

PARAMETRIC ANALYSIS OF
DOUBLY CURVED SHELL STRUCTURES
USING
FINITE ELEMENT METHOD

A DISSERTATION

SUBMITTED TO

JAWAHARLAL NEHRU UNIVERSITY, NEW DELHI
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

MASTER OF TECHNOLOGY

IN

STRUCTURAL ENGINEERING

vi, 48 pt Graph, key

BY

MR. V. P. SINGH

GUIDE

DR. S. C. PAL

FACULTY OF CIVIL ENGINEERING
COLLEGE OF MILITARY ENGINEERING
PUNE - 411031
NOVEMBER 1998

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in this dissertation entitled "***A Parametric Analysis of Doubly Curved Shell Structures using Finite Element Method***" in partial fulfillment of the requirements for the award of the degree of Master of Technology in Structural Engineering to Jawaharlal Nehru University, is an authentic record of my own work, under the supervision and guidance of Dr. S.C. Pal of Civil Engineering Department, College of Military Engineering, Pune.

I have not submitted the matter embodied in this dissertation to any other University or Institute for the award of any degree or diploma.

Dated: November 1998

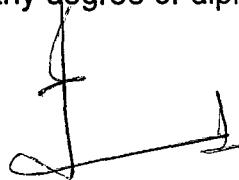
V. P. Singh
(V. P. Singh)

IDSE

CERTIFICATE

This is to certify that the dissertation entitled "***A Parametric Analysis of Doubly Curved Shell Structures using Finite Element Method***" that is being submitted by Mr. V. P. Singh for the award of the degree of Master of Technology in Structural Engineering to the Jawaharlal Nehru University, New Delhi, is a bonafide study carried out by him under my supervision and guidance, hence recommended for acceptance and approval.

The study carried out in this dissertation has not been submitted to any other University or Institute for the award of any degree or diploma.



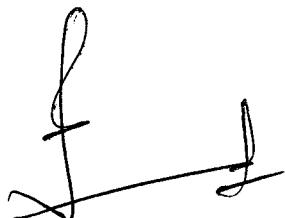
(Dr. S. C. Pal)

Guide

Faculty of Civil Engineering
College of Military Engineering
Pune – 411031

EXAMINER'S CERTIFICATE OF APPROVAL

The dissertation entitled "***A Parametric Analysis of Doubly Curved Shell Structures using Finite Element Method***" submitted by Mr. V. P. Singh, in partial fulfilment of the requirements of the degree of Master of Technology in Structural Engineering, to the Jawaharlal Nehru University, New Delhi, is hereby approved for the award of the degree.



Guide

(Dr. S. C. Pal)

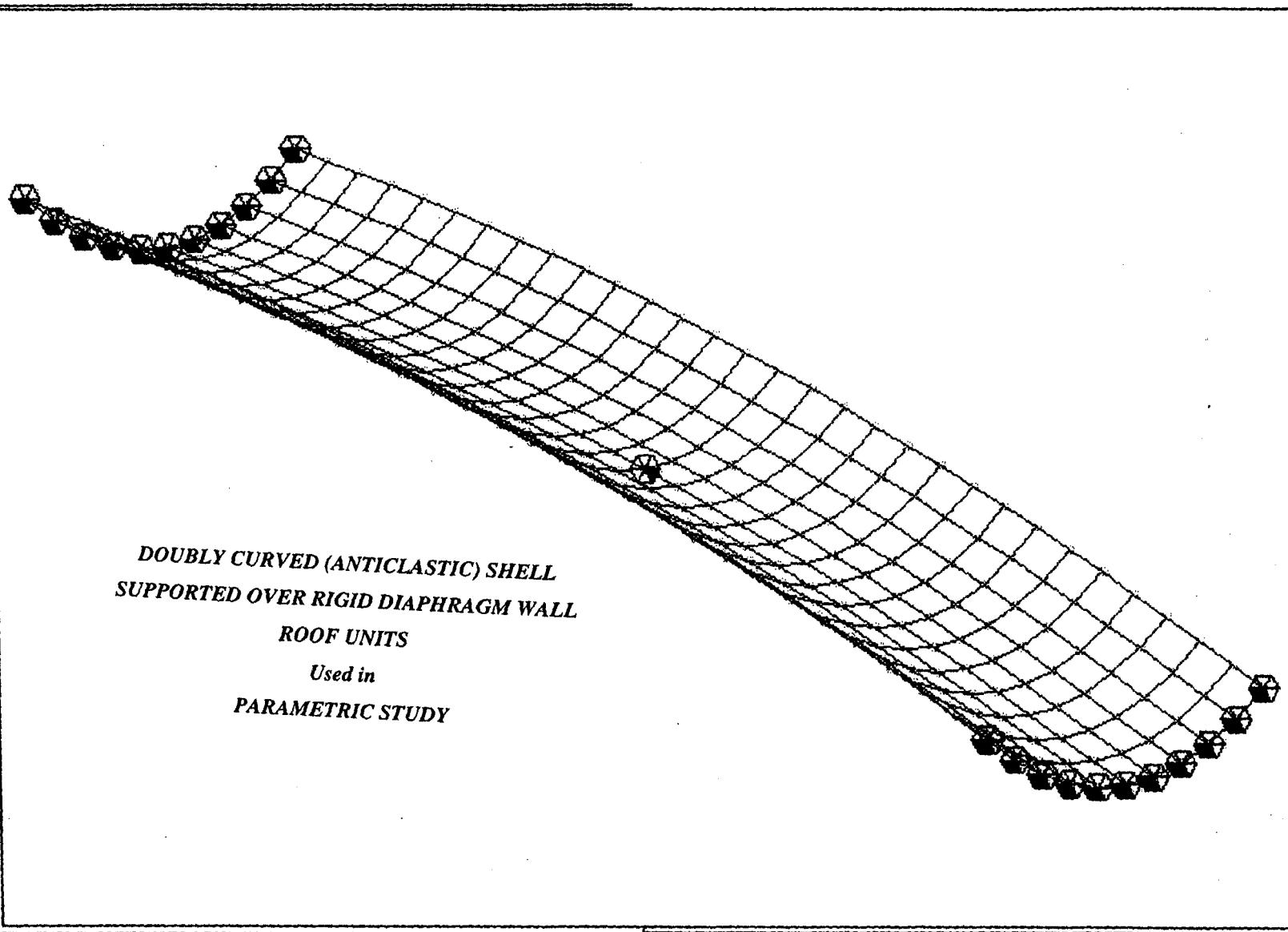
Guide

Faculty of Civil Engineering
College of Military Engineering
Pune – 411031



Examiner





**DOUBLY CURVED (ANTICLASTIC) SHELL
SUPPORTED OVER RIGID DIAPHRAGM WALL**

ROOF UNITS

Used in

PARAMETRIC STUDY

SYNOPSIS

The analysis and design of shell structures is a topic of interest in a variety of engineering disciplines. But the analysis of such structures has been difficult and laborious task before the advent of digital computer followed by speedy progress of finite element method.

The economy and elegance in the use of shells as roofs was the motivation behind the development of the different shell forms and their use. Though the shells are material wise very economical but their construction to precise geometry is difficult and costly. So if prefabricated units of shell can be used to cover rectangular plan, it should reduce the cost of roofing considerably.

- *The parametric analysis of doubly curved (anticlastic) shell units of roof using Finite Element Method* was chosen with dual purpose of understanding the procedures in finite element method and carrying out a part of optimization study of such prefabricated units.

To achieve the aims mentioned above a *four noded bilinear shell element for the analysis of shells as a special case of three-dimensional analysis* was mathematically formulated. On the basis of this formulation a computer code in 'C' language was developed for the element stiffness matrix and integrated with the computer code for *Frontal Solution Technique for Linear Simultaneous Equations* (developed as seminar topic). This code was used to solve a test problem of cylindrical barrel vault, whose solution by Scordelis is given in reference 12. Results in the form of graphs are in excellent agreement with the theoretical one except for twisting moment. But this also closely follows the pattern of exact result but with slightly lower values.

With this computer code proved for its efficacy parametric study was carried out. For the purpose of study a part of practical problem of optimization of prefabricated doubly curved (anticlastic) shell units made of ferrocement for roof *supported over rigid diaphragm walls at the ends*, as idealized by well-known architect Mr. Doshi, was chosen. There were three different profiles of the shell units, and each of them were analyzed for three different thicknesses under *self load* condition. Stress resultants along central transverse and longitudinal sections have been plotted and some maximum values tabulated for comparison and further study.

Two extreme cases of parametric study units were analyzed by *NISA* package of *EMRC* also.

ACKNOWLEDGEMENTS

I take this opportunity to express my heartfelt gratitude and sincere thanks to Dr. S.C. Pal who has guided me through every stage of this work. I want to especially mention his contribution in making me obsessed with the idea of generality of Finite Element Method and power of Frontal Solution Technique to this end.

I am grateful to Mr. A. K. Garg with whom I used to discuss problems during programming in 'C' language, to Capt. Rajiv Gupta for initiating me in Excel and also to Mr. Prasad of Vision Computer Academy, Kirkee, who helped me in giving this shape to the thesis. I am also grateful to Mr. Shailendra Pramod Joshi who helped me in using NISA package of EMRC. I am deeply thankful to Computer Center of CME for all facilities it provided.


V. P. Singh

CONTENTS

SYNOPSIS

ACKNOWLEDGEMENTS

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF GRAPHS

CHAPTER	TOPIC	PAGE NO.
1.	INTRODUCTION	1
	1.1 Preliminary Remarks	1
	1.2 Motivation	2
	1.3 Work done	2
	1.4 Layout of Thesis	3
	1.5 Historical Perspective	4
2.	THEORETICAL ASPECTS	11
	2.1 Classification of Shell Systems	11
	2.2 Assumptions in Shell Analysis	11
	2.3 Shell Elements	13
3.	MATHEMATICAL FORMULATION OF BILINEAR DEGENERATE SHELL ELEMENT	18
	3.1 Shape Function for Geometry and Displacements.	18
	3.2 Local Coordinates	19
	3.3. Jacobian	21
	3.4. Displacements	22
	3.5. Degrees of Freedom	22
	3.6. Strain Displacement Matrix	23
	3.7. Stress Displacement Matrix	32
	3.8. Element Stiffness Matrix	33
	3.9. Torsional Stiffness	35
	3.10. Calculation of Stress Resultants	37

4.	COMPUTER IMPLEMENTATION	41
4.1	Input data	41
4.2	Shape Function and their Derivatives	42
4.3	Calculation of Constitutive Matrix	43
4.4	Calculation of Direction Cosine Matrix	43
4.5	Jacobian	44
4.6	Strain Displacement Matrix Units	44
4.7	Calculation of Stiffness Matrix	45
4.8	Assembly and Solution	46
4.9	Subroutines	46
4.10	Files and Variables	49
5.	VERIFICATION OF FORMULATION OF FOUR NODED DEGENERATE BILINEAR ELEMENT.	53
5.1	The Test Problem	53
5.2	Solution by the Element	53
5.3	Boundary Conditions	53
6.	PARAMETRIC ANALYSIS OF DOUBLY CURVED (ANTICLASTIC) SHELL	60
6.1	Problem Description	60
6.2	Results	62
APPENDIX A	NISA PACKAGE RESULTS	
APPENDIX B	COMPUTER CODE IN 'C' LANGUAGE	
	REFERENCES	

LIST OF TABLES

TABLE NO.	DESCRIPTION	PAGE NO.
6.1	Maximum Vertical Deflection At Centre	63
6.2	Maximum Inplane Longitudinal Force	63
6.3	Maximum Inplane Longitudinal Stress	63
6.4	Maximum Transverse Moment	64
6.5	Maximum Longitudinal Moment	64
6.6	Maximum Twisting Moment	64

LIST OF FIGURES

FIGURE NO.	DESCRIPTION	PAGE NO.
1.1 to 1.8	Different forms of Shells	9 – 10
2.1	Definition of Curvature	16
2.2	Curved, Isoparametric Hexahedral Elements	16
2.3	Stresses in Shell Elements	17
2.4	Rotation of Normal during Deformation	17
3.1	Four Noded Shell Element	39
3.2	Torsional Rotation	39
3.3	Internal Stress Resultants	40
5.1	The Problem – Barrel Vault Displacements	54
5.2	The Problem – Moments in Barrel Vaults	55

LIST OF GRAPHS

GRAPH NO.	DESCRIPTION	PAGE NO.
5.1	Vertical deflection at Centre of cylindrical barrel vault	56
5.2	Longitudinal displacement at the end of cylindrical barrel vault	57
5.3	Moments at the Centre of cylindrical barrel vault	58
5.4	Twisting moment at the end of cylindrical barrel vault	59
6.1 to 6.3	Vertical Deflection at Centre (6m, 5m, 4m)	65 – 67
6.4 to 6.6	Longitudinal Displacement at end. (6m, 5m, 4m)	68 – 70
6.7 to 6.9	Inplane Longitudinal Force at Centre (6m, 5m, 4m).	71 – 73
6.10 to 6.12	Transverse Moment at Centre(6m, 5m, 4m)	74 – 76
6.13 to 6.15	Longitudinal Moment at Center (6m, 5m, 4m)	77 – 79
6.16 to 6.18	Twisting Moment at End (6m, 5m, 4m)	80 – 82
6.19 to 6.21	Central Vertical Deflection along the length (6m, 5m, 4m)	83 – 85
6.22 to 6.24	Central Inplane Longitudinal Force along the length (6m, 5m, 4m)	86 – 88
6.25 to 6.27	Central Transverse Moment along the length (6m, 5m, 4m)	89 – 91
6.28 to 6.30	Central Longitudinal Moment along the length (6m, 5m, 4m)	92 – 94
6.31	Inplane Longitudinal Force along the length (6m, 60mm)	95
6.32	Transverse Moment along the length (6m, 60mm)	96
6.33	Longitudinal Moment along the length (6m, 60mm)	97
A.1	Variation of z- displacement at end support (4m, 60mm)	98
A.2	Variation of central vertical displacement along the length (4m, 60mm)	99
A.3	Variation of M_{xy} along the transverse section (4m, 60mm)	100
A.4	Variation of y- displacement along length (6m, 20mm)	101
A.5	Variation of z- displacement (6m, 20mm)	102
A.6	Variation of M_{xy} along end transverse section (6m, 20mm)	103
A.7	Variation of M_{yy} along central transverse section (6m, 20mm)	104
A.8	Variation of M_{xx} along central transverse section (6m, 20mm)	105

Chapter 1

INTRODUCTION

1.1 Preliminary Remarks

The analysis and design of shell structures is a topic of interest in a variety of engineering disciplines. The civil engineer is concerned with the design of large span roofs, liquid storage facilities, silos and many other structures. The mechanical engineer is interested in the design of pressure vessels, including nuclear reactor containment vessels and pipes. The aeronautical engineer is involved in the structural design of aircrafts, rockets and aerospace vehicles. All of these structures require the analysis and design of shells in one form or other. But the most complicated task involved is analysis of these structures i.e., shells.

Flugge defines shell as an object, which for the purpose of stress analysis, may be considered as the *materialization of a curved surface*. This definition implies that the thickness of a shell is small compared with its other dimensions but it does not require that the smallness be extreme.

In early days shells were analyzed by the *membrane theory*, i.e. the theory of shells whose bending rigidity may be neglected. The spectacular simplifications thus obtained makes it possible to examine a wide variety of shapes and support conditions. There is, of course, a heavy penalty to be paid for the simplification, in terms of accuracy in prediction of stress resultants.

A more elaborate and complicated *bending theory* of shell is required to overcome these inaccuracies of membrane theory. But application of bending theory to different types of shells results in different forms of differential equations. The basic reason for unpopularity of bending theory is extremely difficult solution of these differential equations and even that for a particular boundary condition.

With the advent of digital computers, earlier the mainframe and now increasingly available PCs, the method of analysis of shells have undergone a revolutionary change. Numerical methods of analysis replaced the closed

form solution of differential equation. Finite difference method has given way to a general finite element method. In early sixties this method was applied for the solution of plates and shells and thereafter gradually came a number of elements for general shell analysis which can be broadly classified under the categories of flat element, axisymmetric element, curved element, a special case of three dimension analysis element.

1.2 Motivations

The present study of *Parametric Analysis of doubly curved (anticlastic) shell structures using Finite Element Method* is an effort in the direction of understanding the method as applicable to shells, as in finite element method there is no by pass to frontiers.

For the purpose of *study* a part of practical problem of optimization of prefabricated ferrocement doubly curved (anticlastic) shell units, as idealized by well-known architect Mr. Doshi, was chosen. There were three different profiles of the shell units, and these were analyzed for three different thicknesses under self-load conditions.

So in total there became nine cases of shell roof units, which were then idealized for analysis by finite element method using bilinear four noded degenerate three-dimensional element.

1.3 Work Done

So, in the study, the tasks involved are

- a) Mathematical formulation for a general shell element.
- b) Development of computer code for the formulation.
- c) Integrating the code with earlier developed computer code for *frontal solution technique for linear simultaneous equation* as applicable for equations arising in static analysis of structures (developed as seminar topic).
- d) Checking the formulation for its efficacy against a test problem whose theoretical solution is available.
- e) With this computer code, proved for its efficacy, parametric study of the above mentioned problem was carried out.

1.4 Layout of Thesis

First chapter gives introduction, motivation, work done and historical development in brief, of different modern shell forms and their pioneers.

Second chapter deals with theoretical aspects of shells. First, in this chapter shells are classified on the basis of gaussian curvature, the way of formation and their thickness. Then, membrane theory of shells for elementary analysis, the basics of Kirchoff's theory and Reissner-Mindlin theory as they are used in Finite Element Method has been discussed. In the last, broad classes of finite elements available for general shell analysis are given with their relative merits.

In the third chapter detailed mathematical formulation required for developing computer code is given. There are three basic relations in this chapter. First is shape function, second is relation between global coordinates and natural coordinates with the help of direction Cosines of normal vector and the third is displacements (u, v, w) in global axes direction in terms of shape function in natural coordinate system, displacements corresponding to nodal degrees of freedoms of the elements and direction cosines of normal vector. Then discussed is the generation of direction cosine matrix of local axes w.r.t. global axes. Jacobian, relating derivatives in one reference axes to another are also elaborated here. Now with the help these five entities, strain displacement matrix has been derived. Integration in thickness direction has been done analytically. 3D constitutive matrix has been modified for condition $\sigma_z = 0$. Now with known strain displacement matrix $[B]$, stiffness matrix has been derived as,

$$[K] = \int_v [B]^T [C] [B] dv \quad (1.1)$$

Torsional rotation effect has also been accounted for in stiffness matrix and details are given. Numerical value of torsional coefficient has been adopted as **0.5**.

Chapter four deals with the computer implementation of mathematical formulation of chapter three. In its first part basic units have been discussed.

In second subroutines performing specific tasks have been discussed. And in the last a detailed list of variables used in computer code has been given.

Chapter five reproduces a standard test problem of cylindrical barrel vault supported on rigid diaphragms at the ends and its theoretical solution. Results obtained by the computer code have also been given in the form of graphs here to check the efficacy of the formulation. We can see that the results are in excellent agreement with theoretical one except one for twisting moment. But twisting moment also closely follows pattern of exact result but slightly lower values.

In Chapter six the idealized mathematical details of parametric study are given in detail. Then the analysis results in the form of tables and graphs for deflection, displacements, longitudinal in plane force, transverse moment, longitudinal moment and twisting moments of central transverse section and central longitudinal section are given. These results for one of the geometries along the longitudinal edge are also given. Two extreme cases involved in our parametric study i.e. shortest span with maximum thickness and largest span with minimum thickness were also analyzed on **NISA** Package of EMRC and graphs are given in Appendix A.

Computer code in C Language is given in Appendix B.

1.5 Historical Perspective

Engineers interested in the design of shell structure must have an understanding of how shell can be analyzed to show safe behavior under loads, can be planned for economical construction and can be shaped for attractive appearance. The two types of knowledge that inform all of the best structural designs are scientific and historical.

Scientific knowledge means knowing how to define structural behaviour, expressed in displacements and forces, to the end of discovering how a given form carries loads safely with a minimum of material.

But choice of a form precedes application of scientific knowledge for which the historical knowledge is required.

For engineering structures, historical knowledge means two kind of study. First, studying what has been accomplished in the past to the end of choosing forms that reflect successful experience and avoiding those that have been found defective. Second, studying how successful designers in the past proceeded in the light of limited scientific knowledge, cost restrictions and aesthetic ideals. The major shell forms, in approximate chronological order of development, are cylindrical shell walls as fluid containers, domes, barrel shells, folded plates, hyperbolic paraboloids , elliptic paraboloids and non geometrical shells.

Simplified analysis is based upon considering the thin shell to be defined by its middle surface, shown in Fig. 1.1 as halfway between the inner and outer edges of shell. Furthermore, the internal forces used in shell analysis are defined as stress

$$N_\theta dy = dy \int_{-h/2}^{+h/2} \sigma_\theta dz \quad (1.2)$$

resultants (force per unit length of middle surface) such as

And stress couple (bending moments per unit length of middle surface)

$$M_y rd\theta = rd\theta \int_{-h/2}^{+h/2} \sigma_y z \left(1 - \frac{z}{r}\right) dz \quad (1.3)$$

Positive N_θ is tension and positive M_y is where σ_y is tension (positive) in the positive z direction. The term z/r in above equation arises because the cross section is trapezoidal. Where it appears with unity the term z/r can be neglected. Stresses are found simply by dividing N_θ by h or dividing M_y by $h^2/6$. ***Assumption that all M = 0 leads to the membrane theory of shell.***

1.5.1 Cylindrical Shell Walls

Cylindrical shells as water tanks and gas holders were among the first types of structures built of reinforced concrete. G. A. Wayss described the membrane theory in-plane ring tensions for these in 1887. Robert Maillart made the first studies which considered bending as well as membrane

stresses in fixed base cylinders. The mathematical theory was applied first in its modern form by H. Reissner in 1908 and given by von Emperger in 1910.

1.5.2 Domes

Johann Schwedler had presented the membrane theory in 1886 and Wayss had given it in his 1887 text on reinforced concrete. This theory still seemed satisfactory to designers and even today provides a reasonable basis for preliminary design. By 1915 a complete mathematical theory including bending was developed largely by Ernst Meissner (1883-1939). Walter Bauersfeld (1879-1959) first made the design for 11/5 – in. thick dome with diameter of 52 ft in Jena, East Germany. Geckeler's studies publish in 1926 clarified that mathematical theory. Geckeler's basic idea was to separate the analysis into two parts in the first, the shell stresses are found entirely from equilibrium equations (the membrane theory); in the second bending stresses are studied only in those part of the shell near discontinuities i.e., usually just at edges.

1.5.3 Barrel-shell Roofs

But Dischinger recognized quickly that roofs over circular plans were rare and new ideas to shells covering rectangular spaces is required.

Ulrich Finsterwalder (b.1897) saw the problem much as Geckeler had seen the dome in two steps. Membrane Theory of barrel shells was presented by Geckeler and Dischinger in their 1928 treatise and which Bauersfeld had independently developed as well. The major difficulty was the bending theory because now the formulation led to an 8th –order partial differential equation. Huber Rusch brought the essential mathematical basis for barrel shells in a tractable if not too accessible form.

Two forms of shell, responsible for its introduction in USA were dome and barrel and the man most responsible for it was Anton Tedesco (b.1903)

The development in United States from 1932, largely through barrel shells culminated 20 years later in an unusual publication by the American Society of Civil Engineers : Manual 31 in which Finsterwalder's basic approach was

put in tabular form and made accessible to a wide circle of American Engineers.

1.5.4 Folded Plates

In 1929 Craemer described roof of folded plate, he also showed examples of folded plates used for coal silos. A summary of the methods of analysis appeared in 1963. By then more general computer methods have become common.

1.5.5 The Hyperbolic Paraboloid

It came after dome and barrel. Here the pioneering was by French, Italian and Spanish designers and began with the recognition of those special geometric properties which made the surface easy to analyze and appealing to designer's visualization of construction. Both factors rely upon the straight line generators whose geometry is defined by

$$z = \frac{y^2}{h_2} - \frac{x^2}{h_1} \quad (1.4)$$

Here h_1 and h_2 are equal to twice the principal radii of curvature of the surface at the origin $x=y=0$.

The surface can be generated either by parabolas as in previous equation or by straight lines.

Probably Ferdinand Aimond gave the earliest presentation of this geometry for roof shells in 1933. His membrane theory was basis of design for next three decade.

The major impetus to design came from Felix Candela (b.1910) After Candela, the principal force in popularizing these shells was the Portland Cement Association and especially Alfred Parme. The main idea was that these structures behave as two system of arches, one in compression and one in tension.

1.5.6 Gabled Hyperbolic Paraboloidal Roofs

The other hyperbolic paraboloid is the four quadrant gable shown in Fig. 1.7

1.5.7 Groined Vault

Fig. 1.8 shows a simple groined vault made up of two intersecting segments of paraboloids.

1.5.8 The Elliptic Paraboloid

Fig. 1.6 shows an elliptic paraboloid whose form has the equation.

$$z = \frac{c_1 x^2}{a^2} + \frac{c_2 y^2}{b^2} \quad (1.5)$$

Parme presented in 1958 the membrane theory of these shells.

In 1955 Swiss Heinz Isler presented his idea of pneumatic forms obtained by inflating a rectangular plan rubber membrane clamped along its four horizontal edges. The upward shape was then in pure tension under upward pressure so downward pressure would put the same shape (in concrete) into pure compression. His another idea was hanging membrane reversed.

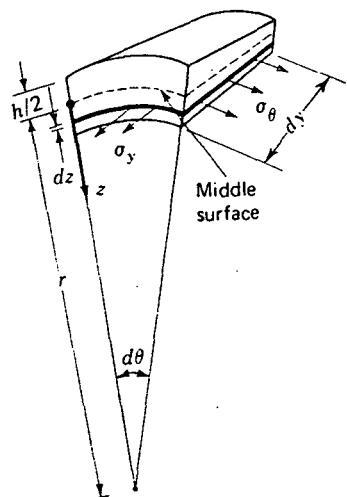


Fig 1.1

Definitions for thin shells.

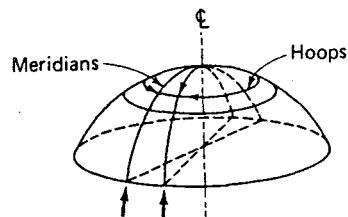
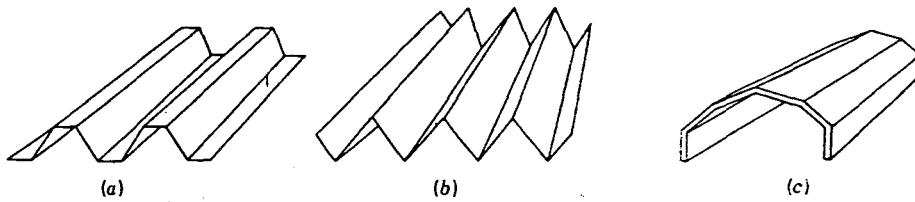


Fig 1.2

Definitions of domes.



Typical folded-plate cross sections.

Fig 1.3

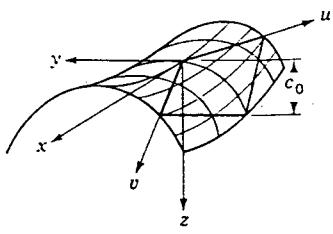


Fig 1.4

Hyperbolic paraboloids with curved boundaries.

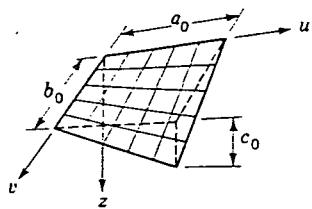


Fig 1.5

Hyperbolic paraboloids with straight boundaries.

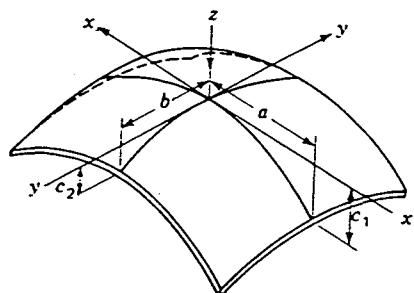
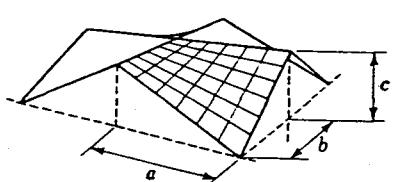
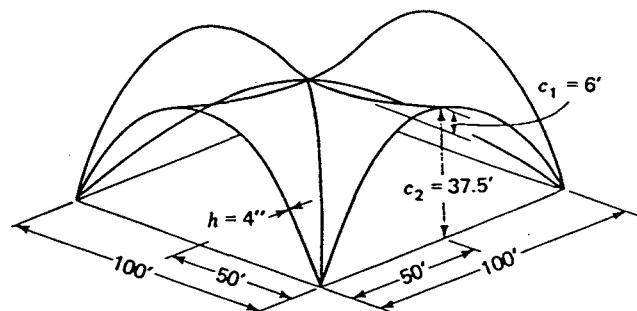


Fig 1.6

Elliptic paraboloid.



Gabled hyperbolic paraboloid.



Rectangular-plan hyperbolic groined vault.

Fig 1.7

Fig 1.8

Chapter 2

THEORETICAL ASPECTS

2.1 CLASSIFICATIONS OF SHELL SYSTEMS

2.1.1 The most general classification of thin shells is by Gaussian Curvature, as illustrated in Fig. 2.1

- [a] Shells of positive Gaussian Curvature, sometimes called *synclastic* shells, are formed by two families of curves both with the same direction. Spherical domes and elliptic paraboloids are examples.
- [b] Shells of zero Gaussian Curvature or singly curved shells are formed by one family of curves only. Some examples are cylinders and cones.
- [c] Shells of Negative Gaussian Curvature, sometimes called *anticlastic*, are formed by two families of curves each in opposite directions. Hyperbolic paraboloids and hyperbolas of revolution are examples.

2.1.2 Another classification divides shells into rotational and translational systems. Domes and tanks are normally surfaces of rotation whereas cylindrical barrels elliptic paraboloids and hyperbolic paraboloids are surfaces of translation.

2.1.3 According to the assumptions made for analysis, shells are classified as thick and thin. If thickness is more and shear deformations are not negligible and accounted for as per Reissner-Mindlin theory, it is thick shell. And if thickness is less and shear deformations are negligible and Kirchhoff's theory is applicable, it is thin shell.

2.2 ASSUMPTIONS IN SHELL ANALYSIS

Although the assumptions regarding transverse distribution of strains and stresses of plates are again valid, the way shell support external load is quite different from that of flat plate. The stress resultants acting parallel to middle plane of the shell now have components normal to (x-sectional) surface and carry a major part of the load. Preliminary analysis can be done by membrane theory.

2.2.1 Membrane Theory

The membrane theory rest on the main assumption that the shell carries loads solely by in-plane stresses and therefore all bending moments and out-of-plane shearing forces are taken as zero.

From this main idea there follows three significant consequences :

1. The state of stress (membrane stress resultants) in the shell is completely determined by equations of equilibrium, i.e., the shell is statically determinate.
2. The boundary conditions must provide for those shell edge forces which are computed from the equations of equilibrium.
3. The boundary conditions must also permit those shell edge displacements (translation and rotations) which are computed from the forces found by the membrane theory.

Thus the membrane theory needs three equations of equilibrium plus expressions for displacements in terms of the membrane stress resultants.

A general theory of thin shells is overly complex and normally general shell analysis is done by numerical method which do not necessarily use such general theory. Numerical methods are generally based on following assumptions.

2.2.2 Theory of Plate Bending and Shell Element

The basic proposition in plate bending and shell analysis is that the structure is thin in one dimension.

Assumptions :

1. The stress through the thickness (i.e. perpendicular to the mid surface) of the plate / shell is zero. (Fig. 2.3)
2. Material particle that are originally on straight line perpendicular to mid-surface of the plate / shell remain on a straight line during deformation. (Fig. 2.4)

2.2.2.1 Kirchoff Theory

In this theory shear deformations are neglected and the straight line referred above remain perpendicular to the mid-surface during deformation. (Fig 2.4 $\phi_x = \phi_y = 0$)

2.2.2.2 Reissner-Mindlin Theory

In this theory shear deformation are included and therefore the straight line originally normal to mid-surface in general do not remain perpendicular to the mid-surface during deformation. (Fig 2.4)

2.2.2.3 Mindlin's Assumption

Mindlin's assumption is that the average rotation of the section may be taken as the rotation in which normal remain perpendicular to the mid-surface plus an additional rotation due to transverse shear. (Fig 2.4 (a), (b))

2.3 Shell Elements

Broadly shell elements can be classified as follows :

2.3.1 Flat Element

A shell is, in essence, a structure that can be derived from a thin plate by initially forming middle plane to a singly (or doubly) curved surface.

It is assumed that the behavior of a continuously curved surface can be adequately represented by the behavior of a surface built up of small flat elements. In a shell, the element will be subject, generally, both to bending and inplane forces. For a flat element these cause independent deformations, provided the local deformation are small. In the division of an arbitrary shell into flat element only triangular element can be used.

2.3.2 Axisymmetrical Shell Element

A special case is presented by axisymmetrical shell element.

2.3.3 Curved Shell Element

As an alternative to flat element, curved shell elements could be used. The physical approximation involved in flat element is now avoided at the expense of reintroducing an arbitrariness of various shell theories.

Such shallow curved shell elements, by coupling the effects of membrane and bending strain in the energy expression, are slightly more efficient than flat ones where such coupling occur on the inter element boundary only.

However, for many practical purposes the flat element approximation gives very adequate answers and indeed permits an easy coupling with edge beam and rib members, a facility sometimes not present in curved elements formulation.

2.3.4 Shells as a special case of three dimensional analysis

As an alternative to above two types of elements (a, c) we can use three-dimensional isoparametric element in the analysis of curved shells simply by reducing their dimensions in shell thickness direction. (Fig 2.2)

With a straight forward use of three dimensional concept certain difficulties will be encountered.

First one is due to large stiffness coefficient for relative displacement along edge corresponding to shell thickness which leads to ill conditioning of set of linear equations.

Second factor is of economy. The use of several nodes across the shell thickness ignores the well-known fact that even for thick shells normal to the middle surface remain practically straight after deformations.

Specialized formulation is presented in next chapter overcoming both difficulties. The constraint of straight normals is introduced to improve economy and strain energy corresponding to stresses perpendicular to the middle surface is ignored to improve numerical conditioning.

The constraint thus introduced are that of Reissner – Mindlin's. The omission of constraint associated with the thin plate theory (normals remaining normal to middle plane after deformation) permits the shell to experience shear deformations – an important factor in thick shell situations.

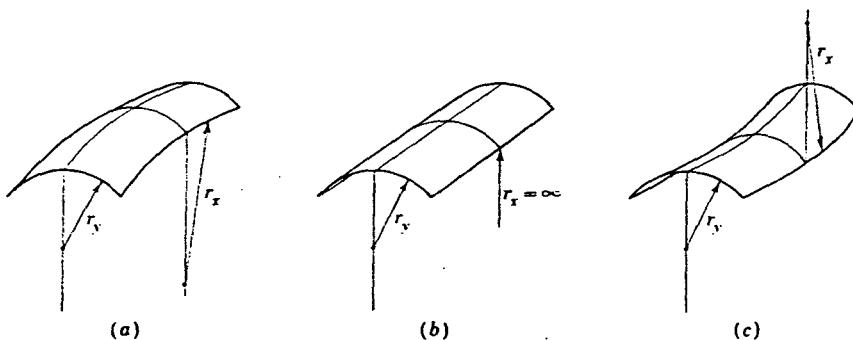
Reduced integration is imperative if thin shells are to be dealt with it to avoid *shear locking* in this pure displacement based formulation.

Fig 2.1

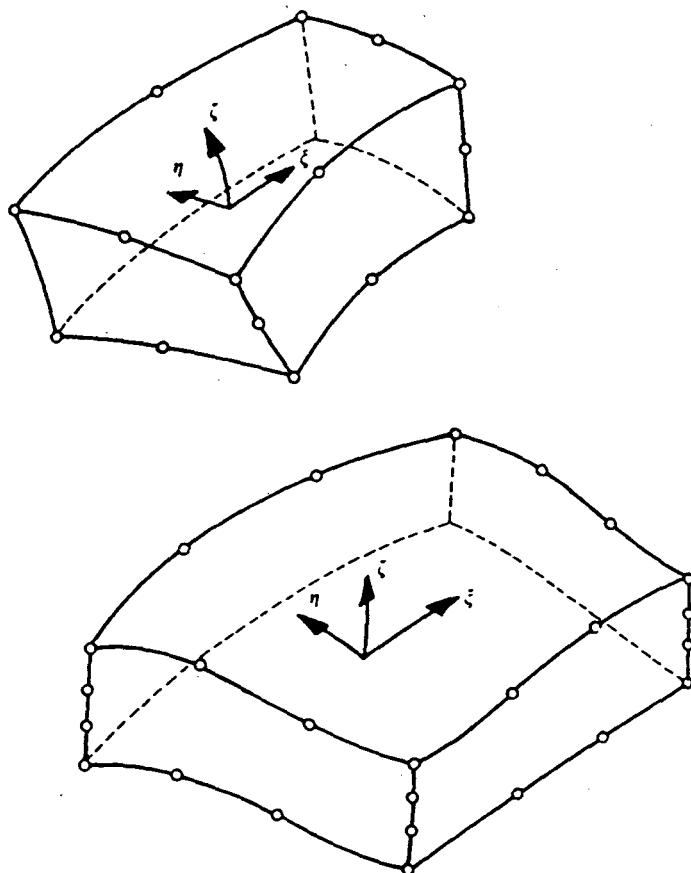
Mathematically the gaussian curvature of a surface is defined as the product of the principal curvatures:

$$K = \frac{1}{r_x} \frac{1}{r_y}$$

where r_x and r_y are the principal radii of curvature.



Definitions of curvature.



Curved, isoparametric hexahedra in a direct approximation to a curved shell

Fig 2.2

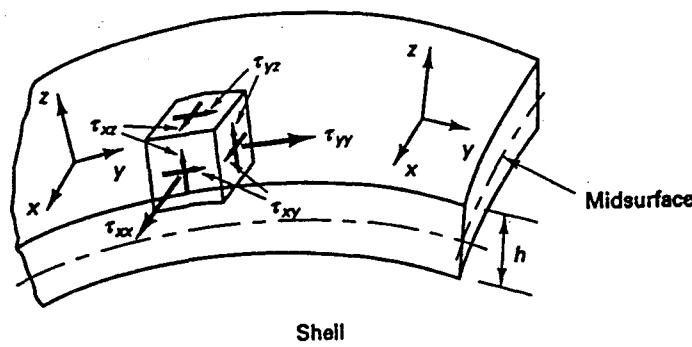


Fig 2.3

$\tau_{zz} = 0$
All other stress components
are nonzero

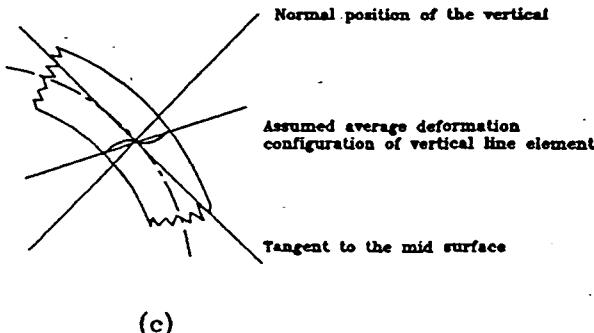
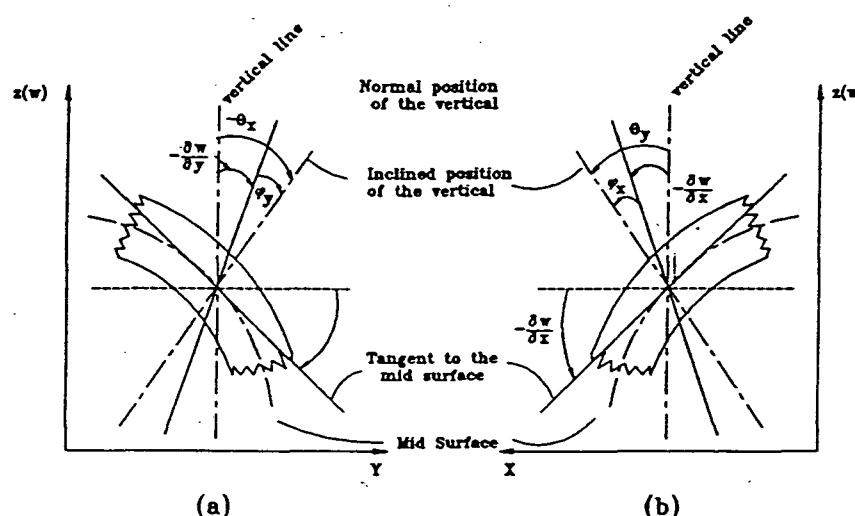


Fig 2.4

Rotation of the normals about x and y axes
considering average shear deformation

Hence, the average shear deformations, ϕ_x and ϕ_y are given by

$$\phi_x = \theta_y + \frac{\partial w}{\partial x}$$

$$\phi_y = -\theta_x + \frac{\partial w}{\partial y}$$

Equations are based on the assumption that the total rotations θ_x and θ_y are small and the transverse strain ϵ_z is negligible.

Chapter 3

MATHEMATICAL FORMULATION OF BILINEAR DEGENERATE SHELL ELEMENT

A simple and efficient four noded shell element based on Reissner – Mindlin's theory (developed by Kanok-Nukulchai on the basis of *shell as a special case of three-dimensional analysis*) as discussed in previous chapter is mathematically formulated in this chapter.

3.1 SHAPE FUNCTIONS FOR GEOMETRY AND DISPLACEMENT

The four noded element is shown Fig.3.1. The midsurface enclosed by four straight sides form a hyperbolic paraboloid. The shape functions are same as used for 2D ISO-P four noded element.

$$N_i = \frac{1}{4} (1 + r_i t) (1 + s_i s) \quad i = 1, 2, 3, 4 \quad (3.1)$$

where, r_i & s_i are the natural coordinates of node i .

The thickness of the shell element in the direction of normal to the midsurface is specified at the node. The coordinates of any point in the element can be uniquely given in terms of nodal coordinates and thicknesses as;

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \sum_{i=1}^4 N_i \left\{ \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} + \frac{1}{2} t H_i \begin{bmatrix} l_{3i} \\ m_{3i} \\ n_{3i} \end{bmatrix} \right\} \quad (3.2)$$

where,

x, y, z are the global coordinates of the point having its natural coordinates as r, s, t .

x_i, y_i, z_i are the global coordinates of the midsurface node i .

H_i is the thickness at the node i and

l_{3i}, m_{3i}, n_{3i} are the components along x, y, z of unit vector normal to midsurface at node i .

Equation 3.2 reduces to following for midsurface, as $t = 0$

$$\begin{Bmatrix} x \\ y \\ z \end{Bmatrix} = \sum_{i=1}^4 N_i \begin{Bmatrix} x_i \\ y_i \\ z_i \end{Bmatrix} \quad (3.3)$$

3.2 Local Coordinates

We identify any point in inside the element by;

- (i) its natural coordinates (r, s, t)
- (ii) its global coordinates (x, y, z)
- (iii) its local coordinates (x', y', z')

First two are straight forward but third one changes from point to point and has to be constructed at every point of reference. In four noded element, it is same for all the points inside an element.

At any point (r, s, t) in an element x' axis is defined as a tangential vector along r axis of curvilinear coordinate axes. z' axis is normal to x' and normal to mid-surface. y' axis is now normal to x' and z' and is tangential to midsurface.

A tangent vector in matrix notation at $(r, s, t = 0)$ along r axis is given by;

$$\begin{Bmatrix} \frac{\partial x}{\partial r} \\ \frac{\partial y}{\partial r} \\ \frac{\partial z}{\partial r} \end{Bmatrix}_{(r,s,t=0)}$$

A tangent vector in matrix notation at $(r, s, t = 0)$ along s axis is given by;

$$\begin{Bmatrix} \frac{\partial x}{\partial s} \\ \frac{\partial y}{\partial s} \\ \frac{\partial z}{\partial s} \end{Bmatrix}_{(r,s,t=0)}$$

From Eqn. (3.3) we can easily find out derivatives of x, y, z with reference to r & s and putting value of r, s in them we find the values at the point.

We know that;

$$\mathbf{a} \times \mathbf{b} = \mathbf{c}$$

where direction of \mathbf{c} is normal to $\mathbf{a} & \mathbf{b}$ and magnitude is $absin\theta$, where, θ is the angle between $\mathbf{a} & \mathbf{b}$. Hence unit vector normal to mid-surface is;

$$v_3 = \begin{Bmatrix} l_3 \\ m_3 \\ n_3 \end{Bmatrix} = \frac{\begin{Bmatrix} \frac{\partial x}{\partial r} \\ \frac{\partial y}{\partial r} \\ \frac{\partial z}{\partial r} \end{Bmatrix}_{(r,s,t=0)} \times \begin{Bmatrix} \frac{\partial x}{\partial s} \\ \frac{\partial y}{\partial s} \\ \frac{\partial z}{\partial s} \end{Bmatrix}_{(r,s,t=0)}}{\left| \begin{Bmatrix} \frac{\partial x}{\partial r} \\ \frac{\partial y}{\partial r} \\ \frac{\partial z}{\partial r} \end{Bmatrix}_{(r,s,t=0)} \times \begin{Bmatrix} \frac{\partial x}{\partial s} \\ \frac{\partial y}{\partial s} \\ \frac{\partial z}{\partial s} \end{Bmatrix}_{(r,s,t=0)} \right|} \quad (3.4)$$

Unit vector normal to v_3 and x' at $(r=0, s=0, t=0)$ but with same origin as of v_3 is constructed by;

$$v_2 = \begin{Bmatrix} l_2 \\ m_2 \\ n_2 \end{Bmatrix}_{(r,s,t=0)} = \frac{\begin{Bmatrix} l_3 \\ m_3 \\ n_3 \end{Bmatrix}_{(r,s,t=0)} \times \begin{Bmatrix} \frac{\partial x}{\partial r} \\ \frac{\partial y}{\partial r} \\ \frac{\partial z}{\partial r} \end{Bmatrix}_{(r=0,s=0,t=0)}}{\left| \begin{Bmatrix} l_3 \\ m_3 \\ n_3 \end{Bmatrix}_{(r,s,t=0)} \times \begin{Bmatrix} \frac{\partial x}{\partial r} \\ \frac{\partial y}{\partial r} \\ \frac{\partial z}{\partial r} \end{Bmatrix}_{(r=0,s=0,t=0)} \right|} \quad (3.5)$$

Now, we have normal vector v_3 and one tangential vector v_2 along y' . So unit vector along x' is;

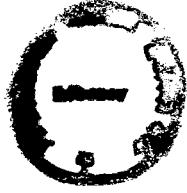
$$\begin{Bmatrix} l_1 \\ m_1 \\ n_1 \end{Bmatrix} = \begin{Bmatrix} l_2 \\ m_2 \\ n_2 \end{Bmatrix}_{(r,s,t=0)} \times \begin{Bmatrix} l_3 \\ m_3 \\ n_3 \end{Bmatrix}_{(r,s,t=0)} \quad (3.6)$$

Now direction cosine matrix of local axes x', y', z' with respect to global axis x, y, z are defined by [D]

$$[D] = \begin{bmatrix} l_1 & l_2 & l_3 \\ m_1 & m_2 & m_3 \\ n_1 & n_2 & n_3 \end{bmatrix} \quad (3.7)$$

3.3 Jacobian

Jacobian is defined as;

$$\begin{Bmatrix} \frac{\partial}{\partial r} \\ \frac{\partial}{\partial s} \\ \frac{\partial}{\partial t} \end{Bmatrix} = [J] \begin{Bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{Bmatrix} \quad (3.8)$$


where, [J] is Jacobian, and is given by;

$$[J] = \begin{bmatrix} \frac{\partial x}{\partial r} & \frac{\partial y}{\partial r} & \frac{\partial z}{\partial r} \\ \frac{\partial x}{\partial s} & \frac{\partial y}{\partial s} & \frac{\partial z}{\partial s} \\ \frac{\partial x}{\partial t} & \frac{\partial y}{\partial t} & \frac{\partial z}{\partial t} \end{bmatrix} \quad \text{TH-7220} \quad (3.9)$$

with relations given in Eqn.(3.1 & 3.2), nodal coordinates of the element and natural coordinates of any point ($r, s, t = 0$) at midsurface, [J] can easily be found at the point.

Eqn.(3.2) is first differentiated with reference to r, s, t then t is made zero.

$$\frac{\partial x}{\partial r} = \sum_{i=1}^4 \frac{\partial N_i}{\partial r} x_i$$

$$\frac{\partial x}{\partial s} = \sum_{i=1}^4 \frac{\partial N_i}{\partial s} x_i \quad (3.10)$$

$$\frac{\partial x}{\partial t} = \sum_{i=1}^4 \frac{1}{2} \times H_i \times N_i \times l_{3i}$$

Similarly derivatives of y & z with respect to r, s, t can be found.

3.4 Displacements

The basic relation between nodal displacements ($u_i, v_i, w_i, \theta_{xi}, \theta_{yi}, \theta_{zi}$), shape function in natural coordinates and components of normal vector at the point where the displacement (u, v, w) are required is given by;

$$\begin{Bmatrix} u \\ v \\ w \end{Bmatrix} = \sum_{i=1}^4 N_i \begin{Bmatrix} u_i \\ v_i \\ w_i \end{Bmatrix} + \frac{1}{2} t H_i \begin{Bmatrix} n_{3i} \theta_{yi} - m_{3i} \theta_{zi} \\ l_{3i} \theta_{zi} - n_{3i} \theta_{xi} \\ m_{3i} \theta_{xi} - l_{3i} \theta_{yi} \end{Bmatrix} \quad (3.11)$$

3.5 Degrees of Freedom

Degrees of freedom required for unique definition of (u, v, w) at any point, associated at nodes are;

u_i, v_i, w_i displacement along global x, y, z axes.

$\theta_{xi}, \theta_{yi}, \theta_{zi}$ rotation along global x, y, z axes following right hand screw rule.

In this way there are six degrees of freedom per node. And total degrees of freedom associated with an element are twenty four.

3.6 Strain Displacement Matrix

Differentiating Eqn.(3.11) with respect to r, s, t

$$\begin{aligned}\frac{\partial u}{\partial r} &= \sum_{i=1}^4 \frac{\partial N_i}{\partial r} u_i + t \sum_{i=1}^4 \frac{\partial N_i}{\partial r} \frac{H_i}{2} (n_{3i} \theta_{yi} - m_{3i} \theta_{zi}) \\ \frac{\partial u}{\partial s} &= \sum_{i=1}^4 \frac{\partial N_i}{\partial s} u_i + t \sum_{i=1}^4 \frac{\partial N_i}{\partial s} \frac{H_i}{2} (n_{3i} \theta_{yi} - m_{3i} \theta_{zi}) \\ \frac{\partial u}{\partial t} &= \sum_{i=1}^4 \frac{1}{2} H_i N_i (n_{3i} \theta_{yi} - m_{3i} \theta_{zi})\end{aligned}\tag{3.12}$$

Similarly differentiating v and w with reference to r, s, t . We find;

$$\begin{aligned}\begin{bmatrix} \frac{\partial u}{\partial r} & \frac{\partial v}{\partial r} & \frac{\partial w}{\partial r} \\ \frac{\partial u}{\partial s} & \frac{\partial v}{\partial s} & \frac{\partial w}{\partial s} \\ \frac{\partial u}{\partial t} & \frac{\partial v}{\partial t} & \frac{\partial w}{\partial t} \end{bmatrix} &= \sum_{i=1}^4 \begin{bmatrix} \frac{\partial N_i}{\partial r} u_i & \frac{\partial N_i}{\partial r} v_i & \frac{\partial N_i}{\partial r} w_i \\ \frac{\partial N_i}{\partial s} u_i & \frac{\partial N_i}{\partial s} v_i & \frac{\partial N_i}{\partial s} w_i \\ 0 & 0 & 0 \end{bmatrix} + \\ \sum_{i=1}^4 \frac{H_i}{2} \times &\begin{bmatrix} t \frac{\partial N_i}{\partial r} (n_{3i} \theta_{yi} - m_{3i} \theta_{zi}) & t \frac{\partial N_i}{\partial r} (l_{3i} \theta_{zi} - n_{3i} \theta_{xi}) & t \frac{\partial N_i}{\partial r} (m_{3i} \theta_{xi} - l_{3i} \theta_{yi}) \\ t \frac{\partial N_i}{\partial s} (n_{3i} \theta_{yi} - m_{3i} \theta_{zi}) & t \frac{\partial N_i}{\partial s} (l_{3i} \theta_{zi} - n_{3i} \theta_{xi}) & t \frac{\partial N_i}{\partial s} (m_{3i} \theta_{xi} - l_{3i} \theta_{yi}) \\ N_i (n_{3i} \theta_{yi} - m_{3i} \theta_{zi}) & N_i (l_{3i} \theta_{zi} - n_{3i} \theta_{xi}) & N_i (m_{3i} \theta_{xi} - l_{3i} \theta_{yi}) \end{bmatrix}\end{aligned}\tag{3.13}$$

The strain components in local and global axes system are related as below;

$$\begin{bmatrix} \frac{\partial u'}{\partial x'} & \frac{\partial v'}{\partial x'} & \frac{\partial w'}{\partial x'} \\ \frac{\partial u'}{\partial y'} & \frac{\partial v'}{\partial y'} & \frac{\partial w'}{\partial y'} \\ \frac{\partial u'}{\partial z'} & \frac{\partial v'}{\partial z'} & \frac{\partial w'}{\partial z'} \end{bmatrix} = [D]^T \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial v}{\partial x} & \frac{\partial w}{\partial x} \\ \frac{\partial u}{\partial y} & \frac{\partial v}{\partial y} & \frac{\partial w}{\partial y} \\ \frac{\partial u}{\partial z} & \frac{\partial v}{\partial z} & \frac{\partial w}{\partial z} \end{bmatrix} [D]\tag{3.14}$$

Derivative of u, v, w with respect to r, s, t can be converted into derivative of u, v, w with respect to x, y, z as below with the help of **Jacobian**

$$\begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial v}{\partial x} & \frac{\partial w}{\partial x} \\ \frac{\partial u}{\partial y} & \frac{\partial v}{\partial y} & \frac{\partial w}{\partial y} \\ \frac{\partial u}{\partial z} & \frac{\partial v}{\partial z} & \frac{\partial w}{\partial z} \end{bmatrix} = [J^{-1}] \begin{bmatrix} \frac{\partial u}{\partial r} & \frac{\partial v}{\partial r} & \frac{\partial w}{\partial r} \\ \frac{\partial u}{\partial s} & \frac{\partial v}{\partial s} & \frac{\partial w}{\partial s} \\ \frac{\partial u}{\partial t} & \frac{\partial v}{\partial t} & \frac{\partial w}{\partial t} \end{bmatrix} \quad (3.15)$$

Now we can find relevant strain component in local axes of the shell element;

$$\begin{bmatrix} \varepsilon_x' \\ \varepsilon_y' \\ \gamma_{xy'} \\ \gamma_{xz'} \\ \gamma_{yz'} \end{bmatrix} = \begin{bmatrix} \frac{\partial u'}{\partial x'} \\ \frac{\partial v'}{\partial y'} \\ \frac{\partial u'}{\partial y'} + \frac{\partial v'}{\partial x'} \\ \frac{\partial u'}{\partial z'} + \frac{\partial w'}{\partial x'} \\ \frac{\partial v'}{\partial z'} + \frac{\partial w'}{\partial y'} \end{bmatrix} \quad (3.16)$$

Since,

$$\begin{aligned} \frac{\partial N_i}{\partial x} &= J^*_{11} \times \frac{\partial N_i}{\partial r} + J^*_{12} \times \frac{\partial N_i}{\partial s} \\ \frac{\partial N_i}{\partial y} &= J^*_{21} \times \frac{\partial N_i}{\partial r} + J^*_{22} \times \frac{\partial N_i}{\partial s} \\ \frac{\partial N_i}{\partial z} &= J^*_{31} \times \frac{\partial N_i}{\partial r} + J^*_{32} \times \frac{\partial N_i}{\partial s} \end{aligned} \quad (3.17)$$

R.H.S of Eqn. (3.15) can be written as;

$$\begin{aligned}
&= \sum_{i=1}^4 \begin{bmatrix} \frac{\partial N_i}{\partial x} u_i & \frac{\partial N_i}{\partial x} v_i & \frac{\partial N_i}{\partial x} w_i \\ \frac{\partial N_i}{\partial y} u_i & \frac{\partial N_i}{\partial y} v_i & \frac{\partial N_i}{\partial y} w_i \\ \frac{\partial N_i}{\partial z} u_i & \frac{\partial N_i}{\partial z} v_i & \frac{\partial N_i}{\partial z} w_i \end{bmatrix} + \sum_{i=1}^4 \frac{H_i}{2} \begin{bmatrix} t \frac{\partial N_i}{\partial x} (n_{3i}\theta_{yi} - m_{3i}\theta_{zi}) & t \frac{\partial N_i}{\partial x} (l_{3i}\theta_{zi} - n_{3i}\theta_{xi}) & t \frac{\partial N_i}{\partial x} (m_{3i}\theta_{xi} - l_{3i}\theta_{yi}) \\ t \frac{\partial N_i}{\partial y} (n_{3i}\theta_{yi} - m_{3i}\theta_{zi}) & t \frac{\partial N_i}{\partial y} (l_{3i}\theta_{zi} - n_{3i}\theta_{xi}) & t \frac{\partial N_i}{\partial y} (m_{3i}\theta_{xi} - l_{3i}\theta_{yi}) \\ t \frac{\partial N_i}{\partial z} (n_{3i}\theta_{yi} - m_{3i}\theta_{zi}) & t \frac{\partial N_i}{\partial z} (l_{3i}\theta_{zi} - n_{3i}\theta_{xi}) & t \frac{\partial N_i}{\partial z} (m_{3i}\theta_{xi} - l_{3i}\theta_{yi}) \end{bmatrix} + \\
&\quad \sum_{i=1}^4 \frac{H_i}{2} \begin{bmatrix} J^*_{13} \times N_i (n_{3i}\theta_{yi} - m_{3i}\theta_{zi}) & J^*_{13} \times N_i (l_{3i}\theta_{zi} - n_{3i}\theta_{xi}) & J^*_{13} \times N_i (m_{3i}\theta_{xi} - l_{3i}\theta_{yi}) \\ J^*_{23} \times N_i (n_{3i}\theta_{yi} - m_{3i}\theta_{zi}) & J^*_{23} \times N_i (l_{3i}\theta_{zi} - n_{3i}\theta_{xi}) & J^*_{23} \times N_i (m_{3i}\theta_{xi} - l_{3i}\theta_{yi}) \\ J^*_{33} \times N_i (n_{3i}\theta_{yi} - m_{3i}\theta_{zi}) & J^*_{33} \times N_i (l_{3i}\theta_{zi} - n_{3i}\theta_{xi}) & J^*_{33} \times N_i (m_{3i}\theta_{xi} - l_{3i}\theta_{yi}) \end{bmatrix} \quad (3.18)
\end{aligned}$$

R.H.S of Eqn. (3.14) can be written as;

$$\begin{bmatrix} l_1 & m_1 & n_1 \\ l_2 & m_2 & n_2 \\ l_3 & m_3 & n_3 \end{bmatrix} \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial v}{\partial x} & \frac{\partial w}{\partial x} \\ \frac{\partial u}{\partial y} & \frac{\partial v}{\partial y} & \frac{\partial w}{\partial y} \\ \frac{\partial u}{\partial z} & \frac{\partial v}{\partial z} & \frac{\partial w}{\partial z} \end{bmatrix} \begin{bmatrix} l_1 & l_2 & l_3 \\ m_1 & m_2 & m_3 \\ n_1 & n_2 & n_3 \end{bmatrix} \quad (3.19)$$

three terms of R.H.S of Eqn.(3.18) independently pre and post multiplied by $[D]^T$ and $[D]$ is equivalent to R.H.S of Eqn.(3.19).

$$\begin{aligned}
&\begin{bmatrix} \frac{\partial u'}{\partial x'} & \frac{\partial v'}{\partial x'} & \frac{\partial w'}{\partial x'} \\ \frac{\partial u'}{\partial y'} & \frac{\partial v'}{\partial y'} & \frac{\partial w'}{\partial y'} \\ \frac{\partial u'}{\partial z'} & \frac{\partial v'}{\partial z'} & \frac{\partial w'}{\partial z'} \end{bmatrix} = \sum_{i=1}^4 \begin{bmatrix} l_1 & m_1 & n_1 \\ l_2 & m_2 & n_2 \\ l_3 & m_3 & n_3 \end{bmatrix} \begin{bmatrix} \frac{\partial N_i}{\partial x} u_i & \frac{\partial N_i}{\partial x} v_i & \frac{\partial N_i}{\partial x} w_i \\ \frac{\partial N_i}{\partial y} u_i & \frac{\partial N_i}{\partial y} v_i & \frac{\partial N_i}{\partial y} w_i \\ \frac{\partial N_i}{\partial z} u_i & \frac{\partial N_i}{\partial z} v_i & \frac{\partial N_i}{\partial z} w_i \end{bmatrix} \begin{bmatrix} l_1 & l_2 & l_3 \\ m_1 & m_2 & m_3 \\ n_1 & n_2 & n_3 \end{bmatrix} + \\
&\quad \sum_{i=1}^4 \frac{H_i}{2} \begin{bmatrix} l_1 & m_1 & n_1 \\ l_2 & m_2 & n_2 \\ l_3 & m_3 & n_3 \end{bmatrix} \begin{bmatrix} J^*_{13} \times N_i A & J^*_{13} \times N_i B & J^*_{13} \times N_i C \\ J^*_{23} \times N_i A & J^*_{23} \times N_i B & J^*_{23} \times N_i C \\ J^*_{33} \times N_i A & J^*_{33} \times N_i B & J^*_{33} \times N_i C \end{bmatrix} \begin{bmatrix} l_1 & l_2 & l_3 \\ m_1 & m_2 & m_3 \\ n_1 & n_2 & n_3 \end{bmatrix} \\
&\quad + \sum_{i=1}^4 \frac{H_i}{2} \begin{bmatrix} l_1 & m_1 & n_1 \\ l_2 & m_2 & n_2 \\ l_3 & m_3 & n_3 \end{bmatrix} \begin{bmatrix} t \frac{\partial N_i}{\partial x} A & t \frac{\partial N_i}{\partial x} B & t \frac{\partial N_i}{\partial x} C \\ t \frac{\partial N_i}{\partial y} A & t \frac{\partial N_i}{\partial y} B & t \frac{\partial N_i}{\partial y} C \\ t \frac{\partial N_i}{\partial z} A & t \frac{\partial N_i}{\partial z} B & t \frac{\partial N_i}{\partial z} C \end{bmatrix} \begin{bmatrix} l_1 & l_2 & l_3 \\ m_1 & m_2 & m_3 \\ n_1 & n_2 & n_3 \end{bmatrix} \quad (3.20)
\end{aligned}$$

$$\text{Where } A = (n_{3i}\theta_y - m_{3i}\theta_z), \quad B = (l_{3i}\theta_z - n_{3i}\theta_x), \quad C = (m_{3i}\theta_x - l_{3i}\theta_y)$$

Every strain component in local axes is summation of three terms. First term is associated with nodal values of u, v, w . Second term is associated with nodal value of $\theta_x, \theta_y, \theta_z$ but not associated with t . Third term is associated with nodal of $\theta_x, \theta_y, \theta_z$ and also with t .

Now if we write **strain displacement matrix** as;

$$[B] = \begin{bmatrix} [B_m] \\ [B_s] \end{bmatrix} = \sum_{i=1}^4 \begin{bmatrix} [B_{1mi} + B_{2mi} + tB_{3mi}] \\ [B_{1si} + B_{2si} + tB_{3si}] \end{bmatrix} \quad (3.21)$$

then;

$$\begin{Bmatrix} \varepsilon_{x'} \\ \varepsilon_{y'} \\ \gamma_{x'y'} \end{Bmatrix} = \sum_{i=1}^4 ([B_{1mi}] \{u_i\} + [B_{2mi}] \{\theta_i\} + t[B_{3mi}] \{\theta_i\}) \quad (3.21.a)$$

$$\begin{Bmatrix} \varepsilon_{z'} \\ \gamma_{yz'} \end{Bmatrix} = \sum_{i=1}^4 ([B_{1si}] \{u_i\} + [B_{2si}] \{\theta_i\} + t[B_{3si}] \{\theta_i\}) \quad (3.21.b)$$

3.6.1 Formulation of $[B_{1mi}]$

From Eqn.(3.20) we can write;

$$\begin{aligned} \frac{\partial u'}{\partial x'} &= \sum_{i=1}^4 \left[l_1 \left(l_1 \frac{\partial N_i}{\partial x} u_i + m_1 \frac{\partial N_i}{\partial x} v_i + n_1 \frac{\partial N_i}{\partial x} w_i \right) + m_1 \left(l_1 \frac{\partial N_i}{\partial y} u_i + m_1 \frac{\partial N_i}{\partial y} v_i + n_1 \frac{\partial N_i}{\partial y} w_i \right) \right. \\ &\quad \left. + n_1 \left(l_1 \frac{\partial N_i}{\partial z} u_i + m_1 \frac{\partial N_i}{\partial z} v_i + n_1 \frac{\partial N_i}{\partial z} w_i \right) \right] \\ &= \sum_{i=1}^4 \left[l_1 \left(l_1 \frac{\partial N_i}{\partial x} + m_1 \frac{\partial N_i}{\partial y} + n_1 \frac{\partial N_i}{\partial z} \right) u_i + m_1 \left(l_1 \frac{\partial N_i}{\partial x} + m_1 \frac{\partial N_i}{\partial y} + n_1 \frac{\partial N_i}{\partial z} \right) v_i \right. \\ &\quad \left. + n_1 \left(l_1 \frac{\partial N_i}{\partial x} + m_1 \frac{\partial N_i}{\partial y} + n_1 \frac{\partial N_i}{\partial z} \right) w_i \right] \end{aligned}$$

$$= \sum_{i=1}^4 \left(\begin{bmatrix} l_1 B'(1,i) & m_1 B'(1,i) & n_1 B'(1,i) \end{bmatrix} \begin{Bmatrix} u_i \\ v_i \\ w_i \end{Bmatrix} \right)$$

where,

$$B'(1,i) = \left(l_1 \frac{\partial N_i}{\partial x} + m_1 \frac{\partial N_i}{\partial y} + n_1 \frac{\partial N_i}{\partial z} \right)$$

If we identify elements of L.H.S matrix of Eqn. (3.20) by (i,j) we notice that i gives subscripts of elements of row of premultiplication direction cosine matrix and j gives subscripts of elements of column of post multiplication direction cosine matrix. Subscript of direction cosine matrix elements inside circular bracket pertains to i . Subscripts of direction cosine matrix elements outside circular bracket pertains to j .

$$\text{so } \frac{\partial v'}{\partial y'} = \sum_{i=1}^4 \left(l_2 B'(2,i) \quad m_2 B'(2,i) \quad n_2 B'(2,i) \begin{Bmatrix} u_i \\ v_i \\ w_i \end{Bmatrix} \right)$$

$$\frac{\partial u'}{\partial y'} + \frac{\partial v'}{\partial x'} = \sum_{i=1}^4 \left([l_1 B'(2,i) + l_2 B'(1,i)] \quad [m_1 B'(2,i) + m_2 B'(1,i)] \quad [n_1 B'(2,i) + n_2 B'(1,i)] \begin{Bmatrix} u_i \\ v_i \\ w_i \end{Bmatrix} \right)$$

$$\begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix} = \sum_{i=1}^4 \left[\begin{array}{ccc} l_1 B'(1,i) & m_1 B'(1,i) & n_1 B'(1,i) \\ l_2 B'(2,i) & m_2 B'(2,i) & n_2 B'(2,i) \\ l_1 B'(2,i) & m_1 B'(2,i) & n_1 B'(2,i) \\ + & + & + \\ l_2 B'(1,i) & m_2 B'(1,i) & n_2 B'(1,i) \end{array} \right] \begin{Bmatrix} u_i \\ v_i \\ w_i \end{Bmatrix} = \sum_{i=1}^4 [B_{1mi}] \{u_i\}$$

where,

$$B'(2,i) = \left(l_2 \frac{\partial N_i}{\partial x} + m_2 \frac{\partial N_i}{\partial y} + n_2 \frac{\partial N_i}{\partial z} \right)$$

$$[B_{1mi}] = \begin{bmatrix} l_1 B'(1,i) & m_1 B'(1,i) & n_1 B'(1,i) \\ l_2 B'(2,i) & m_2 B'(2,i) & n_2 B'(2,i) \\ l_1 B'(2,i) & m_1 B'(2,i) & n_1 B'(2,i) \\ + & + & + \\ l_2 B'(1,i) & m_2 B'(1,i) & n_2 B'(1,i) \end{bmatrix} \quad (3.22)$$

3.6.2 Formulation of $[B_{2mi}]$

$$\frac{\partial u'}{\partial x'} = \sum_{i=1}^4 \frac{H_i}{2} (l_1(l_1 J_{13}^* + m_1 J_{23}^* + n_1 J_{33}^*) N_i A + m_1(l_1 J_{13}^* + m_1 J_{23}^* + n_1 J_{33}^*) N_i B + n_1(l_1 J_{13}^* + m_1 J_{23}^* + n_1 J_{33}^*) N_i C) = 0$$

$$\frac{\partial v'}{\partial y'} = \sum_{i=1}^4 \frac{H_i}{2} (l_2(l_2 J_{13}^* + m_2 J_{23}^* + n_2 J_{33}^*) N_i A + m_2(l_2 J_{13}^* + m_2 J_{23}^* + n_2 J_{33}^*) N_i B + n_2(l_2 J_{13}^* + m_2 J_{23}^* + n_2 J_{33}^*) N_i C) = 0$$

$$\begin{aligned} \frac{\partial u'}{\partial y'} + \frac{\partial v'}{\partial x'} = & \sum_{i=1}^4 \frac{H_i}{2} (l_1(l_2 J_{13}^* + m_2 J_{23}^* + n_2 J_{33}^*) N_i A \\ & + m_1(l_2 J_{13}^* + m_2 J_{23}^* + n_2 J_{33}^*) N_i B \\ & + n_1(l_2 J_{13}^* + m_2 J_{23}^* + n_2 J_{33}^*) N_i C \\ & + l_2(l_1 J_{13}^* + m_1 J_{23}^* + n_1 J_{33}^*) N_i A \\ & + m_2(l_1 J_{13}^* + m_1 J_{23}^* + n_1 J_{33}^*) N_i B \\ & + n_2(l_1 J_{13}^* + m_1 J_{23}^* + n_1 J_{33}^*) N_i C) = 0 \end{aligned}$$

since, $J_{13}^* = l_3$ $J_{23}^* = m_3$ $J_{33}^* = n_3$

and $l_1 l_3 + m_1 m_3 + n_1 n_3 = 0$
 $l_2 l_3 + m_2 m_3 + n_2 n_3 = 0$

$$[B_{2mi}] = [0] \quad (3.23)$$

3.6.3 Formulation of $[B_{3mi}]$

$$\begin{aligned}
\frac{\partial u'}{\partial x'} &= \sum_{i=1}^4 \frac{H_i}{2} t \left(l_1 \left(l_1 \frac{\partial N_i}{\partial x} + m_1 \frac{\partial N_i}{\partial y} + n_1 \frac{\partial N_i}{\partial z} \right) A \right. \\
&\quad \left. + m_1 \left(l_1 \frac{\partial N_i}{\partial x} + m_1 \frac{\partial N_i}{\partial y} + n_1 \frac{\partial N_i}{\partial z} \right) B + n_1 \left(l_1 \frac{\partial N_i}{\partial x} + m_1 \frac{\partial N_i}{\partial y} + n_1 \frac{\partial N_i}{\partial z} \right) C \right) \\
&= \sum_{i=1}^4 \frac{H_i}{2} t (l_1 B'(1,i) A + m_1 B'(1,i) B + n_1 B'(1,i) C) \\
&= t \sum_{i=1}^4 \frac{H_i}{2} [B'(1,i)(m_{3i}n_1 - n_{3i}m_1)B'(1,i)(n_{3i}l_1 - l_{3i}n_1)B'(1,i)(l_{3i}m_1 - m_{3i}l_1)] \begin{bmatrix} \theta_{xi} \\ \theta_{yi} \\ \theta_{zi} \end{bmatrix} \\
\frac{\partial v'}{\partial y'} &= t \sum_{i=1}^4 \frac{H_i}{2} [B'(2,i)(m_{3i}n_2 - n_{3i}m_2)B'(2,i)(n_{3i}l_2 - l_{3i}n_2)B'(2,i)(l_{3i}m_2 - m_{3i}l_2)] \begin{bmatrix} \theta_{xi} \\ \theta_{yi} \\ \theta_{zi} \end{bmatrix} \\
\frac{\partial u'}{\partial y'} + \frac{\partial v'}{\partial x'} &= t \sum_{i=1}^4 \frac{H_i}{2} \left[\begin{array}{ccc} B'(2,i)(m_{3i}n_1 - n_{3i}m_1)B'(2,i)(n_{3i}l_1 - l_{3i}n_1)B'(2,i)(l_{3i}m_1 - m_{3i}l_1) \\ + & + & + \\ B'(1,i)(m_{3i}n_2 - n_{3i}m_2)B'(1,i)(n_{3i}l_2 - l_{3i}n_2)B'(1,i)(l_{3i}m_2 - m_{3i}l_2) \end{array} \right] \begin{bmatrix} \theta_{xi} \\ \theta_{yi} \\ \theta_{zi} \end{bmatrix} \\
[B_{3mi}] &= \begin{bmatrix} B'(1,i)(m_{3i}n_1 - n_{3i}m_1) & B'(1,i)(n_{3i}l_1 - l_{3i}n_1) & B'(1,i)(l_{3i}m_1 - m_{3i}l_1) \\ B'(2,i)(m_{3i}n_2 - n_{3i}m_2) & B'(2,i)(n_{3i}l_2 - l_{3i}n_2) & B'(2,i)(l_{3i}m_2 - m_{3i}l_2) \\ B'(2,i)(m_{3i}n_1 - n_{3i}m_1) & B'(2,i)(n_{3i}l_1 - l_{3i}n_1) & B'(2,i)(l_{3i}m_1 - m_{3i}l_1) \\ + & + & + \\ B'(1,i)(m_{3i}n_2 - n_{3i}m_2) & B'(1,i)(n_{3i}l_2 - l_{3i}n_2) & B'(1,i)(l_{3i}m_2 - m_{3i}l_2) \end{bmatrix} \quad (3.24)
\end{aligned}$$

3.6.4 Formulation of $[B_{1sl}]$

$$\begin{aligned}
\frac{\partial u'}{\partial z'} + \frac{\partial w'}{\partial x'} &= \sum_{i=1}^4 l_1 \left(l_3 \frac{\partial N_i}{\partial x} + m_3 \frac{\partial N_i}{\partial y} + n_3 \frac{\partial N_i}{\partial z} \right) u_i + l_3 \left(l_1 \frac{\partial N_i}{\partial x} + m_1 \frac{\partial N_i}{\partial y} + n_1 \frac{\partial N_i}{\partial z} \right) u_i \\
&\quad + m_1 \left(l_3 \frac{\partial N_i}{\partial x} + m_3 \frac{\partial N_i}{\partial y} + n_3 \frac{\partial N_i}{\partial z} \right) v_i + m_3 \left(l_1 \frac{\partial N_i}{\partial x} + m_1 \frac{\partial N_i}{\partial y} + n_1 \frac{\partial N_i}{\partial z} \right) v_i \\
&\quad + n_1 \left(l_3 \frac{\partial N_i}{\partial x} + m_3 \frac{\partial N_i}{\partial y} + n_3 \frac{\partial N_i}{\partial z} \right) w_i + n_3 \left(l_1 \frac{\partial N_i}{\partial x} + m_1 \frac{\partial N_i}{\partial y} + n_1 \frac{\partial N_i}{\partial z} \right) w_i
\end{aligned}$$

$$= \sum_{i=1}^4 \begin{bmatrix} l_1 B'(3,i) & m_1 B'(3,i) & n_1 B'(3,i) \\ + & + & + \\ l_3 B'(1,i) & m_3 B'(1,i) & n_3 B'(1,i) \end{bmatrix} \begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix}$$

$$\frac{\partial v'}{\partial z'} + \frac{\partial w'}{\partial y'} = \sum_{i=1}^4 l_2 \left(l_3 \frac{\partial N_i}{\partial x} + m_3 \frac{\partial N_i}{\partial y} + n_3 \frac{\partial N_i}{\partial z} \right) u_i + l_3 \left(l_2 \frac{\partial N_i}{\partial x} + m_2 \frac{\partial N_i}{\partial y} + n_2 \frac{\partial N_i}{\partial z} \right) u_i$$

$$+ m_2 \left(l_3 \frac{\partial N_i}{\partial x} + m_3 \frac{\partial N_i}{\partial y} + n_3 \frac{\partial N_i}{\partial z} \right) v_i + m_3 \left(l_2 \frac{\partial N_i}{\partial x} + m_2 \frac{\partial N_i}{\partial y} + n_2 \frac{\partial N_i}{\partial z} \right) v_i$$

$$+ n_2 \left(l_3 \frac{\partial N_i}{\partial x} + m_3 \frac{\partial N_i}{\partial y} + n_3 \frac{\partial N_i}{\partial z} \right) w_i + n_3 \left(l_2 \frac{\partial N_i}{\partial x} + m_2 \frac{\partial N_i}{\partial y} + n_2 \frac{\partial N_i}{\partial z} \right) w_i$$

$$= \sum_{i=1}^4 \begin{bmatrix} l_2 B'(3,i) & m_2 B'(3,i) & n_2 B'(3,i) \\ + & + & + \\ l_3 B'(2,i) & m_3 B'(2,i) & n_3 B'(2,i) \end{bmatrix} \begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix}$$

$$\left[B_{1si} \right] = \begin{bmatrix} l_1 B'(3,i) & m_1 B'(3,i) & n_1 B'(3,i) \\ + & + & + \\ l_3 B'(1,i) & m_3 B'(1,i) & n_3 B'(1,i) \\ l_2 B'(3,i) & m_2 B'(3,i) & n_2 B'(3,i) \\ + & + & + \\ l_3 B'(2,i) & m_3 B'(2,i) & n_3 B'(2,i) \end{bmatrix} \quad (3.25)$$

3.6.5 Formulation of [B_{2si}]

$$\frac{\partial u'}{\partial z'} + \frac{\partial w'}{\partial x'} = \sum_{i=1}^4 \left(\frac{H_i}{2} l_1 (l_3 J_{13}^* + m_3 J_{23}^* + n_3 J_{33}^*) N_i A + l_3 (l_1 J_{13}^* + m_1 J_{23}^* + n_1 J_{33}^*) N_i A \right.$$

$$+ m_1 (l_3 J_{13}^* + m_3 J_{23}^* + n_3 J_{33}^*) N_i B + m_3 (l_1 J_{13}^* + m_1 J_{23}^* + n_1 J_{33}^*) N_i B$$

$$+ n_1 (l_3 J_{13}^* + m_3 J_{23}^* + n_3 J_{33}^*) N_i C + n_3 (l_1 J_{13}^* + m_1 J_{23}^* + n_1 J_{33}^*) N_i C \right)$$

$$= \sum_{i=1}^4 \frac{H_i}{2} N_i (l_3 J_{13}^* + m_3 J_{23}^* + n_3 J_{33}^*) [(m_{3i}n_1 - n_{3i}m_1)(n_{3i}l_1 - l_{3i}n_1)(l_{3i}m_1 - m_{3i}l_1)] \begin{Bmatrix} \theta_{xi} \\ \theta_{yi} \\ \theta_{zi} \end{Bmatrix}$$

$$\begin{aligned} \frac{\partial v'}{\partial z'} + \frac{\partial w'}{\partial y'} &= \sum_{i=1}^4 \left(\frac{H_i}{2} l_2 (l_3 J_{13}^* + m_3 J_{23}^* + n_3 J_{33}^*) N_i A + l_3 (l_2 J_{13}^* + m_2 J_{23}^* + n_2 J_{33}^*) N_i A \right. \\ &\quad + m_2 (l_3 J_{13}^* + m_3 J_{23}^* + n_3 J_{33}^*) N_i B + m_3 (l_2 J_{13}^* + m_2 J_{23}^* + n_2 J_{33}^*) N_i B \\ &\quad \left. + n_2 (l_3 J_{13}^* + m_3 J_{23}^* + n_3 J_{33}^*) N_i C + n_3 (l_2 J_{13}^* + m_2 J_{23}^* + n_2 J_{33}^*) N_i C \right) \end{aligned}$$

$$= \sum_{i=1}^4 \frac{H_i}{2} N_i (l_3 J_{13}^* + m_3 J_{23}^* + n_3 J_{33}^*) [(m_{3i}n_2 - n_{3i}m_2)(n_{3i}l_2 - l_{3i}n_2)(l_{3i}m_2 - m_{3i}l_2)] \begin{Bmatrix} \theta_{xi} \\ \theta_{yi} \\ \theta_{zi} \end{Bmatrix}$$

$$[B_{2si}] = \frac{H_i}{2} N_i (l_3 J_{13}^* + m_3 J_{23}^* + n_3 J_{33}^*) \begin{Bmatrix} (m_{3i}n_1 - n_{3i}m_1)(n_{3i}l_1 - l_{3i}n_1)(l_{3i}m_1 - m_{3i}l_1) \\ (m_{3i}n_2 - n_{3i}m_2)(n_{3i}l_2 - l_{3i}n_2)(l_{3i}m_2 - m_{3i}l_2) \end{Bmatrix} \quad (3.26)$$

3.6.6 Formulation of [B_{3si}]

$$\begin{aligned} \frac{\partial u'}{\partial z'} + \frac{\partial w'}{\partial x'} &= \sum_{i=1}^4 \frac{H_i}{2} \left(l_1 \left(l_3 \frac{\partial N_i}{\partial x} + m_3 \frac{\partial N_i}{\partial y} + n_3 \frac{\partial N_i}{\partial z} \right) tA + l_3 \left(l_1 \frac{\partial N_i}{\partial x} + m_1 \frac{\partial N_i}{\partial y} + n_1 \frac{\partial N_i}{\partial z} \right) tA \right. \\ &\quad + m_1 \left(l_3 \frac{\partial N_i}{\partial x} + m_3 \frac{\partial N_i}{\partial y} + n_3 \frac{\partial N_i}{\partial z} \right) tB + m_3 \left(l_1 \frac{\partial N_i}{\partial x} + m_1 \frac{\partial N_i}{\partial y} + n_1 \frac{\partial N_i}{\partial z} \right) tB \\ &\quad \left. + n_1 \left(l_3 \frac{\partial N_i}{\partial x} + m_3 \frac{\partial N_i}{\partial y} + n_3 \frac{\partial N_i}{\partial z} \right) tC + n_3 \left(l_1 \frac{\partial N_i}{\partial x} + m_1 \frac{\partial N_i}{\partial y} + n_1 \frac{\partial N_i}{\partial z} \right) tC \right) \\ &= \sum_{i=1}^4 \frac{H_i}{2} t \begin{Bmatrix} B'(3,i)(m_{3i}n_1 - n_{3i}m_1) & B'(3,i)(n_{3i}l_1 - l_{3i}n_1) & B'(3,i)(l_{3i}m_1 - m_{3i}l_1) \\ + & + & + \\ B'(1,i)(m_{3i}n_3 - n_{3i}m_3) & B'(1,i)(n_{3i}l_3 - l_{3i}n_3) & B'(1,i)(l_{3i}m_3 - m_{3i}l_3) \end{Bmatrix} \begin{Bmatrix} \theta_{xi} \\ \theta_{yi} \\ \theta_{zi} \end{Bmatrix} \end{aligned}$$

$$\frac{\partial v'}{\partial z'} + \frac{\partial w'}{\partial y'} = \sum_{i=1}^4 \frac{H_i}{2} \left(l_2 \left(l_3 \frac{\partial N_i}{\partial x} + m_3 \frac{\partial N_i}{\partial y} + n_3 \frac{\partial N_i}{\partial z} \right) tA + l_3 \left(l_2 \frac{\partial N_i}{\partial x} + m_2 \frac{\partial N_i}{\partial y} + n_2 \frac{\partial N_i}{\partial z} \right) tA \right)$$

$$+ m_2 \left(l_3 \frac{\partial N_i}{\partial x} + m_3 \frac{\partial N_i}{\partial y} + n_3 \frac{\partial N_i}{\partial z} \right) tB + m_3 \left(l_2 \frac{\partial N_i}{\partial x} + m_2 \frac{\partial N_i}{\partial y} + n_2 \frac{\partial N_i}{\partial z} \right) tB \\ + n_2 \left(l_3 \frac{\partial N_i}{\partial x} + m_3 \frac{\partial N_i}{\partial y} + n_3 \frac{\partial N_i}{\partial z} \right) tC + n_3 \left(l_2 \frac{\partial N_i}{\partial x} + m_2 \frac{\partial N_i}{\partial y} + n_2 \frac{\partial N_i}{\partial z} \right) tC \right)$$

$$= \sum_{i=1}^4 \frac{H_i}{2} t \begin{bmatrix} B'(3,i)(m_{3i}n_2 - n_{3i}m_2) & B'(3,i)(n_{3i}l_2 - l_{3i}n_2) & B'(3,i)(l_{3i}m_2 - m_{3i}l_2) \\ + & + & + \\ B'(1,i)(m_{3i}n_3 - n_{3i}m_3) & B'(1,i)(n_{3i}l_3 - l_{3i}n_3) & B'(1,i)(l_{3i}m_3 - m_{3i}l_3) \end{bmatrix} \begin{bmatrix} \theta_{xi} \\ \theta_{yi} \\ \theta_{zi} \end{bmatrix}$$

$$\therefore [B_{3si}] = \frac{H_i}{2} \begin{bmatrix} B'(3,i)(m_{3i}n_1 - n_{3i}m_1) & B'(3,i)(n_{3i}l_1 - l_{3i}n_1) & B'(3,i)(l_{3i}m_1 - m_{3i}l_1) \\ + & + & + \\ B'(1,i)(m_{3i}n_3 - n_{3i}m_3) & B'(1,i)(n_{3i}l_3 - l_{3i}n_3) & B'(1,i)(l_{3i}m_3 - m_{3i}l_3) \\ B'(3,i)(m_{3i}n_2 - n_{3i}m_2) & B'(3,i)(n_{3i}l_2 - l_{3i}n_2) & B'(3,i)(l_{3i}m_2 - m_{3i}l_2) \\ + & + & + \\ B'(2,i)(m_{3i}n_3 - n_{3i}m_3) & B'(2,i)(n_{3i}l_3 - l_{3i}n_3) & B'(2,i)(l_{3i}m_3 - m_{3i}l_3) \end{bmatrix} \quad (3.27)$$

Direction cosine matrix elements with i subscript relates it to node number i , without i terms are at Gauss Point.

3.7 Stress Displacement Matrix

For isotropic material constitutive relation;

$$\begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{xy} \\ \tau_{yz} \\ \tau_{zx} \end{bmatrix} = \frac{E}{(1+\mu)(1-2\mu)} \begin{bmatrix} (1-\mu) & \mu & \mu & 0 & 0 & 0 \\ \mu & (1-\mu) & \mu & 0 & 0 & 0 \\ \mu & \mu & (1-\mu) & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{(1-2\mu)}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{(1-2\mu)}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{(1-2\mu)}{2} \end{bmatrix} \begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \epsilon_z \\ \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{zx} \end{bmatrix} \quad (3.28)$$

for $\sigma_z = 0$ and in local coordinate system;

$$\begin{bmatrix} \sigma_{x'} \\ \sigma_{y'} \\ \tau_{x'y'} \\ \tau_{y'z'} \\ \tau_{z'x'} \end{bmatrix} = \frac{E}{1-\mu^2} \begin{bmatrix} 1 & \mu & 0 & 0 & 0 \\ \mu & 1 & 0 & 0 & 0 \\ 0 & 0 & \frac{(1-\mu)}{2} & 0 & 0 \\ 0 & 0 & 0 & \frac{\alpha(1-\mu)}{2} & 0 \\ 0 & 0 & 0 & 0 & \frac{\alpha(1-\mu)}{2} \end{bmatrix} \begin{bmatrix} \epsilon_{x'} \\ \epsilon_{y'} \\ \gamma_{x'y'} \\ \gamma_{y'z'} \\ \gamma_{z'x'} \end{bmatrix} \quad (3.29)$$

Here shear stress is assumed constant over the thickness, whereas, actually it is varying. So for better shear deformation representation $\alpha = \frac{5}{6}$ is used which is for the parabolic distribution of shear deformation over the thickness.

To avoid shear locking we use reduced integration for transverse shear strains and exact for inplane (bending) strain. We divide [C] matrix as below;

$$[C] = \begin{bmatrix} [C_m] & [0] \\ [0] & [C_s] \end{bmatrix}$$

$$[C_m] = \frac{E}{1-\mu^2} \begin{bmatrix} 1 & \mu & 0 \\ \mu & 1 & 0 \\ 0 & 0 & \frac{1-\mu}{2} \end{bmatrix} \quad [C_s] = \frac{E\alpha}{2(1+\mu)} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

3.8 Element Stiffness Matrix

Earlier we separated strain displacement matrix into two parts, one for $(\epsilon_x, \epsilon_y, \gamma_{xy})$ and another for $(\gamma_{yz}, \gamma_{zx})$ and constitutive matrix [C] into $[C_m]$ and $[C_s]$. This is to allow the use of appropriate order of numerical integration. Now we extend same idea for element stiffness matrix.

$$[K] = [K]_m + [K]_s \quad (3.30)$$

if $[K_{ij}]$ represents stiffness coefficients pertaining to **DOF** of *i* node due to unit displacements pertaining to **DOF** of *j* node.

$$[K] = \sum_{i=1}^4 \sum_{j=1}^4 \left[[K_{ij}]_m + [K_{ij}]_s \right]$$

where, $[K_{ij}]_m = \int_v [B_{mi}]^T [C_m] [B_{mj}] dv$

$$[K_{ij}]_s = \int_v [B_{si}]^T [C_s] [B_{sj}] dv$$

since, $[B_{mi}] = [B_{1mi}] + t[B_{3mi}]$

$$[B_{si}] = [B_{1si}] + [B_{2si}] + t[B_{3si}]$$

$$[K_{ij}]_m = \int_{-1}^{+1} \int_{-1}^{+1} \int_{-1}^{+1} \begin{bmatrix} [B_{1mi}]^T [C_m] [B_{1mj}] & t[B_{1mi}]^T [C_m] [B_{3mj}] \\ t[B_{3mi}]^T [C_m] [B_{1mj}] & t^2[B_{3mi}]^T [C_m] [B_{3mj}] \end{bmatrix} J_{(r,s,t)} dr ds dt \quad (3.31)$$

integrating exactly with reference to *t*,

$$[K_{ij}]_m = \int_{-1}^{+1} \int_{-1}^{+1} \begin{bmatrix} 2[B_{1mi}]^T [C_m] [B_{1mj}] & 0 \\ 0 & \frac{2}{3}[B_{3mi}]^T [C_m] [B_{3mj}] \end{bmatrix} J_{(r,s,t=0)} dr ds \quad (3.32)$$

similarly,

$$[K_{ij}]_s = \int_{-1}^{+1} \int_{-1}^{+1} \int_{-1}^{+1} \begin{bmatrix} [B_{1si}]^T [C_s] [B_{1sj}] & [B_{1si}]^T [C_s] [B_{2sj}] + t[B_{1si}]^T [C_s] [B_{3sj}] \\ [B_{2si}]^T [C_s] [B_{1sj}] & [B_{2si}]^T [C_s] [B_{2sj}] + t[B_{2si}]^T [C_s] [B_{3sj}] \\ + & + \\ t[B_{3si}]^T [C_s] [B_{1sj}] & t[B_{3si}]^T [C_s] [B_{2sj}] + t^2[B_{3si}]^T [C_s] [B_{3sj}] \end{bmatrix} J_{(r,s,t)} dr ds dt \quad (3.33)$$

integrating exactly with reference to *t*,

$$[K_{ij}]_s = \int_{-1}^{+1} \int_{-1}^{+1} \begin{bmatrix} 2[B_{1si}]^T [C_s] [B_{1sj}] & 2[B_{1si}]^T [C_s] [B_{2sj}] \\ 2[B_{2si}]^T [C_s] [B_{1sj}] & 2[B_{2si}]^T [C_s] [B_{2sj}] \\ + & + \\ \frac{2}{3}[B_{3si}]^T [C_s] [B_{3sj}] & \end{bmatrix} J_{(r,s,t=0)} dr ds \quad (3.34)$$

Here we note that $[C_s]$ is always premultiplied by part of $[B]$ with i subscript after transposing and post multiplied by part of $[B]$ with j subscript without transposing. In above i or j indicate that N_i (pertaining to i or j node) is used.

Since element is 4 noded and **DOF** associated per node is six, $[K_{ij}]_m$ or $[K_{ij}]_s$ are of size 6×6 and element stiffness matrix of 24×24 .

$$[K] = \begin{bmatrix} [K_{11}] & [K_{12}] & [K_{13}] & [K_{14}] \\ [K_{21}] & [K_{22}] & [K_{23}] & [K_{24}] \\ [K_{31}] & [K_{32}] & [K_{33}] & [K_{34}] \\ [K_{41}] & [K_{42}] & [K_{43}] & [K_{44}] \end{bmatrix} \quad (3.35)$$

$[K]$ for membrane and bending is calculated and summed with that calculated for transverse shear.

3.9 Torsional Stiffness

In previous formulation, no stiffness corresponding to torsional rotation degree of freedom exists in the local axes system.

For the four noded shell element, the rotation of the normal to the midsurface and midsurface field are independent. Let the mid-surface displacements be u' and v' Fig. 3.2.

Then,

$$\text{rotation of mid-surface} = \frac{1}{2} \left(\frac{\partial v'}{\partial x'} - \frac{\partial u'}{\partial y'} \right)$$

$$\text{rotation of normal} = \alpha_3$$

Assumed governing strain energy due to torsional rotation of normal from midsurface is,

$$U_T = \alpha_3 GH \int_A \int \left[\alpha_3 - \frac{1}{2} \left(\frac{\partial v'}{\partial x'} - \frac{\partial u'}{\partial y'} \right) \right]_{(r,s,t=0)}^2 dA \quad (3.36)$$

where, α_t = torsional coefficient.

If θ_{xi} , θ_{yi} , θ_{zi} are global rotation at i node, then α_{3i} , the rotation abut z' at node i is given as,

$$\alpha_{3i} = l_{3i}\theta_{xi} + m_{3i}\theta_{yi} + n_{3i}\theta_{zi} \quad (3.37)$$

if α'_3 is rotation at any point $(r, s, t=0)$ then,

$$\begin{aligned} \alpha'_3 &= \sum_{i=1}^4 N_i (l_{3i}\theta_{xi} + m_{3i}\theta_{yi} + n_{3i}\theta_{zi}) \\ &= \sum_{i=1}^4 N_i [l_{3i} \quad m_{3i} \quad n_{3i}] \begin{Bmatrix} \theta_{xi} \\ \theta_{yi} \\ \theta_{zi} \end{Bmatrix} \end{aligned} \quad (3.38)$$

with Eqn. (3.20)

$$\begin{aligned} \frac{\partial u'}{\partial y'} - \frac{\partial v'}{\partial x'} &= \sum_{i=1}^4 l_1 \left(l_2 \frac{\partial N_i}{\partial x} + m_2 \frac{\partial N_i}{\partial y} + n_2 \frac{\partial N_i}{\partial z} \right) u_i - l_2 \left(l_1 \frac{\partial N_i}{\partial x} + m_1 \frac{\partial N_i}{\partial y} + n_1 \frac{\partial N_i}{\partial z} \right) u_i \\ &\quad + m_1 \left(l_2 \frac{\partial N_i}{\partial x} + m_2 \frac{\partial N_i}{\partial y} + n_2 \frac{\partial N_i}{\partial z} \right) v_i - m_2 \left[l_1 \frac{\partial N_i}{\partial x} + m_1 \frac{\partial N_i}{\partial y} + n_1 \frac{\partial N_i}{\partial z} \right] v_i \\ &\quad + n_1 \left(l_2 \frac{\partial N_i}{\partial x} + m_2 \frac{\partial N_i}{\partial y} + n_2 \frac{\partial N_i}{\partial z} \right) w_i - n_2 \left(l_1 \frac{\partial N_i}{\partial x} + m_1 \frac{\partial N_i}{\partial y} + n_1 \frac{\partial N_i}{\partial z} \right) w_i \\ &= \sum_{i=1}^4 \begin{bmatrix} l_1 B'(2,i) & m_1 B'(2,i) & n_1 B'(2,i) \\ -l_2 B'(1,i) & -m_2 B'(1,i) & -n_2 B'(1,i) \end{bmatrix} \begin{Bmatrix} u_i \\ v_i \\ w_i \end{Bmatrix} \\ \alpha'_3 - \frac{1}{2} \left(\frac{\partial v'}{\partial x'} - \frac{\partial u'}{\partial y'} \right) &= \frac{1}{2} \sum_{i=1}^4 \begin{bmatrix} l_1 B'(2,i) & m_1 B'(2,i) & n_1 B'(2,i) \\ l_2 B'(1,i) & m_2 B'(1,i) & n_2 B'(1,i) \end{bmatrix} \begin{Bmatrix} u_i \\ v_i \\ w_i \end{Bmatrix} \\ &\quad + \sum_{i=1}^4 N_i [l_{3i} \quad m_{3i} \quad n_{3i}] \begin{Bmatrix} \theta_{xi} \\ \theta_{yi} \\ \theta_{zi} \end{Bmatrix} \end{aligned}$$

Now if $[K]_t$ is Torsional Stiffness matrix,

$$U_t = \{d\}^T [K]_t \{d\} \quad \text{where } \{d\}^T = \{u_i \ \theta_{xi}\}$$

$$= \infty_t G h \int_A \left[\alpha_3 - \frac{1}{2} \left(\frac{\partial v'}{\partial x'} - \frac{\partial u'}{\partial y'} \right) \right]_0^2 dA$$

so if we define,

$$[R_{mi}] = \frac{1}{2} \begin{bmatrix} l_1 B'(2,i) & m_1 B'(2,i) & n_1 B'(2,i) \\ - & - & - \\ l_2 B'(1,i) & m_2 B'(1,i) & n_2 B'(1,i) \end{bmatrix} \quad (3.39)$$

$$[R_{ni}] = N_i [l_{3i} \ m_{3i} \ n_{3i}] \quad (3.40)$$

then,

$$[K_{ij}] = \infty_t G H \int_{-1}^{+1} \int_{-1}^{+1} \begin{bmatrix} [R_{mi}]^T & [R_{mj}]^T & [R_{ni}]^T & [R_{nj}]^T \\ [R_{ni}] & [R_{mj}] & [R_{ni}] & [R_{nj}] \end{bmatrix} |a| dr ds \quad (3.41)$$

where,

$$|a| = \left| \begin{bmatrix} l_1 \\ m_1 \\ n_1 \end{bmatrix}_{(r,s,t=0)} \times \begin{bmatrix} l_2 \\ m_2 \\ n_2 \end{bmatrix}_{(r,s,t=0)} \right| \quad (3.42)$$

$$= |J| \times \sqrt{(J_{13}^*)^2 + (J_{23}^*)^2 + (J_{33}^*)^2} \quad (3.43)$$

3.10 Calculation of Stress Resultants

Stress resultants are defined in Fig. 3.3

$$\{\varepsilon_x\} = \sum_{i=1}^4 [B_{1mi}] \{u_i\} + \sum_{i=1}^4 t [B_{3mi}] \{\theta_{xi}\} \quad \text{as} \quad [B_{2mi}] = [0]$$

$$\{\varepsilon_x\} = [\varepsilon_x \ \varepsilon_y \ \gamma_{xy}]$$

$$\{u_i\} = [u_i \quad v_i \quad w_i]$$

$$\{\theta_{xi}\} = [\theta_{xi} \quad \theta_{yi} \quad \theta_{zi}]$$

$$\{\sigma_x\} = \sum_{i=1}^4 [C_m \prod B_{1mi}] \{u_i\} + \sum_{i=1}^4 t [C_m \prod B_{3mi}] \{\theta_{xi}\}$$

where

$$\{\sigma_x\} = [\sigma_x \quad \sigma_y \quad \tau_{xy}]$$

if H = thickness of shell at the point h = distance of point from mid plane
then natural co-ordinate

$$t = 2h / H$$

hence,

$$dh = (H/2) dt$$

Force per unit length is calculated by integrating stress over the thickness.

Moment per unit length is calculated by integrating stress times infinitesimal area over the thickness.

After integrating we get,

$$\begin{Bmatrix} N_x \\ N_y \\ N_{xy} \end{Bmatrix} = H \sum_{i=1}^4 [C_m \prod B_{1mi}] \{u_i\} \quad \begin{Bmatrix} M_x \\ M_y \\ M_{xy} \end{Bmatrix} = (H^2/6) \sum_{i=1}^4 [C_m \prod B_{3mi}] \{\theta_{xi}\}$$

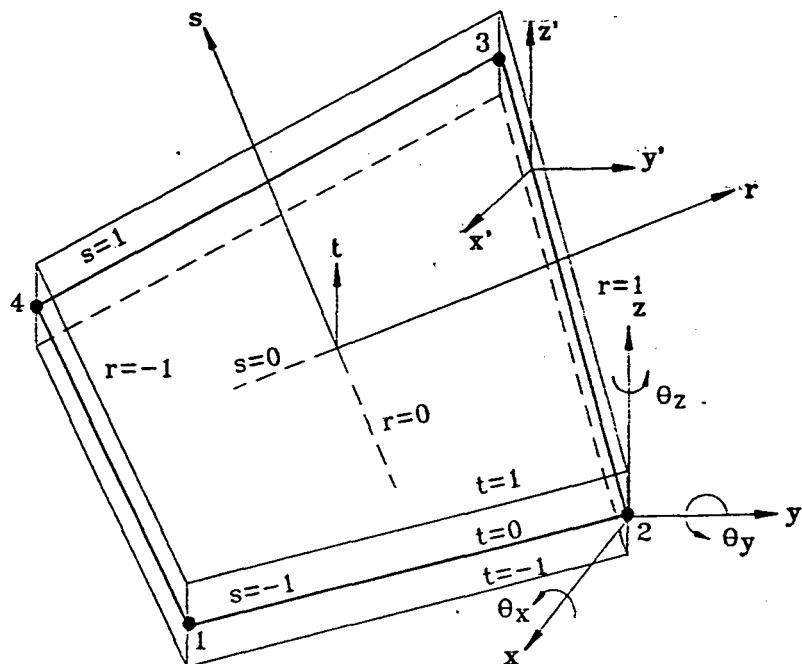


Fig 3.1

Four noded shell element

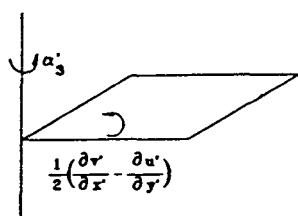
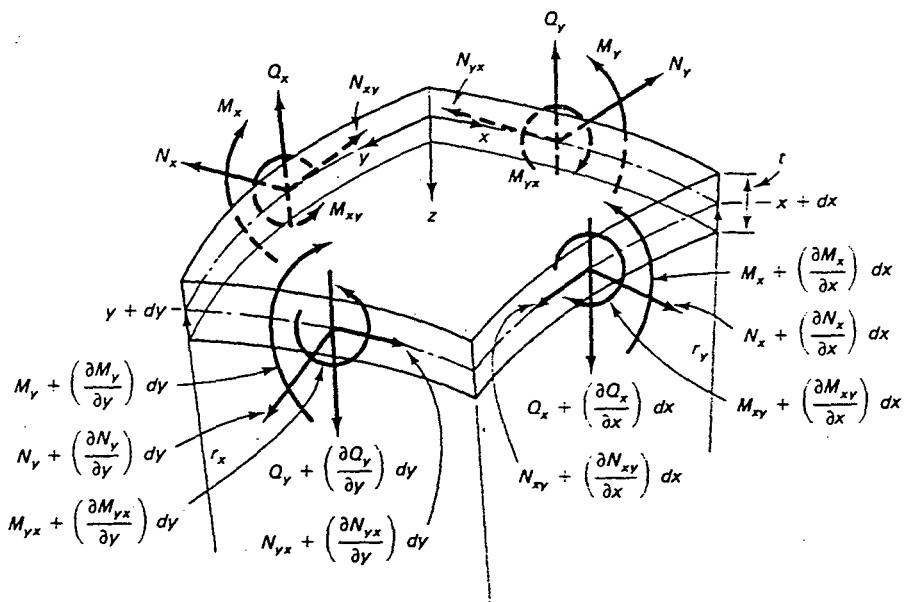
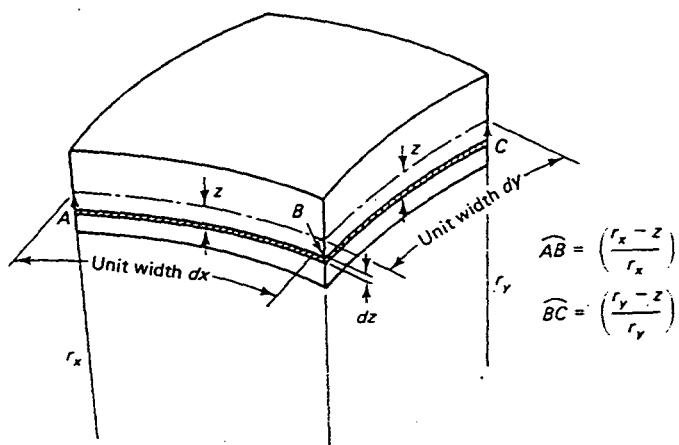


Fig 3.2

Torsional stiffness



(a)



(b)

General shell element. (a) Internal stress resultants. (b) Variation of cross section width over thickness.

Fig 3.3

Chapter 4

COMPUTER IMPLEMENTATION

4.1 Input data

4.1.1 Material Property

Material properties of shell elements are stored in 2D-array **ort**. There are three columns and as many rows as the numbers of materials in shell's finite element idealization. In first column, modulus of elasticity is stored, in second Poisson's ratio and in third density of material.

Ort	=	<table border="1"><tr><td>E₁</td><td>μ_1</td><td>$\rho_1 g$</td></tr><tr><td>E₂</td><td>μ_2</td><td>$\rho_2 g$</td></tr></table>	E ₁	μ_1	$\rho_1 g$	E ₂	μ_2	$\rho_2 g$
E ₁	μ_1	$\rho_1 g$						
E ₂	μ_2	$\rho_2 g$						

4.1.2 Nodal Coordinates

These are read from **coord.fil** for one element at a time. Nodal thicknesses also are read from same file for one element at a time.

$$x[4] = [x_i \quad x_j \quad x_k \quad x_l]$$

$$y[4] = [y_i \quad y_j \quad y_k \quad y_l]$$

$$z[4] = [z_i \quad z_j \quad z_k \quad z_l]$$

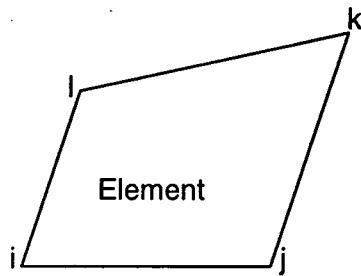
$$th[4] = [th_i \quad th_j \quad th_k \quad th_l]$$

4.1.3 Nodal Connectivity

These are read from **nop.fil** for one element at a time. From the file also read are element type and material code. Sequence of element is important to reduce **front width**.

$$nop[4] = [i \quad j \quad k \quad l]$$

4.2 Shape Function and Their Derivatives



We define a matrix **nrs**. **r** and **s** are natural coordinates of the point inside or at the boundary of the element, and are stored in **rs**, at which shape function and its derivatives are being calculated.

$$\begin{matrix} \text{nrs} & = & \begin{array}{|c|c|} \hline -1 & -1 \\ \hline +1 & -1 \\ \hline +1 & +1 \\ \hline -1 & +1 \\ \hline \end{array} & \begin{array}{l} \text{i node} \\ \text{j node} \\ \text{k node} \\ \text{l node} \end{array} \\ & & \begin{matrix} \text{r} & \text{s} \end{matrix} & \end{matrix}$$

$$N_i = n_i = 0.25 * (1 + nrs[i][0] * rs[0]) * (1 + nrs[i][1] * rs[1])$$

$$\frac{\partial N_i}{\partial r} = dnrs[i][0] = 0.25 * nrs[i][0] * (1 + nrs[i][1] * rs[1])$$

$$\frac{\partial N_i}{\partial s} = dnrs[i][1] = 0.25 * nrs[i][1] * (1 + nrs[i][0] * rs[0])$$

$$dnrs[4][2] = \begin{bmatrix} \frac{\partial N_1}{\partial r} \frac{\partial N_1}{\partial s} \\ \frac{\partial N_2}{\partial r} \frac{\partial N_2}{\partial s} \\ \frac{\partial N_3}{\partial r} \frac{\partial N_3}{\partial s} \\ \frac{\partial N_4}{\partial r} \frac{\partial N_4}{\partial s} \end{bmatrix}$$

$$n_i [4] = [N_1 \quad N_2 \quad N_3 \quad N_4]$$

4.3 Calculation of [C_m]

It is directly filled by finding material property of element from *ort* after material code is read.

$$[C_m] = \frac{E}{(1-\mu^2)} \begin{bmatrix} 1 & \mu & 0 \\ \mu & 1 & 0 \\ 0 & 0 & \frac{1-\mu}{2} \end{bmatrix}$$

4.4 Calculation of Direction cosine matrix [D]

$d = [D]$ is calculated at any point on midsurface ($r,s,t = 0$). Components of vectors along r direction and s direction, and tangent to midsurface can be calculated as;

$$\frac{\partial x}{\partial r} = \sum_{i=1}^4 \frac{\partial N_i}{\partial r} x_i \quad \frac{\partial x}{\partial s} = \sum_{i=1}^4 \frac{\partial N_i}{\partial s} x_i$$

Vector Cross Product

It is done in matrix form as given below

$$\begin{Bmatrix} a_x \\ a_y \\ a_z \end{Bmatrix} \times \begin{Bmatrix} b_x \\ b_y \\ b_z \end{Bmatrix} = \begin{Bmatrix} (a_y.b_z - a_z.b_y) \\ (a_z.b_x - a_x.b_z) \\ (a_x.b_y - a_y.b_x) \end{Bmatrix}$$

Step 1. First column of $d/d1$ is filled with derivatives of x,y,z w.r.t. r and second column w.r.t. s . Cross product elements of vectors of first and second columns are then filled in third column and divided by its magnitude as per Eqn. (3.4).

Step 2. First column is again filled, after again initializing, with derivatives of x,y,z w.r.t. r at ($r=0,s=0,t=0$). Cross product elements of vectors of first and third columns are filled in second column and then divided by its magnitude as per Eqn.(3.5)

Step 3. Cross product elements of vectors of second and third columns are now filled in first column.

4.5 Jacobian

$\mathbf{ja} = [\mathbf{J}]$ is calculated at Gauss Point ($r,s,t=0$). With the known values of N_i

$$\frac{\partial N_i}{\partial r}, \frac{\partial N_i}{\partial s}$$

$x[i], y[i], z[i], th[i]$

we directly fill all the nine elements of jacobian as per Eqn.(3.10).

4.5.1 Jacobian Inverse

$\mathbf{ji} = [\mathbf{J}^{-1}]$ is obtained by directly calculating cofactors of each element and filling them into transposed location and dividing by determinant of jacobian.

4.5.2 Determinant of Jacobian

$$d\mathbf{J} = |\mathbf{J}| = a_1 A_1 + a_2 A_2 + a_3 A_3$$

Where a_1, a_2 and a_3 are the values of first column elements of jacobian and A_1, A_2 , and A_3 are values of cofactors of these elements.

4.6 Strain Displacement Matrix Units

$$\mathbf{b} = [B_{1mi}], [B_{3mi}], [B_{1si}], [B_{2si}], [B_{3si}], [R_{mi}], [R_{ni}]$$

These units are calculated for particular value of i (1,2,3,4), i.e., for shape function corresponding to node i . These are calculated at a Gauss Point. Jacobian at Gauss point, direction cosine matrix at Gauss point and direction cosine matrix at the nodes are calculated first. If j specifies node number, following are calculated.

$$N_j = nj, \quad nrj = \frac{\partial N_j}{\partial r}, \quad nsj = \frac{\partial N_j}{\partial s}$$

Derivatives of N_i w.r.t. x , y and z are calculated as

$$\frac{\partial N_i}{\partial x} = J_{11}^* \frac{\partial N_i}{\partial r} + J_{12}^* \frac{\partial N_i}{\partial s}$$

and

$$B'(1,i) = b1i = l_1 \frac{\partial N_i}{\partial x} + m_1 \frac{\partial N_i}{\partial y} + n_1 \frac{\partial N_i}{\partial z}$$

Similarly B' (2,i) and B' (3, i) are calculated. With these general calculation any of 7 units of strain displacement matrix can be calculated as per Eqn. (3.22), (3.24), (3.25), (3.26), (3.27),(3.39) and (3.40). With specific value of I from among (1,2,3,4,5,6,7) **bce** subroutine returns **b**, the desired strain matrix unit in the sequence shown above.

4.7 Calculation of Stiffness Matrix

Stiffness matrix is formed as sum of **three** matrices.

First for inplane strains

$$(\varepsilon_x, \varepsilon_y, \gamma_{xy})$$

Second for transverse shear strains

$$(\gamma_{yz}, \gamma_{zx})$$

Third for torsional rotations

$$\left\{ \alpha_3 - \frac{1}{2} \left(\frac{\partial v'}{\partial x'} - \frac{\partial u'}{\partial y'} \right) \right\}$$

Integration for first is (2 x 2), and for second and third it is one point, in numerical integration scheme of Gauss Point value and their weight.

As per Eqn. (3.35) each of three stiffness matrices of 24×24 size, are composed of 4×4 numbers of units $[K_{ij}]$ each of size 6×6 . Here i, j of $[K_{ij}]$ varies from 1 to 4 i.e. to cover all the nodes of one element one by one.

All the three stiffness matrices are made separately in different subroutines and are summed to finally obtain complete element stiffness matrix.

4.8 Assembly and Solution

This has been done as per *Frontal Solution Technique* developed as Seminar Topic.

4.9 Subroutines

Here we shall proceed in the backward direction i.e., from subroutine required in the last, in a sequence of calling them.

cee : This takes the material code *imat* as input and creates $[C_m]$, with the help of *ort*.

d_cosine : This takes Gauss Point or Node point coordinates in natural coordinate system, as input, and with the global coordinates of nodes of the element in hand, it creates $d=[D]$ matrix for the point.

jacobian : This takes Gauss Point coordinates in the natural coordinate system, as input, and with the global coordinates of nodes of the element in hand and $[D]$ (by calling *d_cosine*), it creates jacobian inverse *ji* and determinant *dj*.

bee : This takes Gauss Point coordinates in natural coordinates system, i or j of $[K_{ij}]$ of Eqn. (3.35) and an integer l specifying which of the basic unit out of seven (defined by Eqn. 3.22, 3.24, 3.25, 3.26, 3.27, 3.39 and 3.40) is required, as input, and creates the required unit *b*.

st1a_shell : Here *st* is for stiffness, **1** for 4 noded bilinear degenerate element, **a** for in plane strain part and **shell** for shell.

It takes only element nodes' global coordinates in sequence as input. It uses 3 loops, first for Gauss points (uses 2x2 integration scheme) another two for i and j of $[K_{ij}]$. i varies from 1 to 4 whereas j varies from i to 4.

It executes Eqn. 3.32 for every set of i and j to create $[K_{ij}]$, in inner two loops and transfers it in esm1 (element stiffness matrix) as per Eqn. 3.35.

st1b_shell : Here b is for transverse shear strain part of stiffness, and rest all are same as for above.

It operates in similar fashion to ***st1a-shell*** but executes Eqn.(3.34) for every set of i and j to create $[K_{ij}]$, in inner two loops. Uses single point Gauss integration scheme.

st1c_shell : Here c is for torsional rotational strain part of stiffness and rest all are same as for above.

If operates in similar fashion to ***st1a-shell*** but executes Eqn. (3.41) for every set of i and j to create $[K_{ij}]$ in inner two loops. Uses single point Gauss integration scheme.

stiff1 : It calls ***st1a_shell***
 st1b_shell
 st1c_shell
After initialising element stiffness matrix.

stiff : It call ***stiff1*** if element type ***itype*** is 1.

gdata : It calls ***data1***
 data2
 data3
 data4
 data5
 data6 sequentially.

- data1** : It reads number of elements, number of node points, number of restraints, number of restrained joints, number of materials and element types from **fem.fil**. Then reads material properties and fills **ort**.
- data2** : It reads nodal coordinates from **fem.fil** and stores in separate file **coord.fil** for rereading.
- data3** : It reads nodal connectivity from **fem.fil**, element type and material type for all the elements and stores in separate file **nop.fil** for rereading.
- data4** : It reads boundary conditions at restrained joints in **jrl**. And displacements / settlements of boundary in **jdl**.
- data5** : It reads action at joints in **aj**.
- data6** : It read **u.d.l** over the element and its **type**.
- front_sol** : It uses a loop for covering all the elements. It calls different subroutines to create element stiffness matrix, assembling and simultaneous starring according to **Frontal Solution Technique** and finds joint displacements and support reaction in back substitution phase.
- results** : It calculates N_x N_y N_{xy} M_x M_y M_{xy} from displacements of joints.
- graf** : It calls **graf1**, **graf2**, **graf3** sequentially. These graf assist in checking the input data in easiest pictorial form.
- graf 1** : It shows shell grid in plan (xy plane) without taking into account effect of z along with connectivity of elements and element numbers.
- graf 2** : Shows isometric view of the grid.
- graf 3** : It shows grid along with boundary condition. A line passing through the node, on the right side of which, in a vertical column six number are shown (zero or one) as per their being free or restrained for u_i v_i w_i θ_{xi} θ_{yi} θ_{zi} .

4.10 Files and Variables

- **4.10.1** Following files have been used for storing data, used by **SHELL.C** the C Language Computer Code implementing the formulation of Chapter 3.
- fem.fil** : It stores all the input data pertaining to mesh, material, nodal coordinates, thicknesses at nodes, nodal connectivity, element type, material type, boundary nodes and restraint at the nodes, type of loading and loading.
- coord.fil** : It stores coordinates and thicknesses of nodes sequentially after reading from **fem.fil**.
- nop.fil** : It stores element type, material type and nodal connectivity in counter clockwise direction of all elements sequentially.
- gstif.fil** : During forward elimination in **Frontal Solution Technique** of **Linear Simultaneous Equations**, fully starred equations' coefficients and R.H.S. and other relevant information (NIKNO and h) pertaining to all variables (eliminated one by one) are stored in it. These information are reused during backsubstitution.
- res.fil** : Displacements and reactions calculated during back substitution phase are stored in it, to be used later for calculating stress resultants.
- d.t** : This is the output file where all the processed analysis results are stored.

Following are the variables used in **SHELL.C**. These have been divided into Global and Local if they are meant to be accessed by more than one subroutine and only one subroutine respectively. These are subdivided into two parts, one Floating Point Variable and other Integer.

4.10.2 Global Variables

4.10.2.1 Floating Point Variables

X[4], y[4], : These store global nodal coordinates and thicknesses for one element at a time in their nodal connectivity sequence.

cee[3][3] : This is $[c_m]$ the constitutive matrix pertaining to inplane stresses.

ort[nmat][3] : Stores material property.

aj[np x 6] : Stores actions at joints in $u_i, v_i, w_i, \theta_{xi}, \theta_{yi}, \theta_{zi}$ order.

ael[ncn x 6] : Stores equivalent joint load calculated by ldata subroutine.

ar[np x 6] : Reactions at joints' restrained degree of freedom

ac[M_fron] : R.H.S. of equations involving active variables.

dj[np x 6] : Displacements corresponding to all **DOF**

jdl [np x 6] : Settlements corresponding to all **DOF**.

gp[M_fron x (M_fron + 1) / 2]

: Stores elements of assembled stiffness matrix of active variable.

esm1 [2 x (24 + 1) / 2] = esm1 [300]

: Stores coefficients of element stiffness matrix during execution of st1a_shell, st1b_shell, st1c_shell.

equat[M_fron] : Stores stiffness coefficient extracted from **gp** but not yet stored in file and used in starring operation and backsubstitution.

vec[M_fron] : Stores known **values** other than zeros of active **variables** during back substitution.

4.10.2.2 Integer Variables

- jrl[np x 6]*** : Stores boundary condition corresponding to all ***DOF***.
- nacva[M_fron]*:** Stores active variables' global addresses in forward pass.
- locl [6 x ncn] :*** Global address of variables associated with newly activated element alongwith their signs.
- ndest [6 x ncn]:*** Addresses for ***locl*** variables in ***nacva***.
- nop[4]*** : Nodal connectivity of the element when node numbering starts with zero.
- nop1[4]*** : Nodal connectivity of the element when node numbering starts with one and are signed (positive or negative) if they are not appearing or appearing for the last time.
- itype*** : Variable to read (1, 2, 3) type of loading to choose (lData1, lData2, lData3) subroutine to calculate equivalent joint load.
- DOF*** : Degree of Freedom
- ne*** : Number of elements in the grid.
- np*** : Number of node points in the grid.
- nrj*** : Number of restrained joints.
- ncn*** : Number of nodes per element
- M_fron*** : Maximum front width
- imat*** : Material type
- NIKNO*** : Variable being eliminated in forward pass.
- kelva*** : Counter for variables eliminated

4.10.3 Local Variables

4.10.3.1 Floating Point Variables

- dj*** : Determinant of Jacobian
- ji[3][3]*** : Inverse jacobian
- c1s[2][2]*** : Constitutive matrix pertaining to transverse shear
- dnrj*** : $\frac{\partial N_i}{\partial r}$ for given i
- dnsj*** : $\frac{\partial N_i}{\partial s}$ for given i
- dnxzy[3]*** : Stores $\frac{\partial N_i}{\partial x}$ $\frac{\partial N_i}{\partial y}$ $\frac{\partial N_i}{\partial z}$
- d[3][3]*** : Stores direction cosine matrix pertaining to Node Point
- d1[3][3]*** : Stores direction cosine matrix pertaining to Gauss Point.
- b[3][3]*** : Stores strain displacement matrix ***unit***
- ja*** : Jacobian matrix
- gload*** : R.H.S. of extracted equation.
- pivot*** : Stiffness coefficient pertaining to variable being eliminated in extracted equation.

4.10.3.2 Integer Variables

- kdj*** : Number of joints having settlements
- idest*** : Row address in ***gp*** of the stiffness coefficient
- jdest*** : Column address in ***gp*** of the stiffness coefficient
- nsize*** : nc n x DOF per node.

Chapter 5
VERIFICATION OF FORMULATION OF
FOUR NODDED DEGENERATE BILINEAR ELEMENT

5.1 *The Test Problem*

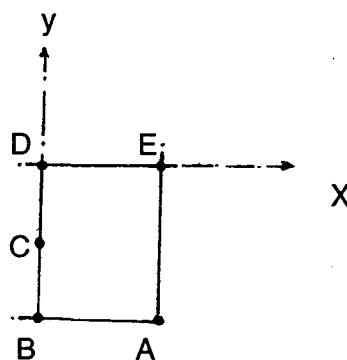
A standard problem of cylindrical barrel vault given in the book under Ref. (12) whose analytical solution by Scordelis is also plotted there has been used to test the efficacy of formulation. The problem and its results are reproduced here in Fig. 5.1 and 5.2.

5.2 *Solution By the Element*

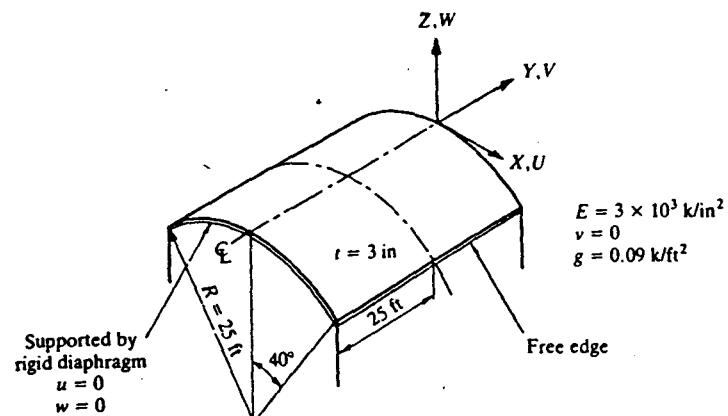
The problem was solved with the help of four noded element developed here and results are plotted in graph 5.1 to 5.4. Mesh size 16x18 for full analysis and 12 x 18 for quadrant were used. Coarser mesh were also used and found to be quite sufficient upto 8x12 mesh for full problem.

5.3 *Boundary Conditions*

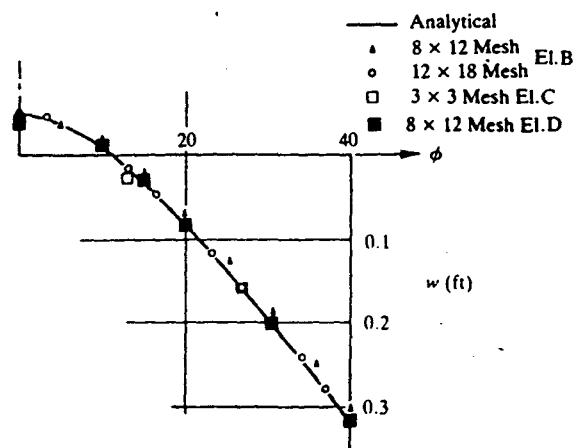
With the help of full analysis results and symmetry, boundary condition of quadrant were idealized as given below for quadrant analysis.



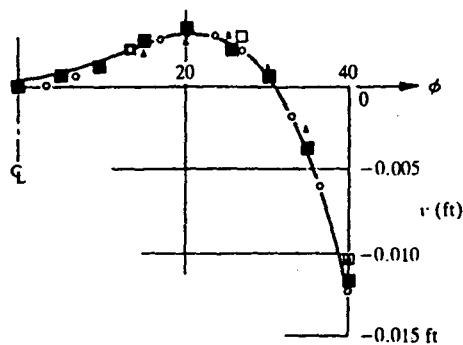
$$\begin{aligned}
 A &= [1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0] \\
 B &= [1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1] \\
 C &= [1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1] \\
 D &= [1 \quad 1 \quad 0 \quad 1 \quad 1 \quad 1] \\
 E &= [0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1]
 \end{aligned}$$



(a) Finite element and exact⁶² solutions under dead loads



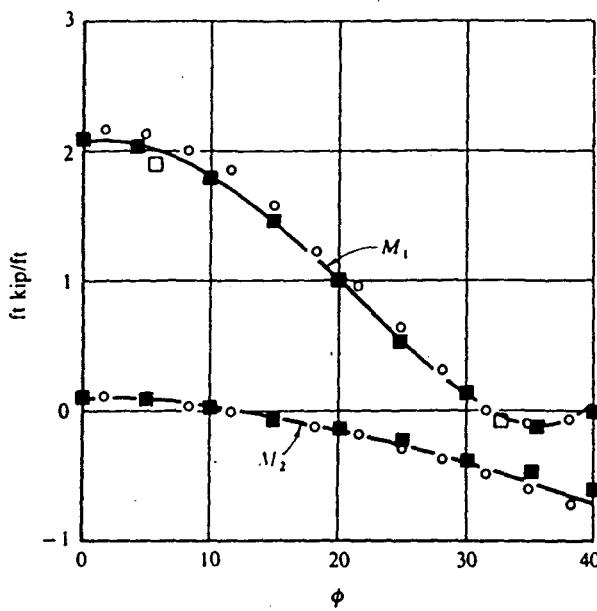
(b) Vertical displacement of central sections



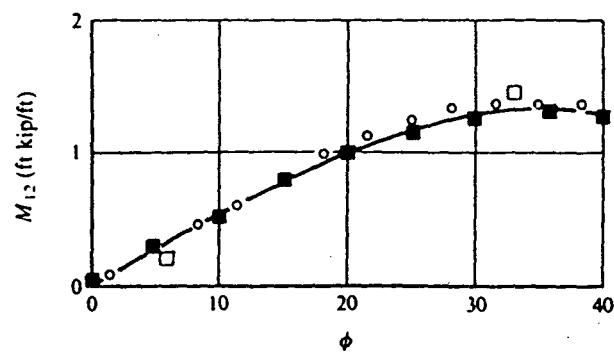
(c) Longitudinal displacement of support

A barrel (cylindrical) vault. $E = 3 \times 10^6 \text{ lb/in}^2$, $v = 0$, weight of shell = 90 lb/ft²

Fig 5.1



(a) M_1 = transverse moment, M_2 = longitudinal moment at central section

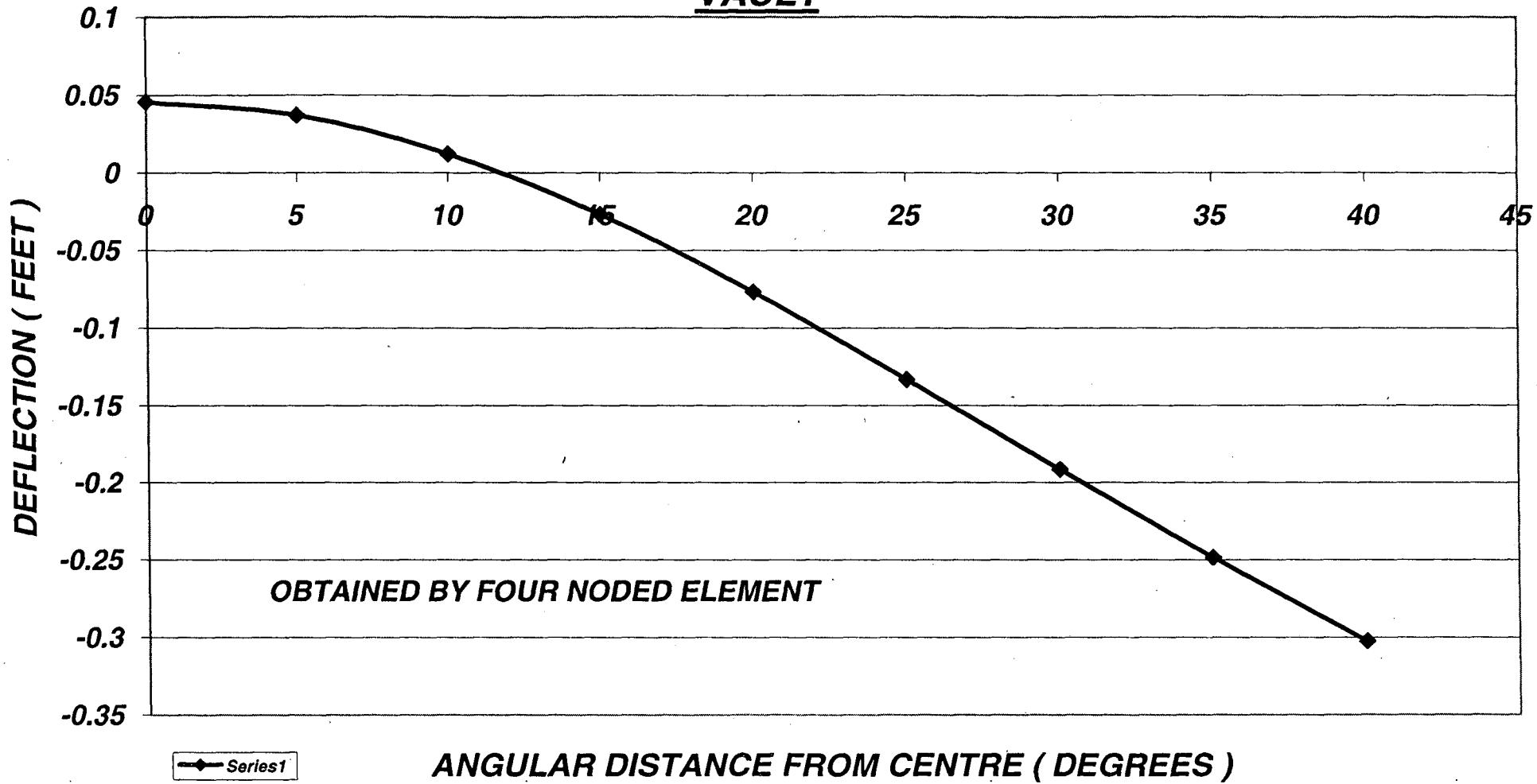


(b) M_{12} = twisting moment at support

Barrel vault

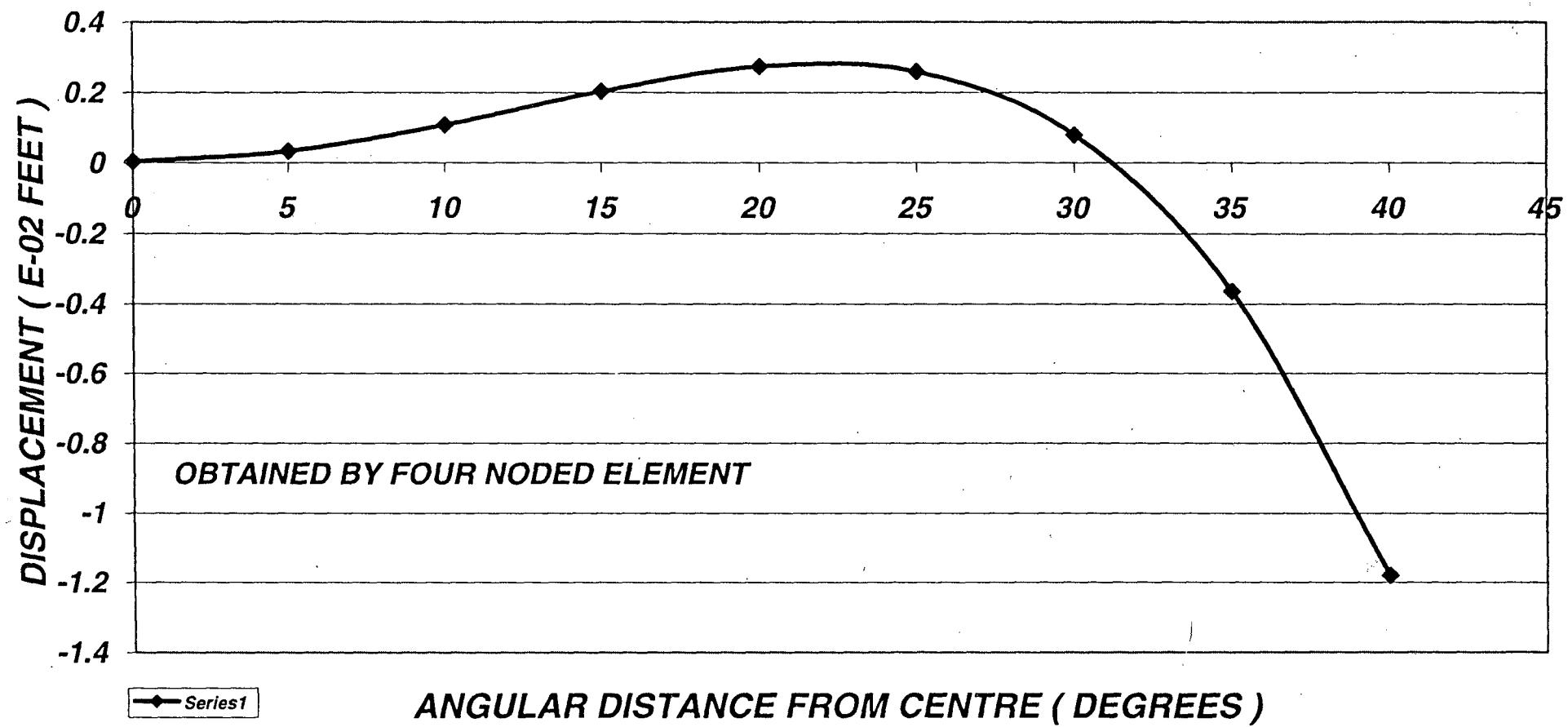
Fig 5.2

**VERTICAL DEFLECTION AT CENTRE OF CYLINDRICAL BARREL
VAULT**



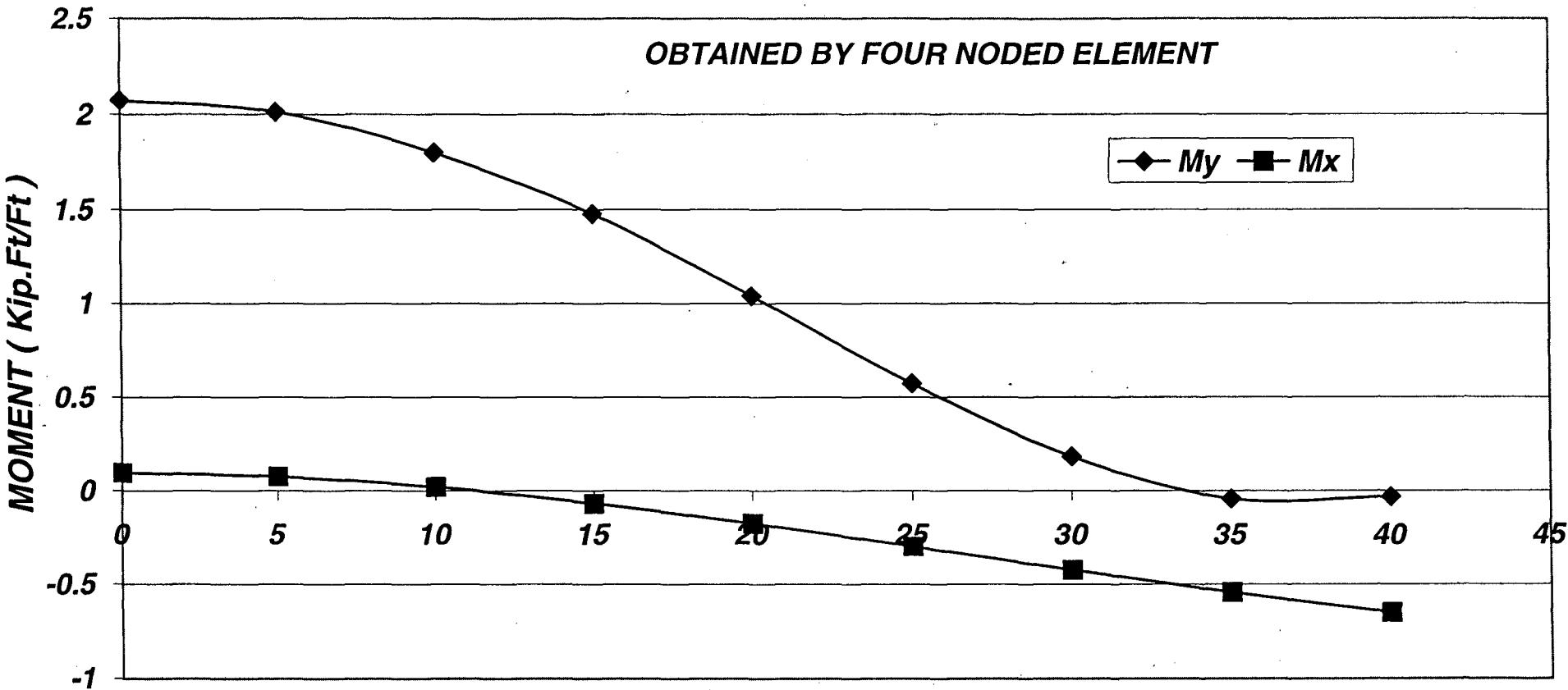
GRAPH 5.1

**LONGITUDINAL DISPLACEMENT AT THE END OF CYLINDRICAL
BARREL VAULT**



GRAPH 5.2

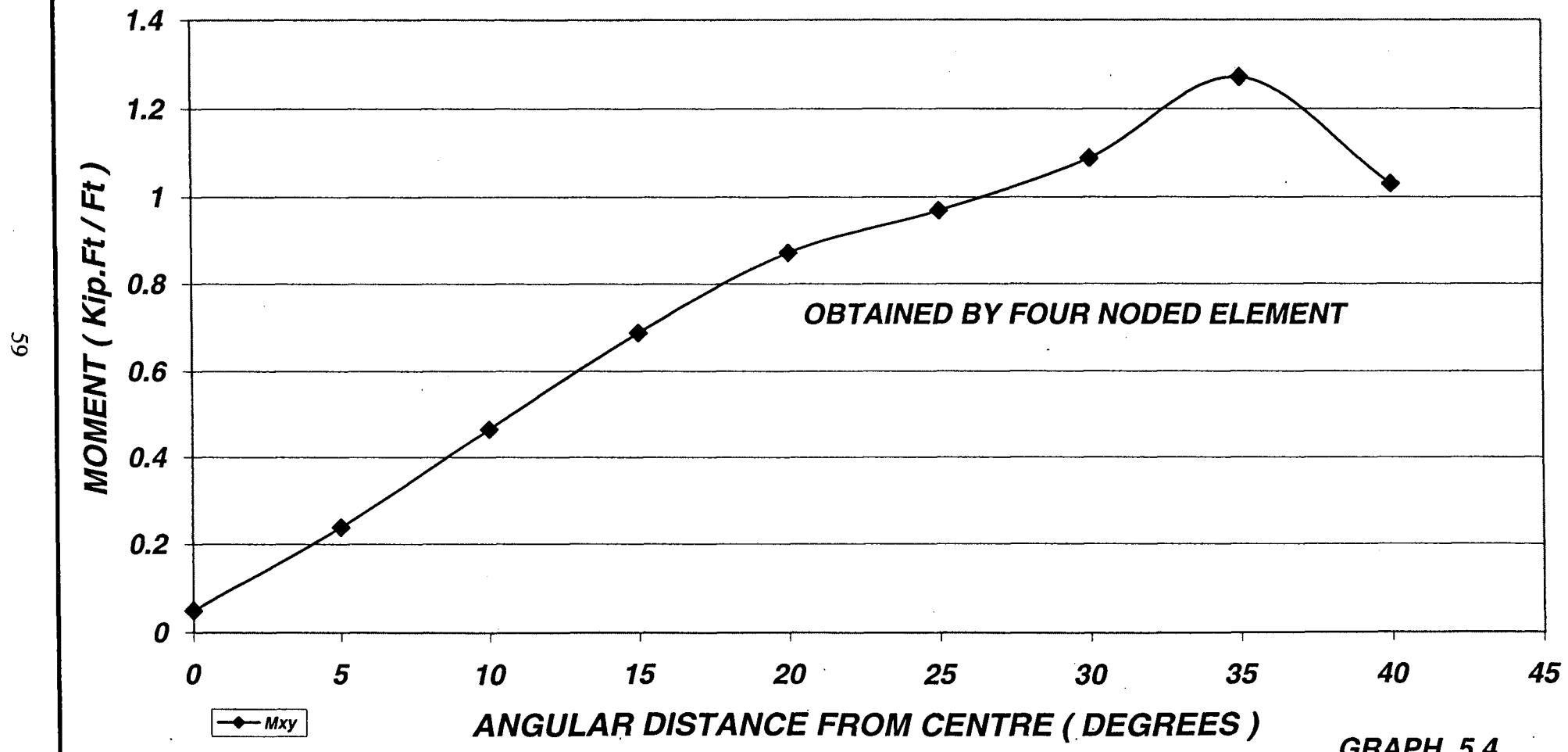
**MOMENTS AT THE CENTRE OF CYLINDRICAL
BARREL VAULT**



*M_x=LONGITUDINAL MOMENT
M_y=TRANSVERSE MOMENT*

GRAPH 5.3

**TWISTING MOMENT AT THE END OF
CYLINDRICAL BARREL VAULT**



Chapter 6
PARAMETRIC ANALYSIS OF
DOUBLY CURVED (ANTICLASTIC) SHELL

With the four noded element developed and checked for its efficacy, we analyzed the double curvature shells of following geometry.

6.1 Problem Description

6.1.1 Transverse Cross Section

For all the three spans transverse cross section is same

M	0	.1	.2	.3	.4	.5
MM	0	10	40	90	160	250

6.1.2 Longitudinal Cross Section

There are three spans, each having its own longitudinal profile explicitly prescribed with coordinates

a) Span 6M

M	$\pm X$	0	.2	.4	.6	.8	1.0	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	2.8	3.0
MM	+y	169	168	166	162	157	150	142	132	121	108	94	78	61	42	21	0

b) Span 5M

M	$\pm X$	0	.25	.5	.75	1.0	1.25	1.50	1.75	2.00	2.25	2.50
MM	+y	117	116	112	106	98	88	75	65	42	22	0

c) Span 4M

M	$\pm X$	0	0.2	0.4	0.6	0.8	1.0	1.2	1.4	1.6	1.8	2.0
MM	+y	75	74	72	68	63	56	48	38	27	14	0

6.1.3 Thickness

Each of the three geometry given above are analysed for three different Thicknesses

- a) 60mm
- b) 40mm
- c) 20mm

6.1.4 Material Property

Modulus of Elasticity (E)	=	20 GPa
Poisson's ratio (μ)	=	0.18
Density (ρ g)	=	25 KN/M ³

6.1.5 Finite Element Idealization

a) SPAN 6M

Widthwise	10 divisions
Lengthwise	30 divisions

b) SPAN 5M

Widthwise	10 divisions
Lengthwise	20 divisions

c) SPAN 4M

Widthwise	10 divisions
Lengthwise	20 divisions

6.1.6 Boundary Condition

Each having same boundary condition. On the both supports every node has u and w restrained and all other free. At central node v is restrained and all other are free.

6.1.7 Loading

All have been analyzed for ***self load*** only.

6.2 Results

Results of such analysis are plotted with the help of **EXCEL** package for **scattered xy** to give smooth curves passing through the points in graphs 6.1 to 6.33. Results are also given in tables 6.1 to 6.6.

Results for three different thickness of same span were plotted on same sheet and for three different spans in sequence for easy comparison.

Results for three thicknesses and three spans are also given in one table for comparison.

Two extreme cases of Parametric Analysis i.e., 6m span of 20mm thickness and 4m span of 60mm thickness were also analysed by **NISA** package of **EMRC**, and graphs are given in **Appendix A**.

TABLE 6.1

MAXIMUM VERTICAL DEFLECTION AT CENTRE (E-03M)

	<i>60 mm</i>	<i>40 mm</i>	<i>20 mm</i>
<i>6m</i>	3.366	3.483	3.735
<i>5m</i>	1.591	1.638	1.737
<i>4m</i>	0.675	0.693	0.714

TABLE 6.2

MAXIMUM INPLANE LONGITUDINAL FORCE (E+01 KN/M)

	<i>60 mm</i>	<i>40 mm</i>	<i>20 mm</i>
<i>6m</i>	9.702	6.618	3.360
<i>5m</i>	6.593	4.498	2.303
<i>4m</i>	4.327	2.956	1.512

TABLE 6.3

MAXIMUM INPLANE LONGITUDINAL STRESS (N/MM²)

	<i>60 mm</i>	<i>40 mm</i>	<i>20 mm</i>
<i>6m</i>	0.162	0.166	0.168
<i>5m</i>	0.110	0.113	0.115
<i>4m</i>	0.072	0.074	0.076

TABLE 6.4

MAXIMUM TRANSVERSE MOMENT (E-01 KNM/M)

	<i>60 mm</i>	<i>40 mm</i>	<i>20 mm</i>
<i>6m</i>	-1.578	-1.058	-0.525
<i>5m</i>	-0.904	-0.580	-0.274
<i>4m</i>	-0.273	-0.154	-0.062

TABLE 6.5

MAXIMUM LONGITUDINAL MOMENT (E-01 KNM/M)

	<i>60 mm</i>	<i>40 mm</i>	<i>20 mm</i>
<i>6m</i>	-3.213	-0.986	-0.133
<i>5m</i>	-2.353	-0.777	-0.141
<i>4m</i>	-1.499	-0.472	-0.070

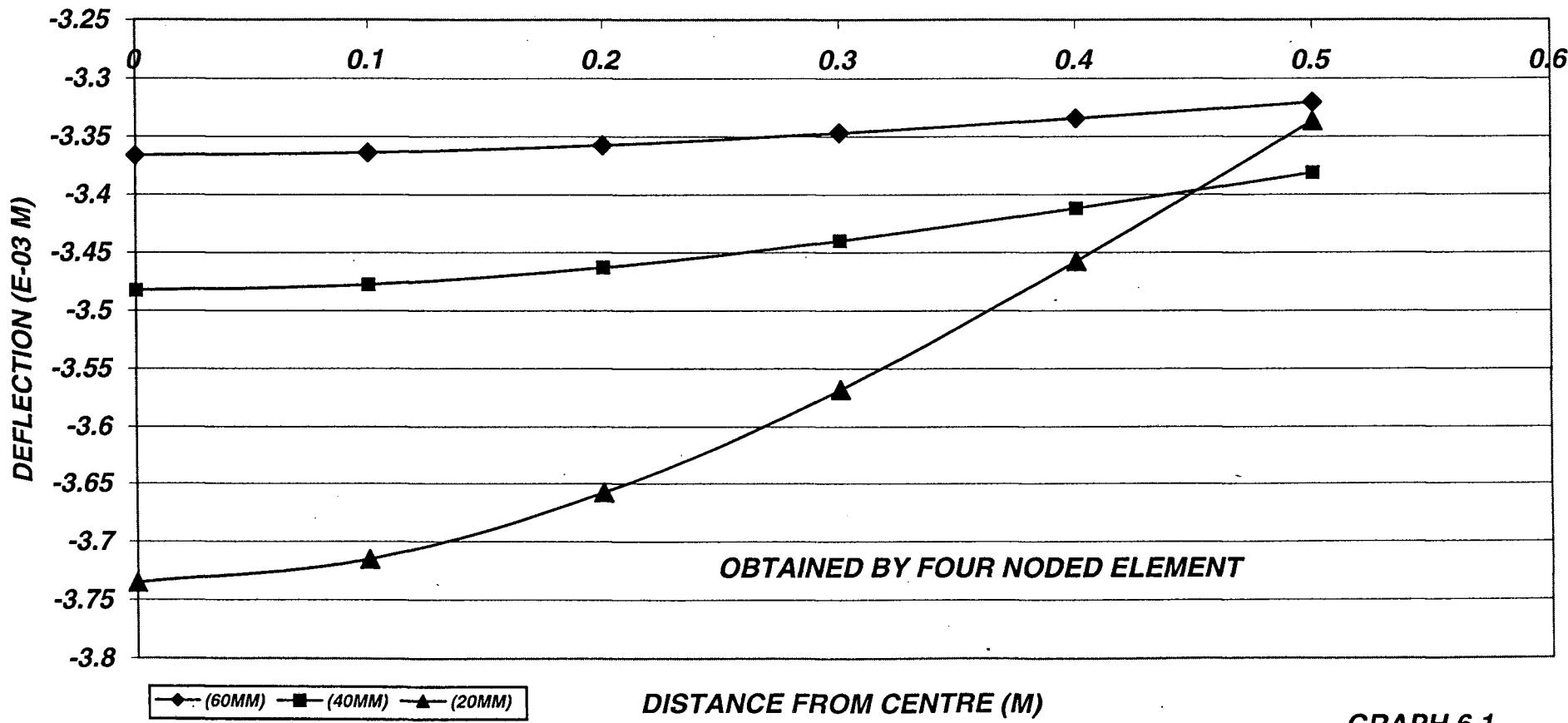
TABLE 6.6

MAXIMUM TWISTING MOMENT (E-02 KNM/M)

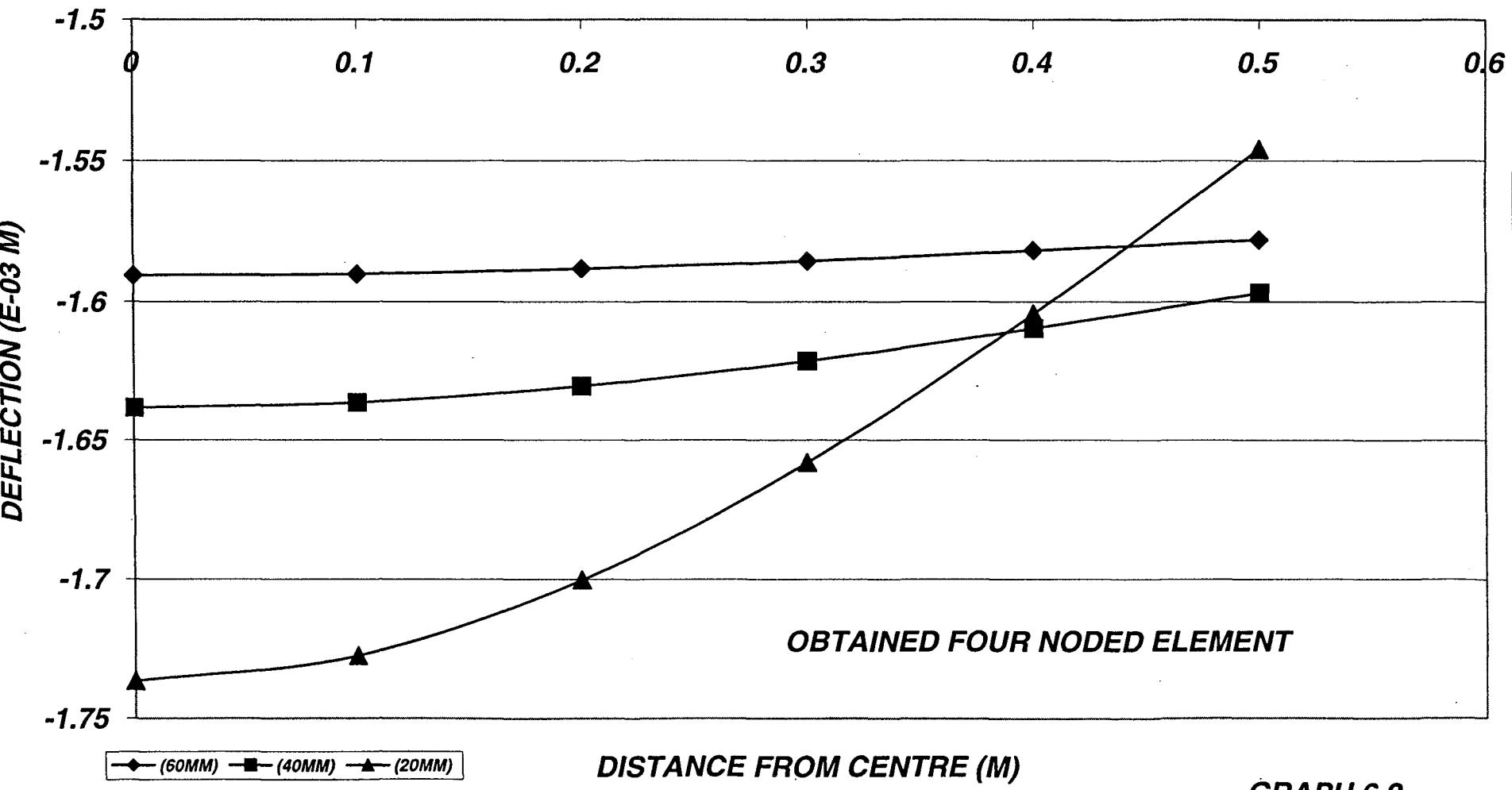
	<i>60 mm</i>	<i>40 mm</i>	<i>20 mm</i>
<i>6m</i>	6.231	0.699	0.431
<i>5m</i>	1.851	6.40 E-02*	0.222
<i>4m</i>	1.022	0.504	0.221

Results are a bit puzzling. Needs further investigation.

VERTICAL DEFLECTION AT CENTRE (6M SPAN)

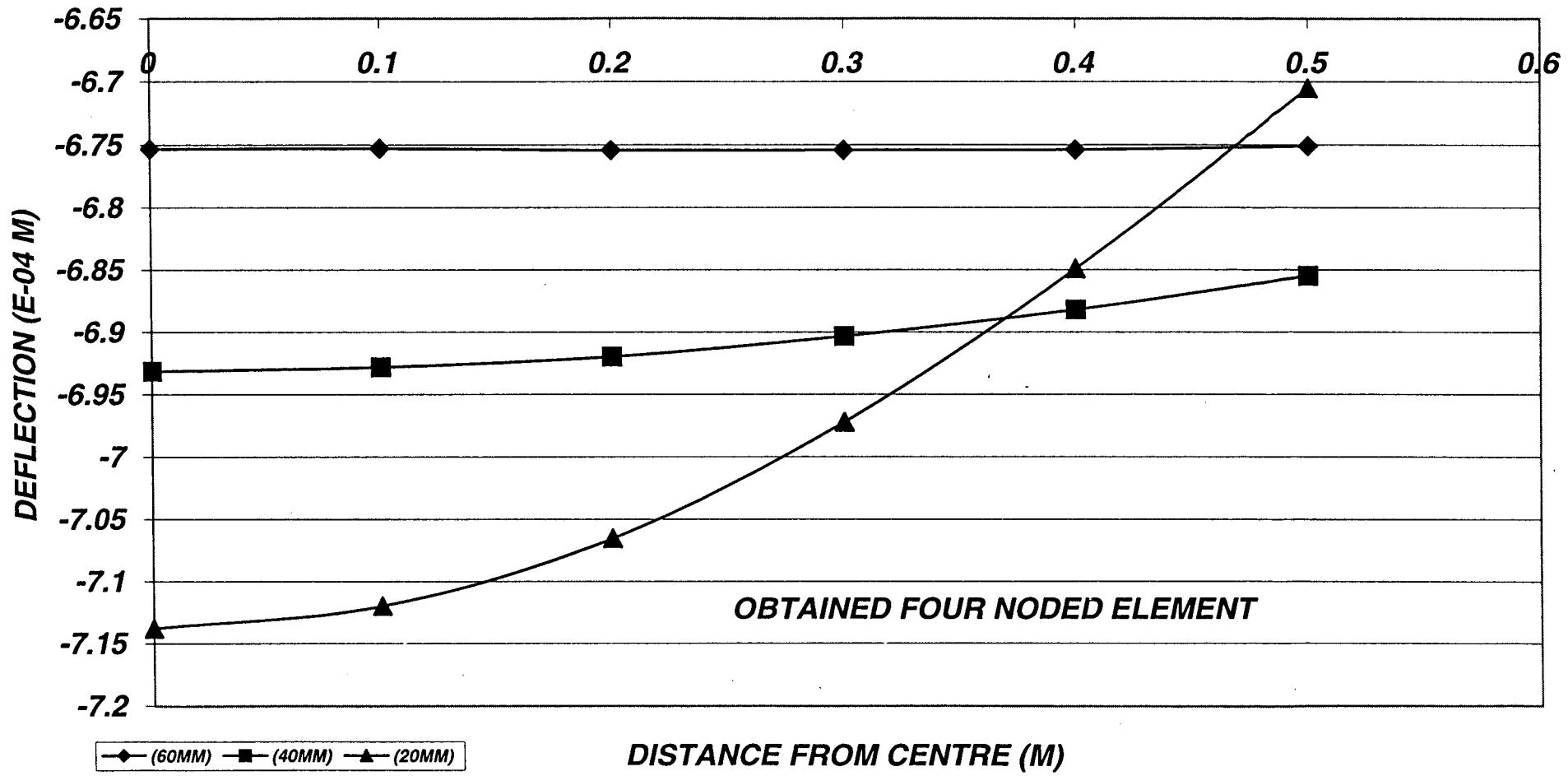


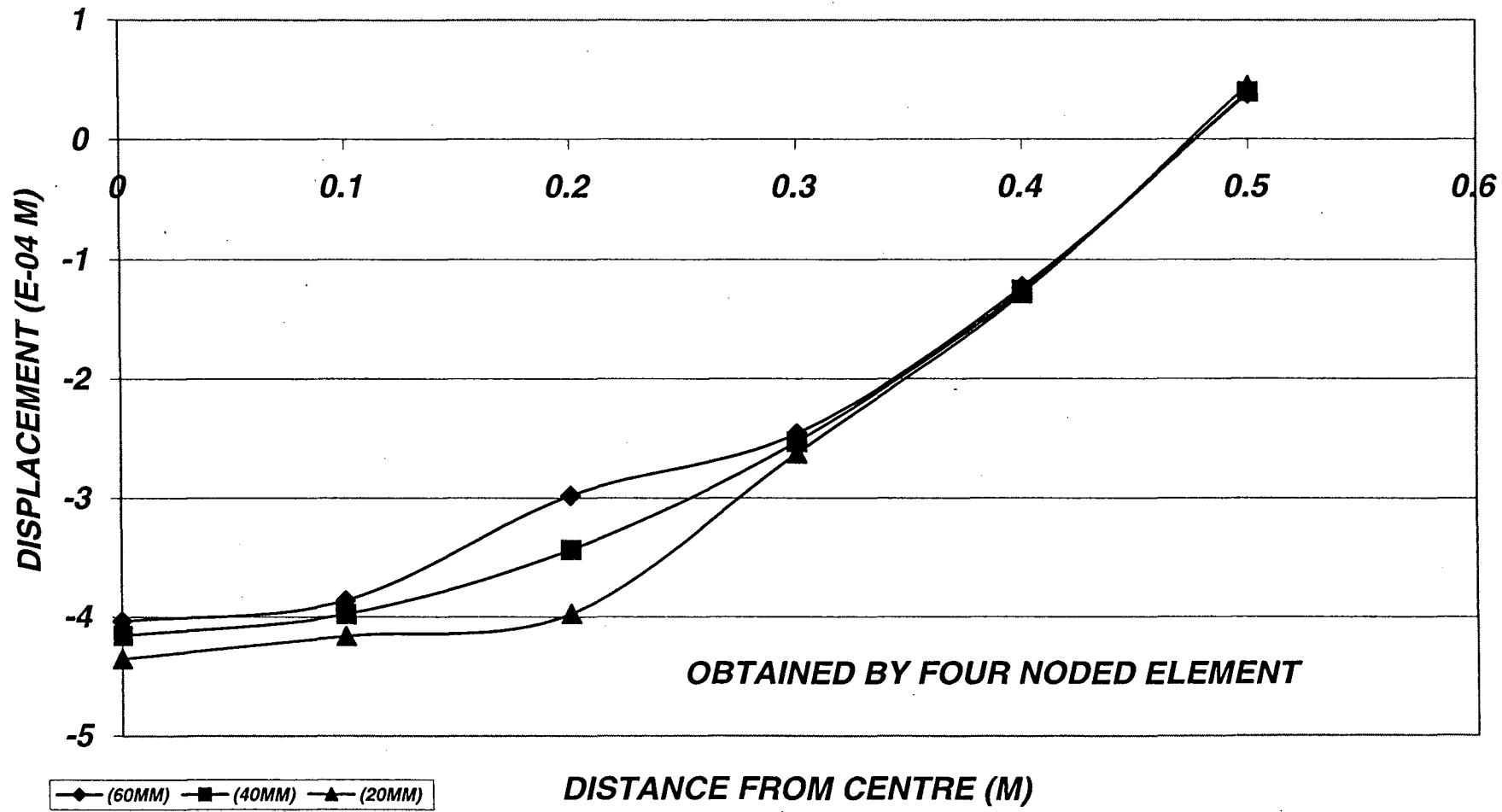
VERTICAL DEFLECTION AT CENTRE (5M SPAN)

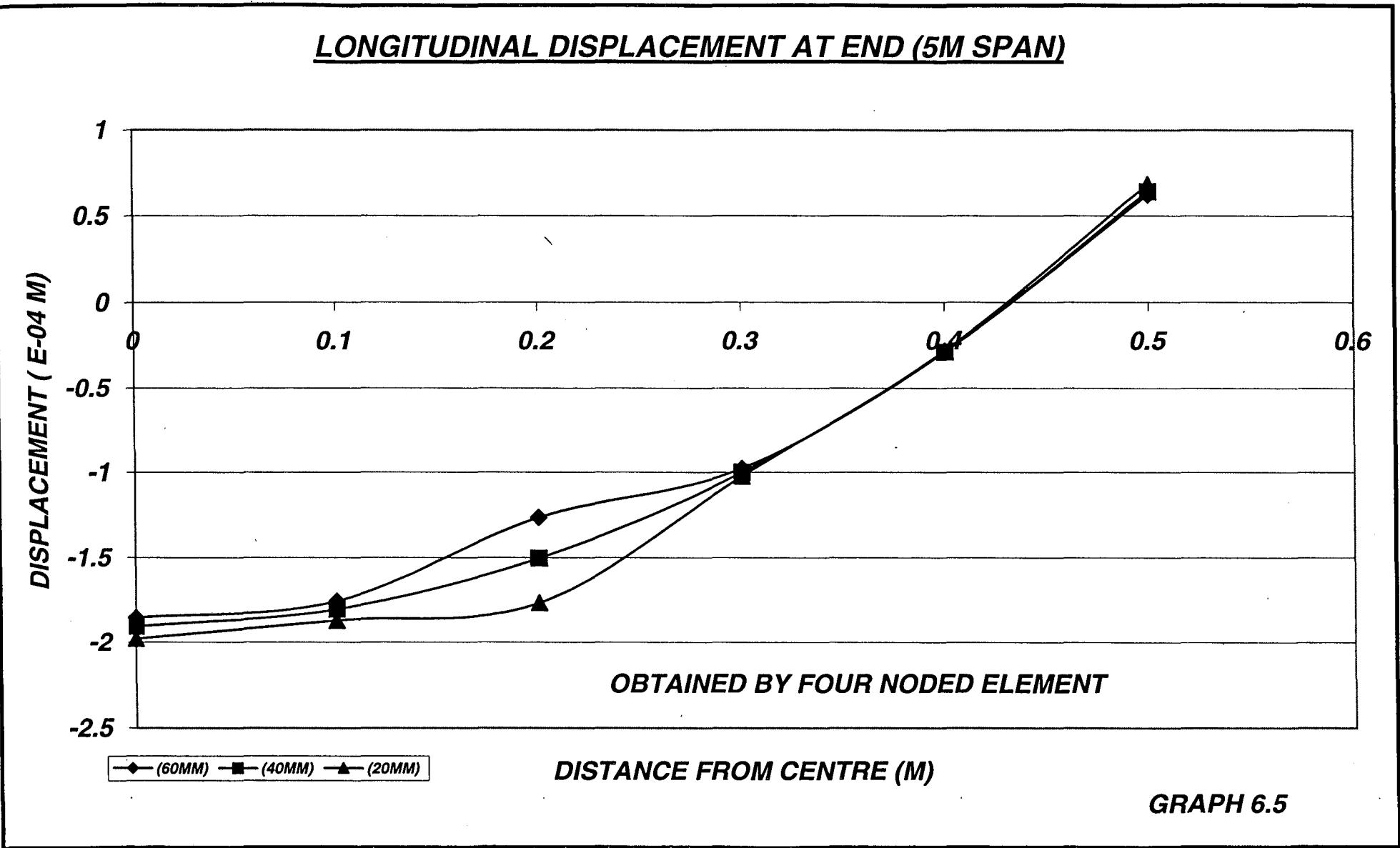


GRAPH 6.2

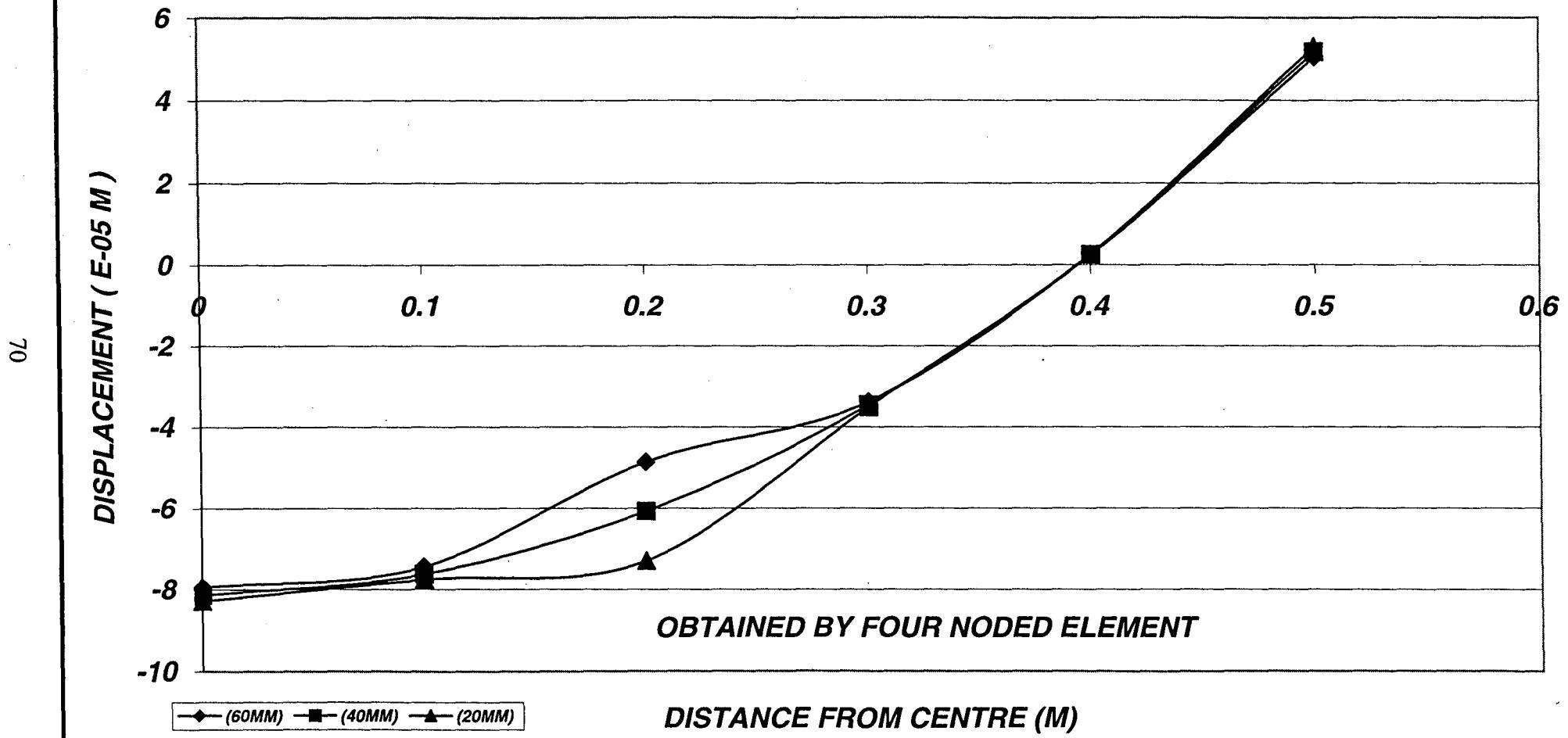
VERTICAL DEFLECTION AT CENTRE (4M SPAN)



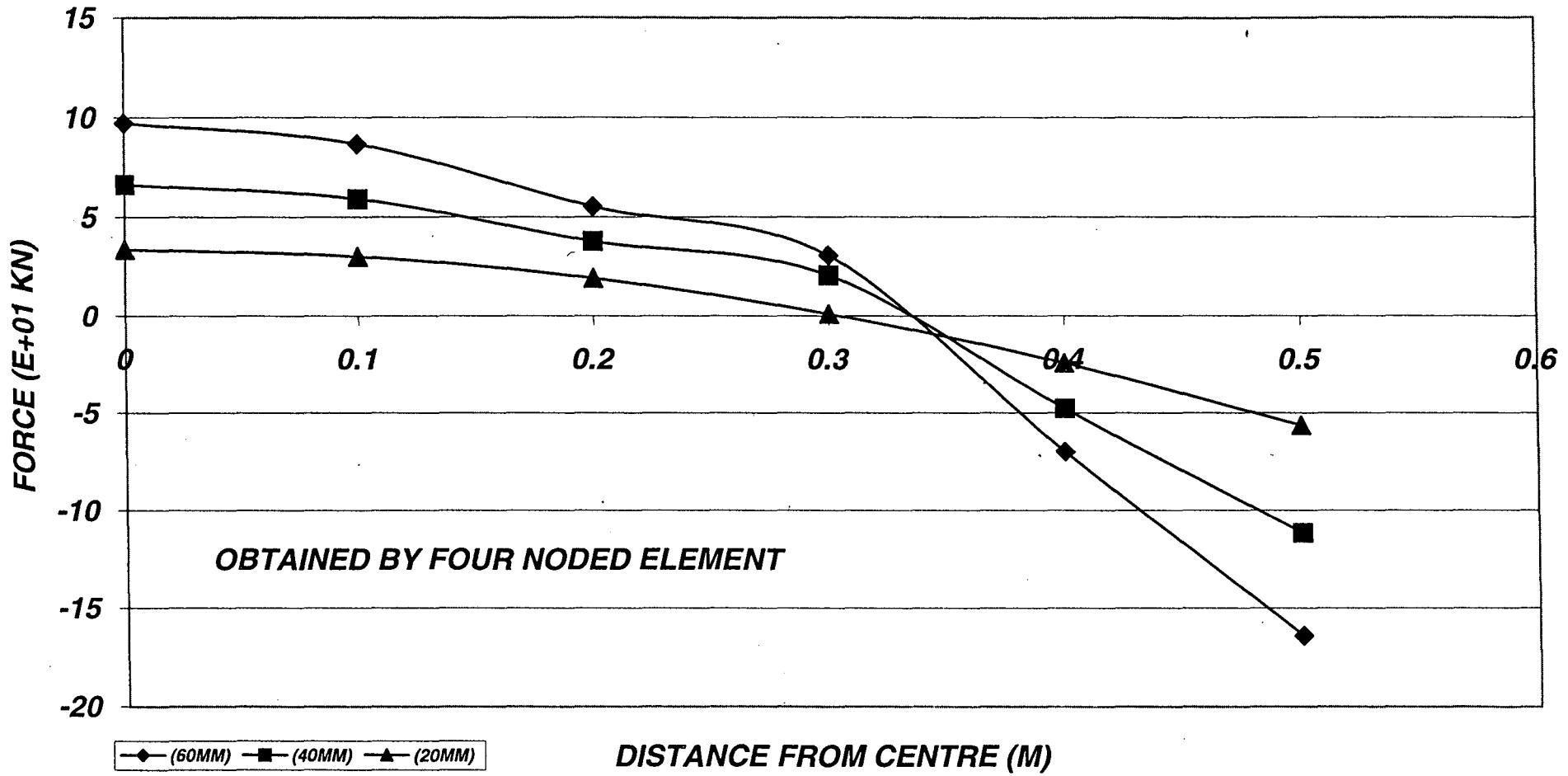
LONGITUDINAL DISPLACEMENT AT END (6M SPAN)**GRAPH 6.4**



LONGITUDINAL DISPLACEMENT AT END (4M SPAN)

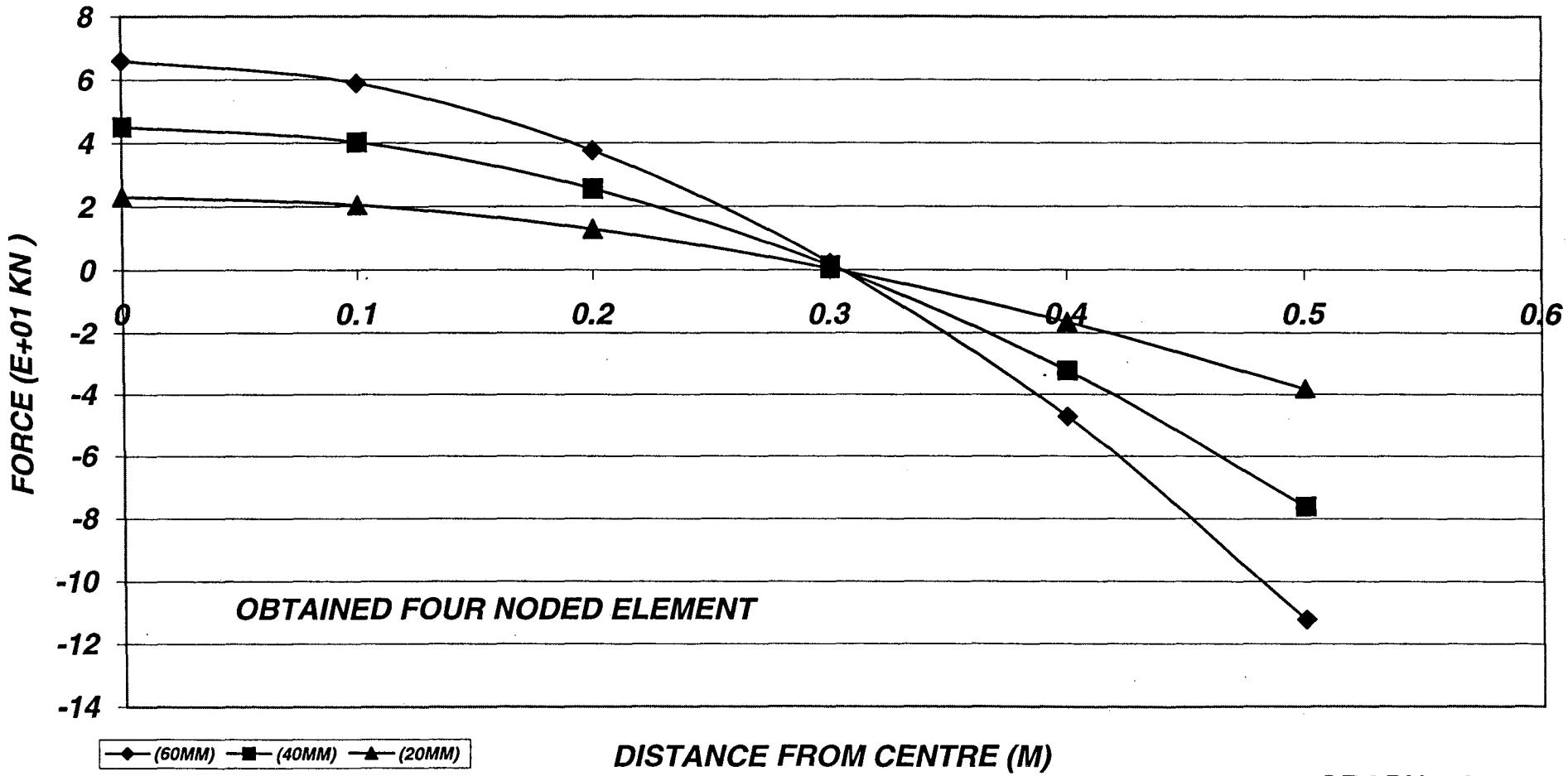


INPLANE LONGITUDINAL FORCE AT CENTRE (6M SPAN)

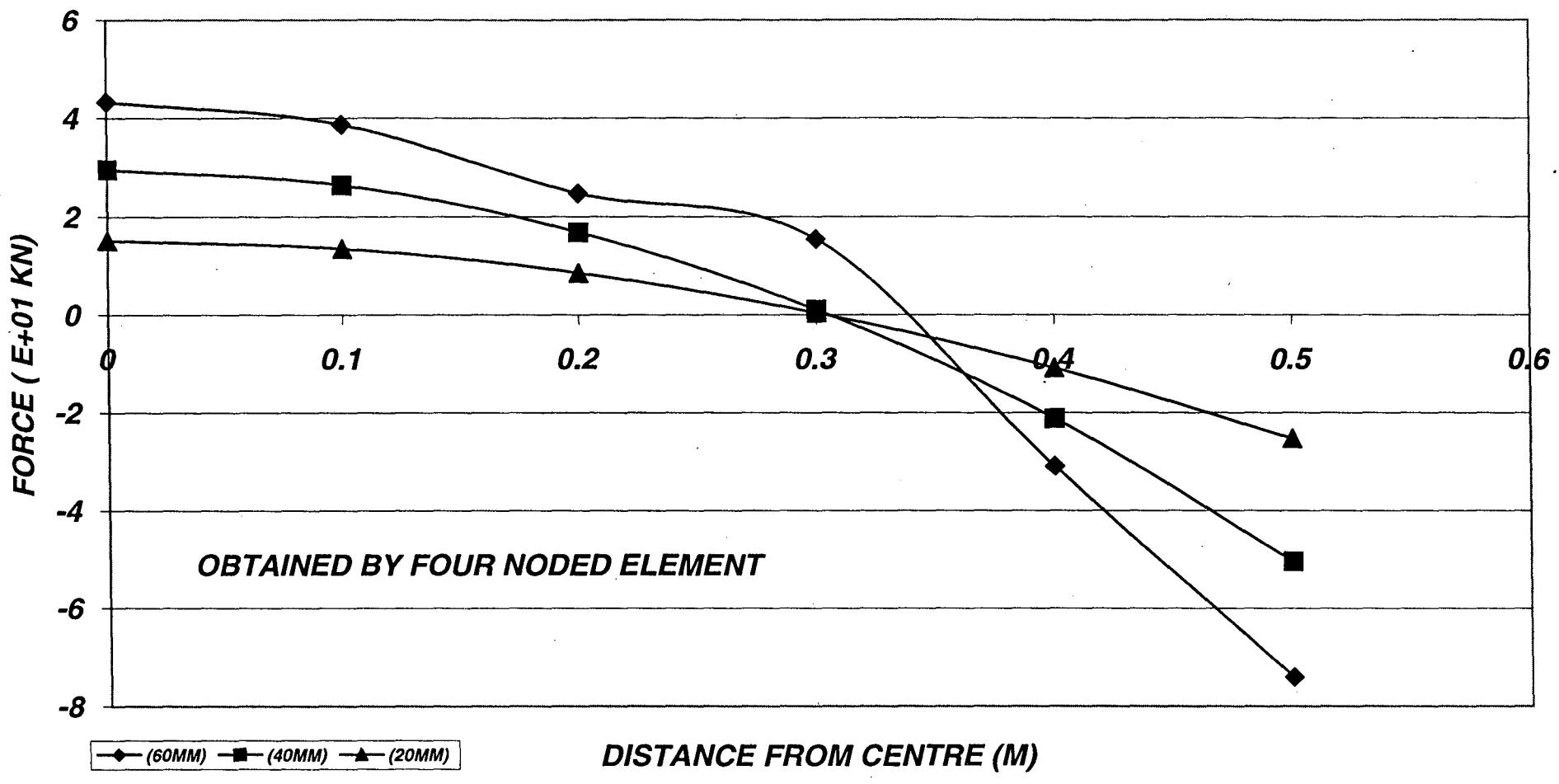


GRAPH 6.7

INPLANE LONGITUDINAL FORCE AT CENTRE (5M SPAN)

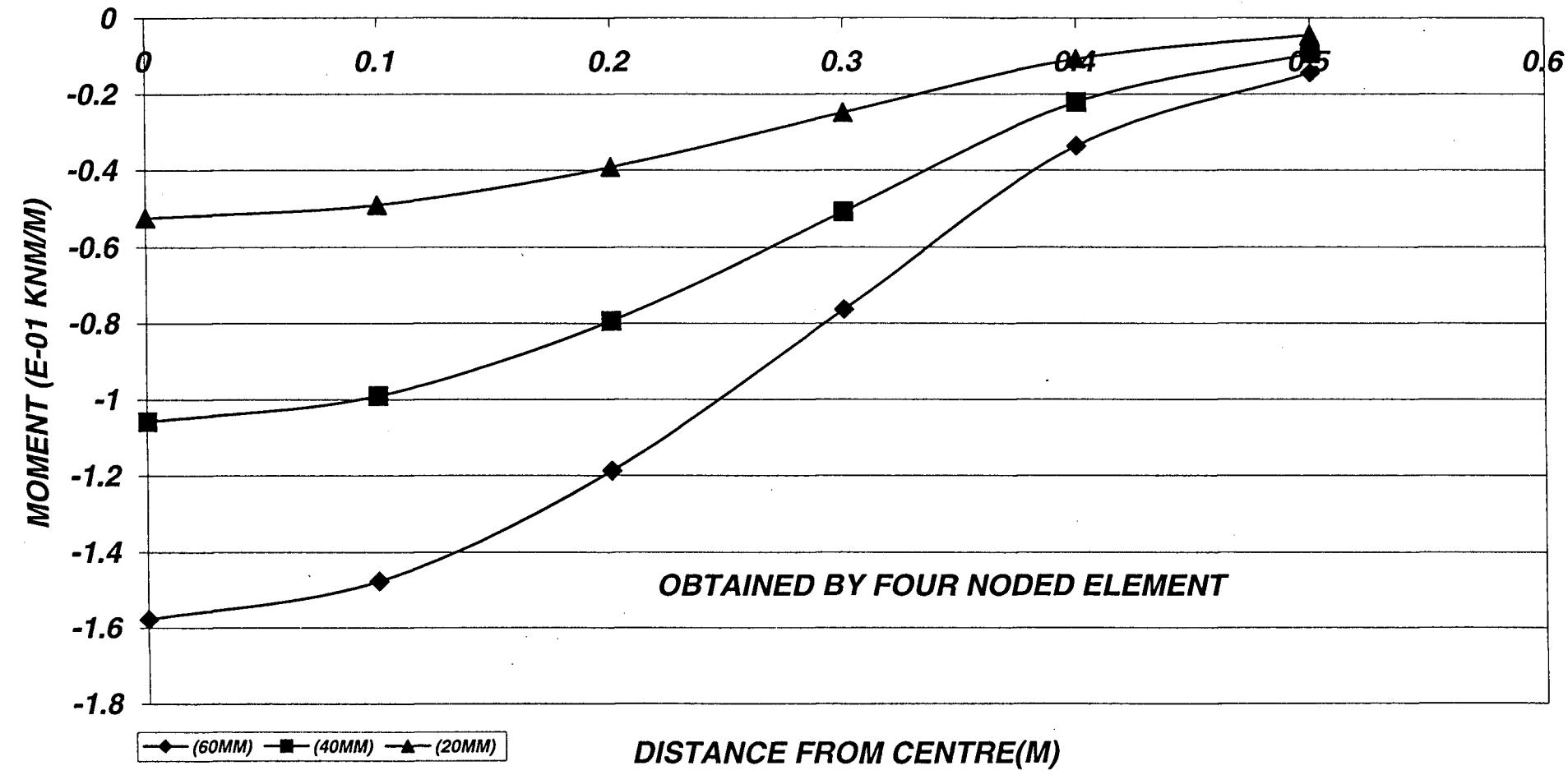


INPLANE LONGITUDINAL FORCE AT CENTRE (4M SPAN)

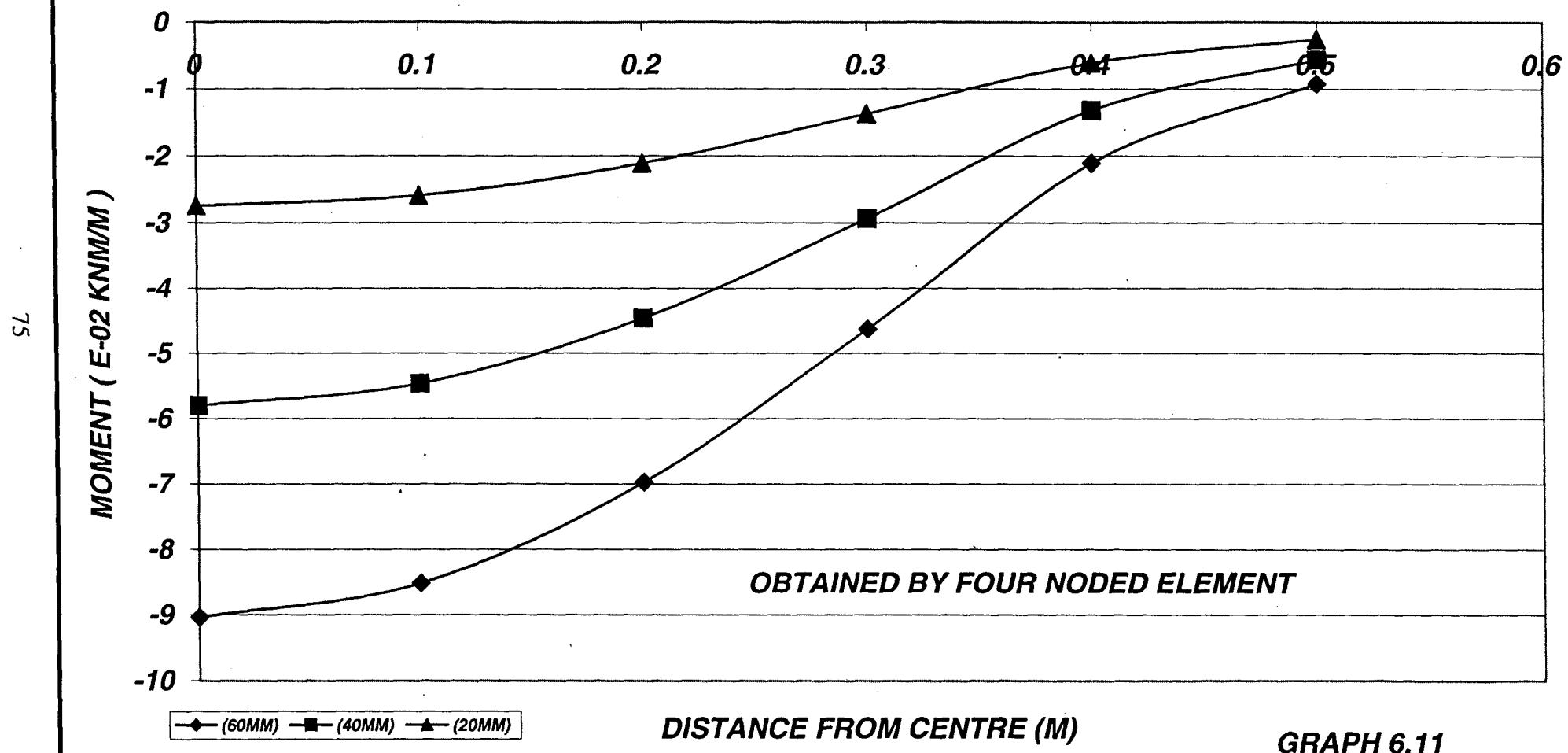


GRAPH 6.9

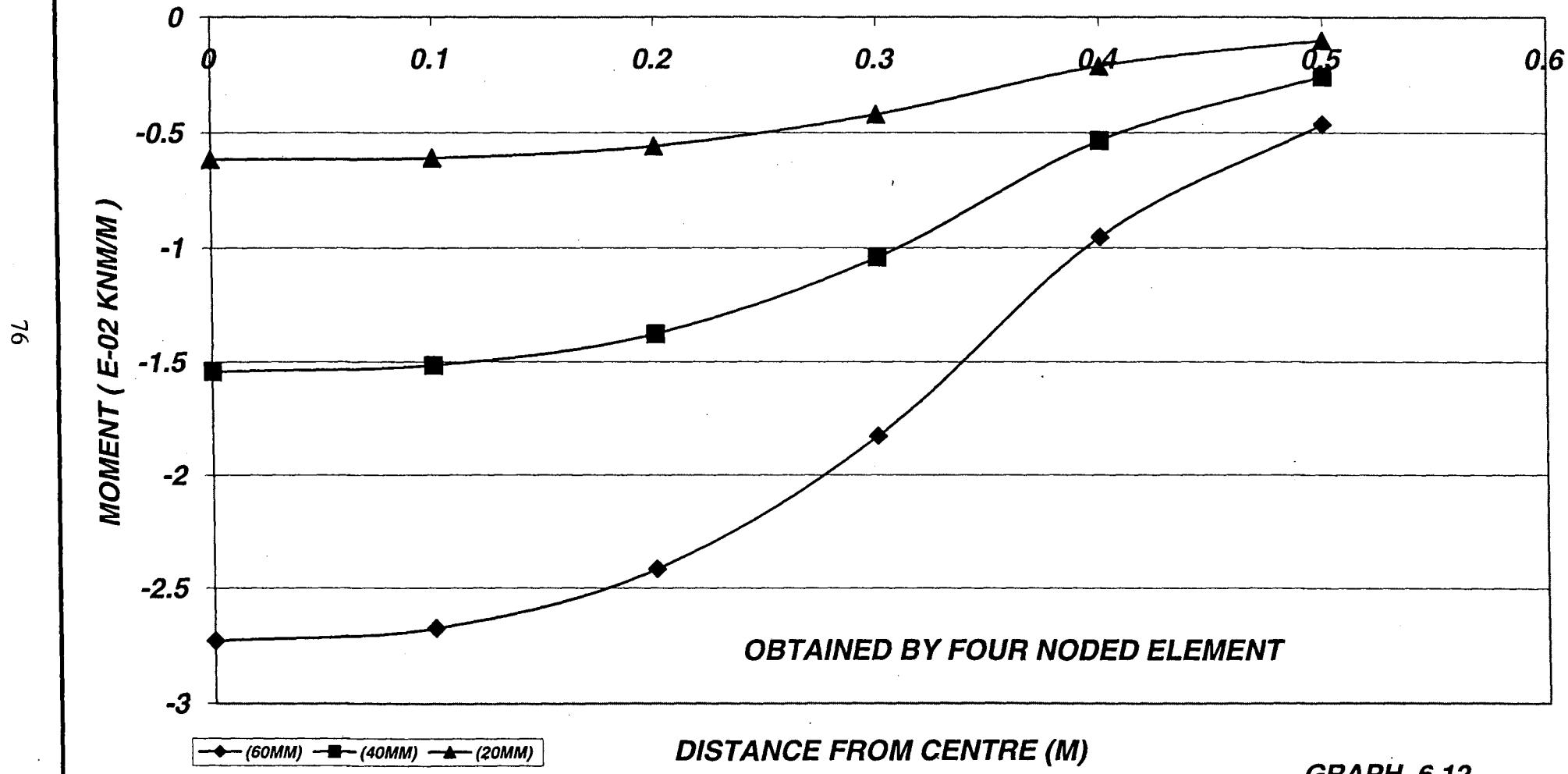
TRANSVERSE MOMENT AT CENTRE (6M SPAN)



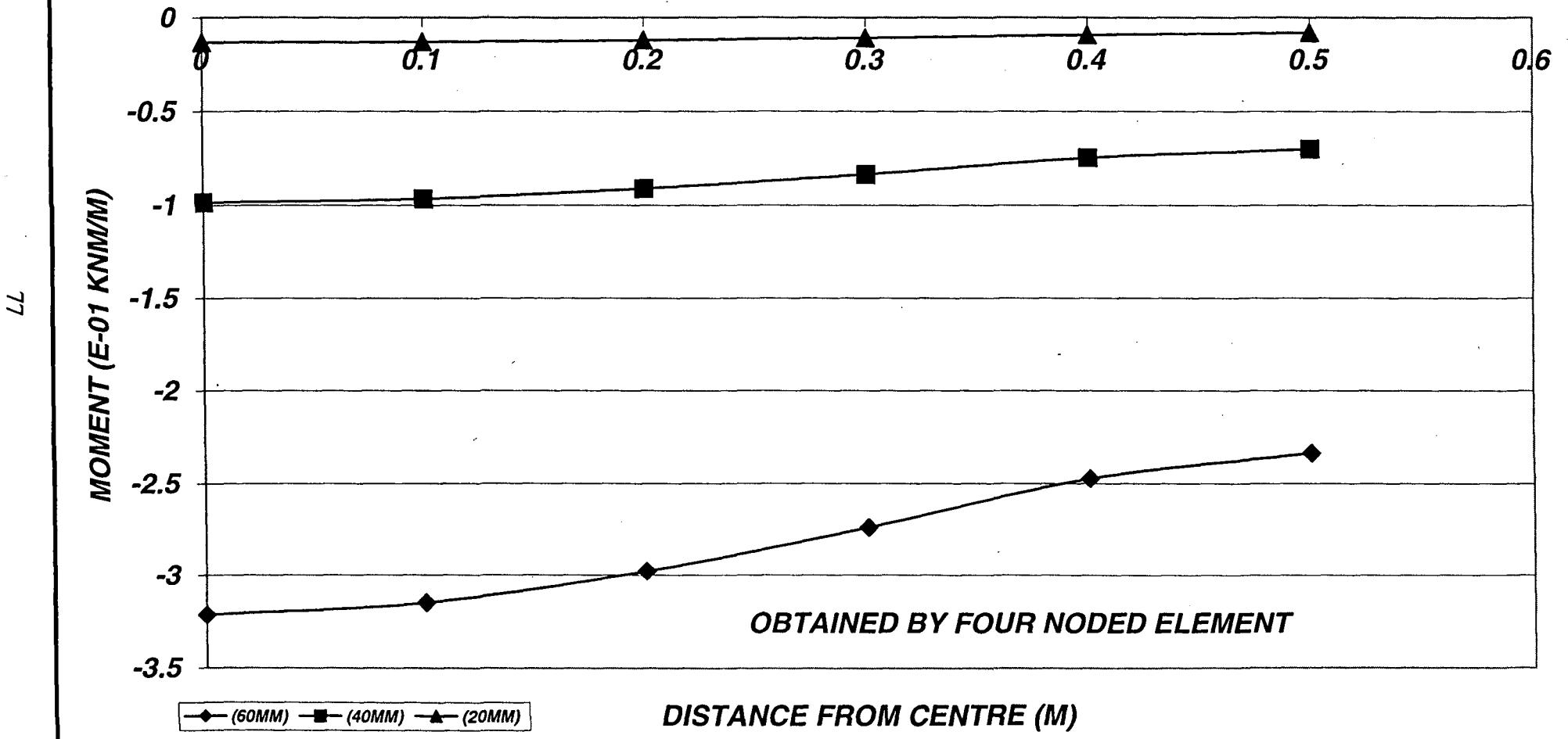
TRANSVERSE MOMENT AT CENTRE (5M SPAN)



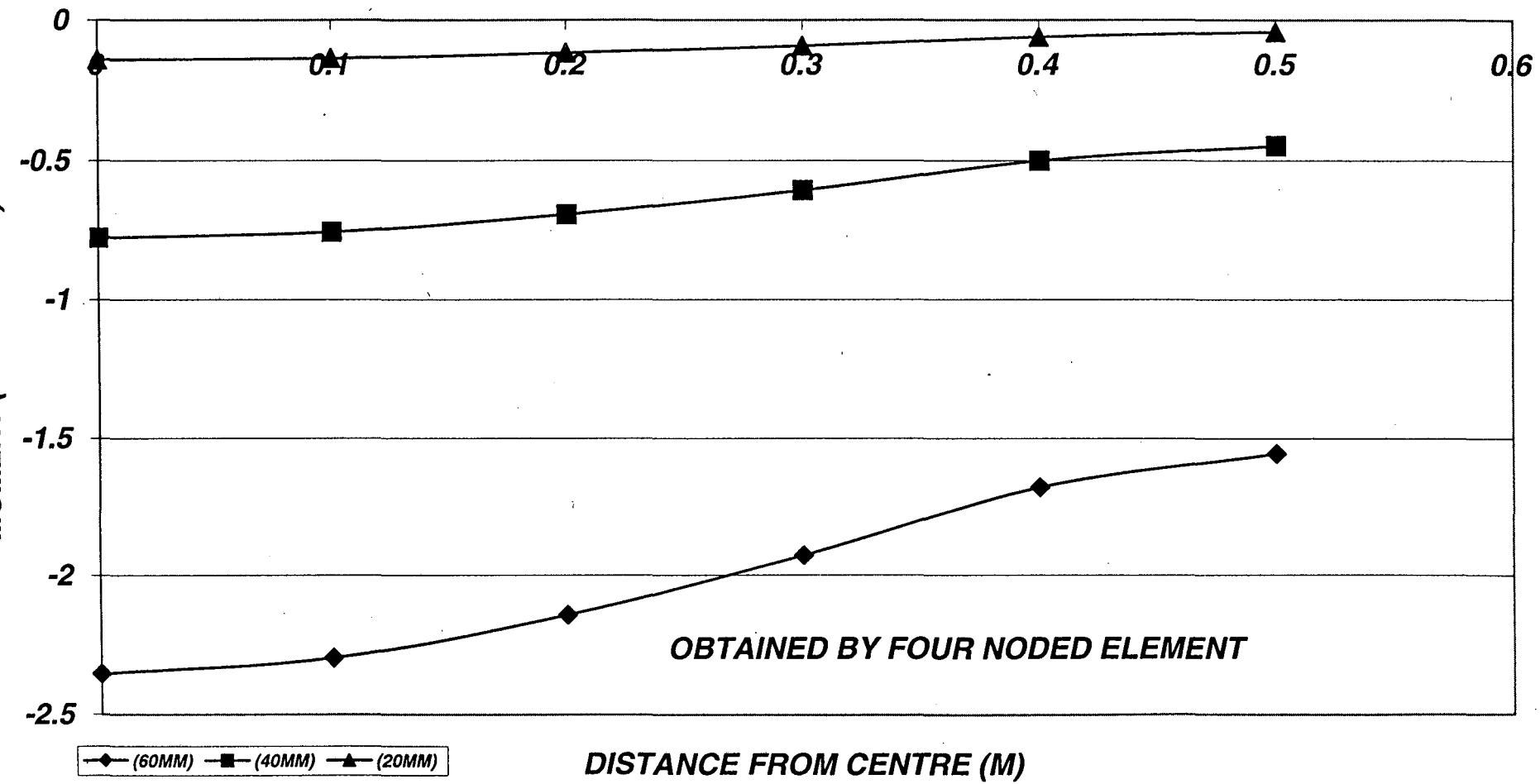
TRANSVERSE MOMENT AT CENTRE (4M SPAN)



LONGITUDINAL MOMENT AT CENTRE (6M SPAN)

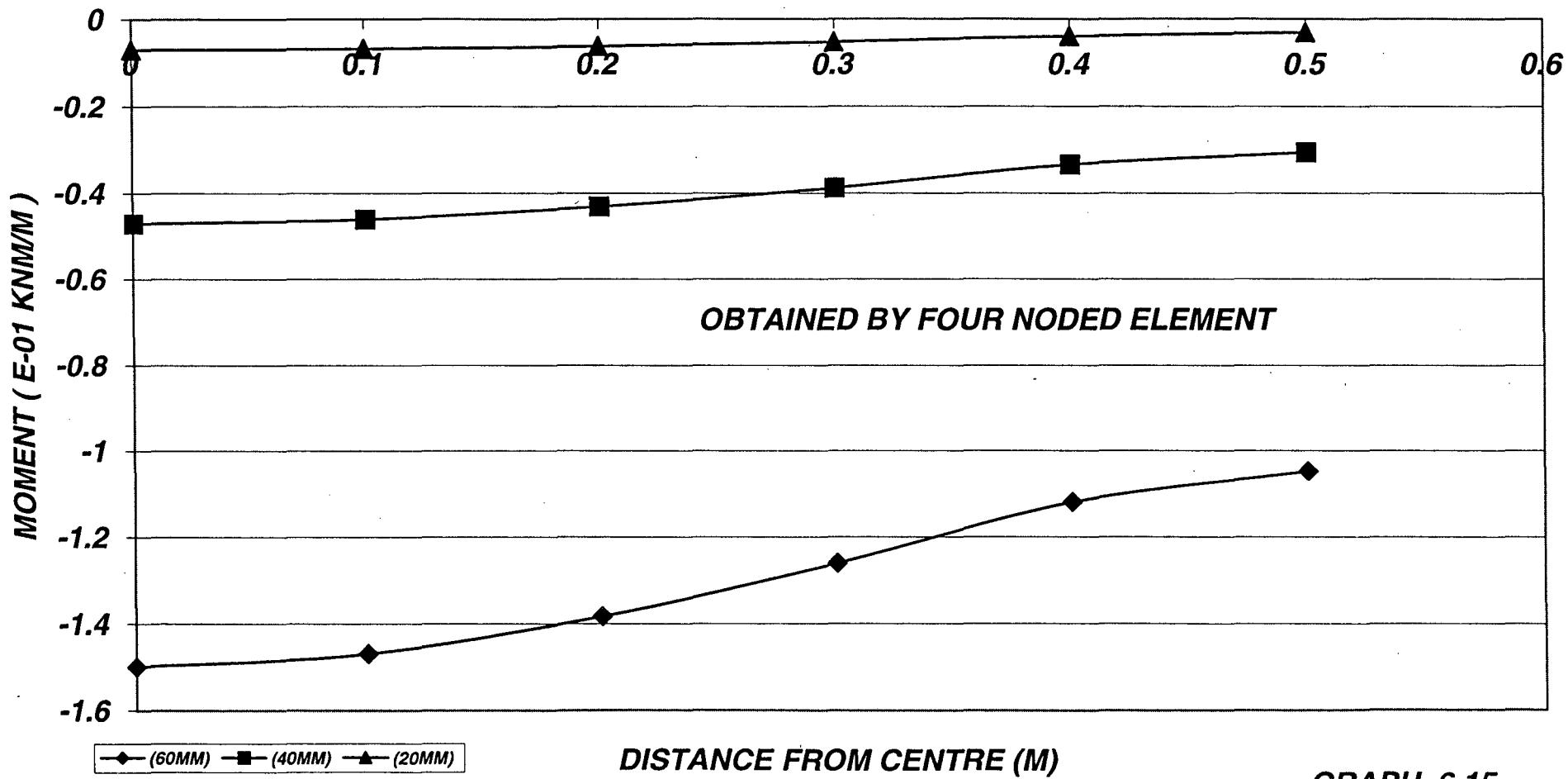


LONGITUDINAL MOMENT AT CENTRE (5M SPAN)



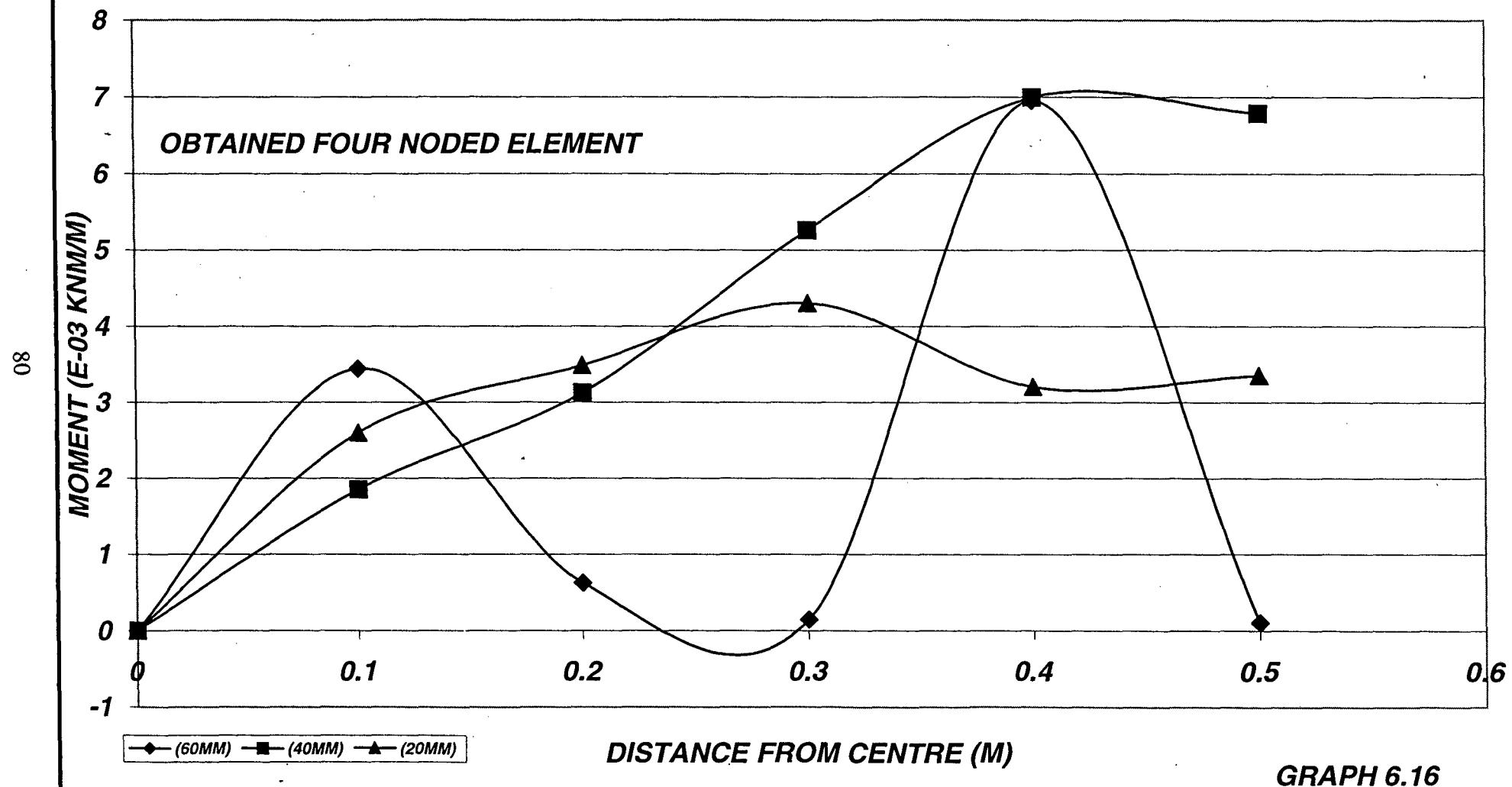
GRAPH 6.14

LONGITUDINAL MOMENT AT CENTRE (4M SPAN)

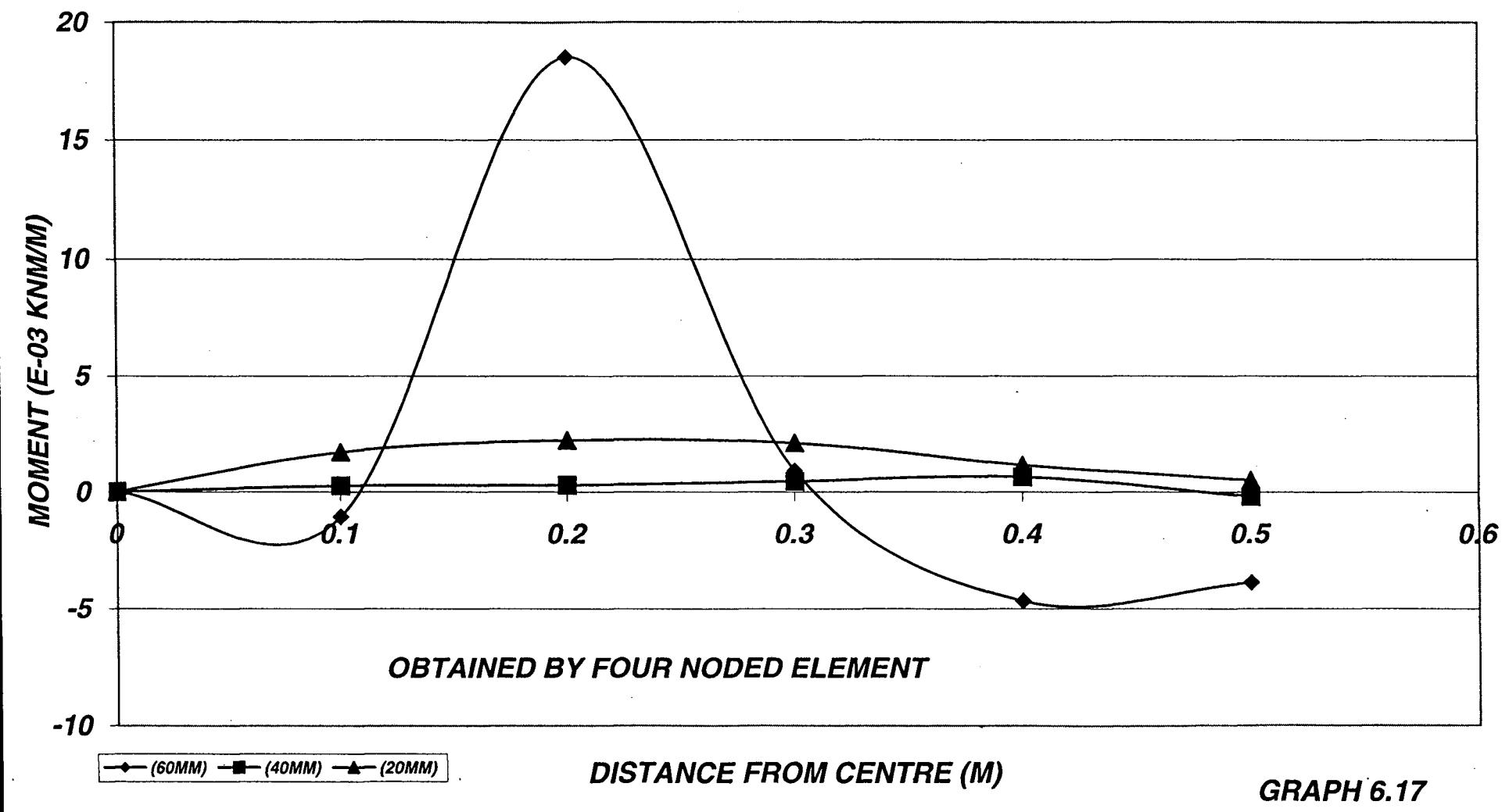


GRAPH 6.15

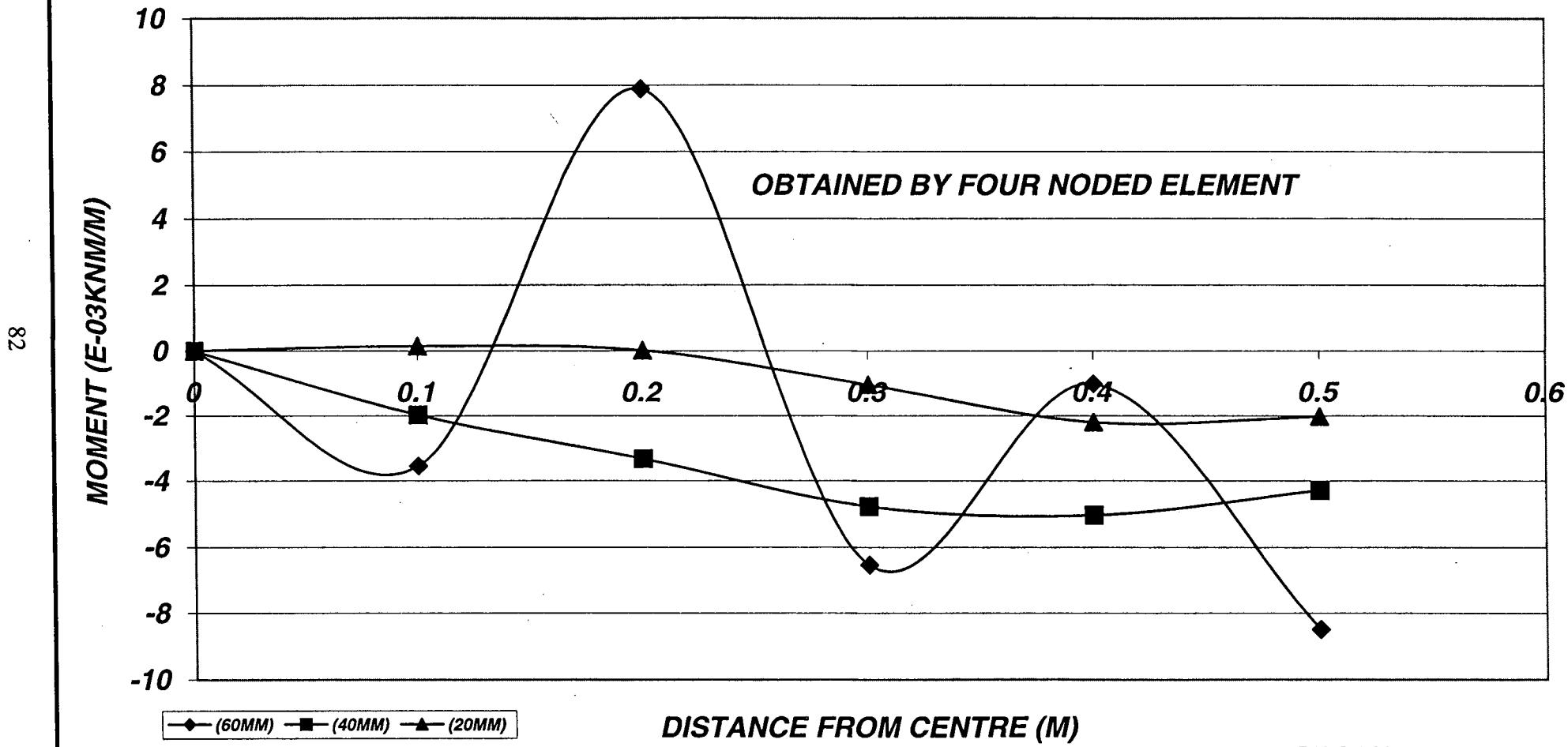
TWISTING MOMENT AT END (6M SPAN)



TWISTING MOMENT AT END (5M SPAN)

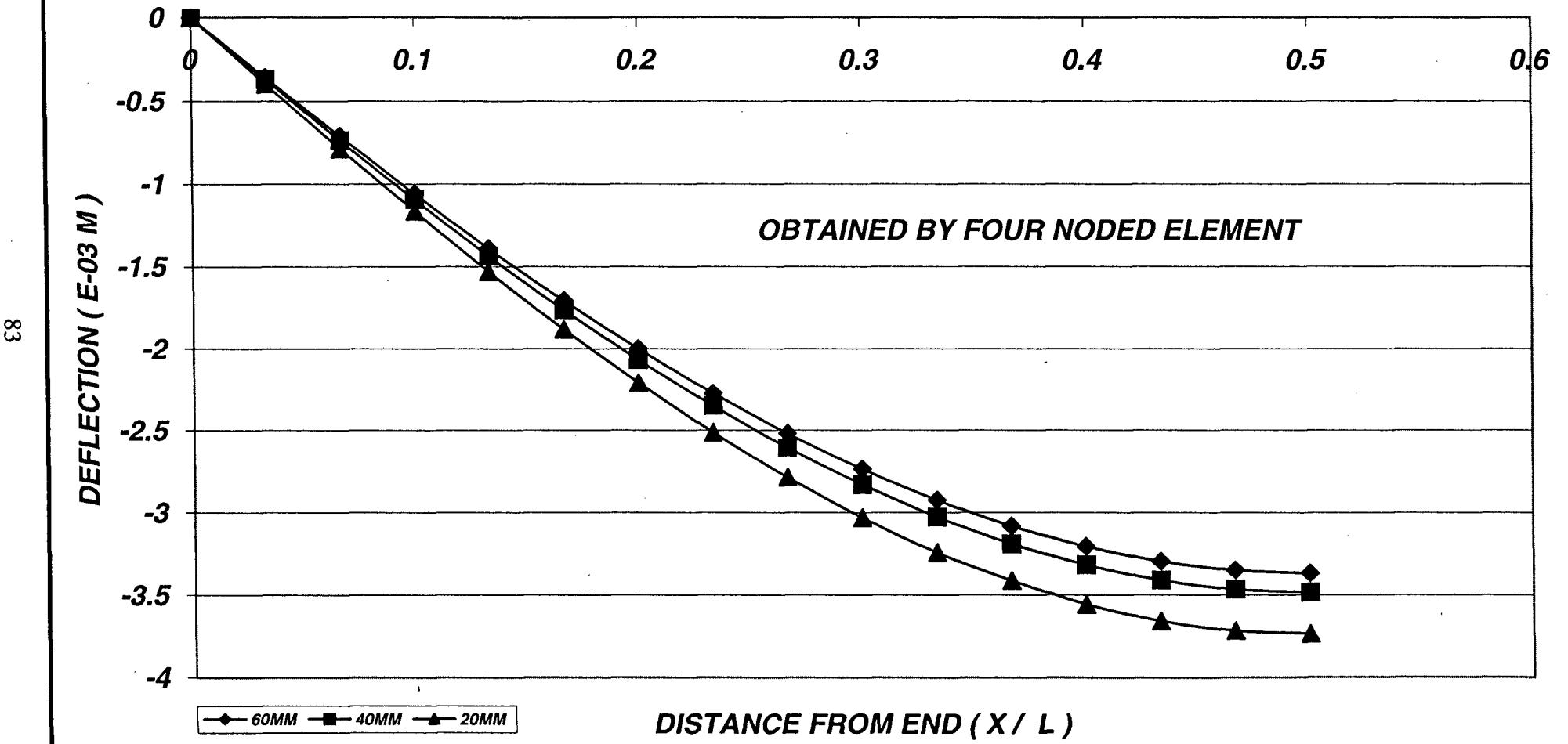


TWISTING MOMENT AT END (4M SPAN)



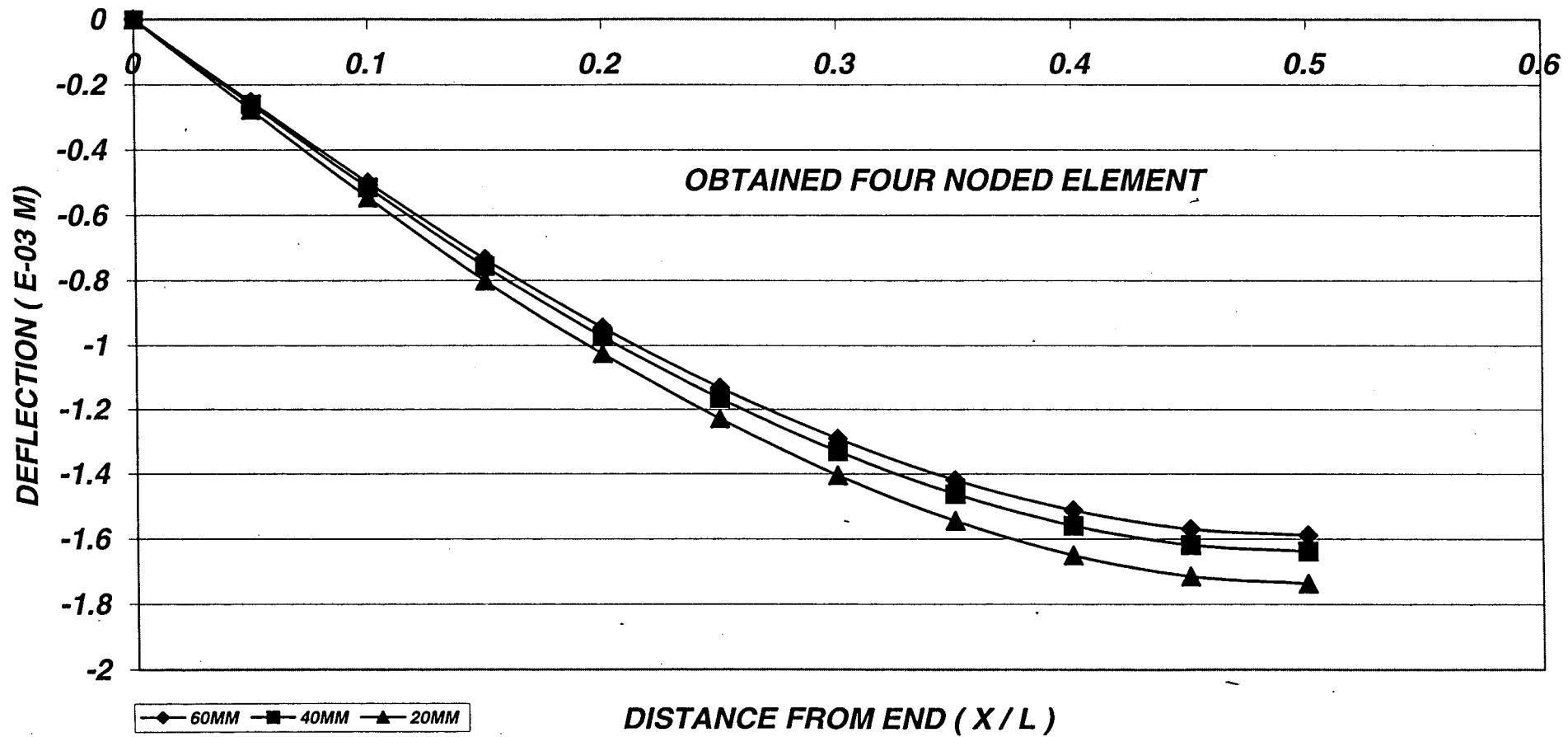
GRAPH 6.18

CENTRAL VERTICAL DEFLECTION ALONG THE LENGTH (6M SPAN)



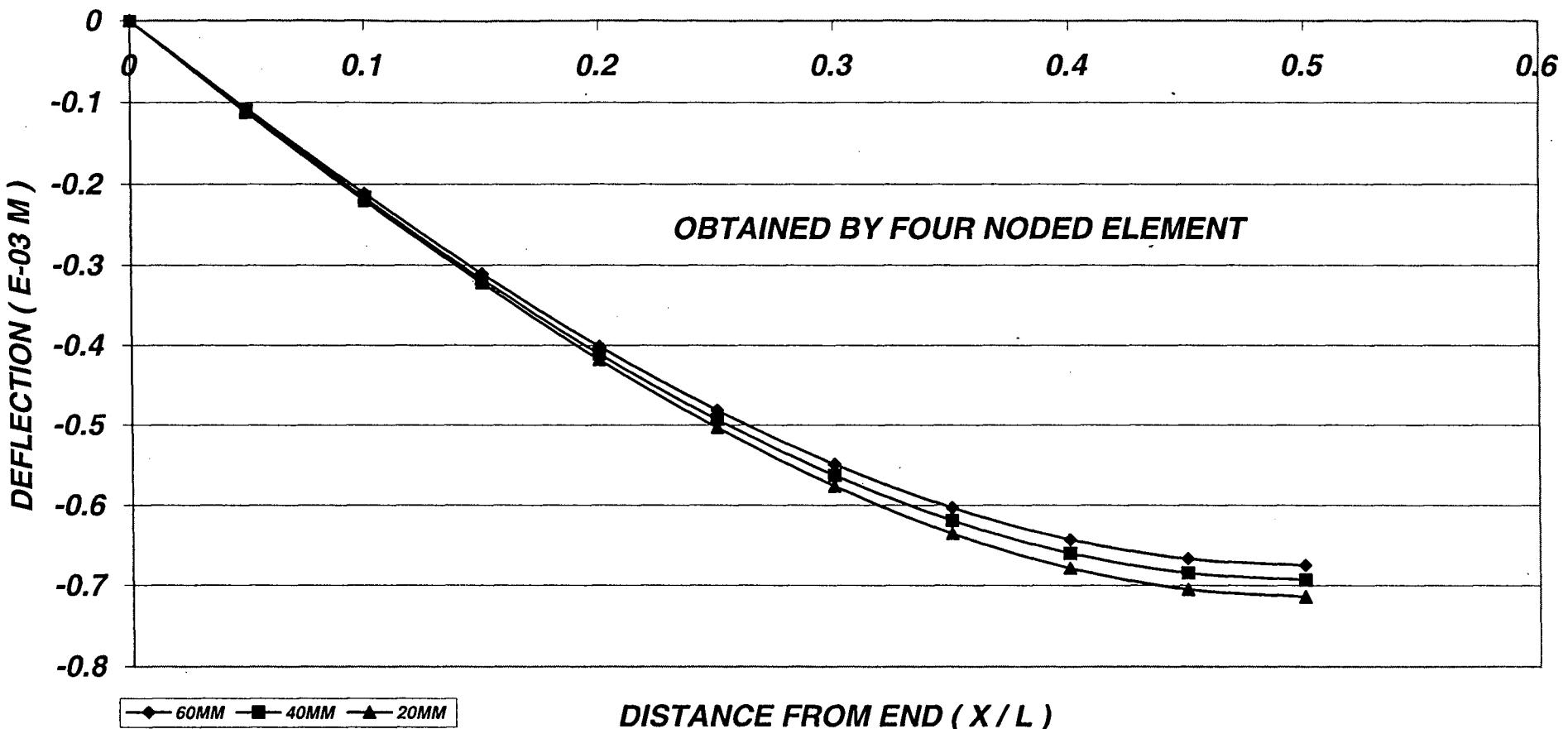
GRAPH 6.19

CENTRAL VERTICAL DEFLECTION ALONG THE LENGTH
(5M SPAN)



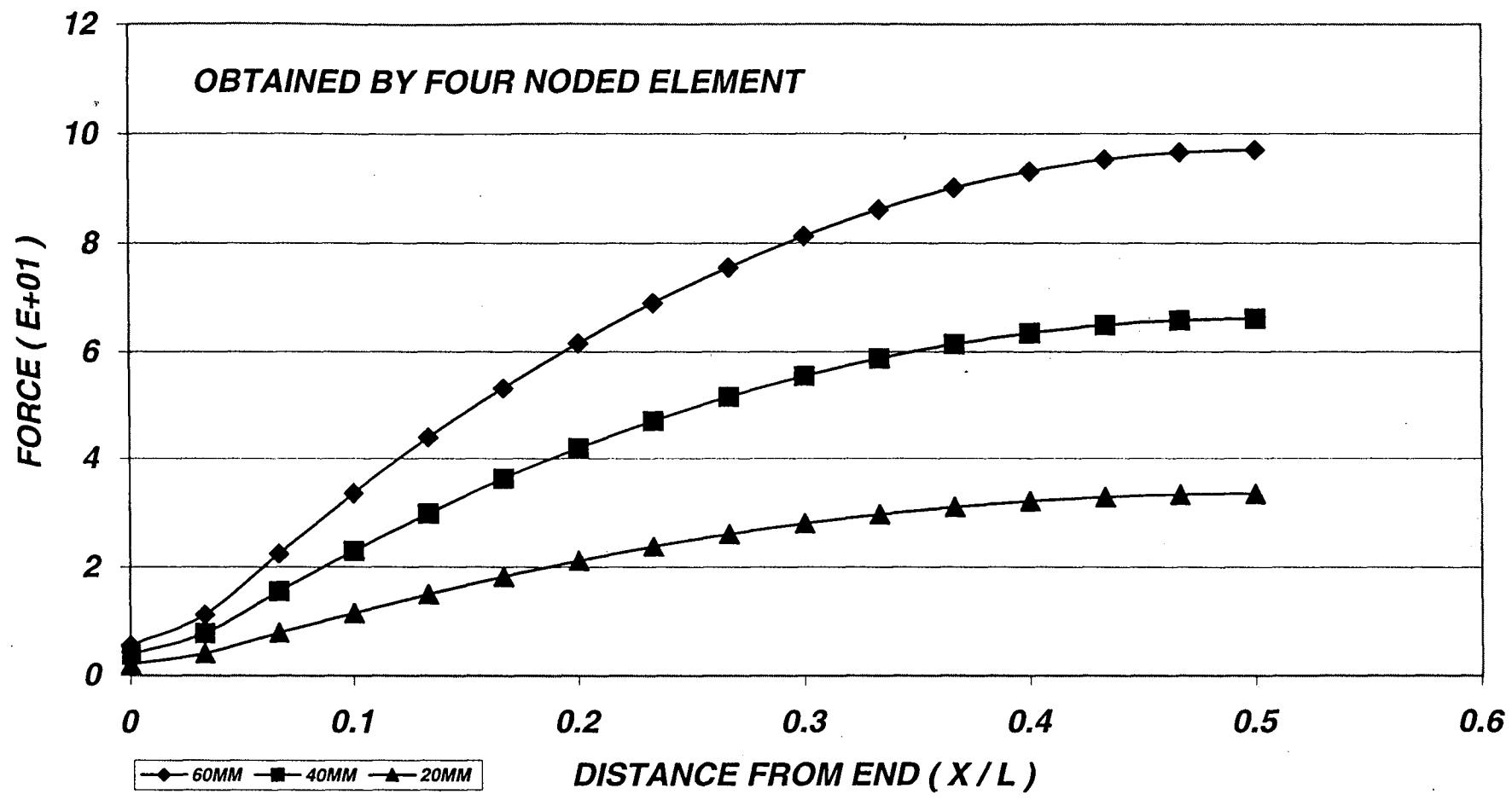
GRAPH 6.20

**CENTRAL VERTICAL DEFLECTION ALONG
THE LENGTH (4M SPAN)**



GRAPH 6.21

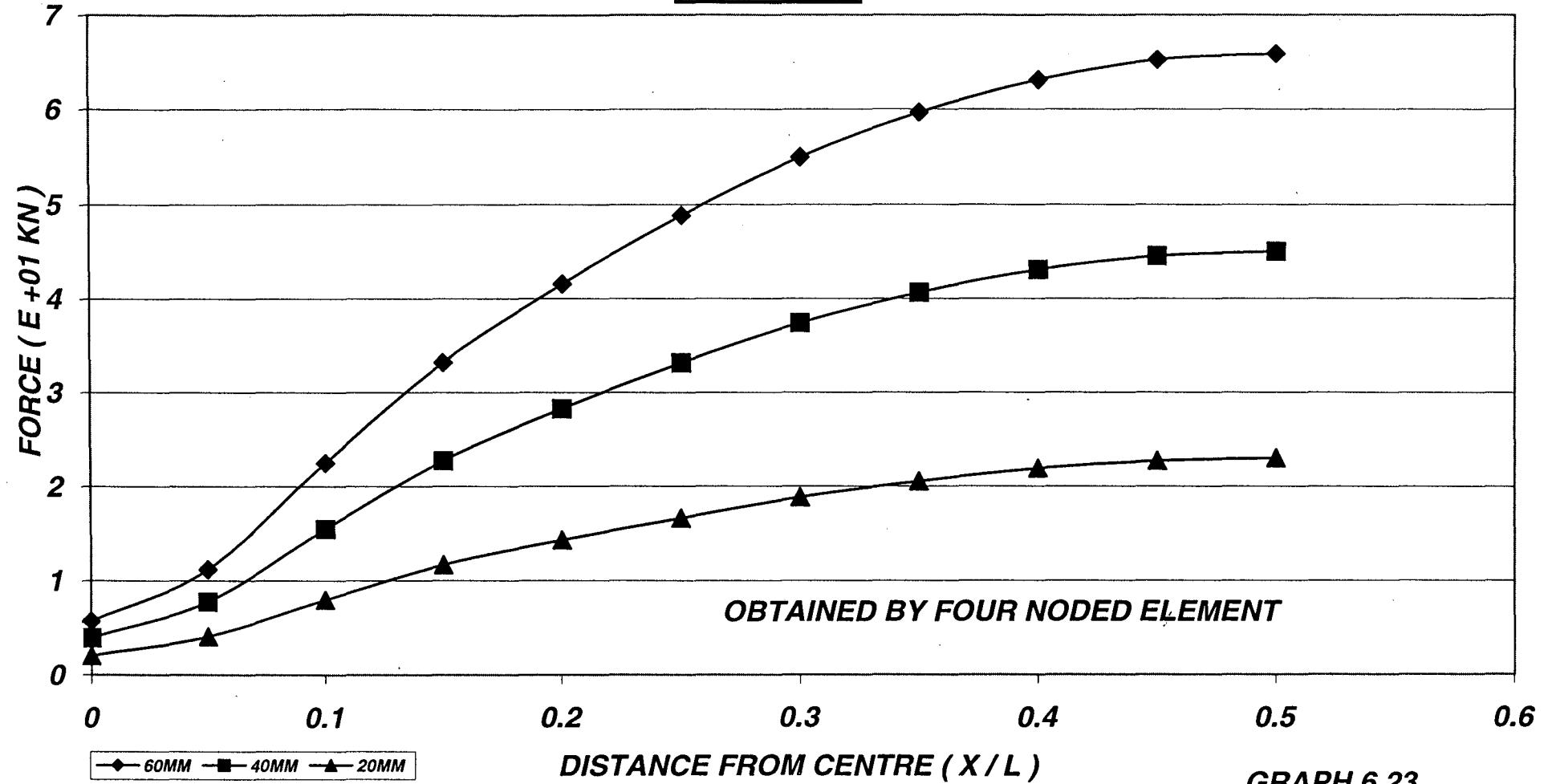
CENTRAL INPLANE LONGITUDINAL FORCE ALONG THE LENGTH (
6M SPAN)



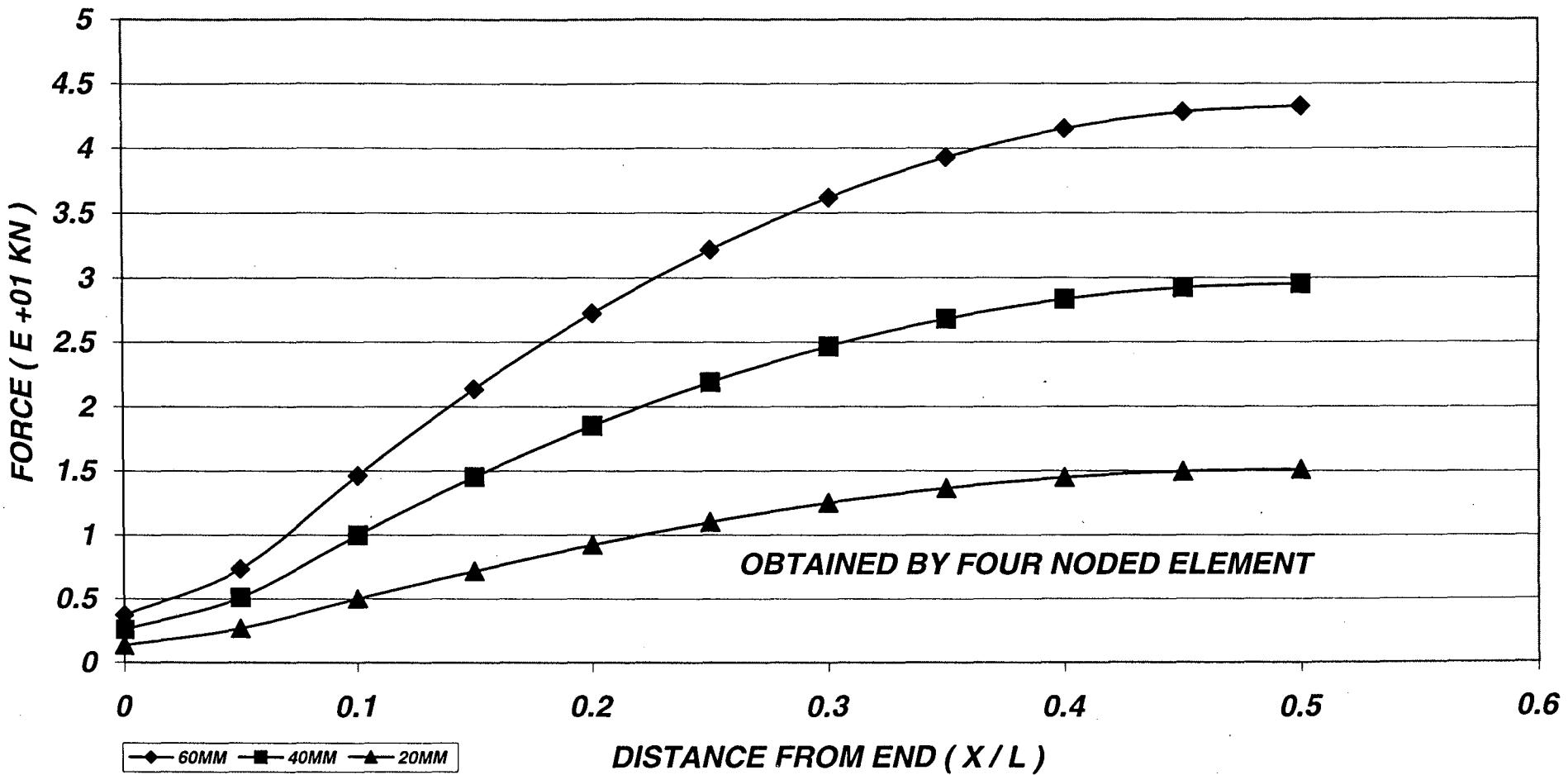
GRAPH 6.22

CENTRAL INPLANE LONGITUDINAL FORCE
(5M SPAN)

L8

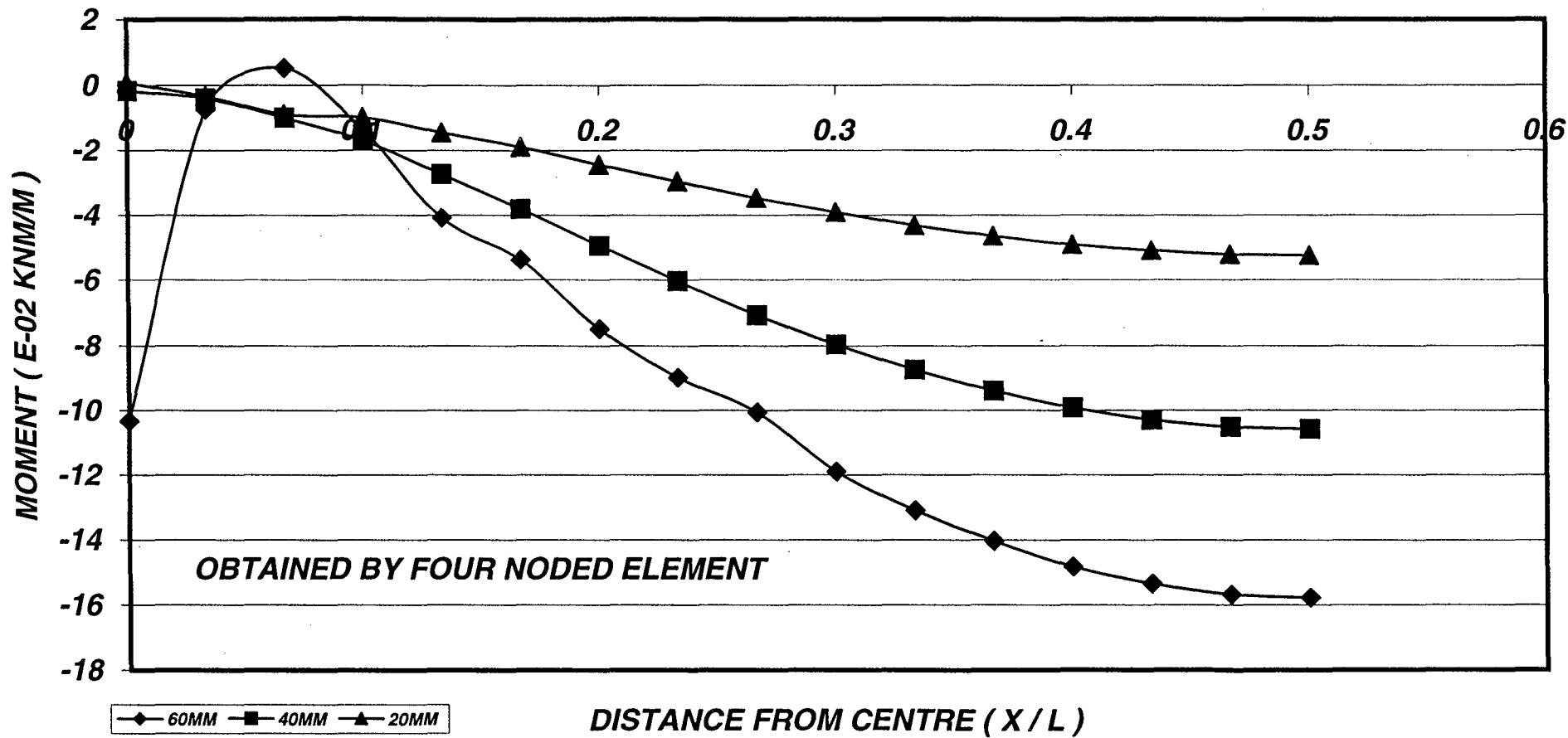


CENTRAL INPLANE LONGITUDINAL FORCE
ALONG THE LENGTH (4M SPAN)

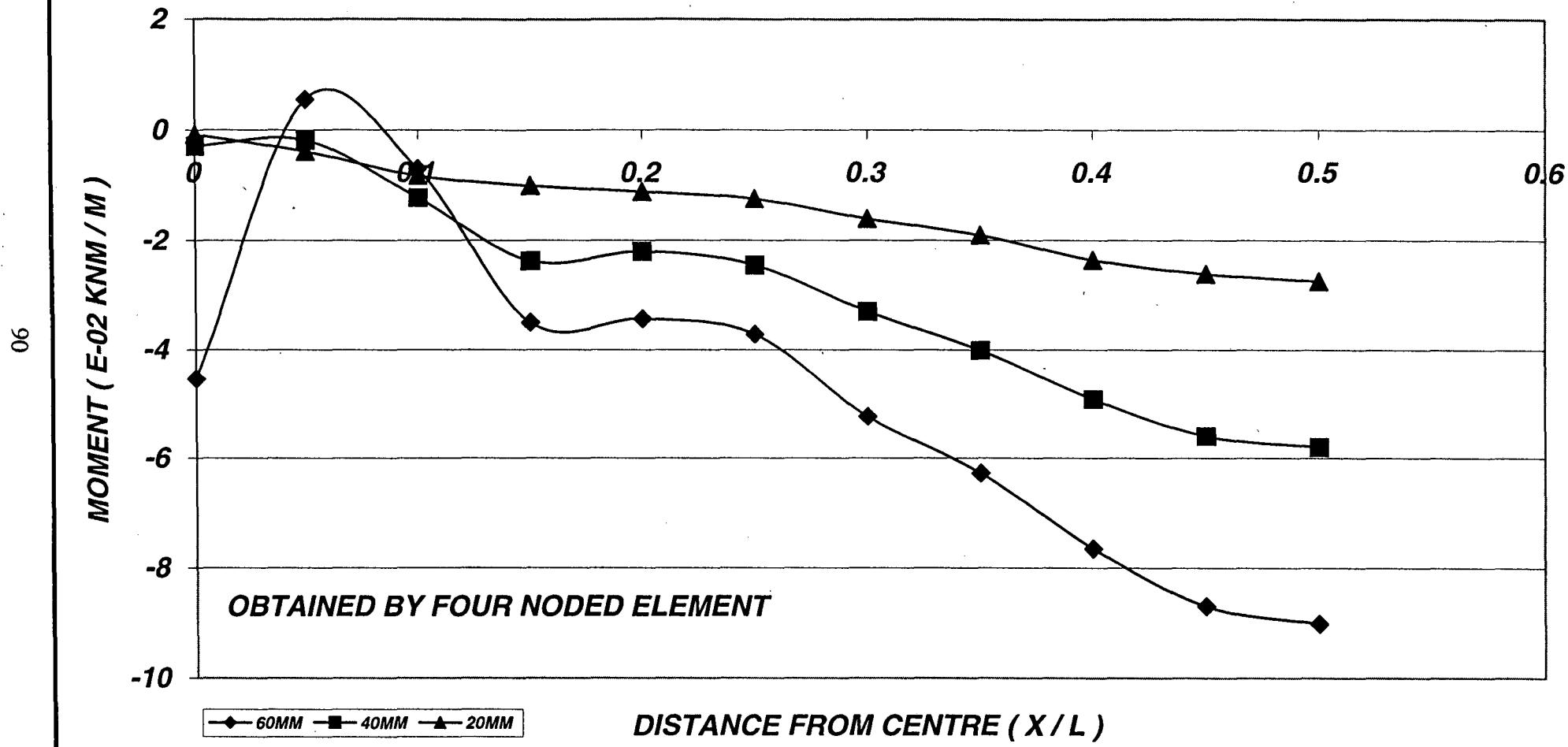


GRAPH 6.24

CENTRAL TRANSVERSE MOMENT ALONG THE LENGTH (6M SPAN)



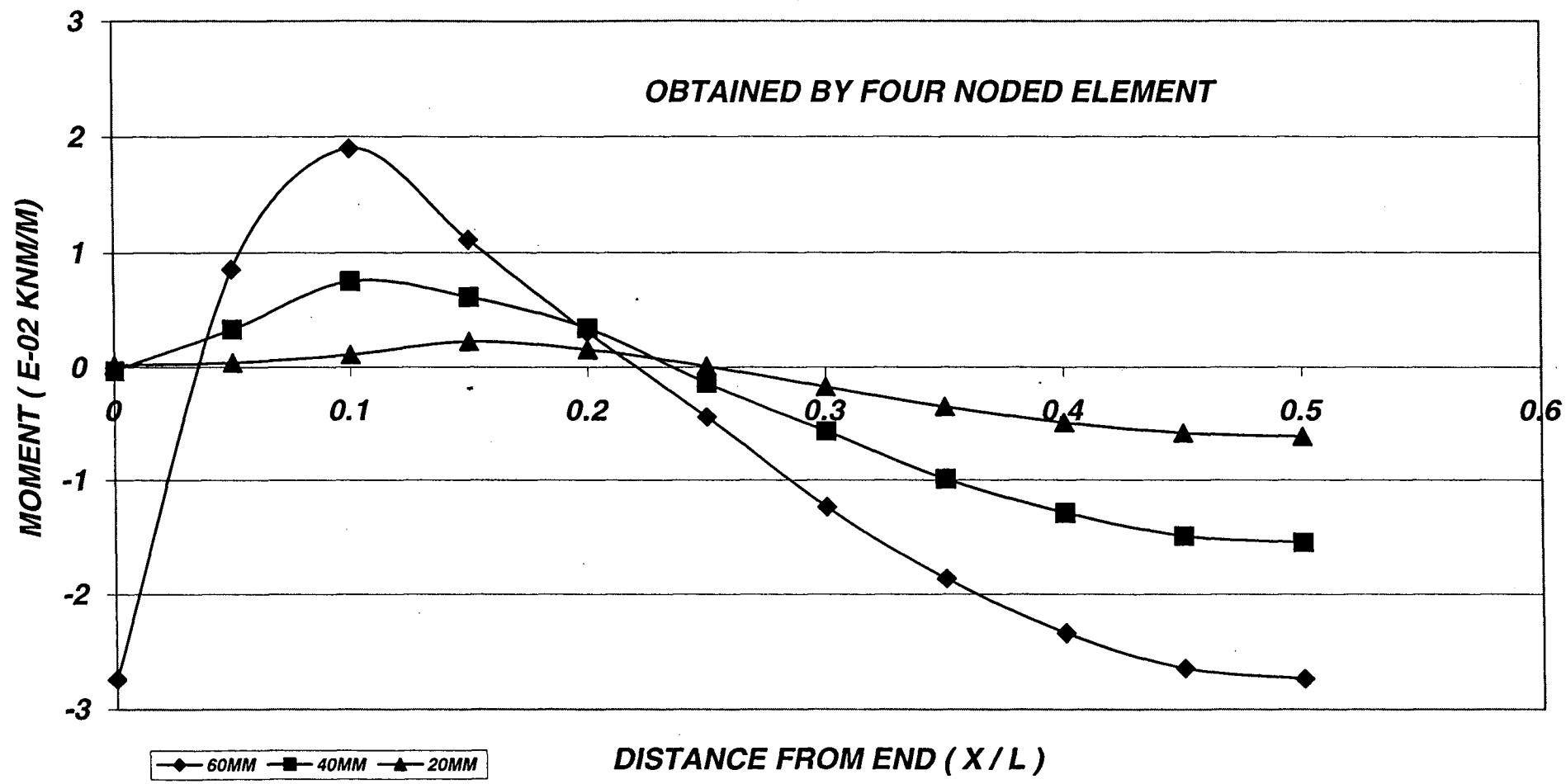
CENTRAL TRANSVERSE MOMENT ALONG THE LENGTH (5M SPAN)



GRAPH 6.26

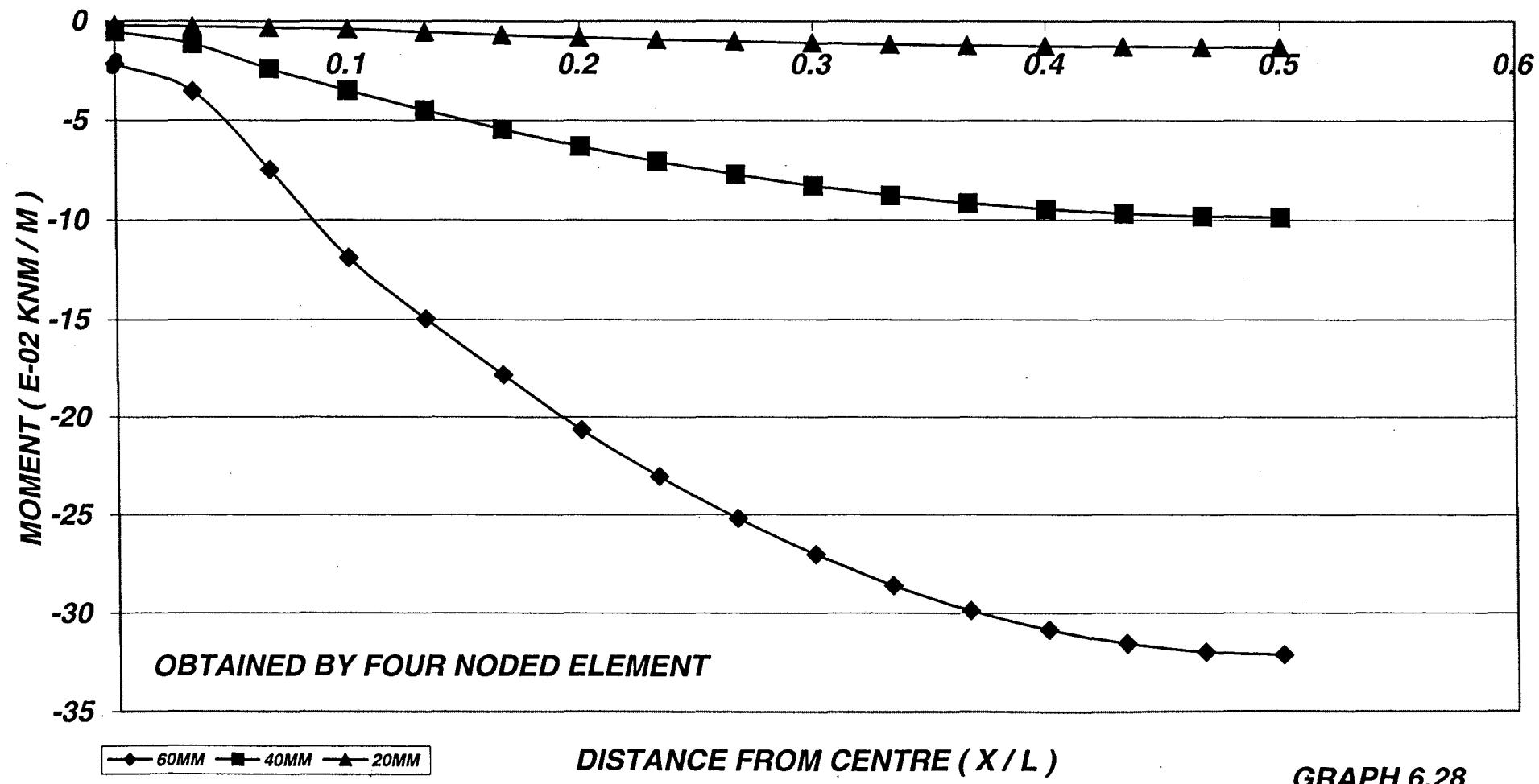
**CENTRAL TRANSVERSE MOMENT ALONG
THE LENGTH (4M SPAN)**

16

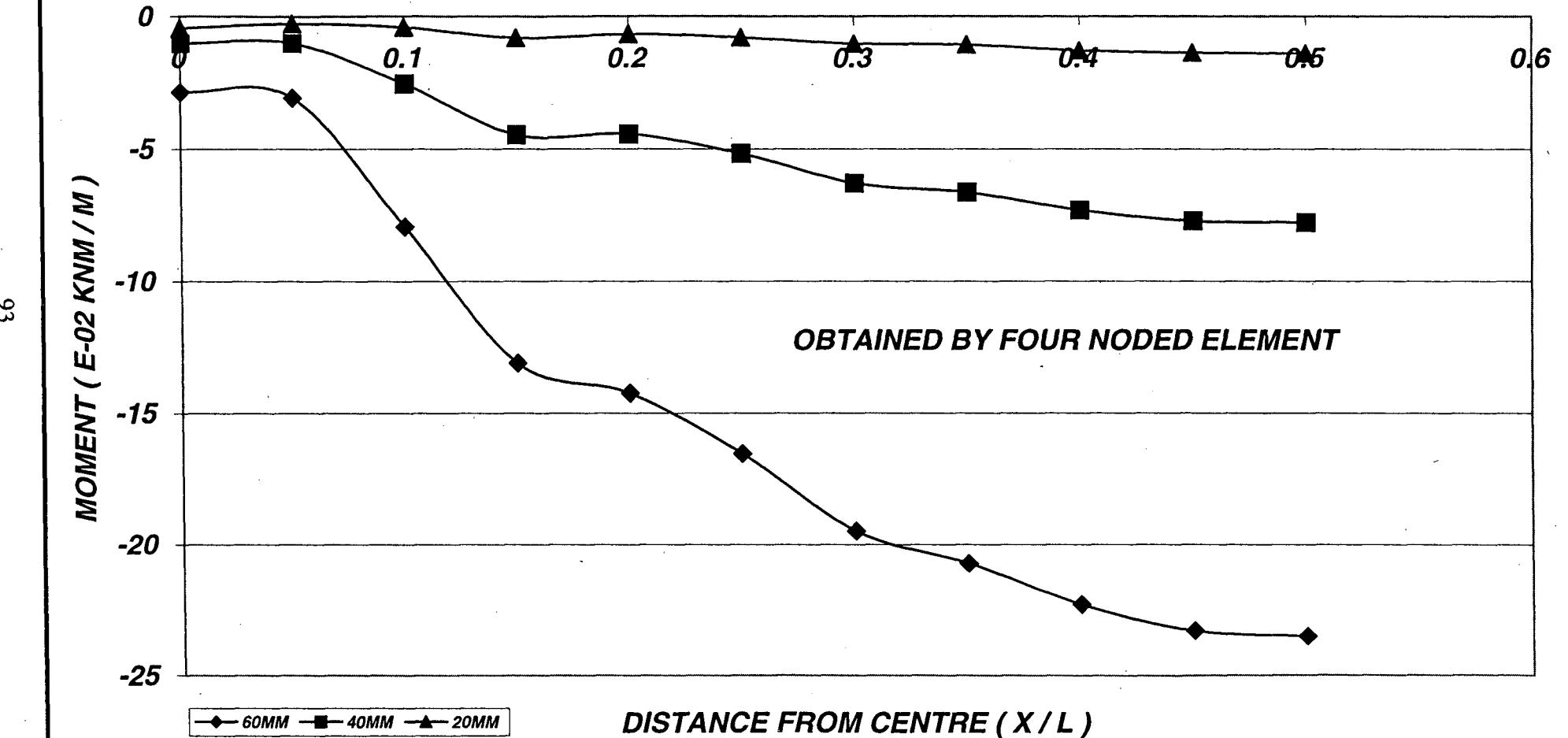


GRAPH 6.27

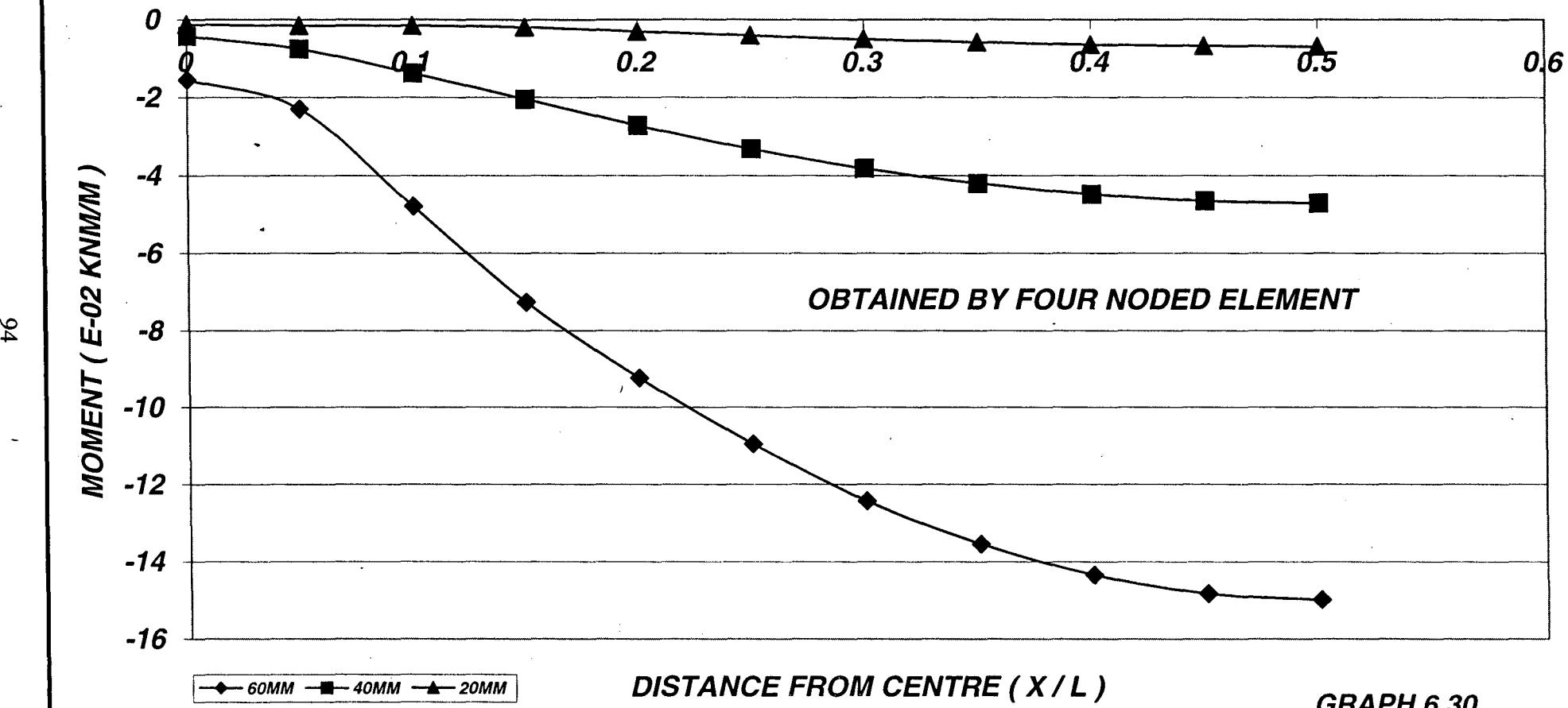
CENTRAL LONGITUDINAL MOMENT ALONG THE LENGTH (6M SPAN)



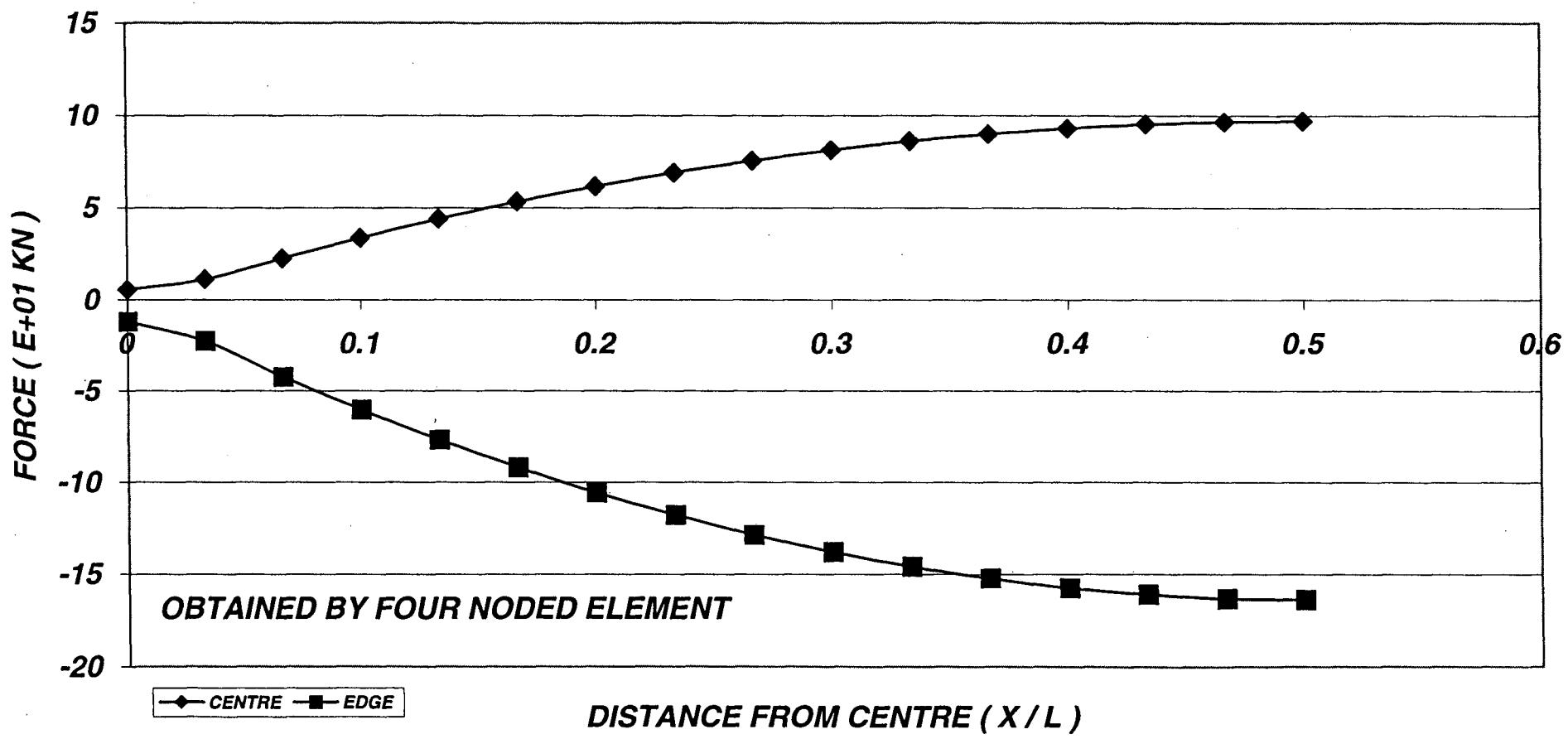
CENTRAL LONGITUDINAL MOMENT ALONG THE LENGTH (5M SPAN)



**CENTRAL LONGITUDINAL MOMENT ALONG
THE LENGTH (4M SPAN)**

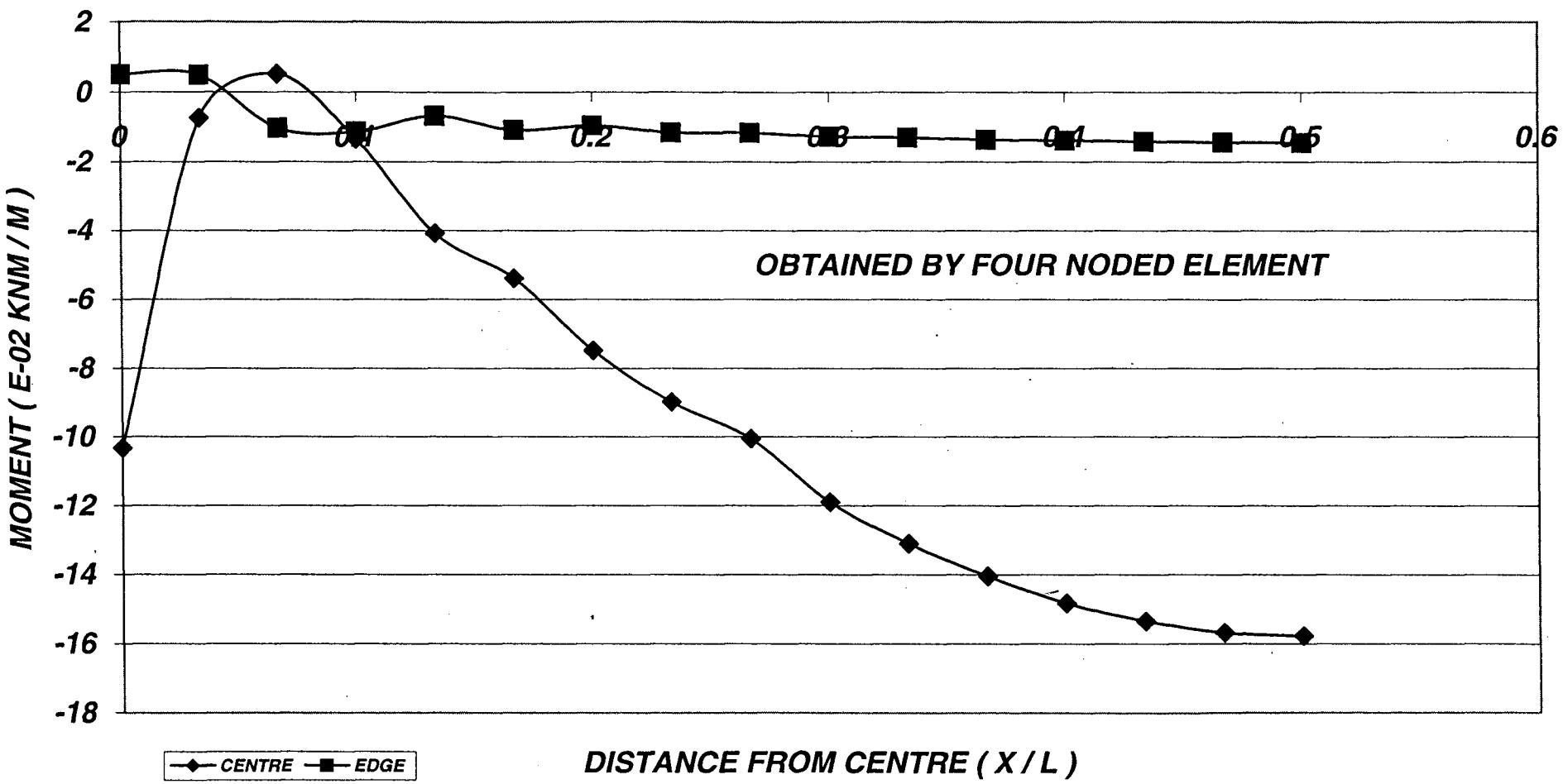


INPLANE LONGITUDINAL FORCE ALONG THE LENGTH
(6M SPAN ,60MM TH.)



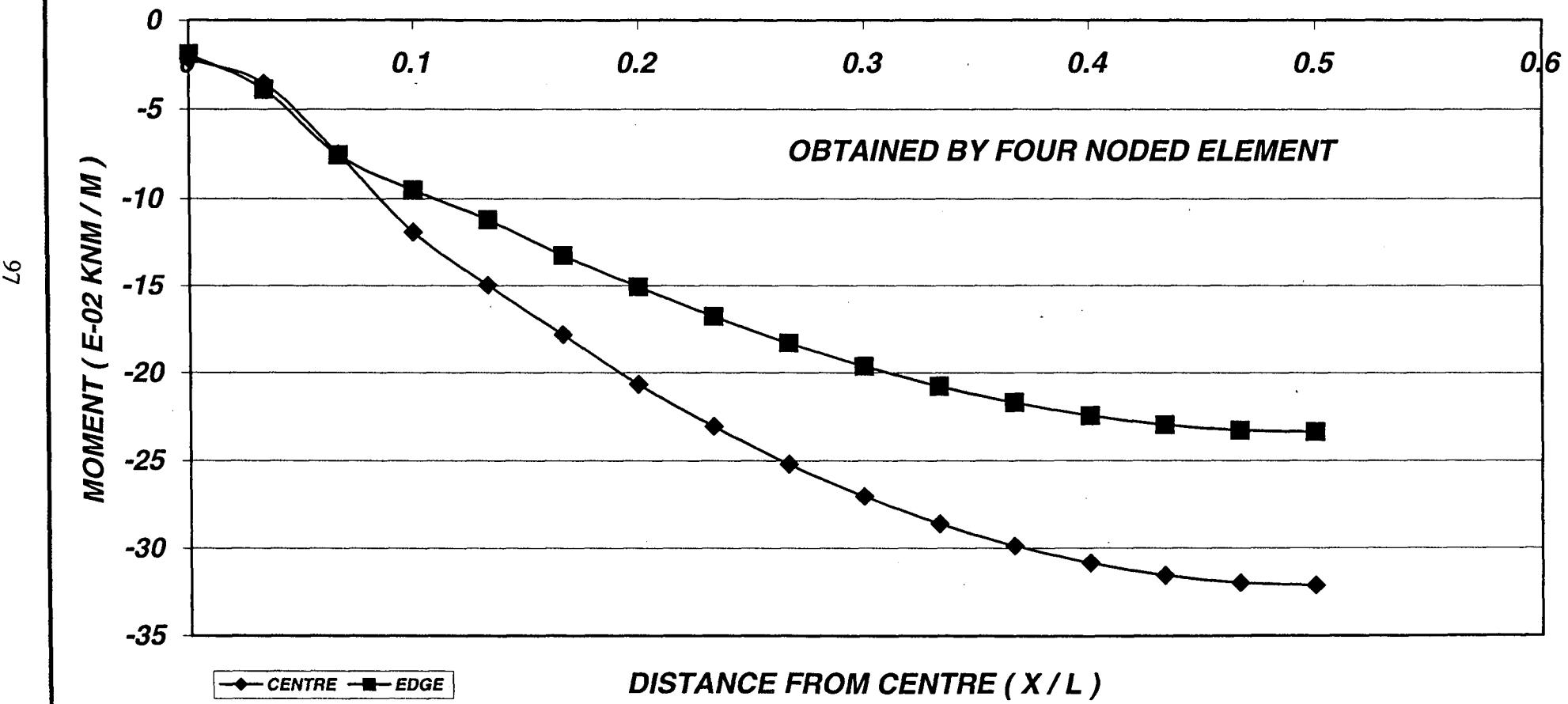
GRAPH 6.31

TRANSVERSE MOMENT ALONG THE LENGTH
(6M SPAN , 60MM TH.)



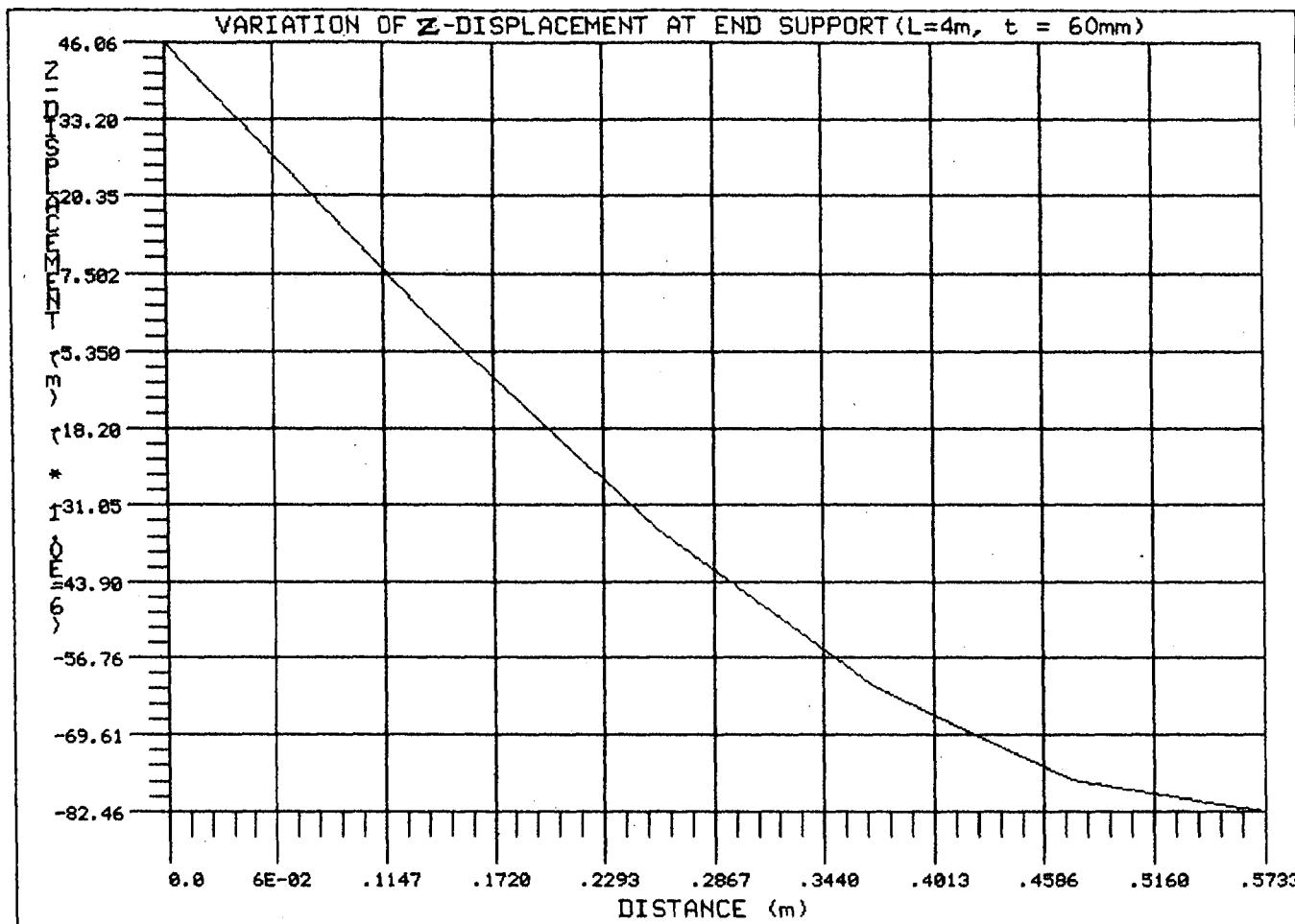
GRAPH 6.32

LONGITUDINAL MOMENT ALONG THE LENGTH
(6M SPAN , 60MM TH.)



GRAPH 6.33

DISPLAY III - GEOMETRY MODELING SYSTEM (7.2.0) PRE/POST MODULE

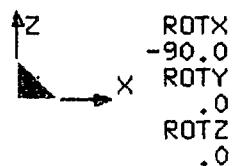


MIDDLE LAYER

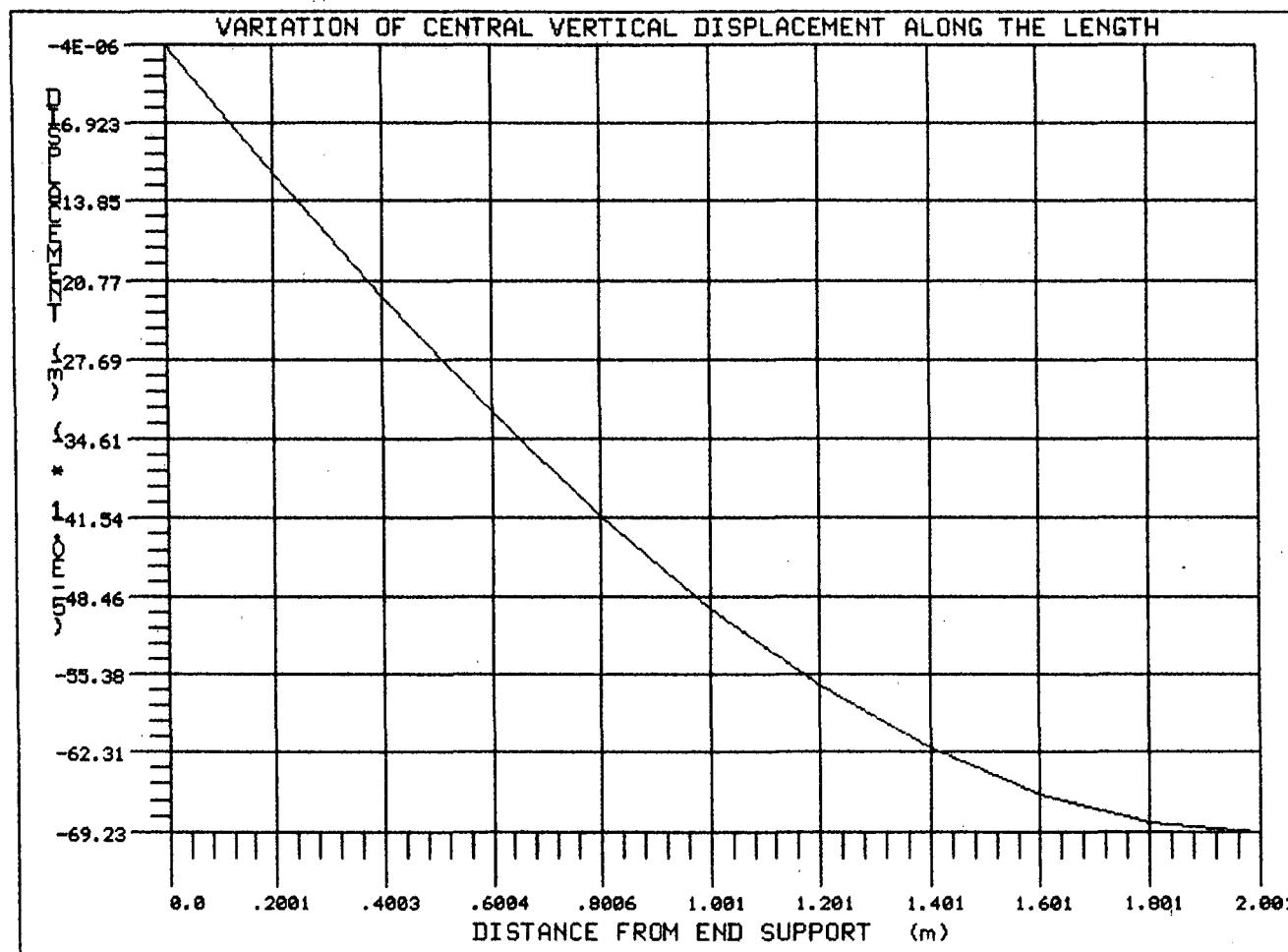
Graph A.1

EMRC-NISA/DISPLAY

NOV/25/98 15:00:15



DISPLAY III - GEOMETRY MODELING SYSTEM (7.2.0) PRE/POST MODULE

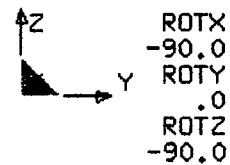


EMRC

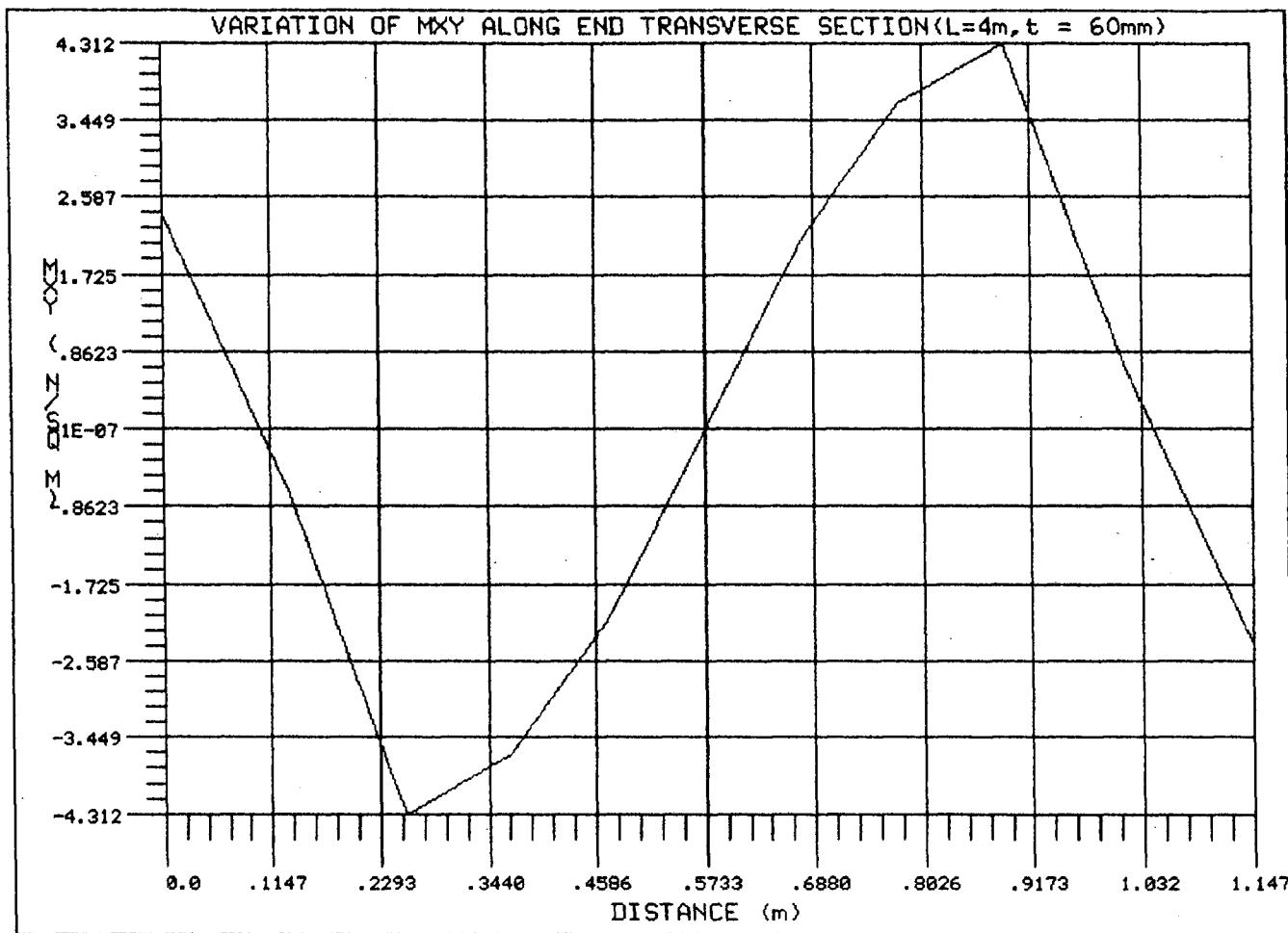
Graph A.2

EMRC-NISA/DISPLAY

NOV/25/98 12:38:59



DISPLAY III - GEOMETRY MODELING SYSTEM (7.2.0) PRE/POST MODULE

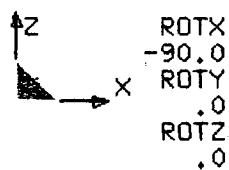


MIDDLE LAYER

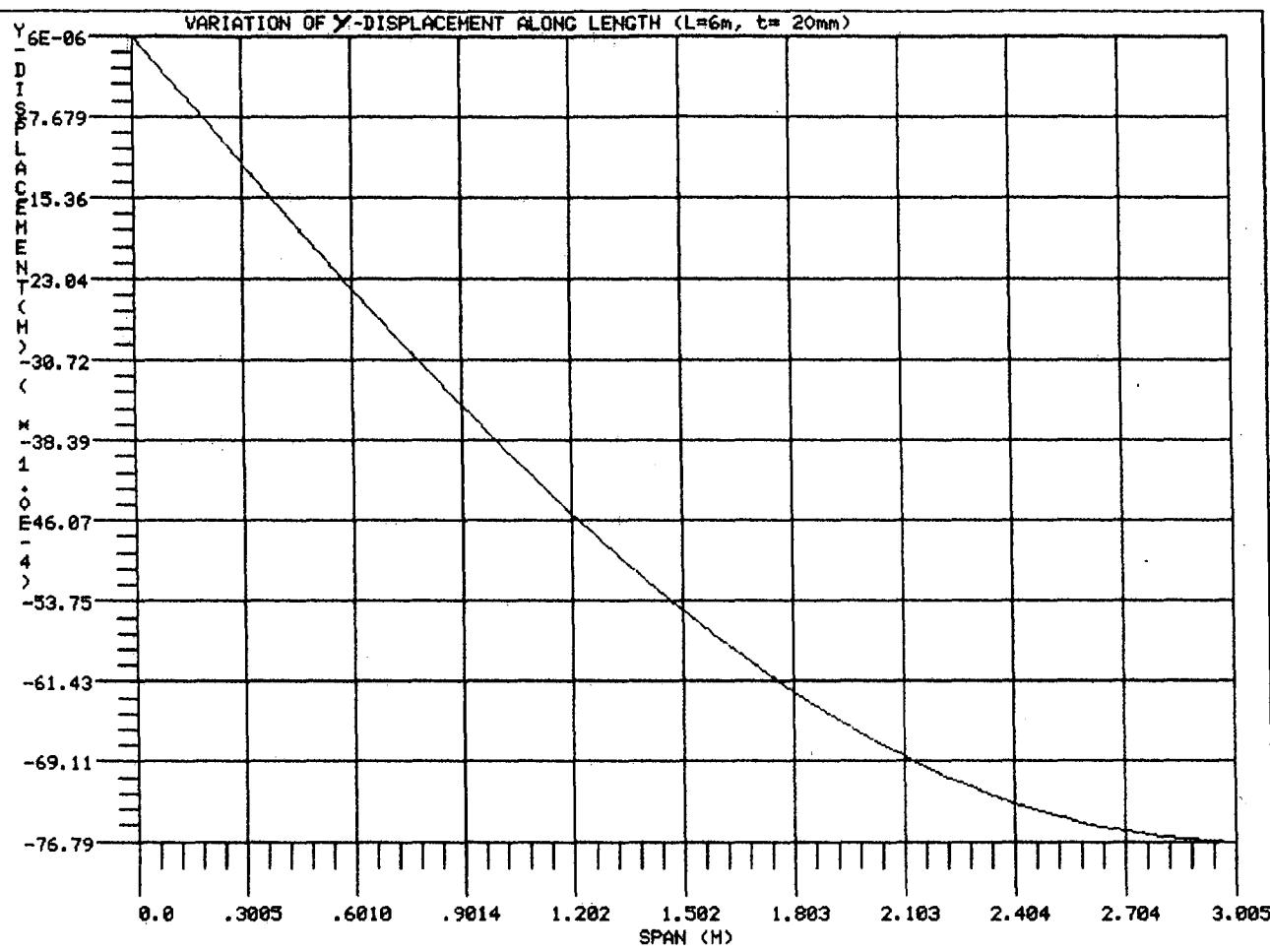
Graph A.3

EMRC-NISA/DISPLAY

NOV/25/98 15:02:04



DISPLAY III - GEOMETRY MODELING SYSTEM (7.2.0) PRE/POST MODULE

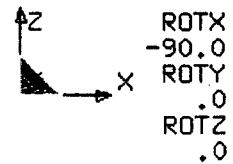


101

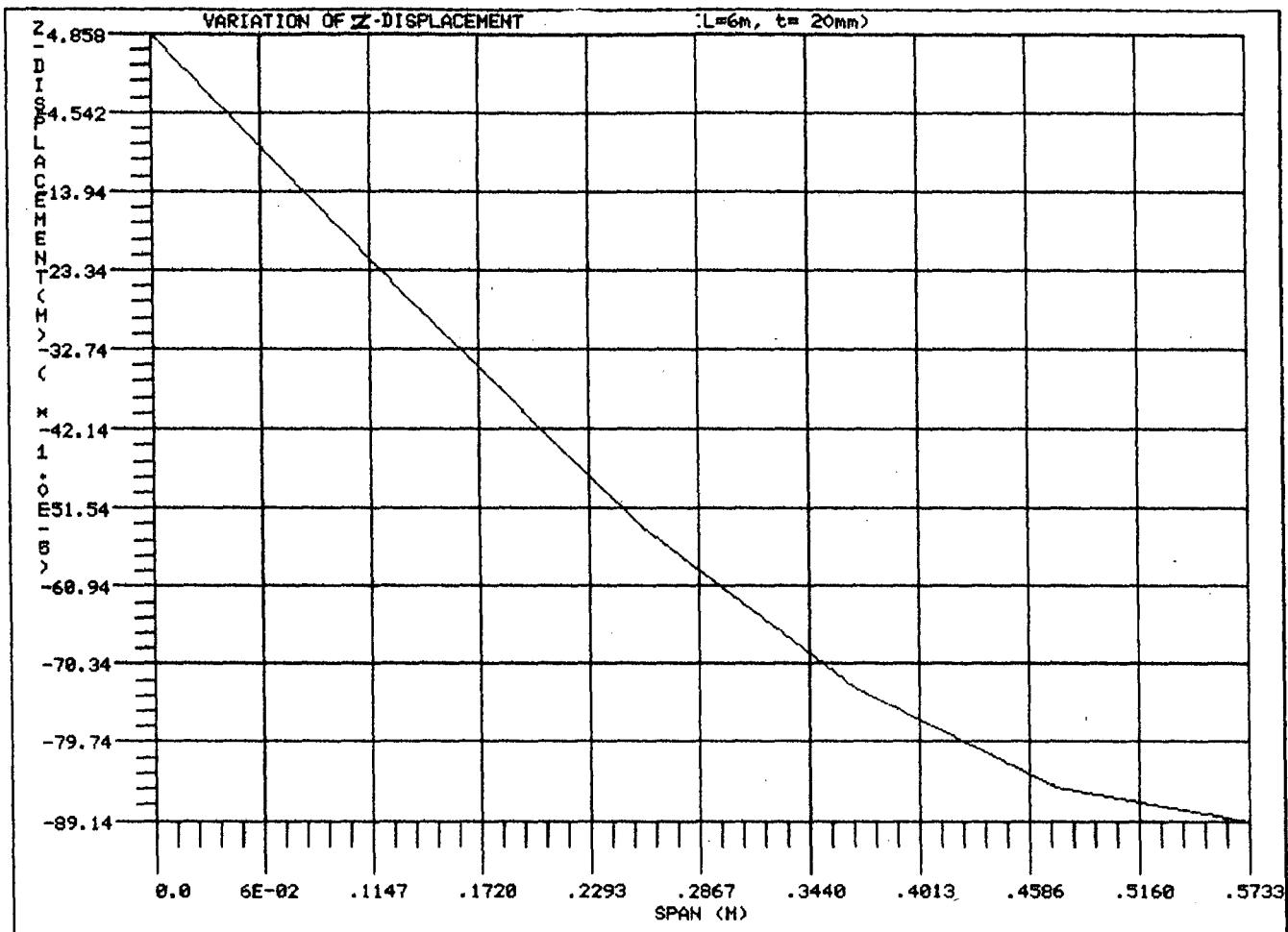


Graph A.4

EMRC-NISA/DISPLAY
NOV/25/98 15:41:39

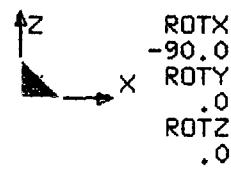


DISPLAY III - GEOMETRY MODELING SYSTEM (7.2.0) PRE/POST MODULE



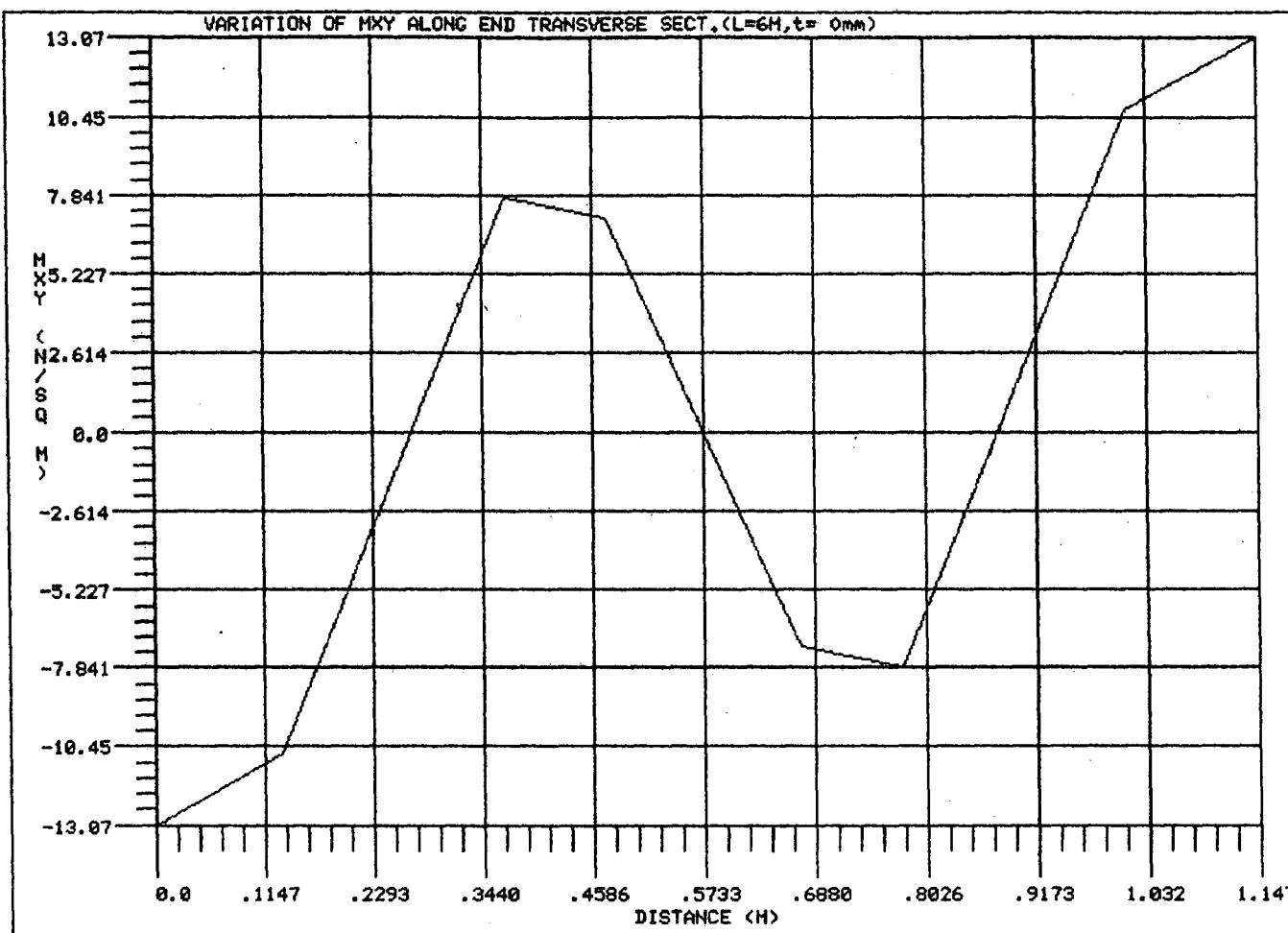
Graph A.5

EMRC-NISA/DISPLAY
NOV/25/98 15:43:20



103

DISPLAY III - GEOMETRY MODELING SYSTEM (7.2.0) PRE/POST MODULE

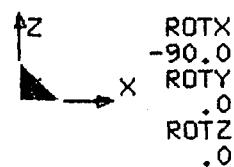


EM
RC

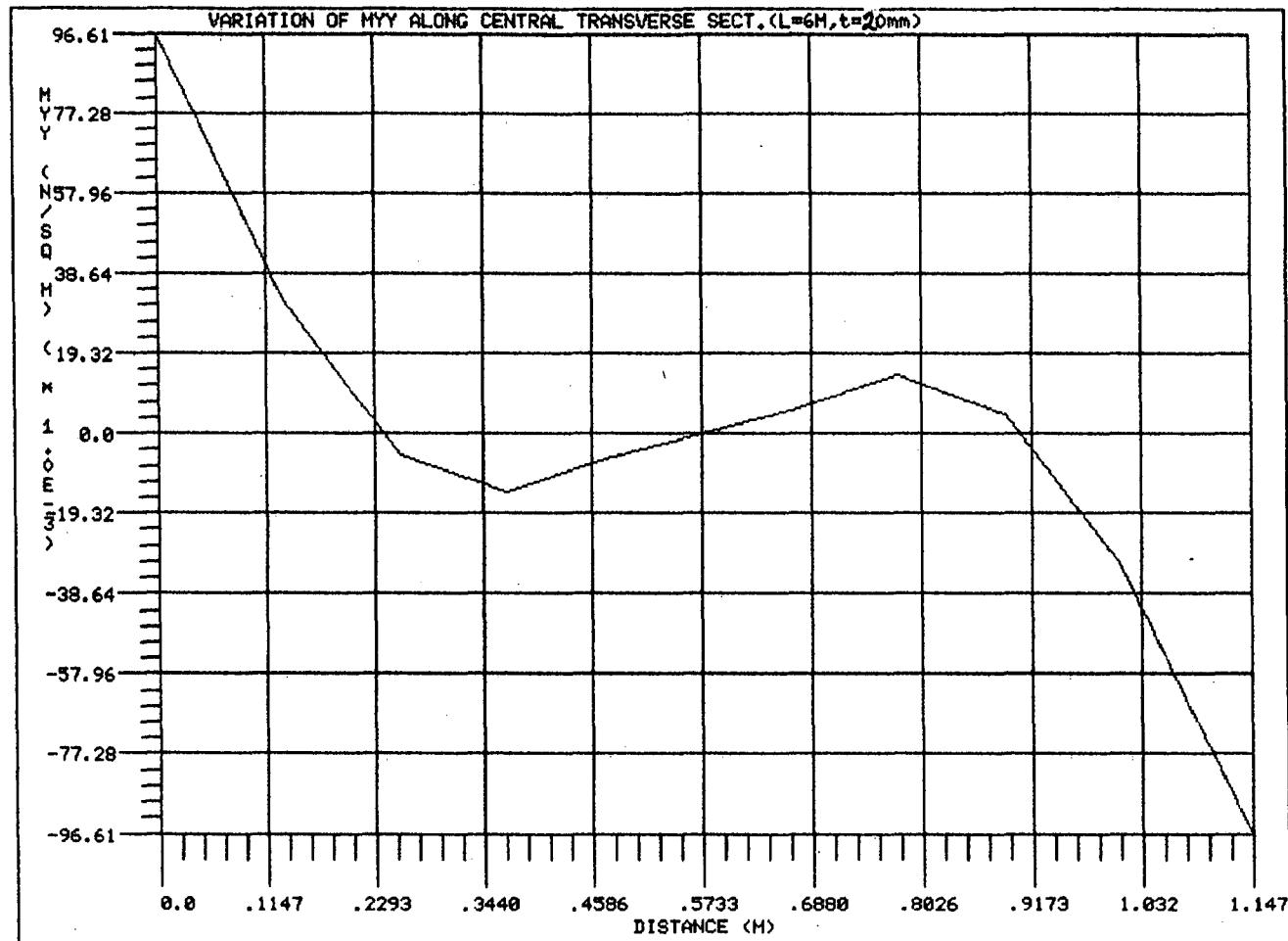
Graph A.6

EMRC-NISA/DISPLAY

NOV/25/98 15:50:07



DISPLAY III - GEOMETRY MODELING SYSTEM (7.2.0) PRE/POST MODULE

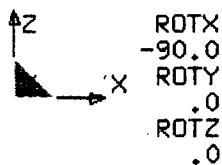


EMRC

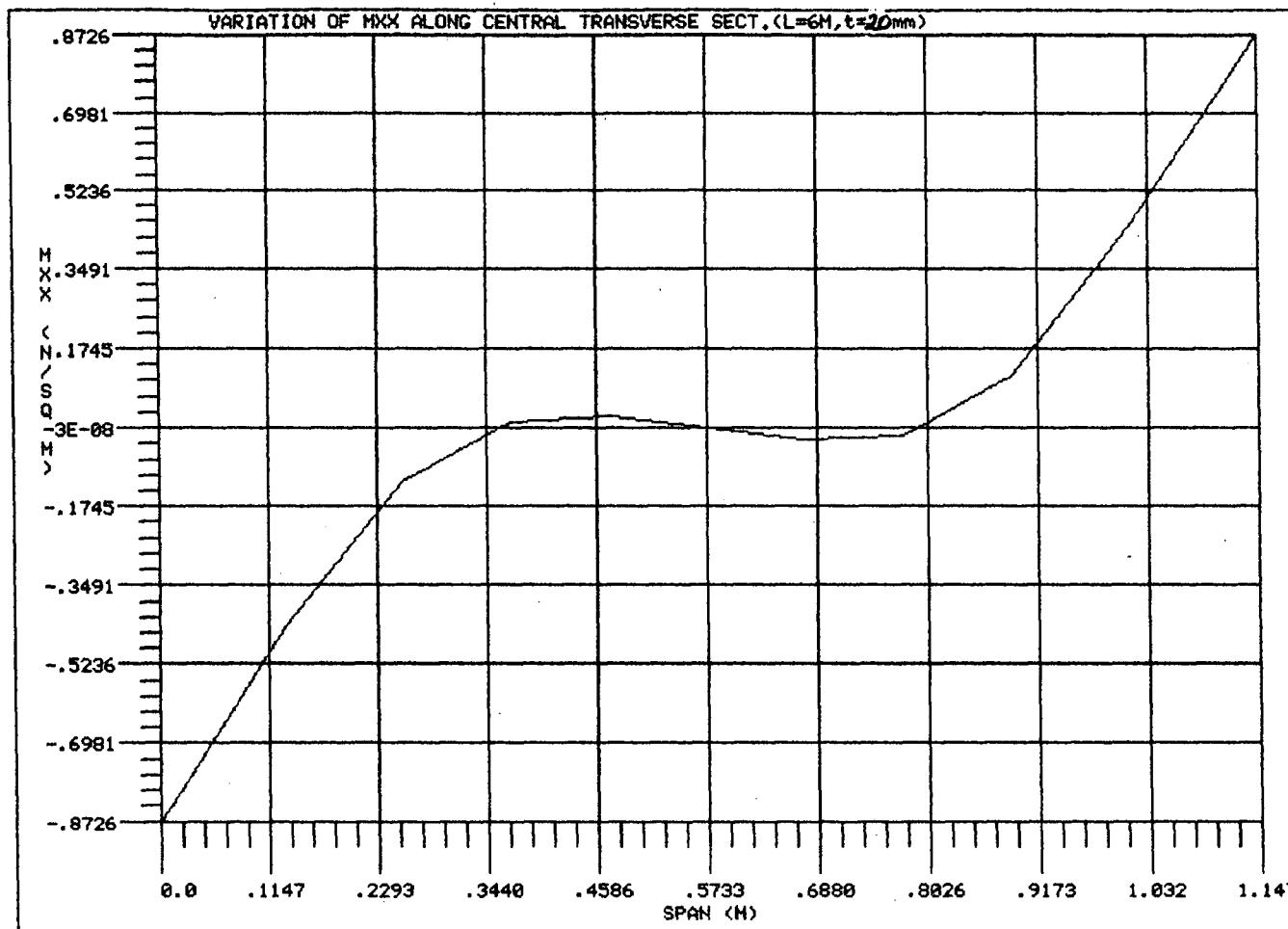
Graph A.7

EMRC-NISA/DISPLAY

NOV/25/98 15:48:38



DISPLAY III - GEOMETRY MODELING SYSTEM (7.2.0) PRE/POST MODULE



105

EM
RC

Graph A.8

ENRC-NISA/DISPLAY

NOV/25/98 15:46:56


 AZ ROTX -90.0
 X ROTY 0
 Z ROTZ 0

SHELL.C

A PROGRAM

In

'C' LANGUAGE

For

STRESS RESULTANTS

And

DISPLACEMENTS

Of

GENERAL SHELL

Using

FOUR NODDED BILINEAR ELEMENT

And

FRONTAL SOLUTION TECHNIQUE

```

/* MAINPROGRAM SHELL.C for Analysis of Shells*/
/*Frontal Solution Technique as Solver */

#define D 6
#define NCN 4
#define NMAT 1
#define K 0.833333
#define NULL 0
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <alloc.h>
#include <conio.h>
#include <graphics.h>

FILE *p1,*p2,*p3,*p4,*p5,*p6;
double x[NCN]={0}, y[NCN]={0},z[NCN]={0}, th[NCN]={0},cee[3][3];
double *aj,*ael,*ar,*ac,*dj,*jdl,*gp,*esm1,*equat,*vec;
double p,ort[NMAT][3];
int *jrl,*imat,*itype,*nacva,*locel,*ndest;
long int M,M1;
int ltype,ne,nd,n,np,nrj,ncn=4,M_fron;
int tp,mat,nn(),NIKNO,kelva=0,nop[NCN],nop1[NCN];
int XX=425,YY=350,MF=75;

main()
{
    clrscr();
    p1=fopen("fem.fil","r+");
    p2=fopen("coord.fil","w+");
    p3=fopen("nop.fil","w+");
    p4=fopen("gstif.fil","w+");
    p5=fopen("d.t","w+");
    Front_sol();
    fclose(p4);
    fclose(p5);
    fclose(p6);
}
/*********/
Front_sol()
/*********/

```

```

{
int l;
gdata();
graf();
mm_fron();
fclose(p1);
fp_files();
for(l=0;l<ne;l++)
{
reread();
locel();
destination();
ldata();
stiff();
trance();
eli_exn();
}
p6=fopen("resul.fil","w+");
bp_files();
back_substitution();
results();
}
*****
gdata()
*****
{
data1();
data2();
data3();
data4();
data5();
data6();
}
*****
graf()
*****
{
graf1();

```

```

graf2();
graf4();
}
*****
data1()
*****
{
double e,mu,den;
int j,k,nr;
/*****************************************/
fscanf(p1,"%d%d%d%d%d%d\n",&ne,&np,&nr,&nrj,&mat,&tp);
fprintf(p5,"\\t\\t\\t*****SHELL ANALYSIS *** 4 NODDED-ELEMENT*****\\n\\n");
fprintf(p5,"\\t**** MESH PARAMETERS ****\\n\\n");
fprintf(p5,"    NE\\tN\\tNP\\tNR\\tNRJ\\tNMAT\\tNTYPE\\n\\n");
nd=D*np;
n=nd-nr;
fprintf(p5,"%6d%6d%6d%6d%6d%6d\\n\\n",ne,n,np,nd,nd,mat,tp);
/*****************************************/
fprintf(p5,"\\t** MATERIAL PROPERTY **\\n\\n\\n");
fprintf(p5,"MATERIAL           E           MU           DENSITY \\n\\n");
for(k=0;k<mat;k++)
{
fscanf(p1,"%d%lf%lf%lf\\n",&j,&e,&mu,&den);
j=j-1;
ort[j][0]=e;
ort[j][1]=mu;
ort[j][2]=den;
fprintf(p5,"%9d%16.2lf%16.2lf%16.2lf\\n",j+1,e,den);
}
}
*****
data2()
*****
{
double x1,y1,z1,th1,*gp1;
int i,j,k;
/*****************************************/
if((gp1=(double*)malloc((np*4)*sizeof(double)))==NULL)

```

```

{
printf("NO SPACE AVAILKABLE *GP1*");
exit(1);
}
fprintf(p5,"\\n\\n\\t** NODE POINT COORDINATES **\\n\\n\\n");
fprintf(p5,"NODE POINT\\t X\\t Y\\t Z\\t TH \\n\\n");
for(k=0;k<np;k++)
{
fscanf(p1,"%d%lf%lf%lf%lf\\n",&j,&x1,&y1,&z1,&th1);
j=j-1;
*(gp1+4*j)=x1;
*(gp1+4*j+1)=y1;
*(gp1+4*j+2)=z1;
*(gp1+4*j+3)=th1;
}
for(i=0;i<np;i++)
{
fprintf(p5,"%6d",i+1);
for(j=0;j<4;j++)
fprintf(p5,"%12.3lf",*(gp1+4*i+j));
fprintf(p5,"\\n");
}
store2(gp1);
free(gp1);
}
/***************/
store2(double *gp1)
/***************/
{
int i,j;
for(i=0;i<np;i++)
{
for(j=0;j<ncn;j++)
fprintf(p2,"%12.3lf",*(gp1+ncn*i+j));
fprintf(p2,"\\n");
}
M=(ftell(p2)/np);
}

```

```

*****
data3()
*****
{
long m;
int i,j,k,l,*nop2;
if((nop2=(int*)malloc((ne*(ncn+2))*sizeof(int)))==NULL)
{
printf("NO SPACE AVAILKABLE *nop2* ");
exit(1);
}
m=ftell(p1);
/*-----*/
for(l=0;l<ne;l++)
{
fscanf(p1,"%d",&i);
i=i-1;
for(k=0;k<(ncn+2);k++)
fscanf(p1,"%d",*(nop2+i*(ncn+2)+k));
}
fprintf(p5,"\\n\\n\\t\\t** ELEMENT INFORMATION **\\n\\n\\n");
fprintf(p5,"ELEMENT\\t\\tTYPE\\t\\tMAT\\tTH\\t\\tJ\\tK\\tL\\n\\n\\n");
for(l=0;l<ne;l++)
{
fprintf(p5,"%6d",l+1);
for(k=0;k<(ncn+2);k++)
fprintf(p5,"%6d",*(nop2+l*(ncn+2)+k));
fprintf(p5,"\\n");
}
free(nop2);
store3(m);
}
*****
store3(long m)
*****
{
int i,k,l,*nop1,*nop2;
if((nop1=(int*)malloc((ne*2)*sizeof(int)))==NULL)

```

```

{
printf("NO SPACE AVAILKABLE *nop1* ");
exit(1);
}
if((nop2=(int*)malloc((ne*ncn)*sizeof(int)))==NULL)
{
printf("NO SPACE AVAILKABLE *nop2* ");
exit(1);
}
fseek(p1,m,0);
for(l=0;l<ne;l++)
{
fscanf(p1,"%d",&i);
i=i-1;
for(k=0;k<2;k++)
fscanf(p1,"%d",*(nop1+i*2+k));
for(k=0;k<ncn;k++)
fscanf(p1,"%d",*(nop2+i*ncn+k));
}
last_appearance(nop2);
for(l=0;l<ne;l++)
{
fprintf(p3,"%4d %4d",*(nop1+l*2+1),*(nop1+l*2)-1));
for(k=0;k<ncn;k++)
fprintf(p3," %4d",*(nop2+ncn*l+k));
fprintf(p3,"\n");
}
rewind(p3);
free(nop1);
free(nop2);
}
*****
data4()
*****
{
int i,j,k,kdj;
if((jrl=(int*)malloc((np*D)*sizeof(int)))==NULL)
{

```

```

printf("NO SPACE AVAILKABLE *JRL*");
exit(1);
}
for(j=0;j<np*D;j++)
*(jrl+j)=0;
fprintf(p5,"\\n\\n\\t\\t** NODE POINT RESTRAINTS **\\n\\n\\n");
fprintf(p5,"NODE POINT   NPR1   NPR2   NPR3   NPR4   NPR5   NPR6\\n\\n");
for(j=0;j<nrj;j++)
{
fscanf(p1,"%d",&k);
fprintf(p5,"\\n%3d",k);
k=k-1;
for(i=0;i<D;i++)
{
fscanf(p1,"%d",(jrl+k*D+i));
fprintf(p5,"%10d",*(jrl+k*D+i));
}
}
*(jrl+229*D)=1;
*(jrl+229*D+2)=1;
*(jrl+230*D)=1;
*(jrl+230*D+2)=1;
/*****************************************/
/*if((jdl=(double*)malloc((np*D)*sizeof(double)))==NULL)
{
printf("NO SPACE AVAILKABLE *JDL*");
exit(1);
}
for(j=0;j<np*D;j++)
*(jdl+j)=0;
fscanf(p1,"%d",&kdj);
if(kdj !=0 )
{
fprintf(p5,"\\n\\n\\t\\t** NODE POINT DISPLACEMENTS **\\n\\n\\n");
fprintf(p5,"NODE POINT   D1   D2   D3   D4   D5   D6\\n\\n");
}
for(j=0;j<kdj;j++)
{

```

```

fscanf(p1,"%d",&k);
fprintf(p5,"n%3d",k);
k=k-1;
for(i=0;i<D;i++)
{
fscanf(p1,"%lf",(jdl+k*D+i));
fprintf(p5,"% .4lf",*(jdl+k*D+i));
}
}/*
}
/*****
last_appearance(int *nop2)
/*****
{
int i,j,k;
int *ptr;
k=ne*ncn;
ptr=nop2+k-1;
for(i=1;i<=np;i++)
for(j=0;j<k;j++)
if(i==*(ptr-j))
{
*(ptr-j)=-*(ptr-j);
break;
}
}
/*****
int nn(int i,int j)
/*****
{
int nn1 ;
if(i>j)
nn1=(i*i+i)/2+j;
else
nn1=(j*j+j)/2+i;
return(nn1);
}
*****/

```

```

trance()
/********/
{
int i,j,k,idest,jdest,nsize;
nsize=ncn*D;
for(i=0;i<nsize;i++)
{
idest=*(ndest+i);
*(nacva+idest)=abs(*(locel+i));
*(ac+idest)=*(ac+idest)+*(ael+i)+*(aj+(abs(*(locel+i))-1));
*(aj+(abs(*(locel+i))-1))=0.0;
for(j=i;j<nsize;j++)
{
jdest=*(ndest+j);
k=nn(idest,jdest);
*(gp+k)=*(gp+k)+*(esm1+nn(i,j));
}
}
}
}

/********/
fp_files()
/********/
{
int j;
if((locel=(int*)malloc((ncn*D)*sizeof(int)))==NULL)
{
printf("NO SPACE AVAILKABLE *LOCEL*");
exit(1);
}
if((ndest=(int*)malloc((ncn*D)*sizeof(int)))==NULL)
{
printf("NO SPACE AVAILKABLE *NDEST*");
exit(1);
}
if((gp=(double*)malloc((M_fron*(M_fron+1)/2)*sizeof(double)))==NULL)
{
printf("NO SPACE AVAILKABLE *GP*");
}

```

```

exit(1);
}

for(j=0;j<(M_fron*(M_fron+1)/2);j++)
*(gp+j)=0;
if((nacva=(int*)malloc(M_fron*sizeof(int)))==NULL)
{
printf("NO SPACE AVAILKABLE *NACVA*");
exit(1);
}
for(j=0;j<M_fron;j++)
*(nacva+j)=0;
if((ac=(double*)malloc(M_fron*sizeof(double)))==NULL)
{
printf("NO SPACE AVAILKABLE *AC*");
exit(1);
}
for(j=0;j<M_fron;j++)
*(ac+j)=0;
if((equat=(double*)malloc(M_fron*sizeof(double)))==NULL)
{
printf("NO SPACE AVAILKABLE *EQUAT*");
exit(1);
}
if((ael=(double*)malloc((ncn*D)*sizeof(double)))==NULL)
{
printf("NO SPACE AVAILKABLE *AEL*");
exit(1);
}
for(j=0;j<(ncn*D);j++)
*(ael+j)=0;
if((esm1=(double*)malloc(300*sizeof(double)))==NULL)
{
printf("NO SPACE AVAILKABLE *ESM1*");
exit(1);
}
}
*******/

bp_files()

```

```

*****
{
free(a);
free(nacva);
free(gp);
free(ael);
free(locel);
free(ndest);
}
*****
reread()
*****
{
int i;
fscanf(p3,"%d%d",&tp,&mat);
for(i=0;i<ncn;i++)
{
fscanf(p3,"%d",&nop1[i]);
nop[i]=abs(nop1[i])-1;
}
for(i=0;i<ncn;i++)
{
fseek(p2,M*nop[i],0);
fscanf(p2,"%lf%lf%lf%lf",&x[i],&y[i],&z[i],&th[i]);
}
}
*****
locel()
*****
{
int i,j,k,k1;
for(i=0;i<ncn;i++)
{
if(nop1[i]<0)
k1=-1;
else
k1=1;
for(k=1;k<=D;k++)

```

```

*(locel+i*D+k-1)=k1*D*(nop1[i]*k1-1)+k*k1;
}

}

*******/

data6()
*******/

{
fscanf(p1,"%d",&ltype);
if(ltype !=1)
fscanf(p1,"%lf",&p);
fprintf(p5,"\\n\\nLTYPE =%3d  LOADING INTENSITY (p)=%12.3lf\\n",ltype,p);
}
*******/

ldata()
*******/

{
if(ltype==1)
ldata1();
if(ltype==2)
ldata2();
if(ltype==3)
ldata3();
}
*******/

ldata1()
*******/

{
int i,j,k,l,m1,m2,nsize=D*ncn,grs[2]={-1,1};
int nrs[4][2]={{-1,-1},{1,-1},{1,1}, {-1,1}};
double w,a,n1,dj,jacobian(),d1[3][3],d[3][3],ji[3][3],rs[2];
for(j=0;j<nsize;j++)
*(ael+j)=0;
/*-----*/
for(m1=0;m1<2;m1++)
for(m2=0;m2<2;m2++)
{
rs[0]=0.577350269189626*grs[m1];

```

```

rs[1]=0.577350269189626*grs[m2];
w=1.0;
dj=jacobian(ji,nrs,rs);
/*-----*/
for(i=0;i<4;i++)
{
n1=0.25*(1.+nrs[i][0]*rs[0])*(1.+nrs[i][1]*rs[1]);
*(ael+D*i+2)=*(ael+D*i+2)+w*n1*dj*ort[mat][2];
}
/*-----*/
}
/*-----*/
/*fprintf(p5,"\\n\\n*AEL*\\n\\n");
for(i=0;i<nsize;i++)
fprintf(p5,"%12.3lf\\n",*(ael+i));
getch(); */
/*-----*/
}
*****/
ldata2()
*****/
{
int i,j,k,l,m1,m2,nsize=D*ncn,grs[2]={-1,1};
int nrs[4][2]={{-1,-1},{1,-1},{1,1}, {-1,1}};
int rot1[3]={0,1,2},rot2[3]={1,2,0},rot3[3]={2,0,1};
double w,a,n1,dj,jacobian(),d1[3][3],d[3][3],ji[3][3],rs[2],rs1[2];
for(j=0;j<nsize;j++)
*(ael+j)=0;
/*-----*/
for(m1=0;m1<2;m1++)
for(m2=0;m2<2;m2++)
{
rs[0]=0.577350269189626*grs[m1];
rs[1]=0.577350269189626*grs[m2];
w=1.0;
/*-----*/
d_cosine(d1,nrs,rs);

```

```

dj=jacobian(ji,nrs,rs);
a=dj*sqrt(ji[0][2]*ji[0][2]+ji[1][2]*ji[1][2]+ji[2][2]*ji[2][2]);
for(i=0;i<4;i++)
{
n1=0.25*(1.+nrs[i][0]*rs[0])*(1.+nrs[i][1]*rs[1]);
rs1[0]=1.*nrs[i][0];
rs1[1]=1.*nrs[i][1];
d_cosine(d,nrs,rs1);
for(j=0;j<3;j++)
{
*(ael+D*i+j)=*(ael+D*i+j)+w*n1*p*d1[rot1[j]][2]*a;
a=a*(d1[rot3[j]][2]*d[rot2[j]][2]-d1[rot2[j]][2]*d[rot3[j]][2]);
*(ael+D*i+3+j)=*(ael+D*i+3+j)+w*0.5*th[i]*p*a;
}
}
/*
*******/
}
/*
*******/
/*fprintf(p5,"\\n\\n*AEL*\\n\\n");
for(i=0;i<nsize;i++)
fprintf(p5,"%12.3lf\\n",*(ael+i));
getch(); */
/*
*******/
******/
ldata3()
******/
{
int i,j,k,l,m1,m2,nsize=D*ncn,grs[2]={-1,1};
int nrs[4][2]={{-1,-1},{1,-1},{1,1}, {-1,1}};
double w,a,n1,dj,jacobian(),d1[3][3],d[3][3],ji[3][3],rs[2],rs1[2];
for(j=0;j<nsize;j++)
*(ael+j)=0;
/*
*******/
for(m1=0;m1<2;m1++)
for(m2=0;m2<2;m2++)
{
rs[0]=0.577350269189626*grs[m1];

```

```

rs[1]=0.577350269189626*grs[m2];
w=1.0;
/*-----*/
dj=jacobian(ji,nrs,rs);
a=dj*sqrt(ji[0][2]*ji[0][2]+ji[1][2]*ji[1][2]+ji[2][2]*ji[2][2]);
for(i=0;i<4;i++)
{
n1=0.25*(1.+nrs[i][0]*rs[0])*(1.+nrs[i][1]*rs[1]);
rs1[0]=1.*nrs[i][0];
rs1[1]=1.*nrs[i][1];
d_cosine(d,nrs,rs1);
*(ael+D*i+2)=*(ael+D*i+2)+w*p*n1*a;
*(ael+D*i+3)=*(ael+D*i+3)+w*0.5*p*th[i]*n1*d[1][2]*a;
*(ael+D*i+4)=*(ael+D*i+4)-w*0.5*p*th[i]*n1*d[0][2]*a;
}
/*-----*/
}
/*-----*/
/*fprintf(p5,"\\n\\n*AEL---LTYPE=%3d\\n\\n",ltype);
for(i=0;i<nsize;i++)
fprintf(p5,"%12.3lf\\n",*(ael+i));*/
/*-----*/
}
*****/
data5()
*****/
{
int i,j,k,nlj;
fscanf(p1,"%d",&nlj);
if(nlj != 0)
{
fprintf(p5,"\\nNo of loaded Joints = %3d",nlj);
fprintf(p5,"\\n\\nt *****NODAL LOADS*****\\n\\n");
fprintf(p5,"\\nNODE    AJ1    AJ2    AJ3    AJ4    AJ5    AJ6\\n");
}
/*-----*/
if((aj=(double*)malloc((np*D)*sizeof(double)))==NULL)
{

```

```

printf("NO SPACE AVAILKABLE *AJ*");
exit(1);
}
for(j=0;j<np*D;j++)
*(aj+j)=0;
for(j=0;j<nlj;j++)
{
fscanf(p1,"%d",&k);
fprintf(p5,"%16d",k);
for(i=0;i<D;i++)
{
fscanf(p1,"%lf",(aj+D*(k-1)+i));
fprintf(p5,"%10.3lf",*(aj+D*(k-1)+i));
}
}
}

/*********/
eli_exn()
/*********/
{
int i,j,k,m,h,gn,nsize=ncn*D,msize=M_fron/D;
double gload,pivot;
for(m=0;m<nsize;m++)
{
/*-----*/
if(*(locel+m)<0)
{
NIKNO=-*(locel+m)-1;
h=*(ndest+m);
/*-----*/
*(nacva+h)=0;
for(i=0;i<M_fron;i++)
{
*(equat+i)=*(gp+nn(h,i));
*(gp+nn(h,i))=0.0;
}
gload=*(ac+h);
}
}

```

```

*(ac+h)=0.0;
fprintf(p4,"%5d %5d %28.16lf\n",NIKNO,h,gload);
for(i=0;i<msize;i++)
{
    for(j=0;j<D;j++)
        fprintf(p4,"%28.16lf",*(equat+i*D+j));
    fprintf(p4,"\n");
}
/*-----*/
pivot=*(equat+h);
*(equat+h)=0.0;
if(*(jrl+NIKNO)==0)
{
    for(i=0;i<M_fron;i++)
    {
        *(ac+i)=*(ac+i)-*(equat+i)*gload/pivot;
        for(j=i;j<M_fron;j++)
        {
            gn=nn(i,j);
            *(gp+gn)=*(gp+gn)-*(equat+i)**(equat+j)/pivot;
        }
    }
}
/*-----*/
else
{
    for(i=0;i<M_fron;i++)
        *(ac+i)=*(ac+i)-0.0**(equat+i);
}
/*-----*/
kelva=kelva+1;
}
}
}
/*******/
back_substitution()
/*******/
{
long m;
int i,j,k,h;

```

```

double gload,sum,pivot,pivot1,x1=0.0;
M1=(ftell(p4)/kelva);
for(j=0;j<M_fron;j++)
*(ac+j)=0;
for(k=0;k<kelva;k++)
{
sum=0.0;
fseek(p4,M1*(kelva-1-k),0);
fscanf(p4,"%d%d%lf",&NIKNO,&h,&gload);
for(i=0;i<M_fron;i++)
fscanf(p4,"%lf",(equat+i));
pivot1=*(equat+h);
if(*(jrl+NIKNO)==0)
{
pivot=*(equat+h);
*(equat+h)=0.0;
for(i=0;i<M_fron;i++)
sum=sum+*(equat+i)**(ac+i);
*(ac+h)=(gload-sum)/pivot;
fprintf(p6,"%6d %12.5lf %12.3lf\n",NIKNO,* (ac+h),x1);
}
else
{
*(ac+h)=0.0;
for(i=0;i<M_fron;i++)
sum=sum+*(equat+i)**(ac+i);
fprintf(p6,"%6d %12.5lf %12.3lf\n",NIKNO,x1,-(gload-sum));
}
printf("BACK-> PIVOT=%8.4e JRL=%4d KELVA=%6d\n",pivot1,* (jrl+kelva-k-1),(kelva-k));
}
rewind(p6);
}
/*********/
destination()
/*********/
{
int i,j,kexis;

```

```

for(i=0;i<ncn*D;i++)
{
kexis=0;
for(j=0;j<M_fron;j++)
if(*(nacva+j)==abs(*(locel+i)))
{
*(ndest+i)=j;
kexis=1;
break;
}
if(kexis==0)
for(j=0;j<M_fron;j++)
if(*(nacva+j)==0)
{
*(ndest+i)=j;
*(nacva+j)=-1;
kexis=1;
break;
}
if(kexis==0)
{
printf("FATAL ERROR *****--***** INCRESE M_FRON");
exit(1);
}
}
}

*******/

mm_fron()
*******/

{
int l,i,j,k,fron1=0,fron=0,mfron=ncn;
/*-----*/
if((nacva=(int*)malloc((ne*ncn)*sizeof(int)))==NULL)
{
printf("NO SPACE AVAILKABLE M_fron");
exit(1);
}
for(j=0;j<ne*ncn;j++)

```

```

*(nacva+j)=0;
/*-----*/
for(l=0;l<ne;l++)
{
fscanf(p3,"%d%d",&tp,&mat);
for(k=0;k<ncn;k++)
fscanf(p3,"%d",(nacva+fron1+k));
/*-----*/
fron=fron1+ncn;
/*-----*/
for(i=0;i<fron1;i++)
for(j=fron1;j<fron;j++)
if(*(nacva+i)==abs(*(nacva+j)))
{
*(nacva+i)=*(nacva+j);
for(k=j;k<fron-1;k++)
*(nacva+k)=*(nacva+k+1);
fron=fron-1;
j=fron;
}
/*-----*/
if(fron>mfron)
mfron=fron;
/*-----*/
for(j=0;j<fron;j++)
if(*(nacva+j)<0)
{
for(k=j;k<fron-1;k++)
*(nacva+k)=*(nacva+k+1);
fron=fron-1;
j=j-1;
}
/*-----*/
fron1=fron;
}
/*-----*/
M_fron=mfron*D;
/*-----*/

```

```

printf("M_fron=%4d",M_fron);
getch();
rewind(p3);
free(nacva);
}
/*****************************************/
st1a_shell(int nrs[4][2])
/*****************************************/
{
int i,j,k,m1,m2,i1,j1,grs[2]={-1,1},l;
double w,dj,jacobian(),b[3][3],temp[3][3];
double k3[3][3],k6[6][6],ji[3][3],rs[2];
/*-----*/
for(m1=0;m1<2;m1++)
for(m2=0;m2<2;m2++)
{
rs[0]=0.577350269189626*grs[m1];
rs[1]=0.577350269189626*grs[m2];
w=1.0;
dj=jacobian(ji,nrs,rs);
/*-----*/
for(i1=0;i1<4;i1++)
for(j1=i1;j1<4;j1++)
{
/*-----*/
for(i=0;i<6;i++)
for(j=0;j<6;j++)
k6[i][j]=0.0;
for(i=0;i<3;i++)
for(j=0;j<3;j++)
{
temp[i][j]=0.0;
k3[i][j]=0.0;
}
/*-----*/
l=1;
bee(l,b,ji,nrs,j1,rs);
for(i=0;i<3;i++)

```

```

for(j=0;j<3;j++)
for(k=0;k<3;k++)
temp[i][j]=temp[i][j]+cee[i][k]*b[k][j];
bee(l,b,ji,nrs,i1,rs);
for(i=0;i<3;i++)
for(j=0;j<3;j++)
for(k=0;k<3;k++)
k3[i][j]=k3[i][j]+w*2.*dj*b[k][i]*temp[k][j];
for(i=0;i<3;i++)
for(j=0;j<3;j++)
{
k6[i][j]=k3[i][j];
k3[i][j]=0.0;
temp[i][j]=0.0;
}
/*-----*/
l=2;
bee(l,b,ji,nrs,j1,rs);
for(i=0;i<3;i++)
for(j=0;j<3;j++)
for(k=0;k<3;k++)
temp[i][j]=temp[i][j]+cee[i][k]*b[k][j];
bee(l,b,ji,nrs,i1,rs);
for(i=0;i<3;i++)
for(j=0;j<3;j++)
for(k=0;k<3;k++)
k3[i][j]=k3[i][j]+w*(2./3.)*dj*b[k][i]*temp[k][j];
for(i=0;i<3;i++)
for(j=0;j<3;j++)
k6[3+i][3+j]=k3[i][j];
/*-----*/
if(i1==j1)
{
for(i=0;i<6;i++)
for(j=i;j<6;j++)
*(esm1+nn(i1*6+i,j1*6+j))=*(esm1+nn(i1*6+i,j1*6+j))+k6[i][j];
}
else

```

```

{
for(i=0;i<6;i++)
for(j=0;j<6;j++)
*(esm1+nn(i1*6+i,j1*6+j))=*(esm1+nn(i1*6+i,j1*6+j))+k6[i][j];
}
}
}
/*
*****
st1b_shell(int nrs[4][2])
*****
{
int i,j,k,i1,j1,m1,m2,l;
double w,dj,jacobian(),b[3][3],temp[3][3];
double k3[3][3],k6[6][6],ji[3][3],c1s[2][2],rs[2];
/*
rs[0]=0.0;
rs[1]=0.0;
w=2.0;
dj=jacobian(ji,nrs,rs);
/*
c1s[0][0]=c1s[1][1]=K*ort[mat][0]/(2.*(1.+ort[mat][1]));
c1s[1][0]=c1s[0][1]=0.0;
/*
for(i1=0;i1<4;i1++)
for(j1=i1;j1<4;j1++)
{
/*
for(i=0;i<6;i++)
for(j=0;j<6;j++)
k6[i][j]=0.0;
for(i=0;i<3;i++)
for(j=0;j<3;j++)
{
k3[i][j]=0.0;
temp[i][j]=0.0;
}
}

```

```

/*-----*/
l=3;
bee(l,b,ji,nrs,j1,rs);
for(i=0;i<2;i++)
for(j=0;j<3;j++)
for(k=0;k<2;k++)
temp[i][j]=temp[i][j]+c1s[i][k]*b[k][j];
bee(l,b,ji,nrs,i1,rs);
for(i=0;i<3;i++)
for(j=0;j<3;j++)
for(k=0;k<2;k++)
k3[i][j]=k3[i][j]+w*2*dj*b[k][i]*temp[k][j];
/*-----*/
for(i=0;i<3;i++)
for(j=0;j<3;j++)
{
k6[i][j]=k3[i][j];
k3[i][j]=0.0;
temp[i][j]=0.0;
}
/*-----*/
bee(l,b,ji,nrs,j1,rs);
for(i=0;i<2;i++)
for(j=0;j<3;j++)
for(k=0;k<2;k++)
temp[i][j]=temp[i][j]+c1s[i][k]*b[k][j];
l=4;
bee(l,b,ji,nrs,i1,rs);
for(i=0;i<3;i++)
for(j=0;j<3;j++)
for(k=0;k<2;k++)
k3[i][j]=k3[i][j]+w*2*dj*b[k][i]*temp[k][j];
/*-----*/
for(i=0;i<3;i++)
for(j=0;j<3;j++)
{
k6[3+i][j]=k3[i][j];
k3[i][j]=0.0;
}

```

```

temp[i][j]=0.0;
}
/*-----*/
bee(l,b,ji,nrs,j1,rs);
for(i=0;i<2;i++)
for(j=0;j<3;j++)
for(k=0;k<2;k++)
temp[i][j]=temp[i][j]+c1s[i][k]*b[k][i];
l=3;
bee(l,b,ji,nrs,i1,rs);
for(i=0;i<3;i++)
for(j=0;j<3;j++)
for(k=0;k<2;k++)
k3[i][j]=k3[i][j]+w*2*dj*b[k][i]*temp[k][j];
/*-----*/
for(i=0;i<3;i++)
for(j=0;j<3;j++)
{
k6[i][3+j]=k3[i][j];
k3[i][j]=0.0;
temp[i][j]=0.0;
}
/*-----*/
l=4;
bee(l,b,ji,nrs,j1,rs);
for(i=0;i<2;i++)
for(j=0;j<3;j++)
for(k=0;k<2;k++)
temp[i][j]=temp[i][j]+c1s[i][k]*b[k][j];
bee(l,b,ji,nrs,i1,rs);
for(i=0;i<3;i++)
for(j=0;j<3;j++)
for(k=0;k<2;k++)
k3[i][j]=k3[i][j]+w*2*dj*b[k][i]*temp[k][j];
/*-----*/
for(i=0;i<3;i++)
for(j=0;j<3;j++)
{

```

```

k6[3+i][3+j]=k3[i][j];
k3[i][j]=0.0;
temp[i][j]=0.0;
}
/*-----*/
l=5;
bee(l,b,ji,nrs,j1,rs);
for(i=0;i<2;i++)
for(j=0;j<3;j++)
for(k=0;k<2;k++)
temp[i][j]=temp[i][j]+c1s[i][k]*b[k][j];
bee(l,b,ji,nrs,i1,rs);
for(i=0;i<3;i++)
for(j=0;j<3;j++)
for(k=0;k<2;k++)
k3[i][j]=k3[i][j]+w*(2./3.)*dj*b[k][i]*temp[k][j];
/*-----*/
for(i=0;i<3;i++)
for(j=0;j<3;j++)
k6[3+i][3+j]=k6[3+i][3+j]+k3[i][j];
/*-----*/
if(i1==j1)
{
for(i=0;i<6;i++)
for(j=i;j<6;j++)
*(esm1+nn(i1*6+i,j1*6+j))=*(esm1+nn(i1*6+i,j1*6+j))+k6[i][j];
}
else
{
for(i=0;i<6;i++)
for(j=0;j<6;j++)
*(esm1+nn(i1*6+i,j1*6+j))=*(esm1+nn(i1*6+i,j1*6+j))+k6[i][j];
}
}
/*-----*/
}
/***************/
st1c_shell(int nrs[4][2])

```

```

/*****************/
{
int i,j,k,i1,j1,m1,m2,l;
double w,dj,jacobian(),b[3][3],temp[3][3];
double a,h,k3[3][3],k6[6][6],ji[3][3],rs[2];
/*-----*/
rs[0]=0.0;
rs[1]=0.0;
w=2.0;
dj=jacobian(ji,nrs,rs);
/*-----*/
a=dj*sqrt(ji[0][2]*ji[0][2]+ji[1][2]*ji[1][2]+ji[2][2]*ji[2][2]);
h=0.25*(th[0]+th[1]+th[2]+th[3]);
a=0.5*a*h*ort[mat][0]/(2.* (1.+ort[mat][1]));
/*-----*/
for(i1=0;i1<4;i1++)
for(j1=i1;j1<4;j1++)
{
/*-----*/
for(i=0;i<6;i++)
for(j=0;j<6;j++)
k6[i][j]=0.0;
for(i=0;i<3;i++)
for(j=0;j<3;j++)
{
k3[i][j]=0.0;
temp[i][j]=0.0;
}
/*-----*/
l=6;
bee(l,b,ji,nrs,i1,rs);
for(j=0;j<3;j++)
temp[0][j]=b[0][j];
bee(l,b,ji,nrs,i1,rs);
for(i=0;i<3;i++)
for(j=0;j<3;j++)
k3[i][j]=b[0][i]*temp[0][j];
/*-----*/

```

```

for(i=0;i<3;i++)
for(j=0;j<3;j++)
{
k6[i][j]=k3[i][j];
k3[i][j]=0.0;
temp[i][j]=0.0;
}
/*-----*/
bee(l,b,ji,nrs,j1,rs);
for(j=0;j<3;j++)
temp[0][j]=b[0][j];
l=7;
bee(l,b,ji,nrs,i1,rs);
for(i=0;i<3;i++)
for(j=0;j<3;j++)
k3[i][j]=b[0][i]*temp[0][j];
/*-----*/
for(i=0;i<3;i++)
for(j=0;j<3;j++)
{
k6[i+3][j]=k3[i][j];
k3[i][j]=0.0;
temp[i][j]=0.0;
}
/*-----*/
bee(l,b,ji,nrs,j1,rs);
for(j=0;j<3;j++)
temp[0][j]=b[0][j];
l=6;
bee(l,b,ji,nrs,i1,rs);
for(i=0;i<3;i++)
for(j=0;j<3;j++)
k3[i][j]=b[0][i]*temp[0][j];
/*-----*/
for(i=0;i<3;i++)
for(j=0;j<3;j++)
{
k6[i][j+3]=k3[i][j];

```

```

k3[i][j]=0.0;
temp[i][j]=0.0;
}
/*-----*/
l=7;
bee(l,b,ji,nrs,j1,rs);
for(j=0;j<3;j++)
temp[0][j]=b[0][j];
bee(l,b,ji,nrs,i1,rs);
for(i=0;i<3;i++)
for(j=0;j<3;j++)
k3[i][j]=b[0][i]*temp[0][j];
/*-----*/
for(i=0;i<3;i++)
for(j=0;j<3;j++)
k6[i+3][j+3]=k3[i][j];
/*-----*/
if(i1==j1)
{
for(i=0;i<6;i++)
for(j=i;j<6;j++)
*(esm1+nn(i1*6+i,j1*6+j))=*(esm1+nn(i1*6+i,j1*6+j))+w*a*k6[i][j];
}
else
{
for(i=0;i<6;i++)
for(j=0;j<6;j++)
*(esm1+nn(i1*6+i,j1*6+j))=*(esm1+nn(i1*6+i,j1*6+j))+w*a*k6[i][j];
}
/*-----*/
}
/*-----*/
}
/*****************************************/
bee(int l,double b[3][3],double ji[3][3],int nrs[4][2],int J,double rs[2])
/*****************************************/
{
int i,j,rot1[3]={0,1,2},rot2[3]={1,2,0},rot3[3]={2,0,1};

```

```

double b1i,b2i,b3i,r1,s1,nj,dnrj,dnsj,dnxyz[3],d[3][3],d1[3][3],rs1[2];
/*-----*/
for(i=0;i<3;i++)
for(j=0;j<3;j++)
b[i][j]=0.0;
/*-----*/
dnrij=0.25*nrs[J][0]*(1.+nrs[J][1]*rs[1]);
dnsj=0.25*nrs[J][1]*(1.+nrs[J][0]*rs[0]);
nj=0.25*(1.+nrs[J][0]*rs[0])*(1.+nrs[J][1]*rs[1]);
/*-----*/
d_cosine(d1,nrs,rs);
/*-----*/
dnxyz[0]=ji[0][0]*dnrij+ji[0][1]*dnsj;
dnxyz[1]=ji[1][0]*dnrij+ji[1][1]*dnsj;
dnxyz[2]=ji[2][0]*dnrij+ji[2][1]*dnsj;
/*-----*/
b1i=b2i=b3i=0.0;
/*-----*/
for(i=0;i<3;i++)
{
b1i=b1i+dnxyz[i]*d1[i][0];
b2i=b2i+dnxyz[i]*d1[i][1];
b3i=b3i+dnxyz[i]*d1[i][2];
}
/*-----*/
rs1[0]=1.*nrs[J][0];
rs1[1]=1.*nrs[J][1];
d_cosine(d,nrs,rs1);
/*-----*/
switch(l)
{
/*-----*/
case 1:
/*-----*/
for(j=0;j<3;j++)
{
b[0][j]=d1[rot1[j]][0]*b1i;
b[1][j]=d1[rot1[j]][1]*b2i;
}
}

```

```

b[2][j]=d1[rot1[j]][0]*b2i+d1[rot1[j]][1]*b1i;
}
break;
/*-----*/
case 2:
/*-----*/
for(j=0;j<3;j++)
{
b[0][j]=0.5*th[J]*b1i*(d[rot2[j]][2]*d1[rot3[j]][0]-d[rot3[j]][2]*d1[rot2[j]][0]);

b[1][j]=0.5*th[J]*b2i*(d[rot2[j]][2]*d1[rot3[j]][1]-d[rot3[j]][2]*d1[rot2[j]][1]);

b[2][j]=0.5*th[J]*(b2i*(d[rot2[j]][2]*d1[rot3[j]][0]-d[rot3[j]][2]*d1[rot2[j]][0])
+b1i*(d[rot2[j]][2]*d1[rot3[j]][1]-d[rot3[j]][2]*d1[rot2[j]][1]));
}

break;
/*-----*/
case 3:
/*-----*/
for(j=0;j<3;j++)
{
b[0][j]=d1[rot1[j]][0]*b3i+d1[rot1[j]][2]*b1i;

b[1][j]=d1[rot1[j]][1]*b3i+d1[rot1[j]][2]*b2i;

}
break;
/*-----*/
case 4:
/*-----*/
r1=0.5*th[J]*nj;
s1=d1[0][2]*ji[0][2]+d1[1][2]*ji[1][2]+d1[2][2]*ji[2][2];
for(j=0;j<3;j++)
{
b[0][j]=r1*s1*(d[rot2[j]][2]*d1[rot3[j]][0]-d[rot3[j]][2]*d1[rot2[j]][0]);

b[1][j]=r1*s1*(d[rot2[j]][2]*d1[rot3[j]][1]-d[rot3[j]][2]*d1[rot2[j]][1]);
}

```

```

break;
/*-----*/
case 5:
/*-----*/
for(j=0;j<3;j++)
{
b[0][j]=0.5*th[J]*(b3i*(d[rot2[j]][2]*d1[rot3[j]][0]-d[rot3[j]][2]*d1[rot2[j]][0])
+ b1i*(d[rot2[j]][2]*d1[rot3[j]][2]-d[rot3[j]][2]*d1[rot2[j]][2]));

b[1][j]=0.5*th[J]*(b3i*(d[rot2[j]][2]*d1[rot3[j]][1]-d[rot3[j]][2]*d1[rot2[j]][1])
+ b2i*(d[rot2[j]][2]*d1[rot3[j]][2]-d[rot3[j]][2]*d1[rot2[j]][2]));
}

break;
/*-----*/
case 6:
/*-----*/
for(j=0;j<3;j++)
b[0][j]=0.5*(d1[rot1[j]][0]*b2i-d1[rot1[j]][1]*b1i);
break;
/*-----*/
case 7:
/*-----*/
for(j=0;j<3;j++)
b[0][j]=nj*d1[rot1[j]][2];
break;
}
/*-----*/
}
/*****************************************/
double jacobian(double ji[3][3],int nrs[4][2],double rs[2])
/*****************************************/
{
int i,j,k;
double rs1[2],dnrs[2],ni,magnitude,ja[3][3],d[3][3];
/*-----*/
for(i=0;i<3;i++)

```

```

for(j=0;j<3;j++)
{
ji[i][j]=0.0;
ja[i][j]=0.0;
}
/*-----*/
for(i=0;i<4;i++)
{
dnrs[0]=0.25*nrs[i][0]*(1.+nrs[i][1]*rs[1]);
dnrs[1]=0.25*nrs[i][1]*(1.+nrs[i][0]*rs[0]);
ni=0.25*(1.+nrs[i][0]*rs[0])*(1.+nrs[i][1]*rs[1]);
for(j=0;j<2;j++)
{
rs1[j]=1.*nrs[i][j];
ja[j][0]=ja[j][0]+dnrs[j]*x[i];
ja[j][1]=ja[j][1]+dnrs[j]*y[i];
ja[j][2]=ja[j][2]+dnrs[j]*z[i];
}
d_cosine(d,nrs,rs1);
for(j=0;j<3;j++)
ja[2][j]=ja[2][j]+0.5*ni*th[i]*d[j][2];
}
/*-----*/
ji[0][0]=ja[1][1]*ja[2][2]-ja[2][1]*ja[1][2];
ji[0][1]=ja[2][1]*ja[0][2]-ja[0][1]*ja[2][2];
ji[0][2]=ja[0][1]*ja[1][2]-ja[1][1]*ja[0][2];
ji[1][0]=ja[2][0]*ja[1][2]-ja[1][0]*ja[2][2];
ji[1][1]=ja[0][0]*ja[2][2]-ja[2][0]*ja[0][2];
ji[1][2]=ja[1][0]*ja[0][2]-ja[0][0]*ja[1][2];
ji[2][0]=ja[1][0]*ja[2][1]-ja[2][0]*ja[1][1];
ji[2][1]=ja[2][0]*ja[0][1]-ja[0][0]*ja[2][1];
ji[2][2]=ja[0][0]*ja[1][1]-ja[1][0]*ja[0][1];
/*-----*/
magnitude=ja[0][0]*ji[0][0]+ja[1][0]*ji[0][1]+ja[2][0]*ji[0][2];
/*-----*/
for(i=0;i<3;i++)
for(j=0;j<3;j++)
ji[i][j]=ji[i][j]/magnitude;

```

```

/*-----*/
return(magnitude);
/*-----*/
}
/*****************************************/
d_cosine(double d[3][3],int nrs[4][2],double rs[2])
/*****************************************/
{
int i,j;
double dnrs[4][2],magnitude;
/*-----*/
for(i=0;i<3;i++)
for(j=0;j<3;j++)
d[i][j]=0.0;
for(i=0;i<4;i++)
for(j=0;j<2;j++)
dnrs[i][j]=0.0;
/*-----*/
for(i=0;i<4;i++)
{
dnrs[i][0]=0.25*nrs[i][0]*(1.+nrs[i][1]*rs[1]);
dnrs[i][1]=0.25*nrs[i][1]*(1.+nrs[i][0]*rs[0]);
}
/*-----*/
for(j=0;j<2;j++)
for(i=0;i<4;i++)
{
d[0][j]=d[0][j]+dnrs[i][j]*x[i];
d[1][j]=d[1][j]+dnrs[i][j]*y[i];
d[2][j]=d[2][j]+dnrs[i][j]*z[i];
}
/*-----*/
d[0][2]=d[1][0]*d[2][1]-d[2][0]*d[1][1];
d[1][2]=d[2][0]*d[0][1]-d[0][0]*d[2][1];
d[2][2]=d[0][0]*d[1][1]-d[1][0]*d[0][1];
/*-----*/
magnitude=sqrt(d[0][2]*d[0][2]+d[1][2]*d[1][2]+d[2][2]*d[2][2]);
/*-----*/

```

```

for(i=0;i<3;i++)
{
d[i][2]=d[i][2]/magnitude;
d[i][0]=0.0;
}
/*-----*/
for(i=0;i<4;i++)
{
d[0][0]=d[0][0]+0.25*nrss[i][0]*x[i];
d[1][0]=d[1][0]+0.25*nrss[i][0]*y[i];
d[2][0]=d[2][0]+0.25*nrss[i][0]*z[i];
}
/*-----*/
d[0][1]=d[1][2]*d[2][0]-d[2][2]*d[1][0];
d[1][1]=d[2][2]*d[0][0]-d[0][2]*d[2][0];
d[2][1]=d[0][2]*d[1][0]-d[1][2]*d[0][0];
/*-----*/
magnitude=sqrt(d[0][1]*d[0][1]+d[1][1]*d[1][1]+d[2][1]*d[2][1]);
/*-----*/
for(i=0;i<3;i++)
d[i][1]=d[i][1]/magnitude;
/*-----*/
d[0][0]=d[1][1]*d[2][2]-d[2][1]*d[1][2];
d[1][0]=d[2][1]*d[0][2]-d[0][1]*d[2][2];
d[2][0]=d[0][1]*d[1][2]-d[1][1]*d[0][2];
/*-----*/
}
/**/
c2()
/**/
{
int j,k,xi,xj;
double ct;
/*-----*/
cee[0][0]=1.0;
cee[0][1]=ort[mat][1];
cee[0][2]=0.0;
cee[1][0]=ort[mat][1];
}

```

```

cee[1][1]=1. ;
cee[1][2]=0.0;
cee[2][0]=0.0;
cee[2][1]=0.0;
cee[2][2]=(1.-ort[mat][1])/2. ;
/*-----*/
ct=ort[mat][0]/(1.-ort[mat][1]*ort[mat][1]);
/*-----*/
for(j=0;j<3;j++)
for(k=0;k<3;k++)
cee[j][k]=ct*cee[j][k];
/*-----*/
/*for(j=0;j<3;j++)
{
for(k=0;k<3;k++)
printf("%18.4lf",cee[j][k]);
printf("\n");
}
getch();*/
}
*******/
stiff()
*******/
{
if(tp==1)
stiff1();
}
*******/
stiff1()
*******/
{
int j,k,nrs[4][2]={{-1,-1},{1,-1},{1,1}, {-1,1}};
/*-----*/
for(j=0;j<300;j++)
*(esm1+j)=0;
/*-----*/
c2();
st1a_shell(nrs);

```

```

st1b_shell(nrs);
st1c_shell(nrs);
/*-----*/
}
*****
results()
*****
{
int i,j;
if((dj=(double*)malloc((np*D)*sizeof(double)))==NULL)
{
printf("NO SPACE AVAILKABLE *DJ*");
exit(1);
}
for(j=0;j<(np*D);j++)
*(dj+j)=0;
if((ar=(double*)malloc((np*D)*sizeof(double)))==NULL)
{
printf("NO SPACE AVAILKABLE *AR*");
exit(1);
}
for(j=0;j<(np*D);j++)
*(ar+j)=0;
for(i=0;i<(np*D);i++)
{
fscanf(p6,"%d",&j);
fscanf(p6,"%lf%lf", (dj+j), (ar+j));
}
fprintf(p5, "\n\n\t***RESULTS***\n\n");
for(i=0;i<np;i++)
{
if(i>110 && i<121)
{
fprintf(p5,"NODE NO.=%3d BC= ",i+1);
for(j=0;j<D;j++)
fprintf(p5,"%16d ",*(jrl+i*D+j));
fprintf(p5,"\n");
fprintf(p5,"NODE NO.=%3d AR= ",i+1);
}
}
}

```

```

for(j=0;j<D;j++)
fprintf(p5,"%16.3lf ",*(ar+i*D+j));
fprintf(p5,"\n");
fprintf(p5,"NODE NO.=%3d DJ= ",i+1);
for(j=0;j<D;j++)
fprintf(p5,"%16.4e ",*(dj+i*D+j));
fprintf(p5,"\n\n");
}
}
resulta(dj);
}
*****
graf1()
*****
{
int i,j,j1,j2,j3,n1,dri,mode,x1,y1,x2,y2;
detectgraph(&dri,&mode);
initgraph(&dri,&mode," ");
clrscr();
setcolor(BLACK);
line(270,180,650,180);
line(270,180,270,0);
setcolor(RED);
for(i=0;i<ne;i++)
{
reread();
for(j=0;j<ncn;j++)
{
x1=XX+x[j]*MF;
y1=YY-y[j]*MF;
j1=j+1;
if(j==ncn-1)
j1=0;
x2=XX+x[j1]*MF;
y2=YY-y[j1]*MF;
line(x1,y1,x2,y2);
}
}
}

```

```

setcolor(8);
line(270,180,650,180);
line(270,180,270,0);
rewind(p3);
settextstyle(GOTHIC_FONT,0,3);
outtextxy(180,370,"( CHECK GRID OF SHELL BEING ANALYSED )");
graf1a();
graf1b();
getch();
closegraph();
}
*****
graf2()
*****
{
int i,j,j1,j2,j3,n1,dri,mode,x1,y1,x2,y2;
detectgraph(&dri,&mode);
initgraph(&dri,&mode," ");
clrscr();
setcolor(RED);
for(i=0;i<ne;i++)
{
reread();
for(j=0;j<ncn;j++)
{
x1=XX+(x[j]-y[j])* .866*MF;
y1=YY-((x[j]+y[j])* .500+z[j])*MF;
j1=j+1;
if(j==ncn-1)
j1=0;
x2=XX+(x[j1]-y[j1])* .866*MF;
y2=YY-((x[j1]+y[j1])* .500+z[j1])*MF;
line(x1,y1,x2,y2);
}
}
rewind(p3);
settextstyle(GOTHIC_FONT,0,3);
outtextxy(20,340,"( GRID OF SHELL BEING ANALYSED )");

```

```

graf2a();
graf2b();
getch();
closegraph();
}
*****
graf4()
*****
{
int i,j,j1,j2,j3,n1,dri,mode,x1,y1,x2,y2;
detectgraph(&dri,&mode);
initgraph(&dri,&mode, " ");
clrscr();
setcolor(RED);
for(i=0;i<ne;i++)
{
reread();
for(j=0;j<ncn;j++)
{
x1=XX+(x[j]-y[j]).866*MF;
y1=YY-((x[j]+y[j]).500+z[j])*MF;
j1=j+1;
if(j==ncn-1)
j1=0;
x2=XX+(x[j1]-y[j1]).866*MF;
y2=YY-((x[j1]+y[j1]).500+z[j1])*MF;
line(x1,y1,x2,y2);
}
}
rewind(p3);
settextstyle(GOTHIC_FONT,0,5);
outtextxy(20,340,"(SUPPORT CONDITION FIXED=1, FREE=0)");
graf4a();
getch();
closegraph();
}
*****
graf2a()

```

```

*****
{
long m;
int l1,ll,i,x1,y1;
double x,y,z;
for(i=0;i<np;i++)
{
fseek(p2,M*i,0);
fscanf(p2,"%lf%lf%lf%*lf",&x,&y,&z);
x1=XX+(x-y)*.866*MF;
y1=YY-((x+y)*.500+z)*MF;
setcolor(BLUE);
ll=i+1;
if(ll>99)
{
x1=x1+12;
l1=fmod(ll,10);
graf3(l1,x1,y1);
ll=ll/10;
x1=x1-12;
}
if(ll>9)
{
x1=x1+6;
l1=fmod(ll,10);
graf3(l1,x1,y1);
ll=ll/10;
x1=x1-6;
}
graf3(ll,x1,y1);
}
}
*****
graf2b()
*****
{
int l1,ll,i,j,x1,y1;
for(i=0;i<ne;i++)

```

```

{
x1=0;
y1=0;
reread();
for(j=0;j<ncn;j++)
{
x1=x1+(x[j]-y[j]).866*MF;
y1=y1+((x[j]+y[j]).500+z[j])*MF;
}
setcolor(9);
x1=XX+x1/4;
y1=YY-y1/4;
ll=i+1;
if(ll>99)
{
x1=x1+12;
l1=fmod(ll,10);
graf3(l1,x1,y1);
ll=ll/10;
x1=x1-12;
}
if(ll>9)
{
x1=x1+6;
l1=fmod(ll,10);
graf3(l1,x1,y1);
ll=ll/10;
x1=x1-6;
}
graf3(ll,x1,y1);
}
rewind(p3);
}
*****
graf1a()
*****
{
long m;

```

```

int l1,ll,i,x1,y1;
double x,y,z;
for(i=0;i<np;i++)
{
fseek(p2,M*i,0);
fscanf(p2,"%lf%lf%lf%lf",&x,&y,&z);
x1=XX+x*MF;
y1=YY-y*MF;
setcolor(BLUE);
ll=i+1;
if(ll>99)
{
x1=x1+12;
l1=fmod(ll,10);
graf3(l1,x1,y1);
ll=ll/10;
x1=x1-12;
}
if(ll>9)
{
x1=x1+6;
l1=fmod(ll,10);
graf3(l1,x1,y1);
ll=ll/10;
x1=x1-6;
}
graf3(ll,x1,y1);
}
}
*/
graf1b()
/*
{
int l1,ll,i,j,x1,y1;
for(i=0;i<ne;i++)
{
x1=0;
y1=0;

```

```

reread();
for(j=0;j<ncn;j++)
{
x1=x1+x[j]*MF;
y1=y1+y[j]*MF;
}
x1=XX+x1/4;
y1=YY-y1/4;
setcolor(6);
circle(x1,y1,9);
setcolor(RED);
x1=x1-10;
y1=y1-6;
setcolor(MAGENTA);
ll=i+1;
if(ll>99)
{
x1=x1+12;
l1=fmod(ll,10);
graf3(l1,x1,y1);
ll=ll/10;
x1=x1-12;
}
if(ll>9)
{
x1=x1+6;
l1=fmod(ll,10);
graf3(l1,x1,y1);
ll=ll/10;
x1=x1-6;
}
graf3(ll,x1,y1);
}
rewind(p3);
}
/*****************************************/
graf3(int l,int x1,int y1)
/*****************************************/

```

```

{
switch(l)
{
case 0:
outtextxy(x1+3,y1+3,"0");
break;
case 1:
outtextxy(x1+3,y1+3,"1");
break;
case 2:
outtextxy(x1+3,y1+3,"2");
break;
case 3:
outtextxy(x1+3,y1+3,"3");
break;
case 4:
outtextxy(x1+3,y1+3,"4");
break;
case 5:
outtextxy(x1+3,y1+3,"5");
break;
case 6:
outtextxy(x1+3,y1+3,"6");
break;
case 7:
outtextxy(x1+3,y1+3,"7");
break;
case 8:
outtextxy(x1+3,y1+3,"8");
break;
case 9:
outtextxy(x1+3,y1+3,"9");
break;
}
}

/*****/
graf4a()
/*****

```

```

{
long m;
int n1,l1,i,j,k,x1,y1;
double x,y,z;
for(i=0;i<np;i++)
{
n1=0;
for(j=0;j<D;j++)
n1=n1+*(jrl+i*D+j);
if(n1 != 0)
{
/*-----*/
fseek(p2,M*i,0);
fscanf(p2,"%lf%lf%lf%lf",&x,&y,&z);
x1=XX+(x-y)*.866*MF;
y1=YY-((x+y)*.500+z)*MF;
setcolor(4);
for(k=0;k<3;k++)
circle(x1,y1,k);
setcolor(3);
line(x1,y1-24,x1,y1+24);
y1=y1-27;
for(k=0;k<D;k++)
{
setcolor(BLACK);
l1=*(jrl+i*D+k);
graf3(l1,x1,y1);
y1=y1+8;
getch();
}
/*-----*/
}
}
}
*****/
remove()
*****/
{

```

```

remove("coord.fil");
remove("nop.fil");
remove("resul.fil");
remove("gstif.fil");
}
/*************/
resulta(double*dj)
/*************/
{
int i,j,k,l,m1,m2,m3,i1,l;
int nrs[4][2]={{-1,-1},{1,-1},{1,1}, {-1,1}};
double dj1,jacobian(),b[3][3],ji[3][3],temp[3][3],rs[2],NM[6];
double grs[2]={-1.,1.};
/*-----*/
rewind(p3);
fprintf(p5,"ELEMENT NO. GAUSS PT.    Nx      Ny    ");
fprintf(p5,"  Nxy      Mx      My      Mxy      LINES\n\n\n");
for(l=0;l<ne;l++)
{
reread();
c2();
m3=1;
for(m1=0;m1<2;m1++)
{
rs[0]=grs[m1];
rs[1]=-1.;
dj1=jacobian(ji,nrs,rs);
/*-----*/
for(k=0;k<6;k++)
NM[k]=0.0;
for(i1=0;i1<4;i1++)
{
for(i=0;i<3;i++)
for(j=0;j<3;j++)
temp[i][j]=0.0;
/*-----*/
l=1;
bee(l,b,ji,nrs,i1,rs);
}
}
}

```

```

for(i=0;i<3;i++)
for(j=0;j<3;j++)
for(k=0;k<3;k++)
temp[i][j]=temp[i][j]+cee[i][k]*b[k][j];
for(i=0;i<3;i++)
for(j=0;j<3;j++)
NM[i]=NM[i]+temp[i][j]**(dj+nop[i1]*D+j)*th[i1];
/*-----*/
l=2;
for(i=0;i<3;i++)
for(j=0;j<3;j++)
temp[i][j]=0.0;
bee(l,b,ji,nrs,i1,rs);
for(i=0;i<3;i++)
for(j=0;j<3;j++)
for(k=0;k<3;k++)
temp[i][j]=temp[i][j]+cee[i][k]*b[k][j];
for(i=0;i<3;i++)
for(j=0;j<3;j++)
NM[i+3]=NM[i+3]+temp[i][j]**(dj+nop[i1]*D+j+3)*th[i1]*th[i1]/6.;
/*-----*/
}
if(l>100 && l<111)
{
fprintf(p5,"%4d\t\t%4d\t",l+1,m3);
for(k=0;k<6;k++)
fprintf(p5,"%12.3lf",NM[k]);
fprintf(p5,"\t%6d\n", (l*4+m3));
}
m3=m3+1;
}
}
}

```

REFERENCES

1. Mathematics for Engineers by Chandrika Prasad, (Prasad Mudranalaya, Allahabad).
2. Advance Mathematics for Engineers by Chandrika Prasad (Prasad Mudranalaya, Allahabad).
3. Advanced Engineering Mathematics by Erwin Kreyszig (WILEY EASTERN PRIVATE LIMITED, New Delhi).
4. Differential Equations and the Calculus of Variations by L.ELSGOLTS (MIR PUBLISHERS, MOSCOW).
5. Theory of Plates and Shells by S.Timoshenko and S. Woinowsky Krieger (McGraw-Hill International Book Company).
6. Stresses in Shells by Wilhelm Flugge, (Springer-Verlag OHG., Berlin/Göttingen/Heidelberg 1960, Germany).
7. Fundamentals of the Analysis & Design of Shells Structures by Vasant S. Kelkar and Robert T. Sewell (Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632).
8. Thin Shell Concrete Structures by David P. Billington, (McGraw-Hill Book Company).
9. IASS SYMPOSIUM, Innovative Applications of Shells and Spatial Forms, Vol. I, Published by Mohan Primalani for Oxford & IBH Publishing Co. Pvt. Ltd., New Delhi.
10. IASS SYMPOSIUM, Innovative Applications of Shells and Spatial Forms, Vol.III, Published by Mohan Primalani for Oxford & IBH Publishing Co. Pvt. Ltd., New Delhi.

11. The Finite Element Method, Fourth Edition by O.C. Zienkiewicz and R.L. Taylor, Vol.1 (McGraw–Hill Book Company).
12. The Finite Element Method, Fourth Edition by O.C. Zienkiewicz and R.L. Taylor, Vol. II (McGraw–Hill Book Company).
13. Finite Element Procedures by Klaus–Jurgen Bathe, (Prentice–Hall of India Private Limited).
14. Techniques of Finite Elements by Bruce Irons and Sohrab Ahmed (ELIS HORWOOD LIMITED).
15. Automated Optimum Design of R.C.C. Skeltons by S.K. Parekh. (TATA McGraw–Hill Publishing Company Limited).
16. Finite Element Analysis Theory and Programming by C.S. Krishnamoorthy, (TATA McGraw–Hill Publishing Company Limited).
17. Introduction to the Finite Element Method by Desai and Abel (CBS Publishers & Distributors, Delhi (INDIA)).
18. An introduction to the Finite Element Method by J.N. Reddy (McGraw Hill Book Company).
19. Matrix Analysis of Framed Structures by William Weaver, Jr. and James M.Gere (CBS Publishers & Distributors, Delhi (India))
20. Programming in ANSI C by E. Balagurusamy (TATA McGraw – Hill Publishing Company Limited).
21. Understanding Pointers in C by Yashavant Kanetkar (BPB Publications Delhi).
22. C the Complete Reference by Herbert Schilt (Oshore McGraw – Hill).
23. Frontal Solution Technique of Linear Simultaneous Equations, A Seminar Report submitted to JNU, New Delhi for M Tech Degree by V.P. SINGH, From F. CIVIL CME, PUNE in 1998.