tube Subjected to Concentrated Load

In this case study, analysis of lateral compression of round tube between two concentrated loads is presented. A copper tube having outer diameter 28.5 mm and thickness 1.5 mm is taken for the investigation. The quarter portion of the tube is modeled and shown in the Fig. 6.25. Mesh consists of 540 nodes with 445 rectangular elements resulting in global stiffness matrix of size 1080 × 1080. The material property of the deforming tube is expressed by expression $\bar{\sigma} = k(1 + a\bar{\varepsilon})\dot{\bar{\varepsilon}}^m$, where numerical values of *k*, *a* and *m* were taken as 360 MPa, 0.23 and 0.013 respectively. This tube is compressed between two concentrated loads applied at the top and bottom of the tube (see Fig. 6.25) at rate of 1.0 mm/s till the top surface touches the bottom surface of the tube. Whole compression is carried out in 91 increments. Figure 6.26 shows the deformed shapes of the copper tube at various stages of deformation.

The same tube is also modeled and analyzed using commercial software FORGE2. Figure 6.27 shows the deformed shapes at various stages of deformation. It can be observed from Fig. 6.26 and 6.27 that deformed shapes obtained using developed code and FORGE2 match fairly well. Contours of velocity, effective strain-rate and equivalent strain obtained using developed code at 12.74 mm compression and using FORGE2 at 12.84 mm compression are shown in Fig. 6.28 and 6.29 respectively.

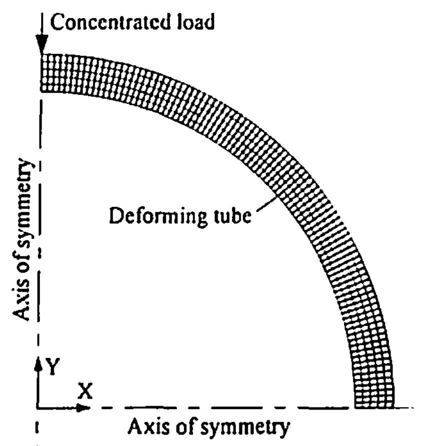
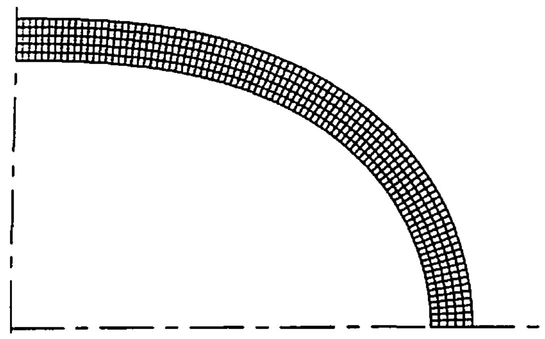
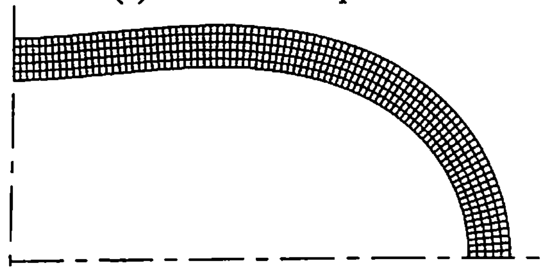


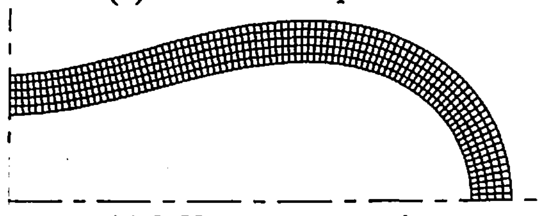
Fig. 6.25 Discretized finite element mesh showing quarter portion of copper tube subjected to concentrated force.



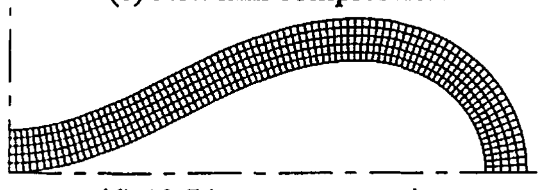
(a) 3.22 mm compression



(b) 6.44 mm compression

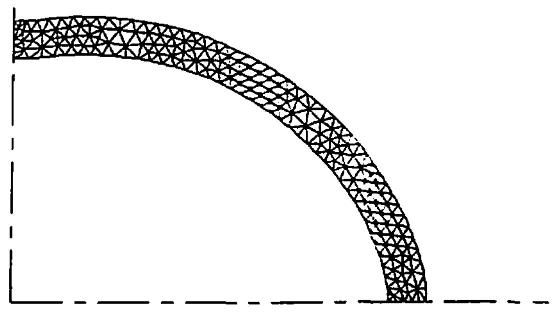


(c) 9.52 mm compression

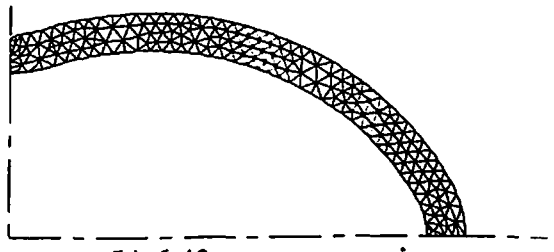


(d) 12.74 mm compression

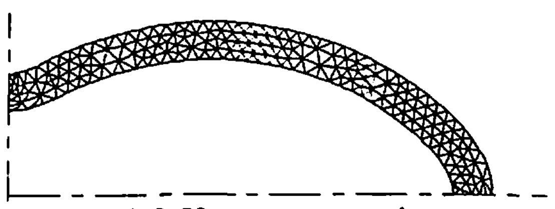
Fig. 6.26 Deformed mesh at various stages of compression obtained by developed code



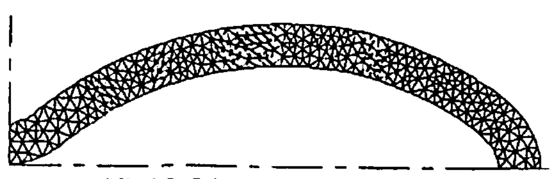
(a) 3.18 mm compression



(b) 6.43 mm compression



(c) 9.53 mm compression



(d) 12.84 mm compression

Fig. 6.27 Deformed mesh at various stages of compression obtained by commercial software FORGE2

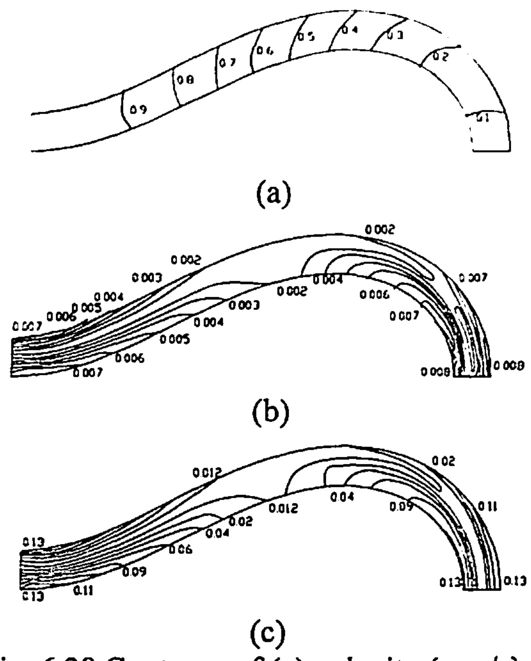


Fig. 6.28 Contours of (a) velocity (mm/s) (b) effective strain rate and (c) equivalent strain at 12.74 mm compression obtained by present investigation.

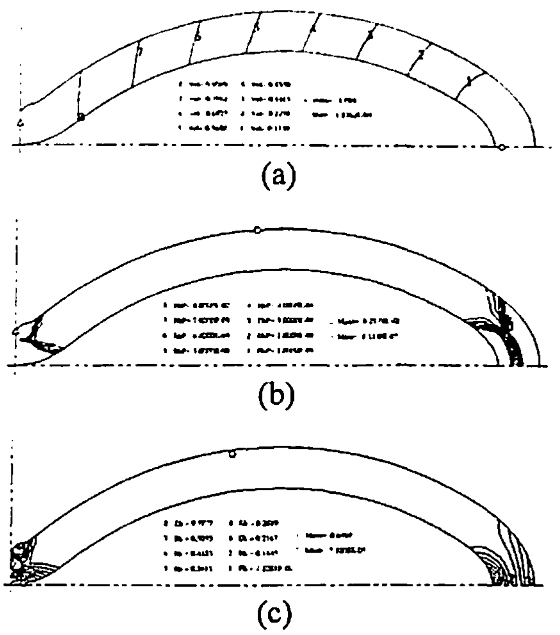
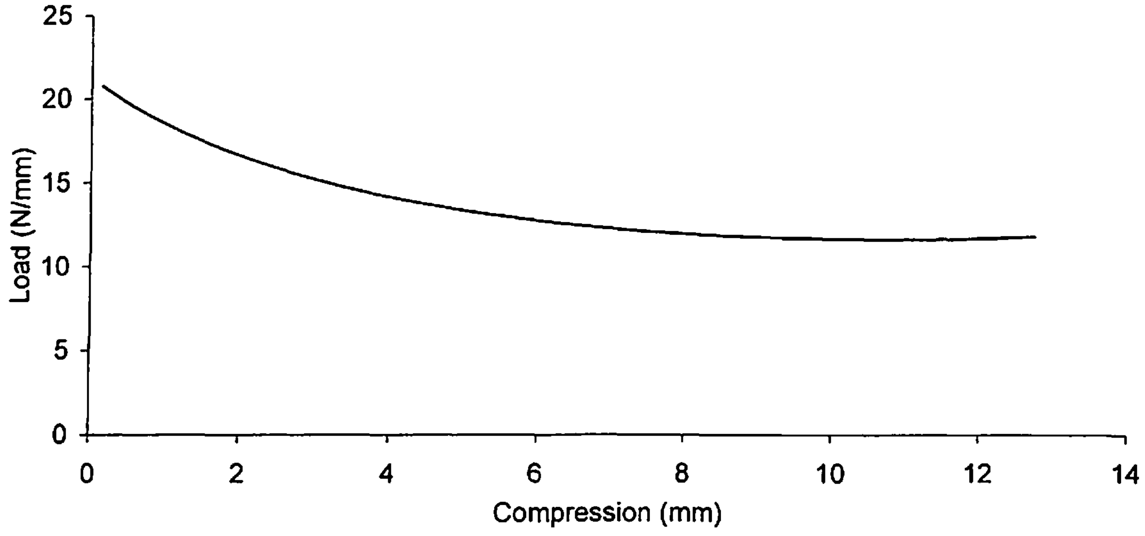


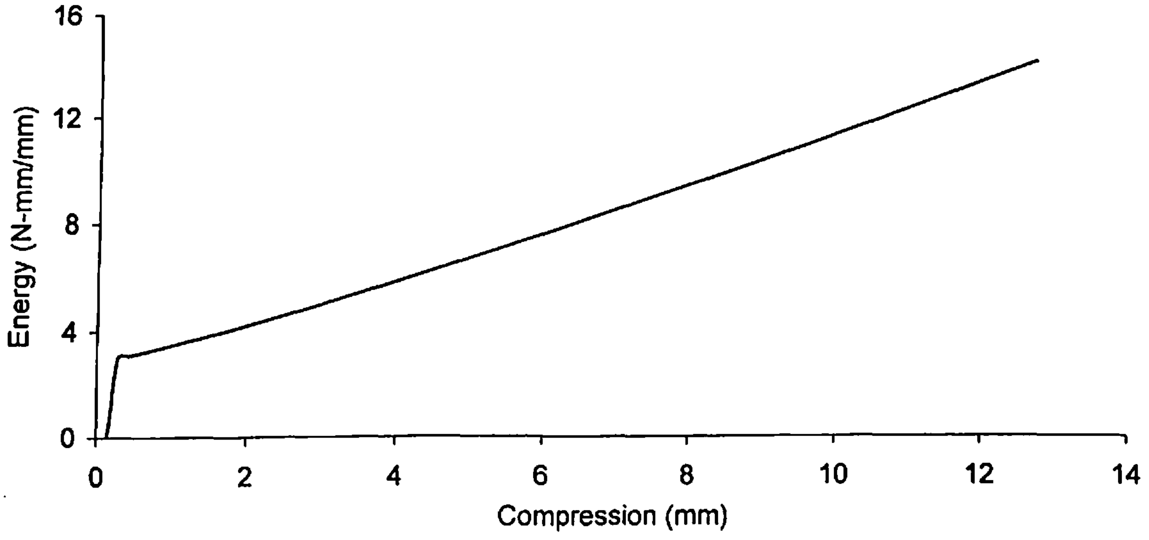
Fig. 6.29 Contours of (a) velocity (mm/s) (b) effective strain rate and (c) equivalent strain at 12.84 mm compression obtained by FOREG2.

It can be seen that contours of velocity obtained by present investigation match well with contours obtained by FORGE2. It can also be observed that the contours of effective strain-rate and equivalent strain do not match well. This may be due to the development of plastic hinges at different locations. Load compression and energy compression relationship for copper tube is illustrated in the Fig. 6.30 (a) and (b) respectively. It can be observed that load reduces with the increase in compression. The absorbed energy continuously increases with increase in compression.

To carry out the complete analysis on supercomputer PARAM 10000, 187 iterations were required. Figure 6.31 (a) shows the computational time variation per iteration with number of processors. It can be observed that reduction in Total time measured in terms of Real time is quite low. This is because of smaller problem size (1080 × 1080). Moreover there is significant difference in Total time measured in terms of Real time and User time, which may be because of large system activities during the analysis. The poor performance can also be seen in Fig. 6.31 (b). Maximum Real time Speedup of 1.43 and maximum User time Speedup of 5.09 was achieved at eight number of processors.

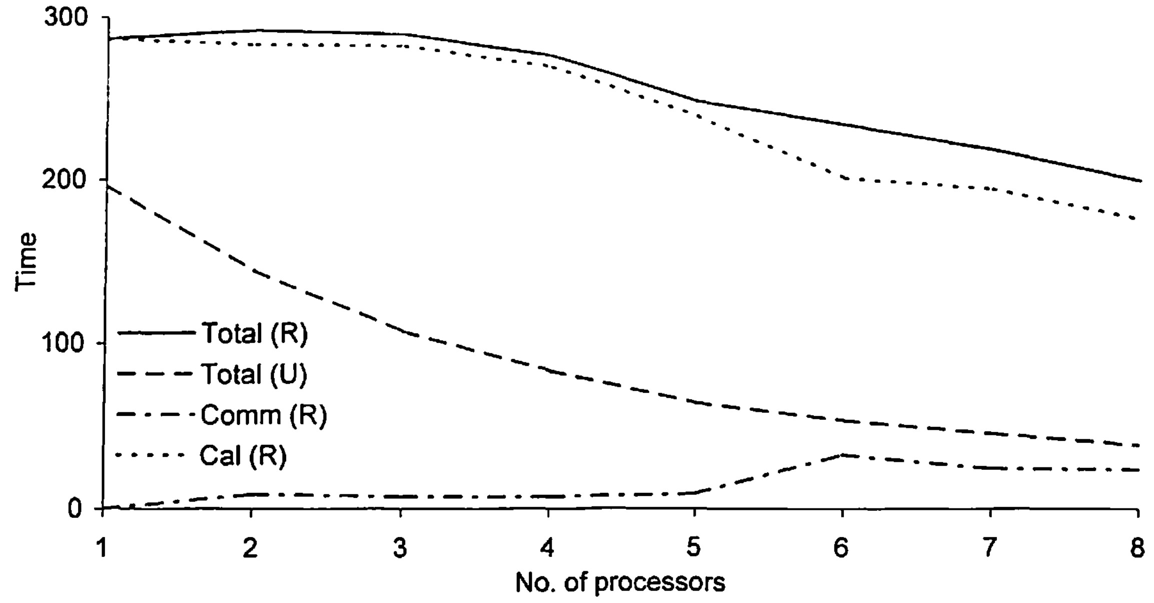


(a) Load compression relationship

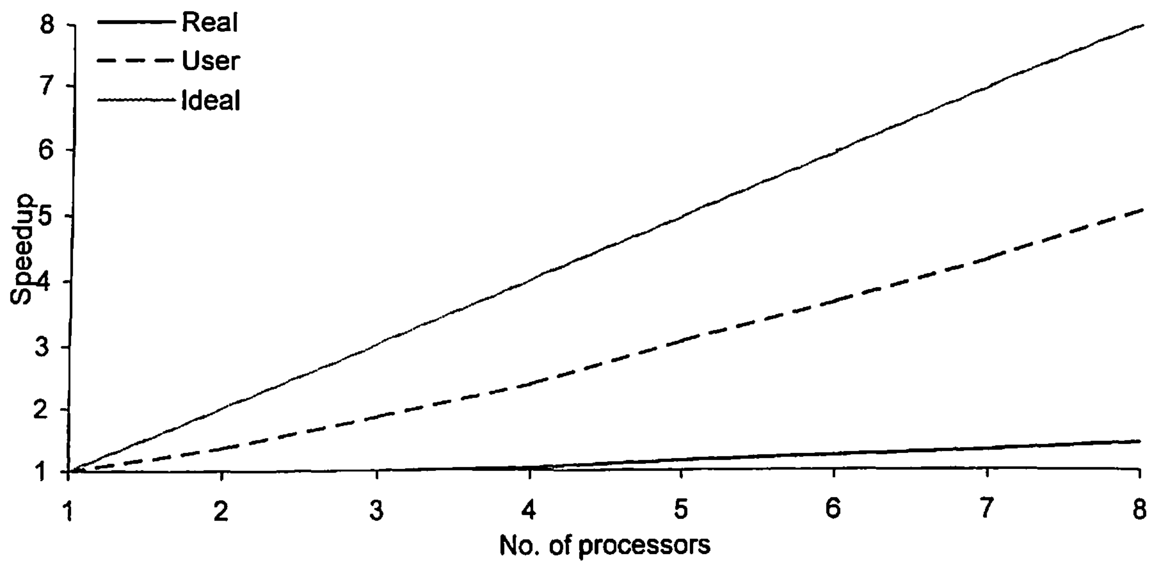


(b) Energy compression relationship

Fig. 6.30 Load compression and energy compression relationship for the circular copper tube subjected to concentrated force



(a) Variation in components of computational time



(b) Variation in Speedup

Fig. 6.31 Variation in components of computational time and Speedup with number of processors for copper tube

6.4 SUMMARY

This chapter initially discusses the drawbacks of the parallelized finite element code FEMLD discussed in chapter 5. Further the chapter discusses the process of simulation of flat die surfaces and its implementation in the developed metal forming code. With the help of modified code, four case study problems are simulated and compared with the findings of FORGE2. The first case study problem was the analysis of axial compression of solid cylinder. Solid cylinder specimen is compressed between flat rigid dies taking different friction factors between deforming cylinder and rigid die surfaces. The compression process was analyzed, studied and discussed. The computational time variation with increasing number of processors of supercomputer PARAM 10000 was obtained and discussed.

In the second case study problem, analysis of flattening of rectangular metallic tube between flat dies is presented. Compression processes of six mild steel tubes with varying geometric properties are simulated with the help of developed code. To validate the results obtained from developed code, sample tube specimen AS26263 was also analyzed using commercial software FORGE2. Various results obtained from present investigation are compared with the corresponding results obtained from FORGE2. After comparison, it was found that the results of present investigation match fairly well with the results of commercial software FORGE2. The analysis was carried out on supercomputer PARAM 10000 and it was found that computational time could be saved by using multiple number of processors.

A problem of fold formation in axial compression of circular tube was considered as third case study problem. Axial compression of aluminium tube was simulated using developed software as well as commercial software FORGE2. Various results obtained by both the computer codes are compared and it was found that the results obtained by developed software match well with the results of FORGE2. The performance of the developed code was also measured and it was found the developed code is capable of saving significant amount of computational time by employing multiple number of processors.

In the last case study problem lateral compression of copper round tube between two concentrated loads was carried out. The tube was compressed laterally till the top and bottom surface of the tube touched each other. This simulation was carried out using developed software as well as using FORGE2. Various results obtained were compared and it was found that results match fairly well. Insignificant saving in computational time was achieved by using multiple number of processors of supercomputer PARAM 10000 due to small size problem.

CHAPTER 7

APPLICATION OF CLUSTER COMPUTING IN FINTIE ELEMENT ANALYSIS

7.1 INTRODUCTION

This chapter presents an alternative to supercomputers by presenting two Windows NT Clusters consisting of eight conventional computers having similar and different configurations. The chapter starts with highlighting the need of Cluster and followed by description of compilers used for development of parallel programs on Windows NT platform. Chapter then covers a case study in which three different data sets of system of linear equations were analyzed using developed Windows NT Cluster (consisting of eight conventional computers having similar configuration) as well as supercomputer PARAM 10000. Comparison of both computing systems is presented through the obtained computational time results.

Another Windows NT Cluster is also developed that consists of conventional computer of different configurations. On this developed Cluster, four parallel solvers of Gauss-Seidel Method, Gauss Elimination Method, Matrix Inversion Method and Modified Matrix Inversion Method are developed. Using these parallel solvers, five distinct data sets were analyzed by increasing number of computers from one to eight and different components of computational time are measured. Based on the obtained variations in these components of computational time, comparison between these parallel solvers is carried out. It was found that Modified Matrix Inversion Method parallel solver performed better as compared to other parallel solvers. Therefore this parallel solver is incorporated in two finite element codes, which are capable of solving linear and non-linear two-dimensional problems. A problem of anchorage zone in prestressed post-tensioned concrete beam is considered for linear elastic finite element analysis on the other hand a problem of simple compression of solid cylinder is considered for non-linear finite element analysis. These problems were discretized using different mesh sizes and analyzed by increasing number of computer from one to eight. Computational time variation is obtained and presented. It was found that Total time in finite element analysis could be effectively reduced on Windows NT Cluster by employing multiple computers.

7.2 NEED FOR CLUSTER

It is seen that supercomputers could be used effectively to solve complex problems. The computing power of the supercomputers is extremely high as compared to the conventional computers. Supercomputers mainly use multiple processors along with parallel computing technique. But the supercomputers are very costly as compared to a conventional computer. Therefore it is tried to develop an inexpensive alternative of the supercomputers by connecting conventional computers with local area network (LAN). This group of conventional computers is called as Cluster.

7.2.1 Cluster Setup

In the present study, a Cluster consisting eight *IBM Personal Computer 300GL* is formed. Each computer has single Intel P II processor operating at 400 MHz at 512 KB Cache. Each computer has 128 MB RAM with 4.2 GB HDD. All computers have TCP/IP socket connections. These computers are connected to each other by Local Area Network (LAN) through a Switch of 100 MBPS capacity. Figure 7.1 illustrates described Cluster. All computers are equipped with Windows NT 4.0 Workstation operating system.

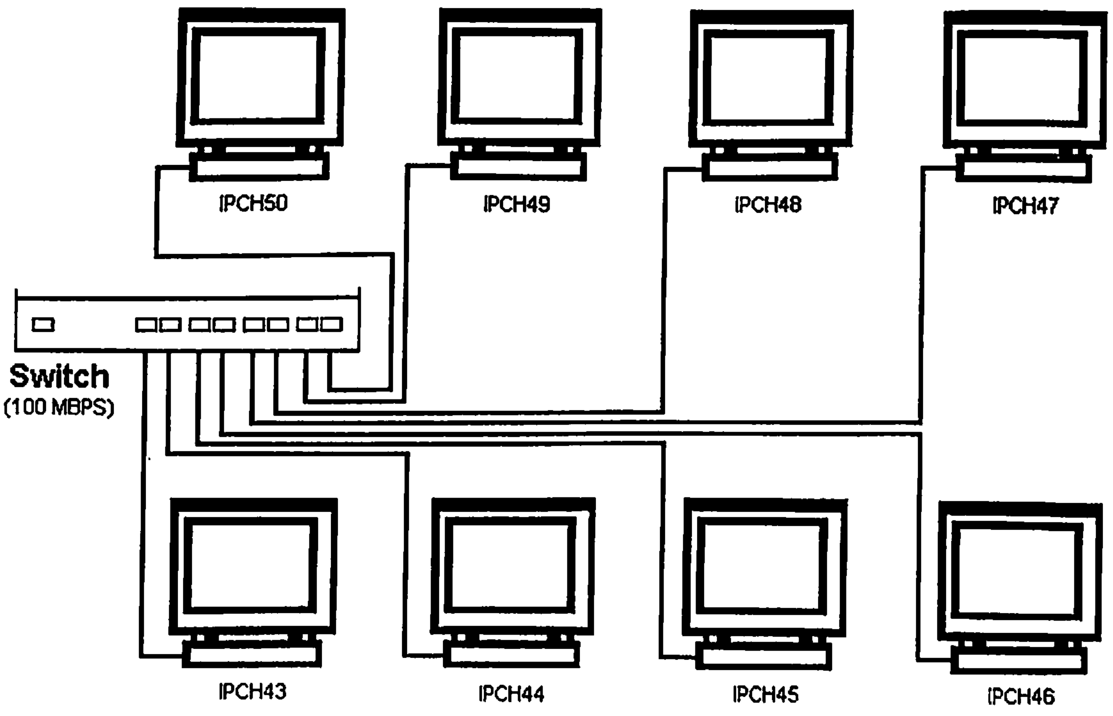


Fig. 7.1 Windows NT cluster of computers having similar configuration

7.2.2 Compilers

MPICH for Microsoft Windows compiler [4] developed by Argonne National Laboratory is installed on every computer of the Cluster. Each computer is then configured to execute parallelized computer codes on Cluster. In addition to this, Microsoft Visual C++ 6.0 compiler is also installed to create executable files.

7.3 CASE STUDY

To test the functioning of developed Cluster, few standard benchmark programs [8] are developed in Visual C++ environment and executed on the developed Cluster to ensure the parallel operation of the Cluster. Matrix Inversion Method parallel solver (presented in section 3.3.3) was written for the developed Cluster. Three data sets of size 840 × 840, 1226 × 1226 and 1722 ×1722 are analyzed on developed Cluster. These data sets are analyzed by changing number of computers from one to eight and the different components of computational time are measured. Table 7.1 shows the computational time results obtained for these data sets under consideration. These components of computational time are measured in terms of Real time only as User time cannot be measured in Windows NT operating system.

It can be observed from Table 7.1 that the Total time and Calculation time reduces while Communication time increases with increase in number of computers. In general, it can be seen that computational time variation is nearly identical for all data sets. It can also be observed from Table 7.1 that Speedup increases with increase in number of computers as well as with increase in data size. Maximum Speedup of 3.74 can be seen at eight computers for data size of 1722 × 1722. Qualitative variation of MFLOPS is similar to the variation of Speedup. Maximum MFLOPS of 33.61 was obtained at eight computers for data size of 1722 × 1722. Decrease in Efficiency with increase in number of computers can be observed.

Table 7.1 Computational time variation and performance of MIM solver for data set of different sizes on Windows NT cluster

(a) Results for data set of size 840 × 840

No. of computers	Total	Comm	Cal	Speedup	Efficiency	MFLOPS
1	120.00	0.00	120.00	1.00	100.00	9.90
2	78.68	6.49	72.19	1.53	76.25	15.10
3	66.97	13.15	53.82	1.79	59.73	17.74
4	58.77	15.81	42.97	2.04	51.04	20.22
5	64.62	24.22	40.40	1.86	37.14	18.39
6	60.53	24.93	35.60	1.98	33.04	19.63
7	60.54	27.32	33.23	1.98	28.31	19.63
8	58.37	23.91	34.46	2.06	25.70	20.36

(b) Results for data set of size 1226 × 1226

No. of computers	Total	Comm	Cal	Speedup	Efficiency	MFLOPS
1	363.02	0.00	363.02	1.00	100.00	10.17
2	214.81	12.60	202.21	1.69	84.50	17.19
3	204.10	28.38	175.72	1.78	59.29	18.09
4	175.70	33.89	141.81	2.07	51.65	21.01
5	156.21	49.75	106.46	2.32	46.48	23.63
6	145.19	51.60	93.59	2.50	41.67	25.43
7	143.32	56.86	86.45	2.53	36.19	25.76
8	140.56	52.88	87.67	2.58	32.28	26.26

(c) Results for data set of size 1722 × 1722

No. of computers	Total	Comm	Cal	Speedup	Efficiency	MFLOPS
1	1138.00	0.00	1138.00	1.00	100.00	8.98
2	577.82	23.72	554.10	1.97	98.47	17.69
3	533.50	53.73	479.77	2.13	71.10	19.16
4	411.99	63.29	348.70	2.76	69.06	24.82
5	444.11	93.61	350.51	2.56	51.25	23.02
6	398.56	100.53	298.03	2.86	47.59	25.65
7	359.14	111.99	247.15	3.17	45.27	28.47
8	304.20	100.73	203.48	3.74	46.76	33.61

Table 7.2 Computational time variation and performance of MIM solver for data set of different sizes on PARAM 10000

(a) Results for data set of size 840 × 840

No. of processors	Total	Comm	Cal	Speedup	Efficiency	MFLOPS
1	186.31	0.00	186.31	1.00	100.00	6.38
2	99.87	0.67	99.20	1.87	93.27	11.90
3	72.09	1.54	70.55	2.58	86.15	16.48
4	56.46	1.71	54.75	3.30	82.49	21.04
5	55.34	5.23	50.11	3.37	67.33	21.47
6	56.03	6.59	49.44	3.33	55.42	21.21
7	49.29	6.30	42.98	3.78	54.00	24.11
8	56.35	2.26	54.09	3.31	41.33	21.09

(b) Results for data set of size 1226 × 1226

No. of processors	Total	Comm	Cal	Speedup	Efficiency	MFLOPS
1	578.16	0.00	578.16	1.00	100.00	6.38
2	299.72	1.50	298.22	1.93	96.45	12.32
3	208.53	3.61	204.92	2.77	92.42	17.70
4	163.36	4.18	159.18	3.54	88.48	22.60
5	155.09	12.96	142.13	3.73	74.56	23.80
6	141.85	15.61	126.23	4.08	67.93	26.02
7	125.84	14.62	111.22	4.59	65.63	29.33
8	134.44	16.15	118.29	4.30	53.76	27.46

(c) Results for data set of size 1722 × 1722

No. of processors	Total	Comm	Cal	Speedup	Efficiency	MFLOPS
1	1625.43	0.00	1625.43	1.00	100.00	6.29
2	851.31	3.05	848.26	1.91	95.47	12.01
3	571.71	9.23	562.47	2.84	94.77	17.88
4	446.75	7.75	439.00	3.64	90.96	22.89
5	405.12	18.19	386.93	4.01	80.24	25.24
6	356.91	29.88	327.03	4.55	75.90	28.65
7	321.85	28.47	293.38	5.05	72.15	31.77
8	457.26	50.36	406.90	3.55	44.43	22.36

The same data sets were also analyzed on supercomputer PARAM 10000 and components of computational time were also measured. Table 7.2 shows the variation in computational time components for three data sets for different number of processors of supercomputer PARAM 10000. It can be seen that Total time and Calculation time reduces whereas Communication time increases with increase in number of processors. The Speedup as well as MFLOPS increases with increase in number of processors. Maximum Speedup and MFLOPS obtained are 5.05 and 31.77 respectively for data of size 1722 × 1722 at seven number of processors.

From Tables 7.1 and 7.2 it can be observed that a single processor of supercomputer PARAM 10000 requires more time than a conventional personal computer (see Total time of single processor of PARAM 10000 and Total time of single computer of Cluster for all three data sets). The difference in Total time required by both computing systems at every number of processors/computers is marginal. It can be observed that reduction in Total time is very rapid in supercomputer PARAM 10000 as compared to the Cluster. This difference can also be seen in Speedup. This is mainly because of higher contribution of Communication time in Total time for Cluster.

Table 7.3 shows percentage contribution of Communication time in Total time for PARAM 10000 and Cluster for all data sets under consideration. It is very clear that contribution of Communication time is very high in Cluster as compared to supercomputer PARAM 10000. This contribution reduces with increase in data size. This signifies that performance of solver improves with increase in size of data on Cluster.

7.4 MIXED PC CLUSTER

In the above section it is seen that with the help of LAN connected computers, parallel programs can be executed. Most of the times LAN connected computers may not have similar configurations. Therefore a Cluster consisting of eight computers with different configurations is created. Eight computers of *IBM Personal Computer 300GL* with different processor speed are used. This Cluster contains four computers each having single Intel P II processor at 300 MHz, three computers each having single Intel P II processor at 400 MHz, and one computer with single processors of Intel P II at 500 MHz.

Table 7.3 Percentage contribution of Communication time towards Total time for different data set on Windows NT cluster and PARAM 10000.

No. of processors/ computers	Windows NT cluster			PARAM 10000		
	840	1226	1722	840	1226	1722
1	0.00	0.00	0.00	0.00	0.00	0.00
2	8.25	5.86	4.10	0.67	0.50	0.36
3	19.64	13.91	10.07	2.13	1.73	1.62
4	26.90	19.29	15.36	3.03	2.56	1.73
5	37.48	31.85	21.08	9.45	8.36	4.49
6	41.18	35.54	25.22	11.75	11.01	8.37
7	45.12	39.68	31.18	12.79	11.62	8.85
8	40.97	37.63	33.11	4.01	12.01	11.01

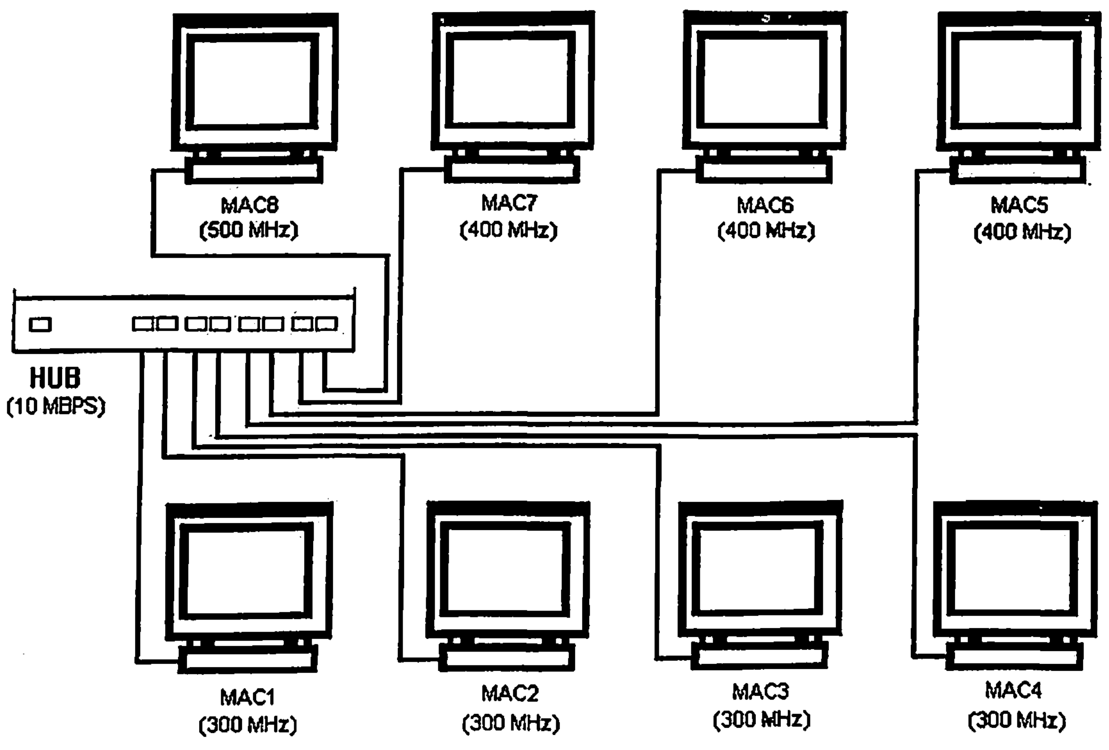


Fig. 7.2 Windows NT cluster of computers with different configurations

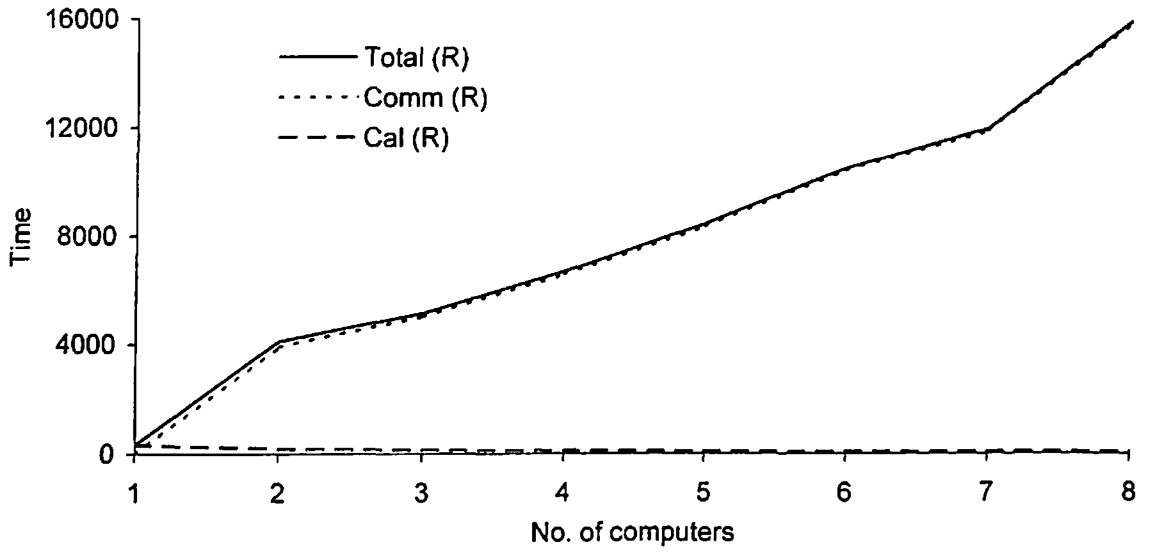
The RAM size of these computers is also different. All computers with 300 MHz processors have RAM of size 64 MB and remaining all computers has RAM of size 128 MB. These computers are then connected through a HUB of 10 MBPS capacity. Windows NT 4.0 workstation operating system was installed on all computers. The developed Cluster is illustrated in Fig. 7.2. MPICH for Microsoft Windows compiler [4] was also installed on every computer of the Cluster. Each computer was then properly configured to execute parallel programs on this developed Cluster.

7.5 PERFORMANCE OF PARALLEL SOLVERS ON CLUSTER

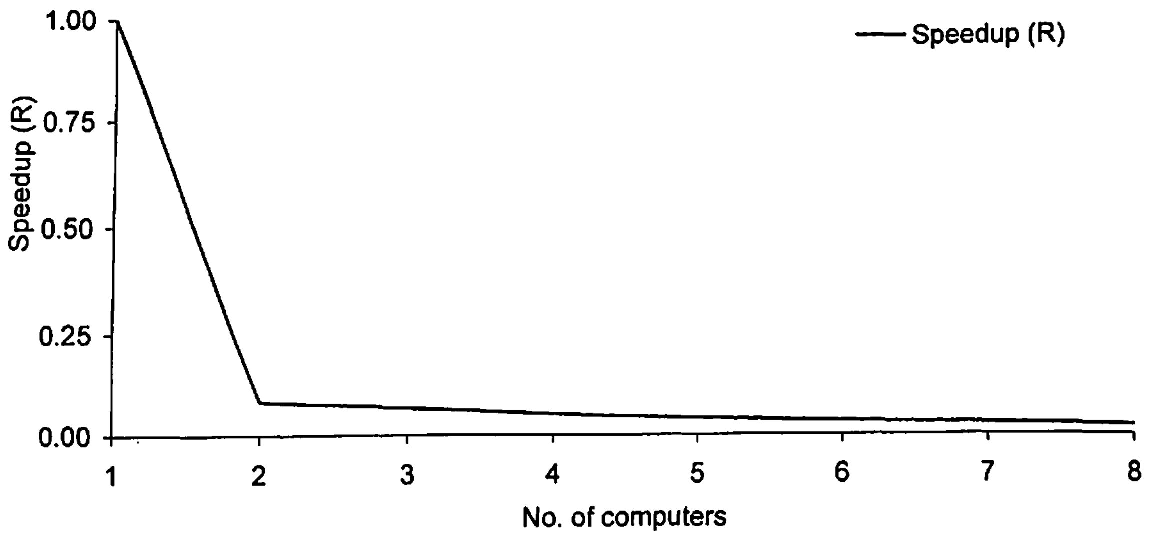
Based on the algorithms discussed in Chapter 3, parallel solvers are developed on Windows NT platform and implemented on developed Cluster discussed above. All four parallel solvers (GSM, GEM, MIM, and MMIM) are used to analyze five different data sets of increasing size (870 × 870, 1226 × 1226, 1352 × 1352, 1722 × 1722 and 2312 × 2312) taken from linear elastic and non-linear plastic finite element analysis problems discussed in Chapter 4 and Chapter 5. Because of RAM limitations (RAM size ranging from 64 MB to 128 MB) solution to data sets having size more that 2500 × 2500 (approximately) was not possible. The computers used in developed Cluster were having different processor speeds which are 300 MHz, 400 MHz and 500 MHz, therefore to maintain the consistency in computational time results, at least one computer with 300 MHz speed is involved in each execution process. Computational time components (in terms of Real time) were measured for all data sets, which were analyzed by four different parallel solvers with increasing number of computers from one to eight.

7.5.1 Gauss-Seidel Method

Figures 7.3 (a) and (b) show typical variation of computational time and Speedup with increasing number of computers for GSM parallel solver for data set of size 1226 × 1226. It can be seen from Fig. 7.3 (a) that Total time increases with increase in number of computers. Communication time variation is exactly similar to the Total time variation. Communication time is marginally lesser than the Total time at every number of computers. Reduction in Calculation time can also be seen with increase in number of computers. Similar type of variation can also be seen in Fig. 3.4 (a) in which a data set



(a) Variation in computational time components



(b) Variation in Speedup (R)

Fig. 7.3 Variation in computational time components and Speedup (R) for GSM solver for data set of size 1226 × 1226

Table 7.4 Computational time variation and performance of GSM solver for data set of different sizes on Windows NT cluster

(a) Results for data set of size 870 × 870

No. of computers	Total	Comm	Cal	Speedup	Efficiency	MFLOPS
1	107.36	0.00	107.36	1.00	100.00	22.91
2	1752.82	1693.54	59.28	0.06	3.06	1.40
3	2177.19	2128.98	48.21	0.05	1.64	1.13
4	2875.72	2834.79	40.93	0.04	0.93	0.86
5	3623.41	3590.52	32.89	0.03	0.59	0.68
6	4517.57	4482.18	35.40	0.02	0.40	0.54
7	5302.78	5272.56	30.22	0.02	0.29	0.46
8	6998.00	6969.79	28.21	0.02	0.19	0.35

(b) Results for data set of size 1226 × 1226

No. of computers	Total	Comm	Cal	Speedup	Efficiency	MFLOPS
1	332.88	0.00	332.88	1.00	100.00	24.25
2	4106.79	3916.92	189.88	0.08	4.05	1.97
3	5118.39	4988.18	130.22	0.07	2.17	1.58
4	6684.69	6569.92	114.78	0.05	1.24	1.21
5	8435.47	8339.23	96.24	0.04	0.79	0.96
6	10484.10	10401.58	82.52	0.03	0.53	0.77
7	11976.12	11897.44	78.68	0.03	0.40	0.67
8	15938.31	15866.78	71.54	0.02	0.26	0.51

(c) Results for data set of size 1352 × 1352

No. of computers	Total	Comm	Cal	Speedup	Efficiency	MFLOPS
1	416.25	0.00	416.25	1.00	100.00	23.75
2	4544.45	4324.90	219.55	0.09	4.58	2.18
3	5661.36	5490.90	170.46	0.07	2.45	1.75
4	7356.33	7227.09	129.24	0.06	1.41	1.34
5	9394.76	9277.95	116.80	0.04	0.89	1.05
6	11136.05	11032.19	103.86	0.04	0.62	0.89
7	13714.61	13619.31	95.29	0.03	0.43	0.72
8	17959.20	17870.79	88.42	0.02	0.29	0.55

(d) Results for data set of size 1722 × 1722

No. of computers	Total	Comm	Cal	Speedup	Efficiency	MFLOPS
1	808.37	0.00	808.37	1.00	100.00	24.63
2	7030.85	6572.49	458.36	0.11	5.75	2.83
3	9063.42	8727.19	336.24	0.09	2.97	2.20
4	11682.02	11410.10	271.92	0.07	1.73	1.70
5	14914.73	14697.55	217.18	0.05	1.08	1.33
6	18465.68	18268.22	197.46	0.04	0.73	1.08
7	18909.75	18730.78	178.97	0.04	0.61	1.05
8	31936.24	31773.62	162.62	0.03	0.32	0.62

(e) Results for data set of size 2312 × 2312

No. of computers	Total	Comm	Cal	Speedup	Efficiency	MFLOPS
1	1963.93	0.00	1963.93	1.00	100.00	25.12
2	12872.67	11787.66	1085.01	0.15	7.63	3.83
3	16707.09	15775.99	931.10	0.12	3.92	2.95
4	21489.25	20881.68	607.57	0.09	2.28	2.30
5	27480.97	26938.27	542.70	0.07	1.43	1.79
6	33974.00	33495.68	478.32	0.06	0.96	1.45
7	39928.97	39516.89	412.08	0.05	0.70	1.24
8	52352.14	51972.14	380.00	0.04	0.47	0.94

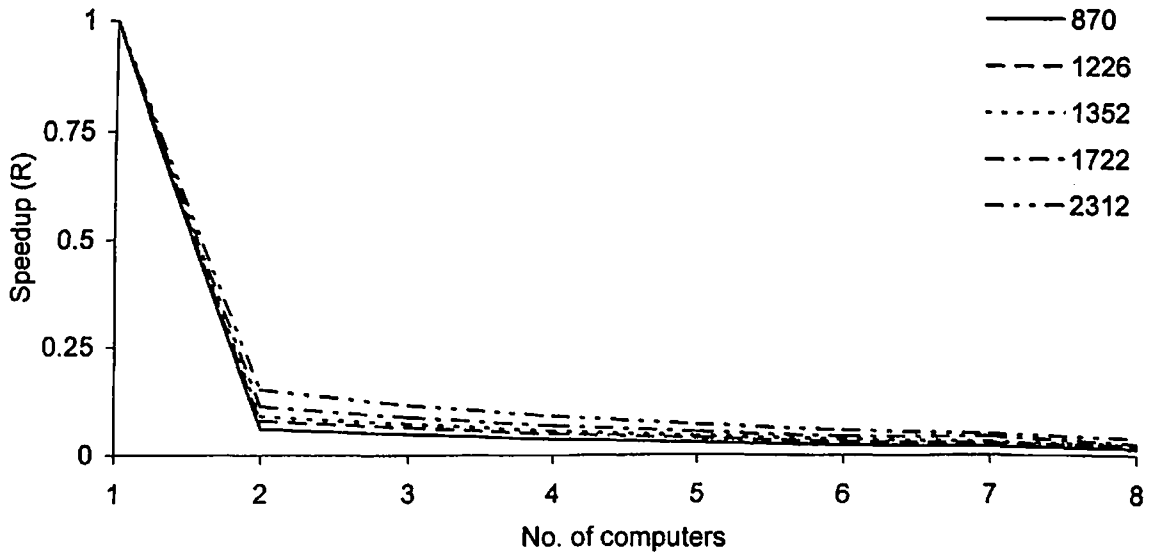


Fig. 7.4 Variation in Speedup (R) for GSM solver for various data sets on Windows NT cluster

1226 × 1226 was analyzed using GSM solver on supercomputer PARAM 10000. From Fig. 7.3 (b) it can be observed that Speedup reduces drastically from one to two number of computers. With further increase in number of computers, Speedup continuously decreases but with reduced rate. Continuous reduction in Speedup was also observed when PARAM 10000 was used (see Fig. 3.5).

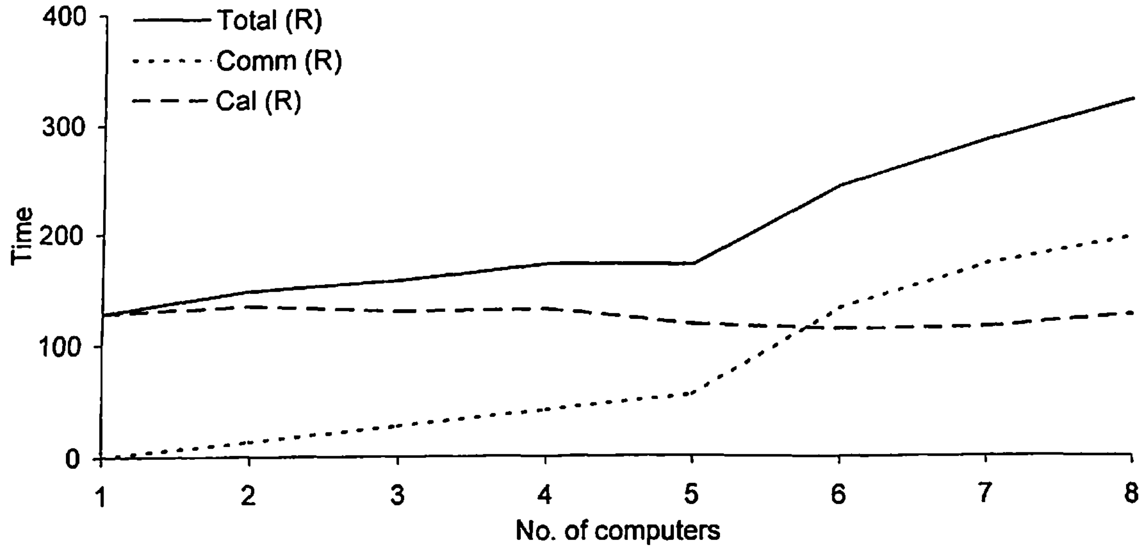
When other data sets of increasing sizes were analyzed, it was found that Total time increases with increase in number of computers employed for performing the analysis (see Table 7.4). Figure 7.4 shows the variation in Speedup with number of computers for all data sets under consideration. It can be seen that as data size increases, the corresponding performance of GSM parallel solver also improves. This improvement is insignificant as compared to the improvement observed in case of PARAM 10000 (see Fig. 3.6). Table 7.5 shows the number of iterations carried out in analyzing various data sets. It can be seen that number of iterations required to solve a particular data sets are independent of data size (see Table 3.8 in Chapter 3).

7.5.2 Gauss Elimination Method

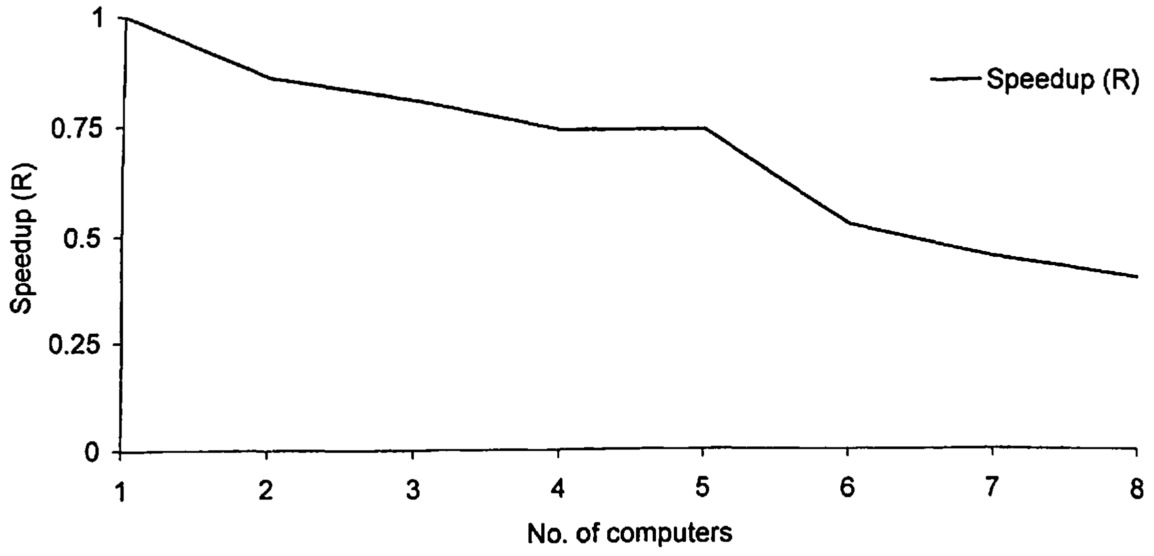
Variation of different components of computational and Speedup with increasing number of computers for data set of size 1226 × 1226 obtained using GEM parallel solver is shown in Fig. 7.5. It can be seen from Fig. 7.5 (a) that Total time increases with increase in number of computers. The increase is gradual from one to five number of computers. After five number of computers sudden increase in Total time can be seen till eight number of computers. The variation of Communication time is exactly similar to variation of Total time with significantly lesser magnitude. It can also be observed from this figure that Calculation time decreases with increase in number of computers. This reduction in Calculation time is very less with increasing number of computers. The Computational time variation obtained on Windows NT Clusters is quite similar to the variation obtained on supercomputer PARAM 10000 (see Fig. 3.9(a)). In case of Windows NT Cluster, Calculation time continuously decreases with increase in number of computers whereas in case of PARAM 10000, it increases with increase in number of processors. This is mainly because of higher contribution of Communication time for Windows NT Cluster as compared to PARAM 10000. Figure 7.5 (b) shows variation in

Table 7.5 Number of iteration for various data sizes carried out by GSM solver

Data size	No. of iterations	Status
870 × 870	1083	Solution found
1226 × 1226	1790	Solution found
1352 × 1352	1803	Solution found
1722 × 1722	2238	Solution found
2312 × 2312	3076	Incomplete



(a) Variation in computational time components



(b) Variation in Speedup (R)

Fig. 7.5 Variation in computational time components and Speedup (R) for GEM solver for data set of size 1226 × 1226

Table 7.6 Computational time variation and performance of GEM solver for data set of different sizes on Windows NT cluster

(a) Results for data set of size 870 × 870

No. of computers	Total	Comm	Cal	Speedup	Efficiency	MFLOPS
1	46.26	0.00	46.26	1.00	100.00	7.13
2	54.81	6.84	47.97	0.84	42.20	6.01
3	62.19	13.38	48.82	0.74	24.79	5.30
4	70.70	20.79	49.92	0.65	16.36	4.66
5	69.72	28.22	41.50	0.66	13.27	4.73
6	104.90	63.08	41.82	0.44	7.35	3.14
7	122.40	80.16	42.24	0.38	5.40	2.69
8	143.92	92.26	51.66	0.32	4.02	2.29

(b) Results for data set of size 1226 × 1226

No. of computers	Total	Comm	Cal	Speedup	Efficiency	MFLOPS
1	127.37	0.00	127.37	1.00	100.00	7.24
2	148.12	13.51	134.61	0.86	42.99	6.23
3	158.10	27.89	130.21	0.81	26.85	5.83
4	172.48	41.36	131.11	0.74	18.46	5.35
5	171.55	54.78	116.78	0.74	14.85	5.38
6	244.35	131.52	112.83	0.52	8.69	3.77
7	286.74	172.46	114.28	0.44	6.35	3.22
8	323.94	197.93	126.02	0.39	4.91	2.85

(c) Results for data set of size 1352 × 1352

No. of computers	Total	Comm	Cal	Speedup	Efficiency	MFLOPS
1	139.89	0.00	139.89	1.00	100.00	8.84
2	190.38	17.68	172.70	0.73	36.74	6.50
3	204.82	29.99	174.82	0.68	22.77	6.04
4	224.99	48.72	176.27	0.62	15.54	5.50
5	215.16	64.86	150.30	0.65	13.00	5.75
6	308.46	156.03	152.42	0.45	7.56	4.01
7	361.36	207.11	154.25	0.39	5.53	3.42
8	387.66	225.90	161.77	0.36	4.51	3.19

(d) Results for data set of size 1722 × 1722

No. of computers	Total	Comm	Cal	Speedup	Efficiency	MFLOPS
1	373.78	0.00	373.78	1.00	100.00	6.83
2	389.12	25.92	363.19	0.96	48.03	6.57
3	418.79	52.00	366.79	0.89	29.75	6.10
4	453.74	85.15	368.58	0.82	20.59	5.63
5	419.79	105.89	313.90	0.89	17.81	6.09
6	557.72	241.68	316.04	0.67	11.17	4.58
7	633.85	325.26	308.59	0.59	8.42	4.03
8	694.11	368.74	325.37	0.54	6.73	3.68

(e) Results for data set of size 2312 × 2312

No. of computers	Total	Comm	Cal	Speedup	Efficiency	MFLOPS
1	953.15	0.00	953.15	1.00	100.00	6.49
2	1002.25	34.10	968.15	0.95	47.55	6.17
3	1053.65	76.38	977.26	0.90	30.15	5.87
4	1100.31	118.99	981.32	0.87	21.66	5.62
5	999.55	144.67	854.87	0.95	19.07	6.18
6	1220.82	392.49	828.32	0.78	13.01	5.06
7	1318.34	478.78	839.56	0.72	10.33	4.69
8	1361.61	532.44	829.17	0.70	8.75	4.54

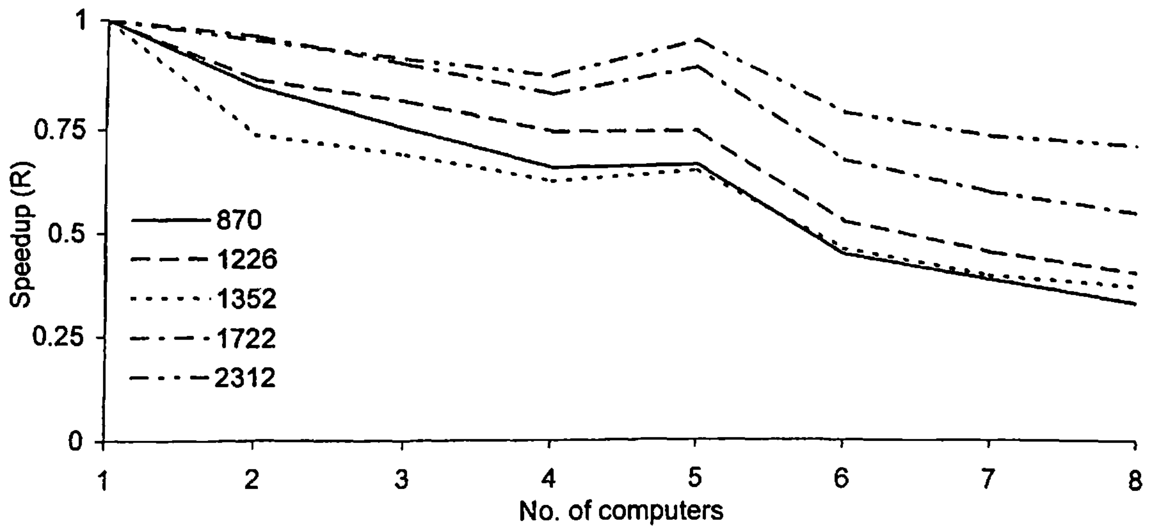


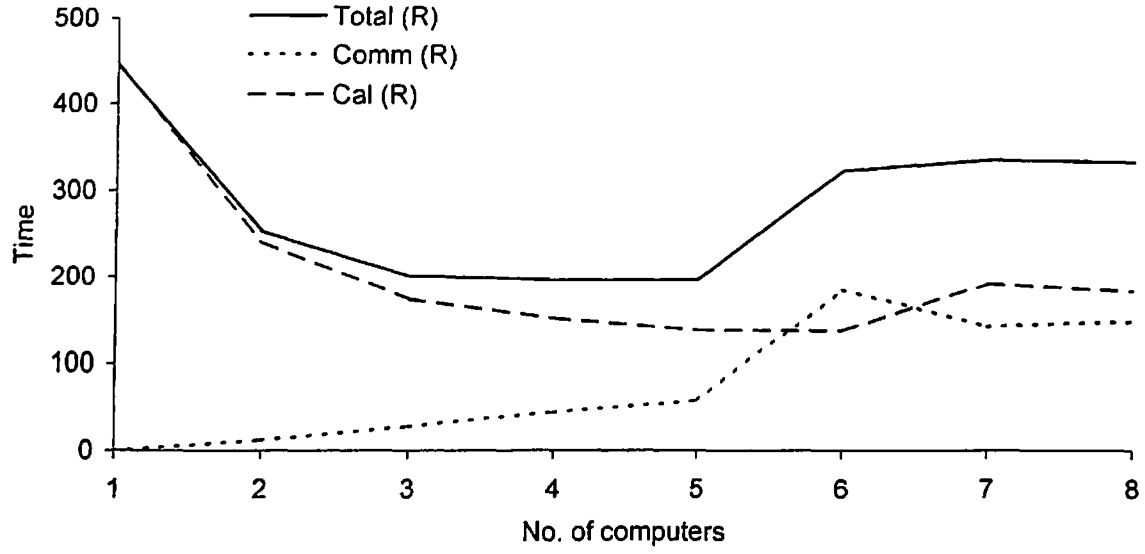
Fig. 7.6 Variation in Speedup (R) for GEM solver for various data sets on Windows NT cluster

Speedup with increasing number of computers. It can be seen that Speedup decreases with increase in number of computers. When this variation was compared with Speedup obtained on supercomputer PARAM 10000 (see Fig. 3.10), it was observed that both variations are nearly identical.

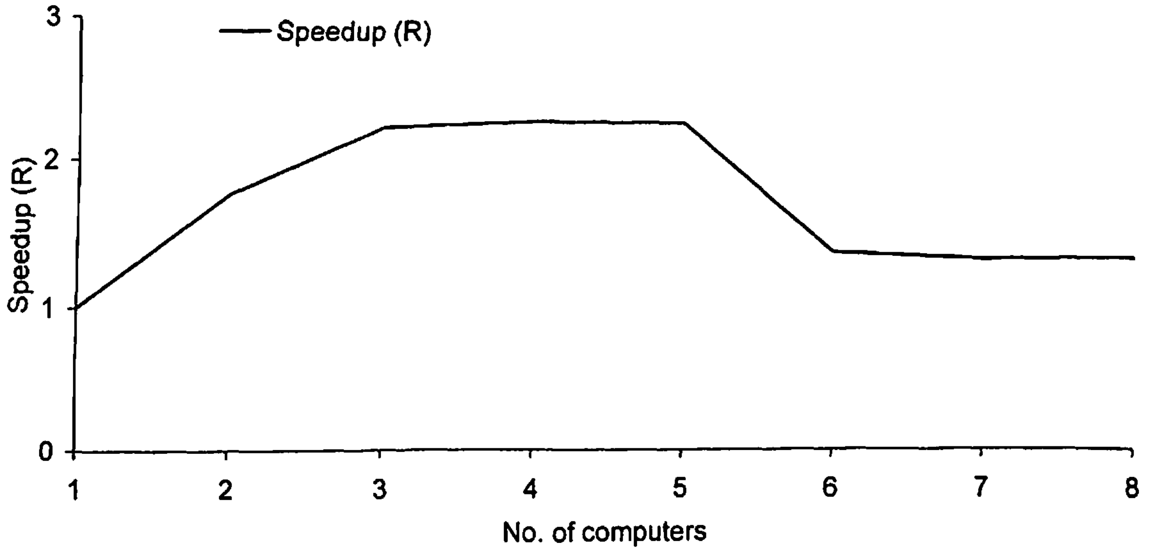
Table 7.6 shows the computational time variation and performance of GEM parallel solver for increasing size of data sets under consideration. It can be observed that variation in computational time components is very much identical for all data sets. In all the cases Total time increases with increase in number of computers. Figure 7.6 shows the variation in Speedup with increasing number of computers for all data sets under consideration. It can be seen from this figure that the performance of the GEM parallel solver improves with increase in data size. Still the performance of GEM parallel solver is poor as the Speedup remains below 1.0 even after employing higher number of computers. After comparing the performance of GEM parallel solver on Cluster and supercomputer PARAM 10000 (see Fig. 3.10 and Fig. 7.5(b)), it was found that this solver performed fairly well on supercomputer PARAM 10000.

7.5.3 Matrix Inversion Method

Computational time variation obtained by MIM parallel solver for data set of size 1226 × 1226 is shown in Fig. 7.7 (a). It can be seen from this figure that Total time reduces with increase in number of computers. Continuous reduction in Total time can be seen for one to five number of computers. After that, for further increase in number of computers, Total time increases slightly and remains almost constant. Continuous reduction in Calculation time with increase in number of computers can also be seen in this figure. It can also be observed that Communication time increases with increase in number of computers. This increase is gradual up to five number of computers. Sudden increase in Communication time can be seen at six number of computers. Similar type of variation in components of computational time was also observed when supercomputer PARAM 10000 was used (see Fig. 3.13(a)). But in case of PARAM 10000, contribution of Communication time in Total time was very less as compared to Windows NT Cluster. Moreover in case of PARAM 10000, continuous reduction in Total time was observed, whereas in case of Windows NT Cluster increase in Total time was observed when more



(a) Variation in computational time components



(b) Variation in Speedup (R)

Fig. 7.7 Variation in computational time components and Speedup (R) for MIM solver for data set of size 1226 × 1226

Table 7.7 Computational time variation and performance of MIM solver for data set of different sizes on Windows NT cluster

(a) Results for data set of size 870 × 870

No. of computers	Total	Comm	Cal	Speedup	Efficiency	MFLOPS
1	162.16	0.00	162.16	1.00	100.00	8.14
2	97.05	5.92	91.14	1.67	83.54	13.60
3	80.92	13.64	67.29	2.00	66.79	16.31
4	84.75	21.68	63.07	1.91	47.83	15.58
5	85.45	29.07	56.38	1.90	37.95	15.45
6	151.58	74.50	77.08	1.07	17.83	8.71
7	169.02	82.85	86.17	0.96	13.71	7.81
8	168.64	87.22	81.42	0.96	12.02	7.83

(b) Results for data set of size 1226 × 1226

No. of computers	Total	Comm	Cal	Speedup	Efficiency	MFLOPS
1	444.95	0.00	444.95	1.00	100.00	8.30
2	252.46	12.02	240.43	1.76	88.12	14.62
3	200.68	26.92	173.76	2.22	73.91	18.40
4	196.78	43.54	153.24	2.26	56.53	18.76
5	197.52	58.46	139.06	2.25	45.05	18.69
6	323.19	184.78	138.41	1.38	22.95	11.42
7	337.01	144.11	192.89	1.32	18.86	10.95
8	335.21	150.56	184.65	1.33	16.59	11.01

(c) Results for data set of size 1352 × 1352

No. of computers	Total	Comm	Cal	Speedup	Efficiency	MFLOPS
1	566.55	0.00	566.55	1.00	100.00	8.74
2	320.15	14.02	306.13	1.77	88.48	15.46
3	257.18	31.81	225.37	2.20	73.43	19.25
4	234.73	52.04	182.69	2.41	60.34	21.09
5	227.67	70.06	157.61	2.49	49.77	21.74
6	370.41	208.64	161.77	1.53	25.49	13.36
7	402.45	168.27	234.18	1.41	20.11	12.30
8	393.11	173.36	219.75	1.44	18.01	12.59

(d) Results for data set of size 1722 × 1722

No. of computers	Total	Comm	Cal	Speedup	Efficiency	MFLOPS
1	1258.91	0.00	1258.91	1.00	100.00	8.12
2	694.45	21.99	672.46	1.81	90.64	14.72
3	522.33	52.05	470.28	2.41	80.34	19.57
4	477.41	84.15	393.25	2.64	65.92	21.42
5	448.47	117.73	330.74	2.81	56.14	22.80
6	648.04	315.66	332.38	1.94	32.38	15.78
7	685.84	255.71	430.13	1.84	26.22	14.91
8	690.67	270.22	420.45	1.82	22.78	14.80

(e) Results for data set of size 2312 × 2312

No. of computers	Total	Comm	Cal	Speedup	Efficiency	MFLOPS
1	3155.25	0.00	3155.25	1.00	100.00	7.84
2	1742.91	44.05	1698.86	1.81	90.52	14.19
3	1269.99	31.42	1238.57	2.48	82.82	19.48
4	1084.88	82.61	1002.26	2.91	72.71	22.80
5	999.99	159.87	840.12	3.16	63.11	24.74
6	1186.51	289.77	896.74	2.66	44.32	20.85
7	1295.10	394.97	900.13	2.44	34.80	19.10
8	1285.33	412.37	872.96	2.45	30.69	19.25

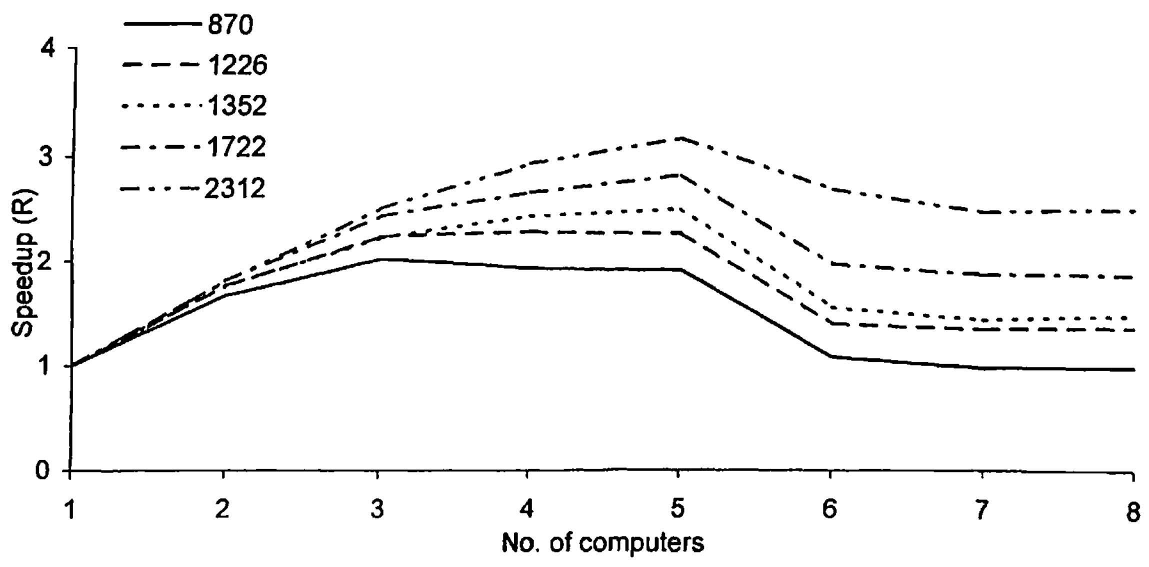


Fig. 7.8 Variation in Speedup for MIM solver for various data sets on Windows NT cluster

than five number of computers were used. This is because of higher contribution of Communication time in Total time on Windows NT Cluster. Figure 7.7 (b) shows variation in Speedup with increase in number of computers for data set of size 1226 × 1226. It can be seen that Speedup increases gradually from one to three number of computers. After three number of computers, Speedup remains almost constant till five number of computers. Sudden decrease in Speedup can be observed from five to six number of computers. Thereafter Speedup remains almost constant and more than 1.0. The performance of MIM parallel solver is better on supercomputer PARAM 10000 as compared to its performance on Windows NT Cluster (see Fig. 3.14). The Speedup remains very close to Ideal Speedup in case of PARAM 10000 whereas in case of Windows NT Cluster maximum Speedup of only 2.5 (approximately) was obtained at five number of computers.

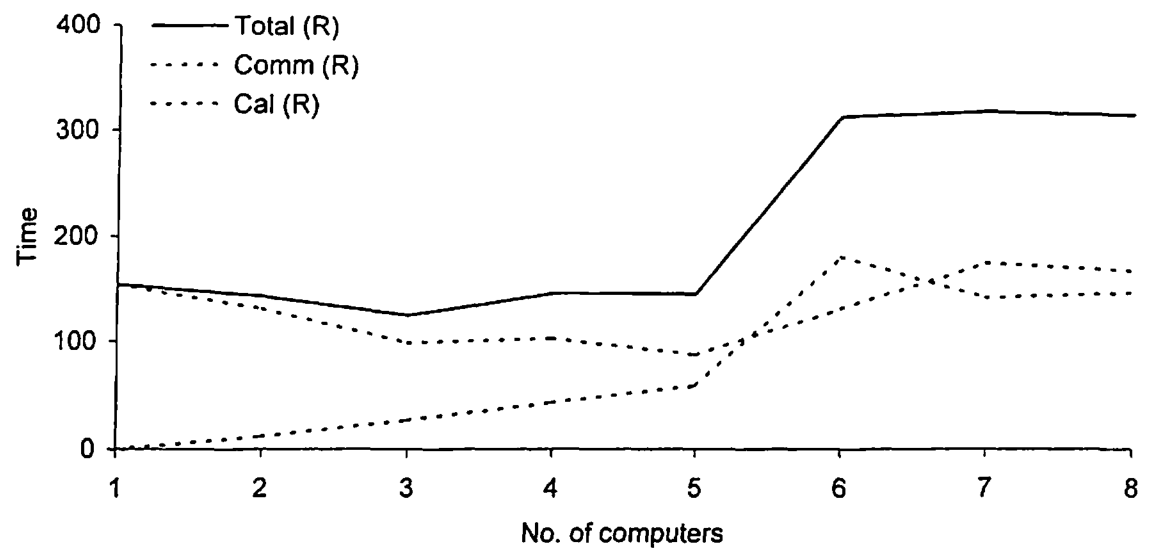
Table 7.7 shows the computational time variation and performance of MIM parallel solver for five data sets on Windows NT Cluster. It can be observed that Total time reduces with increase in number of computers employed for the analysis for all data sets under consideration. Figure 7.8 shows the variation in Speedup obtained on Windows NT Cluster for increasing data sets. It can be observed that the performance of MIM parallel solver improved significantly with increase in data size. Maximum Speedup was observed at five number of computers for all data sets under consideration. Highest Speedup of 3.16 was obtained for data set of size 2312 × 2312 at five number of computers. After observing the performance of same solver on supercomputer PARAM 10000 (see Fig. 3.15 (a)) it was found that the solver performed better on PARAM 10000 as compared to Cluster.

7.5.4 Modified Matrix Inversion Method

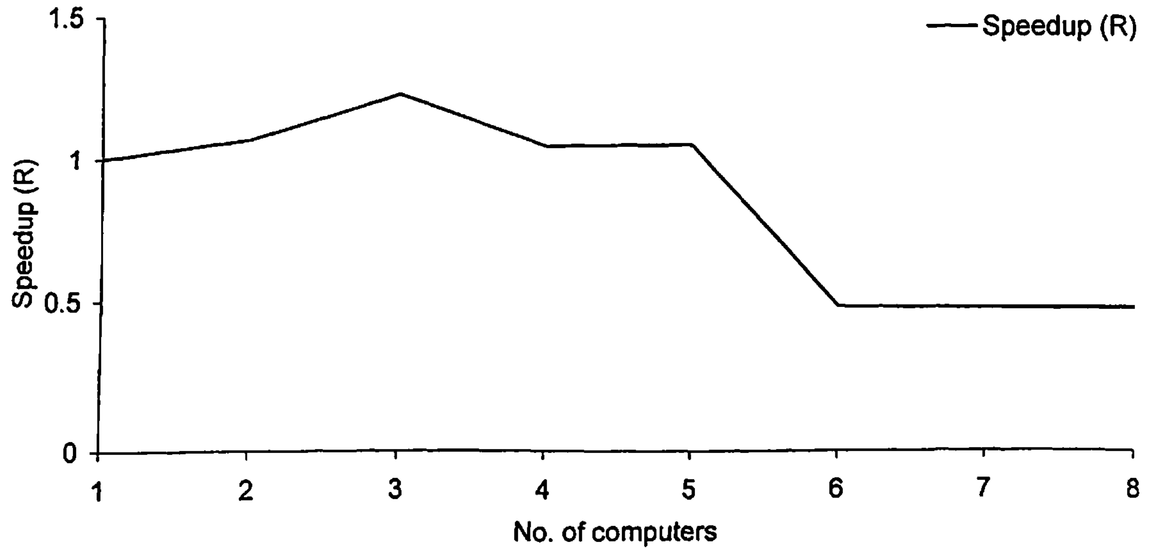
Modified Matrix Inversion Method parallel solver (MMIM) is developed especially to handle the system of linear equations developed in finite element analysis. It ignores the computations at those locations where any element in the matrix has zero value (see section 3.7). This solver takes minimum computational time on supercomputer PARAM 10000 as compared to the other parallel solvers (discussed in Chapter 3). Data size of 1226 × 1226 is analyzed using this parallel solver (MMIM) on Windows NT Cluster.

Figure 7.9 (a) shows obtained computational time variation. It can be observed that Total time reduces by negligible amount when two and three computers are used. Total time increases when four and five number of computers were employed for the analysis. Total time suddenly increases by considerable amount when six computers were used. Further, Total time remains nearly constant for seven and eight number of computers. This is mainly because of higher contribution of Communication time. Communication time increases significantly with increase in number of computers but Calculation time reduces with increase in number of computers. It can be observed that Communication time and Calculation time curves cross each other after five number of computers and at the same instance Total time suddenly increases. This signifies that Communication time is higher than Calculation time that resulted in increased Total time. When this solver was tested for same data set on supercomputer PARAM 10000 it was observed that Total time continuously decreases with increase in number of processors (see Table 3.26 (a)). Figure 7.9 (b) shows the variation in Speedup with number of computers for data set of size 1226 × 1226. It can be seen that Speedup remains greater than 1.0 for number of computer from one to five. Speedup decreases and remains less than 1.0 for six to eight number of computers. The Speedup is not adequate as maximum Speedup is just 1.23 at three number of computers whereas in case of PARAM 10000 maximum Speedup obtained was 2.06 at seven number of processors (Table 3.26 (b)).

Various data sets of increasing sizes were analyzed using MMIM parallel solver and computational time was measured (see Table 7.8). It can be seen in the tables that initially Total time reduces and at a certain number of computers, the Total time starts increasing with further increase in number of computers. It can also be observed that Communication time increases with increase in number of computers. Figure 7.10 shows variation in Speedup with increase in number of computers for various data sets under consideration. It can be observed that Speedup increases initially and reaches to its peak value at five number of computers for almost all data sets. Speedup starts decreasing with further increase in number of computers and in few cases its magnitude reaches below 1.0 also. When similar data sets were analyzed on supercomputer PARAM 10000, it was observed that Speedup increases continuously with increase in number of processors (see Fig. 3.21 (a)).



(a) Variation in computational time components



(b) Variation in Speedup (R)

Fig. 7.9 Variation in computational time components and Speedup (R) for MMIM solver for data set of size 1226 × 1226

Table 7.8 Computational time variation and performance of MMIM solver for data set of different sizes on Windows NT cluster

(a) Results for data set of size 870 × 870

No. of computers	Total	Comm	Cal	Speedup	Efficiency	MFLOPS
1	55.97	0.00	55.97	1.00	100.00	23.59
2	54.49	5.85	48.64	1.03	51.36	24.23
3	51.55	13.35	38.20	1.09	36.19	25.61
4	66.27	21.36	44.90	0.84	21.11	19.92
5	67.99	29.12	38.87	0.82	16.46	19.42
6	152.96	76.35	76.60	0.37	6.10	8.63
7	164.61	83.22	81.39	0.34	4.86	8.02
8	194.86	88.09	106.77	0.29	3.59	6.77

(b) Results for data set of size 1226 × 1226

No. of computers	Total	Comm	Cal	Speedup	Efficiency	MFLOPS
1	154.40	0.00	154.40	1.00	100.00	23.91
2	143.91	11.86	132.05	1.07	53.65	25.65
3	125.15	26.60	98.54	1.23	41.13	29.50
4	146.34	43.02	103.33	1.06	26.38	25.23
5	145.86	58.45	87.40	1.06	21.17	25.31
6	312.20	180.48	131.73	0.49	8.24	11.82
7	317.87	142.65	175.23	0.49	6.94	11.61
8	315.66	147.59	168.06	0.49	6.11	11.69

(c) Results for data set of size 1352 × 1352

No. of computers	Total	Comm	Cal	Speedup	Efficiency	MFLOPS
1	322.81	0.00	322.81	1.00	100.00	15.33
2	225.84	13.74	212.10	1.43	71.47	21.92
3	192.74	31.68	161.06	1.67	55.83	25.68
4	230.50	52.61	177.89	1.40	35.01	21.47
5	198.57	70.59	127.98	1.63	32.51	24.93
6	372.39	206.26	166.13	0.87	14.45	13.29
7	391.33	165.31	226.02	0.82	11.78	12.65
8	408.02	169.15	238.88	0.79	9.89	12.13

(d) Results for data set of size 1722 × 1722

No. of computers	Total	Comm	Cal	Speedup	Efficiency	MFLOPS
1	685.00	0.00	685.00	1.00	100.00	14.93
2	529.34	21.59	507.75	1.29	64.70	19.32
3	445.36	46.16	399.20	1.54	51.27	22.96
4	420.70	84.00	336.70	1.63	40.71	24.30
5	401.02	117.67	283.35	1.71	34.16	25.50
6	618.34	303.23	315.11	1.11	18.46	16.54
7	654.90	249.36	405.54	1.05	14.94	15.61
8	672.94	261.75	411.19	1.02	12.72	15.19

(e) Results for data set of size 2312 × 2312

No. of computers	Total	Comm	Cal	Speedup	Efficiency	MFLOPS
1	1660.00	0.00	1660.00	1.00	100.00	14.90
2	1352.59	41.17	1311.42	1.23	61.36	18.29
3	1047.06	43.29	1003.77	1.59	52.85	23.63
4	1155.18	319.50	835.68	1.44	35.93	21.42
5	1080.75	253.12	827.63	1.54	30.72	22.89
6	1083.00	294.37	788.63	1.53	25.55	22.84
7	1170.79	324.11	846.68	1.42	20.25	21.13
8	1286.45	349.49	936.97	1.29	16.13	19.23

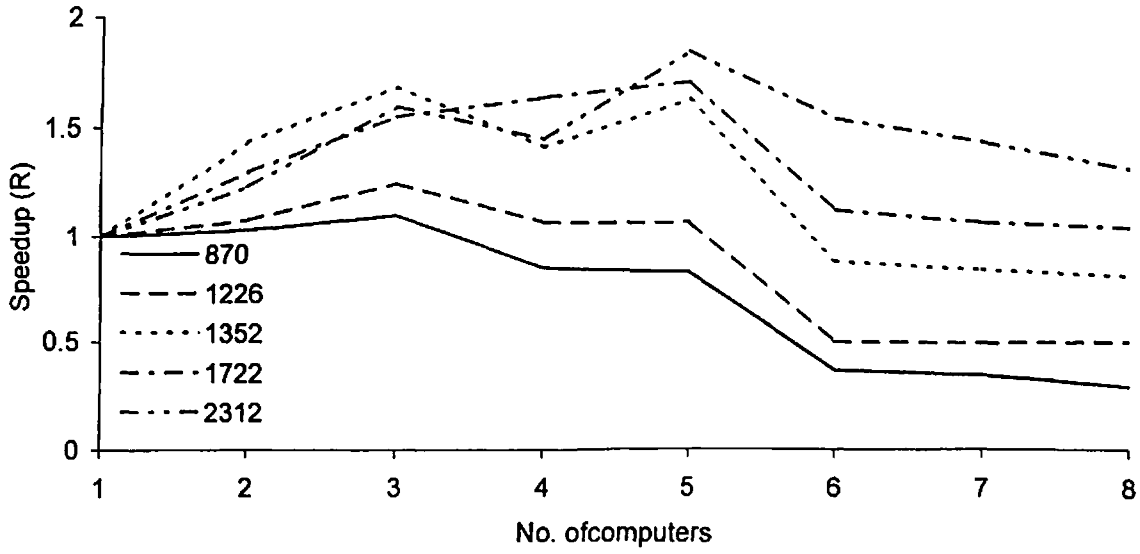


Fig. 7.10 Variation in Speedup for MMIM solver for various data sets on Windows NT cluster

7.6 COMPARISON OF PARALLEL SOLVERS

In the previous section, performance of four different parallel solvers is presented. Five distinct data sets of increasing sizes were analyzed using these parallel solvers on Windows NT Cluster. It was observed that performance of all these solvers is best at five number of computers (see Fig. 7.4, 7.6, 7.8 and 7.10). Table 7.9 shows values of Total time obtained by four parallel solvers with different data sizes at five number of computers. Figure 7.11 shows variation in the Speedup with increasing data size for four parallel solvers when five number of computers were employed. It can be observed from this figure that Speedup obtained by MIM parallel solver is maximum whereas by GSM parallel solver is minimum among all four parallel solvers. It can also be observed that GEM and GSM parallel solvers give Speedup less than 1.0 for all data sets under consideration.

When Total time required for the analysis was compared then it was found that GEM parallel solver takes minimum time for analysis on single computer as compared to the other parallel solvers (see Tables 7.4, 7.6, 7.7 and 7.8). It is mainly because it requires significantly lesser computations for getting solution as compared to the other methods. Moreover no communication is carried out when single computer is employed for the analysis. When multiple computers are used it was found that all solvers performed better when five computers were used. It was also observed that at five number of computers Total time taken by MMIM parallel solver is minimum as compared to the Total time taken by other parallel solvers for different data sets under consideration. It can be observed that Total time required by GEM parallel solver for all data sets is slightly higher than the Total time required by MMIM parallel solver. Furthermore GEM parallel solver can be improved by trimming down the Communication time so that it may perform even better than MMIM parallel solver.

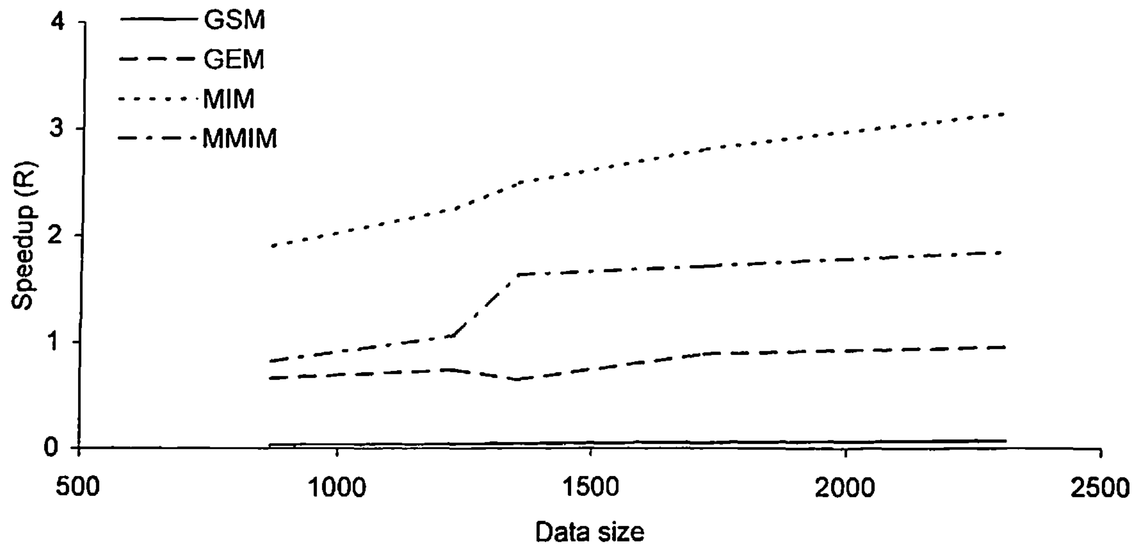


Fig. 7.11 Comparison of four parallel solvers when five computers were employed

Table 7.9 Total time obtained by four parallel solvers with different data sizes at five number of computers

Data size	Parallel solver			
	GSM	GEM	MIM	MMIM
870	3623.41	69.72	85.45	67.99
1226	8435.47	171.55	197.52	145.86
1352	9394.76	215.16	227.67	198.57
1722	14914.73	419.79	448.47	401.02
2312	27480.97	999.55	999.99	897.66

7.7 FINITE ELEMENT ANALYSIS ON WINDOWS NT CLUSTER

As discussed in Chapter 3, major portion of computational time required in finite element analysis is consumed in the process of finding unknowns by solving system of linear equations generated. To reduce this portion of computational time, four parallel solvers were developed and implemented on supercomputer PARAM 10000 as well as on Windows NT Cluster. It was found that MMIM parallel solver takes lesser time on supercomputer PARAM 10000 as well as on Windows NT Cluster as compared to other parallel solvers. Therefore this parallel solver is implemented in finite element codes (presented in Chapter 4 and Chapter 5), which are capable of solving linear elastic and non-linear plastic problems. These finite element codes are once again developed on Windows NT platform with MMIM parallel solver. One problem from each type is analyzed with these finite element codes on Windows NT Cluster and required computational time and its components are measured.

7.7.1 Linear Elastic Finite Element Analysis

As described in Chapter 4, analysis of anchorage zone which develops in prestressed post-tensioned concrete beam is idealized as two-dimensional plane stress problem. The beam was discretized three times using 1600, 1136 and 784 constant strain triangular elements with 861, 613 and 435 nodes resulting into global stiffness matrices of size 1722 × 1722, 1226 × 1226 and 870 × 870 respectively. Other details of the problem are same as described in section 4.5. These three cases were analyzed on Windows NT Cluster by increasing number of computers from one to eight and different components of computational time were measured.

From Table 7.10, it can be seen that Total time reduces with increasing number of computers from one to four. Further increase in number of computers resulted in increase in Total time. It can also be seen that problems having stiffness matrix of size 870 × 870 and 1226 × 1226, minimum Total time was measured at three number of computers. For problem with stiffness matrix of size 1722 × 1722, minimum Total time was obtained at five number of computers. This signifies that number of computers at which minimum Total time can be obtained increases with increase in size of stiffness matrix. This is also

Table 7.10 Computational time and performance of parallelized FEM code on Windows NT cluster for problems with different global stiffness matrix of sizes

(a) Results for problem with global stiffness matrix size 870 × 870

No. of computers	Total	Comm	Cal	Speedup	Efficiency
1	54.63	0.00	54.63	1.00	100.00
2	52.93	6.19	46.74	1.03	51.60
3	49.40	13.56	35.84	1.11	36.86
4	62.47	21.51	40.97	0.87	21.86
5	64.95	29.62	35.33	0.84	16.82
6	142.36	78.47	63.89	0.38	6.40
7	161.34	84.00	77.33	0.34	4.84
8	162.65	92.62	70.03	0.34	4.20

(b) Results for problem with global stiffness matrix size 1226 × 1226

No. of computers	Total	Comm	Cal	Speedup	Efficiency
1	155.01	0.00	155.01	1.00	100.00
2	143.41	12.09	131.33	1.08	54.04
3	124.82	27.20	97.62	1.24	41.40
4	145.66	43.37	102.29	1.06	26.61
5	139.62	58.49	81.12	1.11	22.21
6	294.41	176.77	117.64	0.53	8.78
7	332.71	144.87	187.84	0.47	6.66
8	321.37	154.96	166.42	0.48	6.03

(c) Results for problem with global stiffness matrix size 1722 × 1722

No. of computers	Total	Comm	Cal	Speedup	Efficiency
1	779.31	0.00	779.31	1.00	100.00
2	641.29	17.94	623.36	1.22	60.76
3	532.25	46.42	485.83	1.46	48.81
4	511.54	69.50	442.04	1.52	38.09
5	464.79	116.36	348.43	1.68	33.53
6	1109.72	303.76	805.96	0.70	11.70
7	714.93	248.11	466.82	1.09	15.57
8	692.69	264.35	428.33	1.13	14.06

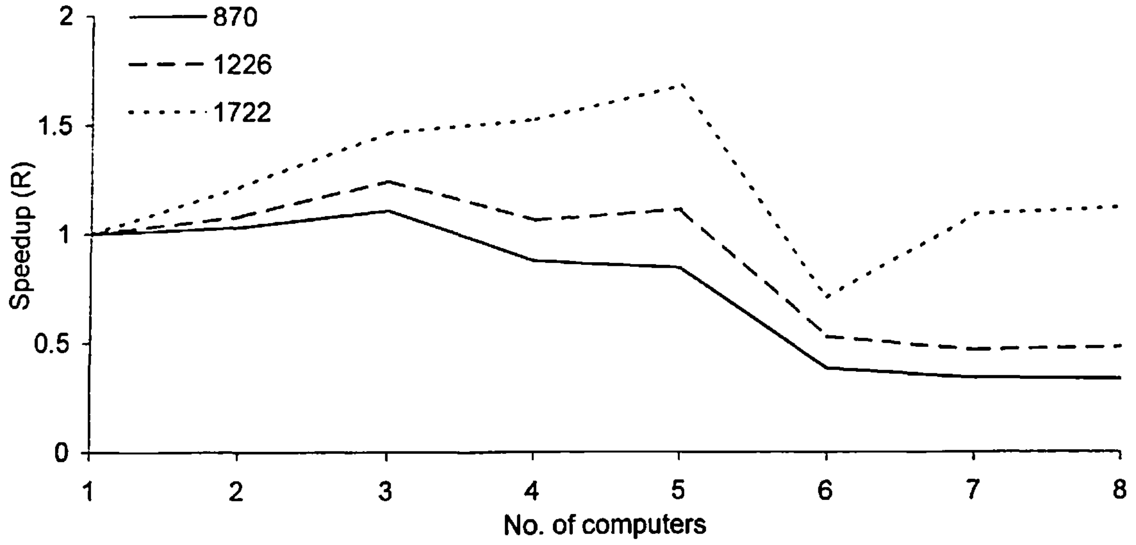


Fig. 7.12 Variation in Speedup (R) for FEM code for problems with stiffness matrices of increasing size

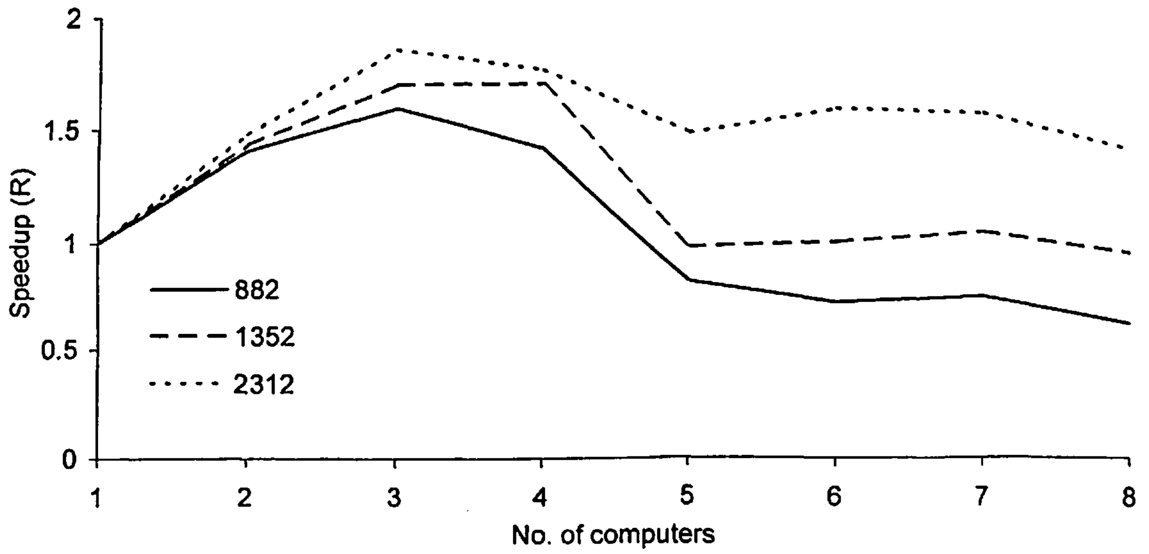


Fig. 7.13 Variation in Speedup (R) for FEMLD for problems with stiffness matrices of increasing size

Table 7.11 Computational time and performance of FEMLD on Windows NT cluster for problems with different global stiffness matrix of sizes

(a) Results for problem with global stiffness matrix size 870 × 870

No. of computers	Total	Comm	Cal	Speedup	Efficiency
1	8123.16	0.00	8123.16	1.00	100.00
2	5784.60	504.00	5280.60	1.40	70.21
3	5102.99	1151.88	3951.11	1.59	53.06
4	5755.10	1854.21	3900.89	1.41	35.29
5	9927.55	5547.29	4380.26	0.82	16.36
6	11462.10	5100.26	6361.84	0.71	11.81
7	11084.43	5413.59	5670.84	0.73	10.47
8	13423.46	7477.81	5945.65	0.61	7.56

(b) Results for problem with global stiffness matrix size 1352 × 1352

No. of computers	Total	Comm	Cal	Speedup	Efficiency
1	12080.67	0.00	12080.67	1.00	100.00
2	8437.57	507.86	7929.70	1.43	71.59
3	7116.41	1176.63	5939.78	1.70	56.59
4	7094.74	1849.59	5245.15	1.70	42.57
5	12356.92	8356.05	4000.88	0.98	19.55
6	12190.18	5050.82	7139.36	0.99	16.52
7	11694.46	4784.83	6909.63	1.03	14.76
8	12956.97	6404.81	6552.16	0.93	11.65

(c) Results for problem with global stiffness matrix size 2312 × 2312

No. of computers	Total	Comm	Cal	Speedup	Efficiency
1	68038.36	0.00	68038.36	1.00	100.00
2	46005.76	1528.44	44477.33	1.48	73.95
3	36645.07	1642.99	35002.08	1.86	61.89
4	38534.73	3876.66	34658.07	1.77	44.14
5	45719.68	16877.30	28842.39	1.49	29.76
6	42637.61	11743.77	30893.84	1.60	26.60
7	43388.94	13204.22	30184.73	1.57	22.40
8	48598.39	18957.33	29641.06	1.40	17.50

reflected in Speedup as shown in Fig. 7.12. It can be seen from this figure that overall performance of the FEM code improves with increase in matrix size. Obtained Speedup is not encouraging as maximum Speedup obtained was 1.68 only at five number of computers for problem with stiffness matrix of size 1722 × 1722.

When performance of FEM code on Windows NT Cluster was compared with its performance on supercomputer PARAM 10000, it was found that FEM code performed slightly better on supercomputer PARAM 10000 as compared to Windows NT Cluster (see Tables 4.2 to 4.9). On supercomputer PARAM 10000, Speedup continuously increases with increase in number of processors, whereas on Windows NT Cluster, Speedup reduces when more than five number of computer were used. Overall performance of FEM code on Windows NT Cluster is encouraging and can be further improved by improving communication technique.

7.7.2 Non-Linear Plastic Finite Element Analysis

The problem of simple compression of a solid cylinder (discussed in section 5.3.1) is solved using the developed code on Windows NT Cluster. This problem was discretized three times using meshes of different sizes to obtain stiffness matrices of sizes 882 × 882, 1352 × 1352 and 2312 × 2312. Other details of the problem are kept same as discussed in section 5.3.1. These problems were analyzed on Windows NT Cluster by employing one to eight computers and different components of computational time were measured. It was observed that 84, 36 and 33 iterations were required to analyze these problems.

From Table 7.11 it can be seen that Total time reduces with increase in number of computers. It reaches to its minimum value at three number of computers for problem with stiffness matrix of size 882 × 882 and 2312 × 2312 and four number of computers for problem with stiffness matrix of size 1226 × 1226. Further increase in number of computers resulted in increase in Total time. Figure 7.13 shows the variation in Speedup obtained by FEMLD on Windows NT Cluster for problem with stiffness matrix of various sizes. Overall it can be seen that Speedup increases with increase in number of computers and reaches to its peak value. Then Speedup starts decreasing with further increase in number of computers. Maximum Speedup of 1.86 was obtained at three number of computers for problem with stiffness matrix of size 2312 × 2312. It can also be seen from this figure that the performance of FEMLD improves with increase in size of stiffness matrix. When the performance was compared with supercomputer PARAM 10000, it was found that performance of FEMLD on Windows NT Cluster is nearly similar to its performance on supercomputer PARAM 10000 (see Table 5.3 to 5.5). It was also observed that PARAM 10000 gives higher Speedup at higher number of processors as compared to Windows NT Cluster.

7.8 SUMMARY

In this chapter, an alternative to supercomputer through Cluster computing technique is suggested for structural analysis using finite element method. Two Windows NT Clusters are formed in which, one consists of PC's having similar configurations (400 MHz) connected by 100 MBPS switch and other consists of PC's having different configurations (300 MHz, 400 MHz, and 500 MHz) connected by 10 MBPS HUB. Matrix Inversion Method parallel solver is developed on Windows NT platform and tested on the Cluster of PC's having similar configurations. Data sets of different sizes are solved on this Cluster and various components of computational time are measured. The comparison of developed Cluster with supercomputer PARAM 10000 is carried out by comparing the performance of MIM parallel solvers on both the computing systems. It was found that developed Cluster is effective in reducing the computational time by employing more number of computers for the solution. It is also observed that the contribution of Communication time in Total time is more in Windows NT Cluster as compared to supercomputer PARAM 10000.

Four parallel solvers (discussed in Chapter 3) are developed on Windows NT platform and tested for various sizes of data sets on Windows NT Cluster of PC's having different configurations. It was found that MIM parallel solver gave highest Speedup as compared to the other solvers. It was also found that MMIM parallel solver takes minimum computational time as compared to the other solvers and hence it was chosen for implementation in finite element analysis.

Two parallelized finite element codes capable of analyzing linear elastic problems (discussed in Chapter 4) and non-linear plastic problem (discussed in Chapter 5) are developed on Windows NT Cluster. One sample problem from each category is analyzed by these codes by increasing the number of computers and their performance is measured. It was found that, Total time required in finite element analysis is reduced by employing more number of computers of developed Windows NT Cluster. It was also found that analysis could be carried out in minimum Total time by employing three to five number of computers depending on the size of the finite element mesh. It was also observed that excessive increase in number of computers resulted in increase in Total time.

It was found that these parallel finite element codes performed better on supercomputer PARAM 10000 as compared to Windows NT Cluster. But still Windows NT Cluster can be used as an alternative of supercomputer PARAM 10000 as it is less expensive than supercomputer PARAM 10000. Moreover, computers that are used in Windows NT Cluster can be very easily replaced with better and faster computers, which would be the biggest advantage of using Windows NT Cluster.

CHAPTER 8

CONCLUDING REMARKS AND FUTURE SCOPE

8.1 SUMMARY AND CONCLUSIONS

The present work shows an implementation of parallel computing technique in two-dimensional linear and non-linear finite element analysis. For parallel processing supercomputer PARAM 10000 and Windows NT Cluster has been employed. Based on the work carried out and presented in this report, following points can be summarize;

- Various components of computational time and effects of user activities on these components were studied. A timer was also developed to measure the User time spent for a particular code segment.
- Using C programming language, three parallel solvers were developed using Gauss-Seidel Method, Gauss Elimination Method and Matrix Inversion Method for solving system of linear equations. After comparing the computational time and their performance results, it was found that Matrix Inversion Parallel solver is better as compared to other two parallel solvers.
- Matrix Inversion parallel solver was further developed in FORTRAN77 programming language. After comparing its performance with the previously developed solver in C language, it was found that parallel solver written in C language is better as compared to the parallel solver developed in FORTRAN77 language.
- Comparison of Blocking and Non-blocking communication mechanism was carried out by implementing both the communication mechanisms in the Matrix Inversion parallel solver and it was found that both the communication mechanisms are equally effective for communication.
- Matrix Inversion parallel solver was further modified to solve system of linear equations generated especially in finite element analysis. After comparing the performance of the original and modified Matrix Inversion parallel solver, it was found that the modified parallel solver is much faster in solving system of linear equation in finite element analysis as compared to the original solver.
- Software for two-dimensional linear elastic finite element analysis was developed on the platform of supercomputer PARAM 10000. Modified Matrix Inversion parallel solver was incorporated in this software so that computational time could be saved in overall computations.

- With the help of the developed software, a case study problem of stress analysis in anchorage zone was presented. Various Stress distribution in anchorage zone in prestressed post tensioned concrete beam subjected to concentric and eccentric prestressing forces was obtained and presented. The obtained stress variation was also compared with the literature and discussed.
- The analysis was carried out on supercomputer PARAM 10000 and it was found that significant amount of computational time could be saved by using multiple number of processors for the analysis.
- The effect of Poisson's ratio and load area ratio on anchorage zone stress was studied and an expression to compute bursting tensile force was developed. This expression includes the effect of Poisson's ratio that was ignored in the expression given in Indian Standard Code.
- Effect of eccentricity of prestressing forces on magnitude of bursting tensile force was studied and it was found that maximum magnitude of bursting tensile force could be found at zero eccentricity.
- Existence of spalling zone in prestressed post-tensioned concrete beams was confirmed in the present investigation. It was also found that the magnitude of transverse tensile stresses developed in spalling zone was higher than the magnitude of transverse tensile stresses developed in anchorage zone.
- Adopting flow formulation in finite element analysis and the Modified Matrix Inversion parallel solver, a generalized software FEMLD was developed to analyze large deformation problems categorized under metal forming problems. This software was developed using three noded constant strain triangular elements (FEMLD3) as well as four noded rectangular elements (FEMLD4).
- Two sample problems of compression of solid cylinder (axisymmetric condition) and compression of prismatic bar (plane strain condition) were analyzed using FEMLD3 as well as FEMLD4. Distribution of various stress obtained were presented and discussed.
- The same case study problems were also analyzed using commercial software FORGE2 and it was found that various results obtained from developed code have excellent agreement with the corresponding results obtained from FORGE2.
- The same case study problems were also analyzed using commercial software ANSYS and obtained load-compression relationship was compared with the load-

compression relationship obtained using developed code. The limitations of the commercial software ANSYS in solving large deformation problems were also discussed.

- The performance of the FEMLD software was tested for huge data sizes and the developed software showed improved performance for large size problems.
- FEMLD was developed for generalized large deformation problem so it has some limitations. These limitations were sorted out and rectified by simulating two flat dies. Modified FEMLD was developed to simulate compression process of various materials with various cross-sections between two rigid moving flat dies.
- To test the modified FEMLD software, four case study problems were analyzed and presented. They includes axisymmetric compression of solid cylinder, lateral compression of rectangular metallic tubes, fold formation in axisymmetric compression of hollow round metallic tubes and lateral compression of round tube between two concentrated loads.
- To verify the results obtained from the developed software, the same four problems were also analyzed using commercial software FORGE2. Various results obtained using developed code and FORGE2 were compared and it was found that the results match well.
- These problems were analyzed using supercomputer PARAM 10000 and reduction in computational time was obtained using more number of processors of supercomputer PARAM 10000.
- Cluster of eight computers having equal hardware configuration operating on Windows NT operating system was formed in order to execute parallel codes. Matrix Inversion parallel solver was implemented on this Cluster and tested for different data sets of increasing sizes. It was found that computational time reduces with the increase in number of computer employed for the analysis.
- The similar data sets were also solved using one to eight number of processors of supercomputer PARAM 10000. Computational time results were obtained and compared with the corresponding results obtained on Windows NT Cluster. It was found that both computing systems are good enough to save computational time. It was also found that the contribution of Communication time towards Total time is more in Windows NT Cluster as compared to the supercomputer PARAM 10000.

- A cluster of eight conventional computers with different hardware configuration was also developed. Four parallel solvers, namely Gauss-Seidel, Gauss Elimination, Matrix Inversion and Modified Matrix Inversion parallel solvers were implemented on this Cluster. Several data sets of varying data sizes were analyzed on this Cluster using four parallel solvers. It was found that Modified Matrix Inversion parallel solver gave best performance among all four parallel solvers.
- Two parallel programs for two-dimensional linear elastic finite element analysis and two-dimensional non-linear finite element analysis of large deformation problems were developed on this Cluster. Case study problems in both the categories were solved by changing number of computers from one to eight and obtained computational time variations were presented. It was found that the computational time reduces with increase in number of computer.

8.2 FURTHER SCOPE OF WORK

In the developed software for linear and non-linear finite element analysis, only two elements have been used namely three noded constant strain triangular elements and four noded rectangular elements. This is one of the major limitation of the presented work. Higher order elements like shell elements could be incorporated in the developed code in order to get more accurate results in finite element analysis. Present work deals with only two-dimensional finite element analysis. It could be extended to three-dimenstional analysis also. In non-linear finite element analysis, simulation of two flat dies was presented. More die shapes could be simulated so that more complex problems like tube inversion can also be analyzed. Temperature effect was ignored in the developed code that could be incorporated in order to study thermal stress in metal forming problems. The problem of crack formation or fracture of material during deformation is quite common in metal forming problems so various criteria to identify the fracture could be included in the developed code.

?

The developed parallel solvers were tested on PARAM 10000 machine that includes processors of 400 MHz speed. With the recent advancements in the area of computer technology, conventional computers with 3 GHz processors are now available. Hence the computational time consumption in supercomputer PARAM 10000 is relatively similar to the computational time consumption on conventional computer with latest hardware. Hence the presented codes should also be tested on supercomputers with latest hardware. Very little work has been carried out in implementation of cluster computing technique in finite element analysis. Significant amount of work is possible in this area also.

REFERENCES

1. M. J. Quinn, "Parallel Computing: Theory and Practice", second edition, McGraw-Hill, New York, 1994.
2. A. Grama, A. Gupta, G. Karypis and V. Kumar, "Introduction to Parallel Computing", second edition, Pearson Education, Singapore, 2004.
3. M. Snir, S. Otto, S. Hass-Lederman, D. Walker and J. Dongarra, "MPI: The Complete Reference", first edition, The MIT Press, Cambridge, 1996.
4. http://www-unix.mcs.anl.gov/mpi/mpich/, Dec. 2005.
5. Y. Kanetkar, "Let Us C", third edition, BPB Publications, New Delhi, 1999.
6. Y. Kanetkar, "Let Us C++", second edition, BPB Publications, New Delhi, 1999.
7. S. Lipschutz and A. Poe, "Schaum's Outline Series: Programming with FORTRAN", first edition, McGraw-Hill, Singapore, 1982.
8. http://param.bits-pilani.ac.in, Dec. 2005.
9. C. S. Krishnamoorthy, "Finite Element Analysis: Theory and Programming", second edition, Tata McGraw-Hill, 1999.
10. J. N. Reddy, "An Introduction to The Finite Element Method", third edition, McGraw-Hill, New York, 1993.
11. T. R. Chandrupatala and A. D. Belagundu, "Introduction to Finite Elements in Engineering", second edition, Prentice-Hall of India, New Delhi, 1997.
12. J. Mackerle, "Finite and Boundary Elements and Supercomputing - A bibliography (1989–1991)", Finite Elements in Analysis and Design, Vol. 12, No 2, 1992, pp 151-159.
13. S. Das, "UNIX: Concepts and Applications", second edition, Tata McGraw-Hill, New Delhi, 1999.
14. K. R. Wadleigh and I. L. Crawford, "Software Optimization for High-Performance Computing", first edition, Prentice Hall PTR, New Jersey, 2000.
15. S. Rajasekaran, "Numerical Methods in Science and Engineering A Practical Approach", second edition, Wheeler Publishers, Allahabad, 1992.
16. S. S. Shastry, "Introductory Methods of Numerical Analysis", Prentice Hall of India, New Delhi, 1994.

17. G. Thiagarajan and V. Aravamuthan, "Parallelization Strategies for Element-By-Element Preconditioned Conjugate Gradient Solver Using High-Performance FORTRAN for Unstructured Finite-Element Applications on Linux Clusters", ASCE Journal of Computing In Civil Engineering, Vol. 16, No. 1, 2002, pp 1-10.
18. A. I. Khan and B. H. V. Topping, "Parallel Finite Element Analysis Using Jacobi-Conditioned Conjugate Gradient Algorithm", Advances in Engineering Software, Vol. 25, No. 3, 1996, pp 309-319.
19. G. Mahinthakumar and F. Saied, "A Hybrid Mpi-OpenMP Implementation of an Implicit Finite-Element Code on Parallel Architectures", International Journal of High Performance Computing Applications, Vol. 16, No. 4, 2002, pp 371-393.
20. K. Danielson and R. Namburu, " Nonlinear Dynamic Finite Element Analysis on Parallel Computers using FORTRAN90 an d MPI", Advances in Engineering Software, Vol. 29, No. 3, 1998, pp 179-186.
21. J. Sziveri and B. H. V. Topping, "Transient Dynamic Nonlinear Analysis Using MIMD Computer Architectures", Journal of Computing in Civil Engineering, Vol. 14, No. 2, 2000, pp 79-90.
22. R. B. King and V. Sonnad, "Implementation of an Element-by-Element Solution Algorithm for The Finite Element Method on A Coarse-Grained Parallel Computer", Computer Methods in Applied Mechanics and Engineering, Vol. 65, No. 1, 1987, pp 47-59.
23. E. Chu, A. George and D. Quesnel, "Parallel Matrix Inversion on A Subcube-Grid", Parallel Computing, Vol. 19, No. 3, 1993, pp 243-256.
24. T. Kant, M. S. Shah and K. S. Ramesh, "Composite Materials Analysis of Parallel Supercomputer", Proceedings of Second Indian Transputers User's Group, Hydrabad, 1994, pp 85-94.
25. T. Kant and M. S. Shah, "Finite Element Analysis of Structures of Composite Materials on Parallel Supercomputer", Proceeding of High Performance Computing, New Delhi, 1995, pp 193-195.
26. T. Kant and M. S. Shah, "Finite Element Analysis of Fiber-Reinforced Polymer Shells Using Higher Order Shear Deformation Theories on Parallel Distributed Memory Machines", International Journal of Computer Applications in Technology, Vol. 31, No. 3, 1998, pp 206-210.

27. K. S. Ramesh and M. S. Shah, "Implementation of Parallel Preconditioned Conjugate Gradient Solver for FEA on PARAM", Proceedings of Scientific Computing and Mathematical Modeling, Bangalore, 1993, pp 49-56.
28. T. Kant, M. S. Shah and B. B. Mahanta, "Thermo-Mechanical Analysis of Fiber Reinforced Composite Plates and Shells on Supercomputers", Proceedings of Structural Engineering Convention, IIT, Kharagpur, 2003, pp 501-510.
29. M. Shah and K. S. Ramesh, "Fracture Analysis on Parallel Supercomputer", Proceedings of Conference of Indian Transputer Users Group, Pune, 19993, pp 204-207.
30. A. R. M. Rao, "MPI-Based Parallel Finite Element Approaches for Implicit Nonlinear Dynamic Analysis Employing Sparse PCG Solvers", Advances in Engineering Software, Vol. 36, No. 3, 2005, pp 181-198.
31. A. R. M. Rao, T. V. S. R. A. Rao and B. Dattaguru, "A New Parallel Overlapped Domain Decomposition Method for Nonlinear Dynamic Finite Element Analysis", Computers and Structures, Vol. 81, No. 26, 2003, pp 2441-2454.
32. N. Krishna Raju, "Prestressed Concrete", second edition, Tata McGraw-Hill, New Delhi, 1995.
33. T. Y. Lin and N. H. Burns, "Design of Prestressed Concrete Structures", second edition, John Wiley & Sons, New York, 1982.
34. Y. Guyon, " Contraintes dans les pieces prismatiques soumises a des forces appliques sur leurs bases, an voisinage de ces bases", International Association for Bridge and Structural Engineering, Vol. 2, 1951, pp 165-226.
35. P. K. Som and K. Ghosh, "Anchorage Zone Stresses in Prestressed Concrete Beams", ASCE, Journal of Structural Division, Vol. 90, 1964, pp 49-62.
36. K. T. S. Iyengar, "Two Dimensional Theories of Anchorage Zone Stresses in Post-Tensioned Prestressed Beams", Journal of American Concrete Institute, Vol. 59, No. 10, 1962, pp 1443-1465.
37. K. T. S. Iyengar and M. K. Prabhakara, "Anchor Zone Stresses in Prestressed Concrete Beams", ASCE, Journal of Structural Division, Vol. 97, No. 3, 1971, pp 807-824.
38. S. P. Christodoulides, "Two Dimensional Investigation of End – Anchorages of Post-Tensioned Concrete Beams", The Structural Engineer, Vol. 33, No. 4, 1955, pp 120-133.

39. S. P. Christodoulides, "The Distribution of Stresses Around The End-Anchorages of Prestressed Concrete Beams. Comparison of The Results Obtained Photoelastically, with Strain-Gauge Measurements and Theoretical Solutions", International Association for Bridge and Structural Engineering, Vol. 16, 1956, pp 50-70.

40. J. Zielinski and R. E. Rowe, "An Investigation of The Stress Distribution in The Anchorage Zones of Post-Tensioned Concrete Members", Research Report No. 9, Cement and Concrete Association, London, 1960.

41. IS : 1343 – 1980 : Indian Standard Code of Practice for Prestressed Concrete. 1st Rev. Bureau of Indian Standards, New Delhi, 1981.

42. A. L. Yettram and K. Robbins, "Anchorage Zone Stressed in Axially Post-Tensioned Members of Uniform Rectangular Section", Magazine of Concrete Research, Vol. 21, No. 67, 1969, pp 103-112.

43. Byung-Wan Jo, Yunn-Ju Byun and Ghi-Ho Tae, "Structural Behavior of Cable Anchorage Zone in Prestressed Concrete Cable-Stayed Bridge", Canadian Journal of Civil Engineering, Vol. 29, No.1, 2002, pp 171-180.

44. A. A. Ezra and R. J. Fay, "An Assessment of Energy Absorbing Devices for Prospective use in Aircraft Impact Situations", Proceedings of International Symposium Dynamic Response of Structures, Stanford University, California, 1971, pp 225-246.

45. W. Johnson and S. R. Reid, "Metallic Energy Dissipating Systems", ASCE Applied Mechanics Reviews, Vol. 31, No. 3, 1978, pp 277-288.

46. P. K. Gupta, "An Investigation into Large Deformation Behaviour of Metallic Tubes", Ph.D. Thesis, Indian Institute of Technology, Delhi, INDIA, 2000.

47. P. K. Gupta, N. K. Gupta and G. S. Sekhon, "A Study of Large Deformation Behaviour of Square and Rectangular Metallic Tubes Subjected to Lateral Compression", Proceedings of International Conference on Advances in Civil Engineering, ACE2002, Indian Institute of Technology, Kharagpur, INDIA, 2002, pp 1201-1210.

48. P. K. Gupta, N. K. Gupta and G. S. Sekhon, " Study of Lateral compression of Square and Rectangular Metallic Tubes", Proceedings of 8th International Symposium on Plasticity and Impact mechanics, IMPLAST2003, Indian Institute of Technology, Delhi, INDIA, 2003, pp 519-527.

49. N. K. Gupta, G. S. Sekhon and P. K. Gupta, "A Study of Lateral Collapse of Square and Rectangular Metallic Tubes", Thin-Walled Structures, Vol. 39, No. 9, 2005, pp 895–922.

50. N. K. Gupta, G. S. Sekhon and P. K. Gupta, "Study of Lateral Compression of Round Metallic Tubes", Thin-Walled Structures, Vol. 43, No. 6, 2005, pp 895–922.

51. N. K. Gupta, P. K. Gupta and G.S. Sekhon, "Investigation of Lateral Compression of Metallic Tubes", Proceedings of 44th Indian Society for Theoretical and Applied Mechanics Congress, Sivakashi, INDIA, pp 121-128.

52. P. K. Gupta, N. K. Gupta and G. S. Sekhon, "Experimental and Computational Investigation of Lateral Compression of Round Metallic Tubes", Proceedings of International Conference on Mathematical Modeling of Non-Linear Systems, Indian Institute of Technology, Kharagpur, INDIA, 2000, pp 251-258.

53. N. K. Gupta, G. S. Sekhon and P. K. Gupta, "A Study of Fold Formation in Axisymmetric Axial Collapse of Round Tubes", International Journal of Impact Engineering, Vol. 27, No. 1, 2002, pp 87–117.

54. P. K. Gupta, N. K. Gupta and G. S. Sekhon, "A Study of Axial Collapse of Round Tubes" Proceedings of 11th Indian Society of Mechanical Engineers, Indian Institute of Technology, Delhi, INDIA, 1999, pp 416-424.

55. G. S. Sekhon, N. K. Gupta and P. K. Gupta, "Study of Axi-symmetric Multiple Barrelling Mode of Collapse of A Round Tube Subjected to Axial Compression", Proceedings of 12th Indian Society of Mechanical Engineers Conference, Crescent Engineering College, Chennai, INDIA, 2001, pp 208-216.

56. N. K. Gupta and Nagesh, "Experimental and Numerical Studies of The Collapse of Thin Tubes Under Axial Compression", Latin American Journal of Solids and Structures, Vol. 1, 2004, pp 233-260.

57. G. S. Sekhon, N. K. Gupta and P. K. Gupta, "An Analysis of External Inversion of Round Tubes", Journal of Material Processing Technology, Vol. 133, No. 3, 2003, pp 243-256.

58. P. K. Gupta, G. S. Sekhon and N. K. Gupta, "Finite Element Simulation of External Inversion of A Round Tube", Proceedings of International Conference on Advances in Civil Engineering, ACE2002, Indian Institute of Technology, Kharagpur, INDIA, 2002, pp 1290-1298.

59. Z. Sun and H. Yang, "Development of Finite Element Simulation System for The Tube Axial Compressive Precision Forming Process", International Journal of Machine Tool and Manufacture, Vol. 42, 2002, pp 15-20.
60. S. R. Reid, "Plastic Deformation Mechanisms In Axially Compressed Metal Tubes Used As Impact Energy Absorbers" International Journal of Mechanical Sciences, Vol. 35, No. 12, 1993, pp 1035-1052.
61. S. R. Guillow, G. Lu and R. H. Grzebieta, "Quasi-static Axial Compression of Thin-Walled Circular Aluminium Tubes", International Journal of Mechanical Sciences, Vol. 43, No. 9, 2001, pp 2103-2123.
62. S. J. Hosseinipour and G. H. Daneshi, "Energy Absorption and Mean Crushing Load of Thin-Walled Grooved Tubes Under Axial Compression", Journal Thin Walled Structures, Vol. 41, 2003, pp 31-46.
63. S. Kobayashi, S. I. Oh, and T. Altan, "Metal forming and The finite-element method," first edition, Oxford University Press, New York, 1989.
64. C. H. Lee and S. Kobayashi, "New Solutions to Rigid-Plastic Deformation Problems Using Matrix Method", ASME Journal of Engineering for Industry, Vol. 95, No. 2, 1973, pp 865-873.
65. C. H. Lee and S. Kobayashi, "Analyses of Axisymmetric Upsetting and Plane-Strain Side-Pressing of Solid Cylinders by The Finite Element Method", ASME Journal of Engineering for Industry, Vol. 93, No. 2, 1971, pp 445-454.
66. S. Kobayashi, "Deformation Characteristics and Ductile Fracture of 1040 Steel In Simple Upsetting of Solid Cylinders and Rings", ASME Journal of Engineering for Industry, Vol. 92, No. 2, 1970, pp 391-399.
67. S. B. Petersen, P. A. F. Martins and N. Bay, "Friction in Bulk Metal Forming: A General Friction Model Vs. The Law of Constant Friction" Journal of Materials Processing Technology, Vol. 66, 1997 pp 186-194.
68. N. K. Gupta and C. B. Shah, "Barreling of A Short Cylinder In Compression", International Journal of Mechanical Tool Design Research, Vol. 26, No. 2, 1986, pp 137-146.
69. D. Singh, "Automatic Mesh Generation, Error Estimation and Adaptivity in Finite Element Analysis of Metal Forming Process", Ph. D. Thesis, Indian Institute of Technology, Delhi, India, 1998.
70. http://www.transvalor.com, Dec. 2005.
71. http://www.ansys.com, Dec. 2005.

72. S. Y. Kim and Y. T. Im, "Parallel Processing of 3D Rigid Viscoplastic Finite Element Analysis using Domain Decomposition and Modified Block Jacobi Preconditioned Technique", Journal of Materials Processing Technology, Vol. 134, 2003, pp 254–264.

73. J. S. Cheon, S. Y. Kim and Y. T. Im, "Application of Parallel Processing in Three-Dimensional Bulk Forming Finite Element Analysis", Journal of Materials Processing Technology, Vol. 152, 2004, pp 106–115.

74. T. Sterling, "An Introduction to PC Clusters For Higher Performance Computing", The International Journal of High Performance Computing Applications, Vol. 15. No. 2, 2001, pp 92-101.

75. Chun-Ho Liu, Chat-Ming Woo and Dennis Y. C. Leung, "Performance Analysis of Linux PC Cluster Using A Direct Numerical Simulation of Fluid Turbulence Code", The International Journal of High Performance Computing Applications, Vol. 19, No. 4, 2005, pp 265-374.

76. J. S. Cheon, S. Y. Kim and Y. T. Im, "Three-Dimensional Bulk Metal Forming Simulations Under A PC Cluster Environment", Journal of Materials Processing Technology, Vol. 140, 2003, pp 36–42.

77. K. Washizu, "Variational Methods In Elasticity and Plasticity", Pergamon Press, Ozford, 1968.

78. C. C. Chen, S. I. Oh and S. Kobayashi, "Ductile Fracture In Axisymmetric Extrusion and Drawing, Part 1: Deformation Mechanics of Extrusion and Drawing", ASME Journal of Engineering for Industry, Vol. 101, No. 2, 1979, pp 23-35.

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

80. S. I. Oh, "Finite Element Analysis of Metal Forming Problems with Arbitrarily Shaped Dies", International Journal of Mechanical Sciences, Vol. 24, No. 8, 1982, pp 479-493.

81. J. F. Lyness, D. R. J. Owen and O. C. Zienkiewicz, "Finite Element Analysis of Steady Flow of Non-Newtonian Fluid Through Parallel Sided Conduits", Proceedings of International Symposium on Finite Element Method in Flow Problems, Swansea, pp 489-501.

82. C. C. Chen and S. Kobayashi, "Rigid-Plastic Finite Element Method Analysis of Ring Compression", ASME Journal of Application of Numerical Method to Forming Process, Vol. 28, 1978, pp 163-172.

LIST OF PUBLICATIONS

DETAILS OF PUBLISHED PAPERS

1. P. K. Gupta and R. N. Khapre, "An Efficient Parallel Solver Using Matrix Inversion Method For Linear and Non-Linear Finite Element Analysis", Journal of Institution of Engineers, INDIA, 2005.
2. R. N. Khapre and P. K. Gupta, "Simulation of Lateral Compression of Rectangular Tubes on Supercomputer PARAM 10000", National Conference on Advances In Mechanical Engineering, AIME, 2006.
3. P. K. Gupta and R. N. Khapre, "An Application of Cluster Computing for Finite Element Analysis", International Conference on Structural Engineering Convention, IISc Bangalore, 2005.
4. P. K. Gupta, R. N. Khapre and A. A. Sinhal, "Analysis of Spalling Zone in Prestressed Post-Tensioned Concrete Beam using Supercomputer PARAM 10000", National Conference on Structural Engineering and Mechanics, BITS, Pilani, 2004.
5. P. K. Gupta, J. P. Mishra, R. N. Khapre, and P. K. Jain, "Comparison of C and FORTRAN 77 Languages Based on Their Performance On PARAM 10000", National Conference on Distributed Computing, NITTE, Karkala, 2004.
6. P. K. Gupta and R. N. Khapre, "Finite Element Analysis of Anchorage Zone Using Supercomputer PARAM 10000", International Conference on Structural Engineering Conventions, IIT, Kharagpur, 2003.
7. P. K. Gupta and R. N. Khapre, "Comparative Study of Solution Methods of System of Linear Equations On Supercomputers", International Conference on Structural Engineering Conventions, IIT, Kharagpur, 2003.
8. P. K. Gupta and R. N. Khapre, "Study of Anchorage Zone in Post-Tensioned Concrete Beams Using Finite Element Method", National Conference on Advances in Civil Engineering Perspective of Developing Countries, HBTI, Kanpur, 2003.

DETAILS OF ACCEPTED PAPERS

1. R. N. Khapre and P. K. Gupta, "Simulation of Axial Compression of Round Metallic Tube on Supercomputer PARAM 10000", 14-National Seminar on Aerospace Structures, NASAS, VNIT, Nagpur, 2006

DETAILS OF COMMUNICATED PAPERS

1. P. K. Gupta and R. N. Khapre, "Parallel Solvers for Solution of System of Linear Equations", International Journal of High Speed Computing, 2005.
2. P. K. Gupta and R. N. Khapre, "Finite Element Analysis of Metal Forming Problems Using Parallel Computing Technique", International Journal of Computational Methods in Engineering Science and Mechanics, 2005.
3. R. N. Khapre and P. K. Gupta, "Finite Element Analysis of Anchorage Zone Using Supercomputer", Journal of Institution of Engineers INDIA, 2005.
4. P. K. Gupta and R. N. Khapre, "Application of Cluster Computing In Finite Element Programming to Study The Large Deformation Problems", International Journal of Computational Methods, 2005.
5. R. N. Khapre and P. K. Gupta, "Simulation of Lateral Compression of Round Metallic Tube Between Two Concentrated Loads on Supercomputer PARAM 10000", 10th East Asia-Pacific Conference on Structural Engineering and Construction, Asian Institute of Technology, Bangkok, 2006.

BIOGRAPHY

P. K Gupta at present is Assistant Professor in Civil Engineering Group at Birla Institute of Technology and Science, Pilani (Raj.), INDIA. He has completed his Ph.D. from IIT, Delhi on large deformation behavior of metallic tubes. He has been actively involved in teaching, research, and consultancy works during last 8 years. He has published a number of papers in International, National Journals and Conferences.

Rajendra N. Khapre at present is Research Scholar in Civil Engineering Group at Birla Institute of Technology and Science, Pilani (Raj.), India. He has completed his masters in Civil Engineering from BITS, Pilani. He is pursuing Ph.D. from BITS, Pilani on parallel computing in finite element analysis. He has been actively involved in teaching and research work during last 4 years. He has published several papers in National Journals and Conferences.