ACCURATE AND FAST ALGORITHMS FOR CONTOUR PLOTTING IN 2D AND 3D DOMAINS FOR FINITE ELEMENT ANALYSIS (FEA) DATA

A thesis submitted in fulfillment of the

requirements for the award of the degree of

DOCTOR OF PHILOSOPHY

in

Mechanical Engineering

Submitted by

JASWINDER SINGH SAINI

(Registration No.: 9030553)

Department of Mechanical Engineering

THAPAR UNIVERSITY

Patiala-147004

Punjab-India

January, 2012

CERTIFICATE

Certified that the thesis entitled, '**ACCURATE AND FAST ALGORITHMS FOR CONTOUR PLOTTING IN 2D AND 3D DOMAINS FOR FINITE ELEMENT ANALYSIS (FEA) DATA'**, being submitted by Jaswinder Singh Saini, to Mechanical Engineering Department, Thapar University, Patiala, in fulfillment of the requirements for the award of the degree of **DOCTOR OF PHILOSOPHY**, is a record of bonafide research work carried out by him under my guidance and supervision. The matter presented in the thesis has not been submitted in part or full to any other University or Institute for the award of any degree.

C Singh
16-1-12

(Dr. Chandan Singh)

Professor, Department of Computer Science,

Punjabi University, Patiala-147002,

Punjab-India.

SUPERVISOR

ACKNOWLEDGEMENTS

This note of acknowledgement is my sincere appreciation to all those who stand out most notably in my mind as contributing directly or indirectly to this research effort.

I am greatly indebted to my revered supervisor, Dr Chandan Singh, Professor, Department of Computer Science, Punjabi University, Patiala for his invaluable guidance, prudent advice and encouragement in accomplishing this research work. His expertise, innovative ways of thinking, constant interest and follow up were the crucial factors in enabling this research work to take its final shape. I extend my heartfelt thanks to my supervisor forever.

I express my regards and gratitude to Dr Ajay Batish, Professor and Head, Mechanical Engineering Department, Thapar University, Patiala for providing requisite facilities, unfailing support, inspiration and ingenuous suggestions.

I am equally beholden to Dr S. K. Mohapatra, Senior Professor and Dean (Academics Affairs), Thapar University, Patiala for his words of encouragement, which I needed most from time to time.

I wish to convey my regards to Dr. Abhijit Mukherjee, Director, Thapar University, Patiala and Dr P. K. Bajpai, Distinguished Professor and Dean (Research and Sponsored Projects), Thapar University, Patiala for their constant help and for providing the necessary institutional facilities.

A sincere token of gratitude is paid to my wife and son for their enormous patience, sustained encouragement and eternal support throughout the research work.

Lastly, on a personal note, I would also like to express my gratitude to all my wonderful students and colleagues of Mechanical Engineering Department, Thapar University, Patiala for their great support, conscientious advice, encouragement, and inspiration throughout this research.

(Jaswinder Singh Saini)

16/1/12

CONTENTS

		Page
ABSTRACT		vi
LIST OF FIGURES		viii
LIST OF TABLES		xxiii
CHAPTER - I INTRODUCTION		**1-26**
1.1	General	1
1.2	Basic Procedure for FEA	3
1.3	Compatibility and Completeness Requirements	4
	1.3.1 Compatibility	4
	1.3.2 Completeness	5
1.4	Polynomial Forms: Geometric Isotropy	5
1.5	Triangular Elements	8
	1.5.1 Area Coordinates	10
	1.5.2 Six-Node Triangular Element	11
	1.5.3 Integration in Area Coordinates	13
1.6	Rectangular Elements	13
1.7	Three-Dimensional Elements	17
	1.7.1 Four-Node Tetrahedral Element	18
	1.7.2 Eight-Node Brick Element	20
1.8	Isoparametric Formulation	21
1.9	Objectives of the Present Work	24
1.10	Organization of Thesis	25
1.11	Concluding Remarks	26
CHAPTER - II LITERATURE REVIEW		**27-54**
2.1	Introduction	27
2.2	Need for Contour Plotting Algorithms	27
2.3	Algorithms for Triangular and Tetrahedral Elements	28
	2.3.1 Surface Approximations using Triangular Element	31
2.4	Algorithms for Rectangular, Quadrilateral and Brick Elements	34
	2.4.1 Marching Cubes (MC) Method	36
2.5	Algorithms using Various Interpolation Techniques	40

2.5.1 Mathematical Interpolation Techniques 41
2.5.2 Shape Functions or Isoparametric Interpolation Techniques 43
2.6 Algorithms using Various Parametric Curves and Surfaces 45
2.7 Miscellaneous 47
2.8 Observations Based on the Review 53
2.9 Concluding Remarks 53

CHAPTER - III CONTOURING IN 2D USING LINEAR ELEMENTS 55-93

3.1 Introduction 55
3.2 Contour Equation over Triangular Element 55
3.3 Contouring Algorithm 57
3.4 Contour Equation over Quadrilateral Element 58
3.4.1 Tracing of Contour over Quadrilateral Element 60
3.5 Plotting of Contours in 2D 66
3.5.1 Determining the Window 66
3.5.2 Determining the Viewport 66
3.6 Implementation of the Algorithm for Triangular Element 67
3.7 Implementation of the Algorithm for Quadrilateral Element 82
3.7.1 Speed of the Algorithm 92
3.8 Concluding Remarks 92

CHAPTER - IV CONTOURING IN 2D USING NON-LINEAR ELEMENTS 94-149

4.1 Introduction 94
4.2 Contour Equation over Six-Node Triangle 95
4.2.1 Finding the Minimum and Maximum of *f(x, y)* over the Element 97
4.2.2 Tracing Contour Segment over an Element 98
4.2.3 Joining the Contour Segments 107
4.3 Contour Equation over Ten-Node Triangle 112
4.4 Contour Equation over Eight-Node Quadrilateral Element 117
4.4.1 Finding the Minimum and Maximum of *f(x, y)* over the Element 118
4.4.2 Tracing Contour Segment over an Element 119
4.5 Implementation of the Algorithm for Six-Node Triangular Element 127
4.6 Implementation of the Algorithm for Eight-Node Quadrilateral Element 142

4.6.1 Speed of the Algorithm 148
4.7 Concluding Remarks 149

CHAPTER – V CONTOURING IN 3D USING LINEAR ELEMENTS 150-204
5.1 Introduction 150
5.2 Contour Equation over Tetrahedral Element 150
5.2.1 Tracing Contour Surface over a Tetrahedral Element 152
5.3 Contour Surfaces in 3D Using Hexahedral Elements 153
5.3.1 Direct Method For Tracing Contour Surfaces 155
5.3.2 Contouring By Linear Interpolation 156
5.3.3 Accurate and Fast Generation of Contour Surface 159
5.4 Plotting Contours in 3D 170
5.4.1 Deriving Matrix Equation For Orthographic Parallel Projection 171
5.5 Implementation of the Algorithms for Tetrahedral Element 175
5.6 Implementation of the Algorithms for Hexahedral Element 191
5.7 Concluding Remarks 203

CHAPTER - VI CONTOURING IN 3D USING NON-LINEAR ELEMENTS 205-232
6.1 Introduction 205
6.2 Contour Equation over Ten-Node Tetrahedral Element 205
6.2.1 Finding the Minimum and Maximum of $f(x, y,z)$ over the Element 211
6.2.2 Tracing Contour Surface over an Element 212
6.3 Implementation of the Algorithm and Numerical Results 220
6.4 Concluding Remarks 232

CHAPTER - VII CONCLUSIONS AND FUTURE SCOPE 233-234
7.1 General 233
7.2 Conclusions 233
7.3 Scope for Future Work 234

LIST OF RESEARCH PUBLICATIONS 235-235
REFERENCES 236-242

ABSTRACT

Contour plotting is one of the basic operations performed in many engineering analyses where visual inspection of results is to be carried out. Such analyses produce a large amount of data which is cumbersome to interpret in its numerical form. The graphical representation of the numerical data in the form of contour lines in 2D and contour surfaces in 3D makes the analyses more informative and faster.

Finite Element Analysis (FEA) is one of such analyses which produces voluminous data and whose graphical representation becomes a necessity. Contour plotting is one method which is used for graphical representation. It substitutes a large amount of numerical data into graphical patterns, which helps to perceive the physical consequences of a calculation. To users, not familiar with the details of FEA, it is possibly one of the best ways to perceive the analysis results. So, once the 'solver' determines the element resultants, a post-processor in the form of contour plots is used to graphically display the domain response to the applied loads and boundary conditions.

For a user, the accuracy of the analysis results depends on the accuracy of the graphical patterns which are displayed as contour lines or surfaces on the screen. Generally, higher order elements are used for accurate analysis, but are degenerated into linear elements for contour plotting to avoid complexity. This leads to loss of information in the graphical plotting. In some cases, to visualize the solutions effectively, approximations on finer meshes are required.

In the present work, an attempt is made to develop accurate and fast algorithms for different meshes used for analysis in FEA. All the algorithms depend upon the contour equations developed using 'Shape Functions' for these meshes.

The simplest way to plot contours on 2D domain is to use linear interpolation over triangular elements. This method works well only if the accuracy of higher order is not required. Higher order triangular elements are generally degenerated into linear triangular elements for fast contour plotting but on expense of accuracy. Higher accuracy may be desirable in regions where the variation in physical quantity is large or in applications such as imaging. In the present work, an attempt is made to use higher order interpolation using a six-node triangular element. The algorithm traces contours as accurately as the analysis results obtained in FEA. The quadratic interpolation function

representing contours is degenerated into various conic sections and the special characteristics of these conic sections are exploited to accurately interpolate the function. After contour segments are traced, they are joined together to form contour lines for fast display on graphical display devices.

Quadrilateral element is the second type of 2D element taken for contour plotting. The contour equation is developed over linear and quadratic quadrilateral elements. An attempt is made to enhance the speed of contour generation without compensating with accuracy. In four-node quadrilateral element, the developed contour equation represents a rectangular hyperbola. The contours are joined based on location of asymptotes of this rectangular hyperbola, thus making the algorithm fast. Further, the accuracy of contour is improved by using elementary calculus. Similarly, in eight-node quadrilateral element, an attempt is made to develop an algorithm to join the contours accurately and quickly.

In 3D, the problems are mainly analyzed using tetrahedral and hexahedral elements. As discussed in case of 2D, to simplify the 3D contour surface generation, a higher order element is splited into linear elements and linear interpolation of the physical quantity is performed, thus resulting in the linear patches of contour surfaces. Even a hexahedral element is decomposed into 6 four-node tetrahedral elements for contour surface generation, thus leading to the linear interpolation of the physical quantity. In the present work, an attempt is made to develop accurate and fast contour surface plotting algorithm by first deriving the boundary curves and then locating its interior points. The method provides the contour surface as a group of bounding curves over an element. Towards the end, the contour surfaces over a quadratic tetrahedral element are generated. The contour equation is developed using shape functions which represents the various quadrics.

The detailed numerical experimentations after implementing the different algorithms are given in respective chapters.

LIST OF FIGURES

	Title of Figure	Page
Figure 1.1	Pascal triangle for polynomials in two dimensions	6
Figure 1.2	Pascal pyramid for polynomials in three dimensions	7
Figure 1.3	Triangular elements: three-node linear, six-node quadratic, ten-node cubic	8
Figure 1.4	A general three-node triangular element referred to global coordinates	8
Figure 1.5	Areas used to define area coordinates for a triangular element	10
Figure 1.6(a)	Area A_1 associated with either P or P' is constant	10
Figure 1.6(b)	Lines of the constant area coordinate L_1	10
Figure 1.7(a)	Six-node triangular elements: Node numbering	12
Figure 1.7(b)	Six-node triangular elements: Lines of constant values of the are coordinates	12
Figure 1.8	A four-node rectangular element defined in global coordinates	14
Figure 1.9(a)	A four-node rectangular element showing the translation to natural coordinates	15
Figure 1.9(b)	A four-node rectangular element showing the natural coordinates of each node	15
Figure 1.10	Eight-node rectangular element showing both global and natural coordinate axes	16
Figure 1.11	A four-node tetrahedral element	18
Figure 1.12	A four-node tetrahedral element, showing an arbitrary point defining four volumes	18
Figure 1.13	Higher-order tetrahedral elements: ten-node and twenty-node	19
Figure 1.14(a)	Eight-node brick element: Global Cartesian coordinates	20
Figure 1.14(b)	Eight-node brick element: Natural coordinates with an origin at the centroid	20
Figure 1.15(a)	A domain to be modeled	22
Figure 1.15(b)	A domain modeled using Triangular elements	22
Figure 1.15(c)	A domain modeled using Rectangular elements	22

Figure 1.15(d)	A domain modeled using Rectangular and quadrilateral elements	22
Figure 1.16	Mapping of a parent element into an isoparametric element	22
Figure 1.17(a)	Isoparametric mapping of quadratic elements into curved elements: Six-node triangle	24
Figure 1.17(b)	Isoparametric mapping of quadratic elements into curved elements: Eight-node rectangle	24
Figure 3.1	A three-node triangular element	55
Figure 3.2(a)	Possible cases of intersection of a side with nodes 1 and 2 with the line: The line intersects at one point of the side	58
Figure 3.2(b)	Possible cases of intersection of a side with nodes 1 and 2 with the line: Intersection point is outside the side	58
Figure 3.2(c)	Possible cases of intersection of a side with nodes 1 and 2 with the line: The line touches the corner point	58
Figure 3.3	Four-node element in global and local co-ordinate systems	60
Figure 3.4	Contour as a straight line over an element	61
Figure 3.5(a)	Contour branches parallel to r-axis and x-axis: Centre of hyperbola inside the element	62
Figure 3.5(b)	Contour branches parallel to r-axis and x-axis: Centre of hyperbola over an edge	62
Figure 3.5(c)	Contour branches parallel to r-axis and x-axis: Centre of hyperbola over a node	63
Figure 3.5(d)	Contour branches parallel to r-axis and x-axis: Centre of hyperbola outside the edge	63
Figure 3.6(a)	Various cases of contour branches intersecting element edges: Two branches intersecting edges	64
Figure 3.6(b)	Various cases of contour branches intersecting element edges: One branch intersecting edges	64
Figure 3.6(c)	Various cases of contour branches intersecting element edges: Branch intersecting any two of four edges	64
Figure 3.6(d)	Various cases of contour branches intersecting element edges: Branch will not intersect edge, $s = +1$	64
Figure 3.7	The maximum error in approximation over an element	65

Figure 3.8	Specification of the beam	67
Figure 3.9(a)	Beam fixed at one end and axially loaded at other end	68
Figure 3.9(b)	Beam fixed at one end and vertically loaded at other end	68
Figure 3.10	Section 1-1, 2-2 and 3-3 considered in the beam	69
Figure 3.11(a)	Contour plot of horizontal deflection using three-node triangles for axially loaded beam (units in mm)	69
Figure 3.11(b)	Contour plot of vertical deflection using three-node triangles for axially loaded beam (units in mm)	70
Figure 3.12(a)	Contour plot of vertical deflection using three-node triangles for vertical loaded beam (units in mm)	71
Figure 3.12(b)	Contour plot of horizontal deflection using three-node triangles for vertically loaded beam (units in mm)	71
Figure 3.13(a)	Specification of the component (all dimensions are in mm)	72
Figure 3.13(b)	Loading and boundary conditions of the component	72
Figure 3.14(a)	Contour plot of vertical deflection using three-node triangles for the component	72
Figure 3.14(b)	Contour plot of horizontal deflection using three-node triangles for the component	73
Figure 3.15(a)	Specifications of the rocker arm (all dimensions in mm)	73
Figure 3.15(b)	Specifications of the rocker arm (all dimensions in mm)	74
Figure 3.16	Contour plot of vertical deflection using three-node triangles for rocker arm	74
Figure 3.17(a)	Specifications of a spanner (all dimensions in mm)	74
Figure 3.17(b)	Boundary conditions and loading of the spanner	75
Figure 3.18	Contour plot of horizontal deflection using three-node triangles for the spanner	75
Figure 3.19(a)	Specifications of connecting rod (all dimensions in mm)	75
Figure 3.19(b)	Boundary conditions and loading of the connecting rod	76
Figure 3.20	Contour plot of vertical deflection using three-node triangles for connecting rod	76
Figure 3.21(a)	Specifications of a plate (all dimensions in mm)	77
Figure 3.21(b)	Boundary conditions and loading of the plate	77
Figure 3.22(a)	Contour plot of vertical deflection using three-node triangles	78

	for the plate	
Figure 3.22(b)	Contour plot of horizontal deflection using three-node triangles for the plate	78
Figure 3.23(a)	Specifications of a plate (all dimensions in mm)	79
Figure 3.23(b)	Boundary conditions and loading of the plate	79
Figure 3.24(a)	Contour plot of vertical deflection using three-node triangles for the plate	80
Figure 3.24(b)	Contour plot of horizontal deflection using three-node triangles for the plate	80
Figure 3.25(a)	Specifications of a reactor (all dimensions in mm)	81
Figure 3.25(b)	Boundary conditions and temperature fixation of the plate	81
Figure 3.26	Contour plot of temperature distribution (in ^{o}C) using three-node triangles for the reactor	82
Figure 3.27	Specifications of the rocker arm (all dimensions in mm)	83
Figure 3.28	Contour plots of horizontal deflection for rocker arm	83
Figure 3.29(a)	Specifications of a plate (all dimensions in mm)	84
Figure 3.29(b)	Boundary conditions and loading of the plate	84
Figure 3.30(a)	Contour plots of vertical deflection for the above specified plate	84
Figure 3.30(b)	Contour plots of horizontal deflection for the above specified plate	85
Figure 3.31(a)	Specifications of a plate (all dimensions in mm)	85
Figure 3.31(b)	Boundary conditions and loading of the plate	86
Figure 3.32(a)	Contour plots of vertical deflection for the above specified plate	86
Figure 3.32(b)	Contour plots of horizontal deflection for the above specified plate	87
Figure 3.33(a)	Specifications of a plate (all dimensions in mm)	87
Figure 3.33(b)	Boundary conditions and loading of the plate	88
Figure 3.34(a)	Contour plots of vertical deflection for the above specified plate	88
Figure 3.34(b)	Contour plots of horizontal deflection for the above specified plate	89
Figure 3.35(a)	Specifications of a safety valve lever (all dimensions in mm)	89
Figure 3.35(b)	Boundary conditions and loading of the safety valve lever	90
Figure 3.36(a)	Contour plots of vertical deflection for the safety valve lever	90

Figure 3.36(b)	Contour plots of horizontal deflection for the safety valve lever	90
Figure 3.37(a)	Specifications of a boiler shell (all dimensions in mm)	91
Figure 3.37(b)	Boundary conditions and temperature fixation of the boiler shell	91
Figure 3.38	Contour plots of temperature distribution (in ^{o}C) for the boiler shell	92
Figure 4.1	A six-node triangular element	95
Figure 4.2(a)	The circle intersects at two points of the sides	99
Figure 4.2(b)	One of the two intersection points is outside the side	99
Figure 4.2(c)	Both the intersection points are outside	99
Figure 4.2(d)	The side does not intersect but the center of the circle is to the left of the side	99
Figure 4.2(e)	The side does not intersect but the center of the circle is to the right of the side	99
Figure 4.2(f)	The circle touches the circle and the center is to the right of the side	99
Figure 4.3(a)	Possible cases of intersection of a side with nodes 1 and 2 with a parabola: repetitive roots and roots lying within the side	100
Figure 4.3(b)	Possible cases of intersection of a side with nodes 1 and 2 with a parabola: nonrepetitive roots, both lying within the side	100
Figure 4.3(c)	Possible cases of intersection of a side with nodes 1 and 2 with a parabola: one root within the side	100
Figure 4.3(d)	Possible cases of intersection of a side with nodes 1 and 2 with a parabola: both roots outside the side	100
Figure 4.3(e)	Possible cases of intersection of a side with nodes 1 and 2 with a parabola: the side does not intersect the parabola (complex roots)	100
Figure 4.4(a)	Possible cases of intersection of a side with nodes 1 and 2 with a hyperbola: real roots when two roots lie within the side	101
Figure 4.4(b)	Possible cases of intersection of a side with nodes 1 and 2 with a hyperbola: real roots when one of the roots lie within the side	101
Figure 4.4(c)	Possible cases of intersection of a side with nodes 1 and 2 with a hyperbola: when no root lies within the side	101

Figure 4.4(d)	Possible cases of intersection of a side with nodes 1 and 2 with a hyperbola: no real root exists	101
Figure 4.5(a)	Possible cases of intersection of a side with nodes 1 and 2 with a pair of straight lines: both contour lines intersect the inner part of the side	102
Figure 4.5(b)	Possible cases of intersection of a side with nodes 1 and 2 with a pair of straight lines: only one line intersects the inner part	102
Figure 4.5(c)	Possible cases of intersection of a side with nodes 1 and 2 with a pair of straight lines: both the intersections lie on the extended part of the side	102
Figure 4.5(d)	Possible cases of intersection of a side with nodes 1 and 2 with a pair of straight lines: one of the contour lines is parallel to the edge	102
Figure 4.5(e)	Possible cases of intersection of a side with nodes 1 and 2 with a pair of straight lines: both contour lines are parallel to the edge	102
Figure 4.5(f)	Possible cases of intersection of a side with nodes 1 and 2 with a pair of straight lines: one of the contour lines overlaps the edge	102
Figure 4.5(g)	Possible cases of intersection of a side with nodes 1 and 2 with a pair of straight lines: both contour lines overlap the edge	102
Figure 4.6(a)	Possible cases of circular arcs intersecting the edges of a triangle: two intersections with an edge	104
Figure 4.6(b)	Possible cases of circular arcs intersecting the edges of a triangle: two intersections with two edges	104
Figure 4.6(c)	Possible cases of circular arcs intersecting the edges of a triangle: four intersections with two edges	104
Figure 4.6(d)	Possible cases of circular arcs intersecting the edges of a triangle: four intersections with two edges	104
Figure 4.6(e)	Possible cases of circular arcs intersecting the edges of a triangle: six intersections with three edges	104
Figure 4.7(a)	The four cases of parabolic contour segment(s) lying over a triangle: two intersections lying on an edge	105

Figure 4.7(b)	The four cases of parabolic contour segment(s) lying over a triangle: two intersections lying on two edges	105
Figure 4.7(c)	The four cases of parabolic contour segment(s) lying over a triangle: four intersections lying on two edges	105
Figure 4.7(d)	The four cases of parabolic contour segment(s) lying over a triangle: four intersections lying on three edges	105
Figure 4.8(a)	The four cases of hyperbola contour segment(s) lying over a triangle: two intersections lie on an edge	106
Figure 4.8(b)	The four cases of hyperbola contour segment(s) lying over a triangle: two intersections lie on two edges	106
Figure 4.8(c)	The four cases of hyperbola contour segment(s) lying over a triangle: four intersections lie on three edges but the numbering on intersections differ	106
Figure 4.8(d)	The four cases of hyperbola contour segment(s) lying over a triangle: four intersections lie on three edges but the numbering on intersections differ	106
Figure 4.9(a)	The three possible cases of a pair of line contour segments intersecting the triangle: two intersections on two edges	107
Figure 4.9(b)	The three possible cases of a pair of line contour segments intersecting the triangle: four intersections on two edges	107
Figure 4.9(c)	The three possible cases of a pair of line contour segments intersecting the triangle: four intersections on three edges	107
Figure 4.10	Representation of a contour segment over an element	108
Figure 4.11(a)	The four possible cases of joining of a contour line with a contour segment: *HEAD* joins with *HEAD*	110
Figure 4.11(b)	The four possible cases of joining of a contour line with a contour segment: *HEAD* joins with *FRONT*	110
Figure 4.11(c)	The four possible cases of joining of a contour line with a contour segment: *FRONT* joins with *HEAD*	110
Figure 4.11(d)	The four possible cases of joining of a contour line with a contour segment: *FRONT* joins with *FRONT*	110
Figure 4.12(a)	A ten-node triangular element	112
Figure 4.12(b)	Eight-node element in local co-ordinate systems	117

Figure 4.13	Possible case of a Loop	120
Figure 4.14	Different possible cases for two intersections	121
Figure 4.15	Different cases of (1,1,1,1) possible intersections	122
Figure 4.16	Different cases of (1,1,2,0) possible intersections	123
Figure 4.17	Different cases of (2,2,0,0) possible intersections: Distance between the roots approaching zero	124
Figure 4.18	Different cases of (1,1,2,2) possible intersections	126
Figure 4.19	Different cases of (2,2,0,0) possible intersections	126
Figure 4.20	Case of (2,2,2,2) possible intersections	127
Figure 4.21	Griding of a square domain using six-node triangles	128
Figure 4.22(a)	Contours over the square domains: circle	129
Figure 4.22(b)	Contours over the square domains: ellipse	129
Figure 4.22(c)	Contours over the square domains: parabola	129
Figure 4.22(d)	Contours over the square domains: rectangular hyperbola	129
Figure 4.22(e)	Contours over the square domains: hyperbola	129
Figure 4.22(f)	Contours over the square domains: pair of parallel straight lines	129
Figure 4.22(g)	Contours over the square domains: pair of perpendicular straight lines	129
Figure 4.22(h)	Contours over the square domains: pair of straight lines	129
Figure 4.23(a)	Contour plot of horizontal deflection using six-node triangles for axially loaded beam (units in mm)	130
Figure 4.23(b)	Contour plot of vertical deflection using six-node triangles for axially loaded beam (units in mm)	130
Figure 4.24(a)	Contour plot of vertical deflection using six-node triangles for vertically loaded beam (units in mm)	131
Figure 4.24(b)	Contour plot of horizontal deflection using six-node triangles for vertically loaded beam (units in mm)	131
Figure 4.25(a)	Contour plot of vertical deflection using six-node triangles for the component	132
Figure 4.25(b)	Contour plot of horizontal deflection using six-node triangles for the component	132
Figure 4.26(a)	Contour plot of vertical deflection using six-node triangles for the rocker arm	133

Figure 4.26(b)	Contour plot of horizontal deflection using six-node triangles for the rocker arm	133
Figure 4.27(a)	Contour plot of vertical deflection using six-node triangles for the spanner	134
Figure 4.27(b)	Contour plot of horizontal deflection using six-node triangles for the spanner	134
Figure 4.28(a)	Specifications of the lever (all dimensions in mm)	135
Figure 4.28(b)	Boundary conditions and loading of the lever	135
Figure 4.29(a)	Contour plot of vertical deflection using six-node triangles for the lever	136
Figure 4.29(b)	Contour plot of horizontal deflection using six-node triangles for the lever	136
Figure 4.30(a)	Specifications of the plate (all dimensions in mm)	137
Figure 4.30(b)	Boundary conditions and loading of the plate	137
Figure 4.31(a)	Contour plot of vertical deflection using six-node triangles for the plate	138
Figure 4.31(b)	Contour plot of horizontal deflection using six-node triangles for the plate	138
Figure 4.32(a)	Specifications of the plate (all dimensions in mm)	139
Figure 4.32(b)	Boundary conditions and loading of the plate	139
Figure 4.33(a)	Contour plot of vertical deflection using six-node triangles for the plate	140
Figure 4.33(b)	Contour plot of horizontal deflection using six-node triangles for the plate	140
Figure 4.34(a)	Specifications of composite plates (all dimensions in mm)	141
Figure 4.34(b)	Boundary conditions and temperature fixation of the plates	141
Figure 4.35	Contour plot of temperature distribution (in ^{o}C) using six-node triangles for the plates	142
Figure 4.36	Contour plots of vertical deflection for rocker arm	142
Figure 4.37	Contour plots of horizontal deflection for connecting rod	143
Figure 4.38(a)	Specifications of the proving ring (all dimensions in mm)	144
Figure 4.38(b)	Boundary and loading conditions	144
Figure 4.39	Contour plots of vertical deflection for proving ring	145

Figure 4.40(a)	Specifications of the chain plate (all dimensions in mm)	145
Figure 4.40(b)	Boundary conditions and loading of the chain plate	146
Figure 4.41(a)	Contour plots of vertical deflection for chain plate	146
Figure 4.41(b)	Contour plots of horizontal deflection for chain plate	146
Figure 4.42(a)	Contour plots of vertical deflection for the plate	147
Figure 4.42(b)	Contour plots of horizontal deflection for the plate	147
Figure 4.43(a)	Contour plots of vertical deflection for the plate	148
Figure 4.43(b)	Contour plots of horizontal deflection for the plate	148
Figure 5.1	A four-node tetrahedral element	151
Figure 5.2(a)	A hexahedral element in: Global coordinates	153
Figure 5.2(b)	A hexahedral element in: Local coordinates	154
Figure 5.3	Pseudocode for direct method of tracing contour surface	156
Figure 5.4	A hexahedral element with its local node numbering and its decomposition into 6 tetrahedral elements	156
Figure 5.5(a)	A tetrahedral element and its local node numbering	158
Figure 5.5(b)	A face of the tetrahedral	158
Figure 5.5(c)	A contour plane with four boundaries	158
Figure 5.6	Pseudocode for generating a planner contour surface within a tetrahedral element	158
Figure 5.7	A hexahedral element (cube with length 2 units) in local coordinates and a contour surface for the contour level $C = 25.0$	160
Figure 5.8(a)	Two disjoint contour surfaces	160
Figure 5.8(b)	Three disjoint contour surfaces	160
Figure 5.8(c)	Four disjoint contour surfaces	161
Figure 5.9	Contour segments as branches of a rectangular hyperbola on a quadrilateral element in local coordinates for various values of K	163
Figure 5.10	A contour segment and its maximum shift from the line AB	164
Figure 5.11	Pseudocode for finding the closed polyline of a contour surface	168
Figure 5.12	Generation of interior points on contour surface	170
Figure 5.13(a)	Specifying viewplane in user coordinate system	171
Figure 5.13(b)	The View plane coordinate system indicating orthogonal unit	173

	vectors $\vec{I}_v$, $\vec{J}_v$ and $\vec{K}_v$	
Figure 5.14(a)	Specifications of a beam (all dimensions in mm)	176
Figure 5.14(b)	Boundary conditions and loading of the plate	176
Figure 5.14(c)	Contour surfaces of horizontal deflection using four-node tetrahedrals for the different deflection levels for the plate	177
Figure 5.15(a)	Specifications of a crane hook	178
Figure 5.15(b)	Boundary conditions and loading of the crane hook	178
Figure 5.16(a)	Contour surfaces of deflection in x-direction using four-node tetrahedrals for the different deflection levels	179
Figure 5.16(b)	Contour surfaces of deflection in y-direction using four-node tetrahedrals for the different deflection levels	179
Figure 5.17(a)	Specifications of the bearing bracket	180
Figure 5.17(b)	Boundary conditions and loading of the bearing bracket	180
Figure 5.18(a)	Contour surfaces of deflection in x-direction using four-node tetrahedrals for the different deflection levels for bearing bracket	181
Figure 5.18(b)	Contour surfaces of deflection in y-direction using four-node tetrahedrals for the different deflection levels for bearing bracket	181
Figure 5.18(c)	Contour surfaces of deflection in z-direction using four-node tetrahedrals for the different deflection levels for bearing bracket	182
Figure 5.19(a)	Specifications of the fork	182
Figure 5.19(b)	Boundary conditions and loading of the fork	183
Figure 5.20(a)	Contour surfaces of deflection in x-direction using four-node tetrahedrals for the different deflection levels	183
Figure 5.20(b)	Contour surfaces of deflection in y-direction using four-node tetrahedrals for the different deflection levels	184
Figure 5.20(c)	Contour surfaces of deflection in z-direction using four-node tetrahedrals for the different deflection levels	184
Figure 5.21(a)	Specifications of the fork	185
Figure 5.21(b)	Boundary conditions and loading of the fork	185
Figure 5.22(a)	Contour surfaces of deflection in x-direction using four-node	186

	tetrahedrals for the different deflection levels for the fork	
Figure 5.22(b)	Contour surfaces of deflection in y-direction using four-node tetrahedrals for the different deflection levels for the fork	186
Figure 5.22(c)	Contour surfaces of deflection in z-direction using four-node tetrahedrals for the different deflection levels for the fork	186
Figure 5.23(a)	Specifications of the bracket	187
Figure 5.23(b)	Boundary conditions and loading of the bracket	187
Figure 5.24(a)	Contour surfaces of deflection in x-direction using four-node tetrahedrals for the different deflection levels for the bracket	188
Figure 5.24(b)	Contour surfaces of deflection in y-direction using four-node tetrahedrals for the different deflection levels for the bracket	188
Figure 5.24(c)	Contour surfaces of deflection in z-direction using four-node tetrahedrals for the different deflection levels for the bracket	189
Figure 5.25(a)	Specification of the reactor	189
Figure 5.25(b)	Boundary conditions and temperature fixation of the reactor	190
Figure 5.26	Contour plot of temperature distribution (in ^{o}C) using four-node tetrahedral elements for the reactor	190
Figure 5.27(a)	Specifications of the beam	191
Figure 5.27(b)	Boundary conditions and loading of the beam	191
Figure 5.28(a)	Contour surfaces of deflection in y-direction using eight-node hexahedral elements by dividing the hexahedral element into different tetrahedral elements	192
Figure 5.28(b)	Contour surfaces of deflection in y-direction using eight-node hexahedral elements by using the discussed algorithm.	192
Figure 5.29	Specifications and conditions of the crane hook	192
Figure 5.30	Contour surfaces of deflection in vertical direction using eight-node hexahedral elements for the different deflection levels for the hook	193
Figure 5.31(a)	Specifications of the plate	194
Figure 5.31(b)	Boundary conditions and loading of the plate	194
Figure 5.32(a)	Contour surfaces of deflection in x-direction using eight-node hexahedral elements for the different deflection levels	195
Figure 5.32(b)	Contour surfaces of deflection in y-direction using eight-node	195

	hexahedral elements for the different deflection levels	
Figure 5.32(c)	Contour surfaces of deflection in z-direction using eight-node hexahedral elements for the different deflection levels	196
Figure 5.33(a)	Specifications of the ring	196
Figure 5.33(b)	Boundary conditions and loading of the ring	197
Figure 5.34(a)	Contour surfaces of deflection in x-direction using eight-node hexahedral elements for the different deflection levels for the ring	197
Figure 5.34(b)	Contour surfaces of deflection in y-direction using eight-node hexahedral elements for the different deflection levels	197
Figure 5.34(c)	Contour surfaces of deflection in z-direction using eight-node hexahedral elements for the different deflection levels	198
Figure 5.35(a)	Specifications of the U-section beam	198
Figure 5.35(b)	Boundary conditions and loading of the U-section beam	199
Figure 5.36(a)	Contour surfaces of deflection in x-direction using eight-node hexahedral elements for the different deflection levels	199
Figure 5.36(b)	Contour surfaces of deflection in y-direction using eight-node hexahedral elements for the different deflection levels	200
Figure 5.36(c)	Contour surfaces of deflection in z-direction using eight-node hexahedral elements for the different deflection levels	200
Figure 5.37(a)	Specifications of the lever	201
Figure 5.37(b)	Boundary conditions and loading of the lever	201
Figure 5.38(a)	Contour surfaces of deflection in x-direction using eight-node hexahedral elements for the different deflection levels for the lever	202
Figure 5.38(b)	Contour surfaces of deflection in y-direction using eight-node hexahedral elements for the different deflection levels for the lever	202
Figure 5.38(c)	Contour surfaces of deflection in z-direction using eight-node hexahedral elements for the different deflection levels for the lever	203
Figure 6.1	A ten-node tetrahedral element	205
Figure 6.2	Intersection of a Plane and a Sphere	213

Figure 6.3	Cross-sectional View of Sphere-Plane Intersection	214
Figure 6.4	Some of the Ways a Plane and a Cylinder can Intersect	215
Figure 6.5	Edge-on View of Plane-Cylinder Intersection	216
Figure 6.6	Some of the Ways a Plane and a Cone can Intersect	218
Figure 6.7	Edge-on View of Plane-Cone Intersection	220
Figure 6.8(a)	Contours over the cube domains: Cylinder	221
Figure 6.8(b)	Contours over the cube domains: Hyperbolic Paraboloid	221
Figure 6.8(c)	Contours over the cube domains: Ellipsoid	221
Figure 6.8(d)	Contours over the cube domains: Hyperboloid of Two Sheets	221
Figure 6.8(e)	Contours over the cube domains: Hyperboloid of One Sheets	222
Figure 6.9	Specifications of the hand wheel	222
Figure 6.10	Boundary conditions and loading of the hand wheel	223
Figure 6.11(a)	Contour surfaces of deflection in x-direction using ten-node tetrahedral elements for the different deflection levels for the hand wheel	223
Figure 6.11(b)	Contour surfaces of deflection in y-direction using ten-node tetrahedral elements for the different deflection levels for the hand wheel	223
Figure 6.11(c)	Contour surfaces of deflection in z-direction using ten-node tetrahedral elements for the different deflection levels for the hand wheel	224
Figure 6.12(a)	Specifications of the component	224
Figure 6.12(b)	Boundary conditions and loading of the component	225
Figure 6.13(a)	Contour surfaces of deflection in x-direction using ten-node tetrahedral elements for the different deflection levels for the component	225
Figure 6.13(b)	Contour surfaces of deflection in x-direction using ten-node tetrahedral elements for the different deflection levels for the component	225
Figure 6.14(a)	Specifications of the bearing holder	226
Figure 6.14(b)	Boundary conditions and loading of the bearing holder	226
Figure 6.15(a)	Contour surfaces of deflection in x-direction using ten-node tetrahedral elements for the different deflection levels for the	227

	bearing holder	
Figure 6.15(b)	Contour surfaces of deflection in y-direction using ten-node tetrahedral elements for the different deflection levels for the bearing holder	227
Figure 6.16(a)	Specifications of the chair	228
Figure 6.16(b)	Boundary conditions and loading of the chair	228
Figure 6.17(a)	Contour surfaces of deflection in y-direction using ten-node tetrahedral elements for the different deflection levels for the chair	229
Figure 6.17(b)	Contour surfaces of deflection in z-direction using ten-node tetrahedral elements for the different deflection levels for the chair	229
Figure 6.18(a)	Specifications of the pan	230
Figure 6.18(b)	Boundary conditions and temperature fixation of the pan	230
Figure 6.19	Contour plot of temperature distribution (in ^{o}C) using ten-node tetrahedral elements for the pan	231
Figure 6.20(a)	Specifications of the shell	231
Figure 6.20(b)	Boundary conditions and temperature fixation of the shell	232
Figure 6.21	Front and top view of contour plot of temperature distribution (in ^{o}C) using ten-node tetrahedral elements for the shell	232

LIST OF TABLES

	Title of Table	Page
Table 2.1	Interpolation Techniques used in Triangular Elements	41
Table 2.2	Different Mathematical Techniques used by Researchers	43
Table 2.3	Different Parametric Curves and Surfaces used by Researchers	47
Table 4.1	Various conic sections represented by a quadratic equation in 2D	97
Table 4.2	Values of coefficients for different contours shown in Fig. 4.22	128
Table 4.3	Comparison of execution time	149
Table 6.1	Different forms of Eq. (6.8)	209
Table 6.2	Different forms of Eq. (6.9)	209
Table 6.3	Values of coefficients for different contours shown in Fig. 6.8	221

CHAPTER - I

INTRODUCTION

1.1 GENERAL

Graphical representation of results is essential to understand numerical behavior of a mathematical model solving a physical problem. The representation substitutes analysis of a mess of large amount of data by a simple visual inspection. Contour plotting of data is one step towards such representation. The contour plot is an indispensable method for graphical representation of results. Contour plots show the same value of a physical parameter in a geometric domain. The term ‘contour’ as used in a topographical survey is defined as a line joining points of equal elevations (Rajasekaran and Venkatesan, 1995).

Contour line plots offer visual information that is not obvious from looking at the voluminous numerical data. It gives a graphical pattern for a numerical data. The large amount of numerical data can be simply represented by this graphical representation for better interpretation. For example, isothermal lines plotted on the visible surfaces of a heat conducting body offer quick information about the spatial temperature distribution prevailing.

Various scientific problems need a contour plotting algorithm that can display a constant-valued surface of a function of two or three variables.

Any piecewise continuous, single-valued function of two continuous independent variables can be represented in the form of a contour map. The most common example is a contour map representing elevation as a function of position in a two-dimensional geographic region. Other geographic-position-dependent variables that are commonly represented in the form of contour maps are temperature (isotherms) and pressure (isobars). However, the use of contour maps need not be restricted to dependent variables relating to geographic position. An example of a contour map used for a more general dependent variable is a plot of equal-loudness curves drawn as a function of the intensity and frequency of an audible tone. In some applications, contour maps may be used to facilitate visualization of data even though an equation may exist that describes this data, for example, a plot of equipotential lines around an electric dipole.

The advantage of a well made contour-map over other types of surface representation is that it exhibits the characteristics of a surface and at the same time provides quantitative information. Even with the present sophistication of 3D computer graphics, the display of contour lines can convey information about the behaviour of a surface that cannot be gleaned even from a shaded colour image.

Algorithms for contour plotting have wide applications in the areas of Medicine and Engineering.

In Medical field, the contour plotting algorithms are mainly used for approximating the surface spanning through a set of 2D and 3D data points. They also help a lot to radiologists to interpret the 2D and 3D images of the data. Researchers have reported the application of 3D medical images in a variety of areas like acetabular fractures, craniofacial abnormalities, complex bone structures, to name a few.

Data analysis in engineering applications frequently involves contour plotting. Although graphics takes many forms, contouring two-dimensional surfaces is one of the most useful techniques, sometimes even indispensable, to represent field variables under investigation.

In Engineering, there are a number of analysis tools which generate results in the form of large amount of numerical data. This data is to be plotted properly for visual inspection. Finite Element Analysis (FEA) is one such tool. FEA is the most prevalent numerical tool used for the design and analysis of engineering systems. It is applicable to virtually any engineering discipline describable by partial differential equations along with proper loads and boundary conditions.

It is a computational technique used to obtain approximate solutions of boundary value problems in engineering. Simply stated, a boundary value problem is a mathematical problem in which one or more dependent variables must satisfy a differential equation everywhere within a known domain of independent variables and satisfy specific conditions on the boundary of the domain. Boundary value problems are also sometimes called *field* problems. The field is the domain of interest and most often represents a physical structure. The *field variables* are the dependent variables of interest governed by the differential equation. The *boundary conditions* are the specified values of field variables on the boundaries of the field. Depending on the type of physical problem being analyzed, the field variables may include physical displacement, temperature, heat flux, and fluid velocity etc.

1.2 BASIC PROCEDURE FOR FEA

The fundamental concept of FEA is that a continuous function can be approximated using a discrete model. The discrete model is composed of one or more *interpolation polynomials*, and the continuous function is divided into finite pieces called *elements*. Each element is defined using an interpolation function to describe its behavior between its end points. The unknown function is approximated in every element by the interpolating polynomial that is continuous with its derivative until a certain order.

Certain steps in formulating a finite element analysis of a physical problem are common to all such analyses, whether structural, heat transfer, fluid flow, or some other problem. The steps (David Hutton, 2004) are described as follows.

(a) Preprocessing: The preprocessing step is, quite generally, described as defining the model and includes defining the

- Geometric domain of the problem
- Element type to be used
- Material properties of the elements
- Element connectivity (mesh the model)
- Physical constraints (boundary conditions)
- Loadings

The preprocessing step is a very critical step.

(b) Solution: During the solution phase, finite element assembles the governing algebraic equations in matrix form and computes the unknown values of the primary field variables.

As it is not uncommon for a finite element model to be represented by large number of equations, special solution techniques are used to reduce data storage requirements and computational time.

(c) Postprocessing: The computed values are used by back substitution to compute additional, derived variables, such as reaction forces, element stresses, and heat flow. Analysis and evaluation of the solution results is referred to as Postprocessing. Postprocessing software contains sophisticated routines used for sorting, printing, and plotting selected results from a finite element solution. The most important objective of Postprocessing is to apply sound engineering judgment in determining whether the solution

results are physically reasonable. Lists of output data are difficult to appraise quickly for judgment so the contour plot algorithms are to be added as a post-processor to give visual representation of results.

1.3 COMPATIBILITY AND COMPLETENESS REQUIREMENTS

In most engineering problems, the domain of interest is a continuous solid body, often of irregular shape, in which the behavior of one or more field variables is governed by one or more partial differential equations. The objective of the finite element method is to discretize the domain into a number of finite elements for which the governing equations are algebraic equations. Solution of the resulting system of algebraic equations then gives an *approximate* solution to the problem.

In FEA, solution accuracy is judged in terms of convergence as the element 'mesh' is refined. There are two major methods of mesh refinement. In the first, known as *h-refinement*, mesh refinement refers to the process of increasing the number of elements used to model a given domain, consequently, reducing individual element size. In the second method, *p-refinement*, element size is unchanged but the order of the polynomials used as interpolation functions is increased. The objective of mesh refinement in either method is to obtain sequential solutions that exhibit asymptotic convergence to values representing the exact solution. If convergence is not obtained, the engineer using the finite element method has absolutely no indication whether the results are indicative of a meaningful approximation to the correct solution. For a general field problem in which the field variable of interest is expressed on an element basis in the discretized form,

$$\phi^{(e)}(x, y, z) = \sum_{i=1}^{M} N_i(x, y, z)\phi_i \tag{1.1}$$

where M is the number of element degrees of freedom and N_i is the interpolation function for the i^{th} node. The interpolation functions must satisfy two primary conditions to ensure convergence during mesh refinement: the *compatibility* and *completeness* requirements.

1.3.1 Compatibility

The field variable and its partial derivatives must be continuous along the element boundaries. These must be continuous up to one order less than the highest-order derivative appearing in the integral formulation of the element equations. Given the discretized

representation of Eq. (1.1), it follows that the interpolation functions must meet this condition, since these functions determine the spatial variation of the field variable.

The compatibility condition can be given a physical meaning as well. In structural problems, the requirement of displacement continuity along element boundaries ensures that no gaps or voids develop in the structure as a result of modeling procedure. In heat transfer problems, the compatibility requirement prevents the physically unacceptable possibility of jump discontinuities in temperature distribution.

1.3.2 Completeness

In the limit as element size shrinks to zero in mesh refinement, the field variable and its partial derivatives up to, and including, the highest-order derivative appearing in the integral formulation must be capable of assuming constant values. Again, because of the discretization, the completeness requirement is directly applicable to the interpolation functions.

The completeness requirement ensures that a displacement field within a structural element can take on a constant value. The completeness requirement also ensures the possibility of constant values of (at least) first derivatives. This feature assures that a finite element is capable of constant strain, constant heat flow, or constant fluid velocity, for example.

The purpose of the following section is to develop the appropriate interpolation functions to a number of commonly used elements of various shapes and complexity.

1.4 POLYNOMIAL FORMS: GEOMETRIC ISOTROPY

Formulation of finite element characteristics requires differentiation and integration of the interpolation functions in various forms. Owing to the simplicity with which polynomial functions can be differentiated and integrated, polynomials are the most commonly used interpolation functions (David Hutton, 2004).

The polynomial representation of the field variable must contain the same number of terms as the number of nodal degrees of freedom. In addition, to satisfy the completeness requirement, the polynomial representation for an *M*-degree of freedom element should contain all powers of the independent variable up to and including *M-1*. Another way of stating the latter requirement is that the polynomial is *complete*. In two and three dimensions, polynomial representations of the field variable, in general, satisfy the compatibility and completeness requirements if the polynomial exhibits the property known as *geometric isotropy*. A

mathematical function satisfies geometric isotropy if the functional form does not change under a translation or rotation of coordinates. In two dimensions, a complete polynomial of order *M* can be expressed as

$$P_M(x,y) = \sum_{k=0}^{N_t^{(2)}} a_k x^i y^j , \qquad i + j \le M \tag{1.2}$$

where $N_t^{(2)} = [(M+1)(M+2)]/2$ is the total number of terms. A complete polynomial as expressed by Eq. (1.2) satisfies the condition of geometric isotropy, since the two variables, *x* and *y*, are included in each term in similar powers. Therefore, a translation or rotation of coordinates in not prejudicial to either of the independent variables.

A graphical method of depicting complete two-dimensional polynomials is the so-called Pascal triangle shown in Fig. 1.1. Each horizontal line represents a polynomial of order *M*. A complete polynomial of order *M* must contain all terms shown above the horizontal line. For example, a complete quadratic polynomial in two dimensions must contain six terms. Hence, for use in a finite element representation of a field variable, a complete quadratic expression requires six nodal degrees of freedom in the element.

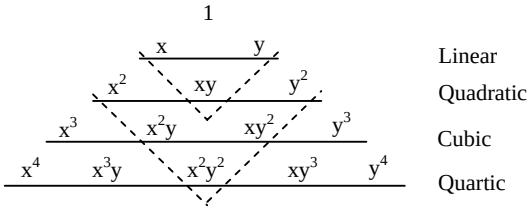


Figure 1.1 Pascal triangle for polynomials in two dimensions

In addition to the complete polynomials, incomplete polynomials also exhibit geometric isotropy if the incomplete polynomial is *symmetric*. In this context, symmetric implies that the independent variables appear as 'equal and opposite pairs', ensuring that each independent variable plays an equal role in the polynomial. For example, the four-term incomplete quadratic polynomial presented in Eq. (1.3) is not symmetric, as there is a quadratic term in *x* but the corresponding quadratic term in *y* does not appear.

$$P(x,y) = a_0 + a_1 x + a_2 y + a_3 x^2 \tag{1.3}$$

On the other hand, the incomplete quadratic polynomial presented in Eq. (1.4) is symmetric, as the quadratic term gives equal 'weight' to both variables.

$$P(x,y) = a_0 + a_1 x + a_2 y + a_3 xy \tag{1.4}$$

A very convenient way of visualizing some of the commonly used incomplete but symmetric polynomials of a given order is also afforded by the Pascal triangle. Referring to Fig. 1.1, the dashed lines show the terms that must be included in an incomplete yet symmetric polynomial of a given order. All terms above the dashed lines must be included in a polynomial representation if the function is to exhibit geometric isotropy. This feature of polynomials is utilized to a significant extent in the development of various element interpolation functions.

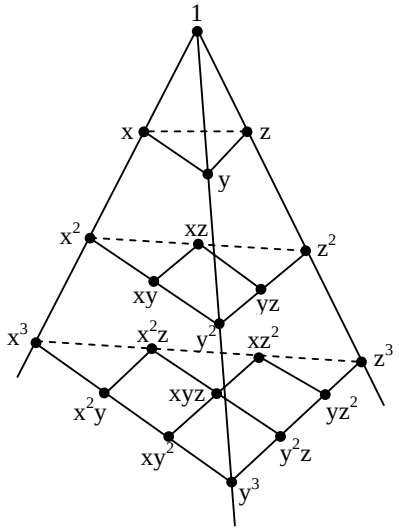


Figure 1.2 Pascal pyramid for polynomials in three dimensions

As in the two-dimensional case, the polynomial expression of the field variable in three dimensions must be complete or incomplete but symmetric. Completeness and symmetry can also be depicted graphically by the 'Pascal Pyramid' shown in Fig. 1.2. While the three-dimensional case is a bit more difficult to visualize, the basic premise remains that each independent variable must be of equal 'strength' in the polynomial. For example, the 3D quadratic polynomial as presented in Eq. (1.5) is complete and could be applied to an element having ten nodes.

$$P(x,y,z) = a_0 + a_1 x + a_2 y + a_3 z + a_4 x^2 + a_5 y^2 + a_6 z^2 + a_7 xy + a_8 xz + a_9 yz \tag{1.5}$$

Similarly, an incomplete, symmetric form presented in Eq. (1.6) could be used for elements having seven nodal degrees of freedom.

$$P(x,y,z) = a_0 + a_1 x + a_2 y + a_3 z + a_4 x^2 + a_5 y^2 + a_6 z^2 \quad \text{or} \tag{1.6}$$

$$P(x,y,z) = a_0 + a_1 x + a_2 y + a_3 z + a_4 xy + a_5 xz + a_6 yz$$

Geometric isotropy is not an *absolute* requirement for field variable representation, hence, interpolation functions. Incomplete representations are quite often used and solution convergence is attained. However, in terms of *h*-refinement, use of geometrically isotropic representations guarantees satisfaction of the compatibility and completeness requirements. For the *p*-refinement method, the interpolation functions in any finite element analysis solution are approximations to the power series expansion of the problem solution. As the number of element nodes is increased, the order of the interpolation function increases and, in the limit, as the number of nodes approaches infinity, the polynomial expression of the field variable approaches the power series expansion of the solution.

1.5 TRIANGULAR ELEMENTS

The interpolation functions (David Hutton, 2004) for triangular elements are inherently formulated in two dimensions and a family of such elements exists. Fig. 1.3 depicts the first three elements (linear, quadratic, and cubic) of the family. The internal node in cubic element is required to obtain geometric isotropy.

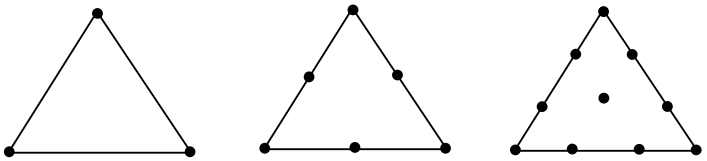


Figure 1.3 Triangular elements: three-node linear, six-node quadratic, ten-node cubic

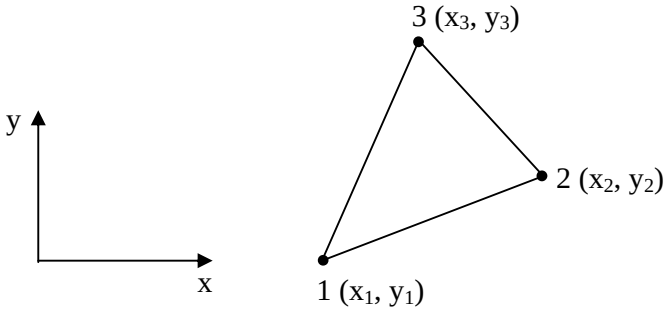


Figure 1.4 A general three-node triangular element referred to global coordinates

Fig. 1.4 depicts a general, three-node triangular element to which an element coordinate system is attached. Here, it is assumed that only *one* degree of freedom is associated with each node. The field variable is expressed in the polynomial form as shown in Eq. (1.7).

$$\phi(x, y) = a_0 + a_1 x + a_2 y \tag{1.7}$$

Applying the nodal conditions

$$\phi(x_1, y_1) = \phi_1$$

$$\phi(x_2, y_2) = \phi_2 \tag{1.8}$$

$$\phi(x_3, y_3) = \phi_3$$

In matrix form,

$$\begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix} \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \end{Bmatrix} = \begin{Bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{Bmatrix} \tag{1.9}$$

Inverting the Eq. (1.9),

$$a_0 = \frac{1}{2A}\left[\phi_1(x_2 y_3 - x_3 y_2) + \phi_2(x_3 y_1 - x_1 y_3) + \phi_3(x_1 y_2 - x_2 y_1)\right]$$

$$a_1 = \frac{1}{2A}\left[\phi_1(y_2 - y_3) + \phi_2(y_3 - y_1) + \phi_3(y_1 - y_2)\right] \tag{1.10}$$

$$a_2 = \frac{1}{2A}\left[\phi_1(x_3 - x_2) + \phi_2(x_1 - x_3) + \phi_3(x_2 - x_1)\right]$$

Substituting the values into Eq. (1.7) and collecting coefficients of the nodal variables,

$$\phi(x, y) = \frac{1}{2A}\begin{Bmatrix} \left[(x_2 y_3 - x_3 y_2) + (y_2 - y_3)x + (x_3 - x_2)y\right]\phi_1 \\ + \left[(x_3 y_1 - x_1 y_3) + (y_3 - y_1)x + (x_1 - x_3)y\right]\phi_2 \\ + \left[(x_1 y_2 - x_2 y_1) + (y_1 - y_2)x + (x_2 - x_1)y\right]\phi_3 \end{Bmatrix} \tag{1.11}$$

Given the form of Eq. (1.11), the interpolation functions are

$$N_1(x, y) = \frac{1}{2A}\left[(x_2 y_3 - x_3 y_2) + (y_2 - y_3)x + (x_3 - x_2)y\right]$$

$$N_2(x, y) = \frac{1}{2A}\left[(x_3 y_1 - x_1 y_3) + (y_3 - y_1)x + (x_1 - x_3)y\right] \tag{1.12}$$

$$N_3(x, y) = \frac{1}{2A}\left[(x_1 y_2 - x_2 y_1) + (y_1 - y_2)x + (x_2 - x_1)y\right]$$

where A is the area of the triangular element. Given the coordinates of the three vertices of a triangle, the area is given by Eq. (1.13).

$$A = \frac{1}{2}\begin{vmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{vmatrix} \tag{1.13}$$

The algebraically complex form of the interpolation functions arises primarily from the choice of the element coordinate system. As the linear representation of the field variable exhibits geometric isotropy, location and orientation of the element coordinate axes can be chosen arbitrarily without affecting the interpolation results.

1.5.1 Area Coordinates

When expressed in Cartesian coordinates, the interpolation functions for the triangular element are algebraically complex. Further, the integrations required to obtain element characteristic matrices are cumbersome. Considerable simplification of the interpolation functions as well as the subsequently required integration is obtained *via* the use of *area coordinates*.

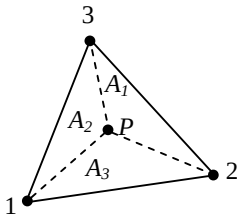


Figure 1.5 Areas used to define area coordinates for a triangular element

Fig. 1.5 shows a three-node triangular element divided into three areas defined by the nodes and an arbitrary interior point $P(x, y)$. The area coordinates of P are defined as

$$L_1 = \frac{A_1}{A} \qquad L_2 = \frac{A_2}{A} \qquad L_3 = \frac{A_3}{A} \tag{1.14}$$

where A is the total area of the triangle. Clearly, the area coordinates are not independent, since $L_1 + L_2 + L_3 = 1$ (1.15)

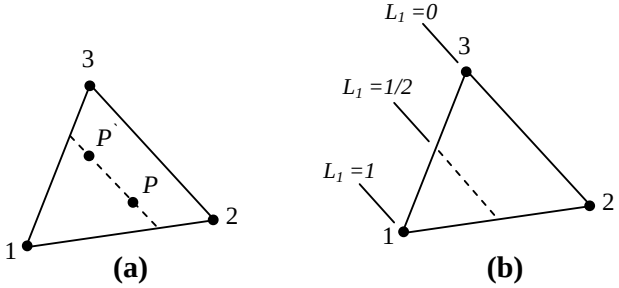


Figure 1.6 (a) Area A_1 associated with either P or P' is constant; (b) Lines of the constant area coordinate L_1

In Fig. 1.6 (a), a dashed line parallel to the side defined by nodes 2 and 3 is indicated. For any two points P and P' on this line, the areas of the triangles formed by nodes 2, 3 and either of P or P' are identical. This is because the base and height of any triangle so formed are constants. Further, as the dashed line is moved closer to node 1, area A_1 increases linearly and has value $A_1 = A$, when evaluated at node 1. Therefore, area coordinate L_1 is constant on any line parallel to the side of the triangle opposite node 1 and varies linearly from a value of *unity* at node 1 to a value of *zero* along the side defined by nodes 2 and 3, as depicted in Fig. 1.6 (b). Similar arguments are made for the behavior of L_2 and L_3. These observations are used to write the following conditions satisfied by the area coordinates when evaluated at the nodes:

$$\begin{aligned} &\text{Node 1:} \quad L_1 = 1 \qquad L_2 = L_3 = 0 \\ &\text{Node 2:} \quad L_2 = 1 \qquad L_1 = L_3 = 0 \\ &\text{Node 3:} \quad L_3 = 1 \qquad L_1 = L_2 = 0 \end{aligned} \tag{1.16}$$

The conditions expressed by Eq. (1.16) are exactly the conditions that must be satisfied by interpolation functions at the nodes of the triangular element. So, the field variables are expressed in terms of area coordinates as

$$\phi(x, y) = L_1\phi_1 + L_2\phi_2 + L_3\phi_3 \tag{1.17}$$

The advantage of using area coordinates is seen in developing interpolation functions for higher-order elements and performing integration of various forms of the interpolation functions.

1.5.2 Six-Node Triangular Element

A six-node element is shown in Fig. 1.7 (a). The additional nodes 4, 5, and 6 are located at the midpoints of the sides of the element. For six node, a complete polynomial representation of the field variable is given by Eq. (1.18) which is ultimately to be expressed in terms of interpolation functions and nodal values as given in Eq. (1.19).

$$\phi(x, y) = a_0 + a_1 x + a_2 y + a_3 x^2 + a_4 xy + a_5 y^2 \tag{1.18}$$

$$\phi(x, y) = \sum_{i=1}^{6} N_i(x, y)\phi_i \tag{1.19}$$

As usual, each interpolation function is such that its value is unity when evaluated at its associated node and zero when evaluated at any of the other five nodes. Further, each interpolation is a quadratic function, since the field variable representation is quadratic.

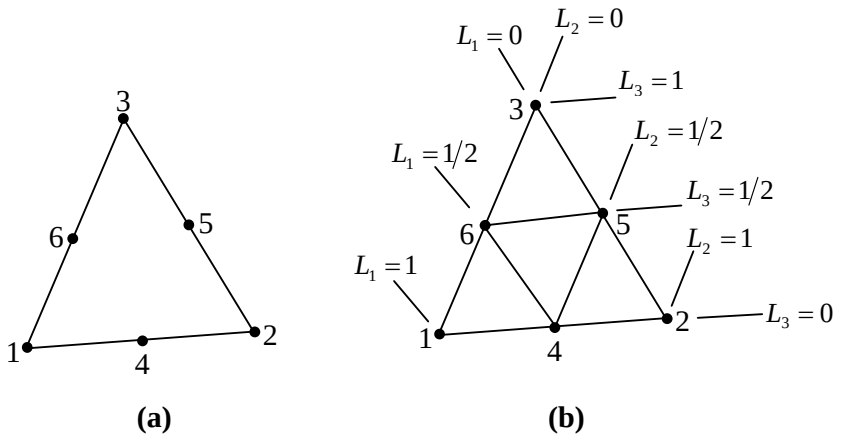


Figure 1.7 Six-node triangular elements: (a) node numbering; (b) lines of constant values of the are coordinates

Fig. 1.7 (b) shows the six-node element with lines of constant values of the area coordinates passing through the nodes. Using this figure (and a bit of logic), the interpolation functions are easily 'constructed' in area coordinates. For example, interpolation function $N_1(x,y) = N_1(L_1, L_2, L_3)$ must have value of zero at nodes 2, 3, 4, 5, and 6. Noting that $L_1 = 1/2$ at nodes 4 and 6, inclusion of the term $L_1 - 1/2$ ensures a zero value at those two nodes. Similarly, $L_1 = 0$ at nodes 2, 3, 4, so the term L_1 satisfies the conditions at those three nodes. Therefore, N_1 is as given by Eq. (1.20).

$$N_1 = L_1\left(L_1 - \frac{1}{2}\right) \tag{1.20}$$

However, evaluation of Eq. (1.20) at node 1, where $L_1 = 1$, results in $N_1 = 1/2$. As N_1 must be unity at node 1, Eq. (1.20) is modified to the following form which satisfies the required conditions at each of the six nodes and is a quadratic function, since L_1 is a linear function of *x* and *y*.

$$N_1 = 2L_1\left(L_1 - \frac{1}{2}\right) = L_1(2L_1 - 1) \tag{1.21}$$

Applying the required nodal conditions to the remaining five interpolation functions in turn, the following is obtained:

$$N_2 = L_2(2L_2 - 1)$$

$$N_3 = L_3(2L_3 - 1)$$

$$N_4 = 4L_1L_2 \tag{1.22}$$

$$N_5 = 4L_2L_3$$

$$N_6 = 4L_1L_3$$

Using a similar straightforward procedure, interpolation functions for additional higher-order triangular elements are constructed.

1.5.3 Integration in Area Coordinates

The second advantage of using area coordinates is in performing integration of various forms of the interpolation functions.

Integration of various forms of the interpolation functions over the domain of an element are required in formulating element characteristic matrices and load vectors. When expressed in area coordinates, integrals of the following form are often used

$$\iint_A L_1^a L_2^b L_3^c dA \tag{1.23}$$

The relation given by Eq. (1.23) is evaluated as

$$\iint_A L_1^a L_2^b L_3^c dA = (2A)\frac{a!b!c!}{(a+b+c+2)!} \tag{1.24}$$

Eq. (1.24) is valid for all exponents *a*, *b*, *c* that are positive integers. Therefore, integration in area coordinates is quite straightforward.

1.6 RECTANGULAR ELEMENTS

Rectangular elements are convenient for use in modeling regular geometries (David Hutton, 2004). The simplest of the rectangular family of elements is the four-node rectangle shown in Fig. 1.8, where it is assumed that the sides of the rectangular are parallel to the global cartesian axes. As there are four nodes and four degrees of freedom, a four-term polynomial expression for the field variable is appropriate. Since there is no complete four-term polynomial in two dimensions, the incomplete, symmetric expression as given in Eq. (1.25) is used to ensure geometric isotropy.

$$\phi(x, y) = a_0 + a_1 x + a_2 y + a_3 xy \tag{1.25}$$

Applying the four nodal conditions and writing in matrix form, we get

$$\begin{Bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \end{Bmatrix} = \begin{bmatrix} 1 & x_1 & y_1 & x_1 y_1 \\ 1 & x_2 & y_2 & x_2 y_2 \\ 1 & x_3 & y_3 & x_3 y_3 \\ 1 & x_4 & y_4 & x_4 y_4 \end{bmatrix} \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{Bmatrix} \tag{1.26}$$

which formally gives the polynomial coefficients as

$$\begin{Bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{Bmatrix} = \begin{bmatrix} 1 & x_1 & y_1 & x_1 y_1 \\ 1 & x_2 & y_2 & x_2 y_2 \\ 1 & x_3 & y_3 & x_3 y_3 \\ 1 & x_4 & y_4 & x_4 y_4 \end{bmatrix}^{-1} \begin{Bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \end{Bmatrix} \tag{1.27}$$

In terms of the nodal values, the field variable is then described by

$$\phi(x, y) = \begin{bmatrix} 1 & x & y & xy \end{bmatrix} \{a\} = \begin{bmatrix} 1 & x & y & xy \end{bmatrix} \begin{bmatrix} 1 & x_1 & y_1 & x_1 y_1 \\ 1 & x_2 & y_2 & x_2 y_2 \\ 1 & x_3 & y_3 & x_3 y_3 \\ 1 & x_4 & y_4 & x_4 y_4 \end{bmatrix}^{-1} \begin{Bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \end{Bmatrix} \tag{1.28}$$

Eq. (1.28) is used to deduce the interpolation function.

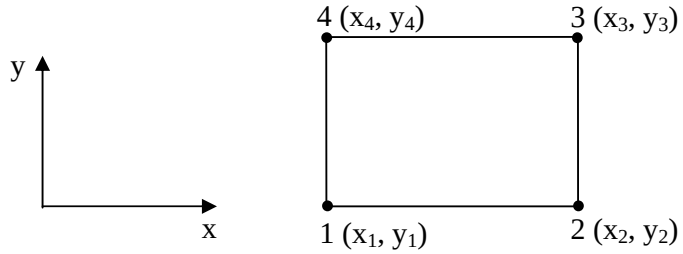


Figure 1.8 A four-node rectangular element defined in global coordinates

The form of Eq. (1.28) suggests that expression of the interpolation functions in terms of the nodal coordinates is algebraically complex. Fortunately, the complexity can be reduced by a more judicious choice of coordinates. For rectangular element, the normalized coordinates (also known as *natural coordinates* or *serendipity coordinates*) r and s are as given in Eq. (1.29).

$$r = \frac{x - \bar{x}}{a} \qquad s = \frac{y - \bar{y}}{b} \tag{1.29}$$

where $2a$ and $2b$ are the width and height of the rectangle, respectively, and the coordinates of the centroid are as given in Eq. (1.30), shown in Fig. 1.9(a).

$$\bar{x} = \frac{x_1 + x_2}{2} \qquad \bar{y} = \frac{y_1 + y_4}{2} \tag{1.30}$$

Therefore, r and s are such that the values range from -1 to +1, and the nodal coordinates are as in Fig. 1.9 (b).

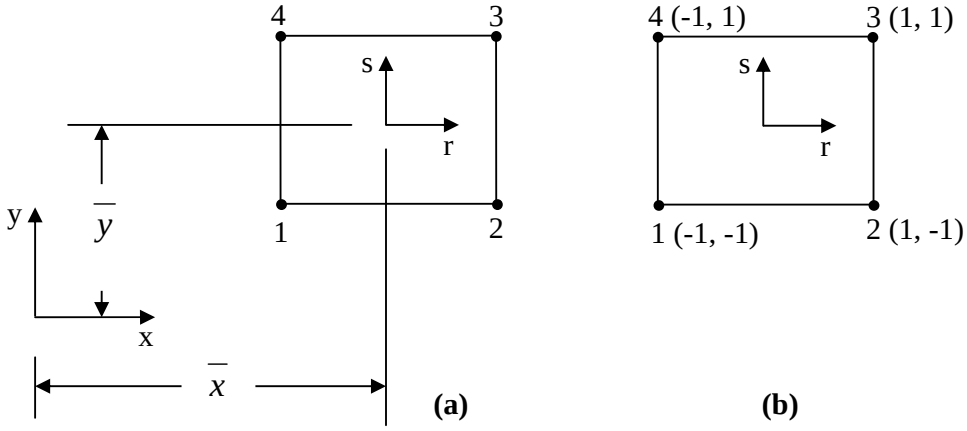


Figure 1.9 A four-node rectangular element showing (a) the translation to natural coordinates; (b) the natural coordinates of each node

Applying the conditions that must be satisfied by each interpolation function at each node, the following is obtained (essentially by inspection)

$$N_1(r,s) = \frac{1}{4}(1-r)(1-s)$$

$$N_2(r,s) = \frac{1}{4}(1+r)(1-s) \tag{1.31}$$

$$N_3(r,s) = \frac{1}{4}(1+r)(1+s)$$

$$N_4(r,s) = \frac{1}{4}(1-r)(1+s)$$

Hence

$$\phi(x,y) = \phi(r,s) = N_1(r,s)\phi_1 + N_2(r,s)\phi_2 + N_3(r,s)\phi_3 + N_4(r,s)\phi_4 \tag{1.32}$$

As in the case of triangular elements using area coordinates, the interpolation functions are much simpler algebraically when expressed in terms of the natural coordinates. Nevertheless,

all required conditions are satisfied and the functional form is identical to that used to express the field variables in Eq. (1.25).

To develop a higher-order rectangular element, the logical progression is to place an additional node at the midpoint of each side of the element, as in Fig. 1.10. This poses an immediate problem, however. Inspection of the Pascal triangle shows that a complete polynomial having eight terms cannot be constructed, but a choice of two incomplete, symmetric cubic polynomials is available:

$$\phi(x,y) = a_0 + a_1 x + a_2 y + a_3 x^2 + a_4 xy + a_5 y^2 + a_6 x^3 + a_7 y^3 \tag{1.33}$$

$$\phi(x,y) = a_0 + a_1 x + a_2 y + a_3 x^2 + a_4 xy + a_5 y^2 + a_6 x^2 y + a_7 xy^2 \tag{1.34}$$

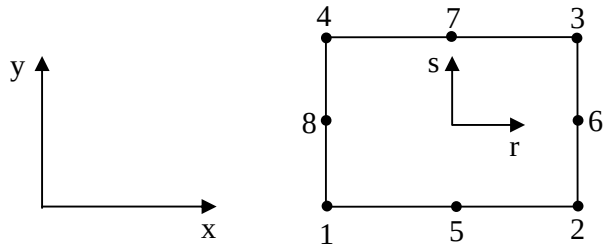


Figure 1.10 Eight-node rectangular element showing both global and natural coordinate axes

Rather than grapple with choosing one or the other, the natural coordinates and the nodal coordinates are used that must be satisfied by each interpolation function to obtain the functions serendipitously. For example, interpolation function N_1 must evaluate to zero at all nodes except node 1, where its value must be unity. At nodes 2, 3, and 6, $r = 1$, so including the term '$r-1$' satisfies the zero condition at those nodes. Similarly, at nodes 4 and 7, $s = 1$ so the term '$s-1$' ensures the zero condition at those two nodes. Finally, at node 5, $(r,s) = (0,-1)$, and at node 8, $(r,s) = (-1,0)$. Hence, at nodes 5 and 8, the term '$r+s+1$' is identically zero. Using this reasoning, the interpolation function associated with node 1 is to be of the form as expressed in Eq. (1.35).

$$N_1(r,s) = (1-r)(1-s)(r+s+1) \tag{1.35}$$

Evaluating at node 1 where $(r,s) = (-1,-1)$, we obtain $N_1 = -4$, so a correction is required to obtain the unity value. The final form is then

$$N_1(r,s) = \frac{1}{4}(r-1)(1-s)(r+s+1) \tag{1.36}$$

A parallel procedure for the interpolation functions associated with the other three corner nodes leads to the following:

$$N_2(r,s) = \frac{1}{4}(r+1)(1-s)(s-r+1) \tag{1.37}$$

$$N_3(r,s) = \frac{1}{4}(1+r)(1+s)(r+s-1) \tag{1.38}$$

$$N_4(r,s) = \frac{1}{4}(r-1)(1+s)(r-s+1) \tag{1.39}$$

The form of the interpolation functions associated with the midpoint nodes is simpler to obtain than those for the corner nodes. For example, N_5 has a value of zero at nodes 2, 3, and 6 if it contains the term '$r-1$' and is also zero at nodes 1, 4, and 8 if the term '$1+r$' is included. Finally, if a zero value is at node 7, $(r,s) = (0,1)$ is obtained by inclusion of '$s-1$'. The form for N_5 is

$$N_5 = \frac{1}{2}(1-r)(1+r)(1-s) = \frac{1}{2}(1-r^2)(1-s) \tag{1.40}$$

where the leading coefficient ensures a *unity* value at node 5. For the mid-side nodes, the shape functions are determined in the same manner.

$$N_6 = \frac{1}{2}(1+r)(1-s^2) \tag{1.41}$$

$$N_7 = \frac{1}{2}(1-r^2)(1+s) \tag{1.42}$$

$$N_8 = \frac{1}{2}(1-r)(1-s^2) \tag{1.43}$$

Many other, successively higher-order, rectangular elements can be developed. In general, these higher-order elements include internal nodes that, in modeling, are troublesome, as they cannot be connected to nodes of other elements.

1.7 THREE-DIMENSIONAL ELEMENTS

As in two-dimensional case, there are two main families of three-dimensional elements. The first is based on extension of triangular elements to tetrahedrons and the other on extension of rectangular elements to rectangular parallelepipeds (often simply called *brick* elements). The

algebraically cumbersome technique for deriving interpolation functions in global cartesian coordinates has been illustrated for two-dimensional elements. These developments are not repeated for three-dimensional elements; the procedures are algebraically identical but even more complex. Instead, only the more amenable approach of natural coordinates is used to develop the interpolation functions (David Hutton, 2004) for the two basic elements of the tetrahedral and brick families.

1.7.1 Four-Node Tetrahedral Element

A four-node tetrahedral element is depicted in Fig. 1.11 in relation to a global cartesian coordinate system.

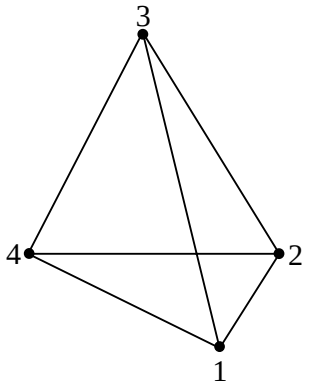


Figure 1.11 A four-node tetrahedral element

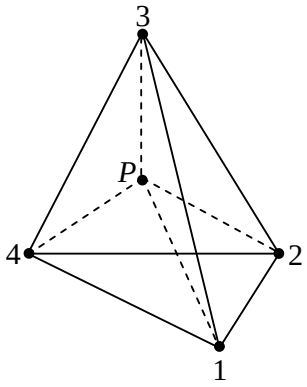


Figure 1.12 A four-node tetrahedral element, showing an arbitrary point defining four volumes

In a manner analogous to use of area coordinates, the concept of *volume coordinates* is shown in Fig. 1.12. Point $P(x, y, z)$ is an arbitrary point in the tetrahedron defined by the four nodes. As indicated by the dashed lines, point P and the four nodes define four other tetrahedra having volumes as given by Eq. (1.44).

$$V_1 = vol(P234) \qquad V_2 = vol(P134) \tag{1.44}$$

$$V_3 = vol(P124) \qquad V_4 = vol(P123)$$

The volume coordinates are defined as follows:

$$L_1 = \frac{V_1}{V} \qquad L_2 = \frac{V_2}{V} \qquad L_3 = \frac{V_3}{V} \qquad L_4 = \frac{V_4}{V} \tag{1.45}$$

where V is the total volume of the element given by

$$V = \frac{1}{6}\begin{vmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{vmatrix} \tag{1.46}$$

As with the area coordinates, the volume coordinates are not independent, since

$$V_1 + V_2 + V_3 + V_4 = V \tag{1.47}$$

The volume coordinates satisfy all required nodal coordinates for interpolation functions. The field variables can be expressed as in Eq. (1.48).

$$\phi(x, y, z) = L_1\phi_1 + L_2\phi_2 + L_3\phi_3 + L_4\phi_4 \tag{1.48}$$

As with area coordinates, integration of functions of volume coordinates is relatively simple. Other elements of the tetrahedral family are depicted in Fig. 1.13.

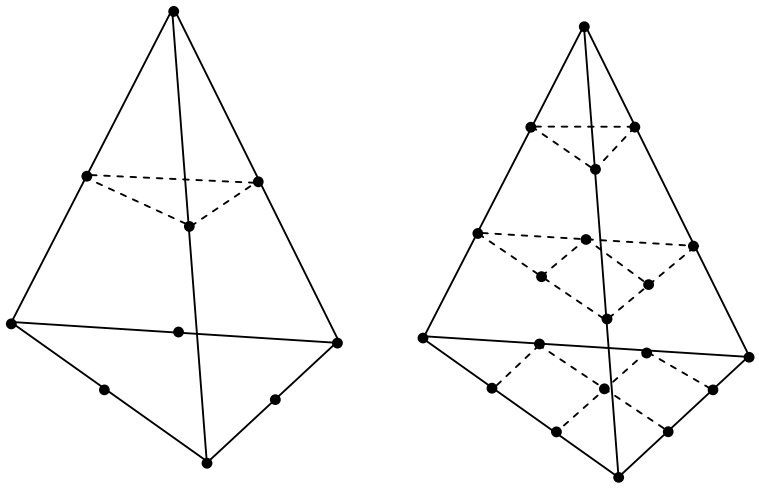


Figure 1.13 Higher-order tetrahedral elements: ten-node and twenty-node

1.7.2 Eight-Node Brick Element

The brick element shown in Fig. 1.14 (a), are in reference to a global cartesian coordinate system. The natural coordinates r, s, t are defined as per Eq. (1.49).

$$r = \frac{x - \bar{x}}{a} \qquad s = \frac{y - \bar{y}}{b} \qquad t = \frac{z - \bar{z}}{c} \tag{1.49}$$

where $2a$, $2b$ and $2c$ are the dimensions of the element in the x, y, z coordinates, respectively, and the coordinates of the element centroid are given by Eq. (1.50).

$$\bar{x} = \frac{x_2 - x_1}{2} \qquad \bar{y} = \frac{y_3 - y_2}{2} \qquad \bar{z} = \frac{z_5 - z_1}{2} \tag{1.50}$$

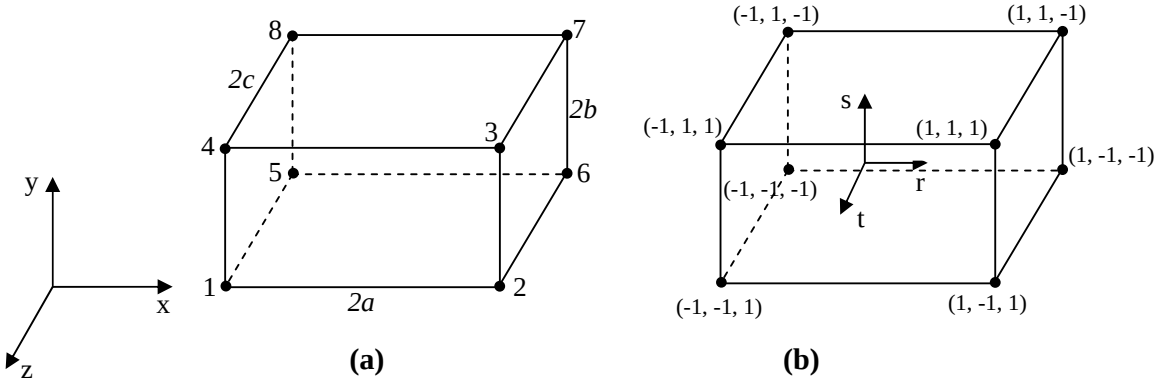


Figure 1.14 Eight-node brick element: (a) global cartesian coordinates; (b) natural coordinates with an origin at the centroid

The natural coordinates are defined such that the coordinate values vary between -1 and +1 over the domain of the element. As with the plane rectangular element, the natural coordinates provide for a straightforward development of the interpolation functions by using the appropriate monomial terms to satisfy nodal conditions. The interpolation functions are written as in Eq. (1.51).

$$N_1 = \frac{1}{8}(1 - r)(1 - s)(1 + t)$$

$$N_2 = \frac{1}{8}(1 + r)(1 - s)(1 + t)$$

$$N_3 = \frac{1}{8}(1 + r)(1 + s)(1 + t)$$

$$N_4 = \frac{1}{8}(1 - r)(1 + s)(1 + t) \tag{1.51}$$

$$N_5 = \frac{1}{8}(1-r)(1-s)(1-t)$$

$$N_6 = \frac{1}{8}(1+r)(1-s)(1-t)$$

$$N_7 = \frac{1}{8}(1+r)(1+s)(1-t)$$

$$N_8 = \frac{1}{8}(1-r)(1+s)(1-t)$$

and the field variable is described by Eq. (1.52).

$$\phi(x,y,z) = \sum_{i=1}^{8} N_i(r,s,t)\phi_i \tag{1.52}$$

If Eq. (1.52) is expressed in terms of the global cartesian coordinates, it is of the form as given in Eq. (1.53).

$$\phi(x,y,z) = a_0 + a_1 x + a_2 y + a_3 z + a_4 xy + a_5 xz + a_6 yz + a_7 xyz \tag{1.53}$$

Eq. (1.53) shows that the field variable is expressed as an incomplete, symmetric polynomial. Geometric isotropy is therefore assured. The compatibility requirement is satisfied, as is the completeness condition.

1.8 ISOPARAMETIC FORMULATION

FEA is a powerful technique for analyzing engineering problems involving complex, irregular geometries. However, the two dimensional and three dimensional elements cannot be efficiently used for irregular geometries (David Hutton, 2004). Consider the plane area shown in Fig. 1.15 (a), which is to be discretized via a mesh of finite elements. A possible mesh using triangular elements is shown in Fig. 1.15 (b). It can be seen that the outermost row of elements provides a chordal approximation to the circular boundary, and as the size of the elements is decreased (*i.e.* the number of elements increased), the approximation becomes increasingly closer to the actual geometry. Although the triangular element can be used to closely approximate a curved boundary, other considerations dictate a relatively large number of elements and associated computation time.

If the rectangular elements are considered, as in Fig. 1.15 (c), the problems are apparent. Unless the elements are small, the area of the domain excluded from the model (the shaded area in the figure) is significant. In such cases, combination of the element types can be used

to improve the geometric accuracy of the model. Such combination of element types may not be the best in terms of solution accuracy since the rectangular element and the triangular element have different order polynomial representation of the field variable.

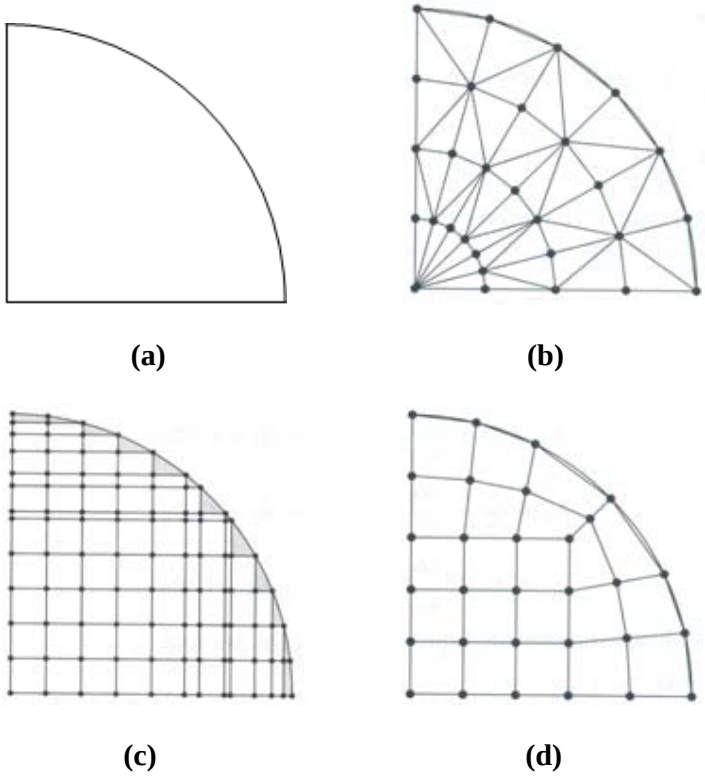


Figure 1.15 (a) A domain to be modeled; (b) Triangular elements; (c) Rectangular elements; (d) Rectangular and Quadrilateral elements

The field variable is continuous across such element boundaries; is guaranteed by the finite element formulation. However, conditions on derivatives of the field variable for the two element types are quite different.

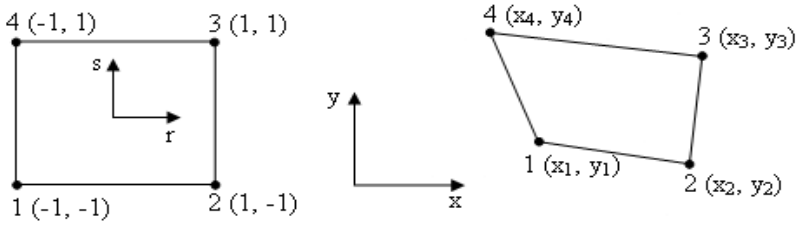


Figure 1.16 Mapping of a parent element into an isoparametric element

Now the Fig. 1.15 (d), shows the same area meshed with rectangular elements and a new element applied near the periphery of the domain. The new element has four nodes, straight sides, but not rectangular. The new element is known as a general two-dimensional

quadrilateral element, which can mesh ideally the curved boundary. The four-node quadrilateral element is derived from the four-node rectangular element (known as the *parent* element) via a mapping process. Fig. 1.16 shows the parent element and its natural (r,s) coordinates and the quadrilateral element in a global cartesian coordinate system.

The geometry of the quadrilateral element is given by Eq. (1.54) and (1.55).

$$x = \sum_{i=1}^{4} G_i(x,y)x_i \tag{1.54}$$

$$y = \sum_{i=1}^{4} G_i(x,y)y_i \tag{1.55}$$

where the $G_i(x,y)$ are the geometric interpolation functions, and each such function is associated with a particular node of the quadrilateral element. Given the geometry and the form of Eq. (1.54) and (1.55), each function $G_i(x,y)$ evaluates to unity at its associated node and to zero at each of the other three nodes, conditions remaining same as those imposed on the interpolation functions of the parent element. Consequently, the interpolation functions for the parent element can be used for the geometric functions, if we map the coordinates, so that

$$\begin{aligned}
&(r,s) = (-1,-1) \Rightarrow (x_1, y_1) \\
&(r,s) = (1,-1) \Rightarrow (x_2, y_2) \\
&(r,s) = (1,1) \Rightarrow (x_3, y_3) \\
&(r,s) = (-1,1) \Rightarrow (x_4, y_4)
\end{aligned} \tag{1.56}$$

where (r,s) are the actual coordinates of the 2 unit by 2 unit parent element.

Consequently, the geometric expressions become as shown in Eq. (1.57).

$$x = \sum_{i=1}^{4} N_i(r,s)x_i \ , \qquad y = \sum_{i=1}^{4} N_i(r,s)y_i \tag{1.57}$$

The field variable variation in the quadrilateral element is also expressed as shown in Eq. (1.58).

$$\phi(x,y) = \phi(r,s) = \sum_{i=1}^{4} N_i(r,s)\phi_i \tag{1.58}$$

Since the same interpolation functions are used for both the field variable and description of the element geometry, the procedure is known as *isoparametric mapping*. The element defined by such a procedure is known as an *isoparametric element*.

The isoparametric formulation is by no means limited to linear parent element. Many higher-order isoparametric elements are being formulated and used. Fig. 1.17 depicts the isoparametric elements corresponding to the six-node triangle and eight-node rectangle. Owing to the mapping being described by quadratic functions of the parent elements, the resulting elements have curved boundaries, which are also described by quadratic functions of the global coordinates. Such elements can be used to closely approximate irregular boundaries.

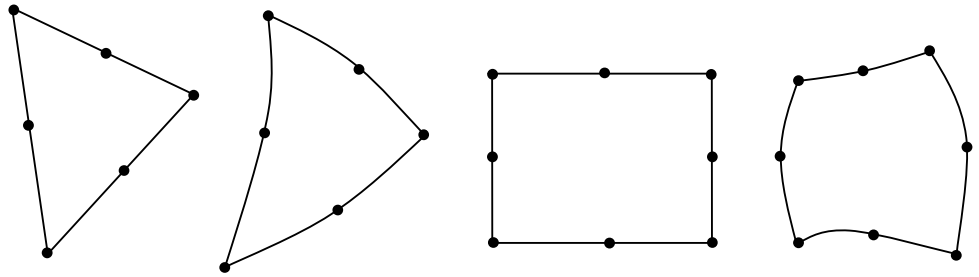


Figure 1.17 Isoparametric mapping of quadratic elements into curved elements: (a) six-node triangle; (b) eight-node rectangle

1.9 OBJECTIVES OF THE PRESENT WORK

Most procedures use some type of linear interpolation to approximate the contour. In general, the use of linear interpolation to locate contour lines on higher order finite elements destroys much of the interelement information contained within the results of an analysis. A number of researchers *viz* Akin and Gray (1977, 1979); Kontopidis and Limbert (1983); Rajasekaran (1984); Akin, Gray and Zhang (1984); Stelzer and Welzel (1987); Singh (1990); Twu and Wang (1992); Rajasekaran and Venkatesan (1995) have presented a general non-linear interpolation procedure for contouring on different isoparametric finite elements based on isoparametric interpolation functions. The use of interpolation functions in contour plotting does not destroy or invent any information that was not already contained within the results. The work presented here extends the procedure with regards to accuracy and speed in following the contour lines or surfaces.

So, the purpose of present research work is to develop accurate and fast contouring in 2D and 3D domains using linear and non-linear interpolation over triangular, quadrilateral and isoparametric meshes. The present work comprises of the following phases:

1. Development of contour plotting algorithm over triangular meshes using linear and non-linear interpolation.
2. Development of contour plotting algorithms over a quadrilateral, four-node and eight-node isoparametric elements.
3. Extension of the above two types of algorithms for contour plotting on tetrahedral and hexahedral elements in 3D.
4. Development of graphics display and visualization routines for displaying contours on 2D and 3D domains.
5. Implementation of above algorithms in C language using Turbo C graphics library for the display of contours.
6. Comparison of results (accuracy of interpolation and speed) of the new algorithms with those of the existing techniques.

1.10 ORGANIZATION OF THESIS

The write up of the thesis has been organized into seven chapters:

Chapter I (Introduction) gives the basic introduction to contour plotting with its application mainly to the FEA analysis results. The purpose of interpolation functions in FEA and contour plotting is highlighted. Further the interpolation functions for different 2D and 3D elements are given.

Chapter II (Literature Review) attempts to record in brief, the findings from literature on various algorithms for contour plotting. Based on description of all the related algorithms proposed by various researchers, the proposed algorithms for different 2D and 3D elements have been developed.

Chapter III (Contouring in 2D using Linear Elements) presents in detail the contouring in 2D using linear triangular and quadrilateral elements. The contour equations are developed over these elements using the interpolation or shape functions. The developed contour equations over these elements represent the linear equations, which are used to plot the contours.

Chapter IV (Contouring in 2D using Non-Linear Elements) describes the detail of contour plotting in 2D using higher order triangular and quadrilateral elements. The contour equations are developed over these elements using these shape functions. The developed contour equation over six-node triangular element represents the equation of various conic sections. In this case, the problem reduces to accurately tracing the contour segment represented by a conic section over a six-node triangle. The contour equation over eight-node quadrilateral element is a cubic equation. The different contour shapes are investigated and a fast algorithm is developed to reduce the time taken to plot these contours.

Chapter V (Contouring in 3D using Linear Elements) covers the contour plotting in 3D using linear tetrahedral and hexahedral elements. The contour equation developed over tetrahedral element using shape functions represents a plane, which is used to plot the contour surfaces. Thereafter, a technique for accurate and fast tracing of contour surfaces using hexahedral elements is discussed.

Chapter VI (Contouring in 3D using Non-Linear Elements) gives the detail of contour plotting using ten-node tetrahedral elements. This element provides a quadratic interpolation functions which is used to accurately interpolate the physical quantity without any loss of information. The contour equation over a ten-node tetrahedral element represents the quadratic surfaces which are used to plot the contour surfaces.

Chapter VII (Conclusions and Future Scope) highlights the conclusions and various potential areas for further research. The major inferences and learnings drawn from various developed algorithms have been presented. Based on the results and findings, various conclusions have been drawn.

1.11 CONCLUDING REMARKS

Contour plot algorithms are a very important part of post processing in FEA to give a visual representation of large amount of generated output data.

The interpolating polynomial which is used to approximate the variation of unknown function in every element can be used to plot the output results of FEA. The use of interpolation polynomial or interpolating functions can plot the contours as accurate as the FEA results.

So the present work uses the interpolation functions to accurately plot the contours over the most commonly used 2D and 3D elements.

An extensive review of literature is the first logical step in a research effort and the next chapter is devoted to the same.

CHAPTER - II

LITERATURE REVIEW

2.1 INTRODUCTION

The results of any research are useful when relevant previous literature has been reviewed and analyzed. This chapter attempts to record in brief, the findings from literature on various algorithms for contour plotting. A review of the various algorithms, their advantages, their limitations and applications has been carried out.

2.2 NEED FOR CONTOUR PLOTTING ALGORITHMS

As stated in the previous chapter, graphic representation of results is essential to understand numerical model behaviors. This representation substitutes analyses of a mess of large amount of data by a simple visual inspection. With a lot of advancements in hardware and software technologies, it is need of the hour to accurately display this representation, that too with high speed. Enormous research has taken place to explain the contour plotting algorithms in the field of Engineering and Medicine.

In Engineering, there are a number of analysis tools which generate results in the form of large amount of data. This data is to be plotted properly for visual inspection.

In Medical field, three-dimensional surfaces of anatomy offer a valuable tool. Images of these surfaces, constructed from multiple 2D slices of computed tomography (CT), magnetic resonance (MR), and single-photon emission computed tomography (SPECT) etc., help physicians to understand the complex anatomy present in the slices. Interpretation of 2D medical images requires special training, and although radiologists have these skills, they must often communicate their interpretations to the referring physicians, who sometimes have difficulty in visualizing the 3D anatomy. Researchers have reported the application of 3D medical images in a variety of areas like acetabular fractures, craniofacial abnormalities, complex bone structures, to name a few.

The literature reported has been organized into the following broad headings:

- Algorithms for triangular and tetrahedral elements.
- Algorithms for rectangular, quadrilateral and brick elements.

- Algorithms using various interpolation techniques.
- Algorithms using various parametric curves and surfaces.
- Algorithms developed by researchers to satisfy a particular application. These type of algorithms are categorized as 'Miscellaneous'.

2.3 ALGORITHMS FOR TRIANGULAR AND TETRAHEDRAL ELEMENTS

Triangular elements in 2D and tetrahedral elements in 3D are the most used elements in contour plotting due to their best fit capability in any domain. A number of researchers have discussed their algorithms based on these types of elements. Mostly the researchers use the linear interpolation over these types of elements. A brief account of such algorithms is given below.

Meek and Beer (1976) developed a method to trace contours through any finite element whether it be rectangular or triangular. The contour was traced through the unit square or triangle and then mapped on to the real surface using the shape functions. The method treated the two-dimensional elements by breaking them up into sub-domains with linear interpolation being used within the sub-domains.

Marlow and Powell (1976) presented a Fortran subroutine to approximate part of a conic by a sequence of straight line segments in order that the approximation was drawn directly by an automatic graph plotter. The main purpose was to use few line segments subjected to an accuracy parameter that was set by the user of the subroutine. All parts of the routine were drawn that were inside a given triangle. The conic was specified by function values at the vertices and at mid-points of the sides of the triangle. The conic section was plotted by the approximation that was composed of straight line segments so that the resultant piecewise linear curve was sufficiently close to the required section of the conic. Much of the calculations were done in homogeneous coordinates.

Gold *et al.* (1977) proposed a method where a map area was automatically divided into suitable triangular domains with a data point at each vertex. In the proposed method the surface estimation and contour plotting were performed independently for each triangle using a measured or best-fit plane associated with each data point. The approach required utilization of three non-original aspects: i) the derivation of a local homogenous 'area' coordinate system for any arbitrary triangle; ii) the construction of a data-structure, linking

each triangular domain with its three neighbouring triangles and three associated data points, and iii) use of a 'conforming' triangular finite-element interpolating function.

The use of first two of these concepts permits the economic generation and optimization of a triangular mesh from a set of data points. The use of first and third concepts permits the interpolation of a smooth surface over the whole map area even though each triangular element is estimated and plotted independently. The requirements for a suitable interpolating function were discussed, and an interpolant was suggested that preserved elevation and slope at each data point as well as elevation and slope continuity between domains. Contour line segments were produced by division of each triangle into N^2 sub-triangles, elevation estimation at each resulting node, and linear interpolation and plotting within each sub-triangle.

Powell and Sabin (1977) described a method of "constructing piecewise quadratic approximations which had the property that, if they were applied on each triangle of a triangulation, then $\emptyset(x, y)$ and its first derivatives are continuous everywhere".

The main conclusion was that it was possible to construct piecewise quadratic functions with first derivative continuity that interpolate to given function values and first derivatives at the vertices of any triangulation. The results were helpful to computer calculations that use approximations to functions of two variables, where it is advantageous to achieve continuous first derivatives and to build the approximation from quadratic pieces. It was also stated that it had an important application in contour plotting.

Sibson and Thomson (1981) introduced a seamed rectangular element, on each of whose sixteen triangular panels the function was quadratic; the element, were internally continuously differentiable, were specified by value and gradient data at its vertices, and by the requirement of linearity of normal component of the derivative along its edges, and had no spare degrees of freedom under the requirements. Its use in conjunction with the Marlow and Powell (1976) quadratic contouring algorithm facilitated the economical production of high-quality contour maps.

On a rectangular grid, the use of this element leads to a sixteen triangles per grid point subdivision, a performance reasonably competitive with that obtained on a triangular grid using the Powell and Sabin (1977) element.

Lyness and Asquith (1983) described an algorithm to "produce piecewise linear contour plots for finite element output". Quadrilateral elements were partitioned to form two triangles.

Curvilinear quadrilaterals were approximated with straight sides which lead to some loss of information on nodal values.

It was shown that the piecewise linear contours produced by the algorithm gave an effective representation of finite element output. The piecewise linear representation was used rather than curve fitting techniques to prevent 'meandering' of contours in coarse mesh regions. Refinement of the finite element mesh was used in regions of high gradient producing contours of greater accuracy than those found in regions of low gradient. Further refinement of the contours, of the eight-noded element, was made by taking an average of the element nodal values at the centroid and dividing these elements into eight rather than two triangles.

Preusser (1984) discussed a method for computing contour lines directly from the polynomial coefficients found for each triangle by solving a nonlinear equation for every point of a line. The lines were computed triangle by triangle so that in principal, a true parallel operation on all triangles is possible.

The main advantages of the proposed method were that it was methodically more direct and more exact as the points plotted were evaluated directly from the interpolation function. It required less memory and in normal cases also less computation time. The characteristics that were regarded as disadvantageous were that the contour lines were created piecewise and the contour lines that did not cross triangle borders were not detected.

Yeo (1984) presented a very fast interactive contour plotting package. "Any finite element mesh was automatically converted into a contour plotting mesh composing of three-node triangular elements". The resulting contours were smooth and accurate provided the analysis mesh was sensible.

Preusser (1990) found explicit formulas for the coefficients of bivariate nonic polynomials with the help of a computer algebra system. A linear system with 55 equations and 45 non-zero right hand sides were solved algebraically. The interpolant was twice differentiable across triangle sides and based on function values and partial derivatives up to fourth order at the nodes. The main difference of the approach was that the coefficients were determined in a more efficient Taylor form which was based on two variables only.

Wiley *et al.* (2003) extracted a contour line (or isosurface) from quadratic elements-specifically from quadratic triangles and tetrahedra. The resulting contour line (or surface) was also transformed into a quartic curve (or surface) based on a curved-triangle (curved-tetrahedron) mapping.

The contour line method was applied to each of the tetrahedron's faces to find the contour intersections. Then, rational-quadratic patches were formed that approximated the contour surface from the contour lines on the faces.

In the bivariate case, a rational quadratic can represent a contour curve exactly since it is a conic section. In the trivariate case, the intersection of the contour surface can be represented with each face exactly. However, the contour surface inside a tetrahedron cannot be represented exactly with a rational-quadratic patch. Some degree of error was inherent in the surface representations because of the chosen patches. An alternative was to tessellate (approximate to) the quadratic tetrahedron with linear tetrahedra and then to extract the isosurface from these linear elements.

To obtain an isosurface with less approximation error one would need to use several linear tetrahedra per quadratic tetrahedron, which is undesirable for two reasons: first, the performance penalty for the tessellation is too high; and second, the amount of data sent to the video hardware would increase. The method guarantees only C^0-continuity between the rational-quadratic patches.

From the literature review discussed above it was found that Meek and Beer (1976) and Lyness and Asquith (1983) used linear interpolation functions on the said domain. All the above researchers except Wiley *et al.* (2003) used linear triangular elements for the contour plotting. Wiley *et al.* (2003) used quadratic elements.

The triangular or tetrahedral elements are also used to approximate the surface for a given number of scattered data points or the slices. This has a very important application in the medical field.

2.3.1 Surface Approximations using Triangular Element

The problem of approximating the surface spanning a given set of 2D or 3D points has many applications in the field of computer graphics, computer vision and medical field. A review of such types of algorithms is given below.

Keppel (1975) described an algorithm for obtaining an optimal approximation, using triangulation, of a three-dimensional surface defined by randomly distributed points along contour lines. The combinatorial problem of finding the best arrangement of triangles was treated by assuming an adequate objective function. “The optimal triangulation was found using classical methods of graph theory”.

Ganapathy and Dennehy (1982) reviewed several solutions to the problem of approximating the surface spanning a given set of 3D points as a polyhedron of triangular faces ('triangulation'). An efficient heuristic for triangulation of the 3D surface formed by spanning a *set* of planar contours was discussed. These contours could have arbitrary geometry, but were limited to be planar.

Ekoule *et al.* (1991) defined a general triangulation procedure that provides solution to a number of limitations of planar contours generated by conventional triangulation algorithms. Some limitations are, incorrect results obtained when the contours are not convex, or when the contours in two successive slices are very different. In the same way, the presence of multiple contours in a slice leads to ambiguities in defining the appropriate links.

A simple heuristic triangulation algorithm was first described which was extended to nonconvex contours. It used an original decomposition of an arbitrary contour into elementary convex subcontours. Then the problem of linking one contour in a slice to several contours in an adjacent slice was examined. A unique interpolated contour was generated between the two slices, and the link was created. Next, a solution to the general case of linking multiple contours in each slice was proposed. Finally, the algorithm was applied to the reconstitution of the external surface of a complex shaped object.

The heuristic general algorithm provided a solution for single-branching of two contours of dissimilar shapes and orientation gave a solution for any number of contours in each slice; and operated automatically without user interaction during the process.

Hamann (1992) discussed a general scheme for computing contours of trivariate data. A method was described that first estimated points on a particular contour, generated a piecewise linear approximation to that contour, and finally used that linear approximation as input for a surface scheme. The surface scheme then yielded a surface which approximated the desired contour.

The whole modeling process was divided into a number of sequence steps. Firstly, a method was given for data reduction. Secondly, the algorithm was presented for obtaining points on a contour considering only the given scattered or rectilinear data points. Triangles were constructed from these contour points as a piecewise linear approximation to a contour. Further, the topological information was generated. Finally, a technique was introduced for estimating the curvature behavior of a surface.

Barequet and Sharir (1996) presented a technique for “piecewise-linear surface reconstruction from a series of parallel polygonal cross sections”. The algorithm used a partial curve matching technique for matching parts of the contours, an optimal triangulation of 3-D polygons for resolving the unmatched parts, and a minimum spanning tree heuristic for interpolating between nonsimply connected regions.

Several techniques were combined to obtain the solution. First, a partial curve matching technique based on geometric hashing was used, adapted from computer for identifying matching contour portions. Then, the matching portions were tiled using a simple ‘merge-like’ advancing rule. Finally, the remaining clefts were identified and employed a minimum spanning-tree technique for simplifying the clefts, followed by a minimum-area triangulation of 3-D polygons in order to fill them. A line-sweep procedure was used to identify the hierarchy of contour nesting in a slice.

The method produced a relatively smooth boundary, due to the contour discretization. In situations where the addition of new vertices was not desired, *e.g.* due to data explosion, the system used the discretization only for the matching step, but the tiling itself and the following minimum-area triangulation were performed on the contours containing only the original points.

Cheng and Dey (1999) revisited a method due to Boissonnat (1988) for surface reconstruction from parallel slices based on Delaunay triangulations. “The costly step of computing the three dimensional Delaunay triangulations was eliminated and instead the surface triangles were computed directly”. A non self-intersecting tiling was automatically guaranteed by these triangulations.

Barequet *et al.* (2003) presented an efficient method for interpolating a piecewise-linear surface between two parallel slices, each consisting of an arbitrary number of (possibly nested) polygons that define 'material' and 'nonmaterial' regions. The method was fully automatic and guaranteed to produce non-self-intersecting surfaces in all cases regardless of the number of contours in each slice, their complexity and geometry, and the depth of their hierarchy of nesting. The method was based on computing cells in the overlay of the slices that form the symmetric difference between them. Then, the straight skeletons of the selected cells guided the triangulation of these cells. Finally, the resulting triangles were lifted up in space to form an interpolating surface.

The algorithm made no prior assumption about the input. It operated on any kind and number of contours, and handled all branching situations and hierarchical structures. It guaranteed to interpolate a valid surface for any possible input, and was intuitive in the sense that it tends to minimize the surface area of the reconstruction. This is because it used an offset distance function to locally decide which contour featured to bind.

From the literature discussed above, it has been found that different researchers have used different techniques for triangulation process *i.e.* Keppel (1975) used Graph theory; Ganapathy and Dennehy (1982) and Ekoule *et al.* (1991) used heuristic triangulation algorithm. In medical field, the contour plotting played a very important role in generating the surfaces from data points like Boissonnat (1988); Hamann (1992); Barequet and Sharir (1996); Cheng and Dey (1999).

2.4 ALGORITHMS FOR RECTANGULAR, QUADRILATERAL AND BRICK ELEMENTS

A number of researchers have used rectangular, quadrilateral and brick elements because of their simplicity. A brief account of such algorithms is given below.

Akima (1974) designed a method for interpolating values given at points of a rectangular grid in a plane by a smooth bivariate function $z = z(x, y)$. The interpolating function was a bicubic polynomial in each cell of the rectangular grid. Emphasis was laid on avoiding excessive undulation between given grid points.

The method was based on a piecewise function composed of a set of bicubic polynomials in x and y; a bicubic polynomial in x and y was a polynomial that had terms $x^{\alpha}y^{\beta}$, where $\alpha = 0, 1, 2, 3$ and $\beta = 0, 1, 2, 3$. Each polynomial is applicable to a rectangular area in the $x - y$ plane bounded by the four straight lines $x = x_i$, $x = x_{i+1}$, $y = y_i$ and $y = y_{i+1}$. Each polynomial was determined by the given values of the function $z(x, y)$ and the values of the partial derivatives $z_x = \left(\partial z/\partial x\right)$, $z_y = \left(\partial z/\partial y\right)$, and $z_{xy} = \left(\partial^2 z/\partial x \partial y\right)$ at the four corner points of the rectangle. The resulting surface was smooth not only at the grid points but also along the line segments that formed the rectangles.

Preusser (1989) described an algorithm plotting contour lines for discrete values z_{ij} given at the nodes of a rectangular mesh. More precisely, it handled the graphical display of curves, which were solutions of the equation $f(x, y) - c = 0$, where $f(x, y)$ were bicubic polynomials defined over rectangles, and *c* the contour levels.

A bicubic Hermite polynomial, $f(x,y)$ was determined for every rectangle of the mesh, interpolating the z_{ij} and the derivatives z_x, z_y, and z_{xy}. The derivatives were optionally computed by the algorithm. The contours found were normally smooth curves. They consisted of polygons approximating intersections with the bicubics. It was also possible to fill the areas between them with certain colors or patterns. This was done with a piecewise technique rectangle by rectangle.

Singh and Sarkar (1990) described a fast algorithm for the plotting of contours using quadrilateral elements. "The contours were generated over each element as accurately as the interpolation functions and joined together using the C^0 continuity of the element". The technique, which was an improvement over the method of Stelzer and Welzel (1987), employed only *m* solutions of the contour equation by dividing either the *r*- axis or the *s*-axis into *m* intervals. Since there was no test performed for a straight line segment or for a branch passing over a small region of element, the total numbers of solutions were not reduced in these special cases. The way the algorithm was designed, it was easy to diagnose these cases resulting in drastic reduction in computation. The algorithm was easy to implement. The contours were represented correctly to the desired accuracy of the displaying media such as graphics terminal and plotter. The increase in time with respect to resolution was linear and did not require steps to be modified. The algorithm for joining of the contour segments over individual elements into complete branches over the entire domain was very useful for faster displays than plotting them on element basis. The technique was however limited to 4-node quadrilateral elements.

Gopalakrishnan and Korttom (1993) developed an algorithm based on the finite element method for contour plotting and interpolation of field data. The method suggested was suitable for both regularly and irregularly distributed data points and especially for output from numerical models. The technique adopted used the bilinear finite element to determine the piecewise smooth surface connecting four data points. The gradients of the piecewise surfaces were then modified at the corners of the quadrilateral elements to produce a globally smooth surface. The main advantage of the present algorithm was its intrinsic ability to recognize structures or other obstructions protruding into the field of analysis.

However, the package had some limitations such as it was not convenient for contouring regionally clustered data. Secondly, the field data could contain points in such a way that not all of them could be connected by a set of quadrilaterals as required by this method. Thirdly,

the application of the package required initial desk work in forming the quadrilateral finite element network and preparing the corresponding data.

Some of the researchers (Sibson and Thomson (1981); Lyness and Asquith (1983)) have even converted the rectangular elements into the triangular elements for contour plotting.

The rectangular or brick elements along with the triangular or tetrahedral elements are also used to approximate the surface for a given number of scattered data points. Such types of applications are very important in the medical field. The Marching Cubes (MC) method (Lorensen and Cline, 1987) played a very important role in such type of applications in 3D.

2.4.1 Marching Cubes (MC) Method

MC method played a very vital role in creating the surfaces from 3D medical data. These surfaces were developed by creating triangle models of constant density surfaces from the given data. Since the MC was proposed, many researchers have focused on extending the basic approach to improve its limitations.

Newman and Yi (2006) gave a survey of the development of the Marching Cube algorithm. The paper's primary aim was to give the development of the algorithm and its computational properties, extensions, and limitations.

A brief account of the MC method and the literature using the MC method is given below:

Lorensen and Cline (1987) presented an algorithm, called *Marching Cubes*, that created triangle models of constant density surfaces from 3D medical data. Using a divide-and-conquer approach to generate inter-slice connectivity, a case table that defines triangle topology was created. The algorithm processed the 3D medical data in scan-line order and calculated triangle vertices using linear interpolation. The gradient of the original data was found, normalized, and used as a basis for shading the models. The detail in images produced from the generated surface models was the result of maintaining the inter-slice connectivity, surface data, and gradient information present in the original 3D data.

There were two primary steps in the approach to surface construction problem. First, locating the surface corresponding to a user-specified value and creating triangles. Then, to ensure a quality image of the surface, the normals to the surface at each vertex of each triangle were calculated.

Marching Cubes uses a divide-and-conquer approach to locate the surface in a logical *cube* created from eight pixels; four each from two adjacent slices. The algorithm determines how

the surface intersects this cube, then moves (or *marchs)* to the next cube. To find the surface intersection in a cube, a one is assigned to a cube's vertex if the data value at that vertex exceeds (or equals) the value of the surface that is constructed. These vertices are inside (or on) the surface. Cube vertices with values below the surface receive a zero and are outside the surface. The surface intersects those cube edges where one vertex is outside the surface (one) and the other is inside the surface (zero). With this assumption, the topology of the surface within a cube is determined, finding the location of the intersection later. The final step in *marching cubes* calculates a unit normal for each triangle vertex which is used for rendering algorithm.

"But the MC method suffers from a tendency to create ill-shaped triangles".

There are a number of algorithms in the literature which were inspired by MC method for the contour plotting algorithm. A brief account of these is given below.

Breitkopf (1998) presented a fast and efficient way to obtain realistic 3D surfaces of constant field value and cross-sectional contour maps of values computed on the finite element mesh using linear interpolation. The technique was equivalent to the reconstruction of 3D forms from sequences of 2D slices used in medical imaging. The algorithm was inspired by the original MC algorithm developed for constant density surface construction from medical data defined on regular cubic grids in computed tomography and magnetic resonance. Marching bricks provided the user with an additional insight into the complex 3D behavior giving straightforward interpretation of the finite element results. The algorithm was illustrated for the basic element forms: tetrahedra, prisms and hexahedra. The more complex forms and higher-order elements could be treated by subdivision into these simple forms. The algorithm worked by determining how the sought surface intersects with a given element and then went to the next element. There were two primary steps in the approach to the surface construction problem. First, the surface corresponding to the user-defined value was located and triangles were created. Then the normals to the surface at each vertex of each triangle were calculated for smoothing, shading and lighting computations for 3D visualization.

Kenmochi *et al.* (1999) solved the topological problem of isosurfaces generated by the MC method using the approach of combinatorial topology. The solution to the problem was to prepare all polyhedral configurations corresponding to all possible cases of point connectivities at the joint. Although this solution enabled to avoid the ambiguity problem if an adequate configuration of triangles were chosen at each joint, it was incoherent from the

viewpoint of the point connectivity; the connectivity of points was considered only at the joint of unit cubes, but not in each unit cube. The approach of combinatorial topology was used to discuss the connectivity of points in a polyhedral surface. Discrete polyhedra were guaranteed not to cause the topological ambiguity.

The problem of ill shaped triangles given by MC method was fixed to some degree by dual contouring given by Ju *et al.* (2002), which also provided adaptive contouring and an elegant means of preserving sharp boundaries. The method for contouring a signed grid whose edges were tagged by Hermite data (i.e., exact intersection points and normals) was described. The method avoided the need to explicitly identify and process 'features'. Using a numerically stable representation for quadratic error functions, an octree-based method was developed for simplifying contours produced by this method.

Nielson (2004) presented the definition and computational algorithms for a new class of surfaces which were dual to the isosurface produced by the widely used MC algorithm. These new isosurfaces had the same separating properties as the MC surfaces but they were comprised of quad patches that tend to eliminate the common negative aspect of poorly shaped triangles of the MC isosurfaces. Based upon the concept of the new dual operator, a simple, but rather effective iterative scheme was described for producing smooth separating surfaces for binary, enumerated volumes which are often produced by segmentation algorithms. Both the dual surface algorithm and the iterative smoothing scheme were easily implemented.

The dual surface was obtained, first by extending the concept of the MC surface to a patch version which eliminated the edges of the MC surface interior to voxels to obtain a surface with polygon bounded patches where each vertex was included in exactly four patches. Then for each patch there was associated a vertex of the dual surface. The dual consisted of quad patches which were connected in exactly the same manner as the connectivity of the vertices of each patch.

Jin *et al.* (2006) improved the MC method by redefining the surface configurations. Applied to the surface rendering, the MC method showed a great promise for surface extraction by using the surface configurations of a cube. In surface rendering procedures, the volume data was first partitioned into cubes, and then the algorithm decided the surfaces of each cube according to 15 surface configurations. But in practical applications, the MC method showed several disadvantages. First, during the surface combination of every two adjacent neighbor

cubes, it may produce a wrong surface for the cubes due to more than one choice of the surface configuration. Second, it may produce some holes for the generated surface during connections of the triangles.

The MC algorithm doesn't distinguish between the vertices whose data value exceeds or equals to the value of the surface reconstructed, and considered these vertices to be inside the surface. When the coordinates of the intersected points were calculated using interpolation, approximate coordinates for points on the iso-surface were obtained. Therefore, some separate surfaces were produced to make the MC algorithm weak. For the improvement over marching cubes method, *zero* or *one* was assigned to these inside vertices according to the positions of the inside vertices, and redefined the 15 cases of configurations.

Because the MC algorithm did not distinguish the vertices, which were in the surface and on the surface and considered as inside vertices entirely. Therefore, when using interpolation to calculate the intersected points of the reconstructed surface with the edges of the cube, the approximate interpolated points were obtained. This configuration operation produced some redundant and separate triangles, and further led to the appearance of the holes. Considering the relations of the inside vertices, which were composed of the vertices in the surface and on the surface, a new configuration scheme was proposed.

The first step was similar to that of the MC algorithm. The volume data was partitioned into cubes. Then a logical cube got recurred to the division of the inside and the outside vertices. Every logical cube was corresponding to a case of the configurations. The number of configurations were limited to 15 patterns due to the reverse and symmetric properties of cube. In the second step, the configurations for the 15 cases were re-specified. Reconsidering the inside vertices in a logical cube, a second index was created to search for the configurations produced. For the inside vertices of the logical cube, the second index assigned one to the vertices that were on the surface and assigned zero to that in the surface. On this condition the algorithm looked for the configurations that were redefined based on the second index. Using the second index for the configurations the amount of the separate surfaces were decreased to reduce the probability of the hole appearance.

Bargteil *et al.* (2006) presented a semi-Lagrangian surface tracking method for use with fluid simulations. The method maintained an explicit polygonal mesh that defined the surface, and an octree data structure that provided both a spatial index for the mesh and a means for efficiently approximating the signed distance to the surface. At each timestep, a new surface

was constructed by extracting the zero set of an advected signed-distance function. Semi-Lagrangian backward path tracking was used to advect the signed-distance function. One of the primary advantages of the formulation was that it enabled tracking of surface characteristics, such as color or texture coordinates, at negligible additional cost. The MC method was used due to its simplicity, robustness and speed. Each cube was divided into six tetrahedra to simplify the implementation. The choice of MC method does result in creating poorly shaped triangles.

Nielson (2008) discussed the Dual Marching Tetrahedra (DMT) method. The DMT was viewed as a generalization of the classical cuberille method of Chen *et al.* (1984) to a tetrahedronal.

The cuberille method produced a rendering of quadrilaterals comprising a surface that separated voxels deemed to be contained in an object of interest from those voxels not in the object. A cuberille was a region of 3D space partitioned into cubes. A tetrahedronal was a region of 3D space decomposed into tetrahedra. The DMT method generalized the cubille method from cubes to tetrahedra and corrected a fundamental problem of the original cuberille method where separating surfaces were not necessarily manifolds. For binary segmented data, a method was proposed for computing the location of vertices that was based upon the use of a minimal discrete norm curvature criterion. The method was extended to the case where there were dependent function values given at every grid point.

The implementation of the algorithm was stated to be considerably easier and less tedious than the standard MC or Dual MC method due to the differences in complexity of the algorithm.

From the above it is observed that the researchers have improved the MC method for accuracy.

2.5 ALGORITHMS USING VARIOUS INTERPOLATION TECHNIQUES

Researchers have used various mathematical and isoparametric interpolation techniques for contour plotting for various applications. Mostly linear interpolations have been used by different researchers. An account of such review is also given in section 2.2 where these techniques have been used on triangular or tetrahedral elements. Table 2.1 provides summary of such literature.

Technique	Literature
Shape Functions (Linear Interpolation)	Meek and Beer (1976)
Curve converted to straight lines	Marlow and Powell (1976)
Interpolation function (Linear)	Gold *et al.* (1977)
Interpolation function	Preusser (1984)

Table 2.1 Interpolation techniques used in triangular elements

2.5.1 Mathematical Interpolation Techniques

A brief literature using various mathematical interpolation techniques is given below.

McLain (1974) described a computer method for drawing, on an incremental plotter, a set of contours when the height is available only for some arbitrary collection of points. The method was based on a distance-weighted, least-squares approximation technique, with the weights varying with the distance of the data points. It is suitable not only for mathematically derived data, but also for data of geographical and other non-mathematical origins, for which numerical approximations are not usually appropriate.

Sutcliffe (1976a) described an algorithm which traced the curve $f(x, y) = 0$, in a region over which there was a method of calculating *f*, using a series of straight lines of length one and two times the step size assumed for the graph plotter or display unit. It was done by determining the sign of the function on the region of the curve and plotting a path between positive and negative values. It was also suggested that by means of a suitable interpolation formula the algorithm may be used for contour plotting over a grid of values.

Sutcliffe (1976b) described an error in the contouring algorithm of Mclain (1974) and suggested the method to avoid such error.

In the contouring algorithm of McLain (1974) an error aroused when the contour passed the mouth of certain valleys. This had the effect that the algorithm could trace the bottom of the valleys rather than the true contours and so produced unwanted false contours. The result of the error was that under some circumstances the algorithm drew a curve which was not a part of a contour.

The corrected method rightly accepts points which were astride the contour and correctly rejected the points that lied across the bottom of a valley if it was reasonably shallow.

However, it tended to be rather insensitive to picking up valleys if one (or both) sides are steep and could trace a valley a long way before detecting it.

Faber *et al.* (1979) gave a review of possible computer methods to plot contour maps for data available on the mesh-points of a regular grid. Thereafter the details were given of an improved method which plots one contour at a time. The method of 'drawing one contour at a time' is particularly suitable when the total surface can be described by an analytical equation, but it can be adapted for data in grid form. It then requires a repetitive search technique to create strings of numbers that represent coordinates of successive points of the contour.

In principle, the 'triangulation method' was modified for 'drawing one contour at a time'. However, the points situated on the diagonals of the gridunits turn out to affect the smoothness of the contours unfavourably compared to curves constructed by using the locations on the gridlines only. Fortunately, the required sequence of points on the gridlines can be obtained by a repetitive search technique from those points alone.

An outstanding advantage of 'drawing one contour at a time' was that all points that constitute a contour are known prior to the drawing of the contour. Smooth curves were then constructed without using explicit analytical expressions for the surface. Dotted or broken curves were easily produced and automatic annotation of the contours was feasible. The scheme discussed consisted of three stages: (a) 'locating' (interpolation of contour locations on the grid lines, (b) 'chaining' (forming chains of successive points), (c) 'smoothing' (drawing smooth curves from weighted averages of two circle segments).

Preusser (1986) presented the principal ideas of the Trip Algorithm, which finds closed polygons for filling the area between contour lines. These polygons consist of points P defining the contours, intersections S with the boundary of the domain, and vertices V of the domain. The points P were computed successively by a nonlinear method that combined extrapolation and the regula falsi in order to adjust the distance between the points to the curvature of the contour. For the extrapolation, derivatives of Lagrange polynomials were used. Empirical parameters for the automatic step size control were given. Once the points S were determined, the method was independent of the type of the interpolation function $f(x, y)$. Two examples for applications of the Trip Algorithm were presented: one from scattered data interpolation, the other from stress analysis by finite element methods.

Table 2.2 gives a summary of various mathematical techniques used by the different researchers.

Technique	**Literature**
Distance-Weighted, Least-Squares	McLain (1974)
Weighted Averages	Faber *et al.* (1979)
Extrapolation, Regula Falsi	Preusser (1986)

Table 2.2 Different mathematical techniques used by researchers

2.5.2 Shape Functions or Isoparametric Interpolation Techniques

Shape functions or isoparametric interpolation techniques are used by different researchers for contour plotting for finite element data.

Akin and Gray (1977) presented a "general non-linear interpolation procedure for contouring on any isoparametric surface". The method was based upon standard isoparametric interpolation functions and a predictor method for tracing element contour lines.

Gray and Akin (1979) presented the method which extended the previous work (Akin and Gray, 1977) by using a predictor-corrector method to trace contour lines, thereby making the contouring algorithm more accurate.

Kontopidis and Limbert (1983) described a contouring algorithm for isoparametric elements which can be used to map three-dimensional scalar fields. The contours were generated on arbitrary planes intersecting finite element structures. Tracing element contour lines were accomplished by an accurate predictor-corrector technique. A method of finding starting points for the algorithm on the boundary of the elements was also given.

The algorithm was the extension of the method given by Gray and Akin (1979). The method gives predictor equations which used a point that belongs to the desired contour and found a new point in the direction of the contour line. The corrector equations used this result to find a new point of the contour line by 'taking a step' perpendicular to the contour, as in Gray and Akin (1979). The algorithm had a recursive form and required starting points in the cut plane at the desired contour values. These points were determined by finding the intersection of the boundaries of the finite elements with the cut plane. Then, the locations of the desired contour values along this line of intersection were found.

Rajasekaran (1984) suggested that it was better to use the 'Iso-parametric type of representation' of shape function to plot the curves rather than simply using the usual interpolations such as Newton's Divided Difference, Lagrangian Interpolation or Least Square Fitting. It is due to the reason that the interpolation of a polynomial function behaves very badly near the extremities of the region of definition.

Akin, Gray and Zhang (1984) presented a procedure for displaying variables in colour on an isoparametric surface. It utilized the isoparametric interpolation functions to produce continuous colour variations on the surface.

Stelzer and Welzel (1987) introduced a simple and straightforward scheme for plotting contours. The presented method has some common roots with the work of Meek and Beer (1976), however, neither subelement regions were used nor linear variations of the variable in the subregion. Instead, the contours were calculated without any approximation disclosing the information contained in the shape functions. The method was straightforward. The contours were smoothly bent. Thereafter the paper dealt with treatment of contour discontinuities across element borders which frequently occurred with stresses and heat fluxes.

Singh (1990) described some aspects experienced during the numerical experimentation. The contour line algorithm given by Stelzer and Welzel (1987) was incorporated because of its exactness and its convincing straightforwardness. Although Stelzer and Welzel (1987) dealt with both linearly interpolated isoparametric elements and quadratic ones but linear type were only used to keep the relationships simple.

Twu and Wang (1992) developed a simple method to get a contour plot in an element with given nodal values. The contour lines were obtained by solving a differential equation obtained from the shape function using symbolic computation. This simple and straightforward method moved stepwise along one axis of coordinates and computed the coordinates of the contour point on the other axis within an element by solving an algebraic equation. Akin and Gray (1977) developed a method using local derivative values to trace the locus of a contour line. But that method depended on the assumed size of step increment and error accumulation could occur. The method developed by Twu and Wang (1992) was similar to that of Akin and Gray (1977) approach but without the need to assume the step size. Both open and closed contours could be constructed. It could also be applied to three-dimensional elements if only the contour lines on the surface of element were desired.

Rajasekaran and Venkatesan (1995) developed an algorithm to obtain the contour plot of principal scalar, vector and tensor valued fields in areas of engineering. The isoparametric style of interpolation was used to draw the contour map in two dimensions and the contour surface in three dimensions. The three dimensional shaded view was also simulated using the programs. Since linear interpolation was used, the domain was divided into smaller elements.

2.6 ALGORITHMS USING VARIOUS PARAMETRIC CURVES AND SURFACES

Some researchers have used the synthetic curves and surfaces for drawing the contour lines or surfaces. These curves and surfaces were used in the parametric form or other parametric representations were used to make the calculations simpler. A brief account of such literature is given below.

Farin (1986) discussed the theory related to the Triangular Bernstein Bezier Patches. It was also stated that the concept can be applied to contouring. It was also suggested that one should use rectangular patches whenever possible since they can be computed quickly and without the need for a complicated data structure. In those cases where rectangular patches were too tedious to use, triangular patches should be employed. They may be more CPU-intensive, but the expense incurred should weigh little compared to cost of (human) efforts to design complex surfaces with insufficient tools.

Dickinson *et al.* (1989) described a system for the interactive editing and contouring of surfaces derived from empirical fields. The approach taken began with the representation of a field as a general-order, nonuniform, tensor-product, B-spline surface. It provided an interactive display for editing the surface by control-vertex manipulation and a contouring algorithm that was specifically designed for the fast and robust contouring of B-spline surfaces. Interactive editing of the resulting model was feasible because of the local nature of editing changes when B-splines were used. The use of nonuniform B-splines gave the flexibility required to model highly irregular data efficiently.

Worsey and Farin (1990) considered the problem of contouring a bivariate quadratic polynomial, defined as a triangular Bezier patch. The important cases where the contour had disjoint sections or was a closed curve were considered in detail. The results led to a simple, yet completely robust algorithm for describing contours as piecewise rational quadratic Bezier curves.

The algorithm developed for solving the problem was said to be more efficient, accurate and robust than that of Marlow and Powell (1976) and Farin (1986).

Berry (1990) used a complete bi-quadratic parametric function to map nodal resultants, and geometry, over the finite element parametric space. The parametric functions were then used to determine contour lines representing the element resultants. The algorithm was developed in C-base and was applicable to 2-D and 3-D linear and quadratic elements. In the method used, the accuracy was independent of the element size or its geometric position in 3D space. It was due to the parametric space mapping algorithm as opposed to a physical space-mapping algorithm.

Grimm and Hughes (1995) presented a method for approximating an isosurface with a smooth parametric representation.

Modeling shapes via an isosurface of a three dimensional data set is an efficient and natural method for specifying the topology and basic geometry of an object but it can be difficult to control fine surface details. Parametric surfaces e.g., B-splines provide good detail control, but creating topologically arbitrary surfaces with them is difficult. A modeling paradigm was proposed wherein an artist creates a sketch of an object using a 3D paint program such as Sculpt from which a smooth parametric representation of an isosurface is constructed, at some level of fidelity. Further adjustment of the surface detail was accomplished using the parametric representation.

Hamann *et al.* (1997) discussed the construction of an approximation of a contour using triangular rational-quadratic surface patches. The approximating rational-quadratic surfaces were represented in triangular rational Bezier form as was done by Farin (1986). The primary motivation for the scheme was to model or postprocess an approximation of a contour of a trivariate function in terms of parametric patches. The approximation of a contour by parametric surface patches was also done by Grimm and Hughes (1995). The construction described yielded patches with three, four, five, and six boundary curves. The topology at vertices was characterized by the fact that exactly four patch boundary curves meet at a shared patch corner vertex, (*i.e.*, each patch corner vertex has valence four). Furthermore, not all vertices produced by a Marching Cube (MC) algorithm were interpolated by the method given by Grimm and Hughes (1995). The algorithm given by Hamann *et al.* (1997) did not require certain valences for patch corner vertices, and it interpolated all the points generated

by an MC algorithm. The algorithm reduced the number of triangular patches, based on the idea of identifying patches in (nearly) planar regions.

Table 2.3 gives a summary of the different parametric curves and surfaces used by the different researchers.

Technique	**Literature**
Triangular Bernstein Bezier Patch	Farin (1986)
B-Spline Surface	Dickinson *et al.* (1989)
Triangular Bezier Patch	Worsey and Farin (1990)
Bi-Quadratic Parametric Function	Berry (1990)
Triangular Rational-Quadratic Patch	Hamann *et al.* (1997)

Table 2.3 Different parametric curves and surfaces used by researchers

2.7 MISCELLANEOUS

Apart from all the methods discussed above, a number of researchers have developed their own contouring methods for a particular application. These all are categorized under the 'Miscellaneous' category.

Stineman (1967) developed a method for constructing an approximate plot of a function of three independent variables. The plot was similar to a conventional contour map except that there were three scales to represent the independent variables. Scale values of the three independent variables were added vectorially, and the value of the function was then read from the values associated with nearby contours. A computer program was coded in Fortran to generate the contour map.

Morse (1969) developed a generalized technique to simplify the solution of problems relating to contour maps. The techniques made use of the topological properties of contour maps. The topology was represented by a graphical structure in which adjacent contour lines appeared as connected nodes. A mathematical model for the analysis of contour map was developed. The model defined contour lines in an abstract sense and took into account the presence of cliff lines but excluded overhanging cliffs. A study of the model led to the conclusion that in certain cases contour lines were nonintersecting closed curves. These properties were exploited to introduce interior and exterior contour lines. An investigation of the relationships

between adjacent contour lines resulted in a graph that displayed these relationships. The topological properties of the contour map were found to be related to the properties of the graph. As the graph contained only a part of the information found on the contour map, the solutions of problems were simplified by examining the graph before examining the entire map. Some non topological properties were considered. In particular, the geometrical properties involving straight lines drawn on the contour map were investigated. As a result of the investigation of the geometrical properties, boundaries were obtained for the location of straight lines satisfying specified conditions.

Snyder (1978) made a subroutine, GCONTR which determined sequence of points in the plane which were used to draw contours through equal values of the surface. The contours were constructed from some starting point until they close or intersect a boundary, or by examining each cell of the grid in turn and drawing all contours found inside the cell. The contour labeling was relatively easy in the algorithm. The contour found inside a cell requires less auxiliary storage. Each cell was completely processed before going on to the next, thereby allowing generation of contours over a much larger array than was accommodated in main memory at one time. GCONTR used linear interpolation to estimate the point at which a contour value intersects the edge of a cell. The nonlinear interpolation was not used as there was the possibility of multiple intersections of a contour with an edge and the possibility of closed contours which intersect no edges. Linear interpolation has the disadvantage that extrema occur only at nodal points and it does not provide smooth and accurate contours. The second disadvantage can be removed by computing more function values or interpolatory subtabulation to a finer mesh. If the required storage is larger than desired, the independent variable plane was easily divided into separately processed segments. GCONTR did not provide automatic subtabulation because different methods work better for different problems. Since GCONTR used a contour following method, any implementation of automatic subtabulation had a higher cost than a reasonable user implementation.

Wright and Humbrecht (1979) presented the algorithm which drew an approximation of the surface or surfaces where the function attains a specified value for a given three-dimensional array containing values of a function of three variables. This was done by contouring two-dimensional subsets of the three-dimensional array and suppressing the invisible parts of the contours. The union of all the contour line parts approximated the desired surface.

Antoy (1983) provided a fortran subroutine for determining the contour lines of a function specified on a mesh made by a set of irregular profiles. The subroutine was independent of

the support medium holding the mesh, thus it was possible to work on a large set of data with a small computer.

Steven (1984) showed how a smoothing scheme based on a minimum surface theory significantly improved the quality of the graphical results. The scheme proposed was compared with many other smoothing schemes, including least squares, on a simple problem. However, the technique would break down at boundary points since information was not always available on the nature of the quantity being smoothed or fitted at that boundary. Since the points being fitted did not fall within a closed curve of element centriodal nodes, the minimum surface theory use to break down.

Zyda *et al.* (1987) presented an expanded algorithm that not only handled the mappings of multiple contours per plane and partial contour mappings, but also allowed human interaction to resolve mapping problems. A discussion of the algorithm's limitations and the proposed solutions to the limitations was also presented.

Yates (1987) described the contouring algorithm which allowed to produce high-quality two-dimensional contour diagrams from values of a dependent variable located on a uniform grid system (i.e. spacing of nodal points in x and y directions was constant). Computer subroutines (supplied) were developed and tested which carried out the computations. Output from the subroutine was in the form of a list of coordinates for each specified contour level which was used in a standard line-plot program to generate user-tailored contour diagrams.

Zyda (1988) presented a study of a highly decomposable algorithm useful for the parallel generation of a contour surface display. The core component of the algorithm was a two-dimensional contouring algorithm that operated on a single 2 x 2 subgrid of a larger grid. An intuitive procedure for the operations used to generate the contour lines for a subgrid was developed. A data structure, the contouring tree, was introduced as the basis of a new algorithm for generating the contour lines for the subgrid. The construction of the contouring tree was detailed. Space requirements for the contouring tree algorithm were described for particular implementations.

The first step in the procedure was to determine whether any contours should be generated for the subgrid. That determination was based on whether any of the subgrid’s edges contained the desired contour level k. An edge contained a contour level k if the contour level was within the range of values defined by the grid points’ edge.

The next part of the procedure was the computation of the contour edge intersections for any subgrid edges shown to contain the contour level. The point of intersection was computed through linear interpolation, using grid values assigned to the endpoints of the edge and their corresponding coordinates. The point of intersection represented the location on the subgrid edge corresponding to the contour level, *k*.

Sewell (1988) presented a technique for the "graphical representation of some contour (level) surfaces of a function of three variables defined by its values on an array of points (x_i, y_j, z_k) ". The algorithm involved drawing and projecting the contour curves corresponding to the cross sections $x = x_i$ and $y = y_i$. The computer program presented, CON3D, visually represented a function of three variables in a single plot by drawing several contour surfaces.

Because the curves were drawn as thick, opaque bands, which mask all bands in the immediate background, a sense of depth was retained, yet background contours were still visible between the bands.

Sergio and Ali (1996) developed a contouring method by finding where a contour crossed the area bounded by the known data (interpolation). Once the location of the contour intersections with the boundaries of a grid node was found using linear interpolation, a isoline was formed by connecting these two intersecting points by a straight line segment. Since the computations were based on the edges of each node, there was no possibility for discontinuities or contour gaps.

Lima and Soriano (1997) presented the stress representation by coloured patterns and contours through the use of the finite element method. Stresses were computed at the center of subregions of a network for every two-dimensional element, or face and cross-section of three-dimensional element to be viewed. Therefore, the method must be used in conjunction with methods for view representations.

Computed subregions stress defined a matrix of colour codes which were used to 'paint' the corresponding subregions. Two different adjacent codes in that matrix indicated that a contour passed between the corresponding subregions and were used to generate contours. As that coloured pattern required stress computation for the subregions, and no additional stress computation was needed, the generation of contours was considered natural and simple.

The method could fail in the two following cases:

(i) When part of the contour coincides with the element boundary. This occurrence identified, the corresponding segment of the contour may be plotted.

(ii) When a contour is circumscribed by the element boundary. To prevent this occurrence, which seldom occurs with linear and quadratic elements, the method may be applied for some subelements of every element.

Grandine (2000) presented several practical applications of the univariate contouring problem, and it demonstrated how these problems could be solved accurately and reliably as numerical solutions of a particular differential algebraic equation derived for that purpose.

Strain (2001) presented a new approach to numerical methods for general moving-interface problems. A semi-Lagrangian advection formula was contoured, evaluated with efficient geometric algorithms, to extract the moving interface. A fast contouring technique controlled the interface resolution independent of the time step, and grid-free adaptive refinement increased accuracy by orders of magnitude. Semi-Lagrangian advection merged and broke complex topology, with stable time stepping independent of the interface resolution, while fast geometric algorithms resolved an N-element interface with optimal $O(N \log N)$ efficiency. The implementation provided fast new intrinsic first-order and embedded second-order geometry evaluation modules for solving specific moving-interface problems.

Owada *et al.* (2003) proposed an algorithm that enumerated all possible cases of the correspondence of contours. It was useful for achieving fully automatic interpolation of contours, although the implementation still required some degree of manual interaction.

When polygonal meshes were constructed from sparse cross-sectional contours, each interspace between adjacent cross-sectional planes had to be filled by interpolation. If the shape of the object was simple, the interpolation was trivial. However, handling real-world medical data, such as the human skeleton or brain, became difficult as the spacing between slices increased when the number of contours in adjacent cross sections differed. Although the topological structure was important, a user's knowledge was not limited to topology. The topology of an object described just one part of the full shape information-the skeleton information. Other knowledge may be specified by means of the integral of the surface curvature, the volume of the object, or its interference with other objects' surfaces, but these

quantities cannot be calculated and compared with possible cases until the surface has been reconstructed. However, the strategy of first determining the topological structure, then constructing the surface, and finally optimizing the shape to match user's knowledge tends to find local minima, because the optimization is usually performed to improve shape of the surface, rather than to change the topology. Even if topological change is allowed, without scanning every possible case, it is hard to find an answer that is sufficiently close to the global optimum, because topological changes strongly and unpredictably affect the above-mentioned quantities. The proposed method enumerates all possible topological structures. A polygonal mesh was constructed for each result in the enumeration. The algorithm formed a basis for calculating the global optimum, namely the shape that best matches the user's knowledge.

Bradbury and Enright (2003) investigated the application of Differential Equation Interpolants (DEIs) to the visualization of solutions to Partial Differential Equations (PDEs). In particular, it was described how a DEI was used to generate a fine mesh approximation from a coarse mesh approximation; this fine mesh approximation was then used by a standard contouring function to render an accurate contour plot of the surface. However, the standard approach had a time complexity equivalent to that of rendering a surface plot, $O(fm^2)$ for each element of the coarse mesh (where fm is the ratio of the width of the coarse mesh to the fine mesh). To address this concern, three fast contouring algorithms were proposed that computed accurate contour lines directly from the DEI, and had time complexity at most $O(fm)$ for each coarse mesh element.

MATLAB has been used to implement the fast contouring algorithms because it provides a convenient environment for prototyping numerical algorithms. If they are to reach their full potential, and be applied in applications involving high performance simulations, the fast contouring algorithms can be rewritten in a procedural language such as C instead of the slower interpreted language of MATLAB.

Jaworska (2006) presented a new method of creating contour lines based on irregularly spaced data. The analysis was focused on contour plots as a useful tool of visualization and real-time verification of the postprocessing stage in multigrid computational environment. Given geometry of the considered domain was used to prepare an initial mesh, which was relatively coarse, and to establish the base for the adaptative procedure. Depending on the solution method, each mesh had to contain different topological information. After calculation, an a'posteriori error analysis was carried out. The result of this analysis was used

to prepare a list of new nodes, which were added to the mesh. As a consequence of new nodes appearing, mesh topology information was updated.

The tool could visualize all the mesh related information for every step of the generation process, allowed for interactive modification, dynamic presentation of the series of meshes and the solution during the adaptation process. The software could work on-line providing instant visualization of intermediate as well as final results. The approach was based on OpenGL and its toolkits GLUT, GLUI.

2.8 OBSERVATIONS BASED ON THE REVIEW

From the detailed review discussed above it has been found that the simplest way to plot contour on 2D domain is to use linear interpolation using triangular elements. Even quadrilateral elements were partitioned into triangular elements for contour plotting. The use of linear interpolation works well if accuracy of higher order is not required. This will result into loss of information in contour plotting even when FEA is done accurately using higher order elements.

It has been found from literature review that very less amount of work is available on contour plotting using quadratic 2D elements. The contour plots on quadratic elements are very important so that accurate results can be plotted without loss of any information. In order to simplify the contour surface generation, generally a higher order element is split into linear elements and linear interpolation of the physical quantity is performed. The linear approximation of a non-linear function creates inaccuracies in the results.

Similarly, there is very little literature dealing with contour plot algorithms using higher order 3D elements. To make the overall task simple, the analysis is performed with higher order elements but the graphical display of results in the form of contour surfaces is carried out using linear elements.

Further, there is even scope of increasing the speed of existing algorithms to make the plotting algorithms fast.

2.9 CONCLUDING REMARKS

The detailed literature review has given an insight of various algorithms used for contour plotting in different application areas.

In medical field, the triangulation process plays a very important role to approximate the surface for a given number of scattered data points or the slices. The Marching Cube (MC) method (Lorensen and Cline, 1987) has played a very important role in such type of applications in 3D.

In Engineering, contour plotting algorithms are mainly used for graphically displaying the output results of various analysis tools, FEA being the front runner. The contour plot algorithms used in FEA are mainly developed in accordance with the type of elements used for analysis. The various methods involving different mathematical interpolation techniques, shape or isoparametric interpolation techniques and parametric curves or surfaces, to name a few, are being used for contour plotting.

The next chapter describes the contour plotting algorithms used for linear 2D elements.

CHAPTER - III

CONTOURING IN 2D USING LINEAR ELEMENTS

3.1 INTRODUCTION

This chapter discusses in detail the contouring in 2D using linear triangular and quadrilateral elements which are more commonly used in FEA.

The triangular elements are popular in FEA because they fit better in an irregular domain as compared to other elements. It is also a common observation that four-node quadrilateral elements perform much better than three-node triangular elements, especially when the discretization is not dense.

The simplest way to plot contours on 2D domain is to use linear interpolation over triangular elements. When the triangular scheme is not available and only data points and values of the physical parameter on the given data points are available, we first resort to Delaunay triangulation and then perform linear interpolation. This method works well if accuracy of higher order is not required. Even the quadrilateral elements are partitioned into triangular elements for contour plotting.

In the present chapter, simple, accurate and fast contour plotting algorithms are developed over triangular and quadrilateral elements.

3.2 CONTOUR EQUATION OVER TRIANGULAR ELEMENT

As stated earlier, a three-node triangular element, shown in Fig. 3.1, which interpolates the function linearly over the element, is used mainly for its simplicity for fast contouring.

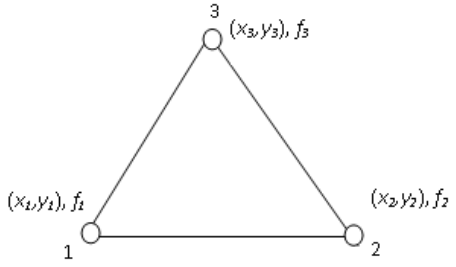


Figure 3.1 A three-node triangular element

The coordinates of the three vertices of the triangle are available prior to analysis and the physical quantity, *f(x,y)*, is known after analysis, *i.e.* after solving a mathematical model. The linear interpolation of coordinates and a function over the element is performed as expressed in the following equations:

$$x = \sum_{i=1}^{3} N_i x_i \tag{3.1}$$

$$y = \sum_{i=1}^{3} N_i y_i \tag{3.2}$$

$$f(x, y) = \sum_{i=1}^{3} N_i f_i \tag{3.3}$$

A contour in 2D is a continuous curve characterized by Eq. (3.4).

$$f(x, y) = C \tag{3.4}$$

From Eqs. (3.3) and (3.4),

$$\sum_{i=1}^{3} N_i f_i = C \tag{3.5}$$

where x, y are spatial coordinates, C is an contour level, a constant, $N_i = L_i$ and

$$L_i = \frac{1}{2\Delta}\left(a_i + b_i x + c_i y\right) \tag{3.6}$$

The coefficients a_i ,b_i and c_i are cofactors of the matrix, M, as represented in Eq. (3.7).

$$M = \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix} \tag{3.7}$$

and, $\Delta = \det(M)$. The functions $L_i = L_i(x, y), \quad i = 1,2,3$ are called shape functions.

After substituting the values of shape function from Eq. (3.6) into Eq. (3.5), the Eq. (3.5) is simplified to

$$l_0 + l_1 x + l_2 y = C \tag{3.8}$$

where,

$$l_0 = \frac{1}{2\Delta}\left(a_1 f_1 + a_2 f_2 + a_3 f_3\right)$$

$$l_1 = \frac{1}{2\Delta}\left(b_1 f_1 + b_2 f_2 + b_3 f_3\right)$$

$$l_2 = \frac{1}{2\Delta}\left(c_1 f_1 + c_2 f_2 + c_3 f_3\right)$$

Eq. (3.8) can be compared with the following general equation which represents a line

$$F(x, y, z) = ax + by + c \tag{3.9}$$

The problem now reduces to accurately trace the contour represented by a line over a triangle. The high speed line algorithm like mid-point algorithm (Hearn and Baker, 2002), which works on integer arithmetic is used to trace the line.

3.3 CONTOURING ALGORITHM

The overall algorithm for contouring is described as follows:

(i) Consider an element.

(ii) Find the minimum and maximum, f_{min} and f_{max}, of $f(x, y)$ over the element.

(iii) If $f_{\min} \le C \le f_{\max}$ then go to step (iv), else go to step (vi).

(iv) Trace the contour over the element.

(v) Save the contour.

(vi) If not all the elements are processed, then consider the next element and go to step (ii), else go to step (vii).

(vii) Join the contours of the same level.

(viii) Plot the contours.

Except for step (iv) all steps are easy to understand. Detailed explanation for this step is given in the following section.

Step (iv) Tracing contour line over a triangle element

The contour line is traced over the triangle element by finding the possible intersection of the line with the edge of the triangle.

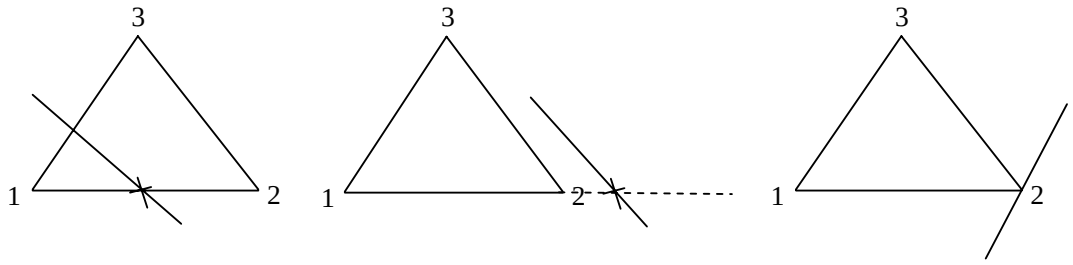


Figure 3.2 Possible cases of intersection of a side with nodes 1 and 2 with the line a) The line intersects at one point of the side; b) Intersection point is outside the side; c) The line touches the corner point

The intersection of a line with one of the sides of the triangle has three possible cases as shown in Fig. 3.2(a) through Fig. 3.2(c). In the second case, the contour line lies outside the element. The last case, if detected, is discarded for contour plotting as it yields a single point which is of no interest.

In general, the intersection of a line with an edge of a triangle is found out using parametric equations (Zeid, 2007) and the point of intersection, if any, is recorded. Let the parametric equation, in vector form, of an edge represented by end points P_1P_2 be

$$P = P_1 + u(P_2 - P_1) , \qquad u \in [0,1] \tag{3.10}$$

And the general equation of the contour line be

$$l_1 x + l_2 y + l_3 = 0 \tag{3.11}$$

On substituting *x* and *y* from Eq. (3.10) into Eq. (3.11) and rearranging, we get

$$u = \frac{(l_1 x_1 + l_2 y_1 + l_3)}{(l_1(x_1 - x_2) + l_2(y_1 - y_2))} \tag{3.12}$$

The intersection point lies between P_1 and P_2 if the value of *u* lies between 0 and 1.

Similarly, the intersection points on all the edges are calculated and joined with the help of a line to form a contour line over an element.

3.4 CONTOUR EQUATION OVER QUADRILATERAL ELEMENT

As stated above, the performance of four-node quadrilateral element is much better than three-node triangular element. A number of researchers have worked on contour plotting using shape or interpolation functions over quadrilateral element. The scheme given by Meek and Beer (1976) is based on subdividing an element into a number of subregions

and then carrying out linear interpolation for contour points. The scheme presented by Akin and Gray (1977) used a starting contour point over an element and the line is moved in the direction of the tangent to the contour with a predefined step length. This technique is applicable to any element but causes discontinuities at element interfaces, particularly with C^0 elements, and in general the contour line deviates from its correct position. Stelzer and Welzel (1987) developed a simple and straightforward scheme for four-node isoparametric element.

The present section reproduces the algorithm given by Singh and Sarkar (1990), as it forms the basis of the thesis. The algorithm uses a technique where the contour lines are as exact as the finite element mesh and the speed is greatly enhanced compared to other methods.

It is assumed that the domain under consideration is discretized into four-node quadrilateral elements and the function values at all the nodes are known. The quadrilateral elements are mapped onto a square $[-1,+1][-1,+1]$ in the *r*-*s* co-ordinates systems (Figs. 3.3(a) and 3.3(b)) and the function is interpolated linearly using the transformation

$$f(x,y) = \sum_{i=1}^{4} N_i f_i \tag{3.13}$$

where f_i is the function value and N_i is the shape function for the i^{th} node. The shape function, N_i is defined by Eq. (3.14).

$$N_i = \frac{1}{4}(1+rr_i)(1+ss_i) \tag{3.14}$$

Here (r_i, s_i) is the position of the i^{th} node. A contour is characterized by the equation

$$f(x,y) = C \quad \text{or} \quad C = \sum_{i=1}^{4} N_i f_i \tag{3.15}$$

After substituting the values of shape function from Eq. (3.14) into Eq. (3.15), the contour Eq. (3.15) is simplified in local co-ordinates as

$$C = a_0 + a_1 r + a_2 s + a_3 rs \tag{3.15}$$

where,

$$a_0 = \frac{1}{4}(f_1 + f_2 + f_3 + f_4)$$

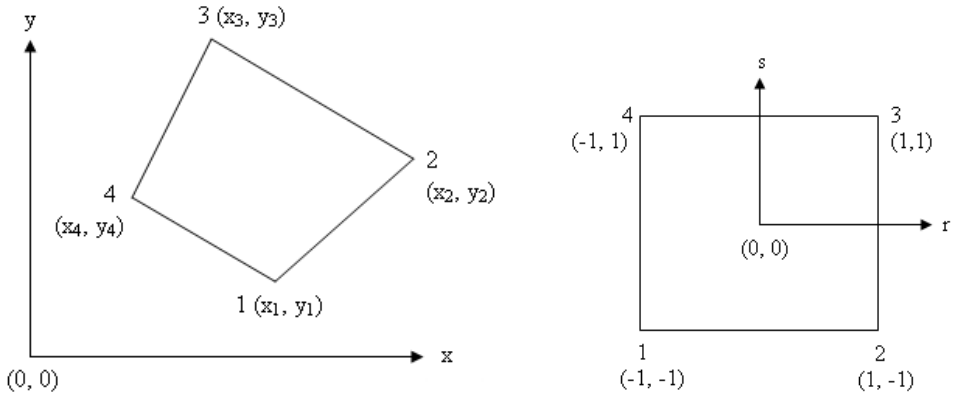
$$a_1 = \frac{1}{4}(-f_1 + f_2 + f_3 - f_4)$$

$$a_2 = \frac{1}{4}(-f_1 - f_2 + f_3 + f_4)$$

$$a_3 = \frac{1}{4}(f_1 - f_2 + f_3 - f_4) \qquad (3.16)$$

If the contour is traced out in local co-ordinates arrays, it can be transformed into global co-ordinates using the expressions in Eq. (3.17).

$$x = \sum_{i=1}^{4} N_i x_i \,, \qquad y = \sum_{i=1}^{4} N_i y_i \qquad (3.17)$$



(a) Element in global co-ordinates **(b) Element in local co-ordinates**

Figure 3.3 Four-node element in global and local co-ordinate systems

3.4.1 Tracing of Contour over Quadrilateral Element

The tracing of the contour is done as per the modification of algorithm discussed in section 3.3.

(i) Consider an element.

(ii) Find the minimum and the maximum of $f(x, y)$ over the element.

(iii) If $f_{\min} \le C \le f_{\max}$ then go to step (iv), else go to step (vii).

(iv) Determine the contour over the element.

(v) Refine the contour branch, if required.

(vi) Store the branch.

(vii) If all elements are not processed, then consider the next element and go to step (ii).

(viii) Join the contour segments of the same level.

(ix) Plot the contour lines on graphics devices.

The steps (iv) and (v) are elaborated in the following sections.

Step (iv) Contour branch over an element

The key to the contouring algorithm is to determine the end points of the branch intersecting with the element edges and subsequently to refine it to the desired accuracy.

All possible cases for the occurrence of the line over an element are explained as follows:

Case 1

If $a_3 = 0$, the contour Eq. (3.15) reduces to

$$C = a_0 + a_1 r + a_2 s \tag{3.18}$$

(i) If $a_1^2 + a_2^2 > 0$, Eq. (3.18) represents a straight line. Fig. 3.4 represents a typical example exhibiting this case. The case when a contour passes through only one vertex is not of practical importance.

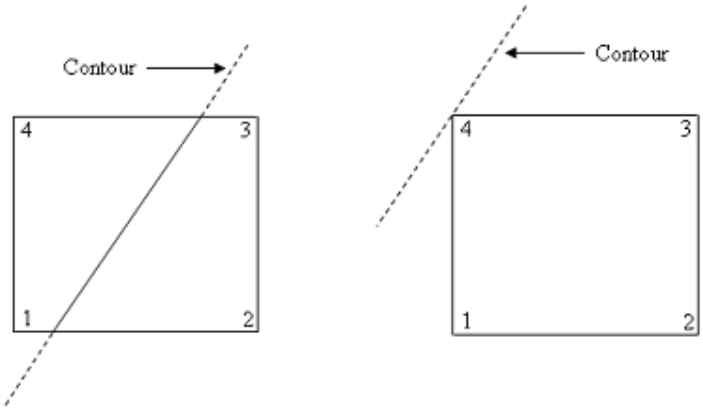


Figure 3.4 Contour as a straight line over an element

(ii) If $a_1^2 + a_2^2 = 0$, Eq. (3.18) represents a contour plane,

$$C = a_0 = f_1 = f_2 = f_3 = f_4$$

Case 2

If $a_3 \neq 0$, equation (3.15) represents a rectangular hyperbola:

$$\left(r + \frac{a_2}{a_3}\right)\left(s + \frac{a_1}{a_3}\right) = K, \tag{3.19}$$

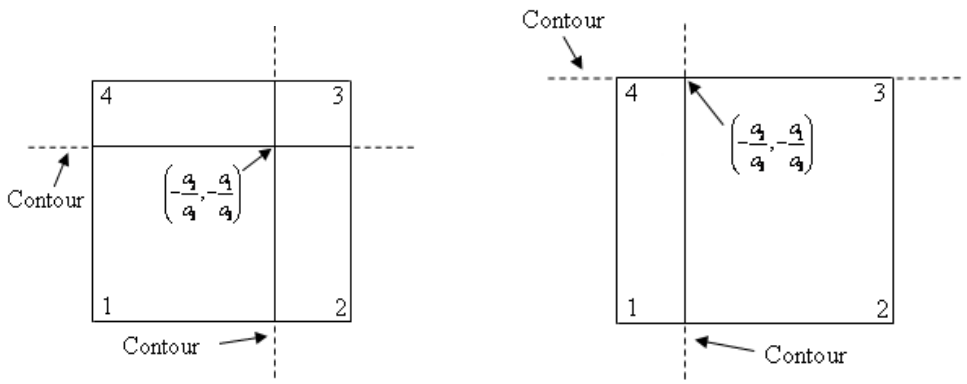
where $K = \frac{1}{a_3}\left(C - a_0 + \frac{a_1 a_2}{a_3}\right)$

The asymptotes $r = r_0 = -a_2/a_3$ and $s = s_0 = -a_1/a_3$ are parallel to s-axis and r-axis and, therefore, perpendicular to each other.

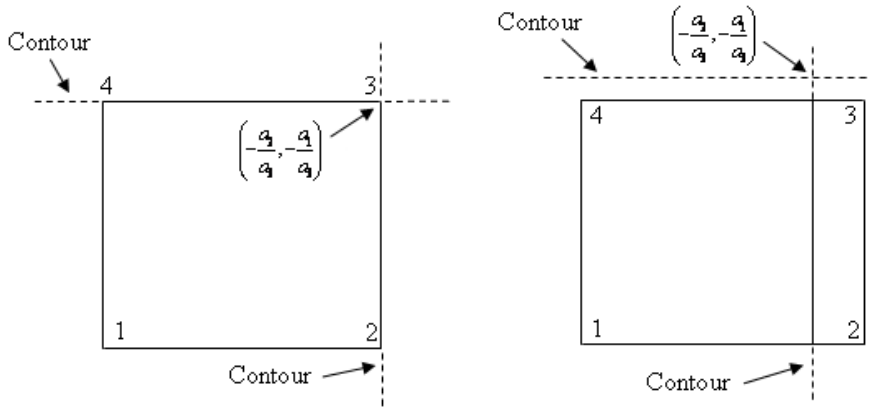
(i) If $K = 0$, the contour branches fall on the asymptotes. Figs. 3.5(a) to Fig. 3.5(d) explain the possible cases when centre of the hyperbola lies in different locations over an element. At least one of the axes will pass over the element, otherwise the condition $f_{\min} \leq C \leq f_{\max}$ will fail.

(ii) If $K > 0$, the contour branches will lie in the first and the third quadrant assuming (r_0, s_0) as the reference point of the hyperbola, because $\left(r + a_2/a_3\right)$ and $\left(s + a_1/a_3\right)$ will have the same sign.

(iii) Similarly, if $K < 0$, the contour branches will fall in the second and fourth quadrants.



(a) Centre of hyperbola inside the element **(b) Centre of hyperbola over an edge**



(c) Centre of hyperbola over a node **(d) Centre of hyperbola outside the edge**

Figure 3.5 Contour branches parallel to r-axis and x-axis

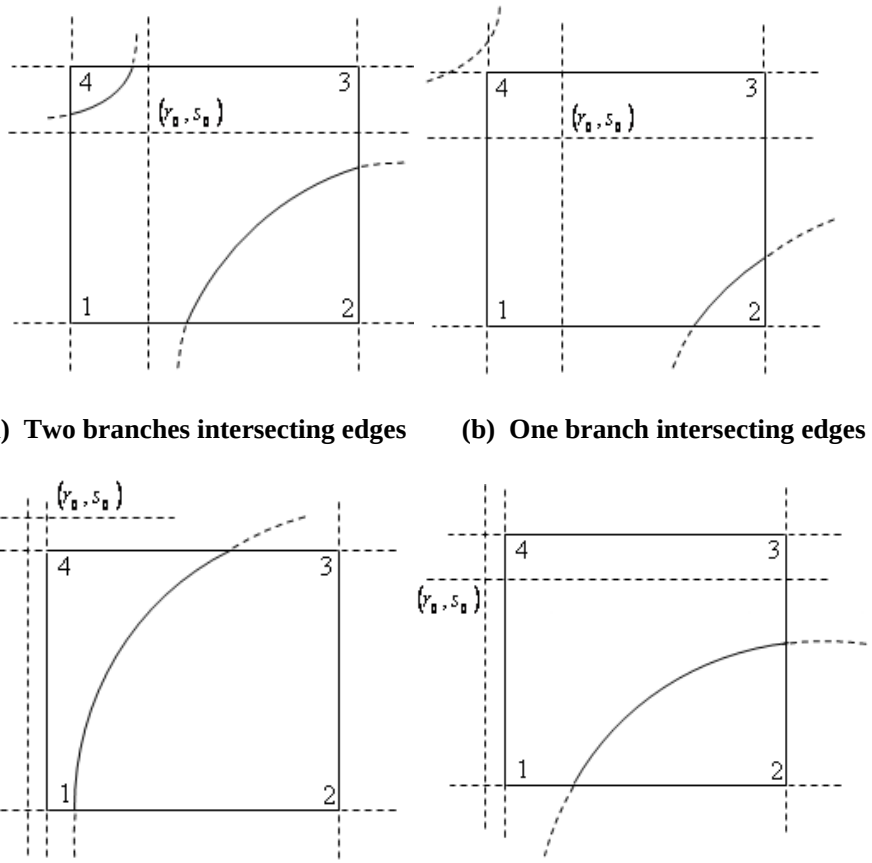
It is now easy to determine the intersection points with the edges and the number of branches. For example, if $K > 0$ and $(r_0, s_0) \in [-1,+1] \times [-1,+1]$ (i.e. the centre of the hyperbola lies within the square region), then there may be a maximum of two branches, the first one lying in the first quadrant and the second in the third quadrant. In order to find the end points of the first branch, we have to take the intersection with $r = +1$ and $s = +1$. Similarly, the second branch will intersect $r = -1$ and $s = -1$. Figs. 3.6(a) and 3.6(b) represent two such cases. Depending upon the location of the centre point and the sign of K, other cases can also be efficiently worked out as shown by Figs. 3.6(c) and 3.6(d).

Intersection with the $r-$edges $(r = \mp 1)$ results in Eq. (3.20).

$$s = \frac{C - a_0 \pm a_1}{(a_2 \mp a_3)} \tag{3.20}$$

and with the $s-$edges $(s = \mp 1)$ results in Eq. (3.21).

$$r = \frac{C - a_0 \pm a_2}{(a_1 \mp a_3)} \tag{3.21}$$



(a) Two branches intersecting edges **(b) One branch intersecting edges**

(c) Branch intersecting any two of four edges **(d) Branch will not intersect edge,** $s = +1$

Figure 3.6 Various cases of contour branches intersecting element edges

Step (v) Refinement of the contour branch

The contour branch over an element obtained so far is a straight line approximation to the branch of the hyperbola represented by the interpolation Eq. (3.15). Fig. 3.7 depicts a true branch and the approximated one. The accuracy is increased by finding the intermediate points which is now a simple task. For practical purposes, we shall first find the maximum error which will be reduced iteratively till it is brought below a predefined permissible quantity.

The maximum error representing the maximum shift in the location of the contour is given by

$$E = \sqrt{(r_m - r_p)^2 + (s_m - s_p)^2} \tag{3.22}$$

where (r_m, s_m) is a point on the hyperbola, the slope of which is the same as that of the straight line joining (s_1, r_1) and (s_2, r_2), the two end points of the contour branch. The

point (r_p, s_p) is the point of intersection of the normal to the straight line *AB* passing through (r_m, s_m).

Elementary calculus shows that s_m is given by Eq. (3.23a) as follows:

$$s_m = \frac{1}{a_3}\left[-a_1 \pm \sqrt{(a_1 + a_3 s_1)(a_1 + a_3 s_2)}\right] \tag{3.23a}$$

and, by substituting $s = s_m$, r_m can be found as in Eq. (3.23b)

$$r = \frac{C - a_0 - a_2 s}{a_1 + a_3 s} \tag{3.23b}$$

It is worth mentioning that the expressions under the square root in Eq. (3.23a) do not change sign owing to the continuity property of the contour branch. Also s_m is chosen such that $s_1 \le s_m \le s_2$.

The accuracy is now increased by subdividing the larger of the intervals $[r_1, r_2]$ and $[s_1, s_2]$. Let the $s-$interval be subdivided so that $s_3 = \frac{1}{2}(s_1 + s_2)$. The corresponding $r-$value, r_3 is found from Eq. (3.23b). The maximum error in the two intervals is computed and the process is repeated for that interval in which the error is still more than the permissible limit *i.e.* the resolution of the graphics system.

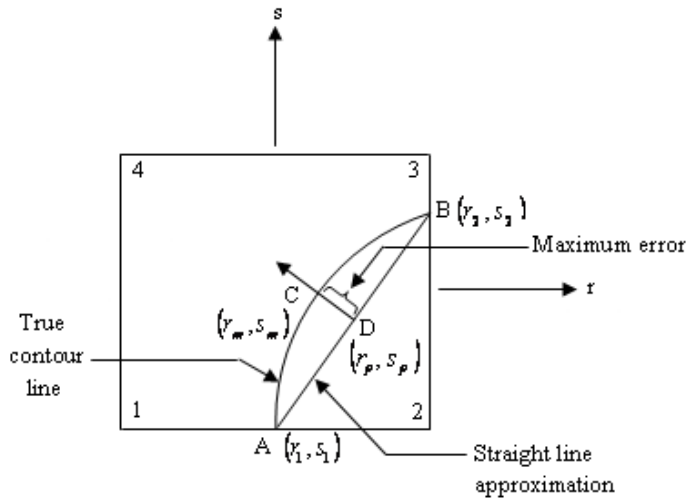


Figure 3.7 The maximum error in approximation over an element

3.5 PLOTTING OF CONTOURS IN 2D

The contour curves and surfaces are derived in user coordinates systems. These are to be plotted on the display devices such as on a computer screen or on a hardcopy device. The coordinates of display devices are called the screen coordinates. These are essentially 2D plane surfaces. Therefore, proper transformation of user coordinates are required to plot the contours onto the screen coordinates. In two dimensions, this process is straightforward, in the sense that only one transformation, called the window-to-viewport transformation, is required to map the coordinates.

3.5.1 Determining the Window

Let (x_i, y_i), $i = 0,1,...,n-1$, be the n-points of a contour line (or contour segment). Let $x_{\min} = \min_{0 \le i < n-1} \{x_i\}$, $y_{\min} = \min_{0 \le i < n-1} \{y_i\}$, $x_{\max} = \max_{0 \le i < n} \{x_i\}$, $y_{\max} = \max_{0 \le i < n} \{y_i\}$. Then the window is a rectangular region defined by $[x_{\min}, y_{\min}] \times [x_{\max}, y_{\max}]$.

3.5.2 Determining the Viewport

Viewport is set according to resolution of the display screen. The contours can be displayed either on the whole screen or on a part thereof. Thus the viewport is set as the default viewport (whole screen) or a sub-screen. Let $[V_{x\min}, V_{y\min}] \times [V_{x\max}, V_{y\max}]$ be the viewport, then for the whole screen $V_{x\min} = V_{y\min} = 0$ and $V_{x\max} = X_R - 1$ and $V_{y\max} = Y_R - 1$, where X_R and Y_R are the maximum resolutions along horizontal and vertical directions. If a part of the viewport is chosen, then we define $V_{x\min}$, $V_{x\max}$, $V_{y\min}$ and $V_{y\max}$ such that $0 \le V_{x\min} < V_{x\max} < X_R$ and $0 \le V_{y\min} < V_{y\max} < Y_R$. Having defined the viewport, the window to viewport transformation is carried out using the following matrix equation in homogenous coordinate system.

$$\begin{bmatrix} V_x \\ V_y \\ 1 \end{bmatrix} = M \begin{bmatrix} W_x \\ W_y \\ 1 \end{bmatrix} \tag{3.24a}$$

where,

$$M = \begin{bmatrix} 1 & 0 & V_{x\min} \\ 0 & 1 & V_{y\min} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_{\min} \\ 0 & 1 & -y_{\min} \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.24b}$$

and,

$$s_x = \frac{V_{x\max} - V_{x\min}}{X_{\max} - X_{\min}}, \qquad s_y = \frac{V_{y\max} - V_{y\min}}{Y_{\max} - Y_{\min}} \tag{3.24c}$$

In Eq. (3.24a), a point (W_x, W_y) of the window is mapped to a point (V_x, V_y) in the viewport such that $0 \le V_x < X_R$ and $0 \le V_y < Y_R$.

3.6 IMPLEMENTATION OF THE ALGORITHM FOR TRIANGULAR ELEMENT

The contour plot algorithm, discussed in section 3.3, is implemented in Turbo C++ and runs on an IBM compatible PC with 3.0 GHz CPU and 256 RAM under Windows operating system. For better graphics output, the algorithms are also implemented in Visual C++ 6.0 environment with OpenGL graphics libraries (Hearn and Baker, 2009; Chen and Cheng, 2005) to display the geometry along with the contour lines and surfaces.

The problems considered here are deflection and thermal problems of the different components.

The first problem is a deflection of a 2D beam with specifications shown in Fig. 3.8.

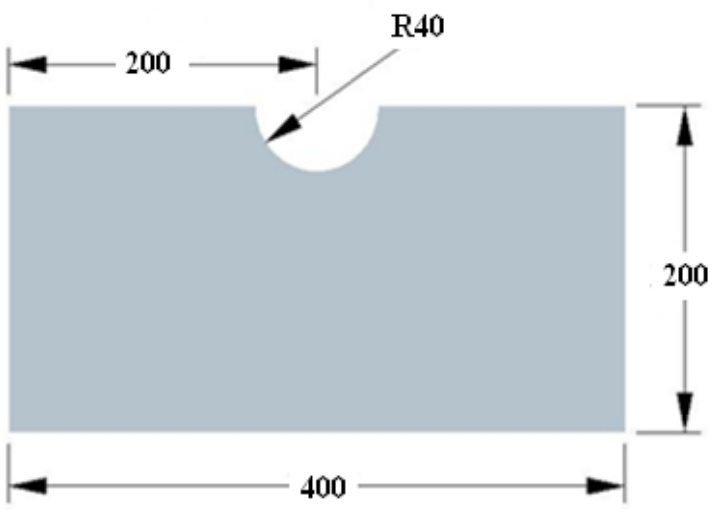


Figure 3.8 Specification of the beam

In the first case the beam is fixed at one end and loaded axially at the other end, as shown in Fig. 3.9(a), and in the second case the beam is fixed at one end and a vertical load is applied at the other end, as shown in Fig. 3.9(b). For a load of 100 N, Young's Modulus $2.1\times10^5\ N/mm^2$, Poisson's ratio 0.3, the values of deflections are derived using FEA, considering the problem as 2D plane problem with negligible thickness.

Figure 3.9(a) Beam fixed at one end and axially loaded at other end

Figure 3.9(b) Beam fixed at one end and vertically loaded at other end

Since the beam is axially loaded in the first case, the general expression for direct horizontal deflection (Subramanian, 2005) is given by Eq. (3.25).

$$\delta_x = \frac{WL}{A_x E} \tag{3.25}$$

where W is the load applied, L is the length of beam, A_x is the area of cross section and E is the Young's Modulus. The value of A_x is constant in section *1-1* (from location A to B) and *3-3* (from location C to D) as shown in Fig. 3.10, but in section *2-2* (from location B to C), the value of A_x varies with respect to length of the beam. Due to the varying cross

section at section *2-2*, shown in Fig. 3.10, there will be eccentricity in loading. There will be more elongation at upper layers than the compression at lower layers. This will give rise to resultant horizontal and vertical deflections, with the vertical deflection being relatively less as compared to the horizontal one. The resultant horizontal and vertical deflections are considered for contouring in the numerical experimentation for the proposed algorithm.

Fig. 3.11(a) and 3.11(b) depict the contour plots of horizontal and vertical deflections using three-node triangles for different levels of deflections.

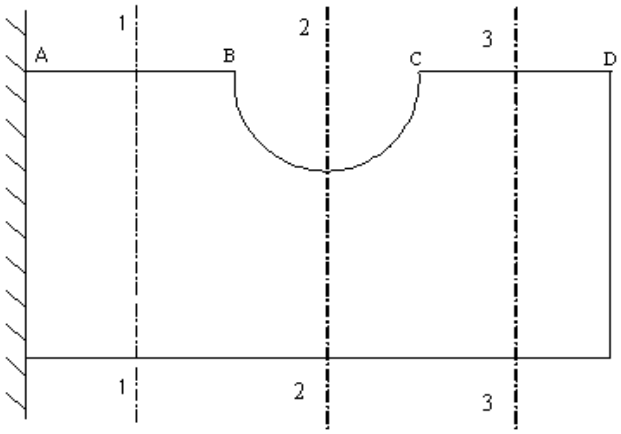
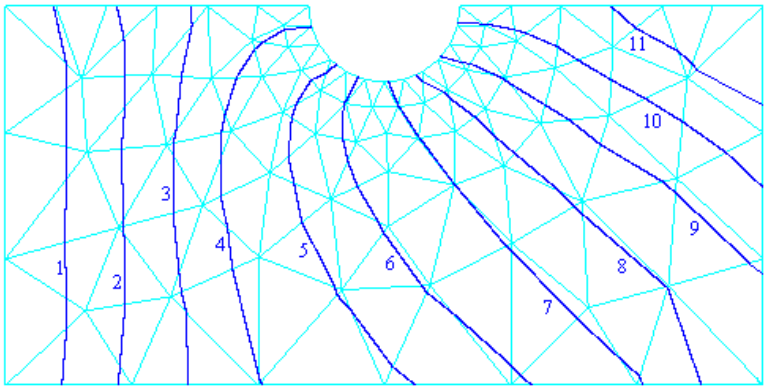
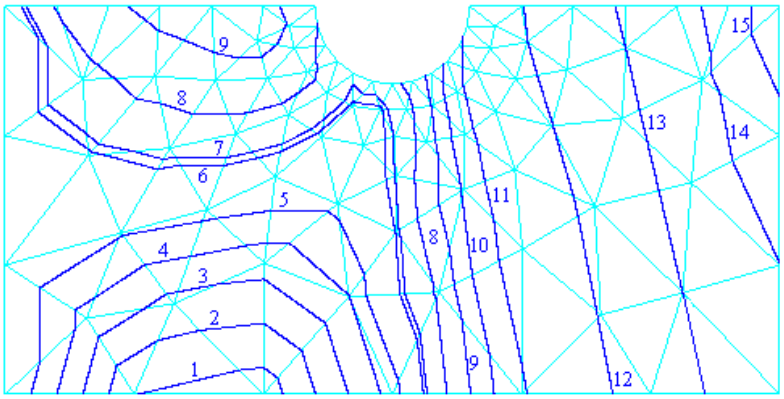


Figure 3.10 Section 1-1, 2-2 and 3-3 considered in the beam



Contour Levels	δ_x
1	0.5 x10^{-4}
2	1 x10^{-4}
3	1.5 x10^{-4}
4	2 x10^{-4}
5	2.8 x10^{-4}
6	3.5 x10^{-4}
7	4.5 x10^{-4}
8	5.5 x10^{-4}
9	6.5 x10^{-4}
10	7.5 x10^{-4}
11	8.5 x10^{-4}

Figure 3.11(a) Contour plot of horizontal deflection using three-node triangles for axially loaded beam (units in mm)



Contour Levels	δ_y
1	0.5 x10^{-4}
2	0.4 x10^{-4}
3	0.3 x10^{-4}
4	0.2 x10^{-4}
5	0.1 x10^{-4}
6	-0.1 x10^{-4}
7	-0.13 x10^{-4}
8	-0.3 x10^{-4}
9	-0.5 x10^{-4}
10	-0.7 x10^{-4}
11	-1 x10^{-4}
12	-2 x10^{-4}
13	-3 x10^{-4}
14	-4 x10^{-4}
15	-5 x10^{-4}

Figure 3.11(b) Contour plot of vertical deflection using three-node triangles for axially loaded beam (units in mm)

For the second case, when the load is applied vertically downwards, the general equation for bending moment (Subramanian, 2005) is given by Eq. (3.26).

$$EI_x \frac{d^2\delta_y}{dx^2} = -W(L-x) \tag{3.26}$$

The solution to this problem under the conditions of deflection and slope being zero at fixed end, *i.e.* at $x = 0$, $\delta_y = \frac{d\delta_y}{dx} = 0$, is given by Eq. (3.27).

$$EI_x\delta_y = -\left(\frac{WLx^2}{2} - \frac{Wx^3}{6}\right) \tag{3.27}$$

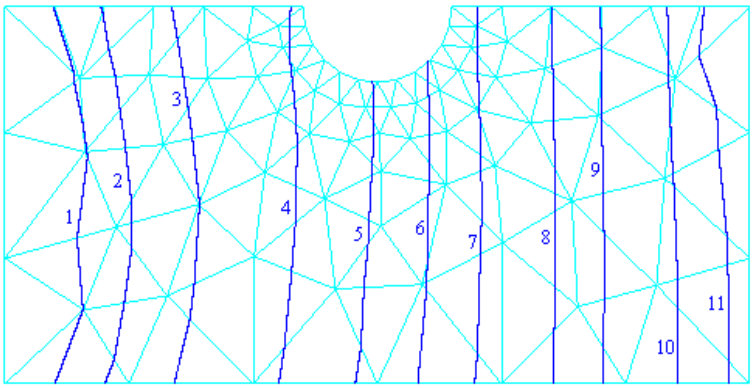
where δ_y is the vertical deflection, E is the Young's Modulus, I_x is the area moment of inertia given by $\frac{th^3}{12}$ for a rectangular section, W is the point load applied, L is the length of beam, x is the distance of a section from the fixed end, t is the thickness and h is the height of the beam. The value of I_x is constant in sections *1-1*(from location A to B) and *3-3* (from location C to D), but in section *2-2* (from location B to C), shown in Fig. 3.10, the value of I_x varies with respect to the length of the beam.

There will also be horizontal shear load due to the horizontal shear stress in the beam which will give rise to horizontal deflection. This horizontal deflection is relatively less than the vertical one. The horizontal shear load per unit length of the beam at a layer above the neutral axis is given by

$$W_x = \frac{Sa\bar{y}}{I_x} \tag{3.28}$$

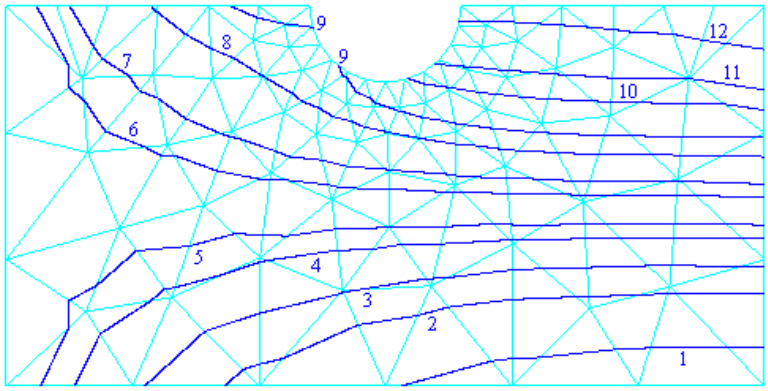
where S is the shear force at a given section, $a\bar{y}$ is the first moment of the area beyond a layer about the neutral axis.

Fig. 3.12(a) and 3.12(b) are the contour plots of vertical and horizontal deflections using three-node triangles for different levels of deflections.



Contour Levels	δ_y
1	-0.5 x10^{-4}
2	-1 x10^{-4}
3	-2 x10^{-4}
4	-4 x10^{-4}
5	-6 x10^{-4}
6	-8 x10^{-4}
7	-10 x10^{-4}
8	-13 x10^{-4}
9	-15 x10^{-4}
10	-18 x10^{-4}
11	-20 x10^{-4}

Figure 3.12(a) Contour plot of vertical deflection using three-node triangles for vertical loaded beam (units in mm)



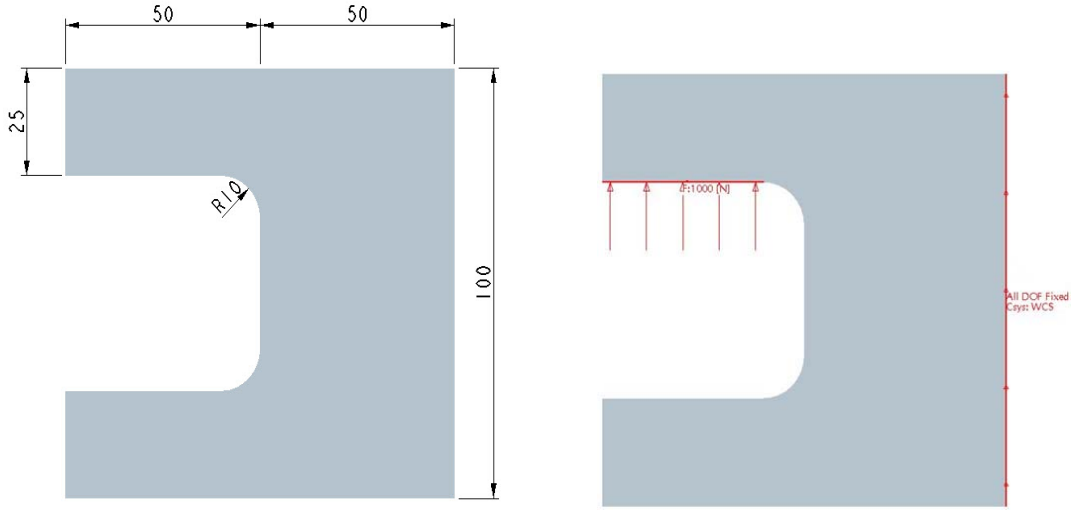
Contour Levels	δ_x
1	-5 x10^{-4}
2	-3 x10^{-4}
3	-2 x10^{-4}
4	-1 x10^{-4}
5	-0.5 x10^{-4}
6	0.5 x10^{-4}
7	1 x10^{-4}
8	2 x10^{-4}
9	2.7 x10^{-4}
10	4 x10^{-4}
11	5 x10^{-4}
12	7 x10^{-4}

Figure 3.12(b) Contour plot of horizontal deflection using three-node triangles for vertically loaded beam (units in mm)

The next component is a beam with specifications shown in Fig. 3.13 (a). The beam is fixed at one end and loaded vertically at the other end, as shown in Fig. 3.13(b). For a load of 1000 N, Young's Modulus 0.2×10^9 *MPa*, Poisson's ratio 0.3, the values of deflections are derived using FEA, considering the problem as 2D plane problem with

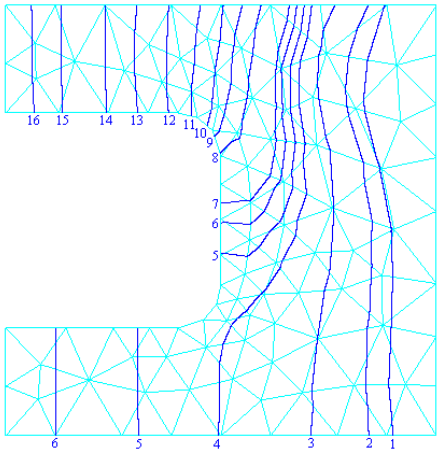
negligible thickness. As the beam is vertically loaded, the horizontal and vertical deflections are given by Eqs. (3.26) to (3.28).

Fig. 3.14 (a) and 3.14(b) present the contour plots of vertical and horizontal deflections using three-node triangles for different levels of deflections.



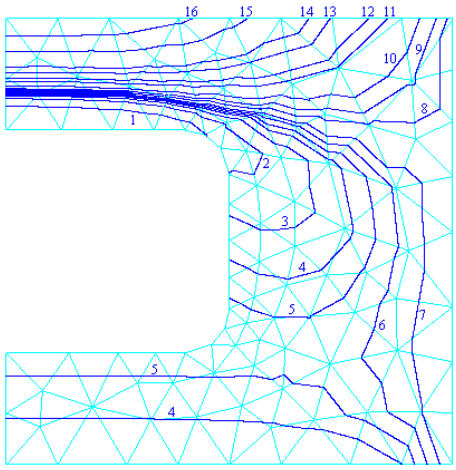
(a) **(b)**

Figure 3.13 (a) Specification of the component (all dimensions in mm); (b) Loading and boundary conditions of the component



Contour Levels	δ_y
1	2 x10^{-4}
2	3 x10^{-4}
3	5 x10^{-4}
4	7 x10^{-4}
5	8 x10^{-4}
6	9 x10^{-4}
7	10 x10^{-4}
8	15 x10^{-4}
9	20 x10^{-4}
10	25 x10^{-4}
11	35 x10^{-4}
12	50 x10^{-4}
13	70 x10^{-4}
14	90 x10^{-4}
15	120 x10^{-4}
16	140 x10^{-4}

Figure 3.14(a) Contour plot of vertical deflection using three-node triangles for the component



Contour Levels	δ_x
1	-10 x10^{-4}
2	-5 x10^{-4}
3	-3 x10^{-4}
4	-2 x10^{-4}
5	-1.5 x10^{-4}
6	-1 x10^{-4}
7	-0.5 x10^{-4}
8	0.5 x10^{-4}
9	1.5 x10^{-4}
10	3 x10^{-4}
11	6 x10^{-4}
12	8 x10^{-4}
13	12 x10^{-4}
14	15 x10^{-4}
15	25 x10^{-4}
16	35 x10^{-4}

Figure 3.14(b) Contour plot of horizontal deflection using three-node triangles for the component

Fig. 3.15(a) shows the specifications of a rocker arm. The arm is fixed at the centre and loaded vertically at both the end, as shown in Fig. 3.15(b). For a load of 1000 N, Young's Modulus $0.2\times10^{9}\,MPa$, Poisson's ratio 0.3, the values of deflection are derived using FEA, considering the problem as 2D plane problem with negligible thickness. As the arm is vertically loaded, the vertical deflection is given by Eq. (3.26).

Fig. 3.16 shows the contour plots of vertical deflections using three-node triangles for different levels of deflection.

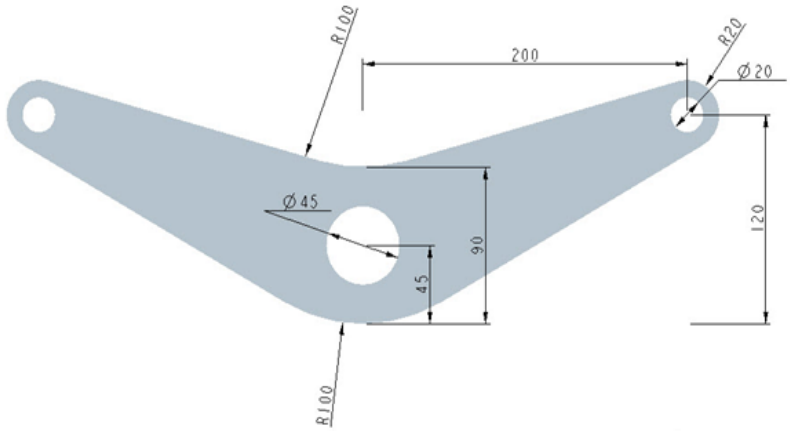


Figure 3.15(a) Specifications of the rocker arm (all dimensions in mm)

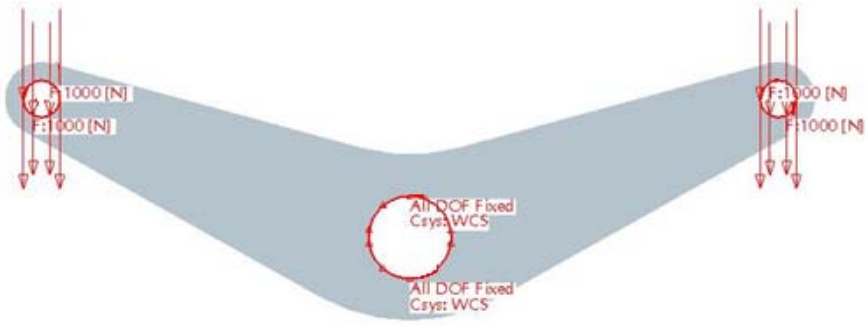
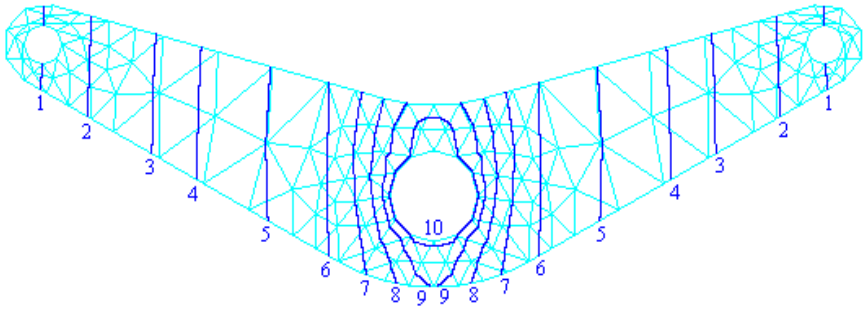


Figure 3.15(b) Boundary conditions and loading of the rocker arm



Contour Levels	δ_y
1	-270 x10^{-4}
2	-210 x10^{-4}
3	-140 x10^{-4}
4	-100 x10^{-4}
5	-50 x10^{-4}
6	-20 x10^{-4}
7	-10 x10^{-4}
8	-5 x10^{-4}
9	-2 x10^{-4}
10	-0.2 x10^{-4}

Figure 3.16 Contour plot of vertical deflection using three-node triangles for rocker arm

Fig. 3.17(a) shows the specifications of a spanner. The spanner is fixed at two different points and loaded vertically, as shown in Fig. 3.17(b). For a load of 1000 N, Young's Modulus $0.2\times10^{9}\,MPa$, Poisson's ratio 0.3, the values of deflection are derived using FEA, considering the problem as 2D plane problem with negligible thickness. As the spanner is vertically loaded, the vertical deflection is given by Eq. (3.26).

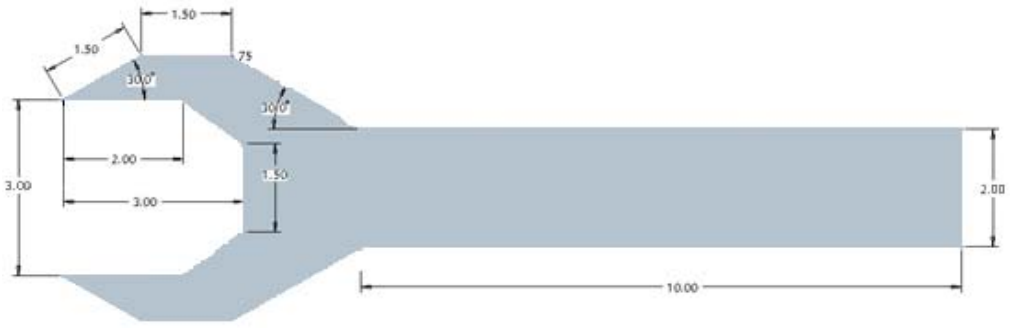
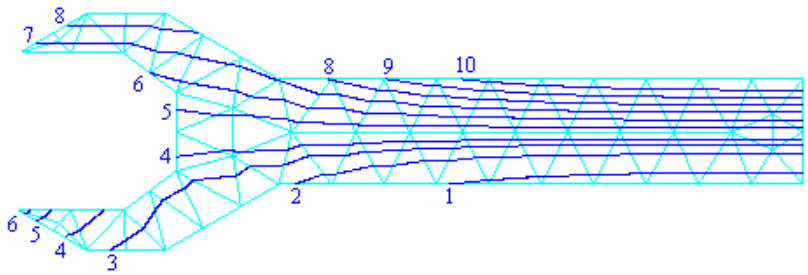


Figure 3.17(a) Specifications of a spanner (all dimensions in mm)

Figure 3.17(b) Boundary conditions and loading of the spanner

Fig. 3.18 shows the contour plots of vertical deflections using three-node triangles for different levels of deflection.



Contour Levels	δ_x
1	-150 x10^{-4}
2	-100 x10^{-4}
3	-75 x10^{-4}
4	-60 x10^{-4}
5	-30 x10^{-4}
6	-10 x10^{-4}
7	10 x10^{-4}
8	30 x10^{-4}
9	40 x10^{-4}
10	50 x10^{-4}

Figure 3.18 Contour plot of horizontal deflection using three-node triangles for the spanner

Fig. 3.19(a) shows the specifications of a connecting rod. The connecting rod is fixed at one end and loaded axially, as shown in Fig. 3.19(b). For a load of 1000 N, Young's Modulus $0.2\times10^{9}\,MPa$, Poisson's ratio 0.3, the values of deflections are derived using FEA, considering the problem as 2D plane problem with negligible thickness. As the connecting rod is axially loaded, the horizontal deflection is given by Eq. (3.25).

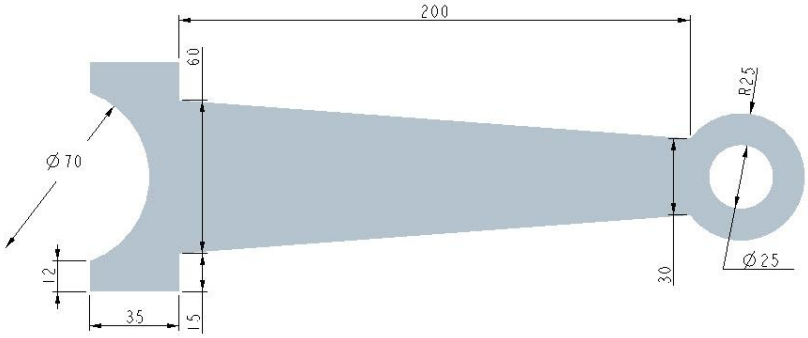


Figure 3.19(a) Specifications of connecting rod (all dimensions in mm)

Figure 3.19(b) Boundary conditions and loading of the connecting rod

Fig. 3.20 shows the contour plots of horizontal deflections using three-node triangles for different levels of deflection.

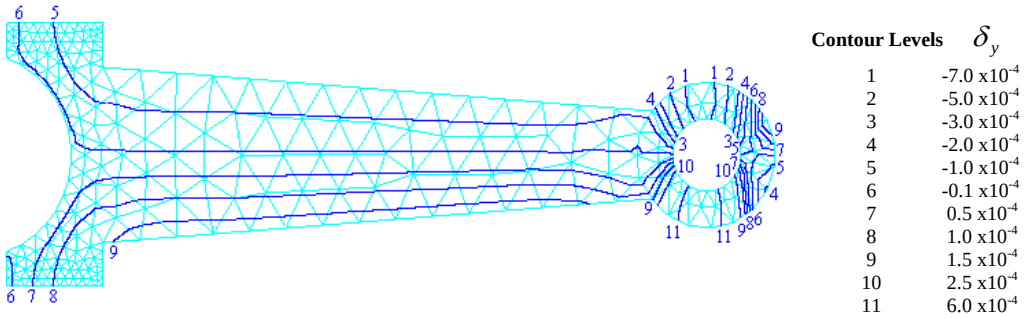


Figure 3.20 Contour plot of vertical deflection using three-node triangles for connecting rod

Fig. 3.21(a) shows the specifications of a connecting plate. The connecting plate is fixed at three holes and loaded vertically at the centre hole, as shown in Fig. 3.21(b). For a load of 1000 N, Young's Modulus $0.2\times10^{9}\,MPa$, Poisson's ratio 0.3, the values of deflection are derived using FEA, considering the problem as 2D plane problem with negligible thickness. As the spanner is axially loaded downwards, the vertical deflection is given by Eq. (3.25). As explained in first problem, the eccentricity will also give rise to horizontal deflection.

Fig. 3.22(a) and 3.22(b) show the contour plots of vertical and horizontal deflections using three-node triangles for different levels of deflection.

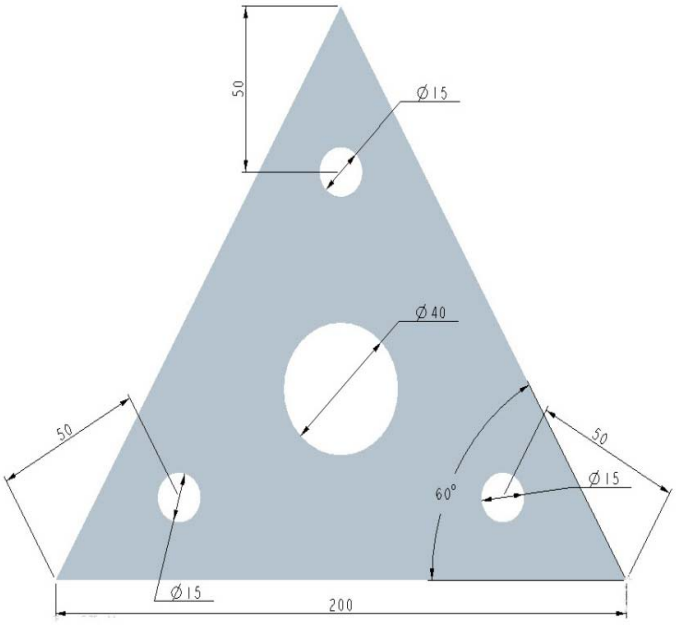


Figure 3.21(a) Specifications of a plate (all dimensions in mm)

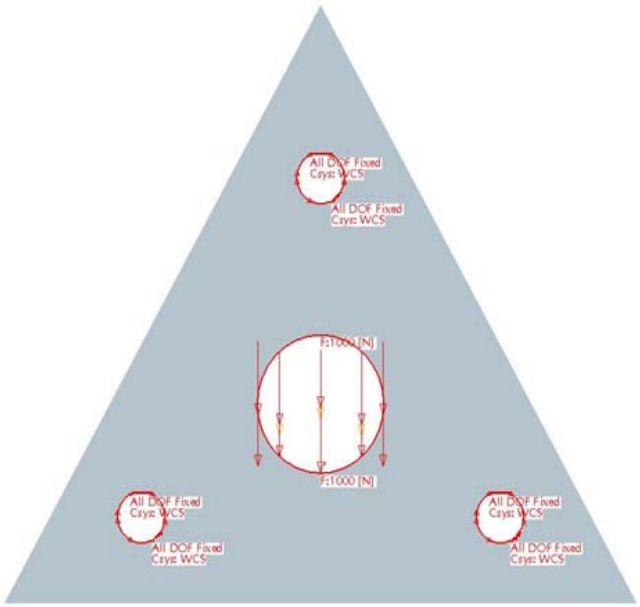


Figure 3.21(b) Boundary conditions and loading of the plate

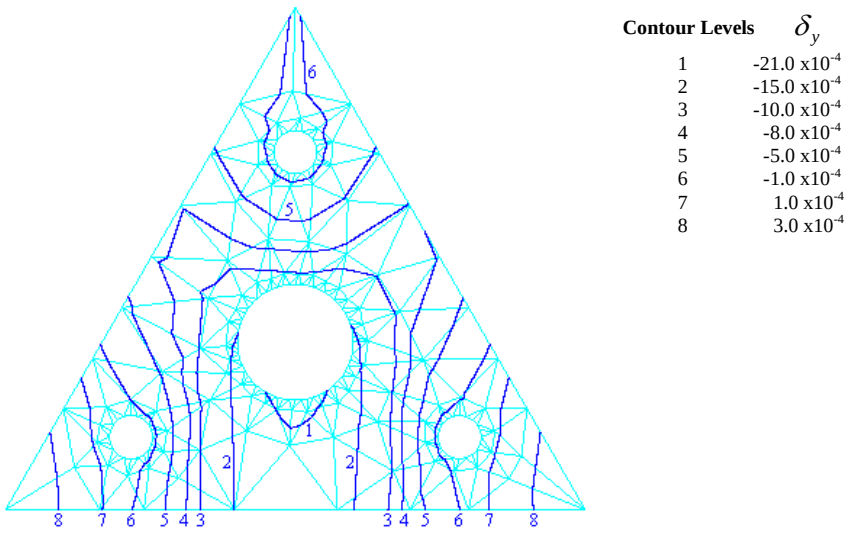


Figure 3.22(a) Contour plot of vertical deflection using three-node triangles for the plate

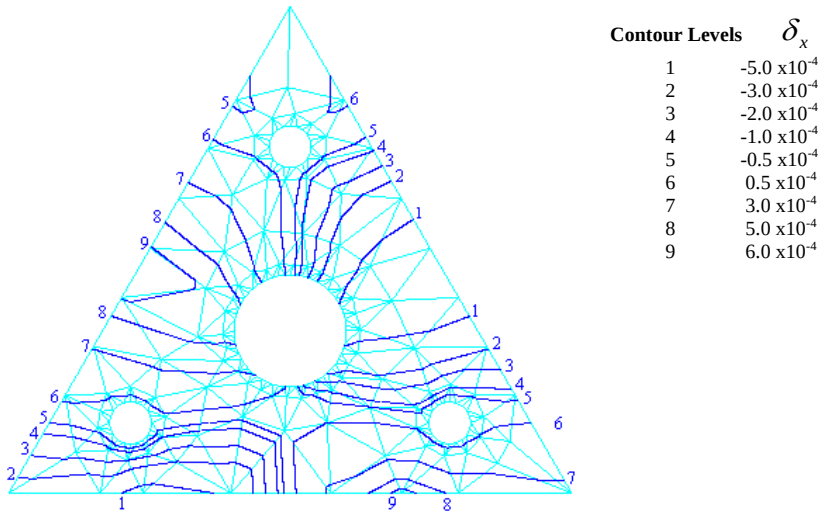


Figure 3.22(b) Contour plot of horizontal deflection using three-node triangles for the plate

Fig. 3.23(a) shows the specifications of a connecting plate. The connecting plate is fixed at the top end and loaded vertically at the hole, as shown in Fig. 3.23(b). For a load of 1000 N, Young's Modulus $0.2\times10^{9}\,MPa$, Poisson's ratio 0.3, the values of deflections are derived using FEA, considering the problem as 2D plane problem with negligible thickness. As the plate is loaded downwards, the vertical deflection is given by

Eq. (3.25). As explained in first problem, the eccentricity will also give rise to horizontal deflection.

Fig. 3.24(a) and 3.24(b) show the contour plots of vertical and horizontal deflections using three-node triangles for different levels of deflections.

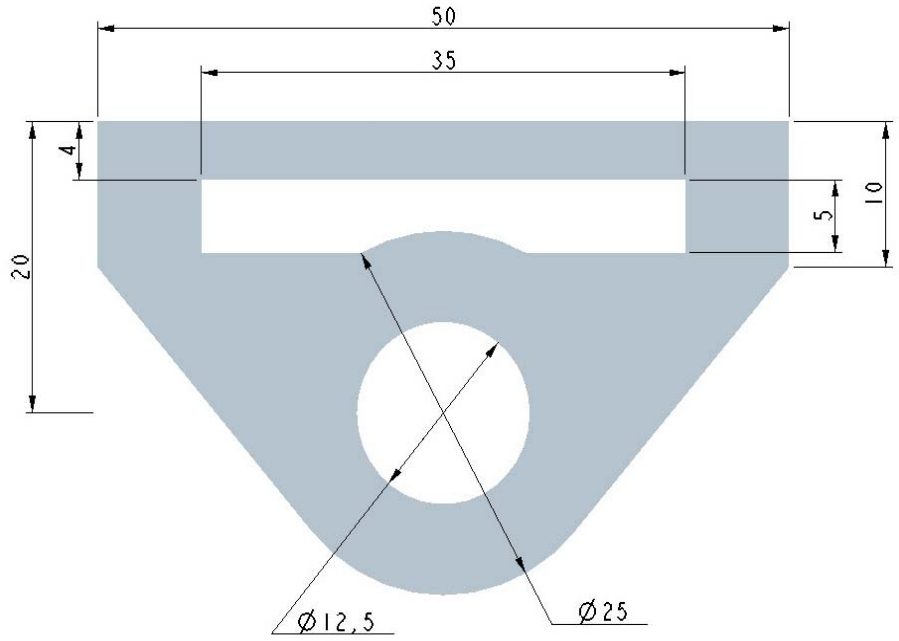


Figure 3.23(a) Specifications of a plate (all dimensions in mm)

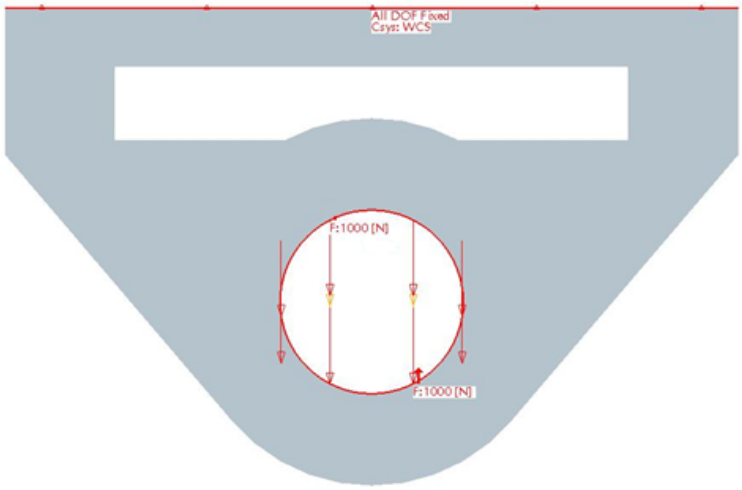
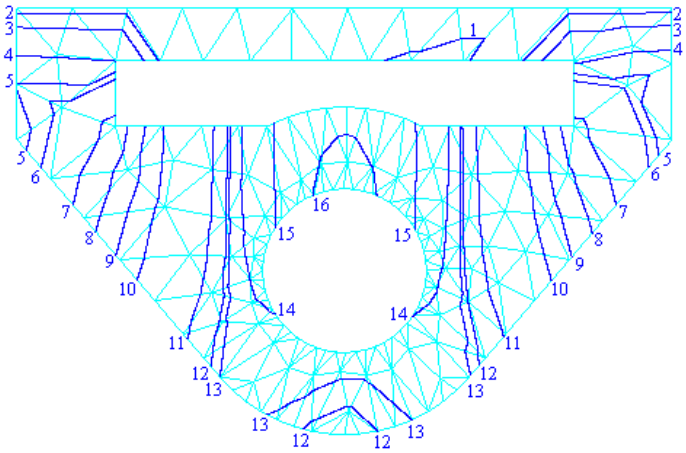
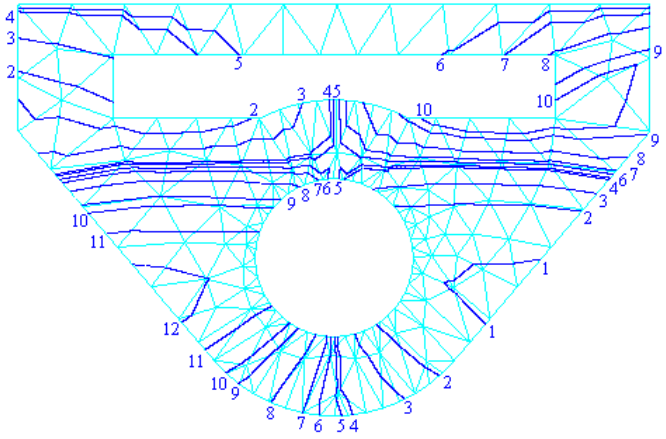


Figure 3.23(b) Boundary conditions and loading of the plate



Contour Levels	δ_y
1	0.03 x10^{-4}
2	-0.5 x10^{-4}
3	-2 x10^{-4}
4	-5 x10^{-4}
5	-8 x10^{-4}
6	-10 x10^{-4}
7	-20 x10^{-4}
8	-30 x10^{-4}
9	-40 x10^{-4}
10	-50 x10^{-4}
11	-70 x10^{-4}
12	-75 x10^{-4}
13	-76 x10^{-4}
14	-80 x10^{-4}
15	-90 x10^{-4}
16	-100 x10^{-4}

Figure 3.24(a) Contour plot of vertical deflection using three-node triangles for the plate



Contour Levels	δ_x
1	20 x10^{-4}
2	10 x10^{-4}
3	5 x10^{-4}
4	1 x10^{-4}
5	0.5 x10^{-4}
6	-0.5 x10^{-4}
7	-1.5 x10^{-4}
8	-4 x10^{-4}
9	-8 x10^{-4}
10	-10 x10^{-4}
11	-15 x10^{-4}
12	-20 x10^{-4}

Figure 3.24(b) Contour plot of horizontal deflection using three-node triangles for the plate

The next problem is a heat conduction problem, with specifications shown in Fig. 3.25(a). The temperature of 673^{0}C and 273^{0}C is fixed at the inside and outside of the reactor, as shown in Fig. 3.23(b). For the known heat flow rate, thermal conductivity of the materials as 0.01 W/m-deg and 0.0057 W/m-deg, and the given boundary temperatures, the values of temperature distribution are derived using FEA.

The temperature distribution through a wall can be calculated by using the Fourier rate equation (Kumar, 2008). The Fourier rate equation is simplified to the expression shown in Eq. (3.29).

$$Q = \frac{kA(t_1 - t_2)}{\delta} \tag{3.29}$$

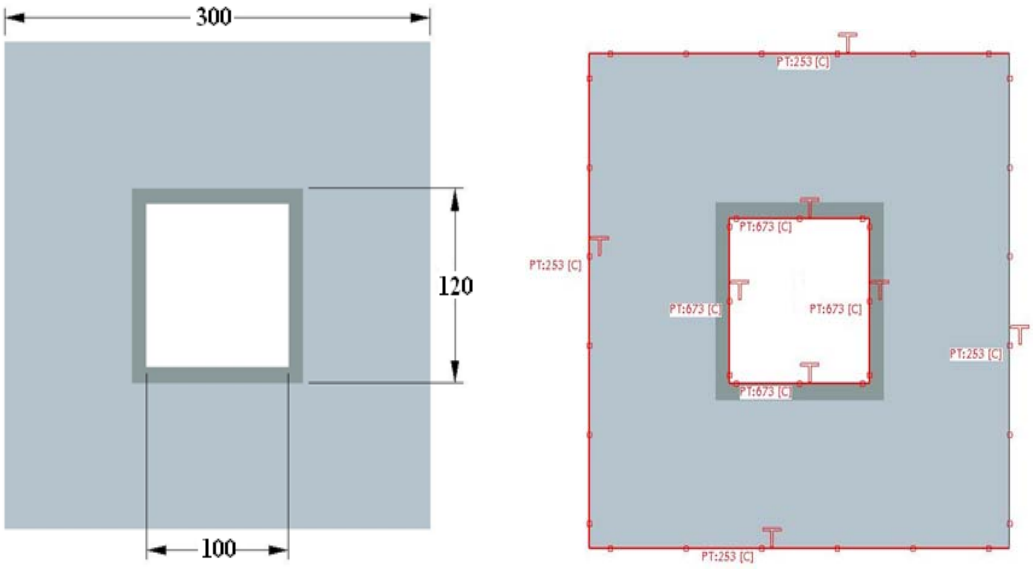
where Q is the heat flow rate, k is the thermal conductivity of the material, A is the area of cross section, δ is the width of wall, t_1 and t_2 are the temperatures at the inside and outside of the wall.

For the n - layer composite wall, the Eq. (3.29) is extended as

$$Q = \frac{(t_1 - t_{n+1})}{\sum_{i=1}^{n} R_i} \tag{3.30}$$

where $R_i = \frac{\delta_i}{k_i A_i}$ is the thermal resistance of i^{th} layer of the composite wall.

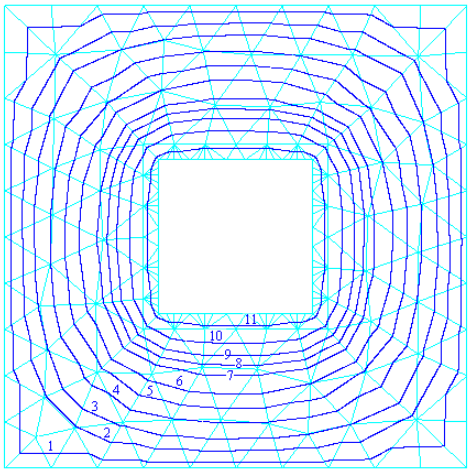
Fig. 3.26 shows the contour plots of temperature distribution using three-node triangles for different levels of temperatures.



(a) **(b)**

Figure 3.25 (a) Specifications of a reactor (all dimensions in mm);

(b) Boundary conditions and temperature fixation of the plate



Contour Levels	T
1	270
2	300
3	330
4	370
5	410
6	450
7	500
8	530
9	560
10	600
11	650

Figure 3.26 Contour plot of temperature distribution (in ^{0}C) using 3-node triangles for the reactor

It is observed that the contouring algorithm is very fast and takes very less time for the different problems shown above. The developed technique can be used to plot contours for applications where high speed is required. Although the discussed algorithm simplifies contour plotting, inaccuracies due to lower order interpolation are however introduced.

3.7 IMPLEMENTATION OF THE ALGORITHM FOR QUADRILATERAL ELEMENT

The contour plot algorithm, discussed in section 3.4, is implemented using the previously defined system specifications.

The first problem analyzed is a deflection problem of a rocker arm with specifications shown in Fig. 3.27. The rocker arm is fixed at the centre and loaded vertically downwards through the two side holes, shown in Fig. 3.15(b). For a load of 1000 *N*, Young's Modulus $200\times10^{6}\,MPa$, Poisson's ratio 0.3, the values of vertical deflections are derived using FEA, considering the problem as 2D plane problem with negligible thickness.

As the load is applied vertically downwards, the horizontal and vertical deflections are given by Eqs. (3.26) to (3.28).

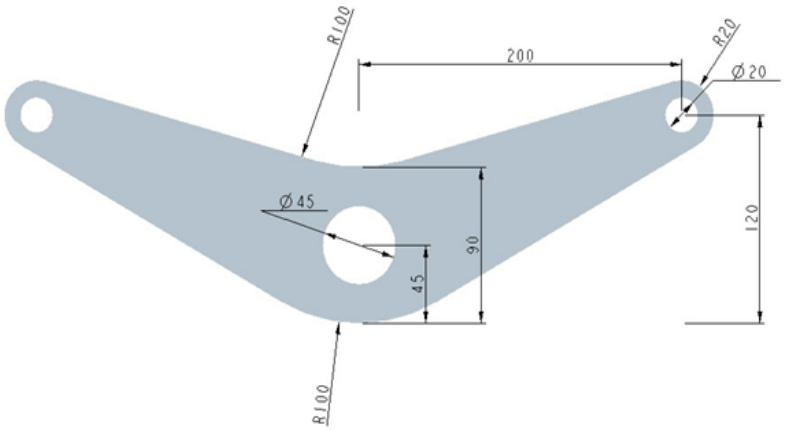
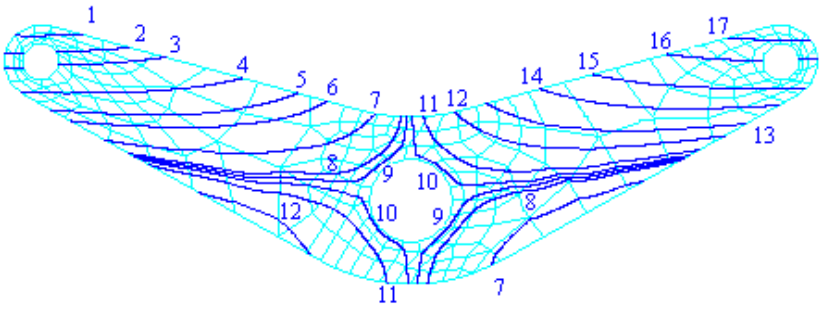


Fig. 3.27 Specifications of the rocker arm (all dimensions in mm)

The horizontal deflections are considered for contouring in the numerical experimentation for the proposed algorithm. Fig. 3.28 depicts the contour plots of horizontal deflections using four-node quadrilateral elements for different levels of deflection for rocker arm.



Contour Levels	δ_x
1	-240 x10^{-4}
2	-200 x10^{-4}
3	-170 x10^{-4}
4	-110 x10^{-4}
5	-70 x10^{-4}
6	-50 x10^{-4}
7	-20 x10^{-4}
8	-5 x10^{-4}
9	-2 x10^{-4}
10	1 x10^{-4}
11	7 x10^{-4}
12	25 x10^{-4}
13	45 x10^{-4}
14	80 x10^{-4}
15	120 x10^{-4}
16	180 x10^{-4}
17	230 x10^{-4}

Fig. 3.28 Contour plots of horizontal deflection for rocker arm

Fig. 3.29(a) shows the specifications of a connecting plate. The connecting plate is fixed at the top end and loaded vertically at the hole, as shown in Fig. 3.29(b). For a load of 1000 N, Young's Modulus $0.2\times10^{9}\,MPa$, Poisson's ratio 0.3, the values of deflection are derived using FEA, considering the problem as 2D plane problem with negligible thickness. As the plate is axially loaded downwards, the vertical deflection is given by Eq. (3.25). As explained in first problem, the eccentricity will also give rise to horizontal deflection. Fig. 3.30(a) and 3.30(b) show the contour plots of vertical and horizontal deflections using four-node quadrilateral elements for different levels of deflections.

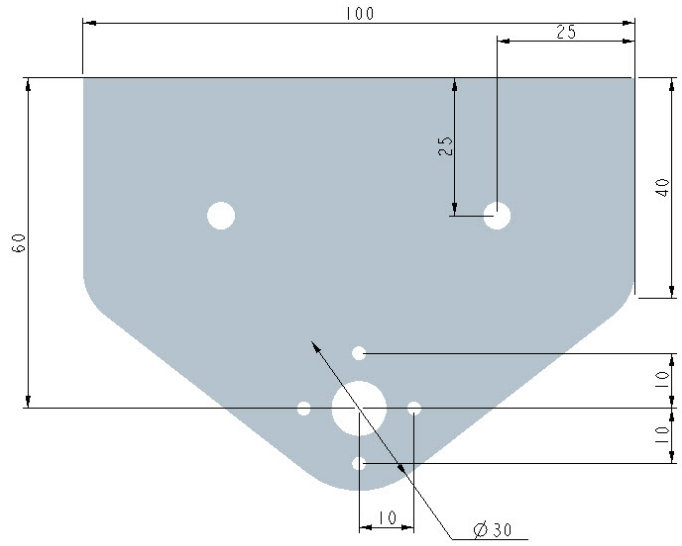


Figure 3.29(a) Specifications of a plate (all dimensions in mm)

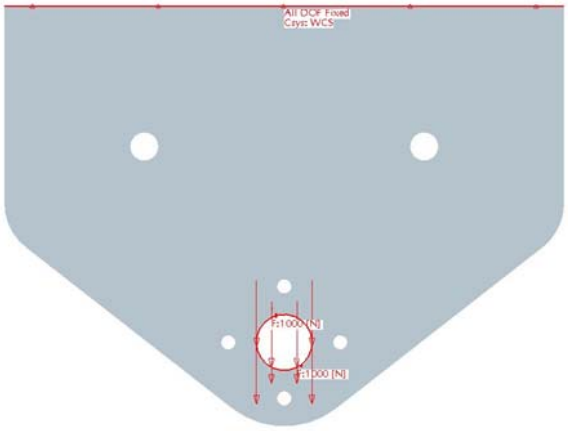
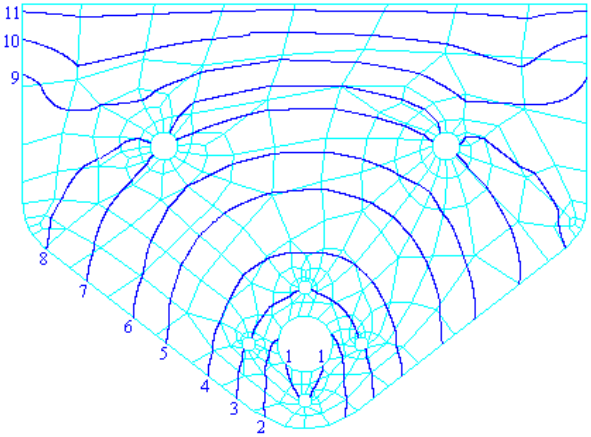
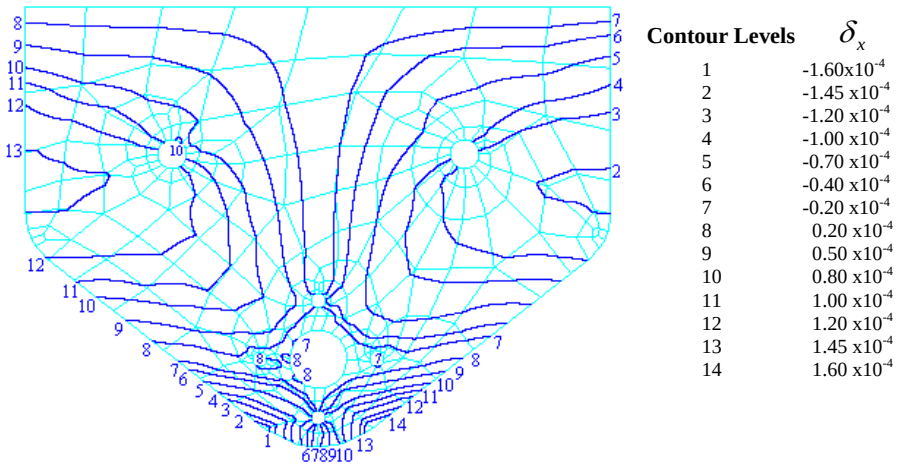


Figure 3.29(b) Boundary conditions and loading of the plate



Contour Levels	δ_y
1	-12 x10^{-4}
2	-10 x10^{-4}
3	-8.0 x10^{-4}
4	-6.0 x10^{-4}
5	-4.0 x10^{-4}
6	-3.0 x10^{-4}
7	-2.0 x10^{-4}
8	-1.5 x10^{-4}
9	-1.0 x10^{-4}
10	-0.5 x10^{-4}
11	-0.1 x10^{-4}

Figure 3.30(a) Contour plots of vertical deflection for the above specified plate



Contour Levels	δ_x
1	-1.60x10^{-4}
2	-1.45 x10^{-4}
3	-1.20 x10^{-4}
4	-1.00 x10^{-4}
5	-0.70 x10^{-4}
6	-0.40 x10^{-4}
7	-0.20 x10^{-4}
8	0.20 x10^{-4}
9	0.50 x10^{-4}
10	0.80 x10^{-4}
11	1.00 x10^{-4}
12	1.20 x10^{-4}
13	1.45 x10^{-4}
14	1.60 x10^{-4}

Figure 3.30(b) Contour plots of horizontal deflection for the above specified plate

Fig. 3.31(a) shows the specifications of a connecting plate. The connecting plate is fixed at the side edges and loaded vertically at the hole, as shown in Fig. 3.31(b). For a load of 1000 N, Young's Modulus $0.2\times10^{9}\,MPa$, Poisson's ratio 0.3, the values of deflection are derived using FEA, considering the problem as 2D plane problem with negligible thickness. As the plate is axially loaded upwards, the vertical deflection is given by Eq. (3.25). As explained in first problem, the eccentricity will also give rise to horizontal deflection.

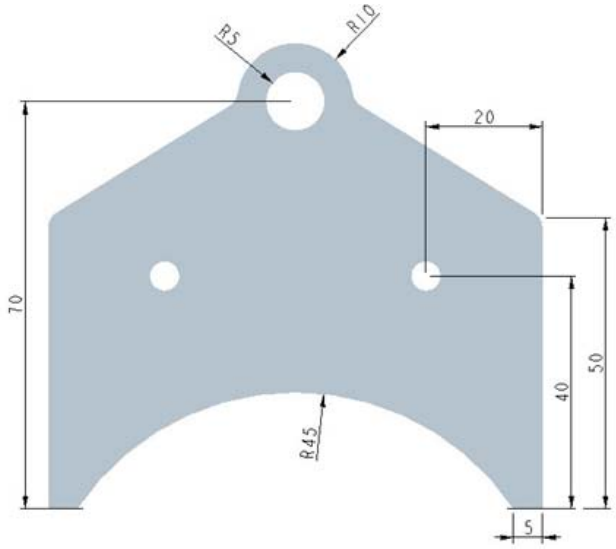


Figure 3.31(a) Specifications of a plate (all dimensions in mm)

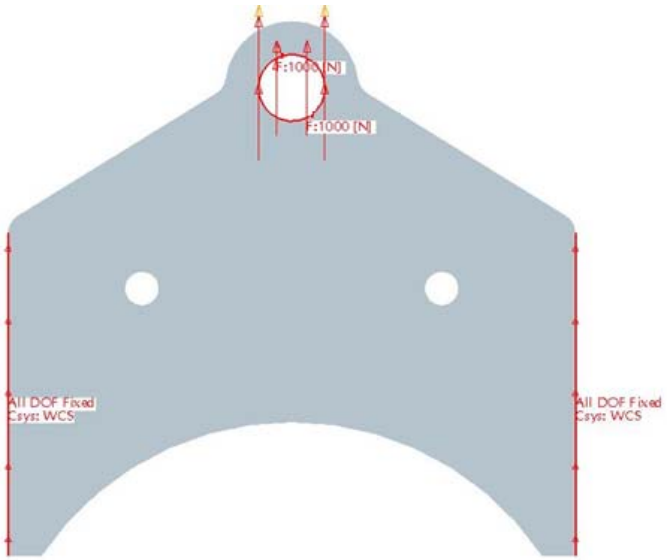
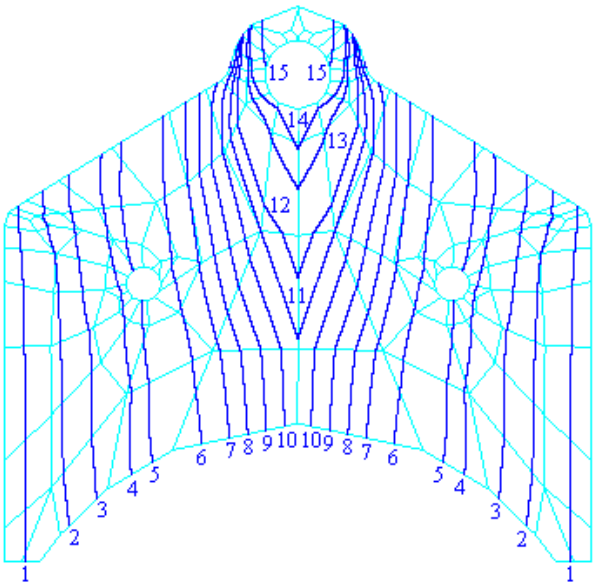


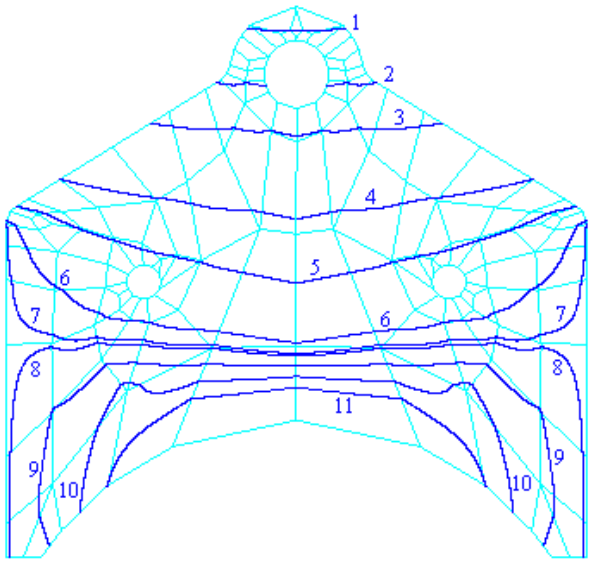
Figure 3.31(b) Boundary conditions and loading of the plate

Fig. 3.32(a) and 3.32(b) show the contour plots of vertical and horizontal deflections using four-node quadrilateral elements for different levels of deflection.



Contour Levels	δ_y
1	0.1 x10^{-4}
2	0.3 x10^{-4}
3	0.5 x10^{-4}
4	0.8 x10^{-4}
5	1 x10^{-4}
6	1.5 x10^{-4}
7	1.8 x10^{-4}
8	2 x10^{-4}
9	2.2 x10^{-4}
10	2.4 x10^{-4}
11	2.6 x10^{-4}
12	2.8 x10^{-4}
13	3.3 x10^{-4}
14	3.6 x10^{-4}
15	5 x10^{-4}

Figure 3.32(a) Contour plots of vertical deflection for the above specified plate



Contour Levels	δ_x
1	8 x10^{-4}
2	3 x10^{-4}
3	2 x10^{-4}
4	1 x10^{-4}
5	0.5 x10^{-4}
6	0.1 x10^{-4}
7	0.01 x10^{-4}
8	-0.01 x10^{-4}
9	-0.1 x10^{-4}
10	-0.2 x10^{-4}
11	-0.3 x10^{-4}

Figure 3.32(b) Contour plots of horizontal deflection for the above specified plate

Fig. 3.33(a) shows the specifications of a connecting plate. The connecting plate is fixed at the centre and loaded vertically and horizontally at the four outer holes, as shown in Fig. 3.33(b). For a load of 1000 N, Young's Modulus $0.2\times10^{9}\,MPa$, Poisson's ratio 0.3, the values of deflections are derived using FEA, considering the problem as 2D plane problem with negligible thickness. As the plate is loaded vertically and horizontally, the deflection is given by Eq. (3.25).

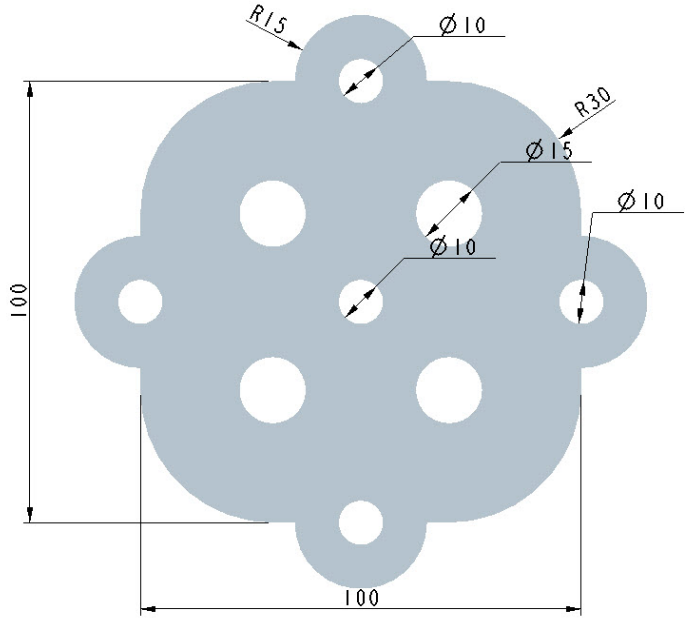


Figure 3.33(a) Specifications of a plate (all dimensions in mm)

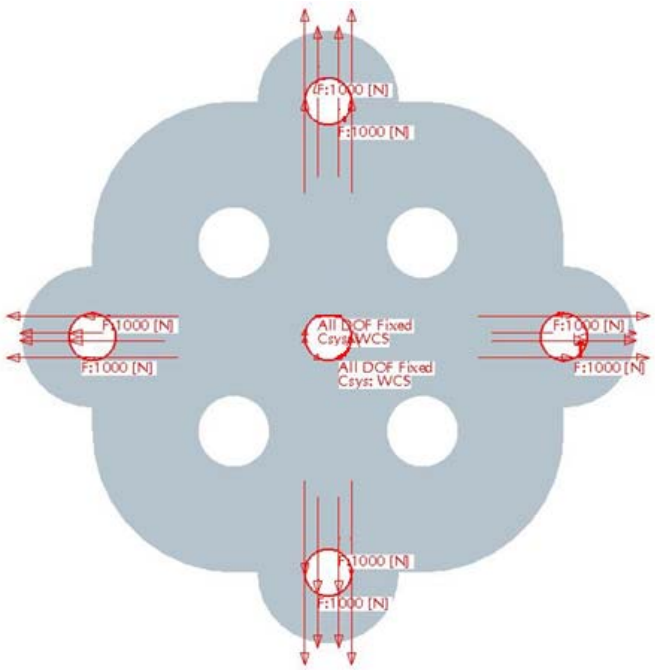
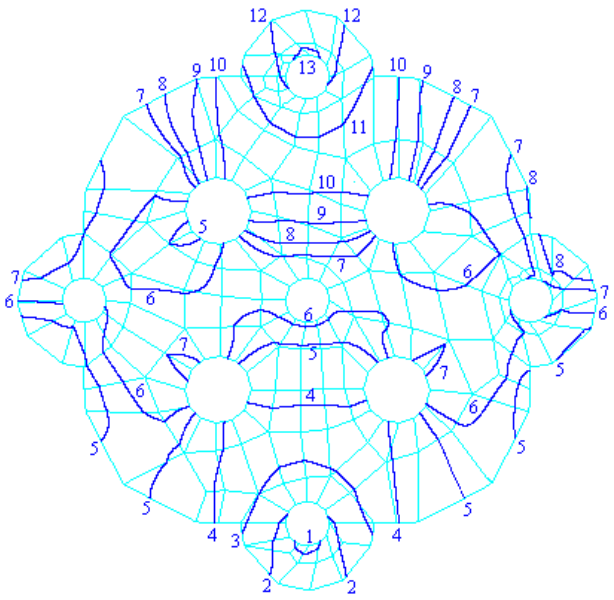


Figure 3.33(b) Boundary conditions and loading of the plate

Fig. 3.34(a) and 3.34(b) show the contour plots of vertical and horizontal deflection using four-node quadrilateral elements for different levels of deflection.



Contour Levels	δ_y
1	-7.5 x10^{-4}
2	-6.0 x10^{-4}
3	-4.0 x10^{-4}
4	-2.0 x10^{-4}
5	-0.5 x10^{-4}
6	-0.1 x10^{-4}
7	0.5 x10^{-4}
8	0.75 x10^{-4}
9	1.2 x10^{-4}
10	2.0 x10^{-4}
11	4.0 x10^{-4}
12	6.0 x10^{-4}
13	7.5 x10^{-4}

Figure 3.34(a) Contour plots of vertical deflection for the above specified plate

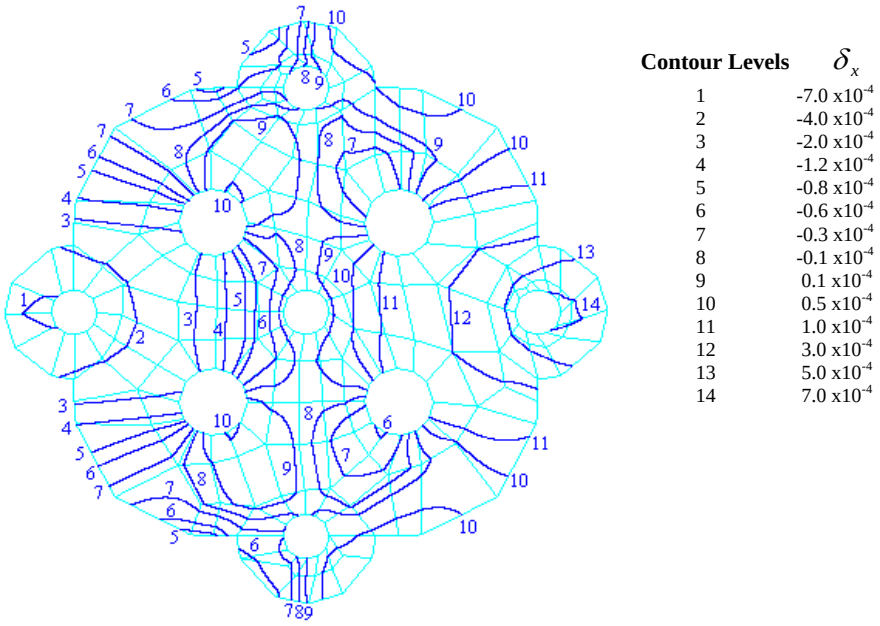


Figure 3.34(b) Contour plots of horizontal deflection for the above specified plate

Fig. 3.35(a) shows the specifications of a safety valve lever. The lever is fixed at two holes on the left hand side and loaded vertically at the right hand side, as shown in Fig. 3.35(b). For a load of 1000 N, Young's Modulus $0.2\times10^{9}\,MPa$, Poisson's ratio 0.3, the values of deflections are derived using FEA, considering the problem as 2D plane problem with negligible thickness. As the lever is loaded downwards, the deflections are given by Eqs. (3.26) to (3.28).

Fig. 3.36(a) and 3.36(b) show the contour plots of vertical and horizontal deflections using four-node quadrilateral elements for different levels of deflection.

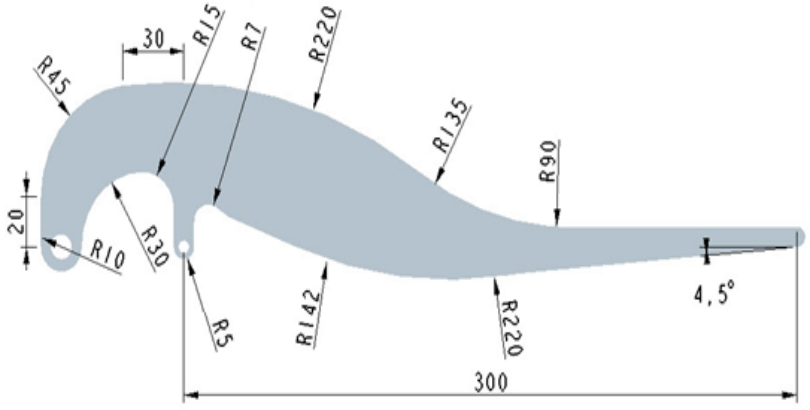


Figure 3.35(a) Specifications of a safety valve lever (all dimensions in mm)

Figure 3.35(b) Boundary conditions and loading of the safety valve lever

Contour Levels	δ_y
1	-120 x10^{-2}
2	-100 x10^{-2}
3	-80 x10^{-2}
4	-40 x10^{-2}
5	-20 x10^{-2}
6	-10 x10^{-2}
7	-5 x10^{-2}
8	-2 x10^{-2}
9	-0.8 x10^{-2}
10	-0.2 x10^{-2}
11	-0.04 x10^{-2}
12	0.05 x10^{-2}
13	0.1 x10^{-2}
14	0.2 x10^{-2}
15	0.3 x10^{-2}

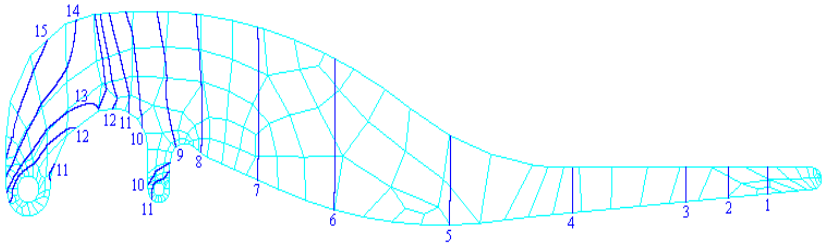


Figure 3.36(a) Contour plots of vertical deflection for the safety valve lever

Contour Levels	δ_x
1	-10 x10^{-2}
2	-7 x10^{-2}
3	-4 x10^{-2}
4	-2.5 x10^{-2}
5	-1.5 x10^{-2}
6	-0.8 x10^{-2}
7	0.1 x10^{-2}
8	1 x10^{-2}
9	1.2 x10^{-2}
10	1.8 x10^{-2}
11	2.5 x10^{-2}

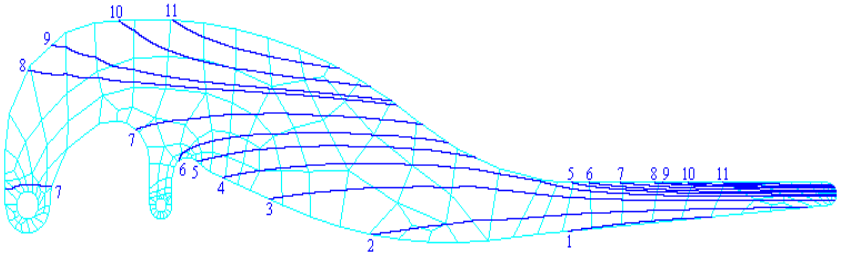


Figure 3.36(b) Contour plots of horizontal deflection for the safety valve lever

The next problem is a heat conduction problem, with the specifications shown in Fig. 3.37(a). The temperature of 673^{0}C and 273^{0}C is fixed at the inside and outside of the shell, as shown in Fig. 3.37(b). For the known heat flow rate, thermal conductivity of the materials as 0.01 W/m-deg and 0.0057 W/m-deg respectively, and the given boundary temperatures, the values of temperature distribution are derived using FEA.

For the n- layer composite concentric cylindrical wall (Kumar, 2008), the Eq. (3.29) is modified to Eq. (3.31).

$$Q = \frac{(t_1 - t_{n+1})}{\sum_{i=1}^{n} \frac{1}{2\pi k_i l} \log_e \frac{r_{i+1}}{r_i}} \tag{3.31}$$

where, l is the length of the cylinder, r_i is the corresponding radius of the cylindrical wall and $\sum_{i=1}^{n} \frac{1}{2\pi k_i l} \log_e \frac{r_{i+1}}{r_i}$ is the sum of the thermal resistances of different layers comprising the composite cylindrical wall. Other parameters are as defined above.

Fig. 3.28 shows the contour plots of temperature distribution using four-node quadrilateral elements for the different levels of temperature.

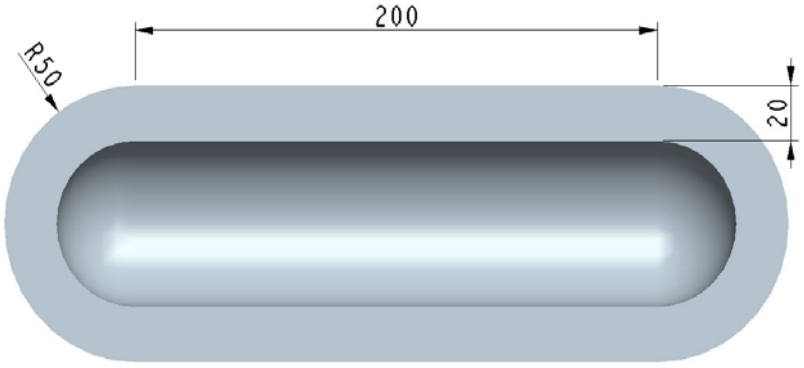


Figure 3.37(a) Specifications of a boiler shell (all dimensions in mm)

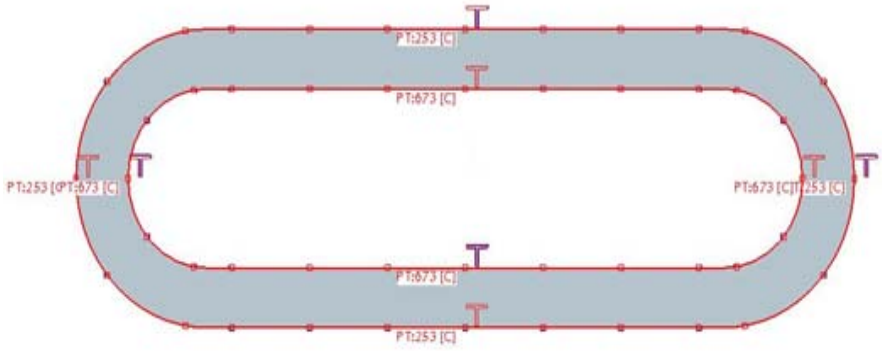


Figure 3.37(b) Boundary conditions and temperature fixation of the boiler shell

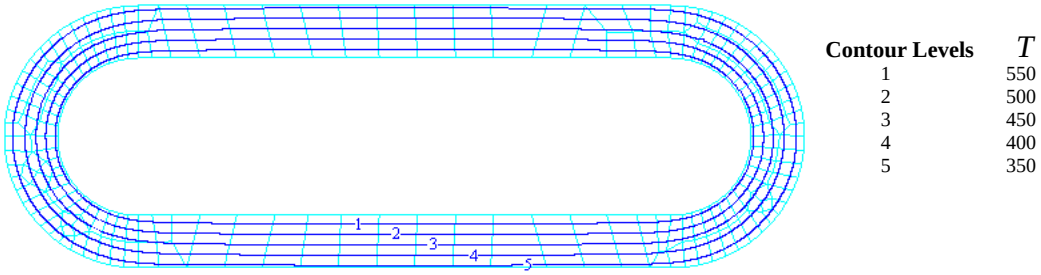


Figure 3.38 Contour plots of temperature distribution (in ^{o}C) for the boiler shell

3.7.1 Speed of the Algorithm

The speed of the discussed algorithm is compared with other, similar techniques (Stelzer and Welzel, 1987; Rajasekaran and Venkatesan, 1995). The algorithms divide the square $[-1,+1]\times[-1,+1]$ into a net of $m\times m$ smaller squares by dividing the r-axis and s-axis into m equal parts and assuming straight lines parallel to the axes, m being arbitrary. The contour Eq. (3.15) is, therefore, solved m^2 times. Thus there is no distinction between a straight line segment and a curved one. Also when the segment intersects a small part of the element, unnecessary computations are performed over the free region. The discussed algorithm is designed to avoid such types of unnecessary computations resulting in drastic reduction in the execution time. The program based on the algorithm takes 30% to 40% less time as taken by the implementation of above said techniques for different problems shown in the present work.

3.8 CONCLUDING REMARKS

Triangular and quadrilateral elements are the most widely used elements for their own respective advantages. In the present chapter, the contour equations are developed using the shape or interpolation function which are same as used in FEA.

In case of triangular element, the developed contour equation represents a line which can be used to quickly plot a contour. The algorithm is mainly used for high speed algorithms but may lead to inaccuracies in results due to the linear variation of field variable.

In quadrilateral element, the contour equation generated using shape function represents a rectangular hyperbola. The decision for joining the contour branches over an element is taken based on asymptotes of a rectangular hyperbola, thus making the algorithm fast.

Thereafter, accuracy of the contour is improved by using the elementary calculus.

To improve the accuracy, higher order elements can be directly used to plot the contours. The next chapter attempts the same.

CHAPTER - IV

CONTOURING IN 2D USING NON-LINEAR ELEMENTS

4.1 INTRODUCTION

Higher-order elements have gained importance since they can be used to represent complex data. Higher-order elements can typically represent the data better when compared to lower-order elements. The accuracy of the results can be improved by using higher order elements. Higher order element tries to achieve a closer approximation to the true value of the unknown parameter (Zienkiewicz, 2002). The higher order elements result in higher-order variations within element, and convergence to the exact solution thus occurs using fewer elements. Wiley *et al.*(2002) showed the potential for substantial reduction in the number of required elements when replacing linear elements with quadratic elements. Another advantage of higher order element is that the curved boundaries of irregularly shaped bodies can be approximated more closely than by the use of simple straight-sided linear elements (Logan, 2008).

It is probably true to say that higher the order of elements used in the analysis, the more the useful it is to produce the contour data, because within each element more variation is allowed. However, generation of contour lines using higher order elements is a complex task. In order to simplify the contour surface generation, normally a higher order element is split into linear elements and linear interpolation of the physical quantity being interpreted is performed. The linear approximation of a non-linear function creates inaccuracies in results which are unacceptable in many situations, particularly when the variation in physical quantity is large in a small neighbourhood.

In this chapter, an attempt is made to use higher order interpolation using a triangular and quadrilateral element. These elements provide a non-linear interpolation function which is used to accurately interpolate the physical quantity without any loss of information.

The contour equation is developed over these elements using the shape functions. The developed contour equation is quadratic over six-node triangle and eight-node quadrilateral elements. The contour equation is cubic in nature over ten-node triangular element.

4.2 CONTOUR EQUATION OVER SIX-NODE TRIANGLE

A quadratic interpolation can be performed by using a triangular element and by taking three more discrete values of function on the middle of the three sides, as shown in Fig. 4.1. This is called a six-node triangular element. The words *side* and *edge* interchangeably have been used throughout the chapter.

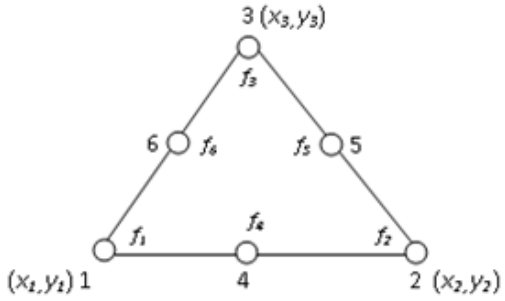


Figure 4.1 A six-node triangular element

The interpolation is performed as shown in Eq. (4.1) to Eq. (4.3).

$$x = \sum_{i=1}^{3} N_i x_i \tag{4.1}$$

$$y = \sum_{i=1}^{3} N_i y_i \tag{4.2}$$

$$f(x, y) = \sum_{i=1}^{6} N_i f_i \tag{4.3}$$

where $N_i = L_i,\ i = 1,2,3$ and $N_i = 4L_{i-3}L_{i-2}, i = 4,5,6$ with $L_4 = L_1$.

Linear interpolation for contour plotting has been investigated by many researchers as discussed in chapter II. In fact, it is the simplest interpolation technique which provides C^0 continuity in the entire domain. A six-node interpolation using a triangular element has been in use in FEA for a long time as it provides quadratic interpolation and requires less number of elements for the same order of approximation in solution using finite element technique. However, in order to plot contours, a six-node element is divided into 4 three-node triangular elements and linear interpolation is performed to plot contours. Although it simplifies contour plotting, inaccuracies due to lower order interpolation are introduced. This degrades the visual presentation of results in regions where the function changes rapidly or where contours are closely placed. Moreover, the benefits due to higher order approximations in finite element results are not reflected in visual

presentation of results. Motivated by these considerations, the exact tracing of contours has been performed over a six-node element using quadratic interpolation. The method is explained as follows.

The contour equation assumes the following form over a six-node triangular element

$$\sum_{i=1}^{6} N_i f_i = C \tag{4.4}$$

After substituting the values of shape function into Eq. (4.4), the Eq. (4.4) is simplified to Eq. (4.5).

$$ax^2 + 2hxy + by^2 + 2gx + 2fy + c = 0 \tag{4.5}$$

where,

$$a = \frac{1}{\Delta^2}\left[b_1^2 f_1 + b_2^2 f_2 + b_3^2 f_3\right] + \frac{1}{\Delta^2}\left[b_1 b_2 f_4 + b_2 b_3 f_5 + b_1 b_3 f_6\right], \tag{4.6}$$

$$b = \frac{1}{\Delta^2}\left[c_1^2 f_1 + c_2^2 f_2 + c_3^2 f_3\right] + \frac{1}{\Delta^2}\left[c_1 c_2 f_4 + c_2 c_3 f_5 + c_1 c_3 f_6\right], \tag{4.7}$$

$$c = \frac{1}{2\Delta^2}\left(a_1^2 - \Delta a_1\right) f_1 + \frac{1}{2\Delta^2}\left(a_2^2 - \Delta a_2\right) f_2 + \frac{1}{2\Delta^2}\left(a_3^2 - \Delta a_3\right) f_3 + \frac{1}{\Delta^2}\left(a_1 a_2\right) f_4 + \frac{1}{\Delta^2}\left(a_2 a_3\right) f_5 + \frac{1}{\Delta^2}\left(a_1 a_3\right) f_6, \tag{4.8}$$

$$h = \frac{1}{2\Delta^2}\left[2b_1 c_1 f_1 + 2b_2 c_2 f_2 + 2b_3 c_3 f_3\right] + \frac{1}{2\Delta^2}\left[\left(c_1 b_2 + b_1 c_2\right) f_4 + \left(c_2 b_3 + b_2 c_3\right) f_5 + \left(c_1 b_3 + b_1 c_3\right) f_6\right], \tag{4.9}$$

$$g = \frac{1}{4\Delta^2}\left[\left(2a_1 b_1 - \Delta b_1\right) f_1 + \left(2a_2 b_2 - \Delta b_2\right) f_2 + \left(2a_3 b_3 - \Delta b_3\right) f_3\right] + \frac{1}{2\Delta^2}\left[\left(b_1 a_2 + a_1 b_2\right) f_4 + \left(b_2 a_3 + a_2 b_3\right) f_5 + \left(b_1 a_3 + a_1 b_3\right) f_6\right], \tag{4.10}$$

and

$$f = \frac{1}{4\Delta^2}\left[\left(2a_1 c_1 - \Delta c_1\right) f_1 + \left(2a_2 c_2 - \Delta c_2\right) f_2 + \left(2a_3 c_3 - \Delta c_3\right) f_3\right] + \frac{1}{2\Delta^2}\left[\left(c_1 a_2 + a_1 c_2\right) f_4 + \left(c_2 a_3 + a_2 c_3\right) f_5 + \left(c_1 a_3 + a_1 c_3\right) f_6\right], \tag{4.11}$$

Eq. (4.5) is the equation of general conics which represents curves in 2D plane depending on the values of the coefficients *a* through *f*. Let α be given by Eq. (4.12).

$$\alpha = \begin{vmatrix} a & h & g \\ h & b & f \\ g & f & c \end{vmatrix} \tag{4.12}$$

The various conics are given in Table 4.1 for different values of α, a, b and h.

Case	**Conditions**	**Conic Section**
1	$\alpha \neq 0$, $a = b$ and $h = 0$	Circle
2	$\alpha \neq 0$, $h^2 < ab$	Ellipse
3	$\alpha \neq 0$, $h^2 = ab$	Parabola
4	$\alpha \neq 0$, $h^2 > ab$, $a + b \neq 0$	Hyperbola
5	$\alpha \neq 0$, $h^2 > ab$, $a + b = 0$	Rectangular Hyperbola
6	$\alpha = 0$, $h^2 \neq ab$	Pair of straight lines
7	$\alpha = 0$, $h^2 = ab$	Pair of parallel straight lines
8	$\alpha = 0$, $a + b = 0$	Pair of perpendicular st. lines

Table 4.1 Various conic sections represented by a quadratic equation in 2D

The problem now reduces to accurately tracing the contour segment represented by a conic section over a triangle. This is achieved by finding the intersection of the sides of the triangle with that of a conic section.

The algorithm discussed in section 3.3 of chapter III is used for plotting of contours. Except for steps (ii), (iv) and (vii), all steps are easy to understand. Detailed explanations are given for these steps in the following sections.

4.2.1 Finding the Minimum and the Maximum of *f(x, y)* over the Element

In general, the minimum and the maximum for a quadratic equation can be found by finding the point of extremum by taking the partial derivatives of the function given by Eq. (4.5) w.r.t. x and y and equating them to zero, *i.e.,*

$$\frac{\partial f}{\partial x} = 2ax + 2hy + 2g = ax + hy + g = 0 \tag{4.13}$$

$$\frac{\partial f}{\partial y} = 2hx + 2by + 2f = hx + by + f = 0 \tag{4.14}$$

Let the solution of Eq. (4.13) and (4.14) be (x_e, y_e) and the value of the function *f(x,y)* as expressed by Eq. (3.3) at (x_e, y_e) be f_e. Now we find f_{min} and f_{max} by finding the minimum and the maximum of the values f_e and f_1, f_2 ….. *and* f_6. It can be noted here that we are simply interested here in deriving the extremum value at (x_e, y_e) notwithstanding the fact whether it is a point of the maximum or the minimum.

4.2.2 Tracing Contour Segment over an Element

The contour segment represented by Eq. (4.5) can have one of the eight possible conic sections. They are divided into four categories – a single closed curve (circle / ellipse), a single open curve (parabola), a pair of two curves (hyperbola / rectangular hyperbola) and a pair of straight lines. The tracing of contours is explained as follows:

i) Circle / Ellipse:

The contour segment turns out to be a circle when $\alpha \neq 0$, $a = b$, $h = 0$ or an ellipse when $\alpha \neq 0$, $h^2 < ab$.

The intersection of a circle with one of the sides of the triangle has six possible cases as shown in Fig. 4.2(a) through Fig. 4.2(f). The six cases of intersection for a side with nodes 1 and 2 are: (a) the circle intersects at two points of the side, (b) one of the two intersection points is outside the side, (c) both the intersection points are outside, (d) the side does not intersect but the center of the circle is to the left of the side, (e) the side does not intersect but the centre of the circle is to the right of the side, and (f) the circle touches the circle and the centre is to the right of the side. The last case, if detected, is discarded for contour plotting as it yields a single point which is of no interest. A circle remains completely inside a triangle when none of the sides intersects the circle and the center of the circle is to the left of all sides, a case represented by Fig. 4.2(d). If all the sides do not intersect but the center is to the right of at least one side then we are confronted with the case represented by Fig. 4.2(e). The case of an edge touching a circle is discarded (shown in Fig. 4.2(f)) as it will give a single point on the contour. This does

not affect the tracing of the contour irrespective of the fact that the circle intersects the other two edges or not.

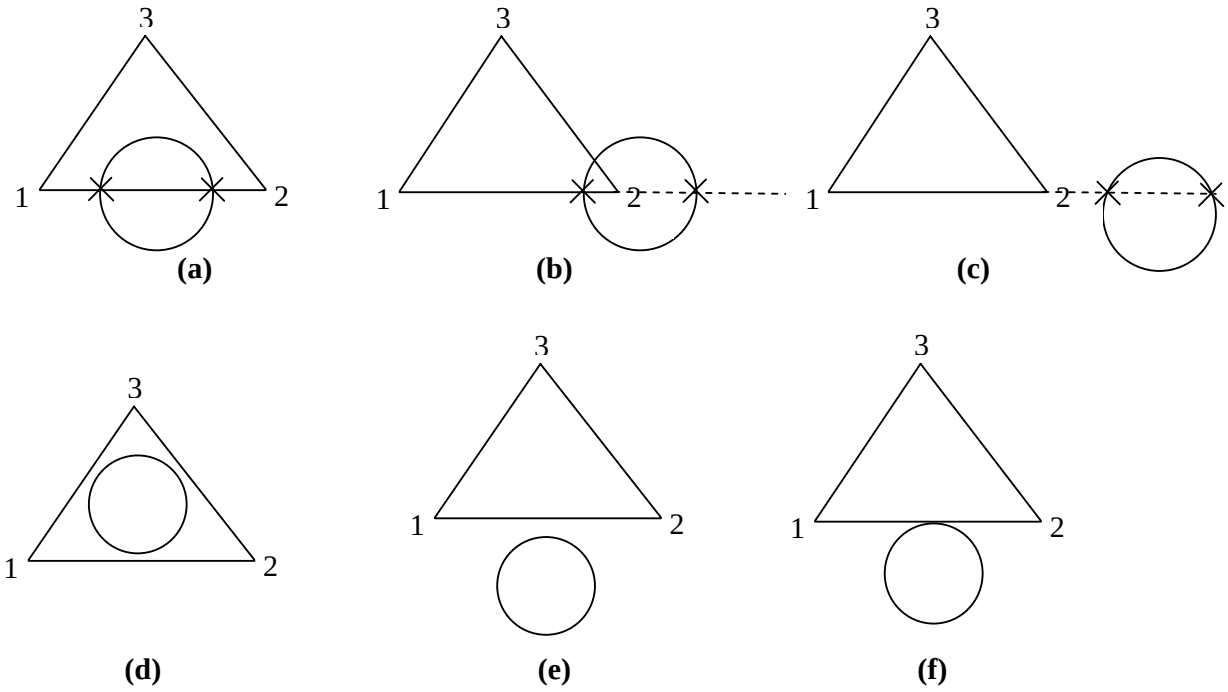


Figure 4.2 Possible cases of intersection of a side with nodes 1 and 2 with the circle: a) the circle intersects at two points of the sides; b) one of the two intersection points is outside the side; c) both the intersection points are outside; d) the side does not intersect but the center of the circle is to the left of the side; e) the side does not intersect but the center of the circle is to the right of the side; f) the circle touches the circle and the center is to the right of the side

In general, we find intersection of a circle with a side of a triangle and record the point of intersection, if any. Let the side be given by Eq. (4.15).

$$y = mx + n \tag{4.15}$$

The general equation of the circle be given by Eq. (4.16).

$$x^2 + y^2 + 2Gx + 2Fy + H = 0 \tag{4.16}$$

On substituting y from Eq. (4.15) into Eq. (4.16), we get Eq. (4.17).

$$Ax^2 + Bx + C = 0 \tag{4.17}$$

where, $A = 1 + m^2$, $B = -2m^2 x_1 + 2y_1 m + 2G + 2Fm$ and

$$C = y_1^2 + m^2 x_1^2 - 2y_1 m x_1 + 2Fy_1 - 2Fmx_1 + H$$

Eq. (4.17) has two real roots, if $B^2 - 4AC \geq 0$, otherwise both the roots will be complex. A point (x, y) is to the left of line segment if the expression $f(x, y) = (x_2 - x_1)(y - y_1) - (y_2 - y_1)(x - x_1)$ is positive. The ordering of the nodes is in counterclockwise direction.

The case of a contour being an ellipse can be processed in a similar manner.

ii) Parabola

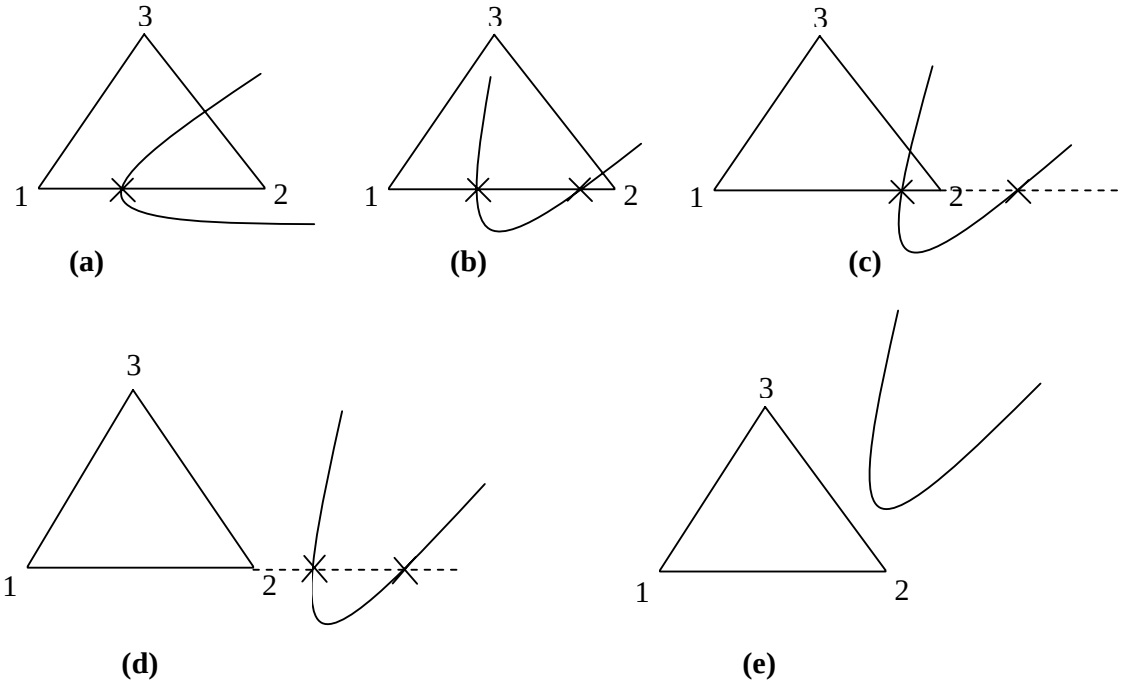


Figure 4.3 Possible cases of intersection of a side with nodes 1 and 2 with a parabola:

a) repetitive roots and roots lying within the side; b) nonrepetitive roots, both lying within the side; c) one root within the side; d) both roots outside the side; and e) the side does not intersect the parabola (complex roots)

The equation of a parabola also turns out to be a quadratic equation which also has either two real roots or two complex roots. When the roots are complex, the parabola does not intersect a side. The real roots may either be repetitive or nonrepetitive. Fig. 4.3(a)-4.3(e) depicts all possible cases of the intersection of the side with nodes 1

and 2 with the parabola. Fig. 4.3(a) represents the case of repetitive roots and the roots lie within the side. Fig. 4.3(b) through Fig. 4.3(d) represents the case of distinct roots. As in the case of a circle, we discard repetitive roots which represent the case of the side touching the parabola.

iii) Hyperbola and rectangular hyperbola

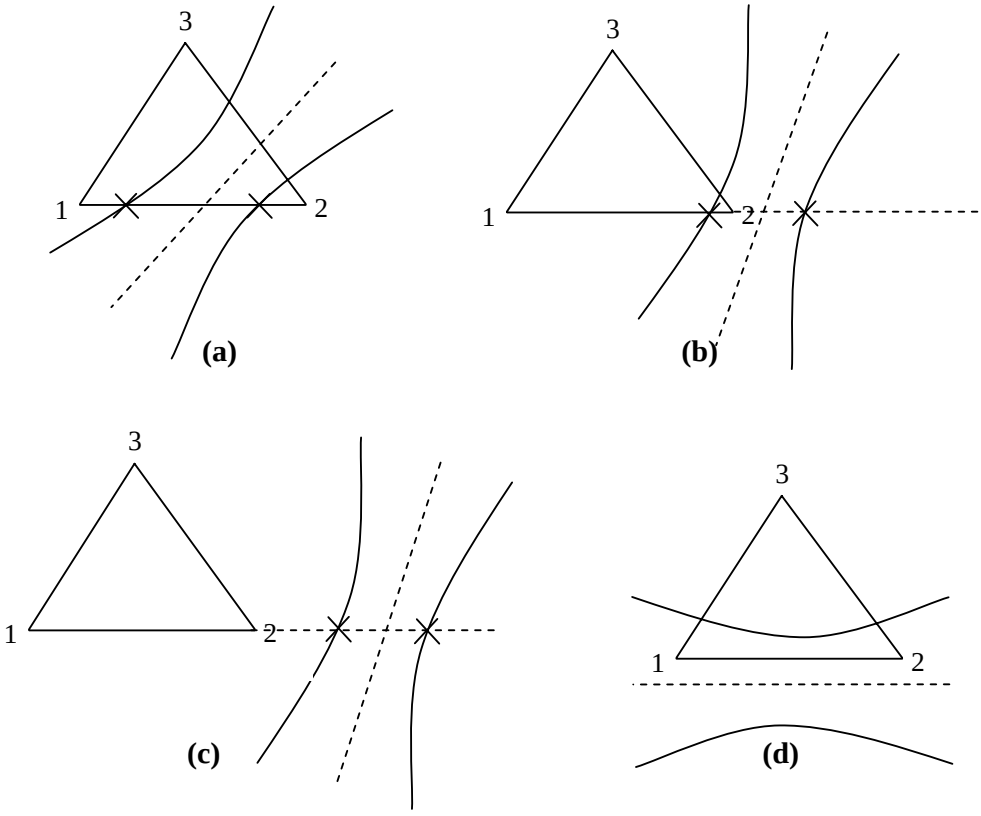


Figure 4.4 Possible cases of intersection of a side with nodes 1 and 2 with a hyperbola:

a) real roots when two roots lie within the side; b) real roots when one of the roots lie within the side; c) when no root lies within the side; d) no real root exists

These are also characterized by quadratic polynomial equations which have either two real roots or two complex roots. The possible cases of intersection by a side of the triangle with the hyperbola are given in Fig. 4.4(a) through Fig. 4.4(d). Fig 4.4(a), 4.4(b) and 4.4(c) represent, respectively, the cases of real roots when two roots, one root and no root lie within the side. Fig. 4.4(d) represents the case when no real root exists. A rectangular hyperbola also depicts a similar characteristic. As before, the case when roots are repetitive is discarded.

vi) Pair of Lines

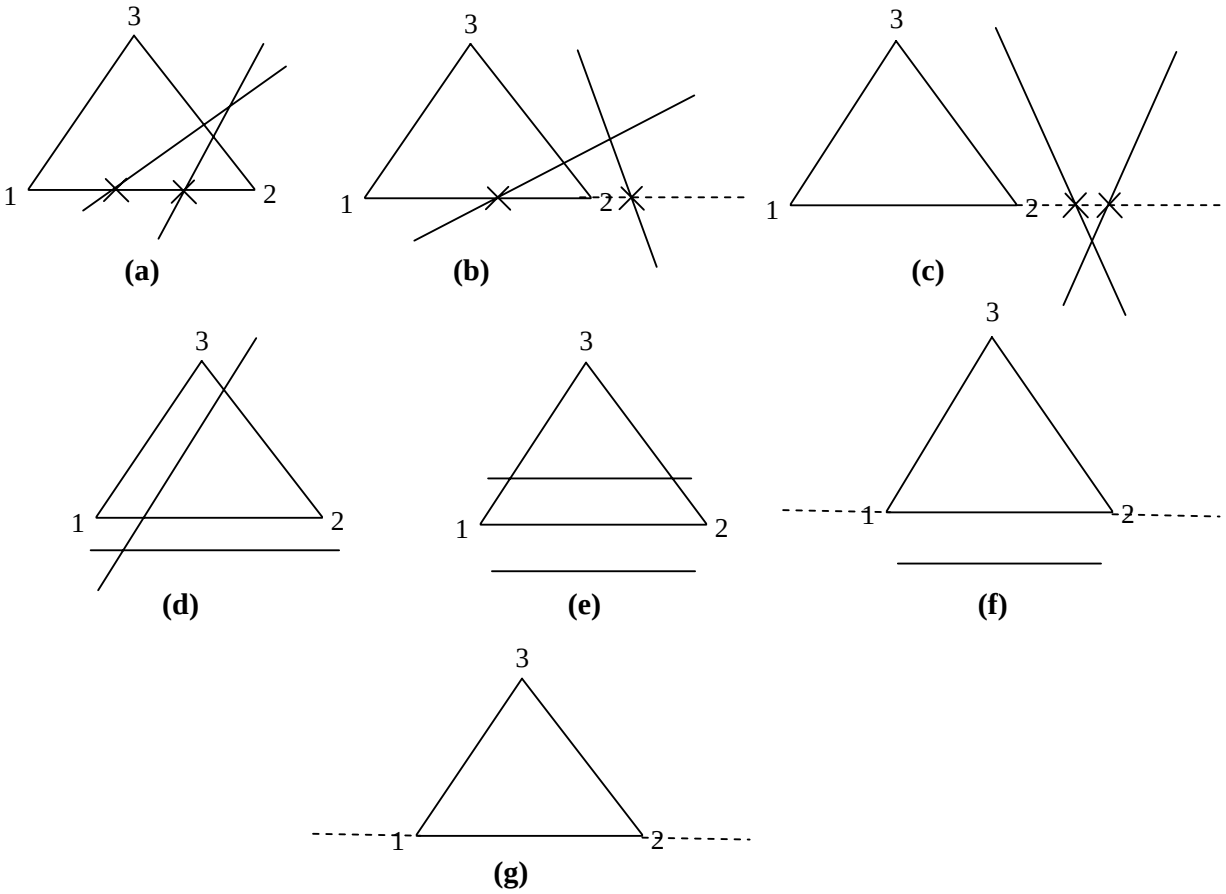


Figure 4.5 Possible cases of intersection of a side with nodes 1 and 2 with a pair of straight lines: a) both contour lines intersect the inner part of the side; b) only one line intersects the inner part; c) both the intersections lie on the extended part of the side; d) one of the contour lines is parallel to the edge; e) both contour lines are parallel to the edge; f) one of the contour lines overlaps the edge; g) both contour lines overlap the edge

The quadratic equation representing the conic section in such a case yields two linear equations which can be written as

$$\left(p_1 x + q_1 y + r_1\right)\left(p_2 x + q_2 y + r_2\right) = 0, \tag{4.18}$$

i.e., $p_1 x + q_1 y + r_1 = 0$ and $p_2 x + q_2 y + r_2 = 0$. When we find intersection with a side of triangle, we get a real root for each of the two lines. Thus there are either two or one intersection points which may fall within the side of the triangle or intersect the extended

part of the side. All possible cases of intersection of a side with nodes 1 and 2 of the triangle with a pair of straight line contour branches are depicted in Fig. 4.5(a)-4.5(g). The possible cases include: (a) both contour lines intersect the inner part of the side, (b) only one line intersects the inner part, (c) both the intersections lie on the extended part of the side, (d) one of the contour lines is parallel to the edge, (e) both contour lines are parallel to the edge, (f) one of the contour lines overlaps the edge and (g) both contour lines overlap the edge. There may be a case when both or one line are/is parallel to the side or may overlap it.

After identifying the type of a conic section and deriving the intersections with all edges, the next task is to trace the contour over the element. The various conic sections are dealt with separately.

Tracing individual conic contour segment

i) Circle / Ellipse: The total number of intersections varies from 2 to 6. In fact a circle will form contour segment(s) over the triangle when the number of intersections is 2, 4, or 6. Fig. 4.6(a) through Fig. 4.6(e) depicts the five cases when the number of intersections is 2 (Fig. 4.6(a) and Fig. 4.6(b)), 4 (Fig. 4.6(c) and (d)), and 6 (Fig. 4.6(e)). None of the other cases, i.e., the number of intersections being 1, 3, or 5, exists, excluding the case when a circle lies completely inside or completely outside the triangle. It is due to the reason that for finding the intersection of side with the circle we solve a quadratic equation. The number of real roots will be either 0 or 2. In case the circle touches a side, there will be two repeated roots. Therefore, the total number of intersections with the three sides can be either 0 or 2 or 4 or 6. We move counterclockwise direction along the sides of the triangle and arrange the coordinates of the intersection points accordingly. The intersection points and their coordinates are represented by P_i and (x_i, y_i), $i = 1,2,.....6,$ respectively. Contour segments are traced according to number of intersections in the intervals as given below.

Number of intersections 2: interval is $[1,2]$

Number of intersections 4: intervals are $[2,3]$ *and* $[4,1]$

Number of intersections 6: intervals are $[2,3]$, $[4,5]$ *and* $[6,1]$

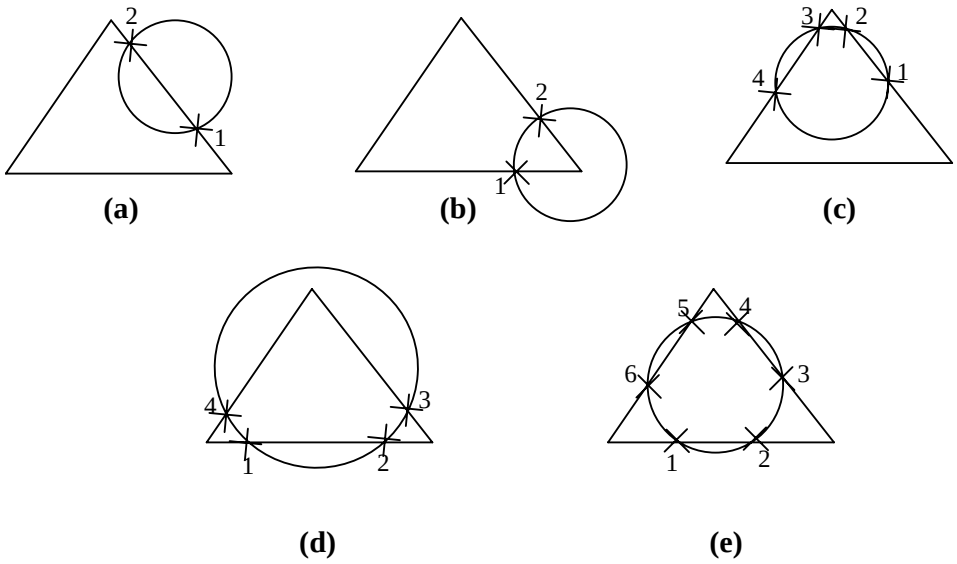


Figure 4.6 Possible cases of circular arcs intersecting the edges of a triangle: a) two intersections with an edge; b) two intersections with two edges; c) four intersections with two edges; d) four intersections with two edges; e) six intersections with three edges. The case of circle lying completely inside or completely outside the triangle is not included

Let $dx = x_2 - x_1$ and $dy = y_2 - y_1$. If $|dx| \geq |dy|$, then starting from $x = x_1$ we increment x by dx/N. Here N is a predetermined integer number which represents the number of straight line contour segments which approximate the circular contour arc and compute y and plot the point or save it. The steps are summarized as

(i) Let $x \leftarrow x_1$, $y \leftarrow y_1$ and $\Delta x \leftarrow dx/N$. Plot (x,y).

(ii) While $x < x_2$

(a) $x \leftarrow x + \Delta x$

(b) Compute y from equation of circle. In general there will be two values of y, say, $y = \alpha$, $y = \beta$. We choose that value of y for which $y_1 \leq y \leq y_2$. Let this value be $y = \alpha$.

(c) Plot (x,y).

The case $|dy| > |dx|$ can be handled in a similar way. When a circle is within the triangle completely, the complete circle can be plotted using a fast algorithm, such as

Bresenham's circle algorithm (Hearn and Baker, 2002), which works on integer arithmetic. In addition, an eight-way symmetry of circle can be used to enhance the speed of tracing.

ii) Parabola: The possible cases of intersection of the edges of the triangle with parabolic segment of contours are shown in Fig. 4.7 (a) through Fig. 4.7(d). The number of roots can be 2 or 4. When the number of roots is 2, we trace the contour between the roots. When the number of roots is 4, we have two possible cases as shown in Fig. 4.7(c) and Fig. 4.7(d). Fig. 4.7(c) represents the case when two intersections lie on two edges each. In such a case we have two contour segments lying inside the triangle. The segments are traced by numbering the intersections in anticlockwise directions and numbering intersections 1 and 2 on the first edge and 3 and 4 on the second edge. The segments are then traced in the intervals [1,4] and [2,3]. The second case of four-intersections arises when an edge has two intersections and the remaining two intersections lie on two other edges of the triangle as shown in Fig. 4.7(d). The two contour segments are traced by ordering the intersections in anticlockwise direction starting the numbering process from the two intersections lying on the same edge as shown in Fig. 4.7(d). The two contour segments then also lie in the intervals [1,4] and [2,3].

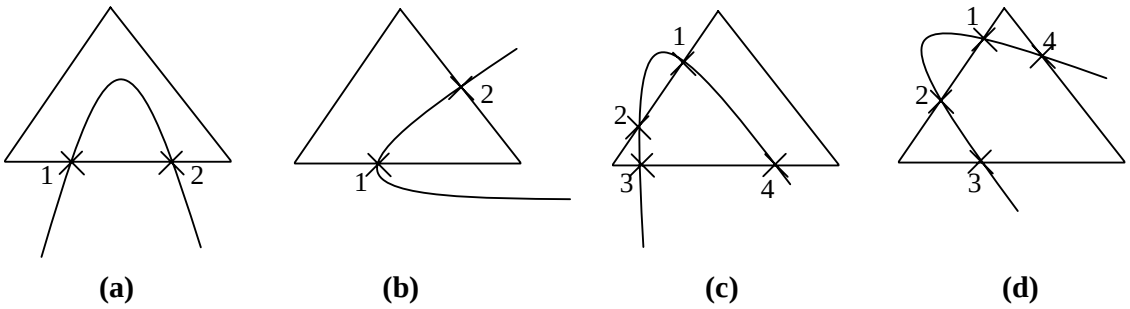


Figure 4.7 The four cases of parabolic contour segment(s) lying over a triangle: a) two intersections lying on an edge; b) two intersections lying on two edges; c) four intersections lying on two edges; and d) four intersections lying on three edges

iii) Hyperbola: There are four possible cases of intersection of a hyperbola with a triangle as represented in Fig. 4.8(a)-4.8(d). The number of intersections can be 2 or 4. A contour segment is traced in the interval [1,2] when the number of intersections is 2. When the number of intersections is 4, we can have ordering as given in Fig. 4.8(c) or 4.8(d), *i.e.*, the two contour segments may lie either in the intervals [1,2] and [3,4] or in

the intervals [1,4] and [2,3]. These two cases can be differentiated on the basis of locating a contour point in the interval [1,2] (or [1,4]) and determining whether the point lies inside the triangle or not.

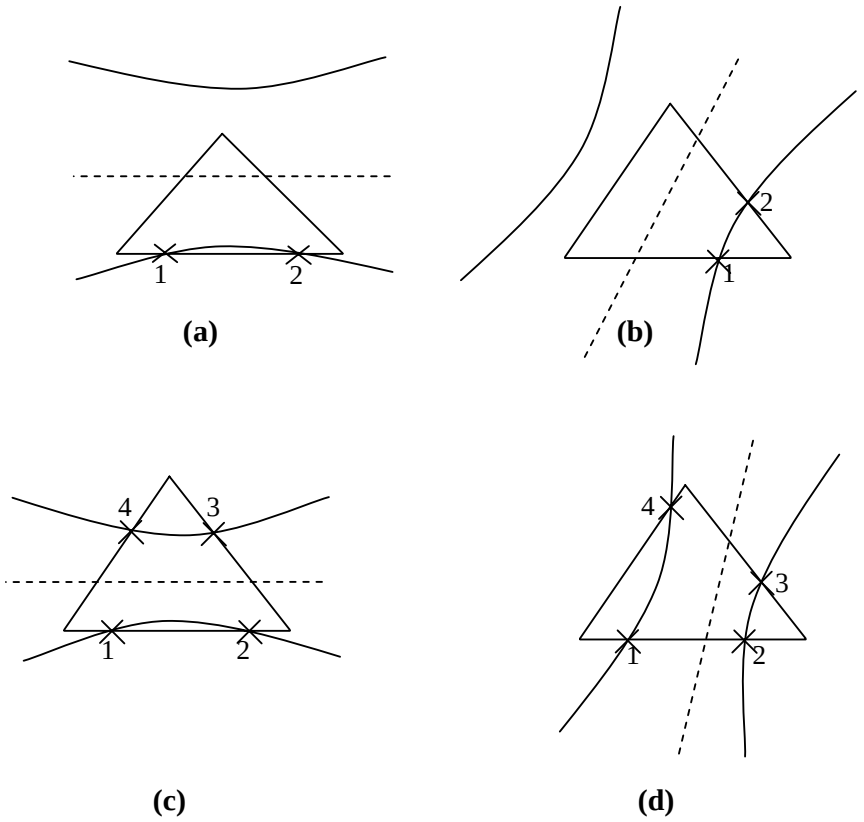


Figure 4.8 The four cases of hyperbola contour segment(s) lying over a triangle: a) two intersections lie on an edge; b) two intersections lie on two edges; c) & d) four intersections lie on three edges but the numbering on intersections differ

iv) Pair of straight lines: There are three possible cases having the number of intersections 2 or 4 as shown in Fig. 4.9(a)-4.9(c). As before, tracing a line contour segment over a triangle is simple when the number of intersections is 2. The contour line lies in the interval [1,2]. When the number of intersections is 4, the contour line segments may lie either in the intervals [1,4] and [2,3] or in the intervals [1,2] and [3,4]. These two cases can be identified by deriving the coordinates of a contour point in a given interval and determining whether the contour point lies inside the triangle or not.

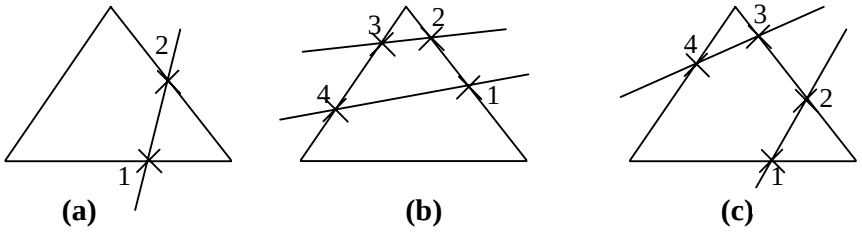


Figure 4.9 The three possible cases of a pair of line contour segments intersecting the triangle: a) two intersections on two edges; b) four intersections on two edges; and c) four intersections on three edges

4.2.3 Joining the Contour Segments

The next task is to join contour segments of same level in order to plot a complete contour line from one end to another. This provides a fast way of contour lines over a 2D domain. If the contour lines are plotted elementwise, then the processing time would be too large. We develop data structures representing the contour segments and provide an algorithm for joining contour segment of same level.

Data structure: A contour segment is represented as a record with the following fields:

C_l	:	contour level
n	:	number of points in contour segment
$P_1, P_2, ..., P_n$	:	n contour points over the segment arranged in an order. The points are arranged in increasing order of their x-coordinates and if the x-coordinates of two points are same, then the points are ordered in their increasing y-coordinates
NP[i]	:	number of points in the *i-th* segment

Fig. 4.10 represents a contour segment over an element with the above information.

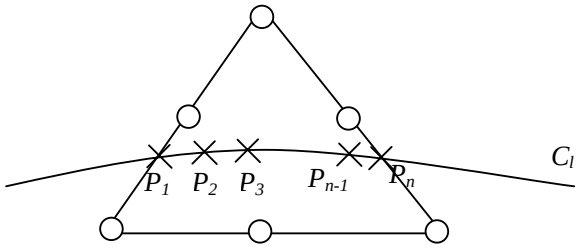


Figure 4.10 Representation of a contour segment over an element

Partitioning contour segments: We perform search operation for joining contour segments having same levels. Two contour segments are joined when the distance between either of their end points is zero. Since floating point operations are involved in determining the coordinates of end points, we take a small number, say 10^{-6}, represented by the symbol *EPS*, instead of zero value and square of the distance is compared with this number to avoid square root operation.

Joining contour segments involves $0(M^2)$ operation, where M is the total number of contour segments. We partition contour segments based on their levels thus reducing the number M to M/L on average, where L is the number of contour levels. This reduces the search space by one-tenth of total space. The following algorithm describes the steps for joining contour segments within a level.

Algorithm: It is assumed that the contour segments are partitioned in L buffers, where L is the total number of contour levels. Let M be the total number of contour segments in a buffer. There can be at the most M distinct contour lines indicating that none of the segments joins with others. The algorithm proceeds by choosing a segment from the buffer which is the first segment of a contour line. A matching segment is searched from the remaining segments in the buffer. If a match is found, it is added with the current contour line forming a new line by updating its list of points. This process is repeated until no match is formed which indicates that a complete contour line is traced. There may be a few contour segments left in the buffer which have not found a match indicating that another contour line exists which can be traced in a similar fashion. The process is repeated until all segments in the buffers are exhausted.

We use a Boolean array $A[i], i=1, 2,\ldots, L$, which is initialized to *false* implying thereby that none of the contour segments has formed a match. The variables *NSEG* save the number of unprocessed segments which is initialized to M and it is decremented by 1

everytime a match is found. A contour line starts building up with an unprocessed *i-th* contour segment determined by the value of the Boolean array *A[i]* which is *false*. Matching is formed by taking the square of distance between the two end points of the contour line with the end points of an unprocessed contour segment. If a match is found the corresponding element of the Boolean array is set *true*. Let the points P_1^l and P_n^l be the two end points of contour line and P_1^s and P_n^s be the end points of contour segment. The variables n_l and n_s represent the number of points in a contour line and in a contour segment, respectively. The joining of a contour line with a contour segment is represented in Fig. 4.11(a) through Fig. 4.11(d) for all the four possible cases:

(i) The *HEAD* of the list of points of contour line, P_1^l, joins with the *HEAD* of the list of points of contour segment, P_1^s. The updated contour line has the new list of points, $P_n^s....P_1^s P_1^l....P_n^l$ (Fig. 4.11(a)).

(ii) The *HEAD* of the list of points of contour line, P_1^l, joins with the *FRONT* of the list of points of contour segments, P_n^s. The updated contour line has the new list of points, $P_1^s....P_n^s P_1^l....P_n^l$ (Fig. 4.11(b)).

(iii) The *FRONT* of the list of points of contour line, P_n^l, joins with the *HEAD* of the list of points of contour segment, P_1^s. The updated contour line has the new list of points, $P_1^l....P_n^l P_1^s....P_n^s$ (Fig. 4.11(c)).

(iv) The *FRONT* of the list of points of contour line, P_n^l, joins with the *FRONT* of the list of points of contour segment, P_n^s. The updated contour line has the new list of points, $P_1^l....P_n^l P_n^s....P_1^s$ (Fig. 4.11(d)).

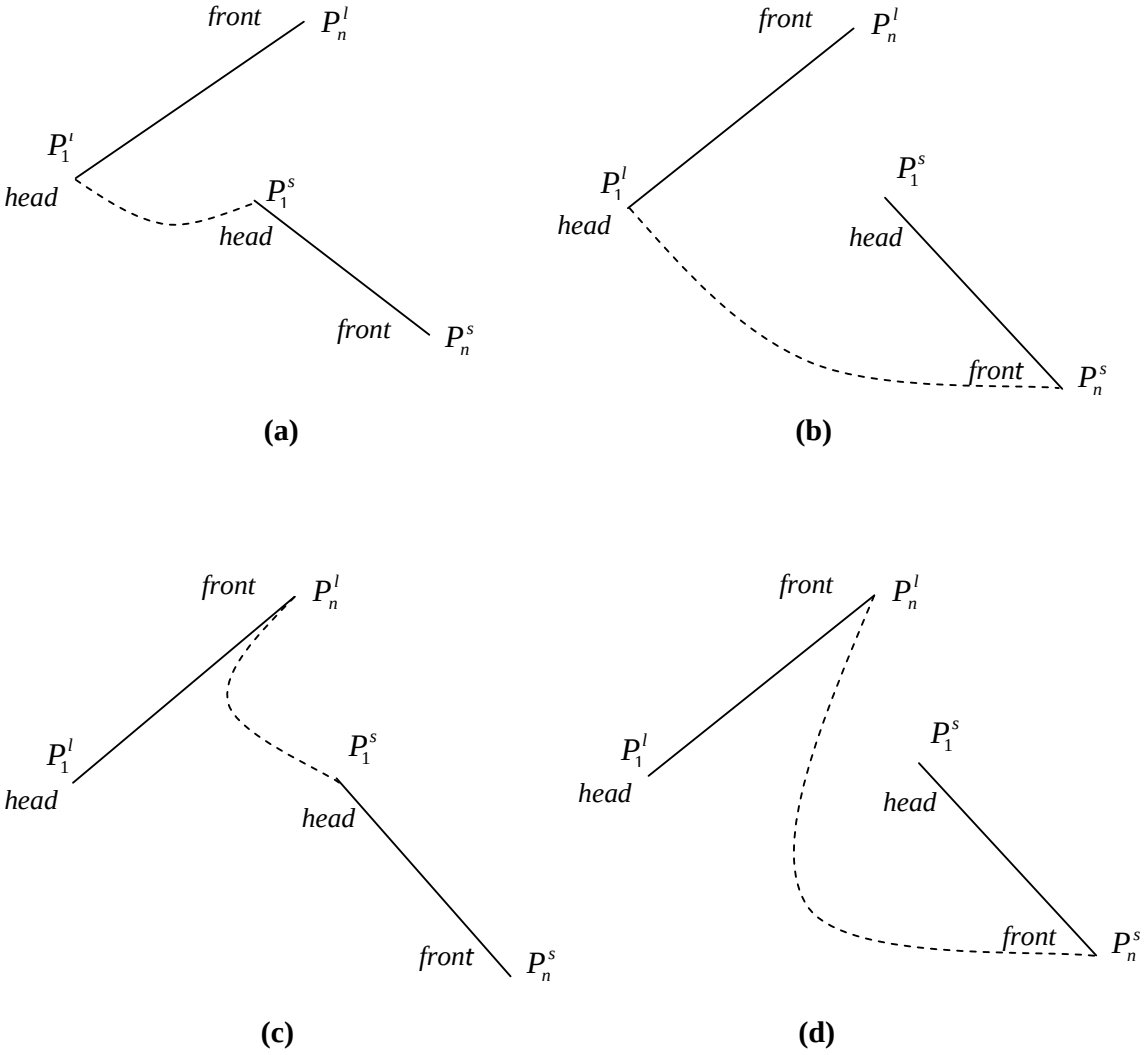


Figure 4.11 The four possible cases of joining of a contour line with a contour segment (a) *HEAD* joins with *HEAD*, (b) *HEAD* joins with *FRONT*, (c) *FRONT* joins with *HEAD*, and (d) *FRONT* joins with *FRONT*

The psuedocode for the algorirthm is given as follows:

(1) $A[i] \leftarrow false$, $i = 1,2,....,M$; $NSEG \leftarrow M$; $l \leftarrow 0$

(2) *While* $NSEG > 0$

$l \leftarrow l + 1$

If $A[l] = false$

$flag \leftarrow 1$

While $flag = 1$

$flag \leftarrow 0$

$A[l] = true;\ x_1 \leftarrow P_1^l.x;\ x_2 \leftarrow P_n^l.x;\ y_1 \leftarrow P_1^l.y;\ y_2 \leftarrow P_n^l.y;\ n_l \leftarrow NP[l]$

For s=1 to M

If $A[s] = false$

$x'_1 \leftarrow P_1^s.x;\ x'_2 \leftarrow P_n^s.x;\ y'_1 \leftarrow P_1^s.y;\ y'_2 \leftarrow P_n^s.y;\ n_s \leftarrow NP[s]$

If dist(x_1, y_1, x_1', y_1')<EPS

$flag \leftarrow 1$

$x_1 \leftarrow x'_2;\ y_1 \leftarrow y'_2;$

Add_List_points($P_n^s ....P_1^s P_1^l ....P_n^l$ *);*

Else If dist(x_1, y_1, x_2', y_2')<EPS

$flag \leftarrow 1$

$x_1 \leftarrow x'_1;\ y_1 \leftarrow y'_1;$

Add_List_points($P_1^s ....P_n^s P_1^l ....P_n^l$ *);*

Else If dist(x_2, y_2, x_1', y_1')<EPS

$flag \leftarrow 1$

$x_2 \leftarrow x'_2;\ y_2 \leftarrow y'_2;$

Add_List_points($P_1^l ....P_n^l P_n^s ....P_1^s$ *);*

Else If dist(x_2, y_2, x_2', y_2')<EPS

$flag \leftarrow 1$

$x_2 \leftarrow x'_1;\ y_2 \leftarrow y'_1;$

Add_List_points($P_1^l ....P_n^l P_1^s ....P_n^s$ *);*

EndIf

If $flag = 1$

$NSEG \leftarrow NSEG - 1;\ n_l \leftarrow n_l + n_s;\ A[s] \leftarrow true$

EndIf

EndIf

EndFor

EndWhile

EndIf

EndWhile

4.3 CONTOUR EQUATION OVER TEN-NODE TRIANGLE

A cubic interpolation can be performed by using the triangular element and by taking two more discrete values of function on each of the three sides, as shown in Fig. 4.12(a). There is also one node at the centre of the element. This is called a ten-node triangular element.

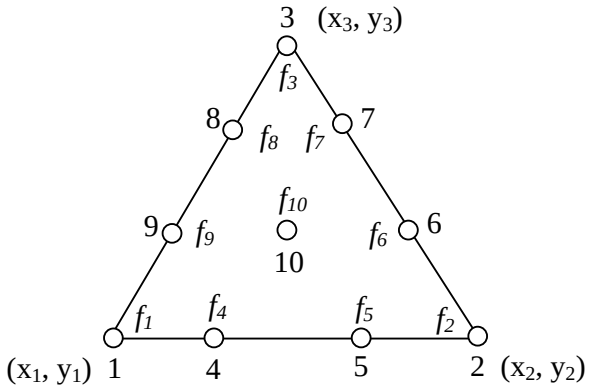


Figure 4.12(a) A ten-node triangular element

The interpolation is performed as shown in Eq. (4.19a) and (4.19b).

$$x = \sum_{i=1}^{3} N_i x_i \tag{4.19a}$$

$$y = \sum_{i=1}^{3} N_i y_i \tag{4.19b}$$

To increase the accuracy of the results further, we perform the exact tracing of contours over a ten-node element using cubic interpolation. The method is explained as follows.

The contour equation assumes the following form over a ten-node triangular element

$$\sum_{i=1}^{10} N_i f_i = c \tag{4.19c}$$

where f_i is the function value and N_i is the shape function for the i^{th} node. The shape function, N_i for ten-node triangular element (Zienkiewicz *et al.*, 2005) is defined by the following equations:

For corner nodes $N_i = \frac{1}{2}(3L_i - 1)(3L_i - 2)L_i \quad with \quad i = 1,2,3$

For mid-side nodes $N_4 = \frac{9}{2}L_1L_2(3L_1 - 1)$

$$N_5 = \frac{9}{2}L_1L_2(3L_2 - 1)$$

The shape functions for other nodes can be written in a similar manner.

For centre node $N_{10} = 27L_1L_2L_3$

where $L_i = \frac{1}{2\Delta}(a_i + b_i x + c_i y)$

$\Delta = \det(M)$ and $M = \begin{vmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{vmatrix}$

After substituting the values of shape function into Eq. (4.19c), the Eq. (4.19c) is simplified to

$$l_0 + l_1x + l_2y + l_3xy + l_4x^2 + l_5y^2 + l_6x^2y + l_7xy^2 + l_8x^3 + l_9y^3 = c \qquad (4.19d)$$

where,

$$l_0 = \frac{f_1}{4\Delta}\left(\frac{9a_1^3}{4\Delta^2} - \frac{9a_1^2}{2\Delta} + 2a_1\right) + \frac{f_2}{4\Delta}\left(\frac{9a_2^3}{4\Delta^2} - \frac{9a_2^2}{2\Delta} + 2a_2\right) + \frac{f_3}{4\Delta}\left(\frac{9a_3^3}{4\Delta^2} - \frac{9a_3^2}{2\Delta} + 2a_3\right) +$$
$$\frac{9f_4}{2}\left(\frac{3a_1^2a_2}{8\Delta^3} - a_1a_2\right) + \frac{9f_6}{2}\left(\frac{3a_2^2a_3}{8\Delta^3} - a_2a_3\right) + \frac{9f_8}{2}\left(\frac{3a_3^2a_1}{8\Delta^3} - a_3a_1\right) +$$
$$\frac{9f_5}{2}\left(\frac{3a_2^2a_1}{8\Delta^3} - a_1a_2\right) + \frac{9f_7}{2}\left(\frac{3a_3^2a_2}{8\Delta^3} - a_2a_3\right) + \frac{9f_9}{2}\left(\frac{3a_1^2a_3}{8\Delta^3} - a_1a_3\right) + \frac{27f_{10}}{8\Delta^3}(a_1a_2a_3)$$

$$l_1 = \frac{f_1}{4\Delta}\left(\frac{27a_1^2b_1}{4\Delta^2} - \frac{18a_1b_1}{2\Delta} + 2b_1\right) + \frac{f_2}{4\Delta}\left(\frac{27a_2^2b_2}{4\Delta^2} - \frac{18a_2b_2}{2\Delta} + 2b_2\right) + \frac{f_3}{4\Delta}\left(\frac{27a_3^2b_3}{4\Delta^2} - \frac{18a_3b_3}{2\Delta} + 2b_3\right) +$$

$$\frac{9f_4}{2}\left(\frac{6a_1a_2b_1}{8\Delta^3}+\frac{3a_1^2b_2}{8\Delta^3}-a_1b_2-b_1a_2\right)+\frac{9f_6}{2}\left(\frac{6a_2a_3b_2}{8\Delta^3}+\frac{3a_2^2b_3}{8\Delta^3}-a_2b_3-b_2a_3\right)+$$

$$\frac{9f_8}{2}\left(\frac{6a_3a_1b_3}{8\Delta^3}+\frac{3a_3^2b_1}{8\Delta^3}-a_3b_1-b_3a_1\right)+\frac{9f_5}{2}\left(\frac{6a_1a_2b_2}{8\Delta^3}+\frac{3a_2^2b_1}{8\Delta^3}-a_1b_2-b_1a_2\right)+$$

$$\frac{9f_7}{2}\left(\frac{6a_2a_3b_3}{8\Delta^3}+\frac{3a_3^2b_2}{8\Delta^3}-a_2b_3-b_2a_3\right)+\frac{9f_9}{2}\left(\frac{6a_1a_3b_1}{8\Delta^3}+\frac{3a_1^2b_3}{8\Delta^3}-a_3b_1-b_3a_1\right)+$$

$$\frac{27f_{10}}{8\Delta^3}\left(a_1a_2b_3+a_1a_3b_2+a_2a_3b_1\right)$$

$$l_2=\frac{f_1}{4\Delta}\left(\frac{27a_1^2c_1}{4\Delta^2}-\frac{18a_1c_1}{2\Delta}+2c_1\right)+\frac{f_2}{4\Delta}\left(\frac{27a_2^2c_2}{4\Delta^2}-\frac{18a_2c_2}{2\Delta}+2c_2\right)+\frac{f_3}{4\Delta}\left(\frac{27a_3^2c_3}{4\Delta^2}-\frac{18a_3c_3}{2\Delta}+2c_3\right)+$$

$$\frac{9f_4}{2}\left(\frac{6a_1a_2c_1}{8\Delta^3}+\frac{3a_1^2c_2}{8\Delta^3}-a_1c_2-c_1a_2\right)+\frac{9f_6}{2}\left(\frac{6a_2a_3c_2}{8\Delta^3}+\frac{3a_2^2c_3}{8\Delta^3}-a_2c_3-c_2a_3\right)+$$

$$\frac{9f_8}{2}\left(\frac{6a_3a_1c_3}{8\Delta^3}+\frac{3a_3^2c_1}{8\Delta^3}-a_3c_1-c_3a_1\right)+\frac{9f_5}{2}\left(\frac{6a_1a_2c_2}{8\Delta^3}+\frac{3a_2^2c_1}{8\Delta^3}-a_1c_2-c_1a_2\right)+$$

$$\frac{9f_7}{2}\left(\frac{6a_2a_3c_3}{8\Delta^3}+\frac{3a_3^2c_2}{8\Delta^3}-a_2c_3-c_2a_3\right)+\frac{9f_9}{2}\left(\frac{6a_1a_3c_1}{8\Delta^3}+\frac{3a_1^2c_3}{8\Delta^3}-a_3c_1-c_3a_1\right)+$$

$$\frac{27f_{10}}{8\Delta^3}\left(a_1a_2c_3+a_1a_3c_2+a_2a_3c_1\right)$$

$$l_3=\frac{f_1}{4\Delta}\left(\frac{54a_1b_1c_1}{4\Delta^2}-\frac{18b_1c_1}{2\Delta}\right)+\frac{f_2}{4\Delta}\left(\frac{54a_2b_2c_2}{4\Delta^2}-\frac{18b_2c_2}{2\Delta}\right)+\frac{f_3}{4\Delta}\left(\frac{54a_3b_3c_3}{4\Delta^2}-\frac{18b_3c_3}{2\Delta}\right)+$$

$$\frac{9f_4}{2}\left(\frac{6a_2b_1c_1}{8\Delta^3}+\frac{6a_1b_2c_1}{8\Delta^3}+\frac{6a_1b_1c_2}{8\Delta^3}-b_1c_2-c_1b_2\right)+$$

$$\frac{9f_6}{2}\left(\frac{6a_3b_2c_2}{8\Delta^3}+\frac{6a_2b_3c_3}{8\Delta^3}+\frac{6a_2b_2c_3}{8\Delta^3}-b_2c_3-c_2b_3\right)+$$

$$\frac{9f_8}{2}\left(\frac{6a_1b_3c_3}{8\Delta^3}+\frac{6a_3b_1c_3}{8\Delta^3}+\frac{6a_3b_3c_1}{8\Delta^3}-b_1c_3-c_1b_3\right)+$$

$$\frac{9f_5}{2}\left(\frac{6a_1b_2c_2}{8\Delta^3}+\frac{6a_2b_1c_2}{8\Delta^3}+\frac{6a_2b_2c_1}{8\Delta^3}-b_1c_2-c_1b_2\right)+$$

$$\frac{9f_7}{2}\left(\frac{6a_2b_3c_3}{8\Delta^3}+\frac{6a_3b_2c_3}{8\Delta^3}+\frac{6a_3b_3c_2}{8\Delta^3}-b_2c_3-c_3b_2\right)+$$

$$\frac{9f_9}{2}\left(\frac{6a_3b_1c_1}{8\Delta^3}+\frac{6a_1b_3c_1}{8\Delta^3}+\frac{6a_1b_1c_3}{8\Delta^3}-b_1c_3-c_1b_3\right)+$$

$$\frac{27 f_{10}}{8\Delta^3}\left(a_1 b_2 c_3 + c_1 a_2 b_3 + b_1 a_2 c_3 + b_1 c_2 a_3 + c_1 a_2 b_3 + c_1 b_2 a_3\right)$$

$$l_4 = \frac{f_1}{4\Delta}\left(\frac{27 b_1^2 a_1}{4\Delta^2} - \frac{9 b_1^2}{2\Delta}\right) + \frac{f_2}{4\Delta}\left(\frac{27 b_2^2 a_2}{4\Delta^2} - \frac{9 b_2^2}{2\Delta}\right) + \frac{f_3}{4\Delta}\left(\frac{27 b_3^2 a_3}{4\Delta^2} - \frac{9 b_3^2}{2\Delta}\right) +$$

$$\frac{9 f_4}{2}\left(\frac{3 a_2 b_1^2}{8\Delta^3} + \frac{6 b_1 b_2 a_1}{8\Delta^3} - b_1 b_2\right) + \frac{9 f_6}{2}\left(\frac{3 a_3 b_2^2}{8\Delta^3} + \frac{6 b_2 b_3 a_2}{8\Delta^3} - b_2 b_3\right) +$$

$$\frac{9 f_8}{2}\left(\frac{3 a_1 b_3^2}{8\Delta^3} + \frac{6 b_3 b_1 a_3}{8\Delta^3} - b_1 b_3\right) + \frac{9 f_5}{2}\left(\frac{3 a_1 b_2^2}{8\Delta^3} + \frac{6 b_1 b_2 a_2}{8\Delta^3} - b_1 b_2\right) +$$

$$\frac{9 f_7}{2}\left(\frac{3 a_2 b_3^2}{8\Delta^3} + \frac{6 b_2 b_3 a_3}{8\Delta^3} - b_2 b_3\right) + \frac{9 f_9}{2}\left(\frac{3 a_3 b_1^2}{8\Delta^3} + \frac{6 b_1 b_3 a_1}{8\Delta^3} - b_1 b_3\right) +$$

$$\frac{27 f_{10}}{8\Delta^3}\left(a_1 b_2 b_3 + b_1 a_2 b_3 + b_1 b_2 a_3\right)$$

$$l_5 = \frac{f_1}{4\Delta}\left(\frac{27 c_1^2 a_1}{4\Delta^2} - \frac{9 c_1^2}{2\Delta}\right) + \frac{f_2}{4\Delta}\left(\frac{27 c_2^2 a_2}{4\Delta^2} - \frac{9 c_2^2}{2\Delta}\right) + \frac{f_3}{4\Delta}\left(\frac{27 c_3^2 a_3}{4\Delta^2} - \frac{9 c_3^2}{2\Delta}\right) +$$

$$\frac{9 f_4}{2}\left(\frac{3 a_2 c_1^2}{8\Delta^3} + \frac{6 c_1 c_2 a_1}{8\Delta^3} - c_1 c_2\right) + \frac{9 f_6}{2}\left(\frac{3 a_3 c_2^2}{8\Delta^3} + \frac{6 c_2 c_3 a_2}{8\Delta^3} - c_2 c_3\right) +$$

$$\frac{9 f_8}{2}\left(\frac{3 a_1 c_3^2}{8\Delta^3} + \frac{6 c_3 c_1 a_3}{8\Delta^3} - c_1 c_3\right) + \frac{9 f_5}{2}\left(\frac{3 a_1 c_2^2}{8\Delta^3} + \frac{6 c_1 c_2 a_2}{8\Delta^3} - c_1 c_2\right) +$$

$$\frac{9 f_7}{2}\left(\frac{3 a_2 c_3^2}{8\Delta^3} + \frac{6 c_2 c_3 a_3}{8\Delta^3} - c_2 c_3\right) + \frac{9 f_9}{2}\left(\frac{3 a_3 c_1^2}{8\Delta^3} + \frac{6 c_1 c_3 a_1}{8\Delta^3} - c_1 c_3\right) +$$

$$\frac{27 f_{10}}{8\Delta^3}\left(a_1 c_2 c_3 + c_1 a_2 c_3 + c_1 c_2 a_3\right)$$

$$l_6 = \frac{f_1}{4\Delta}\left(\frac{27 b_1^2 c_1}{4\Delta^2}\right) + \frac{f_2}{4\Delta}\left(\frac{27 b_2^2 c_2}{4\Delta^2}\right) + \frac{f_3}{4\Delta}\left(\frac{27 b_3^2 c_3}{4\Delta^2}\right) +$$

$$\frac{9 f_4}{2}\left(\frac{3 c_2 b_1^2}{8\Delta^3} + \frac{6 b_1 b_2 c_1}{8\Delta^3}\right) + \frac{9 f_6}{2}\left(\frac{3 c_3 b_2^2}{8\Delta^3} + \frac{6 b_2 b_3 c_2}{8\Delta^3}\right) + \frac{9 f_8}{2}\left(\frac{3 c_1 b_3^2}{8\Delta^3} + \frac{6 b_3 b_1 c_3}{8\Delta^3}\right) +$$

$$\frac{9 f_5}{2}\left(\frac{3 c_1 b_2^2}{8\Delta^3} + \frac{6 b_1 b_2 c_2}{8\Delta^3}\right) + \frac{9 f_7}{2}\left(\frac{3 c_2 b_3^2}{8\Delta^3} + \frac{6 b_2 b_3 c_3}{8\Delta^3}\right) + \frac{9 f_9}{2}\left(\frac{3 c_3 b_1^2}{8\Delta^3} + \frac{6 b_1 b_3 c_1}{8\Delta^3}\right) +$$

$$\frac{27 f_{10}}{8\Delta^3}\left(b_1 b_2 c_3 + b_1 c_2 b_3 + c_1 b_2 b_3\right)$$

$$l_7 = \frac{f_1}{4\Delta}\left(\frac{27 c_1^2 b_1}{4\Delta^2}\right) + \frac{f_2}{4\Delta}\left(\frac{27 c_2^2 b_2}{4\Delta^2}\right) + \frac{f_3}{4\Delta}\left(\frac{27 c_3^2 b_3}{4\Delta^2}\right) + \frac{9 f_4}{2}\left(\frac{3 b_2 c_1^2}{8\Delta^3} + \frac{6 c_1 c_2 b_1}{8\Delta^3}\right) +$$

$$\frac{9f_6}{2}\left(\frac{3b_3c_2^2}{8\Delta^3}+\frac{6c_2c_3b_2}{8\Delta^3}\right)+\frac{9f_8}{2}\left(\frac{3b_1c_3^2}{8\Delta^3}+\frac{6c_3c_1b_3}{8\Delta^3}\right)+\frac{9f_5}{2}\left(\frac{3b_1c_2^2}{8\Delta^3}+\frac{6c_1c_2b_2}{8\Delta^3}\right)+$$

$$\frac{9f_7}{2}\left(\frac{3b_2c_3^2}{8\Delta^3}+\frac{6c_2c_3b_3}{8\Delta^3}\right)+\frac{9f_9}{2}\left(\frac{3b_3c_1^2}{8\Delta^3}+\frac{6c_1c_3b_1}{8\Delta^3}\right)+\frac{27f_{10}}{8\Delta^3}\left(b_1c_2c_3+c_1b_2c_3+c_1c_2b_3\right)$$

$$l_8=\frac{f_1}{4\Delta}\left(\frac{9b_1^3}{4\Delta^2}\right)+\frac{f_2}{4\Delta}\left(\frac{9b_2^3}{4\Delta^2}\right)+\frac{f_3}{4\Delta}\left(\frac{9b_3^3}{4\Delta^2}\right)+\frac{9f_4}{2}\left(\frac{3b_2b_1^2}{8\Delta^3}\right)+\frac{9f_6}{2}\left(\frac{3b_3b_2^2}{8\Delta^3}\right)+$$

$$\frac{9f_8}{2}\left(\frac{3b_1b_3^2}{8\Delta^3}\right)+\frac{9f_5}{2}\left(\frac{3b_1b_2^2}{8\Delta^3}\right)+\frac{9f_7}{2}\left(\frac{3b_2b_3^2}{8\Delta^3}\right)+\frac{9f_9}{2}\left(\frac{3b_3b_1^2}{8\Delta^3}\right)+\frac{27f_{10}}{8\Delta^3}\left(b_1b_2b_3\right)$$

$$l_9=\frac{f_1}{4\Delta}\left(\frac{9c_1^3}{4\Delta^2}\right)+\frac{f_2}{4\Delta}\left(\frac{9c_2^3}{4\Delta^2}\right)+\frac{f_3}{4\Delta}\left(\frac{9c_3^3}{4\Delta^2}\right)+\frac{9f_4}{2}\left(\frac{3c_2c_1^2}{8\Delta^3}\right)+\frac{9f_6}{2}\left(\frac{3c_3c_2^2}{8\Delta^3}\right)+$$

$$\frac{9f_8}{2}\left(\frac{3c_1c_3^2}{8\Delta^3}\right)+\frac{9f_5}{2}\left(\frac{3c_1c_2^2}{8\Delta^3}\right)+\frac{9f_7}{2}\left(\frac{3c_2c_3^2}{8\Delta^3}\right)+\frac{9f_9}{2}\left(\frac{3c_3c_1^2}{8\Delta^3}\right)+\frac{27f_{10}}{8\Delta^3}\left(c_1c_2c_3\right)$$

To trace the contours, we find intersection of the curve defined by Eq. (4.19d) with a side of a triangle and record the point of intersection, if any. Let the equation of a side be as given in Eq. (4.19e).

$$y=mx+n \tag{4.19e}$$

On substituting y from Eq. (4.19e) into Eq. (4.19d), we get Eq. (4.19f).

$$Ax^3+Bx^2+Cx+D=0 \tag{4.19f}$$

where $A=l_6m+l_7m^2+l_8+l_9m^3$

$$B=l_3m+l_4+l_6y_1-l_6mx_1-2x_1m^2l_7+2y_1ml_7-3l_9m^3x_1+3y_1m^2l_9+l_5m^2$$

$$C=l_1+l_2m+l_3y_1-l_3mx_1+l_7y_1^2+l_7m^2x_1^2-2l_7y_1mx_1+3l_9m^3x_1^2-3y_1m^2x_1l_9+$$

$$3my_1^2l_9-3y_1m^2x_1l_9-2x_1m^2l_5+2y_1ml_5$$

$$D=l_0+l_2y_1-l_2mx_1+l_9y_1^3-l_9m^3x_1^3-3y_1^2mx_1l_9+3y_1m^2x_1^2l_9+$$

$$l_5y_1^2+l_5m^2x_1^2-2y_1mx_1l_5-c$$

The Eq. (4.19f) can have maximum of three roots which signifies the intersecting points. Now the contour is traced by using Eq. (4.19d).

Although the direct method used here is straightforward in its implementation, its order of time is very high.

4.4 CONTOUR EQUATION OVER EIGHT-NODE QUADRILATERAL ELEMENT

It is assumed that the domain under consideration is discretized into eight-node quadrilateral elements with the known function values at all the nodes. The elements are mapped onto a square $[-1,+1][-1,+1]$ in the '$r-s$' co-ordinates system, shown in Fig. 4.12(b). The function $f(x,y)$ is interpolated using the transformation

$$f(x,y)=\sum_{i=1}^{8} N_i f_i \tag{4.20a}$$

where f_i is the function value and N_i is the shape function for the i^{th} node. The shape function, N_i is defined by the following equations.

For corner nodes $\quad N_i=\frac{1}{4}(1+rr_i)(1+ss_i)(rr_i+ss_i-1)$

For mid-side nodes $\quad if \quad r_i=0 \qquad N_i=\frac{1}{2}(1-r^2)(1+ss_i)$

$\quad if \quad s_i=0 \qquad N_i=\frac{1}{2}(1+rr_i)(1-s^2)$

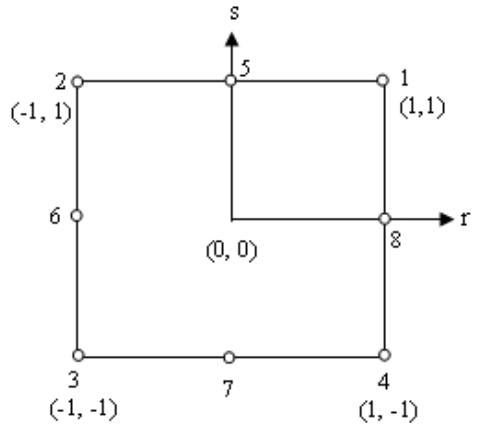


Figure 4.12(b) Eight-node element in local co-ordinate systems

The contour Eq. (4.20a) is simplified to

$$f(r,s)=C=C_0+C_1r+C_2s+C_3rs+C_4r^2s+C_5rs^2+C_6r^2+C_7s^2 \tag{4.20b}$$

where,

$$C_0=0.5(f_5+f_6+f_7+f_8)-0.25(f_1+f_2+f_3+f_4)$$

$$C_1 = 0.5(f_8 - f_6)$$

$$C_2 = 0.5(f_5 - f_7)$$

$$C_3 = 0.25[f_1 + f_3 - (f_2 + f_4)]$$

$$C_4 = 0.25[f_1 + f_2 - (f_3 + f_4) + 2(f_7 - f_5)]$$

$$C_5 = 0.25[f_4 + f_1 - (f_3 + f_2) + 2(f_6 - f_8)]$$

$$C_6 = 0.25[f_3 + f_4 + f_1 + f_2 - 2(f_7 + f_5)]$$

$$C_7 = 0.25[f_3 + f_4 + f_1 + f_2 - 2(f_8 + f_6)]$$

The contour traced in local co-ordinates using Eq. (4.20b) is transformed into global co-ordinates using Eq. (4.20c) and (4.20d).

$$x = \sum_{i=1}^{8} N_i x_i \tag{4.20c}$$

$$y = \sum_{i=1}^{8} N_i y_i \tag{4.20d}$$

The algorithm discussed in section 3.3 of chapter III is used for plotting of contours. Except for steps (ii), (iv) and (vii), all steps are easy to understand. Detailed explanations are given for these steps in the following sections.

4.4.1 Finding the Minimum and the Maximum of *f(x, y)* over the Element

In general, the minimum and the maximum for a quadratic equation can be found by finding the point of extremum by taking the partial derivatives of the function given by Eq. (4.20b) w.r.t. r and s and equating them to zero, *i.e.*,

$$\frac{\partial f(r,s)}{\partial r} = 0 = C_1 + C_3 s + 2C_4 rs + C_5 s^2 + 2C_6 r \tag{4.21}$$

$$\frac{\partial f(r,s)}{\partial s} = 0 = C_2 + C_3 r + C_4 r^2 + 2C_5 rs + 2C_7 s \tag{4.22}$$

From these expressions in general a polynomial of fourth order in s can be deduced. The following steps are helpful to obtain an advantageous formulation.

From Eq. (4.21), we can write

$$r = -\frac{C_1 + C_3 s + C_5 s^2}{2(C_6 + C_4 s)} \tag{4.23}$$

The value of r is substituted in Eq. (4.22) to obtain the expression given in Eq. (4.24).

$$a_4 s^4 + a_3 s^3 + a_2 s^2 + a_1 s + a_0 = 0 \tag{4.24}$$

where,

$$a_4 = -3C_4 C_5^2$$

$$a_3 = 8C_4^2 C_7 - 4C_5^2 C_6 - 4C_3 C_4 C_5$$

$$a_2 = 4C_2 C_4^2 + 16C_4 C_6 C_7 - 6C_3 C_5 C_6 - C_3^2 C_4 - 2C_1 C_4 C_5$$

$$a_1 = 8C_2 C_4 C_6 + 8C_6^2 C_7 - 2C_3^2 C_6 - 4C_1 C_5 C_6$$

$$a_0 = 4C_2 C_6^2 - 2C_1 C_3 C_6 + C_1^2 C_4$$

Eq. (4.24) has four, two or no real roots, s_i which are obtained by standard routines. The appropriate values of r_i are obtained by inserting s_i into Eq. (4.23). Only the values of r_i, s_i that lie in the square $-1 \le r, s \le 1$ are used for further calculations for value of function over an element. These calculated values along with the given values on the nodes of an element decide the minimum and the maximum, $f_{\min}$ and $f_{\max}$, of $f(x, y)$ over the element.

4.4.2 Tracing Contour Segment over an Element

A contour may appear in different forms over an element depending upon the number of intersections with its edges.

Intersection with the r-edges $(r = \pm 1)$ results in

$$s^2(\pm C_5 + C_7) + s(C_2 \pm C_3 + C_4) + (C_0 \pm C_1 + C_6 - C) = 0 \tag{4.25}$$

Intersection with the s-edges $(s = \pm 1)$ results in

$$r^2(\pm C_4 + C_6) + r(C_1 \pm C_3 + C_5) + (C_0 \pm C_2 + C_7 - C) = 0 \tag{4.26}$$

The Eqs. (4.25) and (4.26) can be solved to find the intersections with the edges. The roots that lie between $-1 \le r, s \le 1$ are used for further calculations.

Depending upon the number of intersection, nr the following different cases appear.

Case 1: No intersection with the edges

If there are no intersections, $nr = 0$ and $f_{\min} \le C \le f_{\max}$, then there is a contour loop, shown in Fig. 4.13, tracing of which is done by finding the intersections of required contour curve with $r = r_{\min}$ or $r = r_{\max}$ and $s = s_{\min}$ or $s = s_{\max}$. Substituting the values of $r_{\min}, r_{\max}$ and $s_{\min}, s_{\max}$ in the contour equation will yield its boundary intervals which can be joined. There can be only one contour loop, because the equation is cubic.

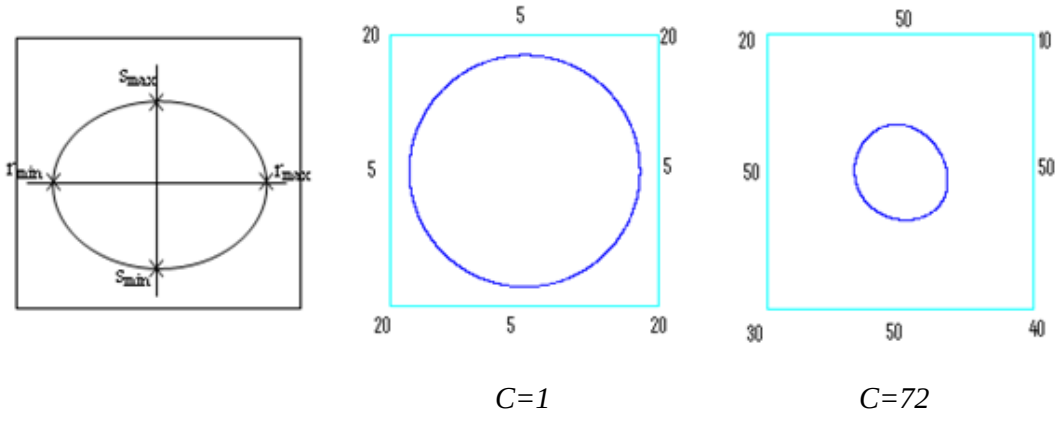
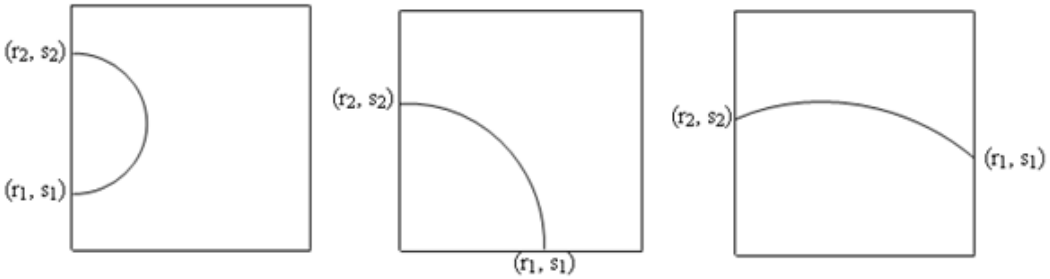


Figure 4.13 Possible case of a loop

Case 2: Two intersections with the edges

If there are two intersections, $nr = 2$, then these are two ends of the contour curve. The possible cases are shown in Fig. 4.14. In case the point of minimum/ maximum lies inside the element, the intersections with $r = r_{\min}$ or $r = r_{\max}$ and $s = s_{\min}$ or $s = s_{\max}$ are found as discussed in previous step, which will yield its boundary intervals. These boundary intervals can be joined to form a contour.



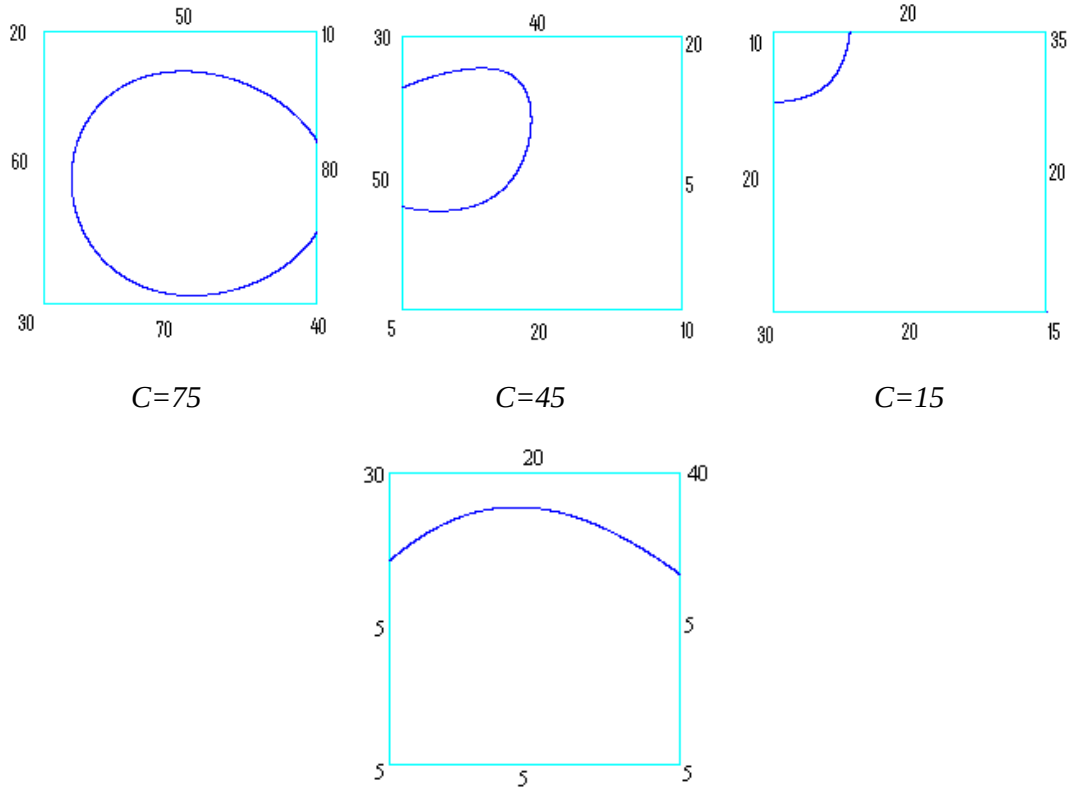
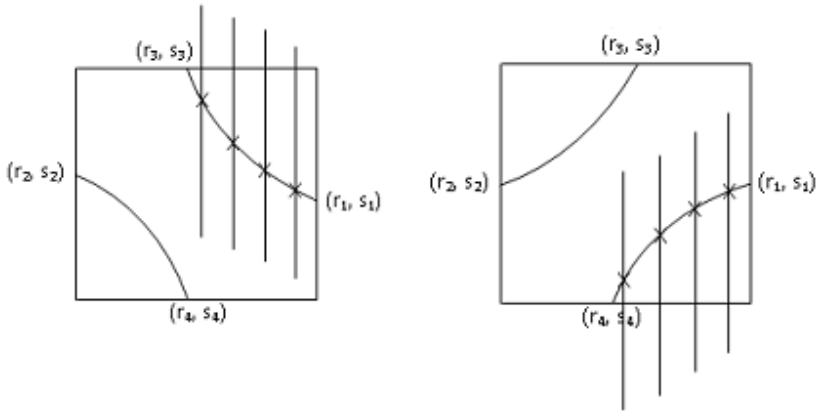


Figure 4.14 Different possible cases for two intersections

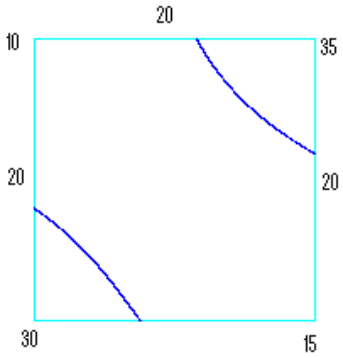
Case 3: Four intersections with the edges

If there are four intersections, $nr = 4$ then the number of possible intersections on the four edges are (1,1,1,1), (1,1,2,0) and (2,2,0,0). All these possible intersections are discussed below.

(a) For one intersection each on the edges, there can be two different cases, shown in Fig.4.15, for joining the intersection points. For example, (r_2, s_2) can be joined with either (r_4, s_4) or with (r_3, s_3). To join the proper intersecting points, the intermediate points between (r_2, s_2) and (r_1, s_1) are calculated. If these intermediate points approach (r_4, s_4), then (r_2, s_2) is joined with (r_4, s_4), else it is joined with (r_3, s_3).



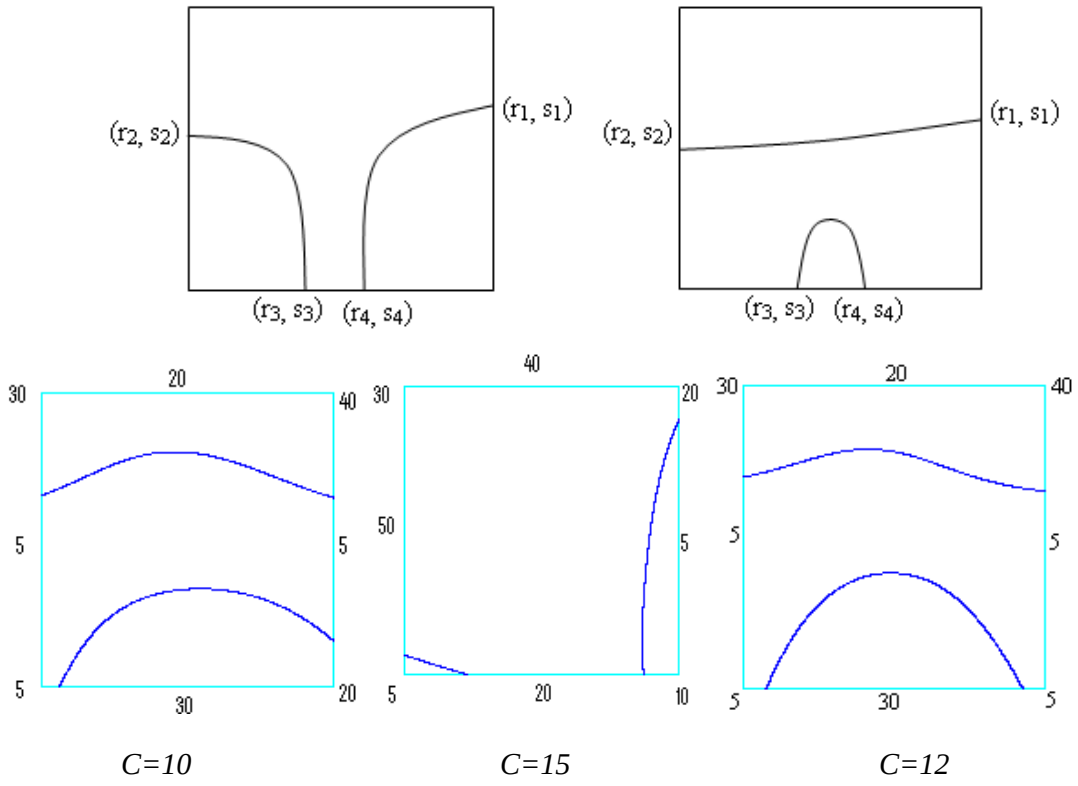
(a) Intermediate points approach (r_3, s_3) (b) Intermediate points approach (r_4, s_4)



C=22

Figure 4.15 Different cases of (1,1,1,1) possible intersections

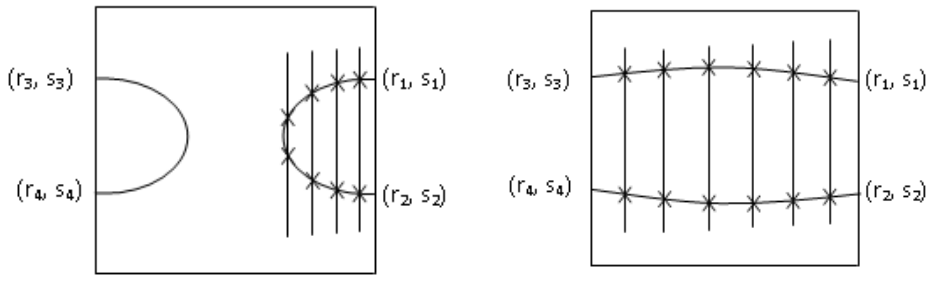
(b) For the possible intersections of (1,1,2,0), there can be two different cases, shown in Fig.4.16, for joining the intersection points. For example, (r_2, s_2) can be joined with either (r_3, s_3) or with (r_1, s_1). To join the proper intersecting points, the intermediate points between (r_2, s_2) and (r_1, s_1) are calculated. If these intermediate points approach (r_3, s_3), then (r_2, s_2) is joined with (r_3, s_3), else it is joined with (r_1, s_1). To generalize the concept, the calculation of intermediate points is started from the edge with only one intersection point.



C=10 *C=15* *C=12*

Figure 4.16 Different cases of (1,1,2,0) possible intersections

(c) For the possible intersections of (2,2,0,0), there can be different cases, shown in Fig. 4.17, for joining the intersection points. For example, (r_1, s_1) can be joined with either (r_3, s_3) or with (r_2, s_2). To join the proper intersecting points, the intermediate points between (r_1, s_1) and (r_3, s_3) are calculated. As we move from point (r_1, s_1) towards (r_3, s_3), there can be one or two roots *i.e.* one or two intersection points which form the curve. If there are two roots, then the distance between the two roots is calculated. If the distance approaches zero value, then (r_1, s_1) is joined with (r_2, s_2), else it is joined with (r_3, s_3).



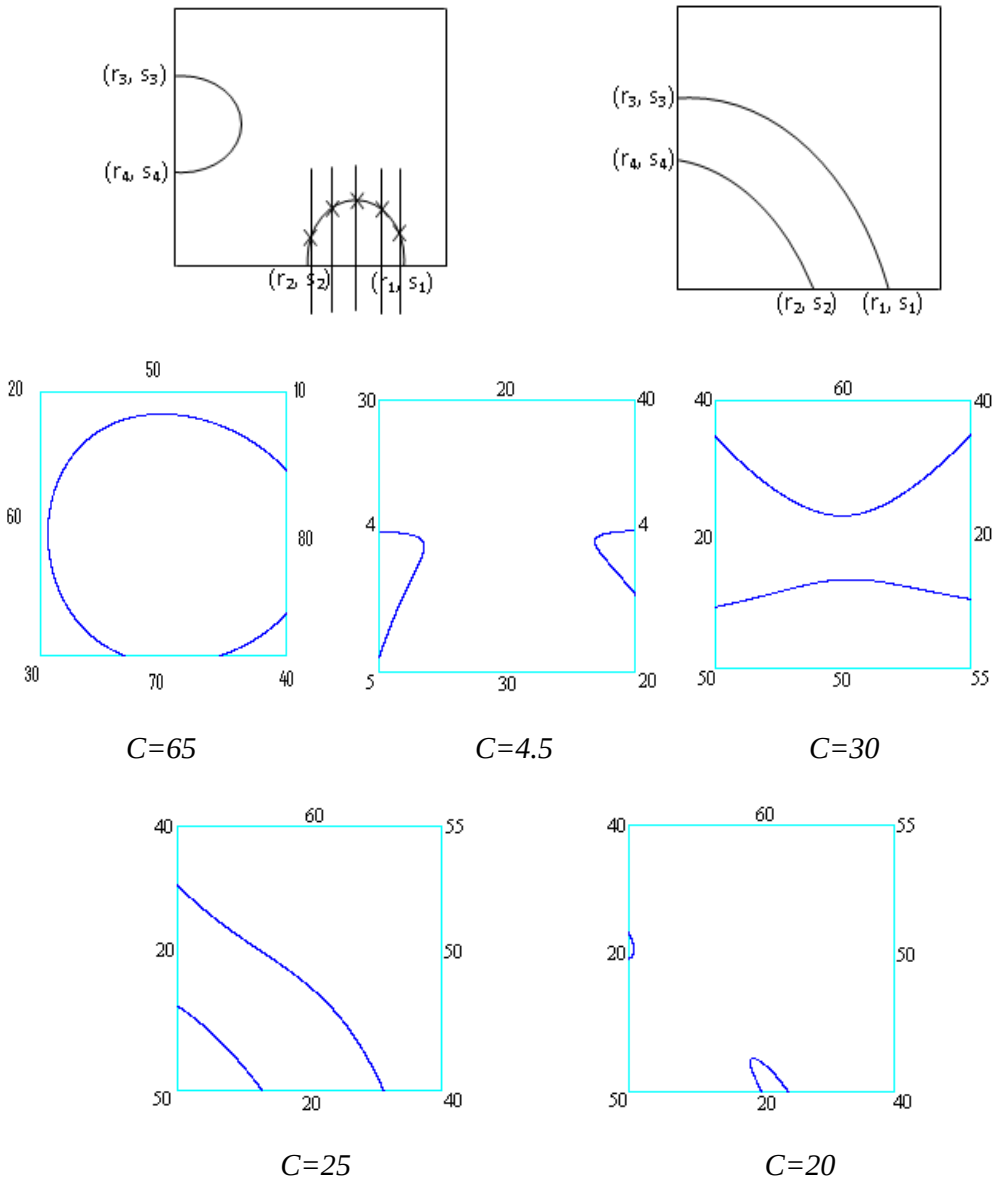


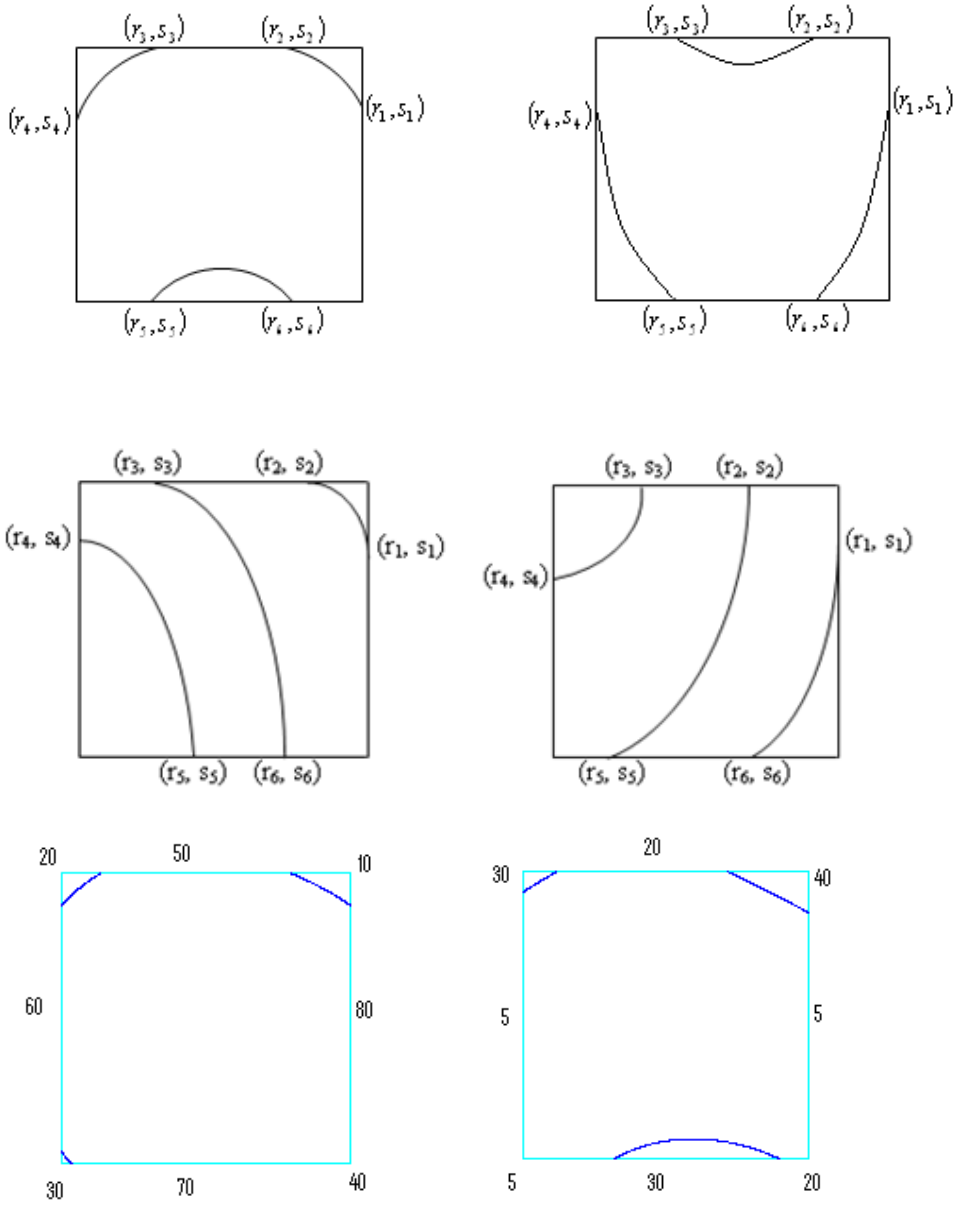
Figure 4.17 Different cases of (2,2,0,0) possible intersections

Case 4: Six intersection with the edges

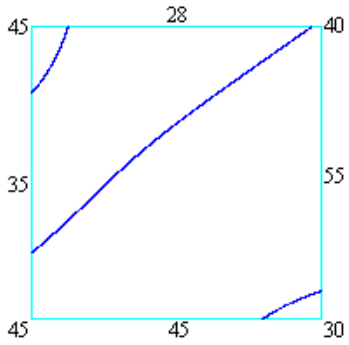
If there are six intersections, $nr = 6$ then the number of possible intersections on the four edges are (1,1,2,2) and (2,2,2,0). All these possible intersections are discussed below.

(a) For the possible intersections of (1,1,2,2), there can be four different cases, shown in Fig. 4.18, for joining the intersection points. Firstly the points are arranged in clockwise or anticlockwise direction. Point (r_1, s_1) can be joined with either (r_2, s_2) or with (r_6, s_6). To join the proper intersecting points, the intermediate points between (r_1, s_1) and (r_4, s_4) are calculated. If these intermediate points approach

(r_2, s_2), then (r_1, s_1) is joined with (r_2, s_2) and rest of the four points are joined as discussed in *Case 3*, else (r_1, s_1) is joined with (r_6, s_6) and rest of the four points are joined as discussed in *Case 3*. To generalize the concept, the calculation of intermediate points is started from the edge with only one intersection point.

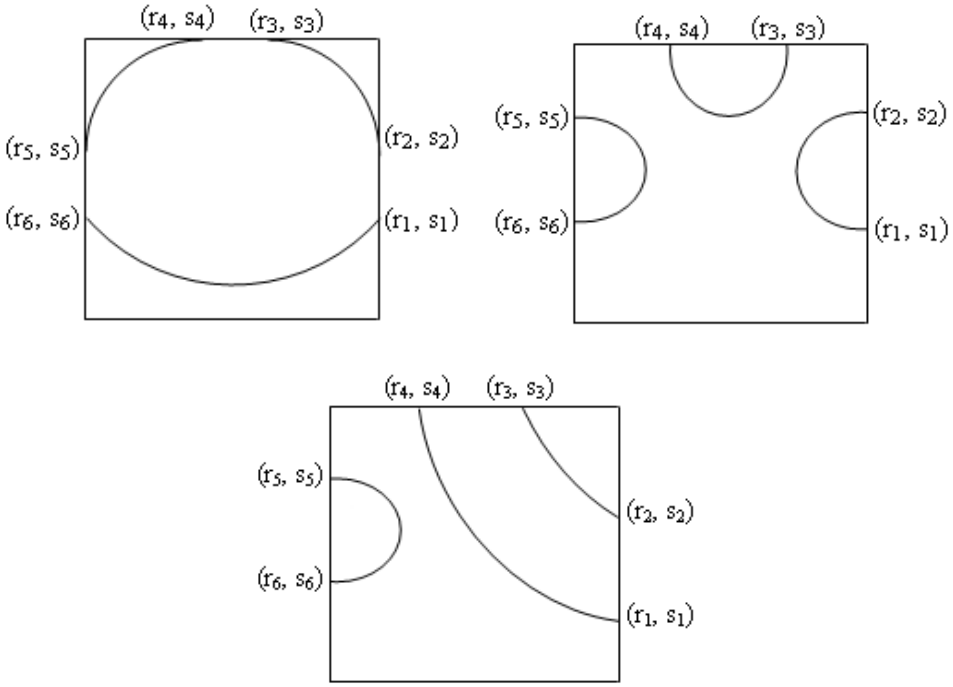


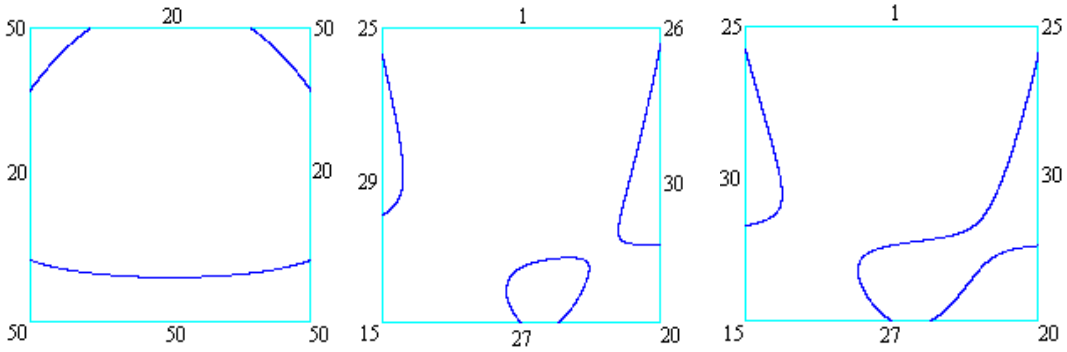
C=35 *C=25*



C=38

Figure 4.18 Different cases of (1,1,2,2) possible intersections





C=30 *C=27* *C=27*

Figure 4.19 Different cases of (2,2,0,0) possible intersections

(b) For the possible intersections of (2,2,2,0), there can be three different cases, shown in Fig.4.19, for joining the intersection points. Joining of these points are done as discussed in *Case 4* (a).

Case 5: Eight intersection with the edges

If $nr = 8$, then the number of possible intersections on the edges is only (2,2,2,2). The possible case for joining the intersecting points is shown in Fig.4.20. Firstly the points are arranged in clockwise or anticlockwise direction and then joined together in pairs i.e., $(r_2, s_2)(r_3, s_3)$, $(r_4, s_4)(r_5, s_5)$, $(r_6, s_6)(r_7, s_7)$ and $(r_8, s_8)(r_1, s_1)$

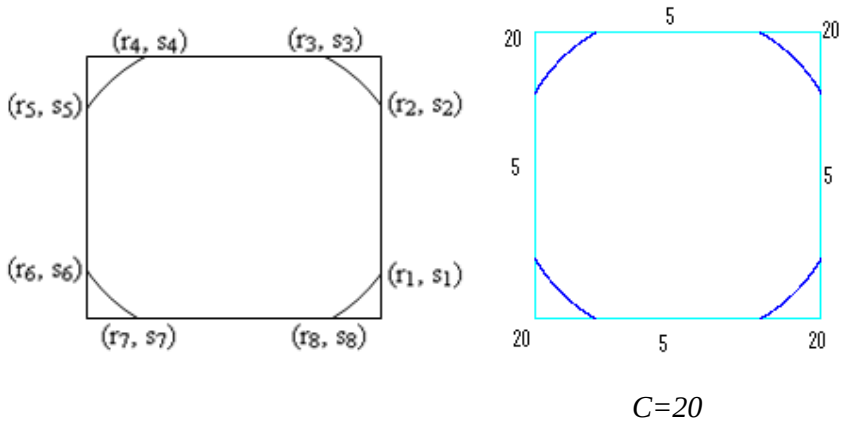


Figure 4.20 Case of (2,2,2,2) possible intersections

4.5 IMPLEMENTATION OF THE ALGORITHM FOR SIX-NODE TRIANGULAR ELEMENT

The contour plot algorithm, discussed in section 4.2, is implemented in Turbo C++ and runs on an IBM compatible PC with 3.0 GHz CPU and 256 RAM under Windows operating system. The OpenGL graphics libraries are used to display the geometry and contour lines for different components.

Firstly, a pure synthetic problem is analyzed which represents all conic sections for the cases represented by Eq. (4.5). We consider a square domain $[-2,2]\times[-2,2]$ in cartesian system and find contours for various values of the coefficients, *a* through *f*, as given in Table 4.2. The coefficients given are for the global function $f(x, y)$.

Conic	***a***	***h***	***b***	***g***	***f***	***c***
Circle	1	0	1	0	0	-3.8 < *c* < -1.16
Ellipse	0.82	0	1.414	0	0	-3.2 < *c* < -1
Parabola	1	0	0	2	-5<*f*<-2	1
Hyperbola	1	-1	-3	0	0	-2.5 <*c* < -0.5
Rectangular Hyperbola	1	-1	-1	0	0	-2.5 < *c* < -0.5
Pair of St. lines	-0.05	-0.19	1	0.74	-1.91	-0.5 < *c* <-0.2
Pair of parallel St. lines	2	0	0	-2.5<*g*<-0.5	0	-2
Pair of Perpendicular St. lines	0	1	0	-1.1<*g*<0.5	-1.1<*f*<0.5	0.25<*c*<1.21

Table 4.2 Values of coefficients for different contours shown in Fig. 4.22

For example, the circle generating function is

$$f(x,y) = x^2 + y^2 \tag{4.27a}$$

i.e., a=b=1, f=g=h=0 and $c \in [-3.8,-1.16]$ which is taken to be arbitrary. For a given point (x,y) in the square, it provides the value of the function $f(x,y)$. Fig. 4.21 depicts the griding of six-node triangles. Fig. 4.22 (a)-4.22(h) displays the contours using six-node triangular elements.

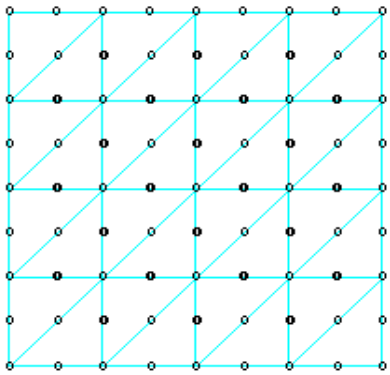


Figure 4.21 Griding of a square domain using six-node triangles

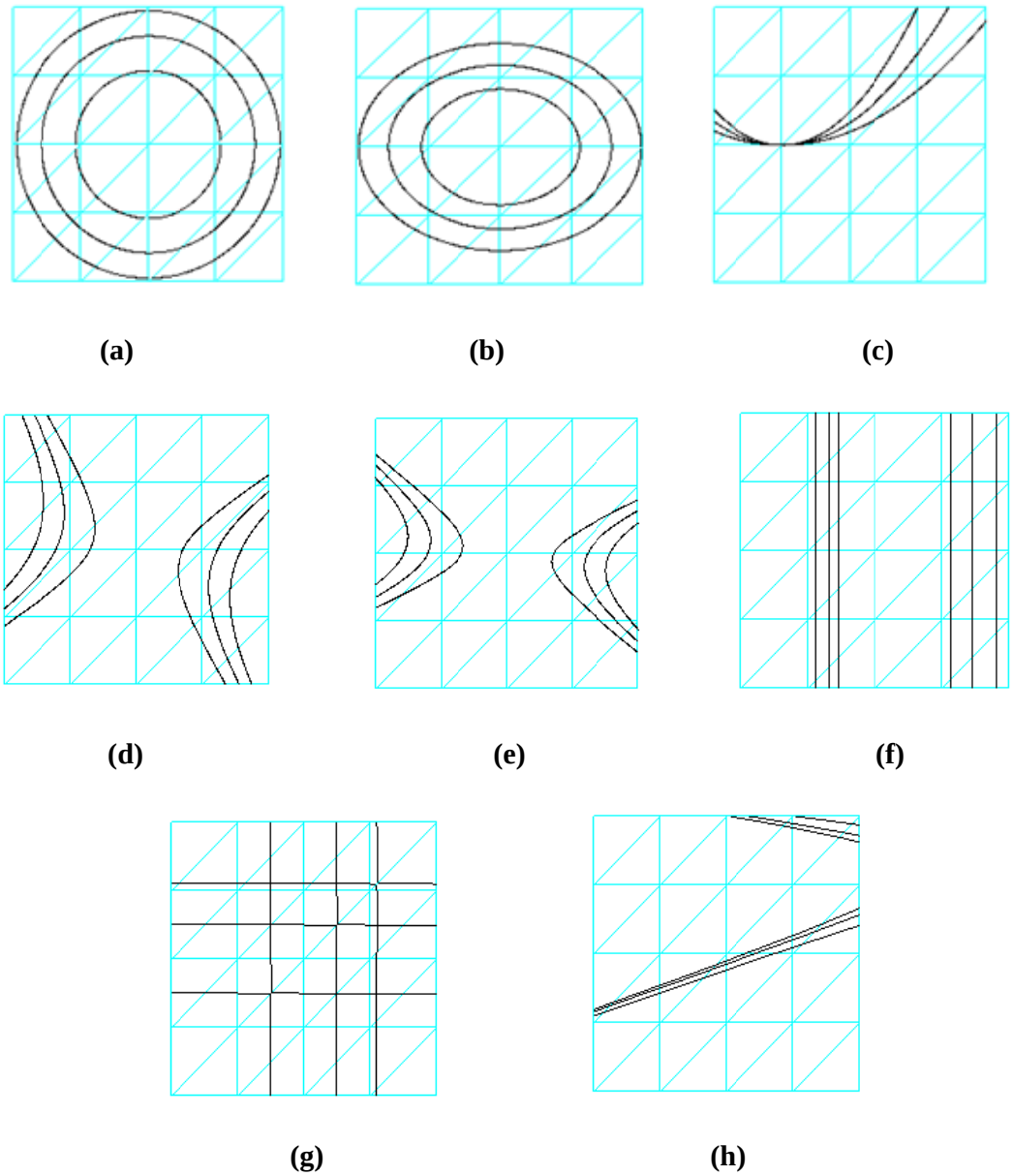
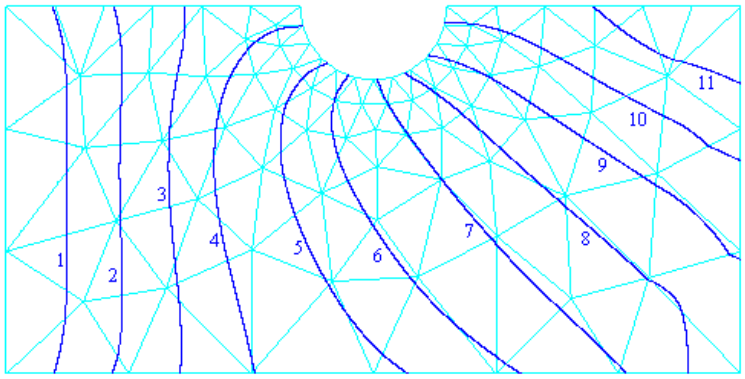


Figure 4.22 Contours over the square domains: (a) circle; (b) ellipse; (c) parabola; (d) rectangular hyperbola; (e) hyperbola; (f) pair of parallel straight lines; (g) pair of perpendicular straight lines; (h) pair of straight lines

Next, the algorithm is implemented using the practical problems for different components.

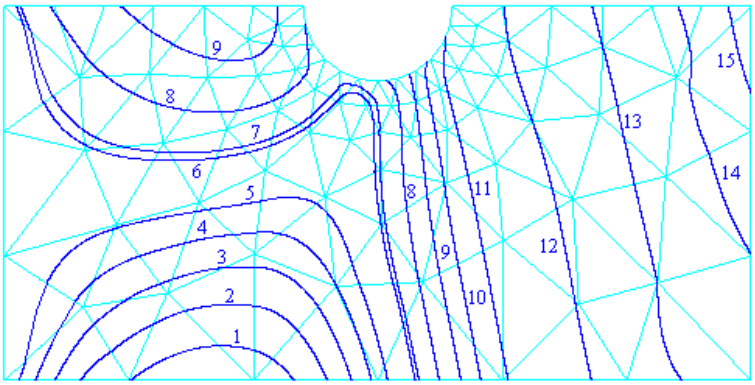
To demonstrate the accuracy of the algorithm, the components that were analyzed using three-node triangular element in chapter III, were again analyzed using six-node triangular elements taking the same number of nodes and elements. Thereafter, the contour lines generated for both the results were compared for accuracy.

The first problem compared is a deflection of a 2D beam with specifications shown in Fig. 3.8 and boundary conditions given in Fig. 3.9(a) and Fig. 3.9(b). After implementing the above discussed algorithm, the contours have been plotted.



Contour Levels	δ_x
1	0.5 x10^{-4}
2	1 x10^{-4}
3	1.5 x10^{-4}
4	2 x10^{-4}
5	2.8 x10^{-4}
6	3.5 x10^{-4}
7	4.5 x10^{-4}
8	5.5 x10^{-4}
9	6.5 x10^{-4}
10	7.5 x10^{-4}
11	8.5 x10^{-4}

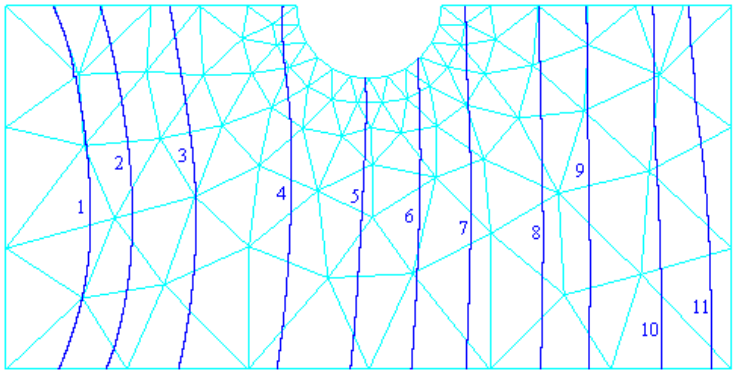
Figure 4.23(a) Contour plot of horizontal deflection using six-node triangles for axially loaded beam (units in mm)



Contour Levels	δ_y
1	0.5 x10^{-4}
2	0.4 x10^{-4}
3	0.3 x10^{-4}
4	0.2 x10^{-4}
5	0.1 x10^{-4}
6	-0.1 x10^{-4}
7	-0.13 x10^{-4}
8	-0.3 x10^{-4}
9	-0.5 x10^{-4}
10	-0.7 x10^{-4}
11	-1 x10^{-4}
12	-2 x10^{-4}
13	-3 x10^{-4}
14	-4 x10^{-4}
15	-5 x10^{-4}

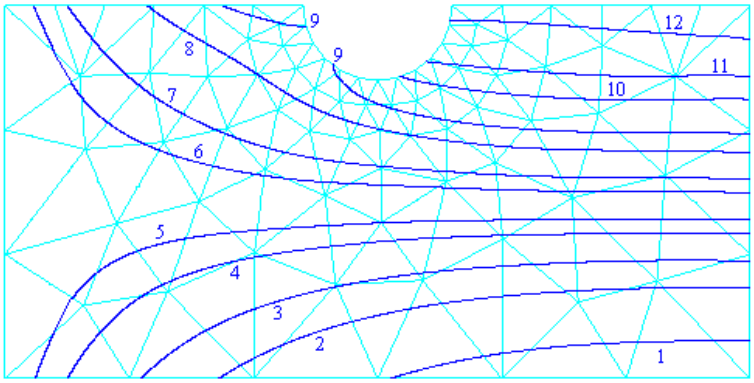
Figure 4.23(b) Contour plot of vertical deflection using six-node triangles for axially loaded beam (units in mm)

Fig. 4.23(a) and 4.23(b) depict the contour plots of horizontal and vertical deflections using six-node triangles for different levels of deflections for axially loaded beam.



Contour Levels	δ_y
1	-0.5 x10^{-4}
2	-1 x10^{-4}
3	-2 x10^{-4}
4	-4 x10^{-4}
5	-6 x10^{-4}
6	-8 x10^{-4}
7	-10 x10^{-4}
8	-13 x10^{-4}
9	-15 x10^{-4}
10	-18 x10^{-4}
11	-20 x10^{-4}

Figure 4.24(a) Contour plot of vertical deflection using six-node triangles for vertically loaded beam (units in mm)



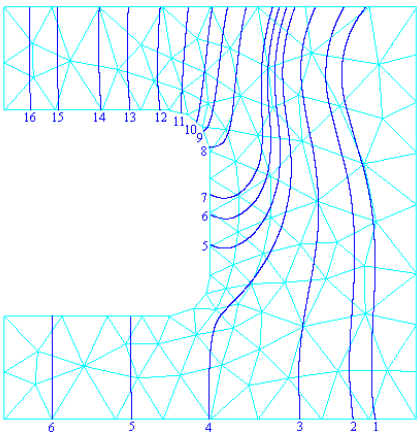
Contour Levels	δ_x
1	-5 x10^{-4}
2	-3 x10^{-4}
3	-2 x10^{-4}
4	-1 x10^{-4}
5	-0.5 x10^{-4}
6	0.5 x10^{-4}
7	1 x10^{-4}
8	2 x10^{-4}
9	2.7 x10^{-4}
10	4 x10^{-4}
11	5 x10^{-4}
12	7 x10^{-4}

Figure 4.24(b) Contour plot of horizontal deflection using six-node triangles for vertically loaded beam (units in mm)

Fig. 4.24(a) and 4.24(b) depict the contour plots of horizontal and vertical deflections using six-node triangles for different levels of deflections for vertically loaded beam.

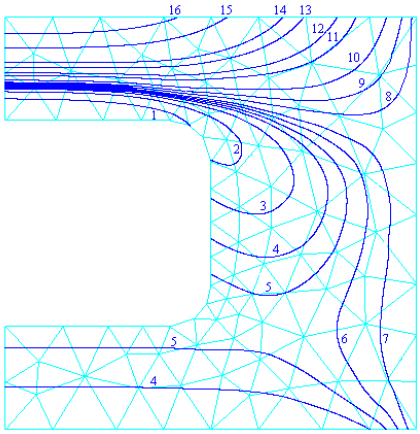
The second problem compared is a deflection of a 2D beam with specifications shown in Fig. 3.13(a) and boundary conditions given in Fig. 3.13(b). The beam is analyzed using six-node triangles with same number of elements, loading, material properties and boundary conditions.

Fig. 4.25(a) and 4.25(b) are the contour plots of vertical and horizontal deflections using six-node triangles for different levels of deflections.



Contour Levels	δ_y
1	2 x10^{-4}
2	3 x10^{-4}
3	5 x10^{-4}
4	7 x10^{-4}
5	8 x10^{-4}
6	9 x10^{-4}
7	10 x10^{-4}
8	15 x10^{-4}
9	20 x10^{-4}
10	25 x10^{-4}
11	35 x10^{-4}
12	50 x10^{-4}
13	70 x10^{-4}
14	90 x10^{-4}
15	120 x10^{-4}
16	140 x10^{-4}

Figure 4.25(a) Contour plot of vertical deflection using six-node triangles for the component



Contour Levels	δ_x
1	-10 x10^{-4}
2	-5 x10^{-4}
3	-3 x10^{-4}
4	-2 x10^{-4}
5	-1.5 x10^{-4}
6	-1 x10^{-4}
7	-0.5 x10^{-4}
8	0.5 x10^{-4}
9	1.5 x10^{-4}
10	3 x10^{-4}
11	6 x10^{-4}
12	8 x10^{-4}
13	12 x10^{-4}
14	15 x10^{-4}
15	25 x10^{-4}
16	35 x10^{-4}

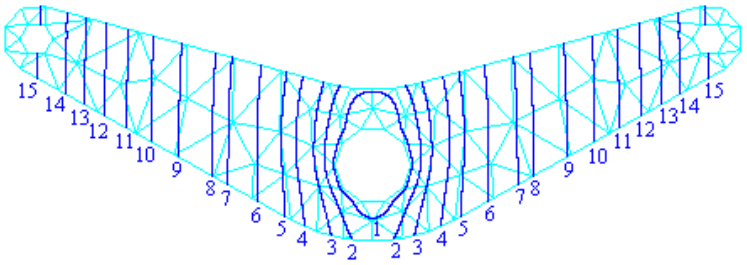
Figure 4.25(b) Contour plot of horizontal deflection using six-node triangles for the component

It is clear from Fig. 4.23(a), 4.23(b), 4.24(a), 4.24(b), 4.25(a) and 4.25(b) that the behavior of contours on six-node triangles is non-linear because they appear to be curves unlike the contours in Fig. 3.11(a), 3.11(b), 3.12(a), 3.12(b), 3.14(a) and 3.14(b) which are straight line segments. It can be seen that the behavior of contours on six-node triangles are more accurate than those on three-node triangular elements. Thus, the contouring using six-node elements is as accurate as the analysis results of FEA.

Further the different components are analyzed and contour plots are generated using six-node triangular element.

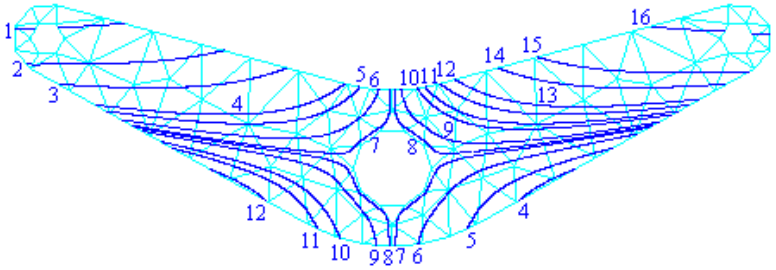
The contours are generated for the rocker arm with specifications, boundary and loading conditions shown in Fig. 3.15(a) and 3.15(b). For a load of 1000 N, Young's Modulus

$0.2\times10^{9}\,MPa$, Poisson's ratio 0.3, the values of deflections are derived using six-node triangular element, considering the problem as 2D plane problem with negligible thickness. As the arm is vertically loaded, the vertical and horizontal deflections are given by Eq. (3.26) and (3.28).



Contour Levels	δ_y
1	-1 x10^{-4}
2	-5 x10^{-4}
3	-10 x10^{-4}
4	-20 x10^{-4}
5	-30 x10^{-4}
6	-50 x10^{-4}
7	-80 x10^{-4}
8	-100 x10^{-4}
9	-150 x10^{-4}
10	-200 x10^{-4}
11	-250 x10^{-4}
12	-300 x10^{-4}
13	-350 x10^{-4}
14	-400 x10^{-4}
15	-480 x10^{-4}

Figure 4.26(a) Contour plot of vertical deflection using six-node triangles for the rocker arm



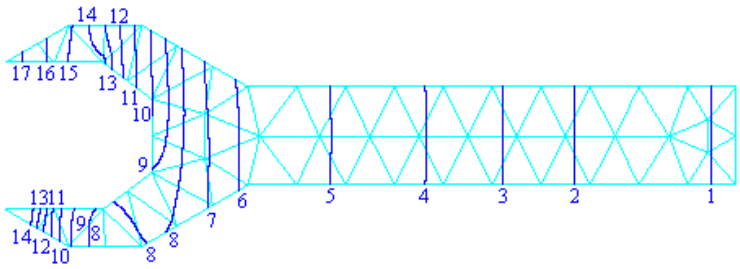
Contour Levels	δ_x
1	-210 x10^{-4}
2	-120 x10^{-4}
3	-70 x10^{-4}
4	-30 x10^{-4}
5	-20 x10^{-4}
6	-8 x10^{-4}
7	-1 x10^{-4}
8	1 x10^{-4}
9	5 x10^{-4}
10	15 x10^{-4}
11	20 x10^{-4}
12	30 x10^{-4}
13	40 x10^{-4}
14	70 x10^{-4}
15	100 x10^{-4}
16	200 x10^{-4}

Figure 4.26(b) Contour plot of horizontal deflection using six-node triangles for the rocker arm

Fig. 4.26(a) and 4.26(b) shows the contour plots of vertical and horizontal deflections using six-node triangles for different levels of deflections.

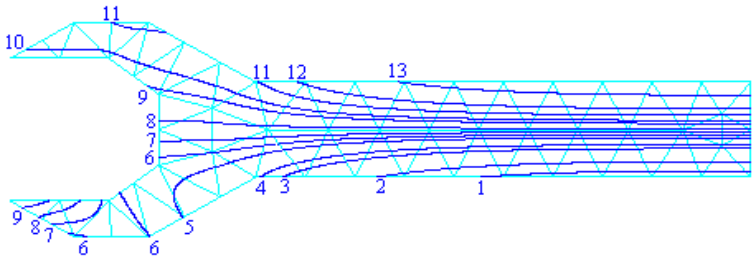
The contours are generated for the spanner with specifications, boundary and loading conditions shown in Fig. 3.17(a) and 3.17(b). For a load of 1000 N, Young's Modulus $0.2\times10^{9}\,MPa$, Poisson's ratio 0.3, the values of deflections are derived using six-node triangular element, considering the problem as 2D plane problem with negligible thickness. As the arm is vertically loaded, the vertical and horizontal deflections are given by Eqs. (3.26) to (3.28).

Figs. 4.27(a) and 4.27(b) shows the contour plots of vertical and horizontal deflections using six-node triangles for different levels of deflections.



Contour Levels	δ_y
1	-30 x10^{-2}
2	-20 x10^{-2}
3	-15 x10^{-2}
4	-10 x10^{-2}
5	-5 x10^{-2}
6	-2 x10^{-2}
7	-1.5 x10^{-2}
8	-1.2 x10^{-2}
9	-1 x10^{-2}
10	-0.8 x10^{-2}
11	-0.6 x10^{-2}
12	-0.4 x10^{-2}
13	-0.2 x10^{-2}
14	-0.05 x10^{-2}
15	0.1 x10^{-2}
16	0.4 x10^{-2}
17	0.7 x10^{-2}

Figure 4.27(a) Contour plot of vertical deflection using six-node triangles for the spanner



Contour Levels	δ_x
1	-3.7 x10^{-2}
2	-3.0 x10^{-2}
3	-2.0 x10^{-2}
4	-1.7 x10^{-2}
5	-1.2 x10^{-2}
6	-1.0 x10^{-2}
7	-0.8 x10^{-2}
8	-0.5 x10^{-2}
9	-0.1 x10^{-2}
10	0.1 x10^{-2}
11	0.5 x10^{-2}
12	1.0 x10^{-2}
13	1.2 x10^{-2}

Figure 4.27(b) Contour plot of horizontal deflection using six-node triangles for the spanner

Fig. 4.28(a) shows the specifications of a lever. The lever is fixed at the centre hole and loaded horizontally and vertically through the other holes, as shown in Fig. 4.28(b). For a load of 1000 N, Young's Modulus $0.2\times10^{9}\ MPa$, Poisson's ratio 0.3, the values of deflections are derived using FEA, considering the problem as 2D plane problem with negligible thickness. For the given loading conditions, the horizontal and vertical deflections are given by Eqs. (3.26) to (3.28).

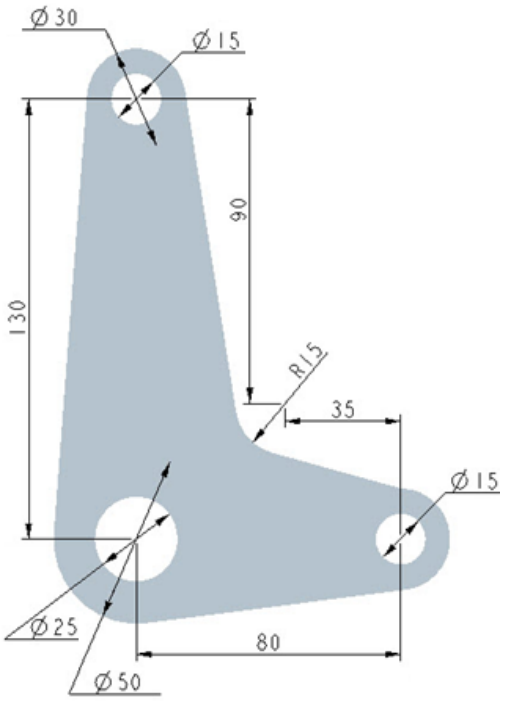


Figure 4.28(a) Specifications of the lever (all dimensions in mm)

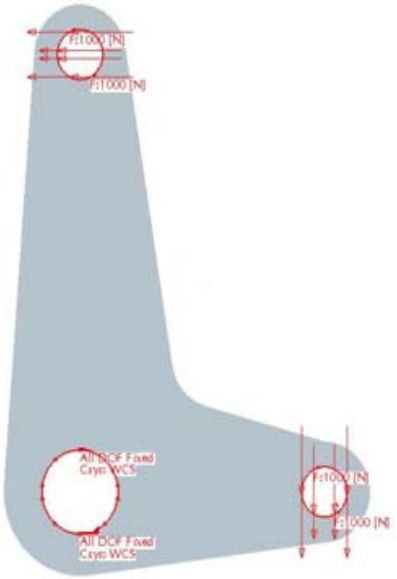
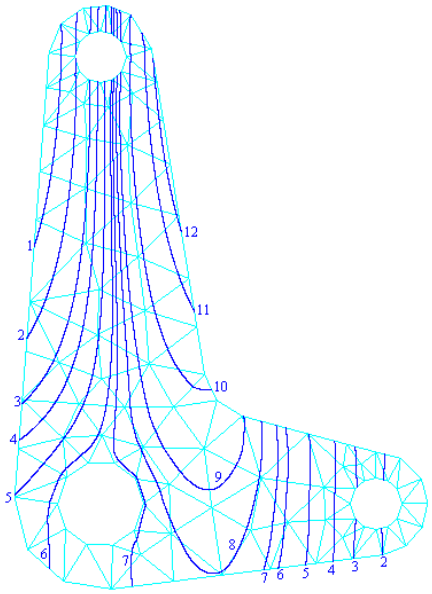


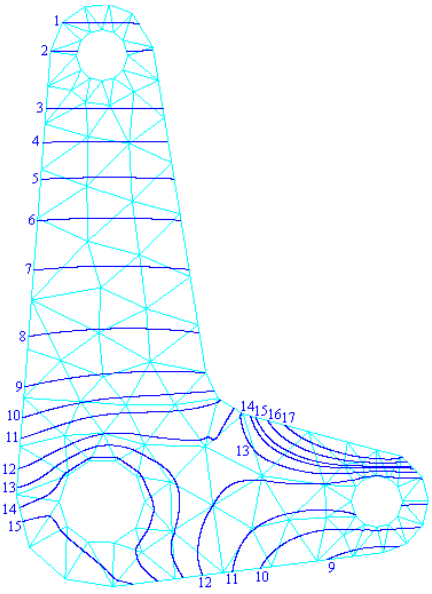
Figure 4.28(b) Boundary conditions and loading of the lever

Fig. 4.29(a) and 4.29(b) shows the contour plots of horizontal and vertical deflections using six-node triangles for different levels of deflection.



tour Levels	δ_y
1	-1.5 x10^{-2}
2	-1.0 x10^{-2}
3	-0.6 x10^{-2}
4	-0.4 x10^{-2}
5	-0.2 x10^{-2}
6	-0.05 x10^{-2}
7	0.01 x10^{-2}
8	0.1 x10^{-2}
9	0.2 x10^{-2}
10	0.5 x10^{-2}
11	1.0 x10^{-2}
12	1.3 x10^{-2}

Figure 4.29(a) Contour plot of vertical deflection using six-node triangles for the lever



Contour Levels	δ_x
1	-9.0 x10^{-2}
2	-8.0 x10^{-2}
3	-6.0 x10^{-2}
4	-5.0 x10^{-2}
5	-4.0 x10^{-2}
6	-3.0 x10^{-2}
7	-2.0 x10^{-2}
8	-1.0 x10^{-2}
9	-0.5 x10^{-2}
10	-0.3 x10^{-2}
11	-0.2 x10^{-2}
12	-0.1 x10^{-2}
13	-0.05 x10^{-2}
14	-0.01 x10^{-2}
15	0.01 x10^{-2}
16	0.05 x10^{-2}
17	0.1 x10^{-2}

Figure 4.29(b) Contour plot of horizontal deflection using six-node triangles for the lever

Fig. 4.30(a) shows the specifications of a plate. The plate is fixed at the hole on left hand side and loaded horizontally through the other two holes, as shown in Fig. 4.30(b). For a load of 1000 N, Young's Modulus $0.2\times10^{9}\ MPa$, Poisson's ratio 0.3, the values of deflections are derived using FEA, considering the problem as 2D plane problem with

negligible thickness. For the given loading conditions, the deflections are given by Eqs. (3.26) to (3.28).

Fig. 4.31(a) and 4.31(b) shows the contour plots of horizontal and vertical deflections using six-node triangles for different levels of deflections.

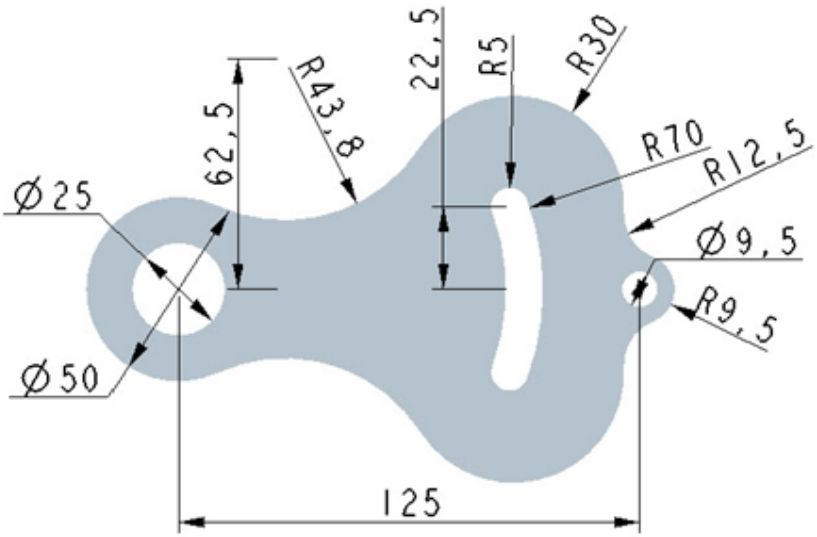


Figure 4.30(a) Specifications of the plate (all dimensions in mm)

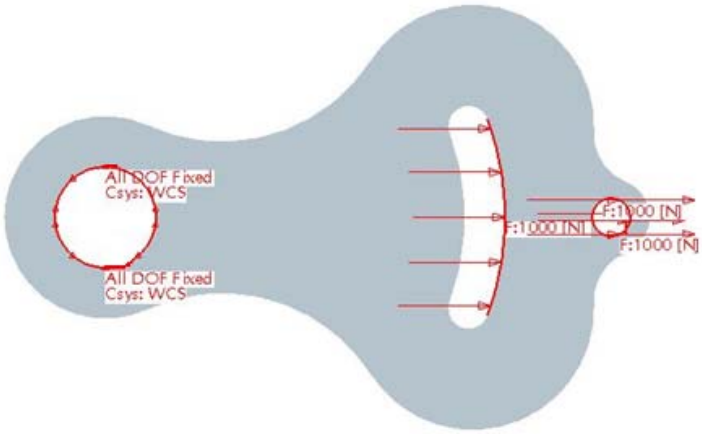
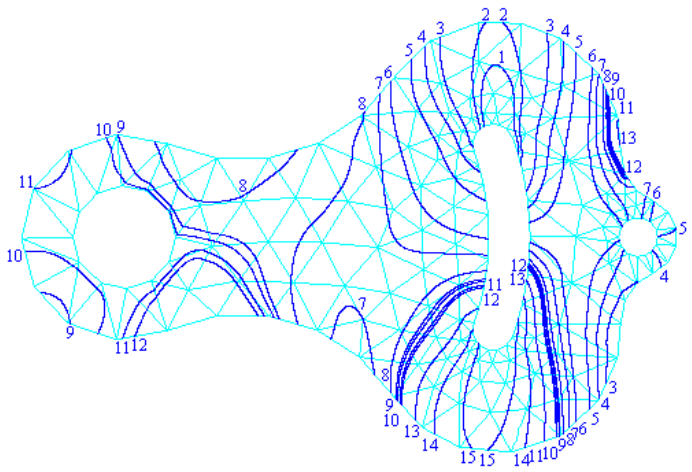
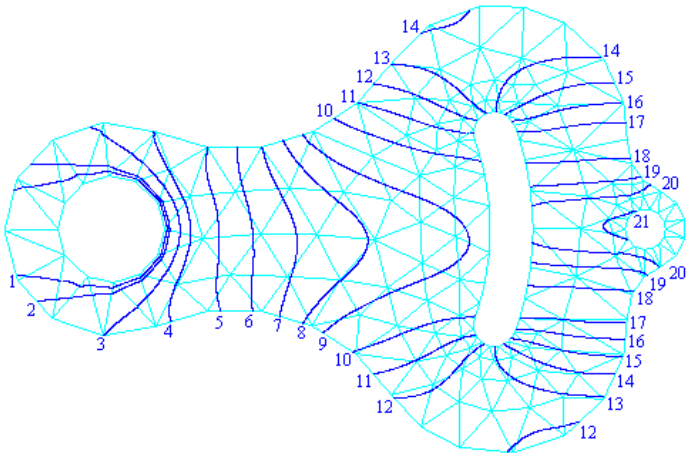


Figure 4.30(b) Boundary conditions and loading of the plate



Contour Levels	δ_y
1	-34 x10^{-4}
2	-32 x10^{-4}
3	-25 x10^{-4}
4	-20 x10^{-4}
5	-15 x10^{-4}
6	-08 x10^{-4}
7	-05 x10^{-4}
8	-02 x10^{-4}
9	-5 x10^{-4}
10	-1 x10^{-4}
11	0.6 x10^{-4}
12	1.0 x10^{-4}
13	5.0 x10^{-4}
14	10 x10^{-4}
15	15 x10^{-4}

Figure 4.31(a) Contour plot of vertical deflection using six-node triangles for the plate



Contour Levels	δ_x
1	0.5 x10^{-4}
2	1.0 x10^{-4}
3	3.0 x10^{-4}
4	5.0 x10^{-4}
5	10 x10^{-4}
6	15 x10^{-4}
7	20 x10^{-4}
8	25 x10^{-4}
9	30 x10^{-4}
10	40 x10^{-4}
11	50 x10^{-4}
12	60 x10^{-4}
13	70 x10^{-4}
14	80 x10^{-4}
15	90 x10^{-4}
16	100 x10^{-4}
17	110 x10^{-4}
18	130 x10^{-4}
19	140 x10^{-4}
20	145 x10^{-4}
21	149.5 x10^{-4}

Figure 4.31(b) Contour plot of horizontal deflection using six-node triangles for the plate

Fig. 4.32(a) shows the specifications of a plate. The plate is fixed at the two holes on right hand side and loaded horizontally through the hole at the left hand side, as shown in Fig. 4.32(b). For a load of 1000 N, Young's Modulus $0.2\times10^{9}\,MPa$, Poisson's ratio 0.3, the values of deflections are derived using FEA, considering the problem as 2D plane problem with negligible thickness. For the given loading conditions, the deflections are given by Eq. (3.25).

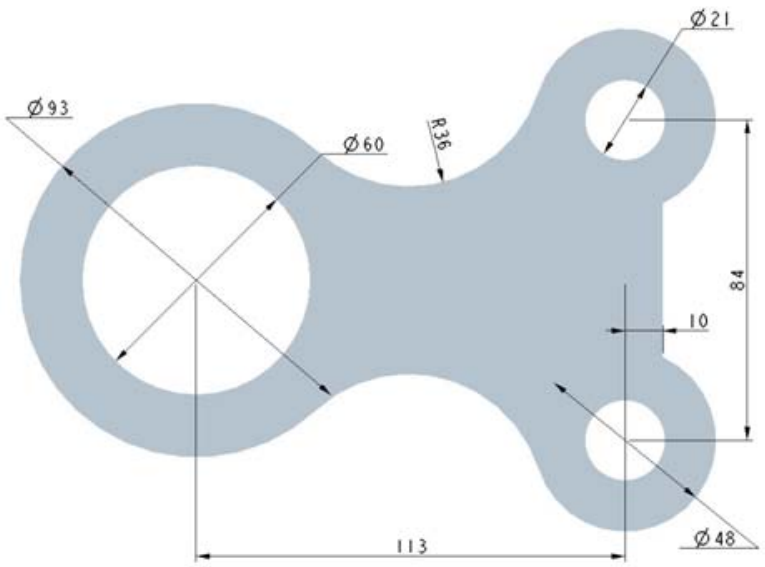


Figure 4.32(a) Specifications of the plate (all dimensions in mm)

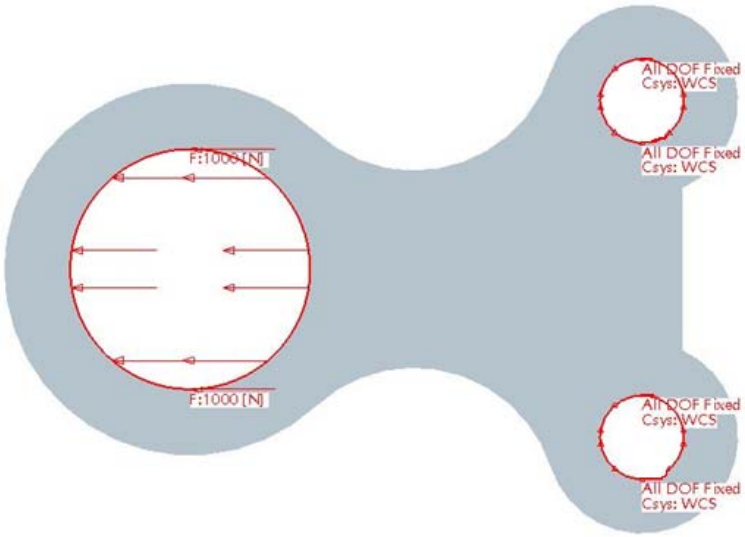


Figure 4.32(b) Boundary conditions and loading of the plate

Fig. 4.33(a) and 4.33(b) shows the contour plots of vertical and horizontal deflections using six-node triangles for different levels of deflections.

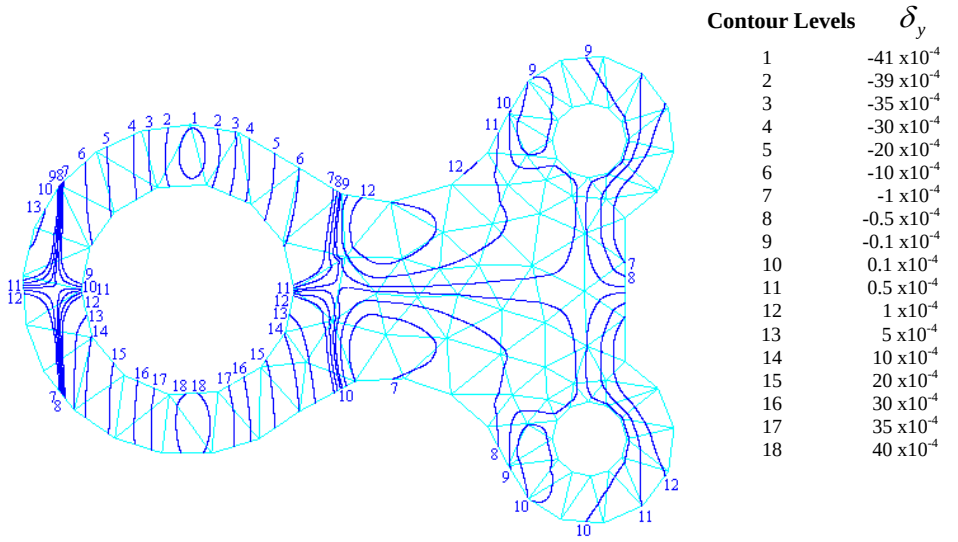


Figure 4.33(a) Contour plot of vertical deflection using six-node triangles for the plate

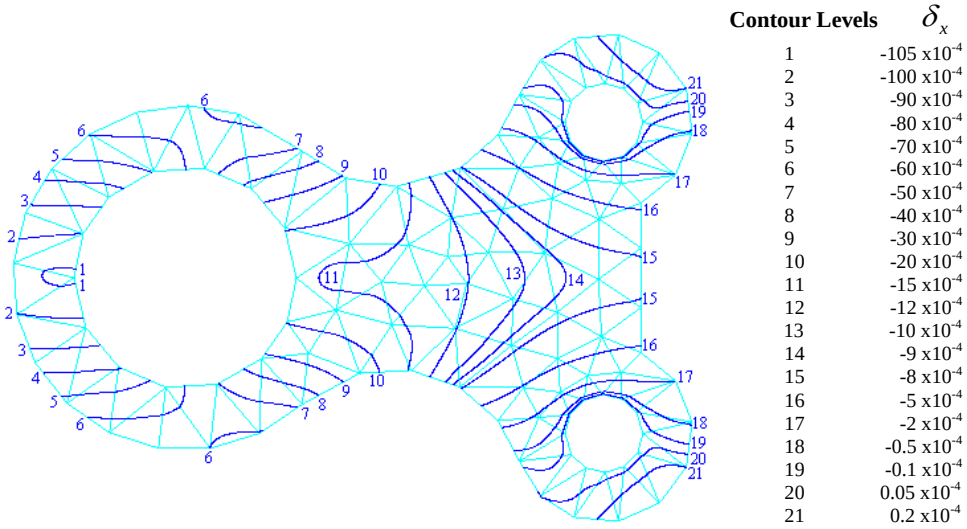


Figure 4.33(b) Contour plot of horizontal deflection using six-node triangles for the plate

The next problem is a heat conduction problem, with the specifications shown in Fig. 4.34(a). The temperature of 673^0C and 273^0C is fixed at the left and right side of the composite plates, as shown in Fig. 4.34(b). For the known heat flow rate, thermal conductivity of the materials is assumed as 0.01 W/m-deg (outer parts), 0.0057 W/m-deg (upper middle part) and 0.0015 W/m-deg (lower middle part) and the given boundary temperatures, the values of temperature distribution is derived using FEA.

The temperature distribution can be calculated by using Eq. (3.30) *i.e.*

$$Q = \frac{(t_1 - t_{n+1})}{\sum R_t}$$

For the given composite plates, the two middle plates are in parallel. Their equivalent thermal resistance, R_{eq} is given as

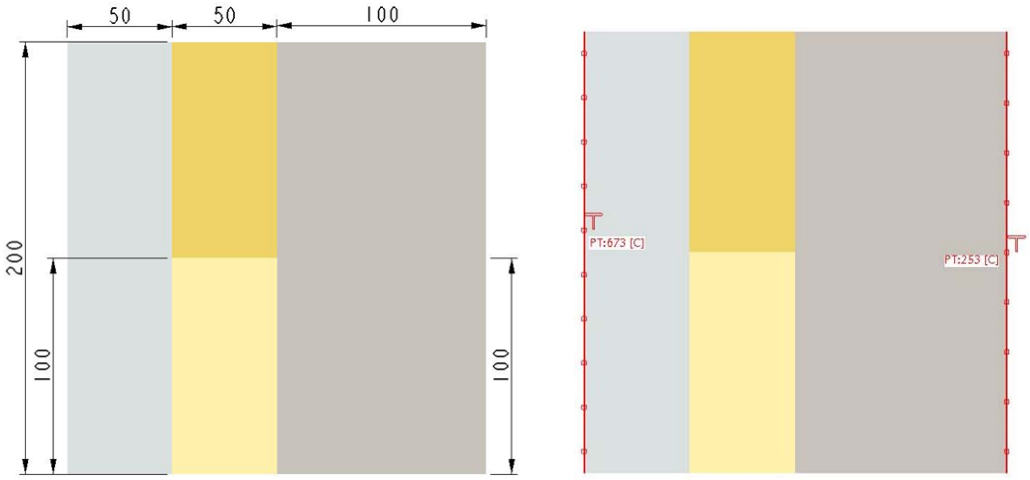
$$R_{eq} = \frac{R_b \times R_c}{R_b + R_c} \tag{4.27b}$$

This equivalent thermal resistance is in series with the two outer resistances. So, the total thermal resistance for the entire circuit becomes

$$\sum R_t = R_a + R_{eq} + R_d \tag{4.27c}$$

where R_a is the thermal resistance of the left outer wall, R_b is the thermal resistance of the upper middle wall, R_c is the thermal resistance of the lower middle wall and R_d is the thermal resistance of the right outer wall.

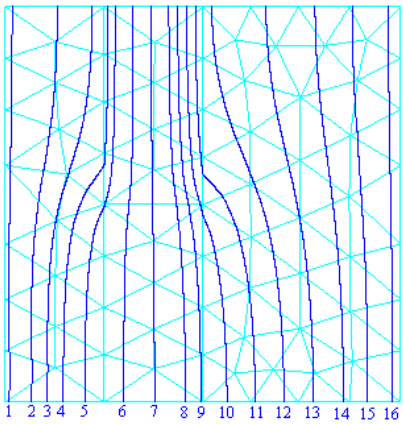
Fig. 4.35 shows the contour plots of temperature distribution using six-node triangles for different levels of temperature.



(a) **(b)**

Figure 4.34 (a) Specifications of composite plates (all dimensions in mm);

(b) Boundary conditions and temperature fixation of the plates



Contour Levels	T
1	670
2	650
3	635
4	620
5	600
6	550
7	500
8	450
9	425
10	400
11	375
12	350
13	325
14	300
15	280
16	260

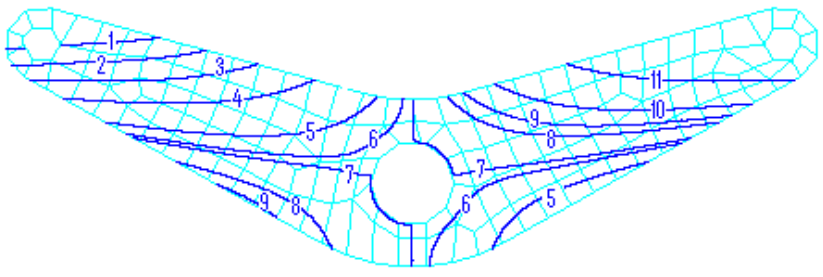
Figure 4.35 Contour plot of temperature distribution (in ^{o}C) using six-node triangles for the plates

4.6 IMPLEMENTATION OF THE ALGORITHM FOR EIGHT-NODE QUADRILATERAL ELEMENT

The contour plot algorithm, discussed in section 4.4 is implemented using the previously defined system configurations.

The contours are generated for the rocker arm with specifications, boundary and loading conditions shown in Fig. 3.15(a) and 3.15(b). For a load of 1000 N, Young's Modulus $0.2\times10^{9}\,MPa$, Poisson's ratio 0.3, the values of deflection are derived using eight-node quadrilateral elements, considering the problem as 2D plane problem with negligible thickness. As the arm is vertically loaded, the vertical and horizontal deflections are given by Eqs. (3.26) to (3.28).

The vertical deflections are considered for contouring in the numerical experimentation for the proposed algorithm. Fig. 4.36 depicts the contour plots of vertical deflections using eight-node quadrilateral elements for different levels of deflections of rocker arm.

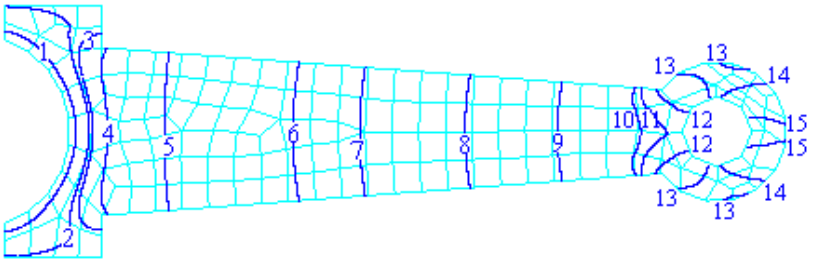


Contour Levels	δ_y
1	-180 x10^{-4}
2	-140 x10^{-4}
3	-100 x10^{-4}
4	-60 x10^{-4}
5	-20 x10^{-4}
6	-5 x10^{-4}
7	0.01 x10^{-4}
8	20 x10^{-4}
9	30 x10^{-4}
10	50 x10^{-4}
11	100 x10^{-4}

Figure 4.36 Contour plots of vertical deflection for rocker arm

The second problem analyzed is a deflection problem of a connecting rod, with specifications shown in Fig. 3.19(a). The rod is fixed at the left end and a horizontal load is applied through the hole on right hand side, as shown in Fig. 3.19(b). For the same material properties, the values of horizontal deflections are derived using FEA. As the rod is axially loaded, the general expression for direct horizontal deflection is given by Eq. (3.25).

Fig. 4.37 shows the contour plots of horizontal deflections using eight-node quadrilateral elements for different levels of deflections for the said component.



Contour Levels	δ_x
1	0.3 x10^{-4}
2	1.0 x10^{-4}
3	1.2 x10^{-4}
4	2.0 x10^{-4}
5	5.0 x10^{-4}
6	12 x10^{-4}
7	16 x10^{-4}
8	23 x10^{-4}
9	30 x10^{-4}
10	36.5 x10^{-4}
11	37.5 x10^{-4}
12	40 x10^{-4}
13	47 x10^{-4}
14	50 x10^{-4}
15	57 x10^{-4}

Figure 4.37 Contour plots of horizontal deflection for connecting rod

The next problem analyzed is a deflection problem of a proving ring, with specifications shown in Fig. 4.38(a). The ring is fixed at the lower end and a vertically upward force is applied at the top end, as shown in Fig. 4.38(b). For a load of 1000 N, Young's Modulus $0.2\times10^{9}\,MPa$, Poisson's ratio 0.3, the values of deflections are derived using eight-node quadrilateral elements, considering the problem as 2D plane problem with negligible thickness.

As the ring is axially loaded, the general expression for resultant stress, which can be used to calculate the vertical displacement (Singh, 2008) is given by Eq. (4.28).

$$\sigma_r = \frac{W\sin\theta}{2A} \pm \sigma \tag{4.28}$$

where W is the load applied, A is the area of the considered cross section, σ is the stress due to bending moment and θ is the angle of inclination along which the section is considered. The angle is with the line of action of the applied load.

The stress, σ is calculated as shown in Eq. (4.29)

$$\sigma_r = \frac{W}{A}\left[\frac{R^2}{\pi\left(R^2+h^2\right)}+\frac{R^2}{2h^2}\left\{\frac{2R^2}{\pi\left(R^2+h^2\right)}-\sin\theta\right\}\times\left(\frac{Ry}{R+y}\right)\right] \tag{4.29}$$

where R is the radius of the centre axis of the ring, h is a constant for the cross-section of the ring and y is the distance of the neutral axis.

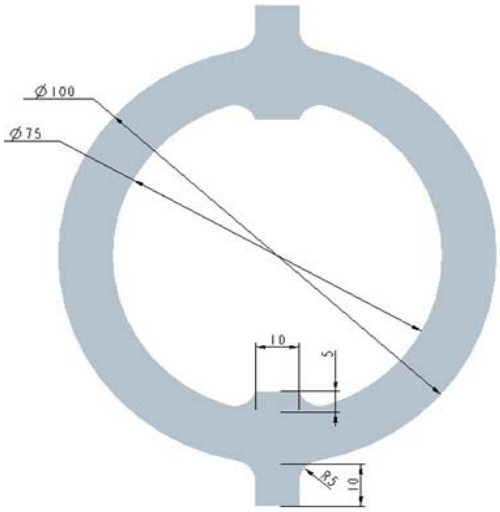


Figure 4.38(a) Specifications of the proving ring (all dimensions in mm)

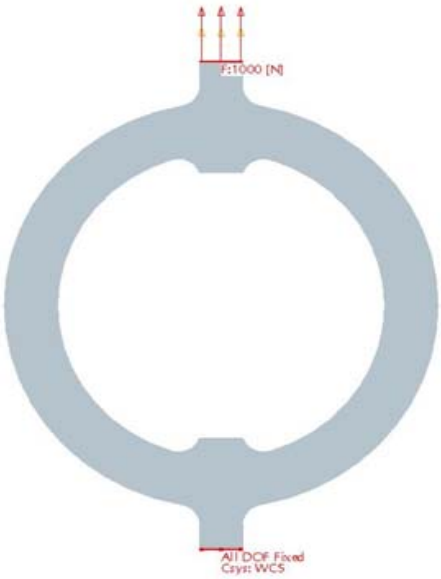
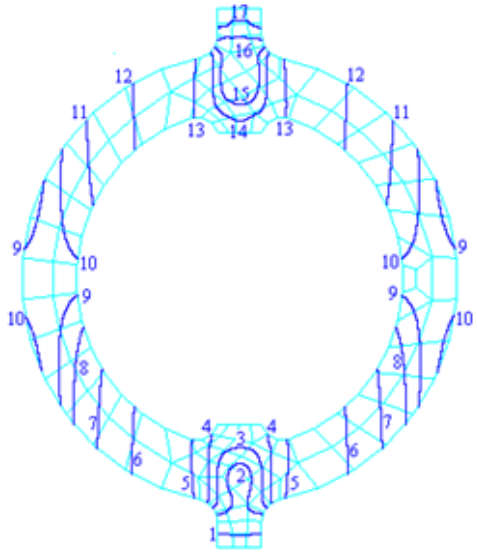


Figure 4.38(b) Boundary and loading conditions of the proving ring

Fig. 4.39 shows the contour plots of vertical deflections using eight-node quadrilateral elements for different levels of deflections for the proving ring.



Contour Levels	δ_y
1	1.0 x10^{-4}
2	2.6 x10^{-4}
3	3.5 x10^{-4}
4	5.0 x10^{-4}
5	10 x10^{-4}
6	50 x10^{-4}
7	80 x10^{-4}
8	100 x10^{-4}
9	110 x10^{-4}
10	120 x10^{-4}
11	140 x10^{-4}
12	180 x10^{-4}
13	220 x10^{-4}
14	225 x10^{-4}
15	226 x10^{-4}
16	227 x10^{-4}
17	228 x10^{-4}

Figure 4.39 Contour plots of vertical deflection for proving ring

Fig. 4.40(a) shows the specifications of a chain plate. The plate is fixed at the hole on left hand side and loaded horizontally through the hole at the right hand side, as shown in Fig. 4.40(b). For a load of 1000 N, Young's Modulus $0.2\times10^{9}\,MPa$, Poisson's ratio 0.3, the values of deflections are derived using FEA, considering the problem as 2D plane problem with negligible thickness. For the given loading conditions, the deflections are given by Eq. (3.25).

Fig. 4.41(a) and 4.41(b) shows the contour plots of vertical and horizontal deflections using eight-node quadrilateral for different levels of deflections.

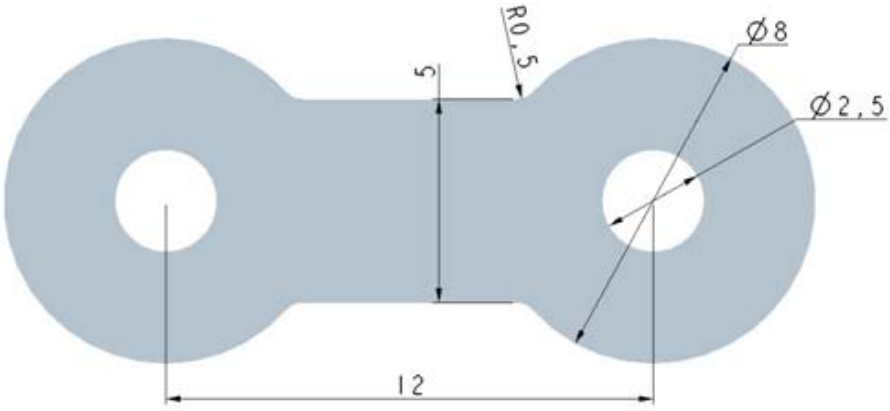


Figure 4.40(a) Specifications of the chain plate (all dimensions in mm)

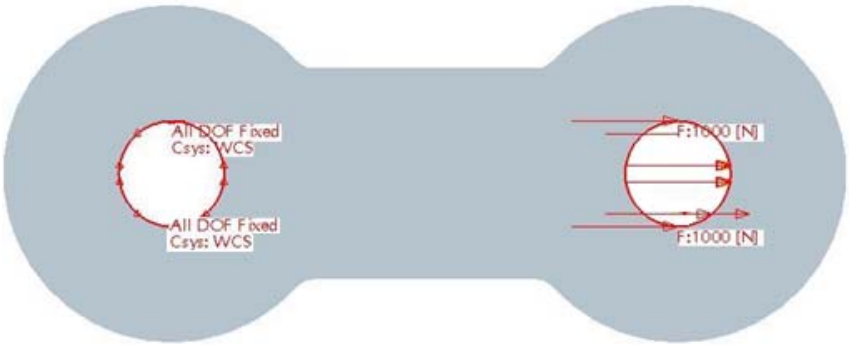
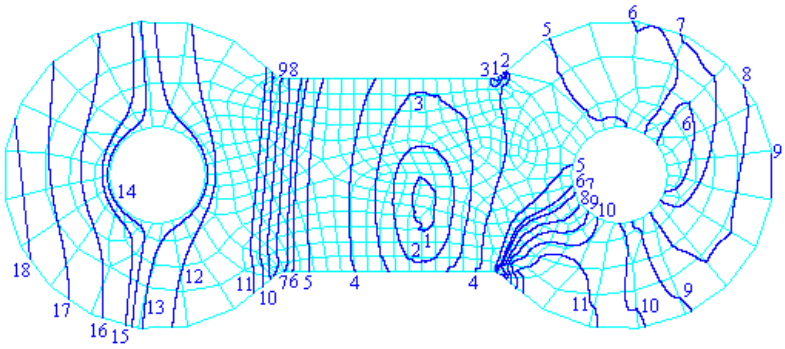
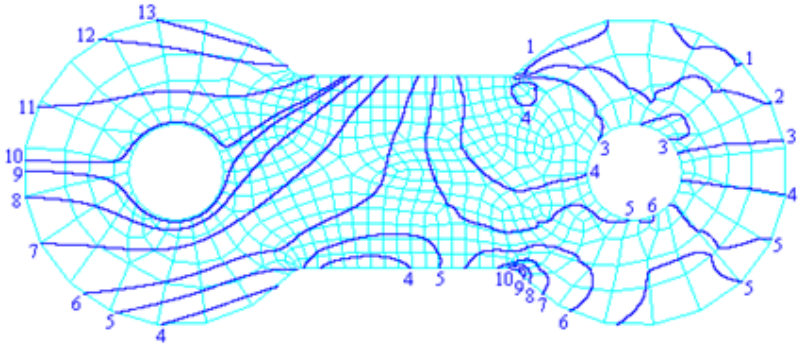


Figure 4.40(b) Boundary conditions and loading of the chain plate



Contour Levels	δ_y
1	-108 x10^{-13}
2	-105 x10^{-13}
3	-100 x10^{-13}
4	-90 x10^{-13}
5	-70 x10^{-13}
6	-63 x10^{-13}
7	-60 x10^{-13}
8	-55 x10^{-13}
9	-50 x10^{-13}
10	-45 x10^{-13}
11	-40 x10^{-13}
12	-10 x10^{-13}
13	-2 x10^{-13}
14	0.5 x10^{-13}
15	2.0 x10^{-13}
16	7.0 x10^{-13}
17	13 x10^{-13}
18	18 x10^{-13}

Figure 4.41(a) Contour plots of vertical deflection for chain plate

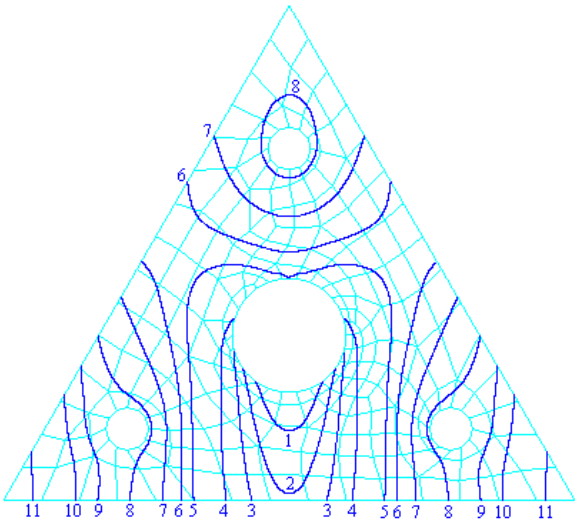


Contour Levels	δ_x
1	-550 x10^{-4}
2	-510 x10^{-4}
3	-450 x10^{-4}
4	-360 x10^{-4}
5	-300 x10^{-4}
6	-240 x10^{-4}
7	-130 x10^{-4}
8	-50 x10^{-4}
9	-10 x10^{-4}
10	5 x10^{-4}
11	80 x10^{-4}
12	180 x10^{-4}
13	230 x10^{-4}

Figure 4.41(b) Contour plots of horizontal deflection for chain plate

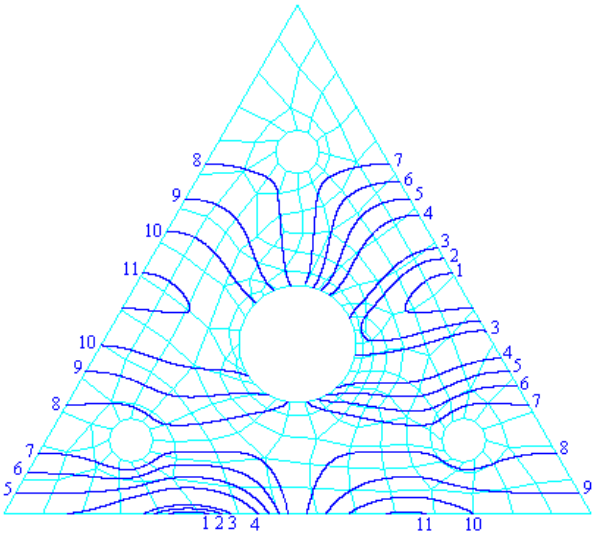
The contour plots are generated for a connecting plate shown in Fig. 3.21(a) and Fig. 3.23(a). The results are generated for the same material properties, boundary and loading conditions using eight-node quadrilateral elements.

Fig. 4.42(a), 4.42(b), 4.43(a) and 4.43(b) shows the contour plots of vertical and horizontal deflections using eight-node quadrilateral elements for different levels of deflections for the said two plates.



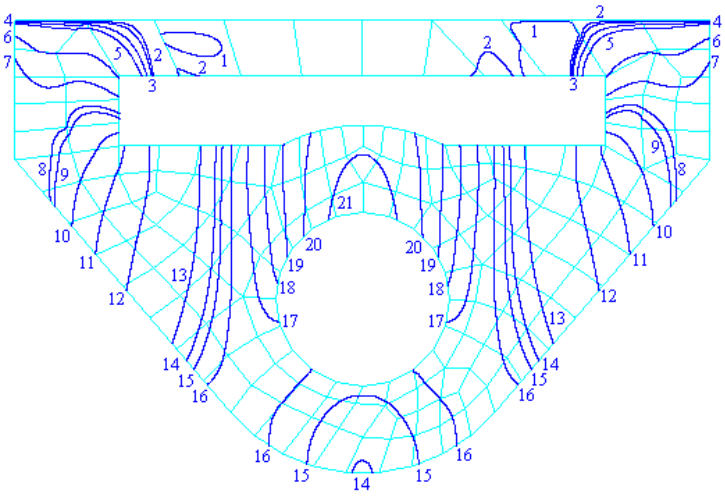
Contour Levels	δ_y
1	-22 x10^{-4}
2	-20 x10^{-4}
3	-18 x10^{-4}
4	-15 x10^{-4}
5	-10 x10^{-4}
6	-8 x10^{-4}
7	-5 x10^{-4}
8	-1 x10^{-4}
9	1 x10^{-4}
10	2 x10^{-4}
11	4 x10^{-4}

Figure 4.42(a) Contour plots of vertical deflection for the plate



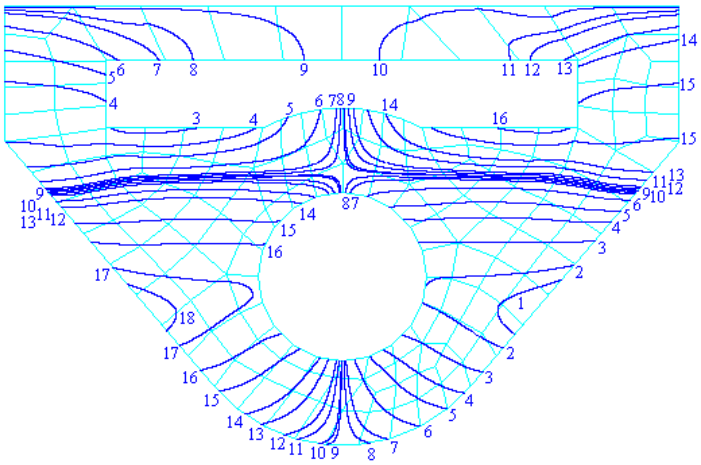
Contour Levels	δ_x
1	-7.0 x10^{-4}
2	-6.5 x10^{-4}
3	-6.0 x10^{-4}
4	-4.0 x10^{-4}
5	-3.0 x10^{-4}
6	-2.0 x10^{-4}
7	-1.0 x10^{-4}
8	1.0 x10^{-4}
9	3.0 x10^{-4}
10	5.0 x10^{-4}
11	7.0 x10^{-4}

Figure 4.42(b) Contour plots of horizontal deflection for the plate



Contour Levels	δ_y
1	0.025 x10^{-4}
2	-0.04 x10^{-4}
3	-0.1 x10^{-4}
4	-0.3 x10^{-4}
5	-0.6 x10^{-4}
6	-3 x10^{-4}
7	-6 x10^{-4}
8	-9 x10^{-4}
9	-10 x10^{-4}
10	-15 x10^{-4}
11	-25 x10^{-4}
12	-40 x10^{-4}
13	-60 x10^{-4}
14	-65 x10^{-4}
15	-67 x10^{-4}
16	-70 x10^{-4}
17	-75 x10^{-4}
18	-80 x10^{-4}
19	-85 x10^{-4}
20	-90 x10^{-4}
21	-95 x10^{-4}

Figure 4.43(a) Contour plots of vertical deflection for the plate



Contour Levels	δ_x
1	22 x10^{-4}
2	20 x10^{-4}
3	15 x10^{-4}
4	10 x10^{-4}
5	6.5 x10^{-4}
6	3 x10^{-4}
7	1 x10^{-4}
8	0.5 x10^{-4}
9	0.05 x10^{-4}
10	-0.05 x10^{-4}
11	-0.6 x10^{-4}
12	-1.2 x10^{-4}
13	-3 x10^{-4}
14	-5 x10^{-4}
15	-10 x10^{-4}
16	-15 x10^{-4}
17	-20 x10^{-4}
18	-22 x10^{-4}

Figure 4.43(b) Contour plots of horizontal deflection for the plate

4.6.1 Speed of the Algorithm

The speed of the algorithm used in eight node quadrilateral element is compared with that of Stelzer and Welzel (1987). The algorithm of Stelzer and Welzel (1987) divides the square $[-1,+1]\times[-1,+1]$ into m equal parts by dividing the $r-$ and $s-$ axes and assuming straight lines parallel to the axes, m being arbitrary. The contour Eq. (4.20) is, therefore solved m^2 times. When the segment intersects a small part of the element, unnecessary computations are performed over the free region. The present algorithm is designed to avoid such type of unnecessary computations resulting in drastic reduction of

execution time. The program based on the present algorithm takes 40-50% of the lesser time as taken by an implementation of the algorithm given by Stelzer and Welzel (1987) for the different problems shown in the present work. Table 4.3 below gives the comparison of execution time using the two algorithms.

Component	No. of Elements	No. of Nodes	No. of Contour Levels	Execution Time (seconds) for Stelzer and Welzel (1987)	Execution Time (seconds) for Proposed Algorithm
Rocker Arm	144	524	11	1.428571	0.824176
Connecting Rod	156	554	15	1.043956	0.604396
Proving Ring	129	493	17	1.648352	0.879121

Table 4.3 Comparison of execution time

4.7 CONCLUDING REMARKS

Higher order elements are mostly used for their better results, closer approximation of curved boundaries and convergence to the exact solution with fewer elements. Higher order triangular and quadrilateral elements are most commonly used in 2D problems for analysis.

The quadratic nature of the interpolation in six-node triangular element is analyzed in detail and algorithms are presented to accurately interpolate the function representing the various conic sections in 2D. Data structures and algorithm for joining the contour segments derived on individual elements are presented. Numerical results presented are compared with contours using three-node triangular elements. The contour generated using six-node elements are found to be more accurate than three-node triangular element.

A fast algorithm for the plotting of contours using eight-node quadrilateral elements is described. The contours are accurately generated over each element using the interpolation functions. The contour joining algorithm discussed here is considerably faster than the existing techniques.

After the completion of algorithms using 2D elements, the concentration is shifted towards 3D elements. The next chapter is devoted to the 3D linear elements.

CHAPTER - V

CONTOURING IN 3D USING LINEAR ELEMENTS

5.1 INTRODUCTION

Tetrahedral elements are very commonly used in analysis because of their simple forms and better adaptability to fit in the irregular 3D domains. Many 3D engineering problems are also analyzed using hexahedral elements because they provide more accurate and stable results as compared to tetrahedral elements.

In 3D, the amount of data becomes manifold as compared to 2D and the representation of data in the form of contour surfaces becomes very important. It is a common practice in FEA to use higher order elements for better accuracy and stability of results of the analysis. However, generation of contour lines or contour surfaces using higher order elements is a complex task. In order to simplify the contour surface generation, normally a higher order element is split into linear elements and linear interpolation of the physical quantity is performed. This results in the linear patches of contour surfaces.

One of the common examples is the decomposition of an eight-node hexahedral element into 6 four-node tetrahedral elements. The eight-node hexahedral element provides a non-linear interpolation of a physical quantity while the four-node tetrahedral element performs linear interpolation. In order to make the overall task simple, the analysis is performed with higher order elements whereas the graphical display of results in the form of contour lines or surfaces is carried out with linear elements.

In the first part of this chapter, the detail of contour plotting using four-node tetrahedral elements is given. It provides a linear interpolation over its domain. The contour equation over four-node tetrahedral element is developed using shape functions. The developed contour equation represents a plane. The contour is traced by finding the intersection of a plane with the edges of the tetrahedral.

In the second part of this chapter, a technique for accurate and fast tracing of contour surfaces using hexahedral elements is discussed.

5.2 CONTOUR EQUATION OVER TETRAHEDRAL ELEMENT

For a four-node tetrahedral element, the contour equation in 3D assumes the form given in Eq. (5.1).

$$f(x, y, z) = C \tag{5.1}$$

where x, y, z are spatial coordinates and C is a constant.

A four node tetrahedral element normally used for analysis is given in Fig. 5.1.

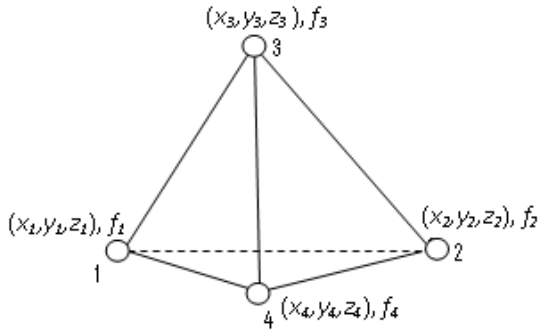


Figure 5.1 A four-node tetrahedral element

The coordinates of the four vertices of the tetrahedral are available prior to analysis and the physical quantity, *f(x,y,z)*, is known after analysis, *i.e.* after solving the mathematical model. The linear interpolation of coordinates and function over the element is performed as shown in Eqs. (5.2) to (5.5).

$$x = \sum_{i=1}^{4} N_i x_i \tag{5.2}$$

$$y = \sum_{i=1}^{4} N_i y_i \tag{5.3}$$

$$z = \sum_{i=1}^{4} N_i z_i \tag{5.4}$$

$$f(x, y, z) = \sum_{i=1}^{4} N_i f_i \tag{5.5}$$

where $N_i = L_i$ and $L_i = \frac{1}{6\Delta}\left(a_i + b_i x + c_i y + d_i z\right)$ (5.6)

The coefficients a_i ,b_i, c_i and d_i are cofactors of the matrix, M, where

$$M = \begin{bmatrix} 1 & x_i & y_i & z_i \\ 1 & x_j & y_j & z_j \\ 1 & x_m & y_m & z_m \\ 1 & x_p & y_p & z_p \end{bmatrix} \tag{5.7}$$

and, $6\Delta = \det(M)$. The functions $L_i = L_i(x, y, z), \quad i = 1,2,3,4$ are called shape functions.

The Eq. (5.5) is simplified to the form shown in Eq. (5.8).

$$l_0 + l_1 x + l_2 y + l_3 z = C \tag{5.8}$$

where,

$$l_0 = \frac{1}{6\Delta}(a_1 f_1 + a_2 f_2 + a_3 f_3 + a_4 f_4)$$

$$l_1 = \frac{1}{6\Delta}(b_1 f_1 + b_2 f_2 + b_3 f_3 + b_4 f_4)$$

$$l_2 = \frac{1}{6\Delta}(c_1 f_1 + c_2 f_2 + c_3 f_3 + c_4 f_4)$$

$$l_3 = \frac{1}{6\Delta}(d_1 f_1 + d_2 f_2 + d_3 f_3 + d_4 f_4)$$

Eq. (5.8) can be compared with the following general equation which represents a plane

$$F(x, y, z) = ax + by + cz + d \tag{5.9}$$

The problem now reduces to accurately trace the contour represented by a plane over a tetrahedral.

5.2.1 Tracing Contour Surface over a Tetrahedral Element

As discussed in 2D, the intersection of an edge of a tetrahedral with the contour plane using parametric equations (Zeid, 2007) is found and the point of intersection, if any, is recorded. Let the parametric equation, in vector form for an edge given by end points P_1 and P_2 be given by Eq. (5.10).

$$P = P_1 + u(P_2 - P_1), \qquad u \in [0,1] \tag{5.10}$$

And the general equation of the contour plane be given by Eq. (5.11).

$$l_1 x + l_2 y + l_3 z + l_4 = 0 \tag{5.11}$$

On substituting *x*, *y* and *z* from Eq. (5.10) into Eq. (5.11) and rearranging, Eq. (5.12) is obtained

$$u = \frac{(l_1 x_1 + l_2 y_1 + l_3 z + l_4)}{(l_1(x_1 - x_2) + l_2(y_1 - y_2) + l_3(z_1 - z_2))} \tag{5.12}$$

The intersection point lies between P_1 and P_2 if the value of *u* lies between 0 to 1.

Similarly, the intersection points on all the edges are calculated and joined with the help of a line to form a contour plane over an element. The shading of the plane is done using OpenGL graphics libraries.

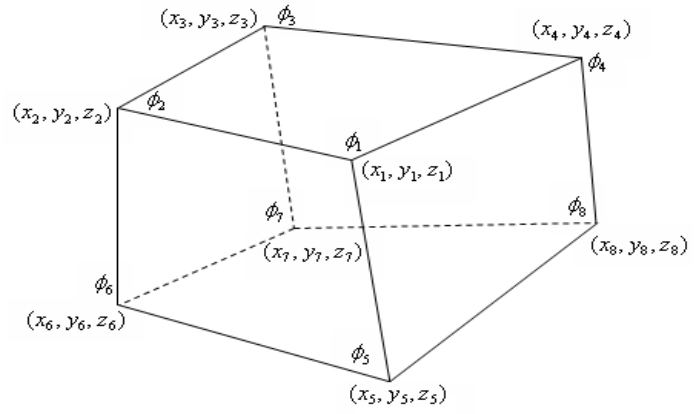
5.3 CONTOUR SURFACES IN 3D USING HEXAHEDRAL ELEMENTS

This section presents a fast and accurate method for generating contour surfaces after deriving its boundary curves and then locating the interior points of the contour surfaces. The accuracy of the contour surfaces is the same as that obtained by direct method (Rajasekaran and Venkatesan, 1995), however, computation time is reduced significantly. The proposed method provides contour surface as a group of bounding curves over an element. The interelement continuity of contour surfaces provides a way for creating independent contour surfaces within the entire domain defined by the assemblage of all hexahedral elements.

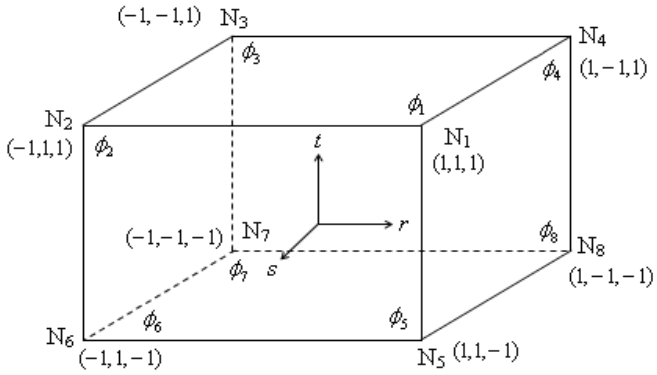
A 3D domain, D is assumed to be represented by non-intersecting hexahedral elements expressed by Eq. (5.13).

$$D = \bigcup_{i=1}^{n} H_i \tag{5.13}$$

where H_1, H_2, ….., H_n are n hexahedral elements, such that $H_i \bigcap H_j = \phi$, $i \neq j$. Such applications arise in finite element analysis of many physical problems. A hexahedral element is represented in Fig. 5.2(a) in global coordinates whose local coordinate representation is given in Fig. 5.2(b).



(a)



(b)

Figure 5.2 A hexahedral element in (a) Global coordinates; (b) Local coordinates

The interpolation functions, $N_i(r,s,t),\ i=1,2,.....,8,$ in local coordinates are given by

$$N_i = \frac{1}{8}(1+rr_i)(1+ss_i)(1+tt_i), \qquad r_i,\ s_i,\ t_i = \pm 1 \tag{5.14}$$

The following expressions are used to convert local coordinates into global coordinates.

$$x = \sum_{i=1}^{8} N_i x_i \quad , \qquad y = \sum_{i=1}^{8} N_i y_i \quad , \qquad z = \sum_{i=1}^{8} N_i z_i \tag{5.15}$$

The interpolation of the function $\phi(r,s,t)$ is carried out in local coordinates as given by

$$\phi(r,s,t) = \sum_{i=1}^{8} N_i \phi_i \tag{5.16}$$

Substituting the values of interpolation functions from Eq. (5.14) into Eq. (5.16).

$$\phi(r,s,t) = a_0 + a_1 r + a_2 s + a_3 t + a_4 rs + a_5 st + a_6 tr + a_7 rst \tag{5.17}$$

where,

$$a_0 = \frac{1}{8}(\phi_1 + \phi_2 + \phi_3 + \phi_4 + \phi_5 + \phi_6 + \phi_7 + \phi_8)$$

$$a_1 = \frac{1}{8}(\phi_1 - \phi_2 - \phi_3 + \phi_4 + \phi_5 - \phi_6 - \phi_7 + \phi_8)$$

$$a_2 = \frac{1}{8}(\phi_1 + \phi_2 - \phi_3 - \phi_4 + \phi_5 + \phi_6 - \phi_7 - \phi_8)$$

$$a_3 = \frac{1}{8}(\phi_1 + \phi_2 + \phi_3 + \phi_4 - \phi_5 - \phi_6 - \phi_7 - \phi_8) \tag{5.18}$$

$$a_4 = \frac{1}{8}(\phi_1 - \phi_2 + \phi_3 - \phi_4 + \phi_5 - \phi_6 + \phi_7 - \phi_8)$$

$$a_5 = \frac{1}{8}(\phi_1 + \phi_2 - \phi_3 - \phi_4 - \phi_5 - \phi_6 + \phi_7 + \phi_8)$$

$$a_6 = \frac{1}{8}(\phi_1 - \phi_2 - \phi_3 + \phi_4 - \phi_5 + \phi_6 + \phi_7 - \phi_8)$$

$$a_7 = \frac{1}{8}(\phi_1 - \phi_2 + \phi_3 - \phi_4 - \phi_5 + \phi_6 - \phi_7 + \phi_8)$$

The contour surface is traced by substituting, $\phi(r,s,t) = C$ in Eq. (5.17), which is given by Eq. (5.19).

$$C = a_0 + a_1 r + a_2 s + a_3 t + a_4 rs + a_5 st + a_6 tr + a_7 rst \tag{5.19}$$

where, C is the contour level.

5.3.1 Direct Method for Tracing Contour Surfaces

The Eq. (5.17) for contour surface is expressed as Eq. (5.20).

$$r = \psi(s,t) = \frac{C - (a_0 + a_2 s + a_3 t + a_5 st)}{a_1 + a_4 s + a_6 t + a_7 st} \tag{5.20}$$

The contour surface is traced by letting the parameters s and t vary between -1.0 and +1.0 and computing r. If $|r| \le 1$, then the point $P(r,s,t)$ lies on the contour surface. The pseudocode given in Fig. 5.3 explains this process more clearly.

1. *Set n (total number of intervals) for* $r \in [-1,1]$ *and* $s \in [-1,1]$ *and a contour level C.*
2. *Compute* $\Delta s = \frac{2}{n}$ *and* $\Delta t = \frac{2}{n}$
3. *for* $i = 0$ *to* n

   $s_i = -1 + i * \Delta s$

   for $j = 0$ *to* n

   $t_j = -1 + j * \Delta t$

$r_{ij} = \psi(s_i, t_j)$ *(Eq. (5.20))*

if $|r_{ij}| \leq 1.0$ *then*

save $P[r_{ij}, s_i, t_j]$

endif

endfor

endfor

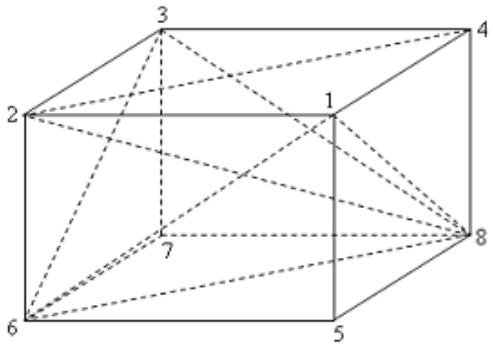
Figure 5.3 Pseudocode for direct method of tracing contour surface

The points $P[r_{ij}, s_i, t_j]$ as determined in the pseudocode lie on the contour surface in local coordinate system. The global coordinates of these points are determined using Eq. (5.15).

Although the direct method is straightforward in its implementation, its order of time complexity, $O(n^2)$, is very high. An entire solution domain for a tetrahedral element requires the evaluation of the function in Eq. (5.20) n^2 times where n is the number of points in the interval $[-1,1]$.

5.3.2 Contouring By Linear Interpolation

The function $r = \psi(s,t)$ is non-linear which is evident from Eq. (5.19). One of the methods of reducing the time complexity is to perform linear interpolation by decomposing a hexahedral element into 6 four-node tetrahedral elements and then tracing the contour surfaces on each tetrahedral which turn out to be planner surfaces. Fig. 5.4 depicts the process of decomposition of a hexahedral element.



Tetrahedral	4-Nodes
1	8 7 6 3
2	8 2 6 3
3	8 6 5 1
4	8 6 2 1
5	8 4 2 1
6	8 4 3 2

Figure 5.4 A hexahedral element with its local node numbering and its decomposition into 6 tetrahedral elements

The process of linear interpolation of contour planes using tetrahedral elements is achieved by performing linear interpolation of the function $\phi(x,y,z)$ in global coordinates over each of the 4 triangular faces of the tetrahedrals. Let the local numbers of the vertices of a triangular face of a tetrahedral (Fig. 5.5(a)) be given as 1, 2 and 3 with the function values ϕ_1, ϕ_2 and ϕ_3, respectively, as shown in Fig. 5.5(b). A contour surface $\phi(x,y,z)=C$ will intersect the face if $\phi_{\min} \le C \le \phi_{\max}$, where $\phi_{\min} = \min_{1\le i\le 3}\{\phi_i\}$ and $\phi_{\max} = \max_{1\le i\le 3}\{\phi_i\}$. Assuming $\phi_1 \le C \le \phi_2$, then the intersection is obtained by computing the parameter u as expressed in Eq. (5.21).

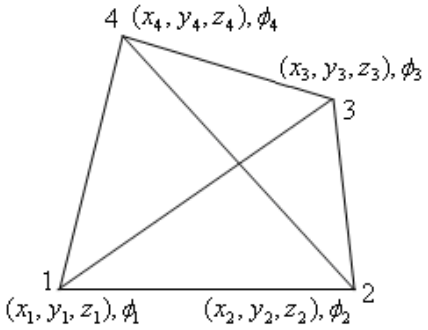
$$u = \frac{C-\phi_1}{\phi_2-\phi_1} \tag{5.21}$$

Using the equation of line passing through the vertices 1 and 2 in parametric form; expression as given by Eq. (5.22) are obtained.

$$x_t = x_1 + u(x_2 - x_1),\quad y_t = y_1 + u(y_2 - y_1),\ \text{and}\ \ z_t = z_1 + u(z_2 - z_1) \tag{5.22}$$

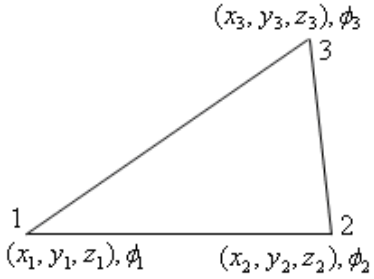
If a contour surface intersects an edge of a triangle, then due to continuity and linear interpolation it will also intersect another edge of the triangle. At the most two edges will be intersected by the contour surface. Two special cases arise. In the first case, if $C=\phi_1=\phi_2$, then the edge will be a contour line and in the second case if $C=\phi_1=\phi_2=\phi_3$ then the triangle itself will be a contour plane.

After the two intersection points are determined, these are joined to form a boundary of the contour surface. Contour boundaries are determined on all four triangular faces of the tetrahedral. The contour boundaries will form a closed polygonal surface in 3D as depicted in Fig. 5.5(c). Fig. 5.5(c) depicts a contour plane with four boundaries for given function values at four vertices of a tetrahedral with $C = 25.0$. The overall procedure is depicted through a pseudocode given in Fig. 5.6.

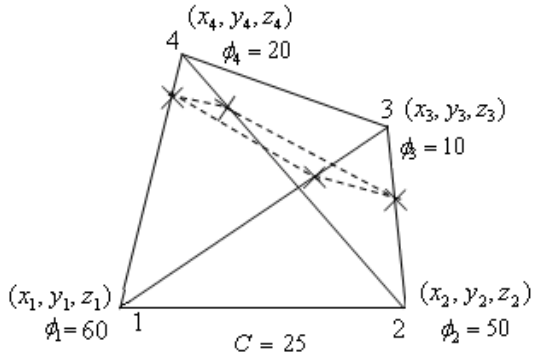


Triangles	Node numbers
1	1 2 3
2	1 2 4
3	2 3 4
4	1 3 4

(a)



(b)



(c)

Figure 5.5(a) A tetrahedral element and its local node numbering; (b) A face of the tetrahedral; (c) A contour plane with four boundaries

1. *Split a hexahedral into 6-tetrahedrals*
2. *For each tetrahedral repeat*
   - *(i) split a tetrahedral into 4 triangular faces*
   - *(ii) for each face do the following*
     - *(a) consider an edge of the triangle and take* ϕ_1 *and* ϕ_2
     - *(b) if* $\phi_1 \le C \le \phi_2$ *then find the point of intersection,* $P_t(x_t, y_t, z_t)$
     - *(c) repeat steps (a) and (b) for all the three edges. Only two edges will make intersections*
   - *(iii) join the two points of intersections* P_{t1} *and* P_{t2} *obtained in step (ii) to form a boundary*
   - *(iv) join all boundaries to form a planner contour surface*

Figure 5.6 Pseudocode for generating a planner contour surface within a tetrahedral element

Thus we approximate the curved contour surfaces by planner surfaces. It is a very fast approach as we do not have to evaluate the contour surface given by Eq. (5.19). However, the speed up is achieved at the cost of accuracy of the contour surface.

5.3.3 Accurate and Fast Generation of Contour Surface

This section proposess an accurate contour surface plotting algorithm which is as accurate as the direct method but with much low time complexity. The algorithm first determines the boundaries of a contour surface within a hexahedral in the form of a closed polygon. The closed boundary provides the reduced extents of the domain of search for contour surface without testing whether a point falls in the interior of the contour surface or not unlike the situation in the direct method. This is the basis of the reduced time complexity of the proposed method. We explain the procedure through an example and later on a general algorithm will be presented. Fig. 5.7 is a hexahedral element in local coordinates which is a cube with length 2 units. The vertices are numbered as $N_1(1.0,1.0,1.0), N_2(-1.0,1.0,1.0), N_3(-1.0,-1.0,1.0)$, $N_4(1.0,-1.0,1.0)$, $N_5(1.0,1.0,-1.0)$, $N_6(-1.0,1.0,-1.0)$, $N_7(-1.0,-1.0,-1.0)$, $N_8(1.0,-1.0,-1.0)$ and the function values are $\phi_1 = 80$, $\phi_2 = 70$, $\phi_3 = 60$, $\phi_4 = 50$, $\phi_5 = 40$, $\phi_6 = 30$, $\phi_7 = 20$, $\phi_8 = 10$. A contour surface for the contour level $C = 25.0$ is traced which is in the form of a four-sided polygon with vertices $P_0(-1.0,\ 0.0,-1.0)$, $P_1(1.0,\ 0.0,\ -1.0)$, $P_2(1.0,\ -1.0,\ -0.25), P_3(-1.0,\ -1.0,\ -0.75)$. The sides of the polygon are, in general, curves which are contour lines on the planner faces of the hexahedral. In the present example, incidentally they are straight lines. The contour line joining the points P_0 and P_1 is derived on the face $N_5N_6N_7N_8$. Similarly the contour lines joining P_1 and P_2 , P_2 and P_3 , P_3 and P_0 are derived on the faces $N_1N_5N_8N_4$, $N_3N_4N_8N_7$, $N_2N_3N_7N_6$, respectively. There is no contour line on the faces $N_1N_2N_6N_5$ and $N_1N_2N_3N_4$. The interpolation function, Eq. (5.19), is C^0 continuous, thus enabling the process of joining the contour lines generated on the individual faces of the cube to form outlines of the curved contour surface. There can be at the most 4 disjoint contour surfaces on a hexahedral element as shown in Fig. 5.7 and Fig. 5.8(a)-5.8(c) which show one, two, three and four disjoint contour surfaces, respectively, on a hexahedral element in local coordinate system.

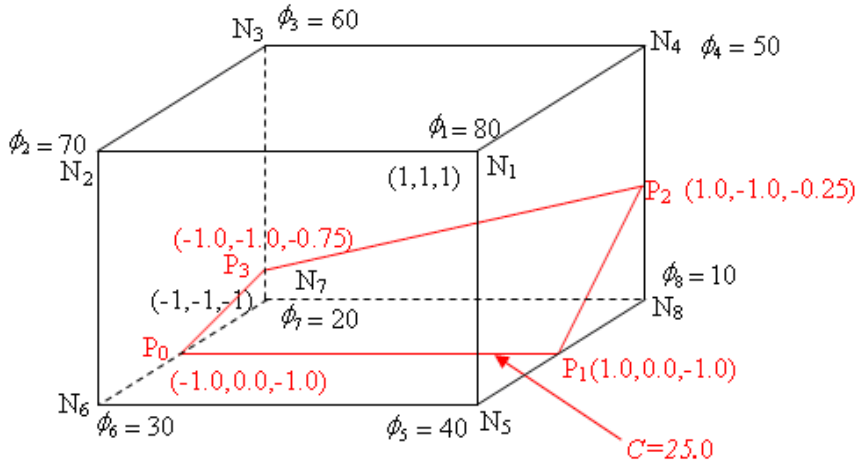
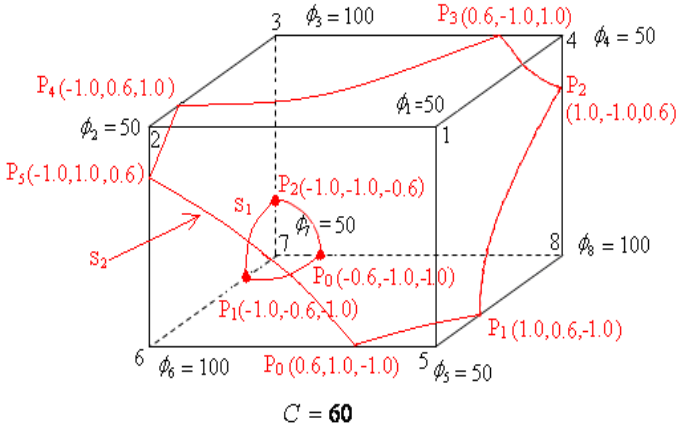


Figure 5.7 A hexahedral element (cube with length 2 units) in local coordinates and a contour surface for the contour level $C = 25.0$



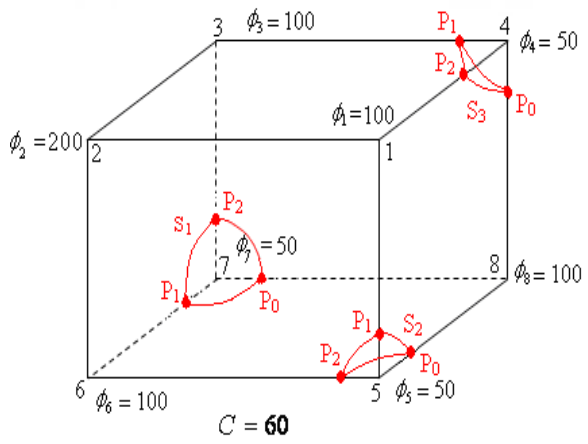
Outline of first contour surface, S_1

P_0	(-0.6,-1.0,-1.0)
P_1	(-1.0,-0.6,-1.0)
P_2	(-1.0,-1.0,-0.6)

Outline of second contour surface, S_2

P_0	(0.6, 1.0,-1.0)
P_1	(1.0, 0.6,-1.0)
P_2	(1.0,-1.0, 0.6)
P_3	(0.6,-1.0, 1.0)
P_4	(-1.0, 0.6,1.0)
P_5	(-1.0, 1.0,0.6)

Figure 5.8(a) Two disjoint contour surfaces



Outline of first contour surface, S_1

P_0	(-0.6,-1.0,-1.0)
P_1	(-1.0,-0.6,-1.0)
P_2	(-1.0,-1.0,-0.6)

Outline of second contour surface, S_2

P_0	(1.0,0.6,-1.0)
P_1	(1.0,1.0,-0.6)
P_2	(0.6,1.0,-1.0)

Outline of third contour surface, S_3

P_0	(1.0,-1.0,0.6)
P_1	(0.6,-1.0,1.0)
P_2	(1.0,-0.6,1.0)

Figure 5.8(b) Three disjoint contour surfaces

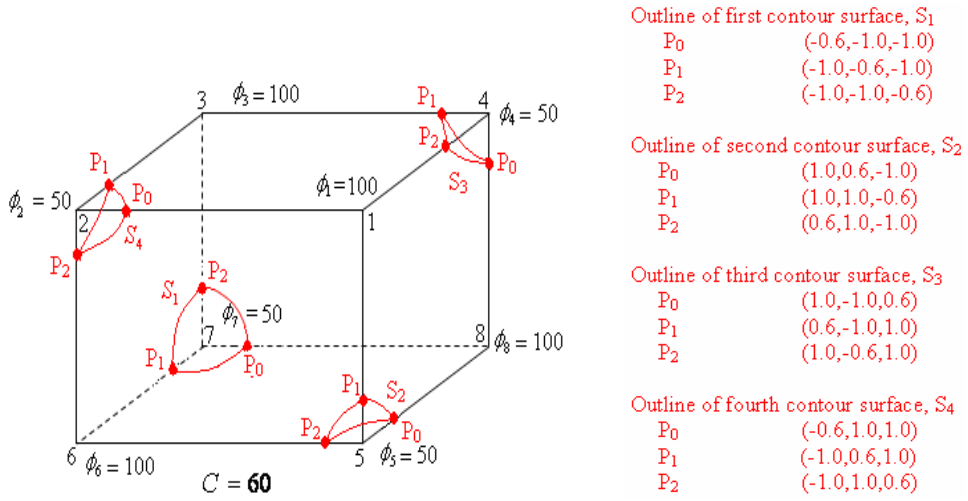


Figure 5.8(c) Four disjoint contour surfaces

The whole process of determining contour surfaces on a hexahedral element is summarized using the following steps.

(1) Split the hexahedral element into six faces.

(2) Determine the contour lines on each face. It is shown in the following section that the contour lines are, in general, branches of rectangular hyperbola on a plane.

(3) Join the contour lines determined on the six faces to form boundaries of disjoint contour surfaces.

(4) Develop the contour surfaces whose boundaries are determined in step (3).

Step (1) of the above procedure is a straightforward task. We develop detailed algorithms for the other three steps.

Step (2) Determining the Contour Lines on a Face

A face of a hexahedral element is a planner face. Therefore, without any loss of generality, we assume that the face is a 2D plane where only two parameters out of the three parameters r, s and t change and the third parameter becomes a constant $+1$ or -1. Let us assume that $t = t_0 = \pm 1$. Then the contour Eq. (5.19) assumes the form as shown in Eq. (5.23).

$$C = a_0 + a_1 r + a_2 s \pm a_3 t_0 + a_4 rs \pm a_5 st_0 \pm a_6 t_0 r \pm a_7 rst_0 \tag{5.23}$$

Eq. (5.23) is rewritten as the expression given in Eq. (5.24).

$$C = b_0 + b_1 r + b_2 s + b_3 rs \tag{5.24}$$

where,

$$b_0 = a_0 \pm a_3 t_0, \quad b_1 = a_1 \pm a_6 t_0, \quad b_2 = a_2 \pm a_5 t_0, \quad b_3 = a_4 \pm a_7 t_0 \tag{5.25}$$

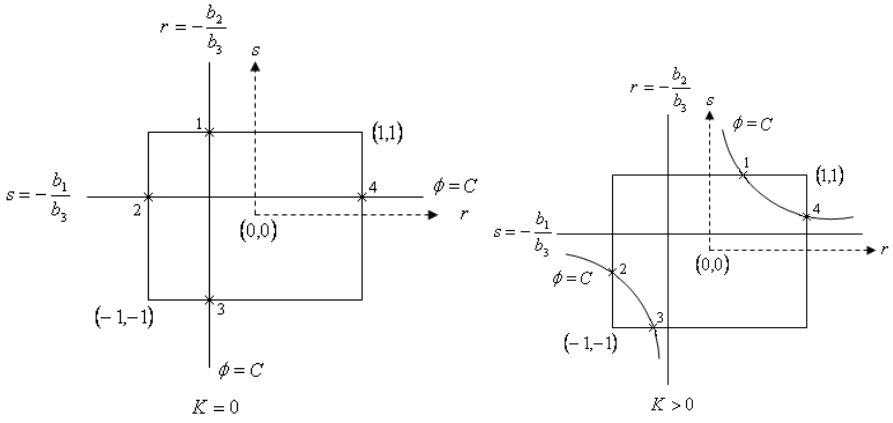
It is pointed out here that the contour Eq. (5.24) is the same which is obtained while generating contour lines using four-node quadrilateral elements in 2D, a fast method for which is derived by Singh and Sarkar (1990). Here, we mention the process briefly for the sake of completeness of the whole algorithm with some modifications to make the process faster. Eq. (5.24) is the equation of a rectangular hyperbola having the following standard form

$$\left(r + \frac{b_2}{b_3}\right)\left(s + \frac{b_1}{b_3}\right) = K, \qquad b_3 \neq 0 \tag{5.26}$$

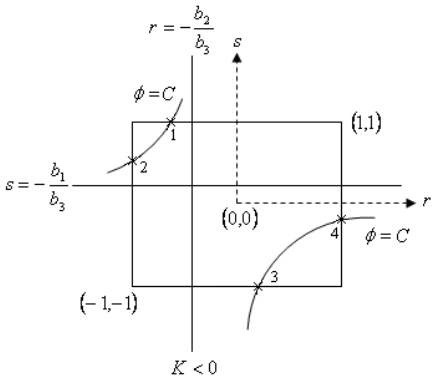
where,

$$K = \frac{1}{b_3}\left[C - b_0 + \frac{b_1 b_2}{b_3}\right] \tag{5.27}$$

For the case $b_3 = 0$, Eq. (5.24) represents a contour line, if $b_1^2 + b_2^2 > 0$ and if $b_1^2 + b_2^2 = 0$, then the face is a contour face. The derivation of contour line or face is a straightforward task, and hence we elaborate the case $b_3 \neq 0$.



(a) **(b)**



(c)

Figure 5.9 Contour segments as branches of a rectangular hyperbola on a quadrilateral element in local coordinates for various values of *K*

The asymptotes of the hyperbola are $r = -\frac{b_2}{b_3}$ and $s = -\frac{b_1}{b_3}$ which are parallel to *s*-axis and *r*-axis, respectively.

Contour branches are traced on a quadrilateral element in 2D for the following values of K.

(i) $K = 0$, the contour branches are the asymptotes $r = -\frac{b_2}{b_3}$ and $s = -\frac{b_1}{b_3}$. The branches lie within the element if $|r| \le 1$ or $|s| \le 1$.

(ii) $K > 0$, the two branches of the contour lie on the first and third quadrant of the quadrilateral.

(iii) $K < 0$, the two branches of the contour lie on the second and fourth quadrant of the quadrilateral.

The above three cases provide an efficient method for tracing contour segments in 2D using quadrilateral element. We find the intersection of the contour line $\phi = C$ with four edges of the quadrilateral and record the number of intersections with the edges. The total number of intersections can be 0, 2 or 4, thus yielding no branches, one branch and two branches, respectively. For two intersections a branch is obtained by considering the two intersections as the end points of the branch. When the number of intersections is 4, we form the two contour branches as follows.

(i) Find the intersections with the edges $s = 1$, $r = -1$, $s = -1$, $r = 1$ and number them as 1, 2, 3 and 4.

(ii) Evaluate K using Eq. (5.27). If $K = 0$, join 1 and 3, and 2 and 4. If $K > 0$, join 1 and 4, and 2 and 3. For $K < 0$, join 1 and 2, and 3 and 4. A graphical representation of the process is given in Fig. 5.9(a) – 5.9(c).

Efficient Tracing of Contour lines

After having determined the end points of contour lines, the efficient derivation of the intermediate points is an important task. Here an efficient approach is presented.

Let $A(r_1, s_1)$ and $B(r_2, s_2)$ be the two end points. It is mentioned here that *t* is taken to be a constant, $+1$ or -1, between A and B. The contour segment is traced by starting at one end point and reaching at the other end. A simplest way is to find r_i for a given s_i, starting with $s = s_1$ and incrementing s by $\Delta s = (s_2 - s_1)/n$, where n is the number of intervals between A and B. The equation of rectangular hyperbola, Eq. (5.24), is used which yields Eq. (5.28).

$$r_i = \frac{C - (b_0 + b_2 s_i)}{b_1 + b_3 s_i}, \quad s_i = s_1 + i\Delta s, \quad i = 0, 1, ..., n \tag{5.28}$$

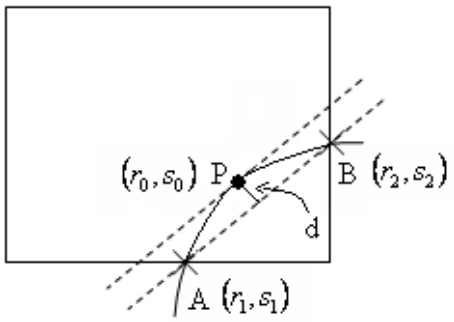


Figure 5.10 A contour segment and its maximum shift from the line *AB*

The efficiency of the algorithm depends on the number of intervals between A and B. If the value of n is fixed, then we might be computing all intermediate points which may not be required for an efficient algorithm. For example, if the contour segment turns out to be a straight line between A and B, then we need not evaluate Eq. (5.28) for all intermediate points. We can use the line plotting algorithm, such as Bresenham's algorithm used in computer graphics (Hearn and Baker, 2009). Thus the number of intervals, n, turns out to be a function of the maximum deviation of the contour segment

from the straight line joining A and B. The maximum deviation is obtained by locating a point P on the curve at which the slope of the curve is the same as the slope of the straight line AB, and then finding the distance of P from the line AB.

The slope of the contour curve, $s = f(r)$ (Eq. (5.24)) is given in Eq. (5.29).

$$\frac{ds}{dr} = -\frac{b_1 b_2 + b_3(C - b_0)}{(b_2 + b_3 r)^2} \tag{5.29}$$

The equation of the straight line AB is given in Eq. (5.30). The slope m is given by Eq. (5.31).

$$r(s_2 - s_1) + s(r_2 - r_1) + r_2 s_1 - r_1 s_2 = 0 \tag{5.30}$$

$$m = \frac{s_2 - s_1}{r_2 - r_1} \tag{5.31}$$

Let the slopes of the curve, $\frac{ds}{dr}$, and the straight line AB be the same, then

$$-\frac{b_1 b_2 + b_3(C - b_0)}{(b_2 + b_3 r)^2} = \frac{(s_2 - s_1)}{(r_2 - r_1)} \tag{5.32}$$

This yields the solution $r = r_0$, where

$$r_0 = \frac{1}{b_3}\left[-b_2 \pm \sqrt{\frac{(b_3(b_0 - C) - b_1 b_2)(r_2 - r_1)}{(s_2 - s_1)}} \right] \tag{5.33}$$

Substituting $r = r_0$, in Eq. (5.24), we get $s = s_0$, where

$$s_0 = \frac{C - (b_0 + b_1 r_0)}{(b_2 + b_3 r_0)} \tag{5.34}$$

The perpendicular distance of the point (r_0, s_0) from the straight line AB is

$$d = \frac{r_0(s_2 - s_1) + s_0(r_1 - r_2) + r_2 s_1 - r_1 s_2}{\sqrt{(s_1 - s_2)^2 + (r_1 - r_2)^2}} \tag{5.35}$$

The magnitude of the parameter d provides the maximum shift of the straight line from the contour segment. We thus split recursively the contour segment between the endpoints (r_1, s_1) and (r_2, s_2) into two contour segments $(r_1, s_1)$and (r_0, s_0), and (r_0, s_0)

and (r_2, s_2). If $|d| < d_0$, where d_0 is a predetermined value which can be set depending on the resolution of display screen, then we stop the process of subdividing the interval. The process can be summarized in the following

(i) Start with the interval (r_1, s_1) - (r_2, s_2).

(ii) Find the point of maximum shift (r_0, s_0).

(iii) Find d using Eq. (5.35).

(iv) If $|d| < d_0$, stop, else, divide (r_1, s_1) - (r_2, s_2) into two intervals (r_1, s_1) - (r_0, s_0) and (r_0, s_0) - (r_2, s_2) and repeat the process recursively.

Step (3) Joining contour segments: Determining boundaries of contour surfaces

The C^0 continuity of linear interpolation of discrete function values and tracing of contour segments on the faces of hexahedral element provide a mechanism to efficiently determine the boundaries of the contour surfaces in 3D. The boundaries are in the form of curved polylines. We demonstrate this by taking up an example given by Fig.5.7. The contour surface represented by the curved *polylines* P_0P_1, P_1P_2, P_2P_3 and P_3P_0 is generated after determining the contour segments on the faces $N_5N_6N_7N_8$, $P_1P_5P_8P_4$, $N_3N_7N_8N_4$, and $N_2N_3N_7N_6$, respectively. The remaining faces do not contain contour segments. Once the boundary of a contour surface is determined, it becomes computationally efficient to trace the contour surface because the domain of the search space is reduced considerably.

Algorithm: The contour segments are represented by two parameters *ns* and $PS[ns,i]$. The parameter *ns* represents segment number and the parameter $PS[ns,i]$ represents the two vertices of a segment *ns* for $i = 1,2$. The parameter $PS[ns,i]$ is a point in 3-D having three components $r[ns,i]$, $s[ns,i]$ and $t[ns,i]$. The algorithm for joining the contour segments to form contiguous boundaries of contour surfaces is given in the form of a pseudocode depicted in Fig. 5.11. The parameter *nsurf* represents the total number of surfaces which is initialized to *zero*. After the execution of the pseudocode, the contour surfaces are available in the form of $nv[nsurf]$ and $PV[i,nvi]$, where $nvi = nv[i]$ gives the number of vertices for the i^{th} surface whose local coordinates are saved in the

array $PV[i,nvi]$. The array $PV[i,nvi]$ represents the $r-$, $s-$, and $t-$coordinates of a vertex. We use the term *polylines* for the boundary of a contour surface to indicate that the boundary is approximated by line segments. The term contour segments are used to point to individual contour lines generated on each face of the hexahedral.

In the beginning of the algorithm the parameter for number of surfaces, *nsurf* , is set to *zero*, and it is assumed that all contour segments are unprocessed indicated by the *flags* , $f[i]$, which are set to *zero*. A contour segment is said to be processed if it forms a part of *polylines* of a contour surface in which case its *flag* is set to 1. The boundary of a contour surface is built up in the form of *polylines* by picking one unprocessed contour segment (with *flag* value *zero*) and then finding its adjoining matching contour segments. The matching is found by taking the euclidean distance between the two endpoints of the *polylines* and that of an unprocessed contour segment. The process is repeated until no unprocessed contour segment is found. This condition is indicated by another *flag* f_1 which is initialized to *zero* before starting the process of finding a match and set to 1 if a match is found. If still some unprocessed contour segments are left, then there could be other contour surfaces and the process must be repeated. As discussed earlier, there can be a maximum of four disjoint contour surfaces in a hexahedral element. At the end there must not be any unprocessed contour segment left out.

1. $nsurf = 0$, $f[i] = 0$, $i = 1, 2, ..., ns-1$
2. *for* $i = 0$ *to* $ns-1$

    $nvi \leftarrow 0$ // *no. of vertices is initialized to zero*

    if $f[i] = 1$ *then next* i // *if the segment already processed, then skip*

    $f[i] \leftarrow 1$

    $P_0 \leftarrow PS[i,0]$, $P_1 \leftarrow PS[i,1]$

    $PV[nsurf, nvi] \leftarrow P_0$ // *save first vertex*

    $nvi \leftarrow nvi + 1$ // *no. of vertex incremented by one*

    $PV[nsurf, nvi] \leftarrow P_1$ // *save second vertex*

    $nvi \leftarrow nvi + 1$ // *no. of vertex incremented by 1*

    for $k = 0$ *to* $ns-1$

        $f_1 \leftarrow 0$ // *initially assume no match found*

        for $j = 0$ *to* $ns-1$

            if $f[j] = 1$ *then next* j

```
        P1' ← PS[j, 0]                          // select the first vertex of contour segment for a match
        P2' ← PS[j, 1]
        if P1 = P0' then                        // P1 joins P0'
            f1 ← 1                              // match found
            PV[nsurf, nvi] ← P1', P1 ← P1'      // save vertex
            nvi ← nvi + 1                       // increment no. of vertices by 1
        else if P1 = P1' then                   // P1 joins P1'
            f1 ← 1                              // match found
            PV[nsurf, nvi] ← P0', P1 ← P0'      // save vertex
            nvi ← nvi + 1                       // increment no. of vertices by 1
        else if P0 = P0' then                   // P0 joins P0'
            f1 ← 1                              // match found
            for l = nvi to 1 step −1            // shift vertices to make room for the new vertex to be added
                PV[nsurf, l] ← PV[nsurf, l − 1]
            endfor
            PV[nsurf, 0] ← P1', P0 ← P1'        // save vertex
            nvi ← nvi + 1                       // increment no. of vertices by 1
        else if P0 = P1' then
            f1 ← 1
            for l = nvi to 1 step −1            // shift vertices to make room for the new vertex to be added
                PV[nsurf, l] ← PV[nsurf, l − 1]
            endfor
            PV[nsurf, 0] ← P0', P0 ← P0'        // save vertex
            nvi ← nvi + 1                       // increment no. of vertices by 1
        endif
    endfor                                      // end of j-loop
    if f1 = 0 then break                        // no connected segment found, jump out of k-loop
endfor                                          // end of k-loop
nv[nsurf] ← nvi                                 // save no. of vertices for the surface- nsurf
nsurf ← nsurf + 1                               // increment no. of surfaces by one.
endfor                                          // end of i-loop
```

Figure 5.11 Pseudocode for finding the closed polyline of a contour surface

Step (4) Developing contour surfaces

The boundary of a contour surface is available in the form of *polylines*. Let the vertices of the *polylines* be given by $P_0, P_1, \ldots, P_n$. The *polylines* comprise of the lines $P_0P_1, P_1P_2, \ldots, P_nP_0$. In general a line P_iP_{i+1} is a curved line whose points can be generated by using the equation of a rectangular hyperbola, Eq. (5.24), in which it is assumed that one of the three parameters is a constant, $+1$ or -1. The surface is developed by using the surface equation $u = \psi(v, w)$ which can be obtained from Eq. (5.19). The parameter w is one of the three parameters r, s, and t whose span is maximum. The span of parameters is determined by computing $r_{\min} = \min_i \{r_i\}$, $r_{\max} = \max_i \{r_i\}$, $s_{\min} = \min_i \{s_i\}$, $s_{\max} = \max_i \{s_i\}$, $t_{\min} = \min_i \{t_i\}$, $t_{\max} = \max_i \{t_i\}$, $r_s = r_{\max} - r_{\min}$, $s_s = s_{\max} - s_{\min}$, $t_s = t_{\max} - t_{\min}$.

Thus, if t_s is maximum of r_s, s_s and t_s, then $w = t$. We choose the parameter with the maximum span to plot the contour surface in the form of a surface mesh because it provides better appearance of the display of the contour surface. The contour surface is displayed in the form of pixels of points on the surface by tracing the contour points on the line $w = w_i$, where $w_{\min} \le w_i \le w_{\max}$. After fixing $w = w_i$, we next consider that parameter out of the remaining two parameters (say, r and s) whose span is higher (say, s) and represent this parameter by v. The parameter v is varied between $v_{\min}$ and $v_{\max}$ which lie on the left and right edges of contour surface. The third parameter, i.e. u is computed using the function $u = \psi(v, w)$. As shown in Fig. 5.12, let $w = w_i$ intersect the left and right edges, AE and BC, respectively, at P and Q, then the value of v at P and Q represented by v_1 and v_2 are computed. Let there be $m+1$ intersection points on this line with $v_j = v_1 + (v_2 - v_1) j/m$, $j = 0, 1, 2, \ldots, m$. The third parameter is computed using the contour surface equation, i.e. $u_{ij} = \psi(v_j, w_i)$. Since one of the three parameters (w_i) is constant on this line, we make the process of determining the points on this line efficient as already outlined in step (2) earlier. The contour point (u_{ij}, v_j, w_i) which is in local coordinates is converted into global coordinates using the transformations given in Eq. (5.15). It is plotted on the screen after making suitable projection transformations from global coordinates to screen coordinates.

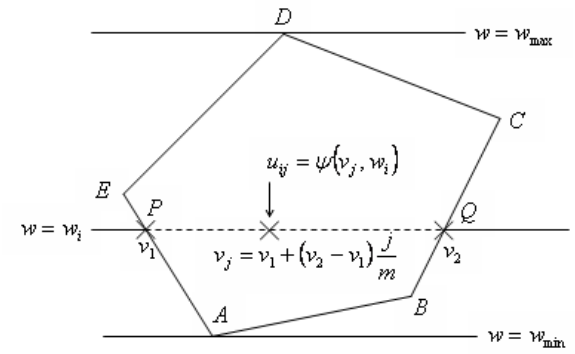


Figure 5.12 Generation of interior points on contour surface

5.4 PLOTTING CONTOURS IN 3D

A contour line in 3D is defined by a set of vertices $P_i(x_i, y_i, z_i),\ 0 \le i < n$ such that the line P_iP_{i+1} connects the vertices P_i and P_{i+1}, thus forming a *polyline* having $n-1$ straight line segments. After the contour lines are derived in the world coordinate system, it is desired that the lines are plotted on the display devices such as a computer screen. This requires projection of a point from 3D to 2D plane. There are two types of projections:

(i) Parallel Projection

(ii) Perspective Projection

Normally, parallel projection is used in engineering drawings. A specific case of parallel projection-parallel orthographic projection-is used which maintains the scale and shape of an object when a 3D object is projected onto a 2D plane. The direction of projection is the prescribed direction for all projectors. Orthographic projections are characterized by the fact that the direction of projection is perpendicular to the viewplane. We define the orthographic parallel projection by defining the following parameters:

(i) View Plane: The view plane is the plane on which the projection is taken. A view plane is determined by its view reference point $R_0(x_0, y_0, z_0)$ and view plane normal $\vec{N}$.

(ii) Reference Point: The reference point determines the location of the viewplane. It is determined by specifying its coordinates in the user coordinate system and is denoted by $R_0(x_0, y_0, z_0)$.

(iii) View Plan Normal Vector: The viewplane normal vector $\vec{N}$ is defined by joining the eye position (e_x, e_y, e_z) and the view reference point $R_0(x_0, y_0, z_0)$. The direction of the vector $\vec{N}$ is given by the direction ratios $(e_x - x_0,\ e_y - y_0,\ e_z - z_0)$.

Fig. 5.13(a) depicts the viewplane specification in the user coordinate system. The vector $\vec{N}$ is given by

$$\vec{N} = (e_x - x_0)\,\vec{I} + (e_y - y_0)\,\vec{J} + (e_z - z_0)\,\vec{K} \tag{5.36}$$

where $\vec{I}$, $\vec{J}$ and $\vec{K}$ are unit vectors along coordinate axes and $R_0(x_0, y_0, z_0)$ and $E(e_x,\ e_y,\ e_z)$ are the reference point and eye position.

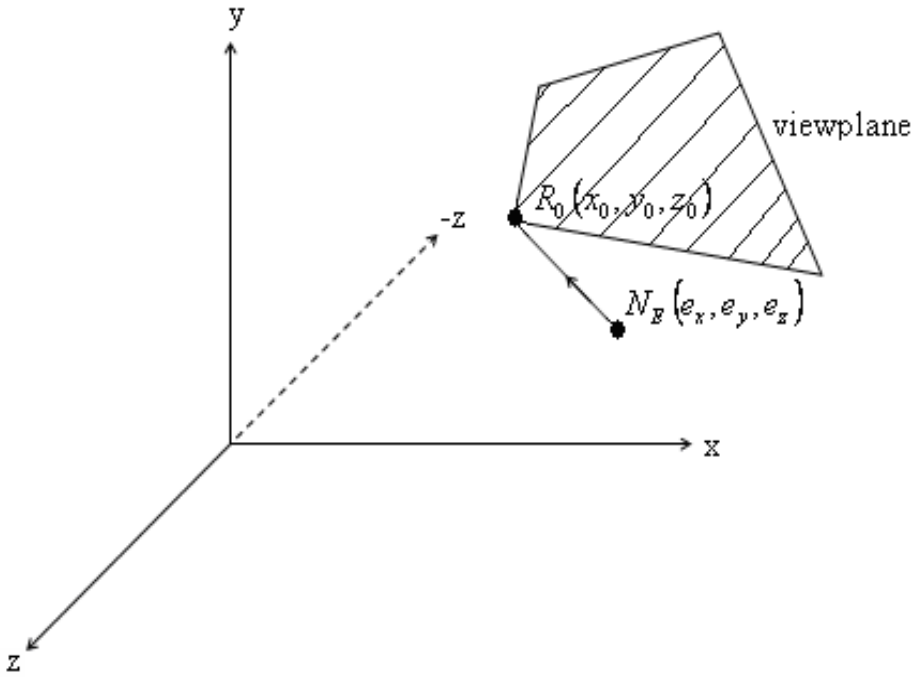


Figure 5.13(a) Specifying viewplane in user coordinate system

5.4.1 Deriving Matrix Equation for Orthographic Parallel Projection

The objective is to find a composite matrix that determines the general transformation from world coordinates to view plane coordinates system. This comprises of two steps:

(i) Determining view plane coordinate system

(ii) Finding a transformation that aligns world coordinates to the view plane coordinates

Determining View Plane Coordinate System: We define view plane coordinate system by specifying the view plane reference point $R_0(x_0, y_0, z_0)$ and eye position $E(e_x, e_y, e_z)$. The normal vector to the view plane is determined as given by Eq. (5.36).

We determine the three unit vectors $\vec{I}_v, \vec{J}_v$ and $\vec{K}_v$ for the view plane coordinates system. These unit vectors are orthogonal to each i.e. $\vec{I}_v \times \vec{J}_v = \vec{J}_v \times \vec{K}_v = \vec{K}_v \times \vec{I}_v = \vec{0}$.

Let $\vec{K}_v = \frac{\vec{N}}{|\vec{N}|}$, which is the unit vector normal to the view plane. The two other vectors $\vec{I}_v$ and $\vec{J}_v$ will lie on the plane and will be perpendicular to each other as shown in Fig. 5.13(b).

The unit vector $\vec{J}_v$ is given by Eq. (5.37).

$$\vec{J}_v = \frac{\vec{U}_v}{|\vec{U}_v|} \tag{5.37}$$

where,

$$\vec{U}_v = \vec{U} - \left(\vec{N}.\vec{U}\right)\vec{N} \tag{5.38}$$

Here $\vec{U}$ is the up-vector which specifies the direction of the vector $\vec{J}_v$ in the view plane. The concept of up-vector is used to align the image along a particular direction. We normally, take $\vec{U}$ as given by Eq. (5.39).

$$\vec{U} = 0.\vec{I} + 1.\vec{J} + 0.\vec{K} \tag{5.39}$$

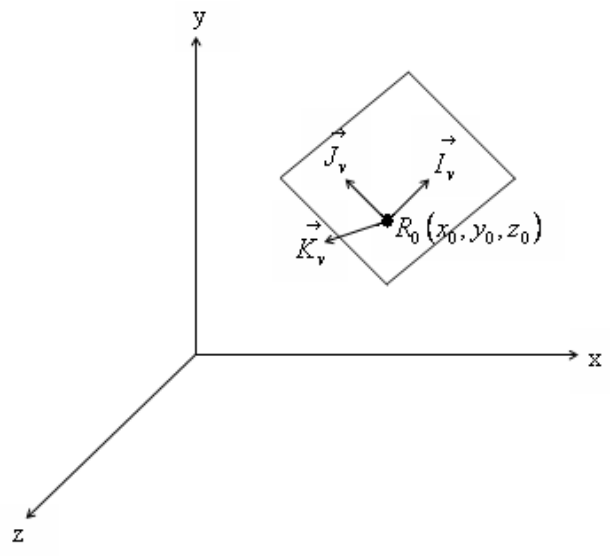


Figure 5.13(b) The View plane coordinate system indicating orthogonal unit vectors $\vec{I}_v$, $\vec{J}_v$ and $\vec{K}_v$

This indicates that the *y-axis* of the user coordinate system will be aligned with the $\vec{J}_v$ axis of the view plane coordinate system. We must make sure that the vectors $\vec{N}$ and $\vec{U}$ are not parallel to each other, otherwise $\vec{U}_v$ will turn out to be undefined. The vector $\vec{U}_v$ is the projection of the vector $\vec{U}$ on the view plane.

Having defined $\vec{K}_v$ and $\vec{J}_v$, we can find $\vec{I}_v$ as given by Eq. (5.40).

$$\vec{I}_v = \frac{\vec{J}_v \times \vec{K}_v}{|\vec{J}_v \times \vec{K}_v|} \tag{5.40}$$

This completes the derivation of the unit vectors $\vec{I}_v$, $\vec{J}_v$ and $\vec{K}_v$ for the view plane coordinate system.

Transformation that Aligns World Coordinates to the View Plane Coordinates

The following are defined:

(i) The view reference point $R_0(x_0, y_0, z_0)$.

(ii) The eye position $E(e_x, e_y, e_z)$.

(iii) The up vector, $\vec{U}(u_x, u_y, u_z)$. We can take $\vec{U} = (0,1,0)$, i.e. the unit vector $\vec{J}_v$ of the view plane coordinate system is upward. If $\vec{U}$ and $\vec{N}$ are parallel to each other, i.e. $\vec{U}.\vec{N} = |\vec{N}||\vec{U}|$, then $\vec{U} = (0,0,1)$, i.e. the unit vector $\vec{J}_q$ is along the direction of $\vec{K}$.

(iv) Derive $\vec{I}_v$, $\vec{J}_v$ and $\vec{K}_v$ as explained in the previous section.

(v) Translate the view plane reference point $R_0(x_0, y_0, z_0)$ to the origin. The homogenous matrix for translation is

$$T_1 = \begin{bmatrix} 1 & 0 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.41}$$

(vi) Align the view plane normal vector $\vec{N}$ with $\vec{K}$ using the transformation matrix A_n, where

$$A_n = \begin{bmatrix} \frac{m}{n} & \frac{-ab}{mn} & \frac{-ac}{mn} & 0 \\ 0 & \frac{c}{m} & \frac{-b}{m} & 0 \\ \frac{a}{n} & \frac{b}{n} & \frac{c}{n} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.42}$$

where

$$a = e_x - x_0,\ b = e_y - y_0,\ c = e_z - z_0,\quad m = \sqrt{b^2 + c^2},\quad n = \sqrt{a^2 + m^2} \tag{5.43}$$

If $m = 0$, then

$$A_n = \begin{bmatrix} 0 & 0 & \frac{-a}{|a|} & 0 \\ 0 & 1 & 0 & 1 \\ \frac{a}{|a|} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.44}$$

(vii) We perform the transformation

$$I'_v = A_N . I_v \tag{5.45}$$

(viii) Rotate I'_p about *z-axis* so that it aligns with I, i.e. along positive *x-axis*. Let θ be the angle of rotation, then we rotate I'_p by angle $(-\theta)$. This is achieved by using the coordinates of I'_v derived in Eq.(5.45). Let $I'_v = (u, v, w)$ then

$$\cos\theta = \frac{u}{\sqrt{u^2 + v^2}}, \qquad \sin\theta = \frac{v}{\sqrt{u^2 + v}} \tag{5.46}$$

The rotation matrix is

$$R_\theta = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.47}$$

(ix) The composite matrix is

$$T = R_\theta . A_N . T_1 \tag{5.48}$$

(x) Transform all points of the objects in the world coordinates to viewplane coordinates by using

$$P_v = T.P \tag{5.49}$$

where $P = (x, y, z, 1)$ and $P_v = (x_v, y_v, w_v, 1)$ in the homogenous coordinate system.

5.5 IMPLEMENTATION OF THE ALGORITHMS FOR TETRAHEDRAL ELEMENT

The contour plot algorithm, discussed in section 5.2, is implemented in Turbo C++ and runs on an IBM compatible PC with 3.0 GHz CPU and 256 RAM under Windows operating system. The OpenGL graphics libraries are used to display the geometry and contour lines for different components.

The problems considered here are the deflection and thermal problems of the different components.

The first problem is a deflection of a 3D beam with specification shown in Fig. 5.14(a). The beam is fixed at one end and an axial load is applied at the other end, shown in Fig. 5.14(b). For a load of 48 N, Young's Modulus $200 GPa$, Poisson's ratio 0.3, the values of horizontal deflections are derived using FEA taking four-node tetrahedral elements.

Since the beam is axially loaded, the general expression for direct horizontal deflection is given by Eq. (3.25).

Fig. 5.14(c) shows the contour surfaces for horizontal deflections using four-node tetrahedrals for different levels of deflections.

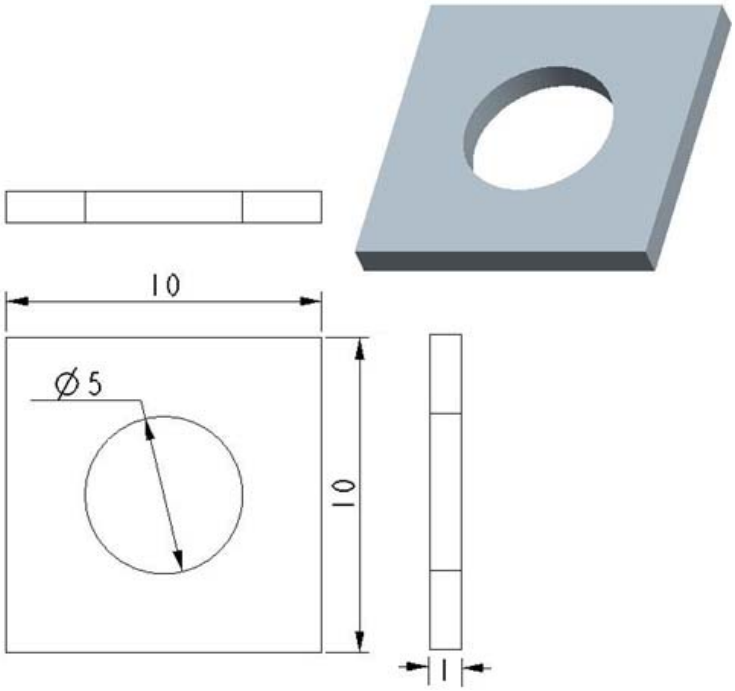


Figure 5.14(a) Specifications of a beam (all dimensions in mm)

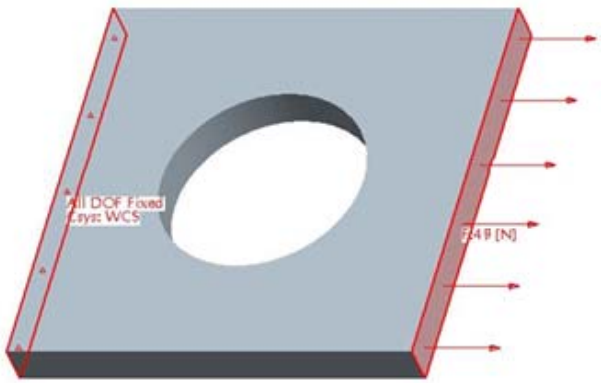
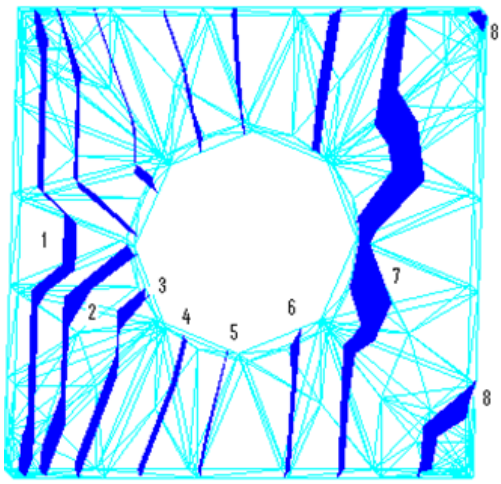


Figure 5.14(b) Boundary conditions and loading of the plate



Contour Levels	C = δ_x
1	2 x10^{-3}
2	5 x10^{-3}
3	10 x10^{-3}
4	20 x10^{-3}
5	30 x10^{-3}
6	45 x10^{-3}
7	55 x10^{-3}
8	70 x10^{-3}

Figure 5.14(c) Contour surfaces of horizontal deflection using 4-node tetrahedrals for the different deflection levels for the plate

Fig. 5.15(a) shows the specifications of a crane hook. The hook is fixed at the top and loaded vertically, as shown in Fig. 5.15(b). For a load of 1000 N, Young's Modulus $0.2\times10^{9}\ MPa$, Poisson's ratio 0.3, the values of deflections are derived using FEA, considering it as the 3D problem.

As the hook is vertically loaded, the general expression for resultant stress, which can be used to calculate the displacements (Singh, 2008) is given by Eq. (5.50) and (5.51).

$$\sigma_1 = -\frac{W}{A} + \frac{W}{A}\left[1 - \frac{R^2}{h^2}\left(\frac{d_1}{R - d_1}\right)\right] \tag{5.50}$$

$$\sigma_2 = -\frac{W}{A} + \frac{W}{A}\left[1 + \frac{R^2}{h^2}\left(\frac{d_2}{R + d_2}\right)\right] \tag{5.51}$$

where W is the load applied, A is the area of the considered cross section, R is the radius of curvature of the centroidal axis, h is a constant depending upon the cross-section of the hook, d_1 and d_2 are the inside and outside diameter of the hook, σ_1 and σ_2 are the stresses at the inside and outside layer of the hook.

Fig. 5.16(a) and 5.16(b) shows the contour plots of horizontal and vertical deflections using four-node tetrahedral elements for different levels of deflections.

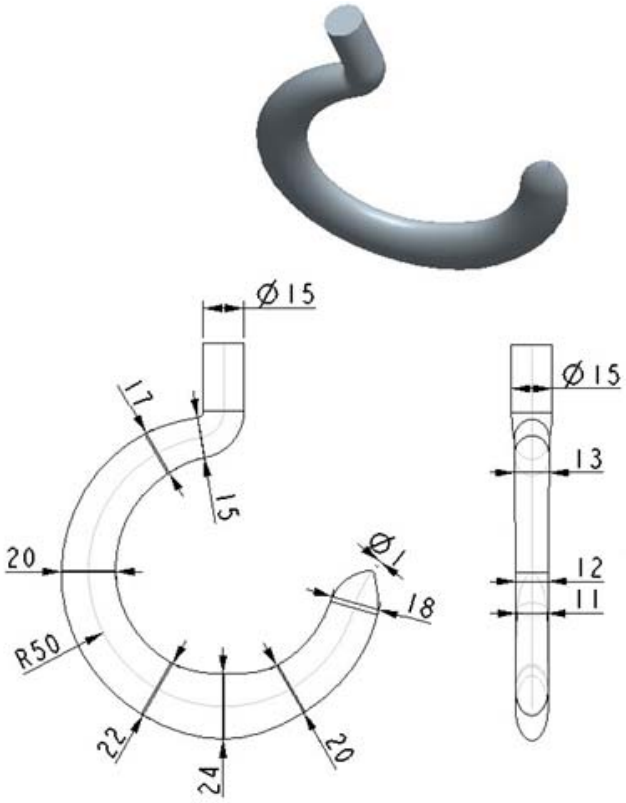


Figure 5.15(a) Specifications of a crane hook

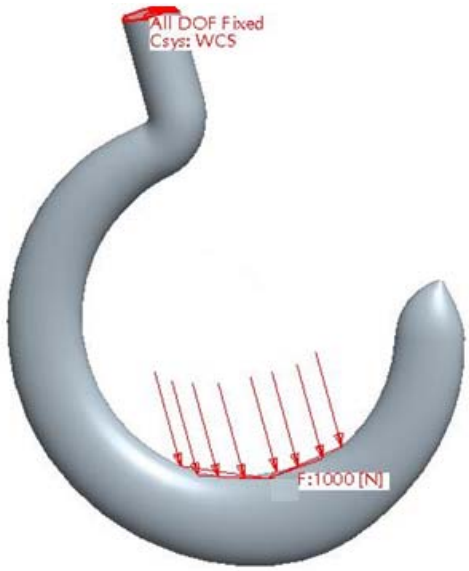
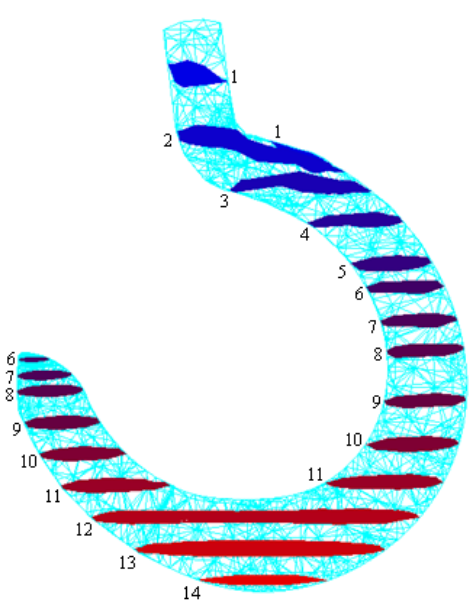
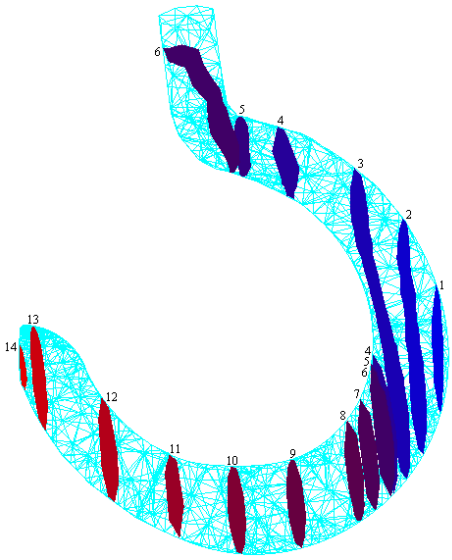


Figure 5.15(b) Boundary conditions and loading of the crane hook



Contour Levels	C = δ_x
1	-0.05 x10^{-4}
2	-0.3 x10^{-4}
3	-1 x10^{-4}
4	-3 x10^{-4}
5	-7 x10^{-4}
6	-10 x10^{-4}
7	-15 x10^{-4}
8	-20 x10^{-4}
9	-30 x10^{-4}
10	-40 x10^{-4}
11	-50 x10^{-4}
12	-60 x10^{-4}
13	-70 x10^{-4}
14	-80 x10^{-4}

Figure 5.16(a) Contour surfaces of deflection in x-direction using four-node tetrahedrals for the different deflection levels



Contour Levels	C = δ_y
1	15 x10^{-4}
2	10 x10^{-4}
3	5 x10^{-4}
4	1 x10^{-4}
5	0.1 x10^{-4}
6	-0.1 x10^{-4}
7	-0.5 x10^{-4}
8	10 x10^{-4}
9	-30 x10^{-4}
10	-50 x10^{-4}
11	-70 x10^{-4}
12	-90 x10^{-4}
13	-110 x10^{-4}
14	-115 x10^{-4}

Figure 5.16(b) Contour surfaces of deflection in y-direction using four-node tetrahedrals for the different deflection levels

Fig. 5.17(a) shows the specifications of a bearing bracket. The bracket is fixed at the base and loaded vertically, as shown in Fig. 5.17(b). For a load of 1000 N, Young's Modulus 0.2×10^{9} *MPa*, Poisson's ratio 0.3, the values of deflections are derived using FEA,

considering it as the 3D problem. For the given loading conditions, the deflections are due to the bending moment which is given by Eqs. (3.26) to (3.28).

Fig. 5.18(a), 5.18(b) and 5.18(c) shows the contour plots of deflections in *x*, *y* and *z*-directions using four-node tetrahedral elements for different levels of deflections.

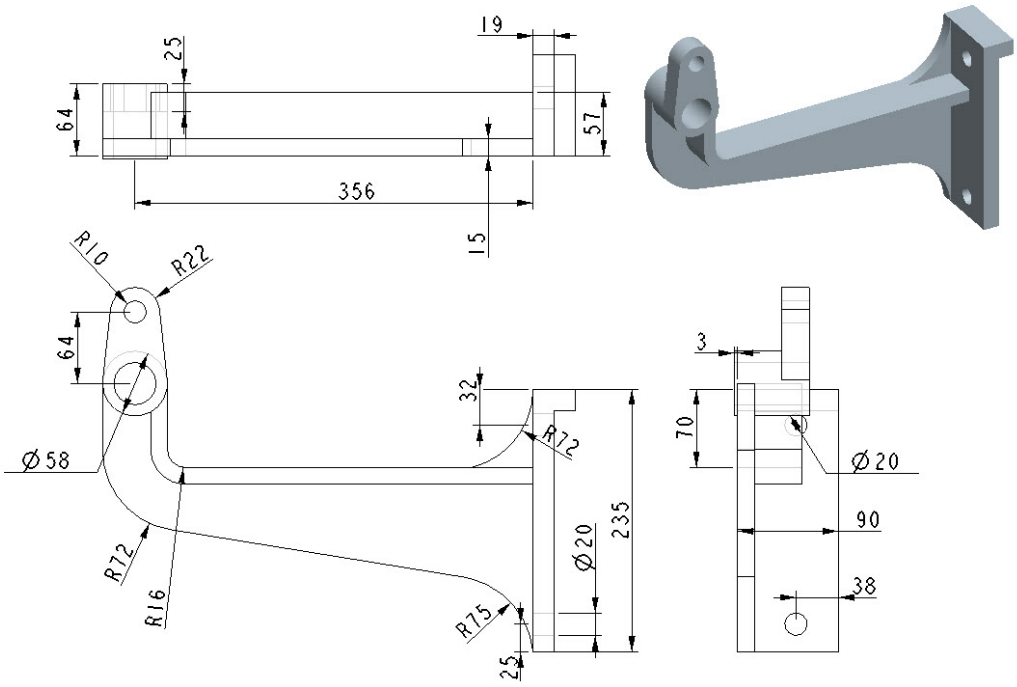


Figure 5.17(a) Specifications of the bearing bracket

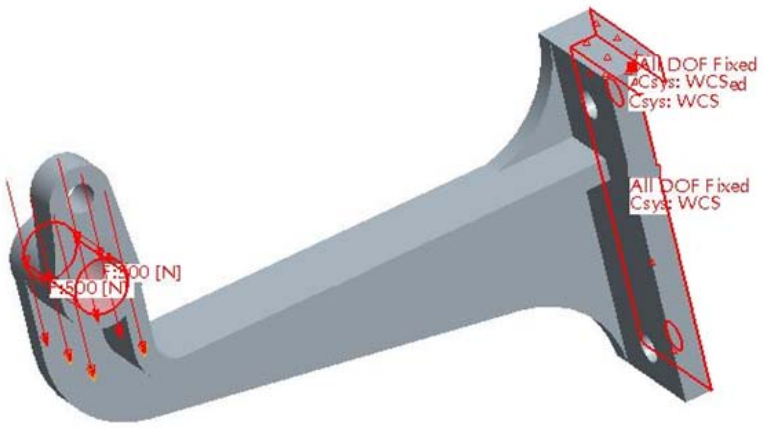
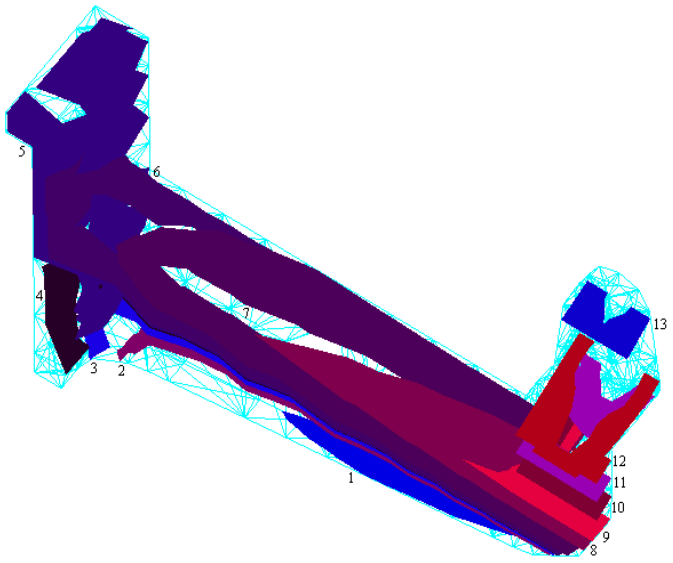
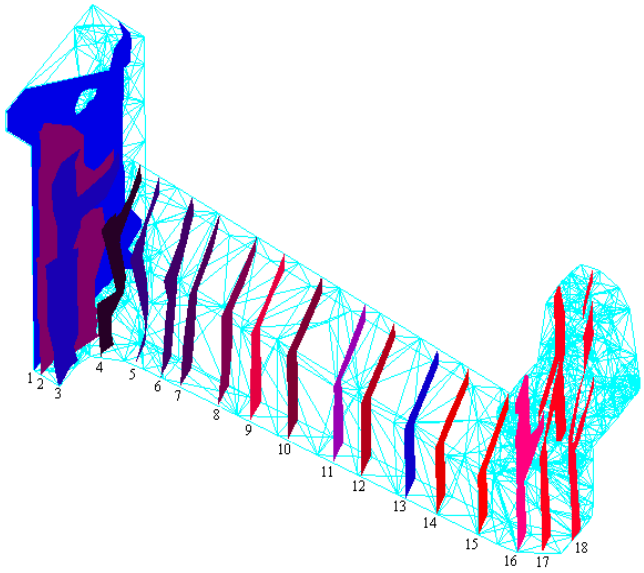


Figure 5.17(b) Boundary conditions and loading of the bearing bracket



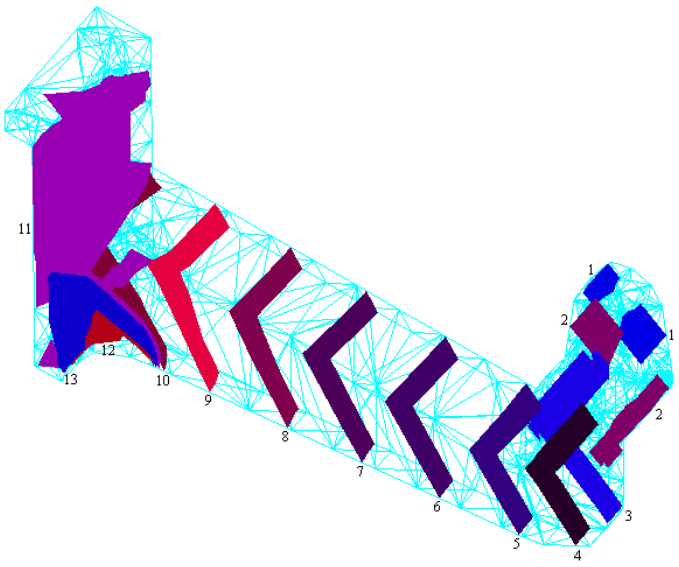
Contour Levels	C = δ_x
1	4 x10^{-2}
2	1 x10^{-2}
3	0.5 x10^{-2}
4	0.1 x10^{-2}
5	-0.0005 x10^{-2}
6	-0.1 x10^{-2}
7	-1 x10^{-2}
8	-4 x10^{-2}
9	-7 x10^{-2}
10	-12 x10^{-2}
11	-16 x10^{-2}
12	-20 x10^{-2}
13	-33 x10^{-2}

Figure 5.18(a) Contour surfaces of deflection in x-direction using four-node tetrahedrals for the different deflection levels for bearing bracket



Contour Levels	C = δ_y
1	0.0005 x10^{-2}
2	0.03 x10^{-2}
3	0.1 x10^{-2}
4	0.5 x10^{-2}
5	1 x10^{-2}
6	2 x10^{-2}
7	3 x10^{-2}
8	5 x10^{-2}
9	7 x10^{-2}
10	10 x10^{-2}
11	14 x10^{-2}
12	17 x10^{-2}
13	22 x10^{-2}
14	26 x10^{-2}
15	32 x10^{-2}
16	38 x10^{-2}
17	42 x10^{-2}
18	47 x10^{-2}

Figure 5.18(b) Contour surfaces of deflection in y-direction using four-node tetrahedrals for the different deflection levels for bearing bracket



Contour Levels	C = δ_z
1	35 x10^{-2}
2	30 x10^{-2}
3	25 x10^{-2}
4	20 x10^{-2}
5	15 x10^{-2}
6	10 x10^{-2}
7	6 x10^{-2}
8	3 x10^{-2}
9	1 x10^{-2}
10	0.1 x10^{-2}
11	0.0006 x10^{-2}
12	-0.1 x10^{-2}
13	-0.05 x10^{-2}

Figure 5.18(c) Contour surfaces of deflection in z-direction using four-node tetrahedrals for the different deflection levels for bearing bracket

Fig. 5.19(a) shows the specifications of a fork. The fork is fixed at the axial hole and loaded horizontally, as shown in Fig. 5.19(b). For a load of 1000 N, Young's Modulus $0.2\times10^{9}\ MPa$, Poisson's ratio 0.3, the values of deflections are derived using FEA, considering it as the 3D problem. For the given loading conditions, the deflections are due to the combined effect of direct force and bending moment which are given by Eqs. (3.25) to (3.28).

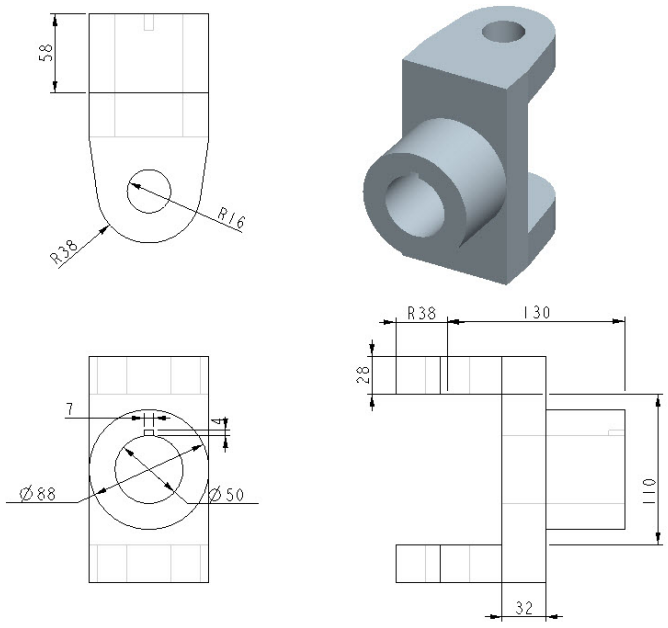


Figure 5.19(a) Specifications of the fork

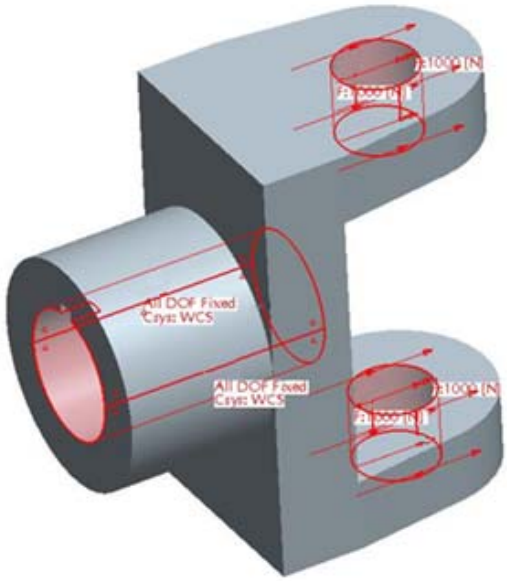
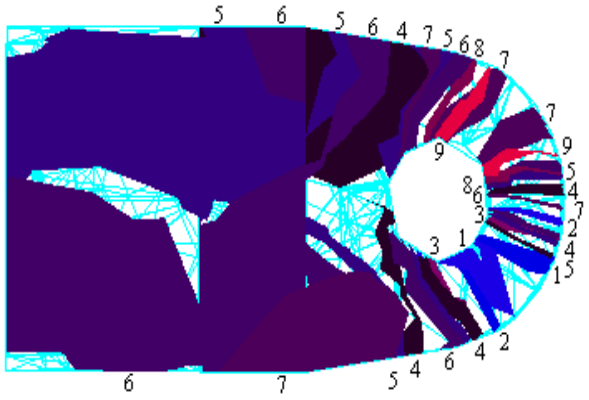


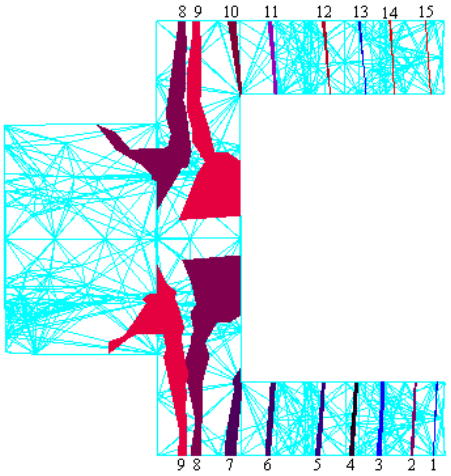
Figure 5.19(b) Boundary conditions and loading of the fork

Fig. 5.20(a), 5.20(b) and 5.20(c) shows the contour plots of deflections in *x*, *y* and *z*-directions using four-node tetrahedral elements for different levels of deflections.



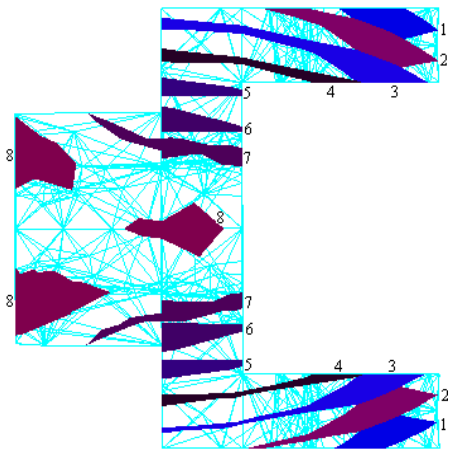
Contour Levels	C = δ_x
1	-2.8 x10^{-4}
2	-2.0 x10^{-4}
3	-1.5 x10^{-4}
4	-0.5 x10^{-4}
5	-0.1 x10^{-4}
6	0.1 x10^{-4}
7	1.0 x10^{-4}
8	2.0 x10^{-4}
9	2.5 x10^{-4}

Figure 5.20(a) Contour surfaces of deflection in x-direction using four-node tetrahedrals for the different deflection levels



Contour Levels	C = δ_y
1	-42 x10^{-4}
2	-38 x10^{-4}
3	-32 x10^{-4}
4	-27 x10^{-4}
5	-21 x10^{-4}
6	-12 x10^{-4}
7	-6 x10^{-4}
8	-1 x10^{-4}
9	1 x10^{-4}
10	6 x10^{-4}
11	12 x10^{-4}
12	21 x10^{-4}
13	27 x10^{-4}
14	32 x10^{-4}
15	38 x10^{-4}

Figure 5.20(b) Contour surfaces of deflection in y-direction using four-node tetrahedrals for the different deflection levels



Contour Levels	C = δ_z
1	-30 x10^{-4}
2	-25 x10^{-4}
3	-20 x10^{-4}
4	-15 x10^{-4}
5	-10 x10^{-4}
6	-5 x10^{-4}
7	-2 x10^{-4}
8	0.05 x10^{-4}

Figure 5.20(c) Contour surfaces of deflection in z-direction using four-node tetrahedrals for the different deflection levels

Fig. 5.21(a) shows the specifications of a fork. The fork is fixed at the two vertical holes and loaded horizontally through the other two holes, as shown in Fig. 5.21(b). For a load of 1000 N, Young's Modulus $0.2\times10^{9}\,MPa$, Poisson's ratio 0.3, the values of deflections are derived using FEA, considering it as the 3D problem. For the given loading conditions, the deflections are due to the combined effect of direct force and bending moment which are given by Eqs. (3.25) to (3.28).

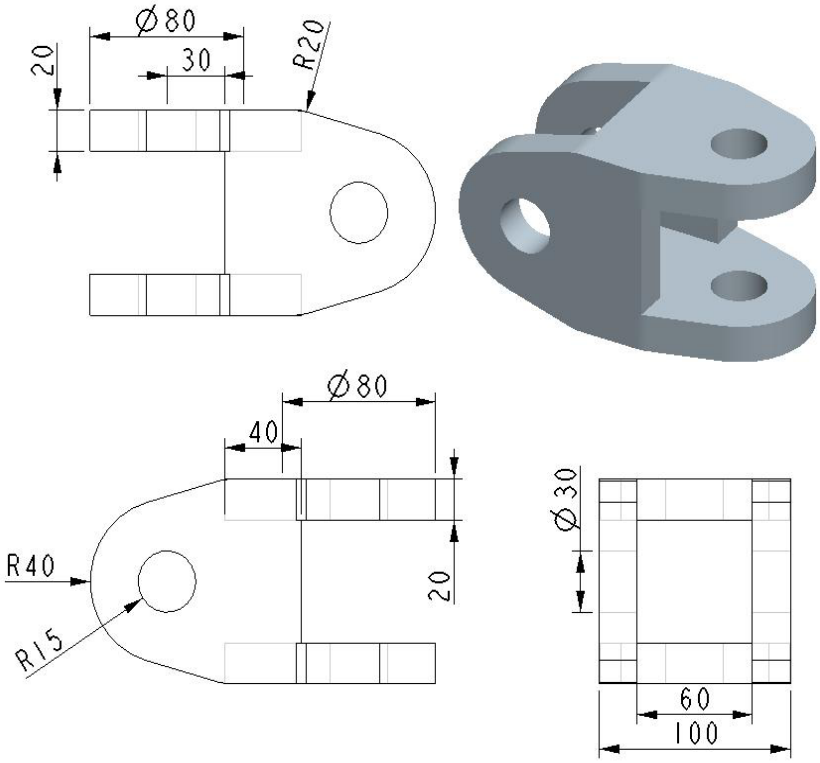


Figure 5.21(a) Specifications of the fork

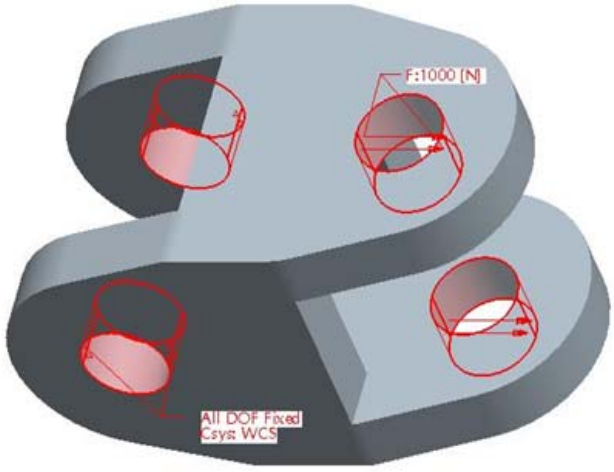
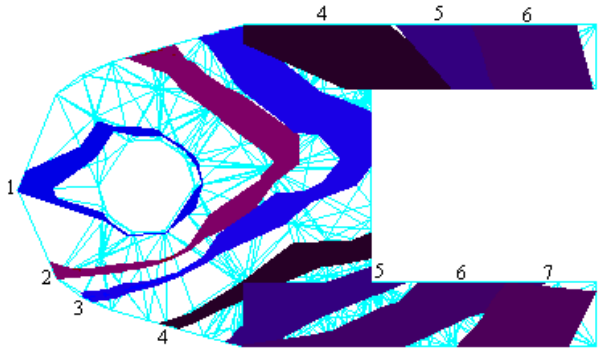


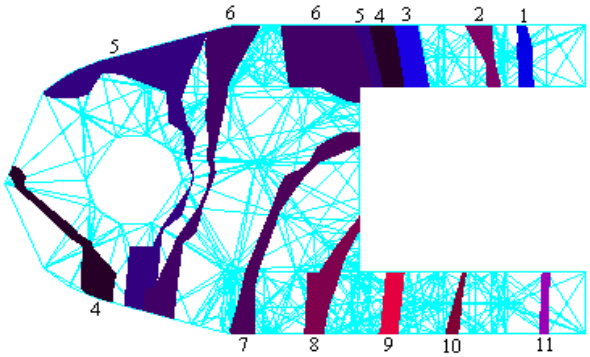
Figure 5.21(b) Boundary conditions and loading of the fork

Fig. 5.22(a), 5.22(b) and 5.22(c) show the contour plots of deflections in *x*, *y* and *z*-directions using four-node tetrahedral elements for different levels of deflections.



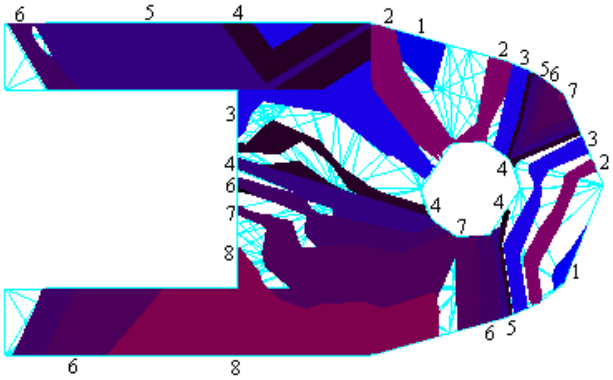
Contour Levels	C = δ_x
1	0.05 x10^{-6}
2	0.6 x10^{-6}
3	1 x10^{-6}
4	2 x10^{-6}
5	3 x10^{-6}
6	4.5 x10^{-6}
7	7 x10^{-6}

Figure 5.22(a) Contour surfaces of deflection in x-direction using four-node tetrahedrals for the different deflection levels for the fork



Contour Levels	C = δ_y
1	-2 x10^{-6}
2	-1.5 x10^{-6}
3	-0.5 x10^{-6}
4	-0.2 x10^{-6}
5	0.07 x10^{-6}
6	0.2 x10^{-6}
7	0.8 x10^{-6}
8	1.8 x10^{-6}
9	3.5 x10^{-6}
10	5.2 x10^{-6}
11	7.5 x10^{-6}

Figure 5.22(b) Contour surfaces of deflection in y-direction using four-node tetrahedrals for the different deflection levels for the fork



Contour Levels	C = δ_z
1	8 x10^{-7}
2	5 x10^{-7}
3	3 x10^{-7}
4	1 x10^{-7}
5	0.5 x10^{-7}
6	-0.1 x10^{-7}
7	-1 x10^{-7}
8	-3 x10^{-7}

Figure 5.22(c) Contour surfaces of deflection in z-direction using four-node tetrahedrals for the different deflection levels for the fork

Fig. 5.23(a) shows the specifications of a bracket. The bracket is fixed at the base and loaded vertically, as shown in Fig. 5.23(b). For a load of 1000 N, Young's Modulus $0.2\times10^{9}\ MPa$, Poisson's ratio 0.3, the values of deflections are derived using FEA,

considering it as a 3D problem. For the given loading conditions, the deflections are due to the bending moment which is given by Eqs. (3.26) to (3.28).

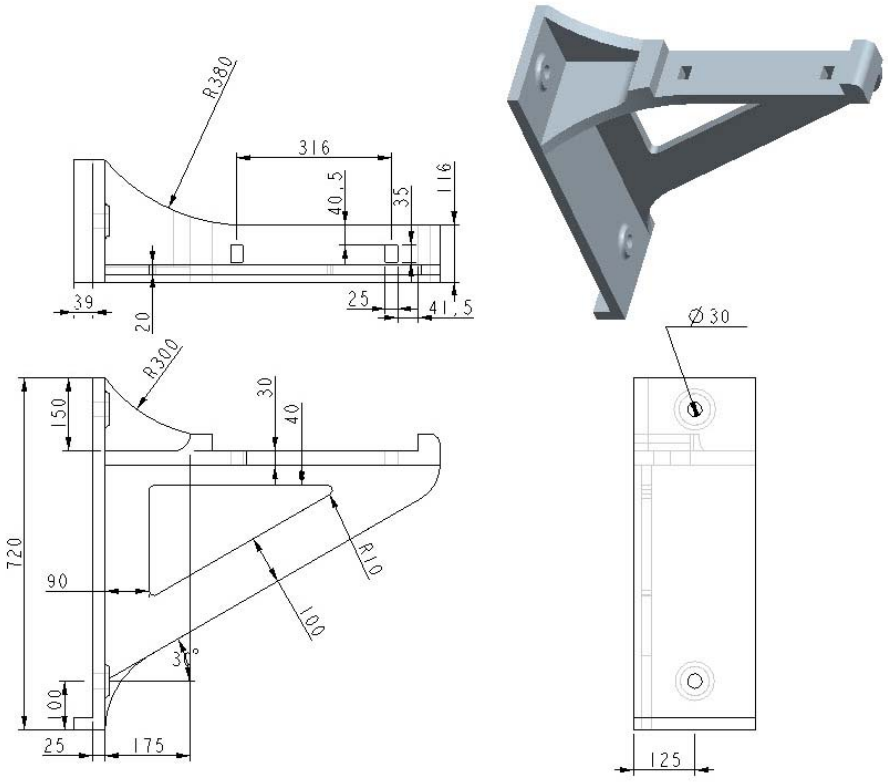


Figure 5.23(a) Specifications of the bracket

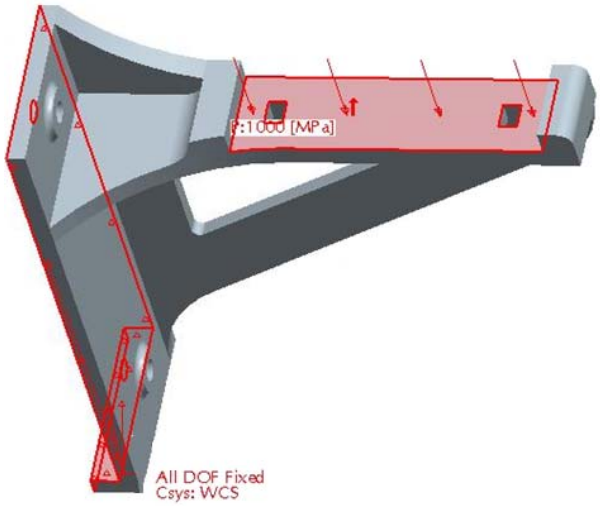
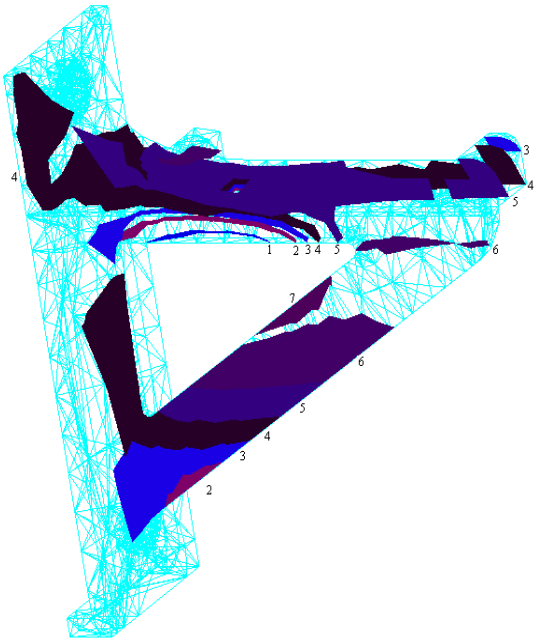


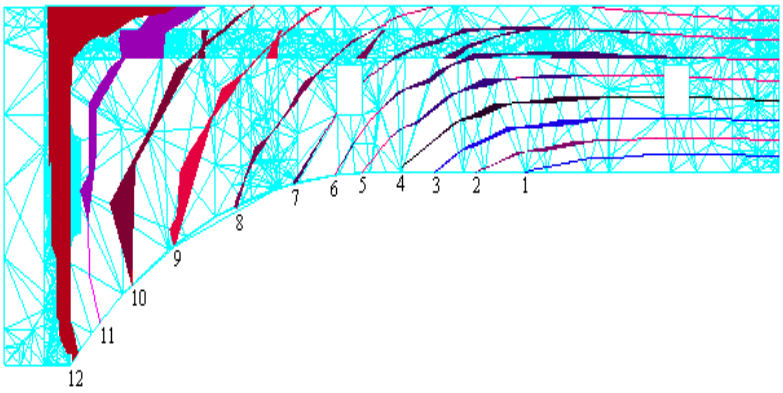
Figure 5.23(b) Boundary conditions and loading of the bracket

Fig. 5.24(a), 5.24(b) and 5.24(c) show the contour plots of deflections in *x*, *y* and *z* directions using four-node tetrahedral elements for different levels of deflections.



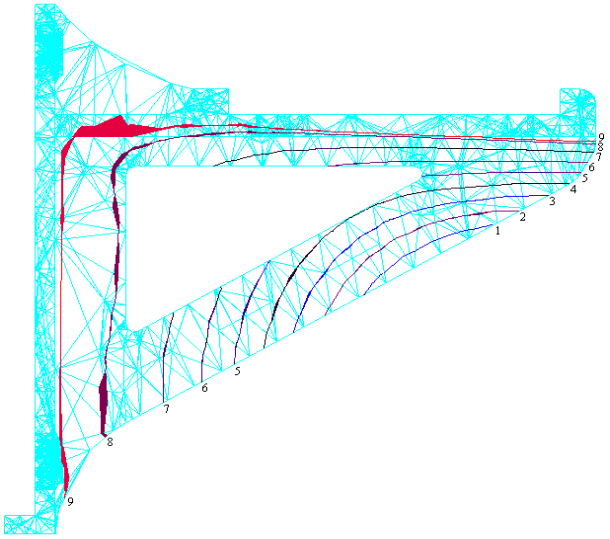
Contour Levels	C = δ_x
1	200 x10^{-4}
2	100 x10^{-4}
3	50 x10^{-4}
4	-5 x10^{-4}
5	-100 x10^{-4}
6	-250 x10^{-4}
7	-500 x10^{-4}

Figure 5.24(a) Contour surfaces of deflection in x-direction using four-node tetrahedrals for the different deflection levels for the bracket



Contour Levels	C = δ_y
1	440 x10^{-3}
2	400 x10^{-3}
3	350 x10^{-3}
4	300 x10^{-3}
5	250 x10^{-3}
6	200 x10^{-3}
7	150 x10^{-3}
8	100 x10^{-3}
9	50 x10^{-3}
10	20 x10^{-3}
11	5 x10^{-3}
12	0.1 x10^{-3}

Figure 5.24(b) Contour surfaces of deflection in y-direction using four-node tetrahedrals for the different deflection levels for the bracket



Contour Levels	C = δ_z
1	350 x10^{-3}
2	300 x10^{-3}
3	250 x10^{-3}
4	200 x10^{-3}
5	150 x10^{-3}
6	100 x10^{-3}
7	50 x10^{-3}
8	10 x10^{-3}
9	1 x10^{-3}

Figure 5.24(c) Contour surfaces of deflection in z-direction using four-node tetrahedrals for the different deflection levels for the bracket

The next problem is a heat conduction problem, with the specifications shown in Fig. 5.25(a). The temperature of 673^0C and 273^0C is fixed at the inside and outside of the reactor, as shown in Fig. 5.25(b). For the known heat flow rate, thermal conductivity of the materials as 0.01 W/m-deg and 0.0057 W/m-deg, and the given boundary temperatures, the values of temperature distribution are derived using FEA. For the given conditions, the temperature distribution is given by Eq. (3.30).

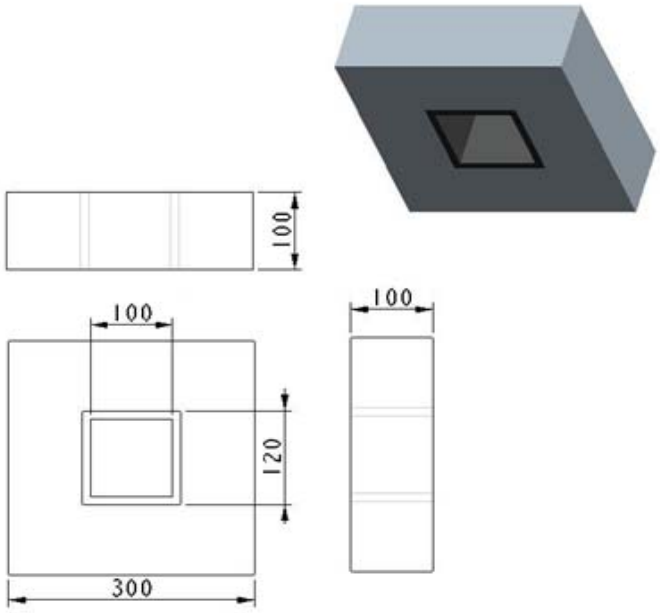


Figure 5.25(a) Specifications of the reactor

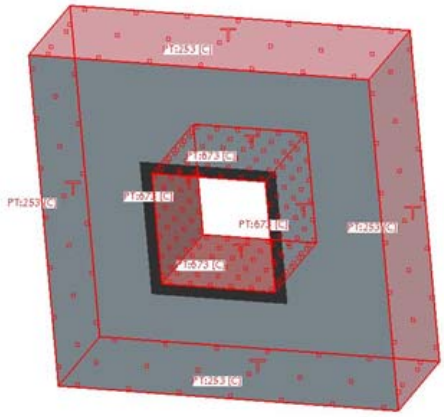
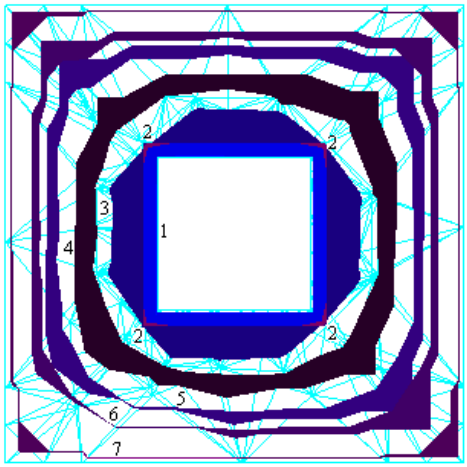


Figure 5.25(b) Boundary conditions and temperature fixation of the reactor

Fig. 5.26 shows the contour plots of temperature distribution using four-node tetrahedral elements for different levels of temperature.



Contour Levels	C = *T*
1	650
2	480
3	420
4	350
5	310
6	290
7	260

Figure 5.26 Contour plot of temperature distribution (in ^{o}C) using four-node tetrahedral elements for the reactor

It is observed that the contouring algorithm used here is very fast and takes 0.054945 seconds for the problem involving 276 four-node tetrahedral elements. The technique developed here can be used to plot contours for applications where high speed is required. Although the discussed algorithm simplifies contour plotting, inaccuracies due to lower order interpolation are however introduced.

5.6 IMPLEMENTATION OF THE ALGORITHMS FOR HEXAHEDRAL ELEMENT

The contour surface generation algorithms discussed in section 5.2, are implemented in the said system configuration.

The first problem is a deflection of a cantilever beam with specifications shown in Fig. 5.27(a). The beam is fixed at one end and a vertical load is applied at the other end, shown in Fig. 5.27(b). For a load of 100 N, Young's Modulus $200\,GPa$, Poisson's ratio 0.3, the values of vertical deflections are derived using FEA, considering it as a 3D problem. For the given loading conditions, the deflections are due to the bending moment given by Eqs. (3.26) to (3.28).

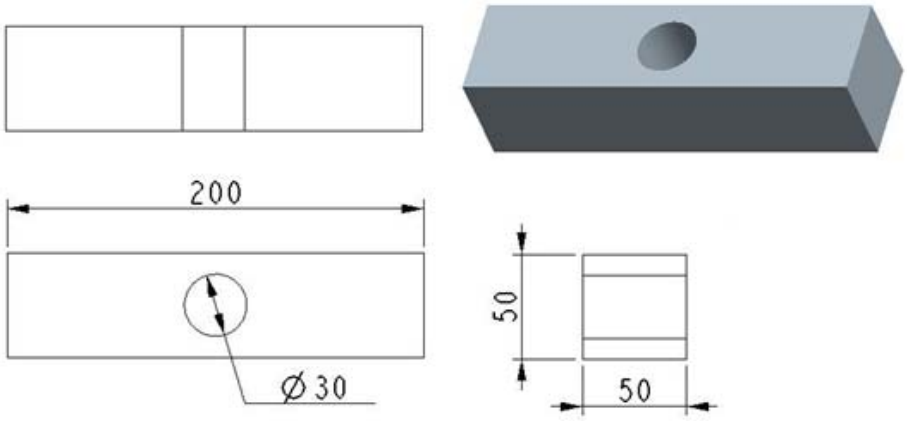


Figure 5.27(a) Specifications of the beam

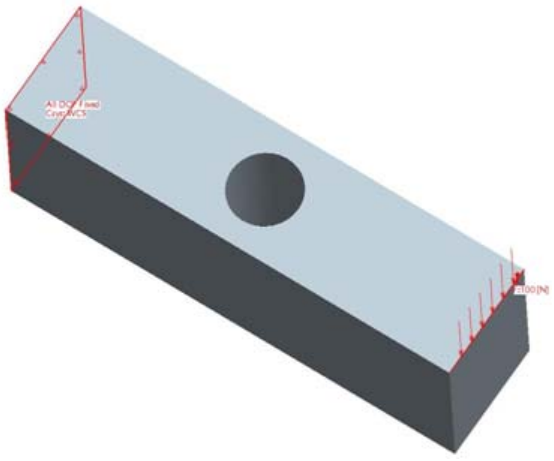
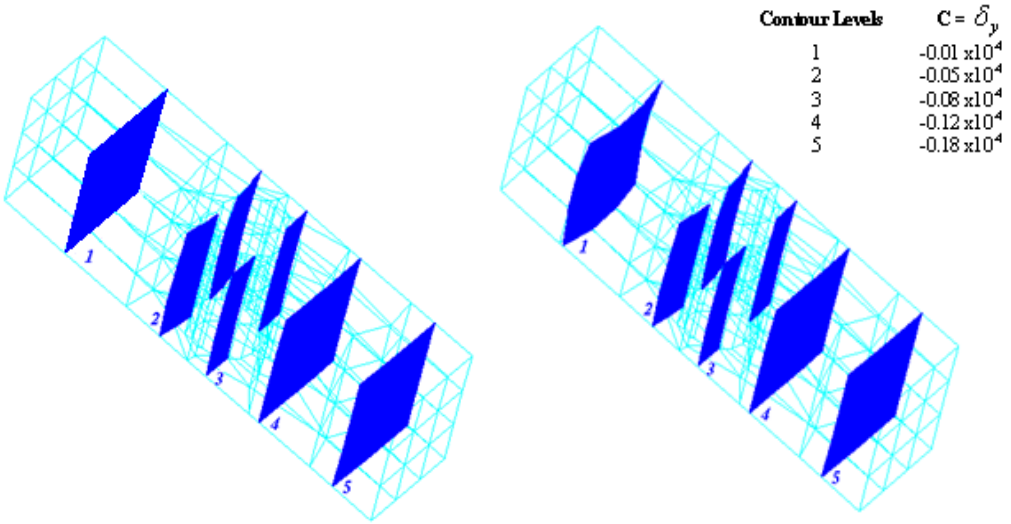


Figure 5.27(b) Boundary conditions and loading of the beam

To demonstrate the accuracy of the algorithm, the contour surfaces were plotted by dividing the hexahedral into different tetrahedral elements (as discussed in section 5.3.2) and then plotting the same using the discussed algorithm. Fig. 5.28 shows the contour surface plots of vertical deflections using the two methods for different levels of deflection. It can be seen from Fig. 5.28(a) and 5.28(b) that the behavior of contours using the discussed algorithm is more accurate.



(a) **(b)**

Figure 5.28 Contour surfaces of deflection in y-direction using eight-node hexahedral element by: (a) Dividing the hexahedral element into different tetrahedral elements; (b) Using the discussed algorithm

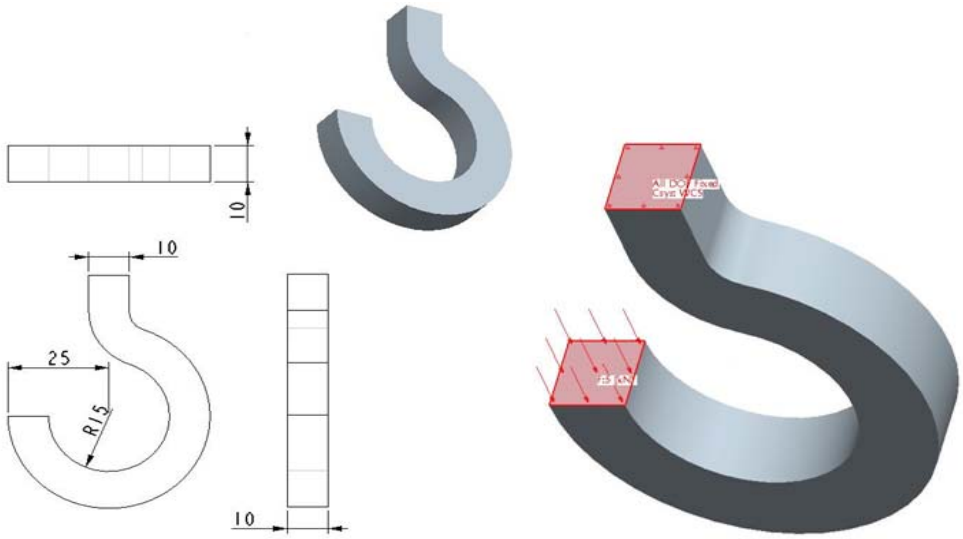


Figure 5.29 Specifications and conditions of the crane hook

The second problem is a deflection problem of a curved beam in the form of a hook shown in Fig. 5.29. The hook is fixed at one end and a vertical load is applied at the other end, shown in Fig. 5.29. For a load of 5 kN, Young's Modulus $200\,GPa$, Poisson's ratio 0.3, the values of vertical deflections are derived using FEA.

The general equation of strain (Singh, 2008) in the curved beam is given by Eq. (5.52).

$$\varepsilon = \frac{-My}{EAe(r_0 - y)} \tag{5.52}$$

with $r_0 = \dfrac{h}{\log_n\left(\dfrac{r_2}{r_1}\right)}$, $e = \rho_0 - \dfrac{h}{\log_n\left(\dfrac{r_2}{r_1}\right)}$ for a square cross section.

where M is the bending moment, y is the distance from the neutral surface, E is the Young's Modulus, A is the area of cross section and ρ_0 is the internal radius of curvature of the centroidal surface.

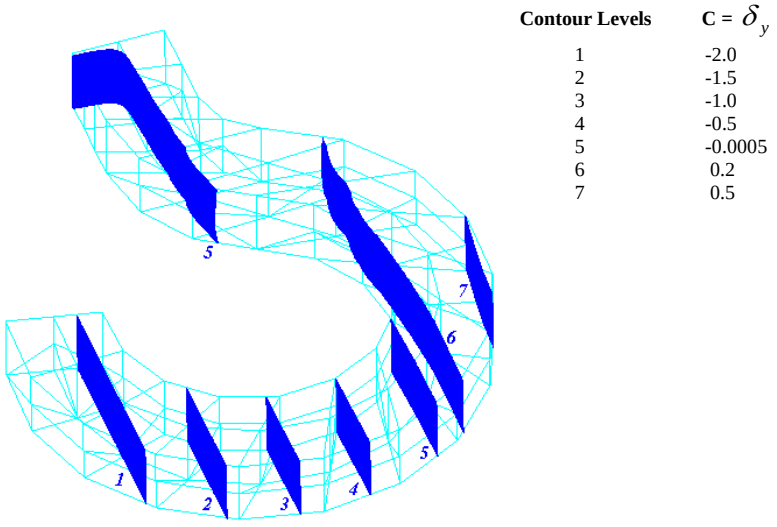


Figure 5.30 Contour surfaces of deflection in vertical direction using eight-node hexahedral elements for different deflection levels for the hook

We can derive deflection by the usual expression of strain. Fig. 5.30 shows the contour surface plots of vertical deflections using eight-node hexahedral elements for different levels of deflection.

Fig. 5.31(a) shows the specification of a plate. The beam is fixed at one end and a horizontal load is applied at the other end, shown in Fig. 5.31(b). For a load of 1000 N, Young's Modulus $0.2\times10^9\,MPa$, Poisson's ratio 0.3, the values of deflection are derived

using FEA, considering it as a 3D problem. For the given loading conditions, the deflections are due to the direct force which is given by Eq. (3.25).

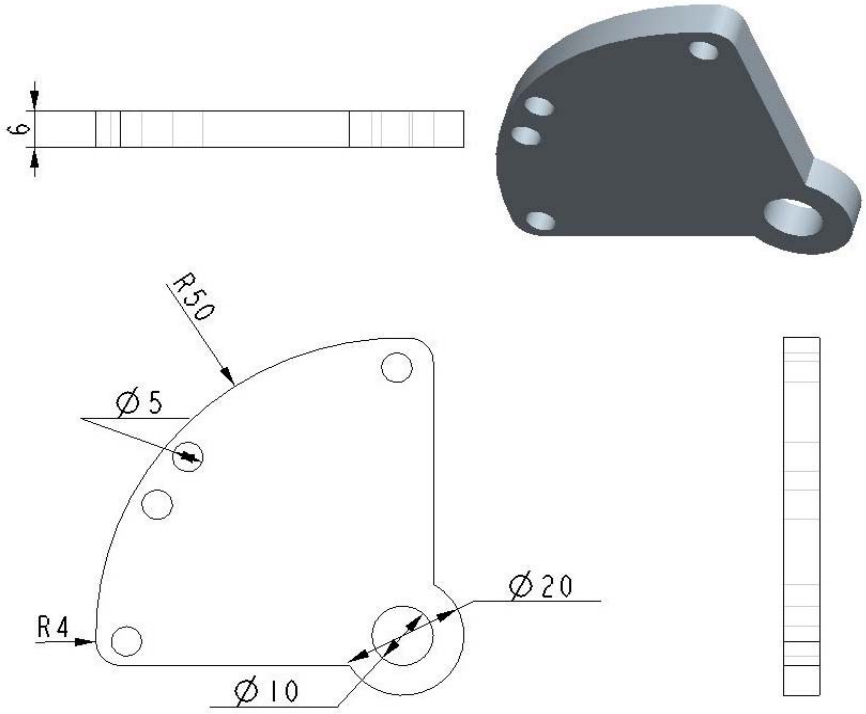


Figure 5.31(a) Specifications of the plate

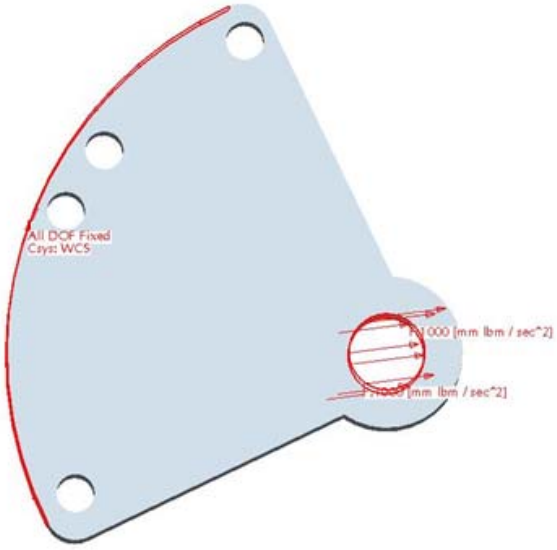
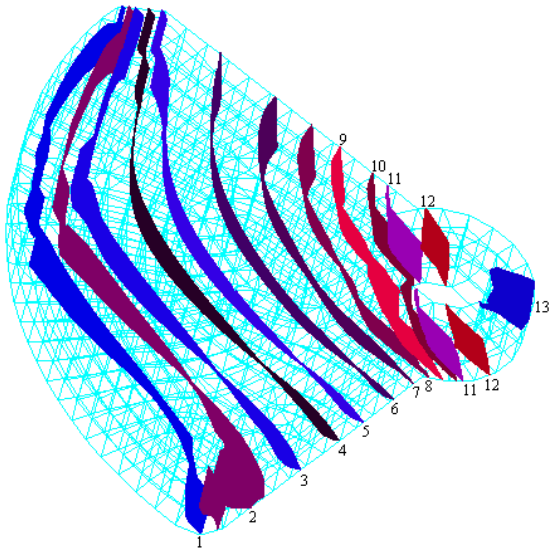


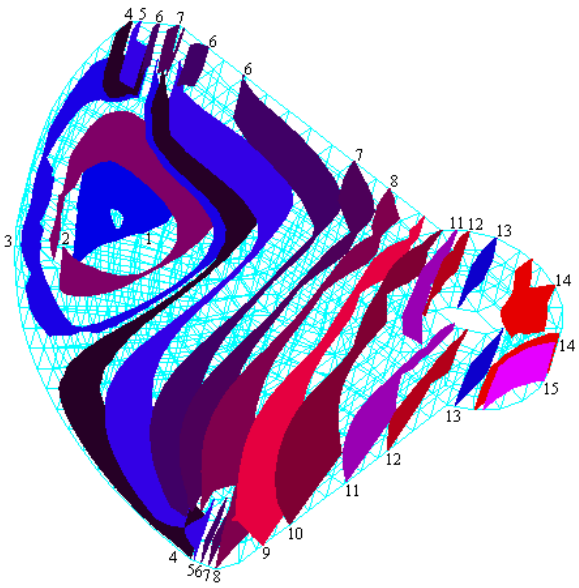
Figure 5.31(b) Boundary conditions and loading of the plate

Fig. 5.32(a), 5.32(b) and 5.32(c) show the contour plots of deflections in *x*, *y* and *z*-directions using eight-node hexahedral elements for different levels of deflection.



Contour Levels	C = δ_x
1	0.05 x10^{-2}
2	0.15 x10^{-2}
3	0.25 x10^{-2}
4	0.45 x10^{-2}
5	0.65 x10^{-2}
6	1.0 x10^{-2}
7	1.4 x10^{-2}
8	1.8 x10^{-2}
9	2.2 x10^{-2}
10	2.8 x10^{-2}
11	3.2 x10^{-2}
12	5.0 x10^{-2}
13	7.5 x10^{-2}

Figure 5.32(a) Contour surfaces of deflection in x-direction using eight-node hexahedral elements for the different deflection levels



Contour Levels	C = δ_y
1	-0.09 x10^{-2}
2	-0.05 x10^{-2}
3	-0.01 x10^{-2}
4	0.005 x10^{-2}
5	0.05 x10^{-2}
6	0.15 x10^{-2}
7	0.25 x10^{-2}
8	0.35 x10^{-2}
9	0.55 x10^{-2}
10	0.75 x10^{-2}
11	1.2 x10^{-2}
12	1.6 x10^{-2}
13	3.0 x10^{-2}
14	4.1 x10^{-2}
15	4.4 x10^{-2}

Figure 5.32(b) Contour surfaces of deflection in y-direction using eight-node hexahedral elements for the different deflection levels

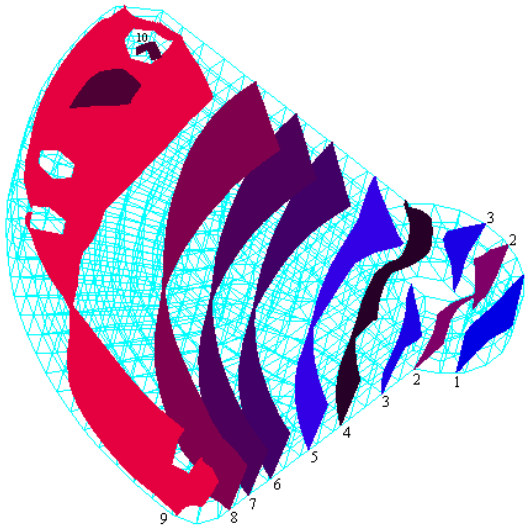


Figure 5.32(c) Contour surfaces of deflection in z-direction using eight-node hexahedral elements for the different deflection levels

Fig. 5.33(a) shows the specification of the ring. The ring is fixed and loaded as shown in Fig. 5.33(b). For a load of 1000 N, Young's Modulus $0.2\times10^{9}\,MPa$, Poisson's ratio 0.3, the values of deflections are derived using FEA, considering it as a 3D ring problem. For the given loading conditions, the deflections are due to the combined effect of direct force and bending moment.

As the ring is vertically loaded, the general expression for resultant stress which can be used to calculate the displacements is given by Eqs. (5.50) and (5.51).

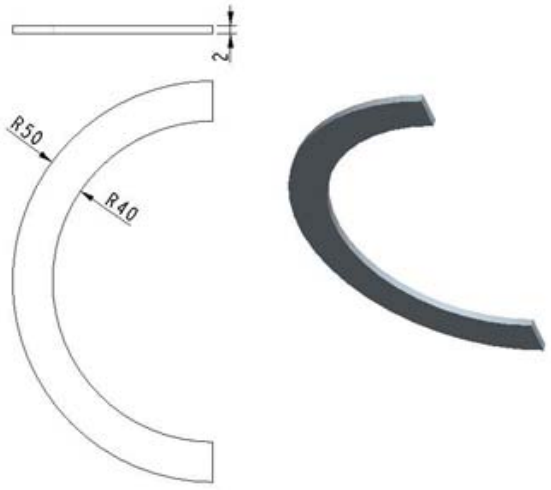


Figure 5.33(a) Specifications of the ring

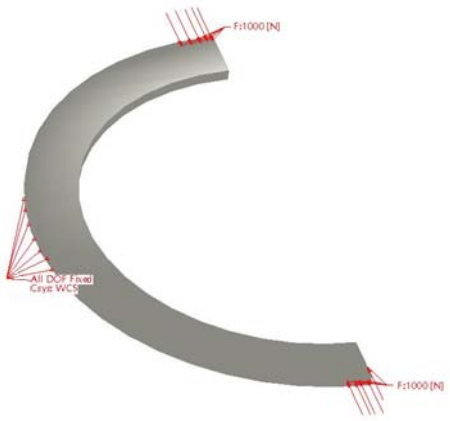
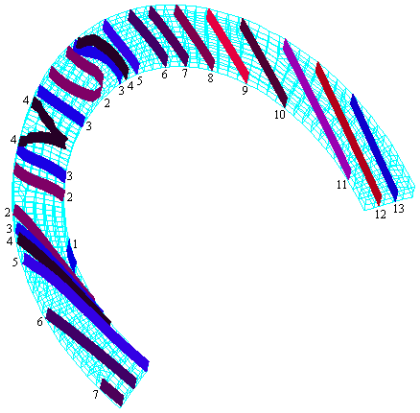


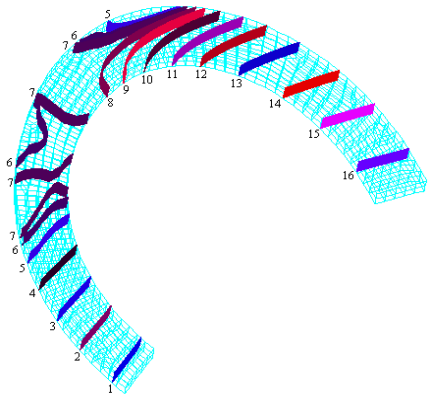
Figure 5.33(b) Boundary conditions and loading of the ring

Fig. 5.34(a), 5.34(b) and 5.34(c) show the contour plots of deflections in *x*, *y* and *z*-directions using eight-node hexahedral elements for different levels of deflections.



Contour Levels	C = δ_x
1	-20 x10^{-2}
2	-10 x10^{-2}
3	-5 x10^{-2}
4	-1 x10^{-2}
5	10 x10^{-2}
6	70 x10^{-2}
7	130 x10^{-2}
8	220 x10^{-2}
9	350 x10^{-2}
10	500 x10^{-2}
11	700 x10^{-2}
12	850 x10^{-2}
13	1000 x10^{-2}

Figure 5.34(a) Contour surfaces of deflection in x-direction using eight-node hexahedral elements for the different deflection levels for the ring



Contour Levels	C = δ_y
1	-250 x10^{-2}
2	-150 x10^{-2}
3	-80 x10^{-2}
4	-30 x10^{-2}
5	-5 x10^{-2}
6	1 x10^{-2}
7	3 x10^{-2}
8	15 x10^{-2}
9	30 x10^{-2}
10	60 x10^{-2}
11	120 x10^{-2}
12	200 x10^{-2}
13	350 x10^{-2}
14	600 x10^{-2}
15	900 x10^{-2}
16	1300 x10^{-2}

Figure 5.34(b) Contour surfaces of deflection in y-direction using eight-node hexahedral elements for the different deflection levels

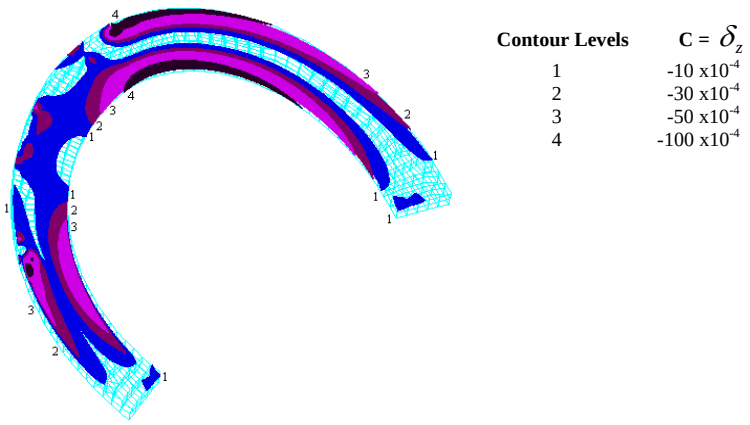


Figure 5.34(c) Contour surfaces of deflection in z-direction using eight-node hexahedral elements for the different deflection levels

Fig. 5.35(a) shows the specification of a U-section beam. The beam is fixed and vertically loaded as shown in Fig. 5.35(b). For a load of 1000 N, Young's Modulus 0.2×10^{9} *MPa*, Poisson's ratio 0.3, the values of deflections are derived using FEA, considering it as a 3D problem. For the given loading conditions, the deflections are due to bending moments which are given by Eqs. (3.26) to (3.28).

Fig. 5.36(a), 5.36(b) and 5.36(c) show the contour plots of deflections in *x*, *y* and *z*-directions using eight-node hexahedral elements for different levels of deflection.

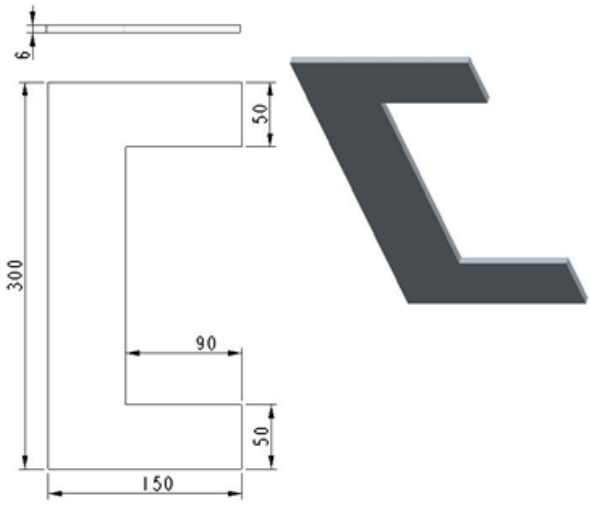


Figure 5.35(a) Specifications of the U-section beam

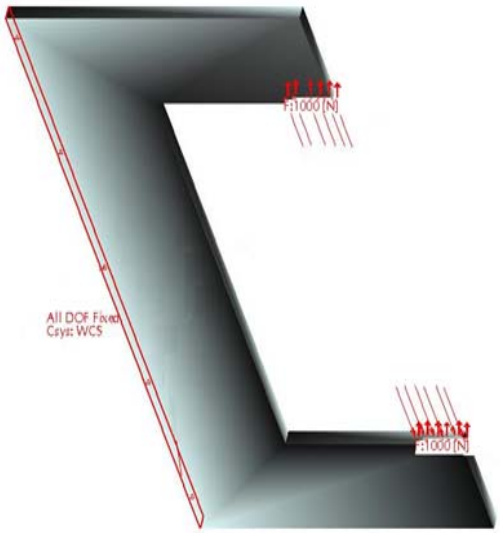
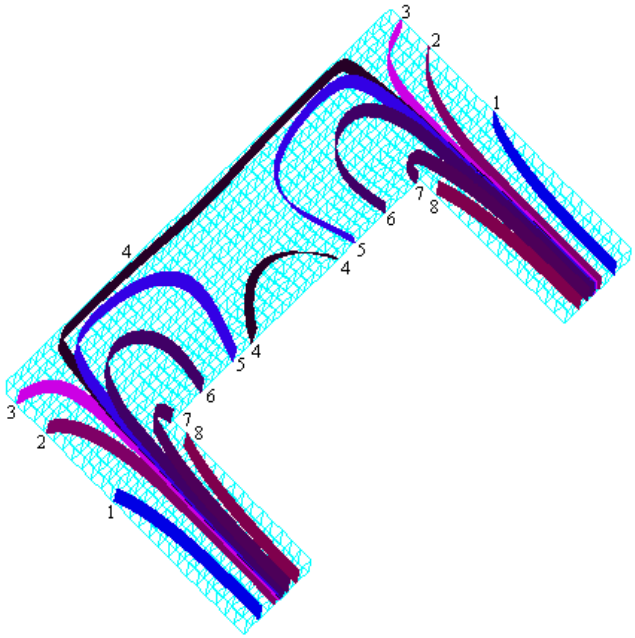
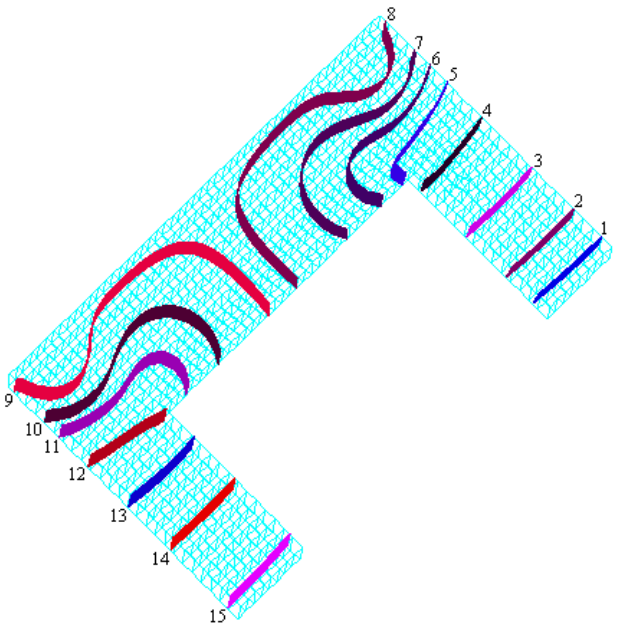


Figure 5.35(b) Boundary conditions and loading of the U-section beam



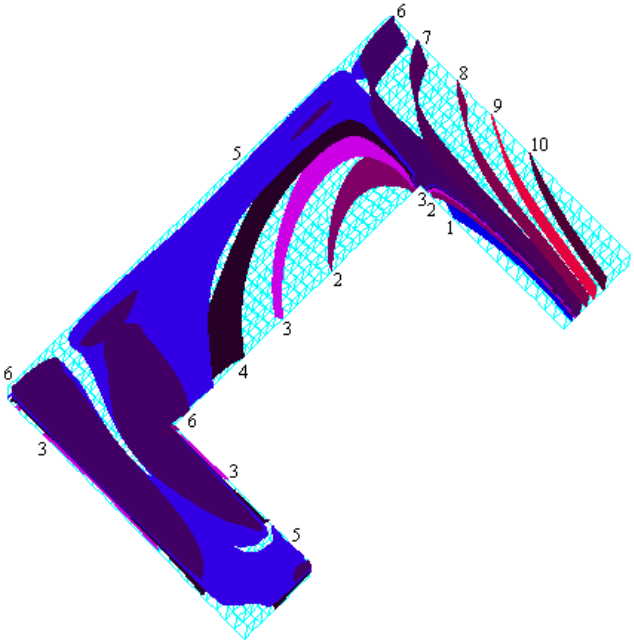
Contour Levels	C = δ_x
1	-7.0 x10^{-2}
2	-2.0 x10^{-2}
3	-0.5 x10^{-2}
4	0.009 x10^{-2}
5	0.1 x10^{-2}
6	0.5 x10^{-2}
7	2.0 x10^{-2}
8	5.0 x10^{-2}

Figure 5.36(a) Contour surfaces of deflection in x-direction using eight-node hexahedral elements for the different deflection levels



Contour Levels	C = δ_y
1	-37 x10^{-2}
2	-29 x10^{-2}
3	-17 x10^{-2}
4	-6 x10^{-2}
5	-2 x10^{-2}
6	-1 x10^{-2}
7	-0.5 x10^{-2}
8	-0.1 x10^{-2}
9	0.1 x10^{-2}
10	0.5 x10^{-2}
11	1 x10^{-2}
12	3 x10^{-2}
13	9 x10^{-2}
14	19 x10^{-2}
15	35 x10^{-2}

Figure 5.36(b) Contour surfaces of deflection in y-direction using eight-node hexahedral elements for the different deflection levels



Contour Levels	C = δ_z
1	-50 x10^{-4}
2	-20 x10^{-4}
3	-8 x10^{-4}
4	-3 x10^{-4}
5	-0.5 x10^{-4}
6	2 x10^{-4}
7	25 x10^{-4}
8	90 x10^{-4}
9	160 x10^{-4}
10	240 x10^{-4}

Figure 5.36(c) Contour surfaces of deflection in z-direction using eight-node hexahedral elements for the different deflection levels

Fig. 5.37(a) shows the specification of a lever. The lever is fixed at the hole and loaded as shown in Fig. 5.37(b). For a load of 1000 N, Young's Modulus 0.2×10^{9} *MPa*, Poisson's ratio 0.3, the values of deflection are derived using FEA, considering it as a 3D problem. For the given loading conditions, the deflections are due to bending moments which are given by Eqs. (3.26) to (3.28).

Fig. 5.38(a), 5.38(b) and 5.38(c) shows the contour plots of deflections in *x*, *y* and *z*-directions using eight-node hexahedral elements for different levels of deflections.

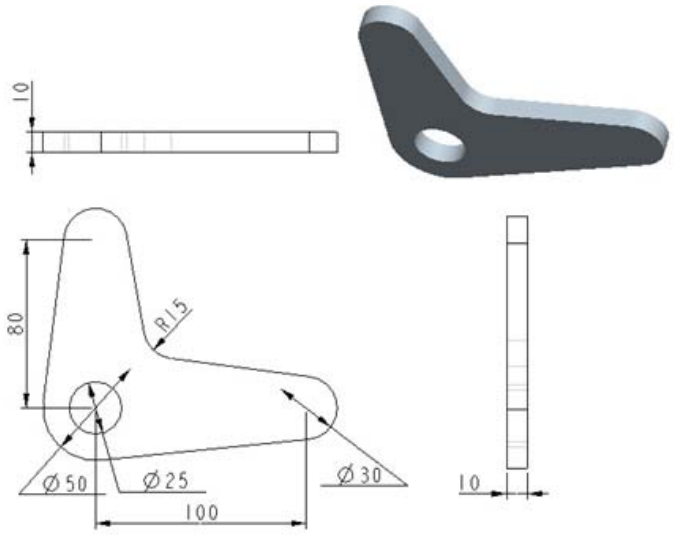


Figure 5.37(a) Specifications of the lever

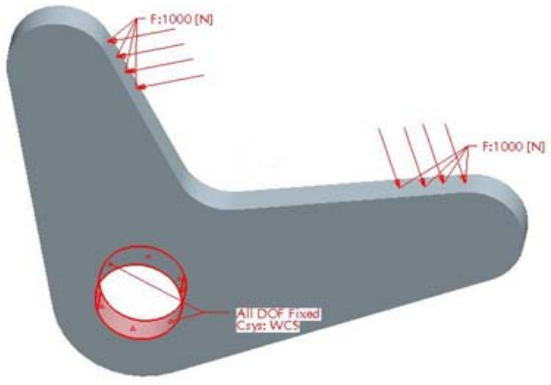
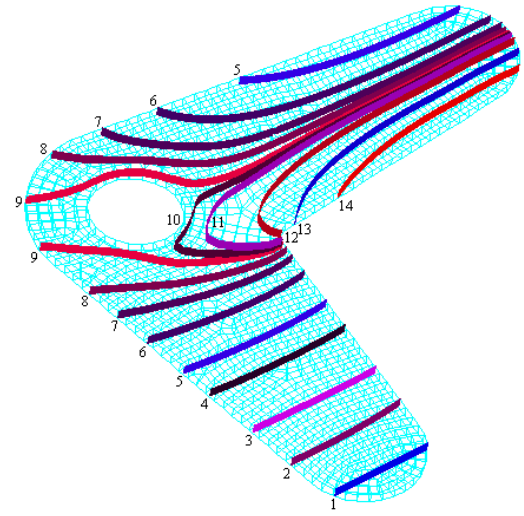
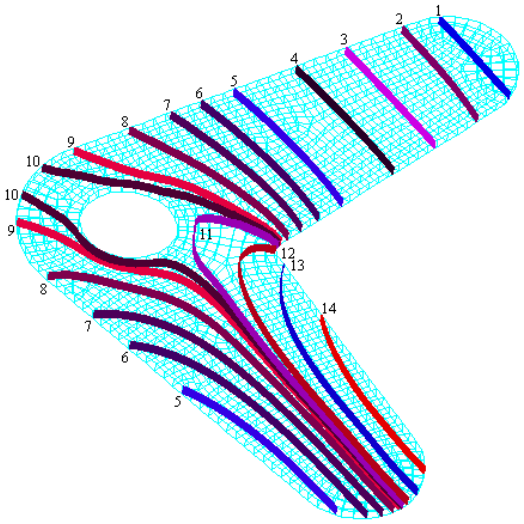


Figure 5.37(b) Boundary conditions and loading of the lever



Contour Levels	C = δ_x
1	-180 x10^{-3}
2	-140 x10^{-3}
3	-100 x10^{-3}
4	-60 x10^{-3}
5	-40 x10^{-3}
6	-20 x10^{-3}
7	-10 x10^{-3}
8	-5.0 x10^{-3}
9	-1.0 x10^{-3}
10	0.5 x10^{-3}
11	2.0 x10^{-3}
12	7.0 x10^{-3}
13	17 x10^{-3}
14	30 x10^{-3}

Figure 5.38(a) Contour surfaces of deflection in x-direction using eight-node hexahedral elements for the different deflection levels for the lever



Contour Levels	C = δ_y
1	-20 x10^{-2}
2	-17 x10^{-2}
3	-12 x10^{-2}
4	-8 x10^{-2}
5	-4 x10^{-2}
6	-2.5 x10^{-2}
7	-1.5 x10^{-2}
8	-0.7 x10^{-2}
9	-0.2 x10^{-2}
10	-0.05 x10^{-2}
11	0.1 x10^{-2}
12	0.5 x10^{-2}
13	1.5 x10^{-2}
14	3.0 x10^{-2}

Figure 5.38(b) Contour surfaces of deflection in y-direction using eight-node hexahedral elements for the different deflection levels for the lever

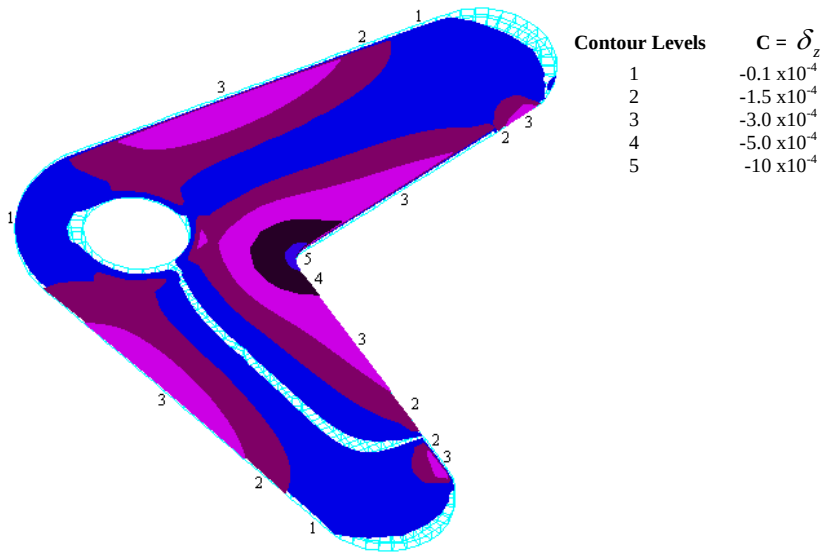


Figure 5.38(c) Contour surfaces of deflection in z-direction using eight-node hexahedral elements for the different deflection levels for the lever

The contour surfaces of all the problems show the non-linear form of different surfaces plotted. This shows that the contour plots are more accurate. The execution time for some of the problems shows that the developed algorithm takes nearly 50% of less time as taken by a direct method.

We have demonstrated the algorithm by analyzing the number of different problems. The algorithm takes 0.5495 seconds for the generation and plotting of one contour surface for 126 hexahedral elements on a P-IV PC with 3.0 GHz CPU and 256 MB RAM for the cantilever beam shown in Fig. 5.27(a). The direct approach takes 2.107 seconds for the same problem run on the same environment. For the problem depicted in Fig. 5.29(a), the time taken for 68 elements is 0.3297 seconds while the direct approach takes 0.604396 seconds for generation and plotting of one contour surface.

5.7 CONCLUDING REMARKS

The contour plotting technique using shape functions is demonstrated in the present chapter.

The algorithm developed using four-node tetrahedral element can be used to plot contours for applications where high speed is required, however, inaccuracies due to lower order interpolation are introduced.

Fast and accurate algorithms for the generation of contour surfaces are described using hexahedral elements. The contour surfaces are described in the form of groups of boundaries of contour segments and their interior points are derived using the contour equation. The locations of contour boundaries and the interior points on contour surfaces are as accurate as the interpolation results obtained by hexahedral elements and thus there are no discrepancies between the analysis and visualization results.

To further increase the accuracy of results, higher order interpolation is investigated in the next chapter.

CHAPTER - VI

CONTOURING IN 3D USING NON-LINEAR ELEMENTS

6.1 INTRODUCTION

As discussed in the previous chapter, higher-order elements represent the data better than lower-order elements. However, these generate large amount of numerical data as compared to lower-order elements. So the generation of contour surfaces on higher-order elements is more important, but it is a complex task. In order to simplify the contour surface generation, normally a higher order element is split into linear elements and linear interpolation of the physical quantity being interpreted is performed. The linear approximation of a non-linear function creates inaccuracies in the results which is unacceptable in many situations, particularly when the variation in the physical quantity is large in a small neighbourhood. In this chapter, a technique is proposed for accurate tracing of contours using ten-node tetrahedral element, as the tetrahedral elements are commonly used in FEA due to their shape. The literature shows that no work related to contour plotting is done on quadratic tetrahedral elements using shape functions.

6.2 CONTOUR EQUATION OVER TEN-NODE TETRAHEDRAL ELEMENT

A quadratic interpolation can be performed by taking six more discrete points on the middle of the six sides of the tetrahedral element, as shown in Fig. 6.1. This is called ten-node tetrahedral element.

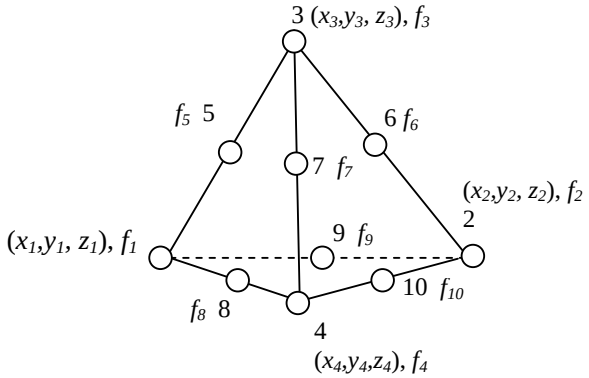


Figure 6.1 A ten-node tetrahedral element

The interpolation is performed as shown in Eqs. (6.1) to (6.4a).

$$x = \sum_{i=1}^{4} N_i x_i \tag{6.1}$$

$$y = \sum_{i=1}^{4} N_i y_i \tag{6.2}$$

$$z = \sum_{i=1}^{4} N_i z_i \tag{6.3}$$

$$f(x, y) = \sum_{i=1}^{10} N_i f_i \tag{6.4a}$$

where $N_i = (2L_i - 1)L_i$, $i = 1,2,3,4$ and $N_5 = 4L_1L_2$, $N_6 = 4L_1L_3$, $N_7 = 4L_1L_4$,

$N_8 = 4L_2L_3$, $N_9 = 4L_3L_4$, $N_{10} = 4L_2L_4$ and $L_i = \frac{1}{6V}(a_i + b_i x + c_i y + d_i z)$ (6.4b)

The coefficients a_i ,b_i , c_i and d_i are cofactors of the matrix, *M*, where

$$M = \begin{bmatrix} 1 & x_i & y_i & z_i \\ 1 & x_j & y_j & z_j \\ 1 & x_m & y_m & z_m \\ 1 & x_p & y_p & z_p \end{bmatrix} \tag{6.5a}$$

and, $6V = \det(M)$. The functions $L_i = L_i(x, y, z), \quad i = 1,2,3,4$ are called shape functions.

A ten node interpolation using a tetrahedral element provides quadratic interpolation and requires less number of elements for the same order of approximation in solution using finite element technique. However, in order to plot contours, a ten node element is split into number of four node tetrahedral elements and linear interpolation is performed to plot contours. Although it simplifies contour plotting, inaccuracies are introduced. This degrades the visual presentation of results in regions where the function changes rapidly or where contours are closely placed. Moreover, the benefits due to higher order approximations in finite element results are not reflected in visual presentation of results. Motivated by these considerations, the exact tracing of contours over a ten-node element is performed using quadratic interpolation. The method is explained as follows:

The contour equation assumes the following form over a ten-node tetrahedral element:

$$\sum_{i=1}^{10} N_i f_i = C \tag{6.5b}$$

Substituting the values of shape function, N_i from Eq. (6.4b) into Eq. (6.5b) and simplifying, we get

$$l_0 + l_1 x^2 + l_2 y^2 + l_3 z^2 + l_4 x + l_5 y + l_6 z + l_7 xy + l_8 yz + l_9 zx = C \tag{6.6}$$

where

$$l_0 = \frac{f_1}{18V^2}\left(a_1^2 - 3Va_1\right) + \frac{f_2}{18V^2}\left(a_2^2 - 3Va_2\right) + \frac{f_3}{18V^2}\left(a_3^2 - 3Va_3\right) + \frac{f_4}{18V^2}\left(a_4^2 - 3Va_4\right) +$$

$$\frac{f_5}{9V^2}\left(a_1a_2\right) + \frac{f_6}{9V^2}\left(a_1a_3\right) + \frac{f_7}{9V^2}\left(a_1a_4\right) + \frac{f_8}{9V^2}\left(a_2a_3\right) + \frac{f_9}{9V^2}\left(a_3a_4\right) + \frac{f_{10}}{9V^2}\left(a_2a_4\right),$$

$$l_1 = \frac{f_1}{18V^2}\left(b_1^2\right) + \frac{f_2}{18V^2}\left(b_2^2\right) + \frac{f_3}{18V^2}\left(b_3^2\right) + \frac{f_4}{18V^2}\left(b_4^2\right) +$$

$$\frac{f_5}{9V^2}\left(b_1b_2\right) + \frac{f_6}{9V^2}\left(b_1b_3\right) + \frac{f_7}{9V^2}\left(b_1b_4\right) + \frac{f_8}{9V^2}\left(b_2b_3\right) + \frac{f_9}{9V^2}\left(b_3b_4\right) + \frac{f_{10}}{9V^2}\left(b_2b_4\right),$$

$$l_2 = \frac{f_1}{18V^2}\left(c_1^2\right) + \frac{f_2}{18V^2}\left(c_2^2\right) + \frac{f_3}{18V^2}\left(c_3^2\right) + \frac{f_4}{18V^2}\left(c_4^2\right) +$$

$$\frac{f_5}{9V^2}\left(c_1c_2\right) + \frac{f_6}{9V^2}\left(c_1c_3\right) + \frac{f_7}{9V^2}\left(c_1c_4\right) + \frac{f_8}{9V^2}\left(c_2c_3\right) + \frac{f_9}{9V^2}\left(c_3c_4\right) + \frac{f_{10}}{9V^2}\left(c_2c_4\right),$$

$$l_3 = \frac{f_1}{18V^2}\left(d_1^2\right) + \frac{f_2}{18V^2}\left(d_2^2\right) + \frac{f_3}{18V^2}\left(d_3^2\right) + \frac{f_4}{18V^2}\left(d_4^2\right) +$$

$$\frac{f_5}{9V^2}\left(d_1d_2\right) + \frac{f_6}{9V^2}\left(d_1d_3\right) + \frac{f_7}{9V^2}\left(d_1d_4\right) + \frac{f_8}{9V^2}\left(d_2d_3\right) + \frac{f_9}{9V^2}\left(d_3d_4\right) + \frac{f_{10}}{9V^2}\left(d_2d_4\right),$$

$$l_4 = \frac{f_1}{18V^2}\left(2a_1b_1 - 3Vb_1\right) + \frac{f_2}{18V^2}\left(2a_2b_2 - 3Vb_2\right) + \frac{f_3}{18V^2}\left(2a_3b_3 - 3Vb_3\right) + \frac{f_4}{18V^2}\left(2a_4b_4 - 3Vb_4\right) +$$

$$\frac{f_5}{9V^2}\left(a_1b_2 + b_1a_2\right) + \frac{f_6}{9V^2}\left(a_1b_3 + b_1a_3\right) + \frac{f_7}{9V^2}\left(a_1b_4 + b_1a_4\right) + \frac{f_8}{9V^2}\left(a_2b_3 + b_2a_3\right) +$$
$$\frac{f_9}{9V^2}\left(a_3b_4 + b_3a_4\right) + \frac{f_{10}}{9V^2}\left(a_2b_4 + b_2a_4\right),$$

$$l_5 = \frac{f_1}{18V^2}\left(2a_1c_1 - 3Vc_1\right) + \frac{f_2}{18V^2}\left(2a_2c_2 - 3Vc_2\right) + \frac{f_3}{18V^2}\left(2a_3c_3 - 3Vc_3\right) + \frac{f_4}{18V^2}\left(2a_4c_4 - 3Vc_4\right) +$$

$$\frac{f_5}{9V^2}\left(a_1c_2 + c_1a_2\right) + \frac{f_6}{9V^2}\left(a_1c_3 + c_1a_3\right) + \frac{f_7}{9V^2}\left(a_1c_4 + c_1a_4\right) + \frac{f_8}{9V^2}\left(a_2c_3 + c_2a_3\right) +$$
$$\frac{f_9}{9V^2}\left(a_3c_4 + c_3a_4\right) + \frac{f_{10}}{9V^2}\left(a_2c_4 + c_2a_4\right),$$

$$l_6 = \frac{f_1}{18V^2}(2a_1d_1 - 3Vd_1) + \frac{f_2}{18V^2}(2a_2d_2 - 3Vd_2) + \frac{f_3}{18V^2}(2a_3d_3 - 3Vd_3) + \frac{f_4}{18V^2}(2a_4d_4 - 3Vd_4) +$$
$$\frac{f_5}{9V^2}(a_1d_2 + d_1a_2) + \frac{f_6}{9V^2}(a_1d_3 + d_1a_3) + \frac{f_7}{9V^2}(a_1d_4 + d_1a_4) + \frac{f_8}{9V^2}(a_2d_3 + d_2a_3) +$$
$$\frac{f_9}{9V^2}(a_3d_4 + d_3a_4) + \frac{f_{10}}{9V^2}(a_2d_4 + d_2a_4),$$

$$l_7 = \frac{f_1}{18V^2}(2b_1c_1) + \frac{f_2}{18V^2}(2b_2c_2) + \frac{f_3}{18V^2}(2b_3c_3) + \frac{f_4}{18V^2}(2b_4c_4) +$$
$$\frac{f_5}{9V^2}(b_1c_2 + c_1b_2) + \frac{f_6}{9V^2}(b_1c_3 + c_1b_3) + \frac{f_7}{9V^2}(b_1c_4 + c_1b_4) + \frac{f_8}{9V^2}(b_2c_3 + c_2b_3) +$$
$$\frac{f_9}{9V^2}(b_3c_4 + c_3b_4) + \frac{f_{10}}{9V^2}(b_2c_4 + c_2b_4),$$

$$l_8 = \frac{f_1}{18V^2}(2c_1d_1) + \frac{f_2}{18V^2}(2c_2d_2) + \frac{f_3}{18V^2}(2c_3d_3) + \frac{f_4}{18V^2}(2c_4d_4) +$$
$$\frac{f_5}{9V^2}(c_1d_2 + d_1c_2) + \frac{f_6}{9V^2}(c_1d_3 + d_1c_3) + \frac{f_7}{9V^2}(c_1d_4 + d_1c_4) + \frac{f_8}{9V^2}(c_2d_3 + d_2c_3) +$$
$$\frac{f_9}{9V^2}(c_3d_4 + d_3c_4) + \frac{f_{10}}{9V^2}(c_2d_4 + d_2c_4),$$

$$l_9 = \frac{f_1}{18V^2}(2b_1d_1) + \frac{f_2}{18V^2}(2b_2d_2) + \frac{f_3}{18V^2}(2b_3d_3) + \frac{f_4}{18V^2}(2b_4d_4) +$$
$$\frac{f_5}{9V^2}(b_1d_2 + d_1b_2) + \frac{f_6}{9V^2}(b_1d_3 + d_1b_3) + \frac{f_7}{9V^2}(b_1d_4 + d_1b_4) + \frac{f_8}{9V^2}(b_2d_3 + d_2b_3) +$$
$$\frac{f_9}{9V^2}(b_3d_4 + d_3b_4) + \frac{f_{10}}{9V^2}(b_2d_4 + d_2b_4).$$

Eq. (6.6) can be compared with the following most general equation of second degree which represents the quadratic surfaces or quadrics (Vasishtha and Agarwal, 2000)

$$F(x, y, z) = ax^2 + by^2 + cz^2 + 2fyz + 2gzx + 2hxy + 2ux + 2vy + 2wz + d = 0 \quad (6.7)$$

To select the type of quadric surface, Eq. (6.7) is to be reduced to either of the following two, by the transformation of axes:

First form: $\lambda_1 x^2 + \lambda_2 y^2 + \lambda_3 z^2 = \mu$ (6.8)

Second form: $\lambda_1 x^2 + \lambda_2 y^2 = 2\mu z$ (6.9)

By giving different values to λ_1, λ_2, λ_3 and μ the Eqs. (6.8) and (6.9) will give rise to different types of surfaces which are given in Table 6.1 and 6.2.

Case	Condition	Quadric Surface
1	$Ax^2 + By^2 + Cz^2 = 1$	Ellipsoid
2	$A\left(x^2 + y^2\right) + Cz^2 = 1$	Ellipsoid of revolution
3	$A\left(x^2 + y^2 + z^2\right) = 1$	Sphere
4	$Ax^2 + By^2 - Cz^2 = 1$	Hyperboloid of one sheet
5	$Ax^2 - By^2 - Cz^2 = 1$	Hyperboloid of two sheet
6	$A\left(x^2 - z^2\right) + By^2 = 1$	Hyperboloid of revolution
7	$Ax^2 + By^2 + Cz^2 = 0$	Cone
8	$Ax^2 + By^2 = 1$	Elliptic cylinder
9	$Ax^2 - By^2 = 1$	Hyperbolic cylinder
10	$Ax^2 - By^2 = 0$	Pair of intersecting planes
11	$Ax^2 = 1$ or $By^2 = 1$ or $Cz^2 = 1$	Pair of parallel planes

Table 6.1 Different forms of Eq. (6.8)

Case	Condition	Quadric Surface
1	$Ax^2 + By^2 = 2Cz$	Elliptic paraboloid
2	$Ax^2 - By^2 = 2Cz$	Hyperbolic paraboloid
3	$A\left(x^2 + y^2\right) = 2Cz$	Paraboloid of revolution
4	$Ax^2 = \pm 2Cz$	Parabolic cylinder

Table 6.2 Different forms of Eq. (6.9)

The coordinates of the centre of the surface $F(x, y, z) = 0$ is found by taking the centre as (α, β, γ). Shifting the origin to (α, β, γ), Eq. (6.7) transforms to the form shown in Eq. (6.10).

$$a(x+\alpha)^2 + b(y+\beta)^2 + c(z+\gamma)^2 + 2f(y+\beta)(z+\gamma) + 2g(z+\gamma)(x+\alpha) +$$
$$2h(x+\alpha)(y+\beta) + 2u(x+\alpha) + 2v(y+\beta) + 2w(z+\gamma) + d = 0 \qquad (6.10)$$

Now the centre of Eq. (6.10) is the origin and hence it will be a homogeneous equation. Therefore the first degree terms should vanish, so

$$a\alpha + h\beta + g\gamma + u = 0 \quad i.e. \quad \frac{1}{2}\partial F/\partial\alpha = 0 \quad i.e. \quad \partial F/\partial\alpha = 0$$

$$h\alpha + b\beta + f\gamma + v = 0 \quad i.e. \quad \frac{1}{2}\partial F/\partial\beta = 0 \quad i.e. \quad \partial F/\partial\beta = 0 \qquad (6.11)$$

$$g\alpha + f\beta + c\gamma + w = 0 \quad i.e. \quad \frac{1}{2}\partial F/\partial\gamma = 0 \quad i.e. \quad \partial F/\partial\gamma = 0$$

Rearranging Eq. (6.10), we get

$$a\alpha^2 + b\beta^2 + c\gamma^2 + 2f\beta\gamma + 2g\gamma\alpha + 2h\alpha\beta + 2u\alpha + 2v\beta + 2w\gamma + d$$

$$\alpha(a\alpha + h\beta + g\gamma + u) + \beta(h\alpha + b\beta + f\gamma + v) + \gamma(g\alpha + f\beta + c\gamma + w) + u\alpha + v\beta + w\gamma + d$$

Using Eq. (6.11), the above equation can be written as

$$u\alpha + v\beta + w\gamma + d = d' \qquad (6.12)$$

Hence the transformed Eq. (6.10) reduces to the following form:

$$ax^2 + by^2 + cz^2 + 2fyz + 2gzx + 2hxy + d' = 0$$

where d' is given by Eq. (6.12) and the centre (α, β, γ) is given by the Eq. (6.11).

The following steps are used to reduce a general equation, $F(x, y, z) = 0$ of second degree to the standard form provided the terms of second degree do not form a perfect square.

1. Form the discriminating cubic and solve it.
2. If all the three characteristic roots (say λ_1, λ_2 and λ_3) are different from zero, then find the coordinates, (α, β, γ) of the centre by solving the Eq. (6.11).
3. Shift the origin to the centre, (α, β, γ) and then rotate the axes, the reduced equation becomes

   $$\lambda_1 x^2 + \lambda_2 y^2 + \lambda_3 z^2 = \mu$$

   where $\mu = -(u\alpha + v\beta + w\gamma + d)$
4. If one root (say $\lambda_3 = 0$) is equal to zero, then corresponding to this value, find the principal directions, l_3, m_3, n_3 from any two of the following equations:

   $$al_3 + hm_3 + gn_3 = 0$$

$$hl_3 + bm_3 + fn_3 = 0 \qquad (6.13)$$

$$gl_3 + fm_3 + cn_3 = 0$$

And evaluate $k = ul_3 + vm_3 + wn_3 = 0$

If $k \neq 0$. The reduced equation is $\lambda_1 x^2 + \lambda_2 y^2 + 2kz = 0$, and represents an elliptic or hyperbolic paraboloid whose vertex is given by solving any two of the Eqs. (6.7), (6.10) and (6.11) along with the Eq. (6.13).

5. If $k = 0$. There is a line of centers, given by any two of the following equations

$$\partial F/\partial x = 0,\ \partial F/\partial y = 0,\ \partial F/\partial z = 0.$$

Let any point (α, β, γ) on the line of centers be chosen as centre; then the reduced equation is

$$\lambda_1 x^2 + \lambda_2 y^2 = \mu$$

where $\mu = -(u\alpha + v\beta + w\gamma + d)$

The algorithm discussed in section 3.3 of Chapter III is used for plotting of contours. Except for steps (ii) and (iv), all steps are easy to understand. Detailed explanations are given for these steps in the following sections.

6.2.1 Finding the Minimum and the Maximum of *f(x, y,z)* over the Element

In general, the minimum and the maximum for a quadratic equation can be found by finding the point of extremum by taking the partial derivatives of the function given by Eq. (6.7) w.r.t. x, y and z and equating them to zero, *i.e.*

$$\frac{\partial f}{\partial x} = 2ax + 2gz + 2hy + 2u = ax + gz + hy + u = 0 \qquad (6.14a)$$

$$\frac{\partial f}{\partial y} = 2by + 2fz + 2hx + 2v = by + fz + hx + v = 0 \qquad (6.14b)$$

$$\frac{\partial f}{\partial z} = 2cz + 2fy + 2gx + 2w = cz + fy + gx + w = 0 \qquad (6.14c)$$

Let the solution of Eqs. (6.14a), (6.14b) and (6.14c) be (x_e, y_e, z_e) and the value of the function *f(x,y,z)* at (x_e, y_e, z_e) be f_e. Now we find f_{min} and f_{max} by finding the minimum

and the maximum of the values f_e and f_1, f_2 ….. *and* f_{10}. It can be noted here that we are simply interested here in deriving the extremum value at (x_e, y_e, z_e) notwithstanding the fact whether it is a point of the maximum or the minimum.

6.2.2 Tracing Contour Surface over an Element

The contour surface represented by Eq. (6.6) can have one of the different possible quadric surfaces. The contour surface is traced over the tetrahedral element by finding the possible intersection of the surface with the face of the tetrahedral (Schneider and Eberly, 2003). The face of the tetrahedral is taken as a plane. The intersection of the possible quadric surface with the plane is one of the conic section, as discussed below.

Let us consider an arbitrary plane given by Eq. (6.15a).

$$ax + by + cz + d = 0 \tag{6.15a}$$

where at least one of the coefficients *a*, *b*, *c* is distinct from zero.

Rearranging the Eq. (6.15a), we get

$$z = -\frac{ax + by + d}{c} \quad \text{with} \quad c \neq 0 \tag{6.15b}$$

The intersection equation of quadratic surface and the plane is obtained by substituting the value of *z* from Eq. (6.15b) into Eq.(6.6), which yields the equation of the form

$$ax^2 + 2hxy + by^2 + 2gx + 2fy + c = 0 \tag{6.16}$$

Eq. (6.16) represents the equation of general conics, discussed in detail in chapter IV.

The tracing of contours for some of the quadric surfaces are explained as follows:

i) Sphere: A sphere can be defined implicitly:

$$(x - C_x)^2 + (y - C_y)^2 + (z - C_z)^2 - r = 0 \tag{6.17}$$

where *C* is the center of the sphere and *r* is the radius. If a plane is defined parametrically as

$$\boldsymbol{P}(t_u, t_v) = p + t_u\vec{u} + t_v\vec{v} \tag{6.18}$$

Substitute Eq. (6.18) into Eq. (6.17) for the given equation of the form

$$At_u^2 + Bt_u t_v + Ct_v^2 + Dt_u + Et_v + F = 0 \tag{6.19}$$

where,

$$A = u_x^2 + u_y^2 + u_z^2$$

$$B = 2\left(u_x v_x + u_y v_y + u_z v_z\right)$$

$$C = v_x^2 + v_y^2 + v_z^2$$

$$D = 2\left(p_x u_x + p_y u_y + p_z u_z\right) - 2\left(C_x v_x + C_y v_y + C_z v_z\right)$$

$$E = 2\left(p_x v_x + p_y v_y + p_z v_z\right) - 2\left(C_x v_x + C_y v_y + C_z v_z\right)$$

$$F = C_x^2 + C_y^2 + C_z^2 + p_x^2 + p_y^2 + p_z^2 + 2\left(C_x P_x + C_y P_y + C_z P_z\right) - r$$

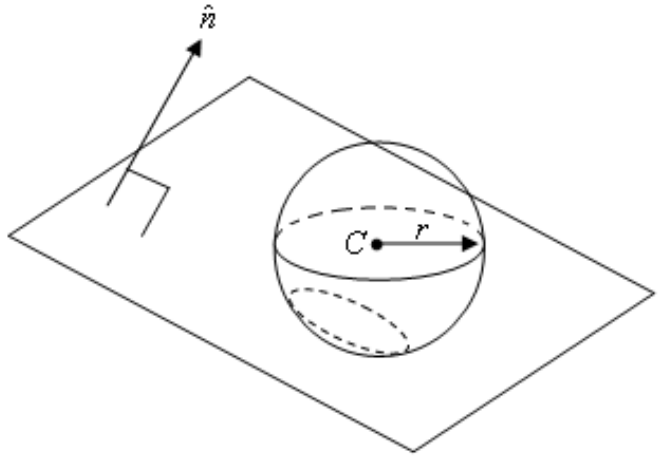


Figure 6.2 Intersection of a plane and a sphere

An alternative is to use a more direct geometric approach. Clearly, the intersection of a plane and a sphere, if it exists, is simply a circle lying in the plane as shown in the Fig. 6.2. The plane's equation provides part of a 3D circle specification, so we need to find the circle's centre and radius. The insight for the solution lies in observing that the centre *C* of the sphere is located at some distance along a line that passes through the centre of the circle *Q* of intersection and is normal to the plane ***P***, as can be seen more directly in cross section in Fig. 6.3. If plane ***P*** is given in the normalized coordinate-free version of the implicit form

$$P.\hat{n} + d = 0$$

then the distance between the plane and the centre of the sphere can be simply written as

$$b = \hat{n}.Q + d$$

If $|b| > r$, then there is no intersection; otherwise, the centre of the circle ***Q*** is simply

$$Q = C - b\hat{n}$$

Now all that remains is to determine the radius of the circle of intersection. From Fig. 6.3, it is see that

$$a^2 + b^2 = r^2 \quad \text{and so } a = \sqrt{r^2 - b^2} \text{ is the radius.}$$

If the plane just barely 'grazes' the sphere, then $r^2 - b^2$ will be a very small number; in this case, intersection is just a point.

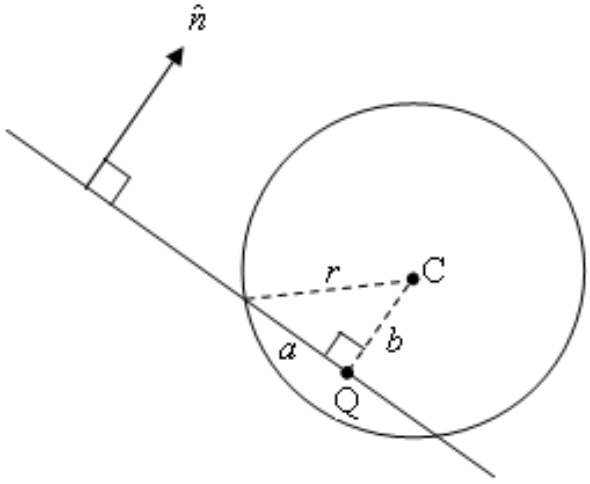
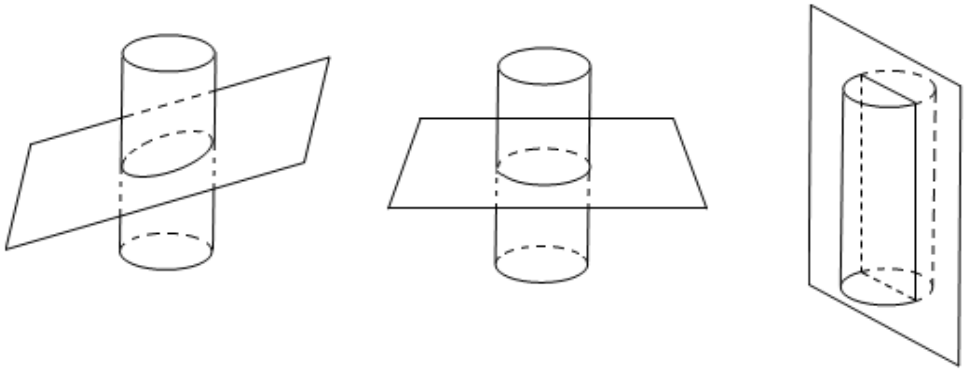
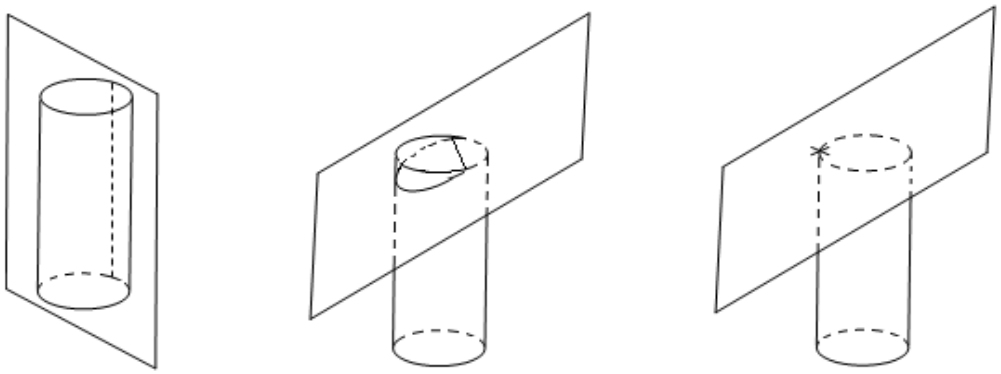


Figure 6.3 Cross-sectional view of sphere-plane intersection

ii) Cylinder: There are a number of ways a finite cylinder can intersect with a plane, six of which are shown in Fig. 6.4.



Ellipse **Circle** **Quadrilateral**



Line **Elliptical Arc / Line Segment** **Point**

Figure 6.4 Some of the ways a plane and a cylinder can intersect

For the intersection detection algorithm, a plane is defined implicitly

$$P.\hat{n} + d = 0 \tag{6.20}$$

For the purposes of detecting an intersection, a plane ***P*** and a cylinder C can be in one of several configurations:

1. ***P*** may be parallel to ***C****'s* axis: $\left|\hat{d}.\hat{n}\right| = 1$. Then there will be an intersection if the distance between ***P*** and ***C****'s* axis is less than or equal to the radius of the cylinder (in which case the intersection will be a quadrilateral or a single line, respectively).

2. ***P*** may be perpendicular to ***C****'s* axis: $\hat{d}.\hat{n} = 0$. Then there will be an intersection if the distance between ***P*** and ***C*** is less than or equal to the half-height of the cylinder (in which case the intersection will be a circle).

3. ***P*** may be neither parallel nor perpendicular to ***C****'s* axis. Then there are two cases to consider:

    a. The intersection of ***P*** and the axis of ***C*** is closer to the centrepoint of ***C*** than the half-height; in this case, there is definitely an intersection.

    b. ***P*** intersects the axis of ***C*** outside the end caps of the cylinder, in which case there may or may not be an intersection, depending on the relative location of the point of intersection and the angle between the plane and the axis.

In either case, the intersection will be either an ellipse, an elliptical arc and a straight line, or two elliptical arcs and two straight lines, depending on the relative orientation of the plane.

All of the cases but the last are fairly trivial. Determining whether or not the plane is parallel or perpendicular to the cylinder's axis involves only a dot product. Computing the distance between the cylinder's centre point *C* and the plane is simple and inexpensive. Computing the intersection of the plane and the cylinder's axis is also inexpensive.

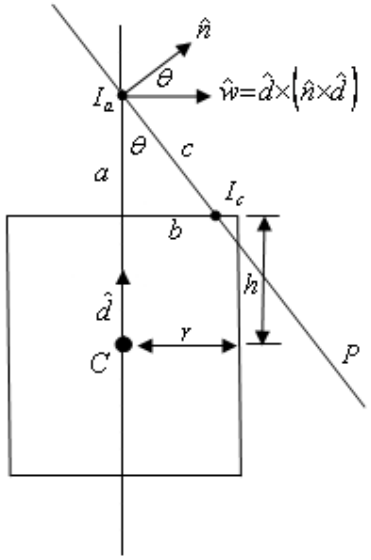


Figure 6.5 Edge-on view of plane-cylinder intersection

The last case is illustrated in Fig. 6.5. The edge-on view is not simply a diagrammatic convenience -the method for determining whether or not an intersection exists is done in a plane perpendicular to ***P*** and going through *C*. The rest is basic trigonometry.

If I_a is the intersection of ***P*** and the axis of the cylinder, then if the point I_c is closer to the axis than *r* (the radius of the cylinder), we have an intersection. The plane ***P***' perpendicular to ***P*** and going through *C* is parallel to $\hat{d}$, and so its normal is '$\hat{n}\times\hat{d}$'. A vector in ***P***' that is perpendicular to $\hat{d}$:

$$\hat{w}=\hat{d}\times\left(\hat{n}\times\hat{d}\right) \tag{6.21}$$

The angle θ between $\hat{n}$ and $\hat{w}$ is given by Eq. (6.22).

$$\cos(\theta)=\hat{n}.\hat{w} \tag{6.22}$$

The distance *a* is given by Eq. (6.23).

$$a = \|I_a - C\| - h \tag{6.23}$$

By the definition of the cosine function, it is known that

$$\cos(\theta) = a / c \tag{6.24}$$

Substituting the values from Eq. (6.22) and (6.23) into Eq. (6.24), we get

$$\hat{n}.\hat{w} = \frac{\|I_a - C\| - h}{c} \text{ and so } c = \frac{\|I_a - C\| - h}{\hat{n}.\hat{w}} \tag{6.25}$$

Invoking the Pythagorean Theorem,

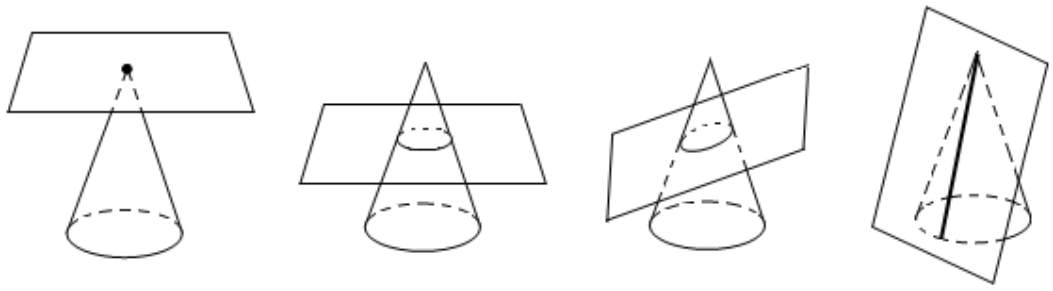
$$a^2 + b^2 = c^2$$

$$\left(\|I_a - C\| - h\right)^2 + b^2 = \left(\frac{\|I_a - C\| - h}{\hat{n}.\hat{w}}\right)^2$$

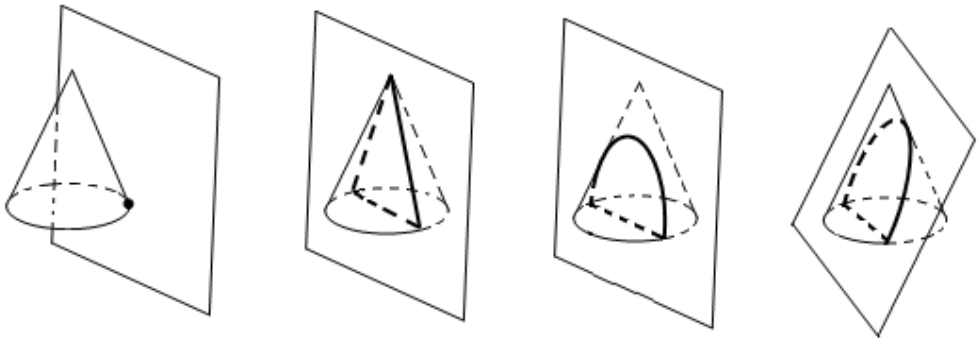
$$b^2 = \left(\frac{\|I_a - C\| - h}{\hat{n}.\hat{w}}\right)^2 - \left(\|I_a - C\| - h\right)^2 \tag{6.26}$$

And so if $b^2 \le r^2$, we have an intersection; otherwise, not.

(iii) Cone: There are actually quite a number of ways a cone and a plane can intersect, eight of which are shown in Fig. 6.6.



Point **Circle** **Ellipse** **Straight Line**



Point (finite cone only) **Lines** **Hyperbola** **Parabola**

Figure 6.6 Some of the ways a plane and a cone can intersect

For the intersection detection algorithm, a plane is defined implicitly by Eq. (6.20). The finite single cone is defined in 'general position' – as a base point *B*, axis $\hat{d}$, and height *h*.

For the purposes of detecting an intersection, a plane ***P*** and a cone ***C*** can be in one of several configurations:

1. ***P*** may be parallel to ***C****'s* axis: $\left|\hat{d}.\hat{n}\right| = 1$. Then there will be an intersection if the distance between ***P*** and ***C****'s* axis is less than or equal to the radius of the cone.
2. ***P*** may be perpendicular to ***C****'s* axis: $\hat{d}.\hat{n} = 0$. Then there will be an intersection if the signed distance (relative to $\hat{d}$) from *B* to ***P*** is between 0 and *h*.
3. If ***P*** is neither parallel nor perpendicular to ***C****'s* axis, then there are two cases to consider:
    a. The signed distance (relative to $\hat{d}$) from *B* to ***P*** is between 0 and *h*; in this case, there is definitely an intersection.
    b. ***P*** intersects the axis of ***C*** outside the apex or end cap of the cone, in this case, there may or may not be an intersection, depending on the relative location of the point of intersection and the angle between the plane and the axis.

All of the cases but the last are fairly trivial. Determining whether or not the plane is parallel or perpendicular to the cone's axis involves only a dot product. Computing the distance between the cone's base point *B* and the plane, is simple and inexpensive. Computing the intersection of the plane and the cone's axis is inexpensive as well.

The last case is illustrated in Fig. 6.8. The edge-on view is not simply a diagrammatic convenience – the method for determining whether or not an intersection exists is done in a plane perpendicular to ***P*** and going through *B*. The rest is basic trigonometry.

If I_a is the intersection of ***P*** and the axis of the cone, then if the point I_c is closer to the axis than *r* (the radius of the cone), there is an intersection. The plane ***P'*** perpendicular to ***P*** and going through *B* is parallel to $\hat{d}$, and so its normal is $\hat{n} \times \hat{d}$

We define a vector in $\boldsymbol{P}'$ that is perpendicular to $\hat{d}$:

$$\hat{w} = \hat{d} \times \left(\hat{n} \times \hat{d} \right) \tag{6.27}$$

The angle θ between $\hat{n}$ and $\hat{w}$ is given by Eq. (6.28).

$$\cos(\theta) = \hat{n}.\hat{w} \tag{6.28}$$

The distance *a* is given by Eq. (6.29).

$$a = \left\| I_a - B \right\| - h \tag{6.29}$$

By the definition of the cosine function,

$$\cos(\theta) = a / c \tag{6.30}$$

Substituting, the values from Eq. (6.28) and (6.29) into Eq. (6.30),

$$\hat{n}.\hat{w} = \frac{\left\| I_a - B \right\| - h}{c} \text{ and so } c = \frac{\left\| I_a - B \right\| - h}{\hat{n}.\hat{w}} \tag{6.31}$$

Invoking the Pythagorean Theorem, the following is obtained,

$$a^2 + b^2 = c^2$$

$$\left(\left\| I_a - B \right\| - h \right)^2 + b^2 = \left(\frac{\left\| I_a - B \right\| - h}{\hat{n}.\hat{w}} \right)^2$$

$$b^2 = \left(\frac{\left\| I_a - B \right\| - h}{\hat{n}.\hat{w}} \right)^2 - \left(\left\| I_a - B \right\| - h \right)^2 \tag{6.32}$$

And so if $b^2 \le r^2$, there is an intersection.

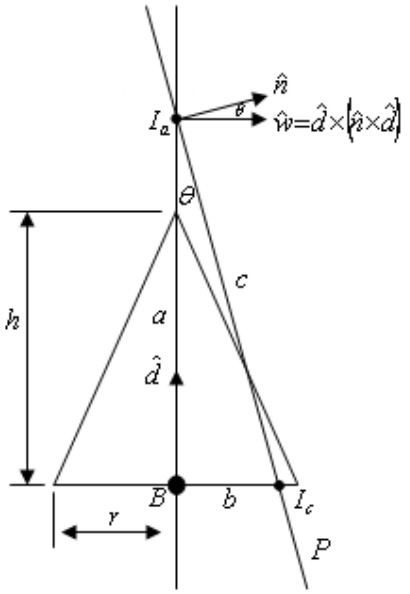


Figure 6.7 Edge-on view of plane-cone intersection

Similarly the intersections of all the other quadric surfaces with the plane are defined.

The next task is to plot the contour surfaces of same level in order to make a complete contour surface from one end to another. The plotting of the contour surface is done through rasterization. This provides a fast way of contour surfaces over a 3D domain.

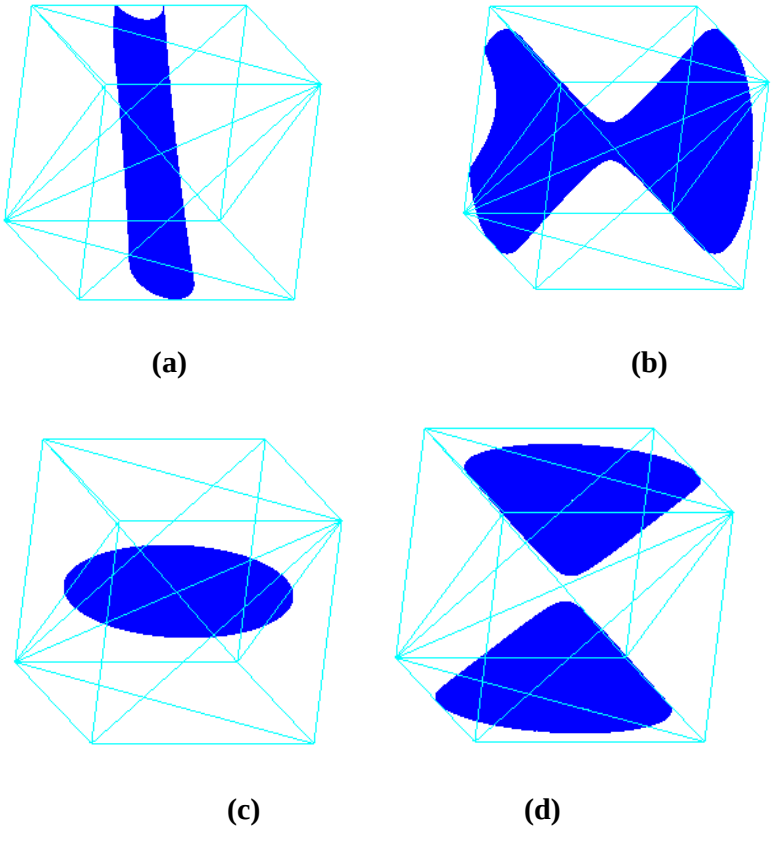
6.3 IMPLEMENTATION OF THE ALGORITHM AND NUMERICAL RESULTS

The contour plot algorithm, discussed in section 6.2, is implemented in Turbo C++ and runs on an IBM compatible PC with 3.0 GHz CPU and 256 RAM under Windows operating system. The OpenGL graphics libraries are used to display the geometry and contour lines for different components.

The first problem is a pure synthetic problem which represents some of the quadric surfaces for the cases represented by Eq. (6.6). We consider a cube domain $[-2,2]\times[-2,2]\times[-2,2]$ in cartesian system which is divided into number of ten-node tetrahedral elements and find some of the contour surfaces for various values of the coefficients, l_0 through l_9, as given in Table 6.3. Fig. 6.8 demonstrates that the contour surfaces are more accurate as compared to four-node tetrahedral elements where the contour surfaces can only be the linear planes.

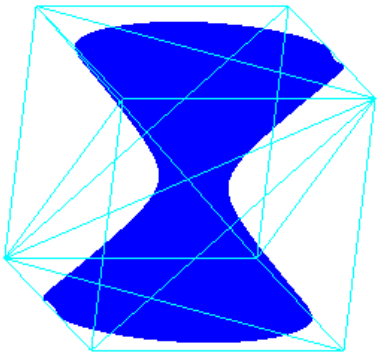
Quadric Surface	l_0	l_1	l_2	l_3	l_4	l_5	l_6	l_7	l_8	l_9
Cylinder	1.5	-10	-1	-2	-3	-4	-5	-2	-3	-3
Hyperbolic Paraboloid	0	1	-1	-1	0	0	-1	0	0	0
Ellipsoid	-1	0.11	1	0.25	0	0	0	0	0	0
Hyperboloid of Two Sheets	0.25	1	-1	2	0	0	-1	0	0	0
Hyperboloid of One Sheets	-0.5	1	-1	2	0	0	-1	0	0	0

Table 6.3 Values of coefficients for different contours shown in Fig. 6.8



(a) **(b)**

(c) **(d)**



(e)

Figure 6.8 Contours over the cube domains: (a) Cylinder; (b) Hyperbolic Paraboloid; (c) Ellipsoid; (d) Hyperboloid of Two Sheets; (e) Hyperboloid of One Sheets

Further, the problems considered here are the deflection and thermal problems of the different components.

The first problem is a deflection of a 3D hand wheel with specification shown in Fig. 6.9. The wheel is fixed at one end and a load is applied at the handle, shown in Fig. 6.10. For a load of 1000 N, Young's Modulus $0.2\times10^{9}\,MPa$, Poisson's ratio 0.3, the values of deflections are derived using FEA, considering it as a 3D problem using ten-node tetrahedral elements.

For the given loading conditions, the deflections are due to the bending moment which is given by Eqs. (3.26) to (3.28).

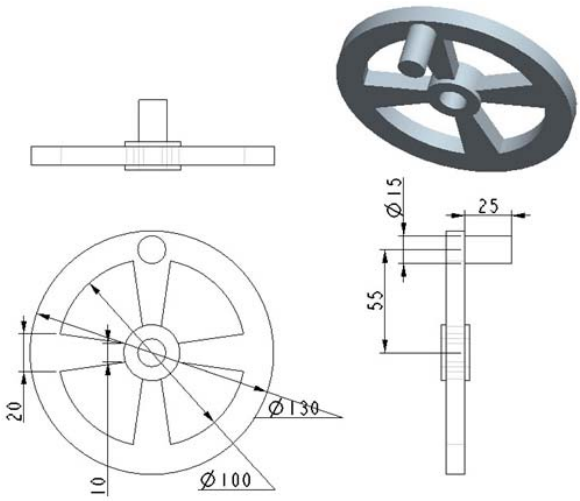


Figure 6.9 Specifications of the hand wheel

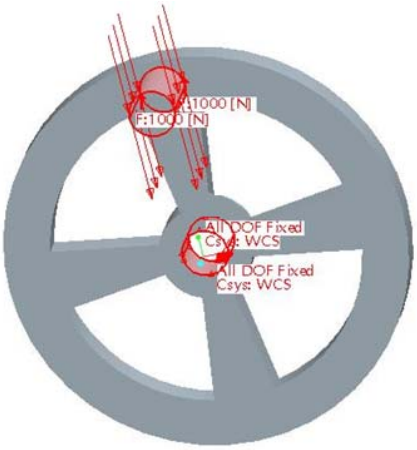
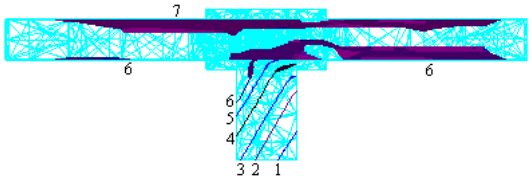


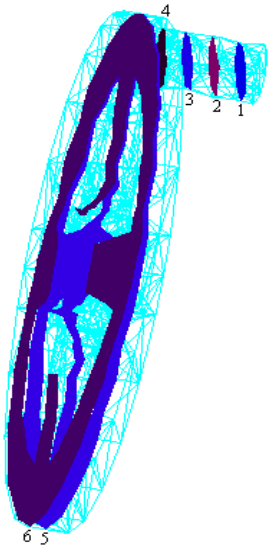
Figure 6.10 Boundary conditions and loading of the hand wheel

Fig. 6.11(a), 6.11(b) and 6.11(c) shows the contour plots of deflections in *x*, *y* and *z*-directions using ten-node tetrahedral elements for different levels of deflection.



Contour Levels	C = δ_x
1	-70 x10^{-6}
2	-50 x10^{-6}
3	-38 x10^{-6}
4	-22 x10^{-6}
5	-12 x10^{-6}
6	-5 x10^{-6}
7	5 x10^{-6}

Figure 6.11(a) Contour surfaces of deflection in x-direction using ten-node tetrahedral elements for the different deflection levels for the hand wheel



Contour Levels	C = δ_y
1	-4.0 x10^{-4}
2	-3.0 x10^{-4}
3	-2.0 x10^{-4}
4	-1.0 x10^{-4}
5	0.01 x10^{-4}
6	0.1 x10^{-4}

Figure 6.11(b) Contour surfaces of deflection in y-direction using ten-node tetrahedral elements for the different deflection levels for the hand wheel

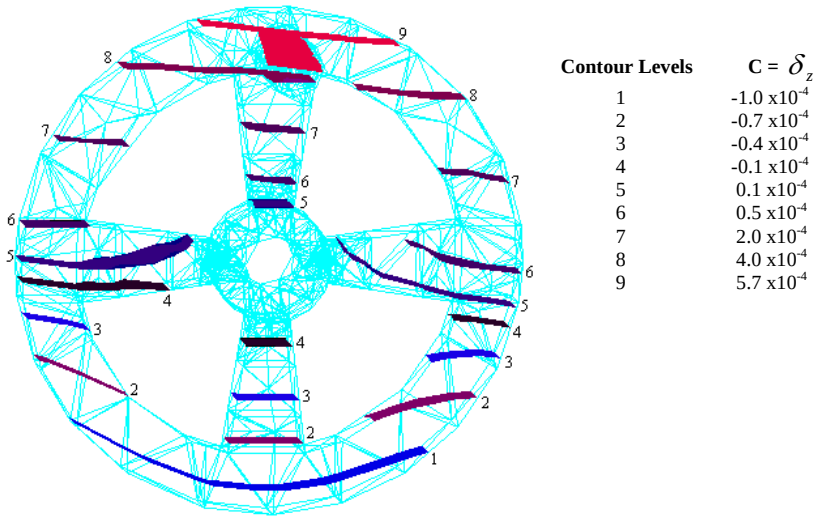


Figure 6.11(c) Contour surfaces of deflection in z-direction using ten-node tetrahedral elements for the different deflection levels for the hand wheel

Fig. 6.12(a) shows the specification of a component. The component is fixed and loaded as shown in Fig. 6.12(b). For a load of 1000 N, Young's Modulus $0.2\times10^{9}\ MPa$, Poisson's ratio 0.3, the values of deflections are derived using FEA, considering it as a 3D problem. For the given loading conditions, the deflections are due to the combined effect of direct force and bending moments which are given by Eqs. (3.25) to (3.28).

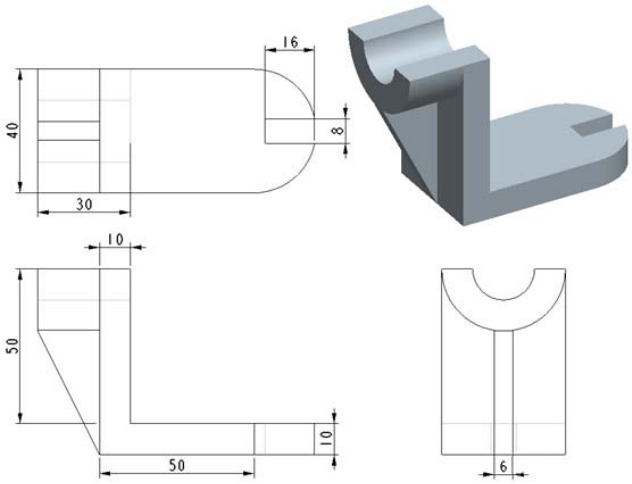


Figure 6.12(a) Specifications of the component

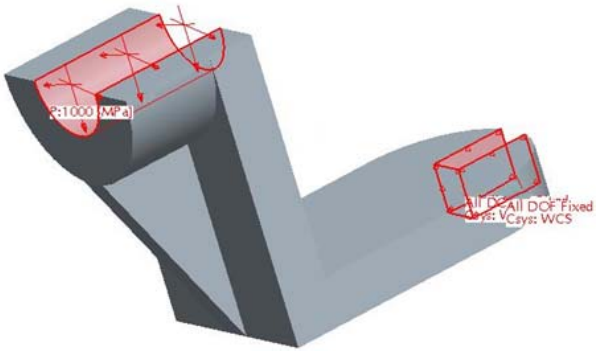
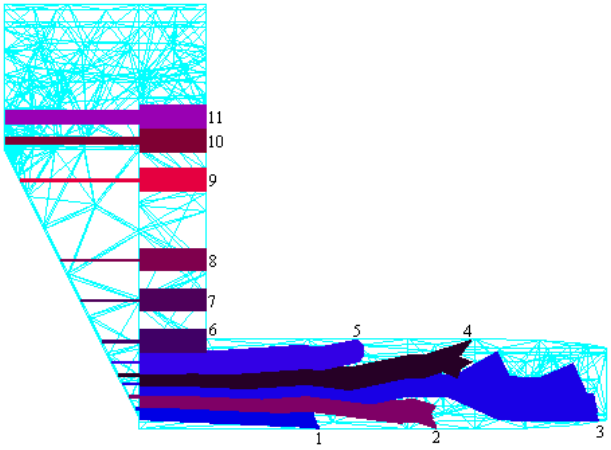


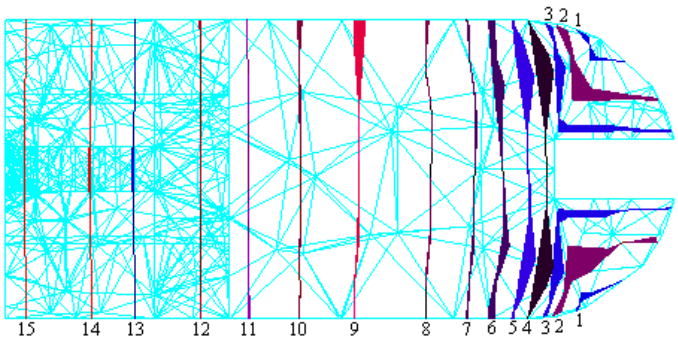
Figure 6.12(b) Boundary conditions and loading of the component

Fig. 6.13(a) and 6.13(b) shows the contour plots of deflections in *x* and *y*-directions using ten-node tetrahedral elements for different levels of deflections.



Contour Levels	C = δ_x
1	-60 x10^{-4}
2	-30 x10^{-4}
3	-1 x10^{-4}
4	20 x10^{-4}
5	50 x10^{-4}
6	100 x10^{-4}
7	200 x10^{-4}
8	300 x10^{-4}
9	500 x10^{-4}
10	600 x10^{-4}
11	661 x10^{-4}

Figure 6.13(a) Contour surfaces of deflection in x-direction using ten-node tetrahedral elements for the different deflection levels for the component



Contour Levels	C = δ_y
1	-10 x10^{-4}
2	-5 x10^{-4}
3	-1 x10^{-4}
4	5 x10^{-4}
5	15 x10^{-4}
6	30 x10^{-4}
7	50 x10^{-4}
8	100 x10^{-4}
9	200 x10^{-4}
10	300 x10^{-4}
11	400 x10^{-4}
12	500 x10^{-4}
13	650 x10^{-4}
14	750 x10^{-4}
15	900 x10^{-4}

Figure 6.13(b) Contour surfaces of deflection in y-direction using ten-node tetrahedral elements for the different deflection levels for the component

Fig. 6.14(a) shows the specification of the bearing holder. The bearing holder is fixed and loaded as shown in Fig. 6.14(b). For a load of 1000 N, Young's Modulus 0.2×10^{9} *MPa*, Poisson's ratio 0.3, the values of deflections are derived using FEA, considering it as a 3D problem. For the given loading conditions, the deflections are due to the combined effect of direct force and bending moments which are given by Eqs. (3.25) to (3.28).

Fig. 6.15(a) and 6.15(b) shows the contour plots of deflections in *x* and *y* directions using ten-node tetrahedral elements for different levels of deflections.

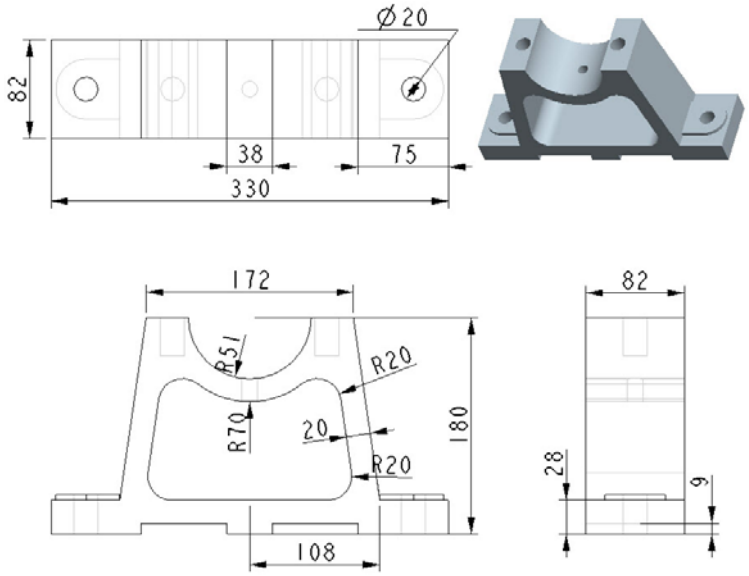


Figure 6.14(a) Specifications of the bearing holder

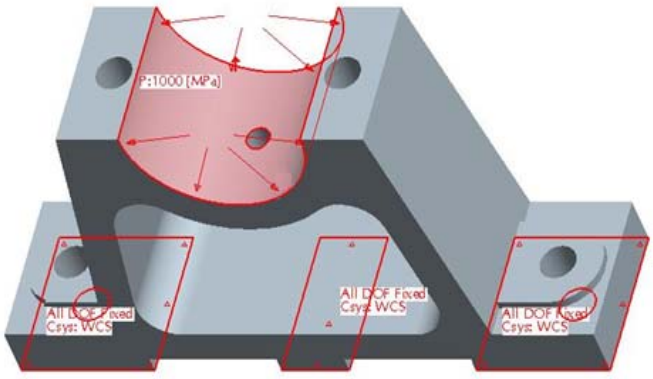
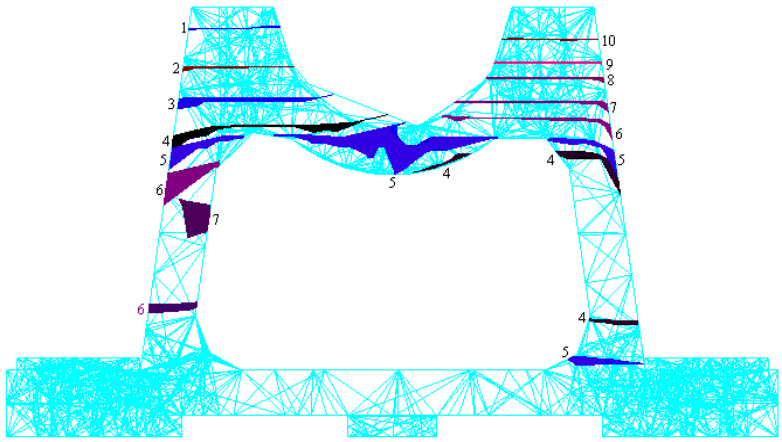
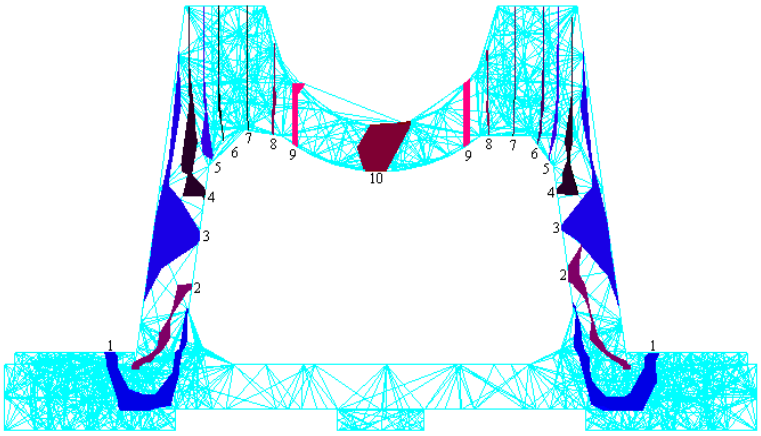


Figure 6.14(b) Boundary conditions and loading of the bearing holder



Contour Levels	C = δ_x
1	-70 x10^{-4}
2	-45 x10^{-4}
3	-25 x10^{-4}
4	-8 x10^{-4}
5	-2 x10^{-4}
6	10 x10^{-4}
7	20 x10^{-4}
8	35 x10^{-4}
9	45 x10^{-4}
10	60 x10^{-4}

Figure 6.15(a) Contour surfaces of deflection in x-direction using ten-node tetrahedral elements for the different deflection levels for the bearing holder



Contour Levels	C = δ_y
1	1 x10^{-4}
2	3 x10^{-4}
3	7.5 x10^{-4}
4	15 x10^{-4}
5	25 x10^{-4}
6	35 x10^{-4}
7	55 x10^{-4}
8	75 x10^{-4}
9	90 x10^{-4}
10	122 x10^{-4}

Figure 6.15(b) Contour surfaces of deflection in y-direction using ten-node tetrahedral elements for the different deflection levels for the bearing holder

Fig. 6.16(a) shows the specification of chair. The chair is fixed at the base and vertically loaded as shown in Fig. 6.16(b). For a load of 1000 N, Young's Modulus $0.2\times10^{9}\,MPa$, Poisson's ratio 0.3, the values of deflection are derived using FEA, considering it as a 3D problem. For the given loading conditions, the deflections are due to the direct force which is given by Eq. (3.25).

Fig. 6.17(a) and 6.17(b) shows the contour plots of deflections in *y* and *z*- directions using ten-node tetrahedral elements for different levels of deflection.

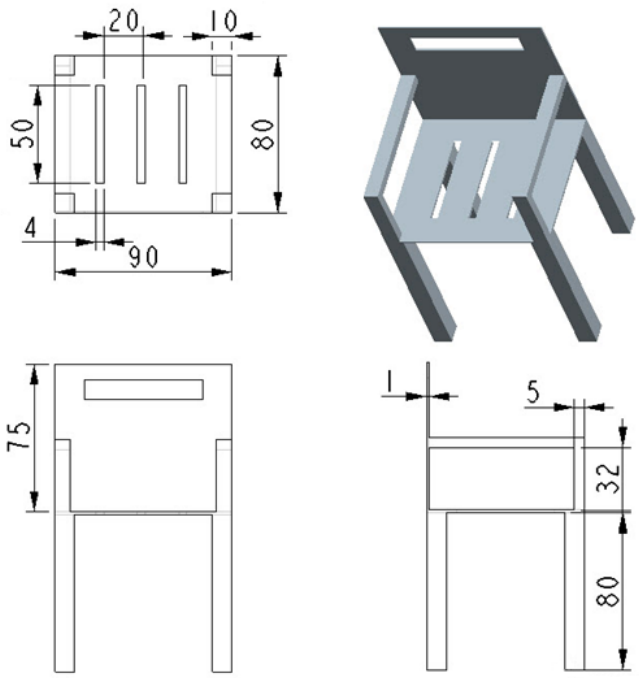


Figure 6.16(a) Specifications of the chair

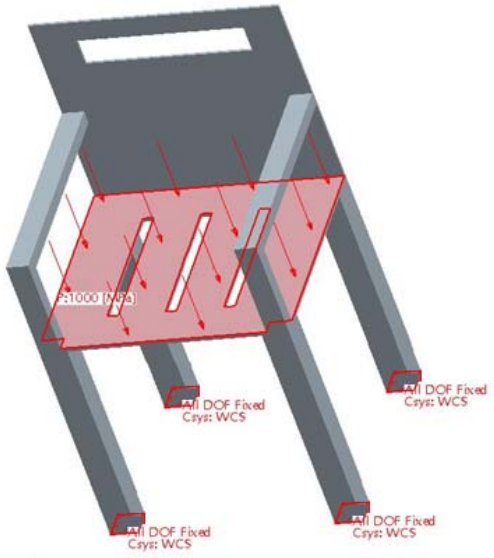
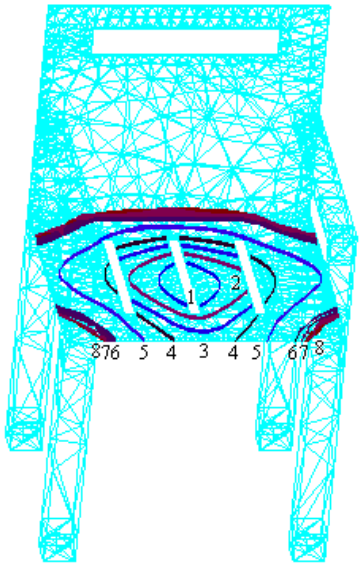
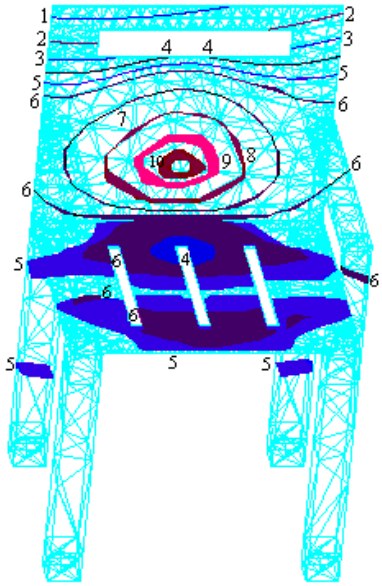


Figure 6.16(b) Boundary conditions and loading of the chair



Contour Levels	C = δ_y
1	-220 x10^{-2}
2	-180 x10^{-2}
3	-140 x10^{-2}
4	-100 x10^{-2}
5	-50 x10^{-2}
6	-10 x10^{-2}
7	-6 x10^{-2}
8	-3 x10^{-2}

Figure 6.17(a) Contour surfaces of deflection in y-direction using ten-node tetrahedral elements for the different deflection levels for the chair



Contour Levels	C = δ_z
1	-27 x10^{-2}
2	-17 x10^{-2}
3	-10 x10^{-2}
4	-5 x10^{-2}
5	-1 x10^{-2}
6	3 x10^{-2}
7	13 x10^{-2}
8	23 x10^{-2}
9	30 x10^{-2}
10	33.5 x10^{-2}

Figure 6.17(b) Contour surfaces of deflection in z-direction using ten-node tetrahedral elements for the different deflection levels for the chair

The next problem is a heat conduction problem, with the specifications shown in Fig. 6.18(a). The temperature of 673^0C and 273^0C is fixed at the base and top of the pan, as shown in Fig. 6.18(b). For the known heat flow rate, thermal conductivity of the material as 0.20800 W/m-deg and the given boundary temperatures, the values of temperature distribution is derived using FEA. For the given conditions, the temperature distribution is given by Fourier Equation (Kumar, 2008).

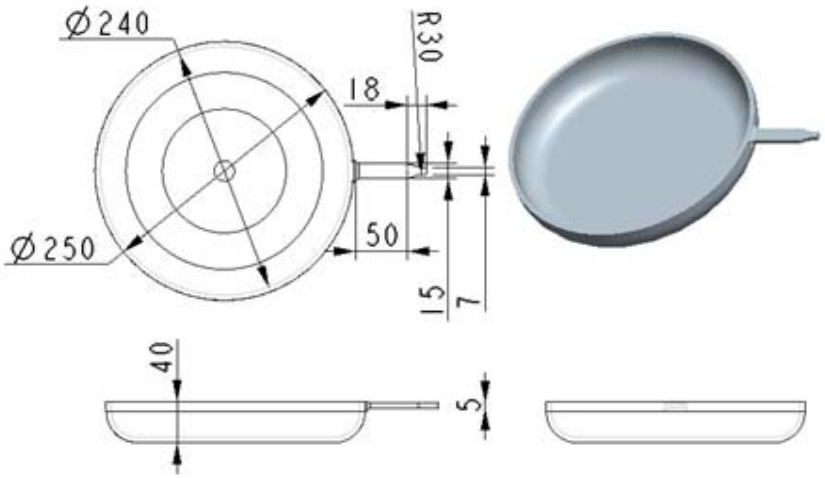
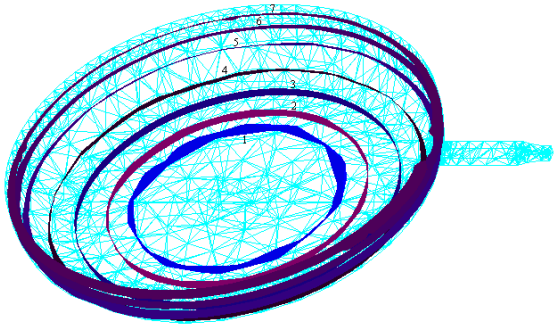


Figure 6.18(a) Specifications of the pan

Figure 6.18(b) Boundary conditions and temperature fixation of the pan

Fig. 6.19 shows the contour plots of temperature distribution using ten-node tetrahedral elements for the different levels of temperature.



Contour Levels	C = *T*
1	630
2	550
3	450
4	380
5	320
6	280
7	260

Figure 6.19 Contour plot of temperature distribution (in ^{o}C) using ten-node tetrahedral elements for the pan

Fig. 6.20(a) shows the specification of the boiler shell for the heat conduction problem. The temperature of 673^{0}C and 273^{0}C is fixed at the inside and outside of the shell, as shown in Fig. 6.20(b). For the known heat flow rate, thermal conductivity of the materials as 0.20800 W/m-deg and 0.12800 W/m-deg, and the given boundary temperatures, the values of temperature distribution is derived using FEA. For the given conditions, the temperature distribution is given by Eq. (3.31).

Fig. 6.21 shows the front and top view of contour plots of temperature distribution using ten-node tetrahedral elements for the different levels of temperature.

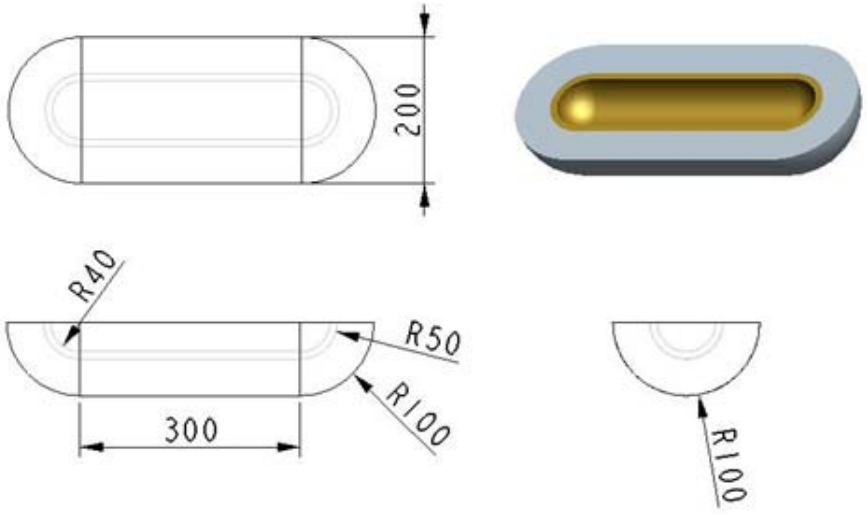


Figure 6.20(a) Specifications of the shell

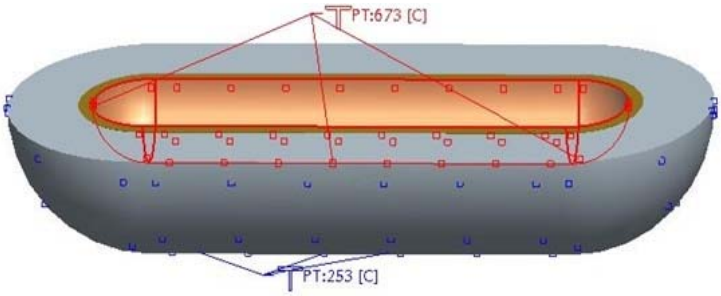


Figure 6.20(b) Boundary conditions and temperature fixation of the shell

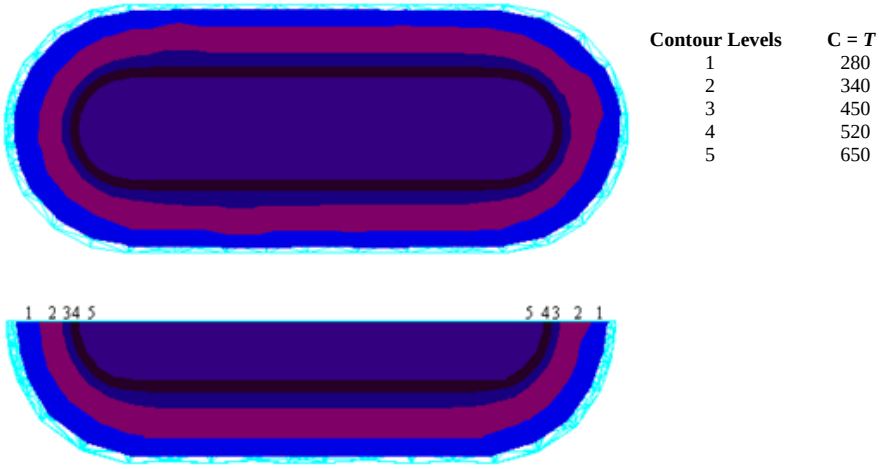


Figure 6.21 Front and top view of contour plot of temperature distribution (in ^{o}C) using ten-node tetrahedral elements for the shell

The contour surfaces generated in the different problems discussed above shows that the contour equation developed using shape functions can be used to plot the surfaces. The algorithm takes less than 4 seconds for the generation and plotting of one contour surface for 825 ten-node tetrahedral elements on a P-IV PC with 3.0 GHz CPU and 256 MB RAM.

6.4 CONCLUDING REMARKS

Higher order elements are mainly used for better results. These results are also to be accurately plotted to conceive the information generated by these results. To simplify the contour plotting, generally the higher order elements are splited into number of linear elements. To increase the accuracy of plotting, the contour equation is developed using shape functions for ten-node tetrahedral element. The developed contour equation represents the quadratic surfaces or quadrics, which are traced as contour surfaces.

CHAPTER - VII

CONCLUSIONS AND FUTURE SCOPE

7.1 GENERAL

This chapter covers the summary of research work and presents the conclusions. It also discusses the scope for further research.

The main conclusions of the present research work have been derived from the numerical results presented by implementing the developed algorithms in the different chapters.

7.2 CONCLUSIONS

The following conclusions have been drawn:

- A simple and fast method for contour plotting is developed using three-node triangular element in 2D.
- An accurate and fast algorithm for contour plotting using four-node quadrilateral element is developed. The contour equation is developed using shape functions, which generates the contours as accurate as the FEA results. The algorithm for the joining of the contour segments is very useful for faster displays.
- Accurate contour plotting algorithm using six-node triangular elements is developed. The quadratic interpolation function representing contours is degenerated into various conic sections for accurate and fast display. It is observed that the contouring algorithm is very fast and takes less than 0.17 seconds for the different shown cases in the problem involving 132 six-node elements. The accuracy is demonstrated by comparing contour lines derived with three-node elements. The technique developed in this paper can be used to plot contours accurately for applications where high accuracy is required.
- A fast algorithm for the plotting of contours using eight-node quadrilateral elements is described. The contours are accurately generated over each element using the interpolation functions. The exactness of the contours matches that of the finite element analysis. The developed contour joining algorithm makes the technique faster than the existing techniques.

- The contour surfaces in 3D are developed to show the variation of the physical parameter within the considered domain as against the techniques which show the variation only on the surface.
- A simple and fast algorithm for contour plotting is developed using four-node tetrahedral element in 3D.
- Fast and accurate algorithms for the generation of contour surfaces in 3D are described using hexahedral elements. The algorithm provides contour surfaces in the form of polygons whose sides are curves in 3D. The method for the generation of interior points of contour surfaces is given which provides their accurate locations. However, the contour surfaces are C^0 continuous at interelement boundaries.
- An accurate algorithm for plotting contour surfaces in 3D using ten-node tetrahedral elements is presented. The interpolation function representing contours is degenerated into various quadric surfaces for ten-node tetrahedral elements.

The main drawback of the interpolation formulation used in the development of contour equations in all the above algorithms is that as the order of the element increases, the complexities in terms of computations are increased.

7.3 SCOPE FOR FUTURE WORK

While carrying out the present study, a number of areas have come to focus, where detailed research can be taken up. These areas demand more exploration and analysis through further research. The scope of future work may be as follows:

- The algorithm for contour plotting can be developed over twenty-node hexahedral element.
- Perspective projections can be taken into account for displaying the contour surfaces.
- Hidden line or surface algorithm can be implemented to show the variation only on the outer surface of the domain.

LIST OF RESEARCH PUBLICATIONS

Publications in Journals	
1.	Chandan Singh, **Jaswinder Singh**, Accurate Contour Plotting Using 6-Node Triangular Elements in 2D, *Finite Elements in Analysis and Design*, Vol. 45, No. 2, Jan 2009, pp. 81-93. **(Impact Factor: 1.030)**.
2.	Chandan Singh, **Jaswinder Singh**, A Simple and Fast Contour Plotting Algorithm for Linear 2D and 3D Elements, *International Journal of Engineering Science and Technology*, Vol. 3, No. 4, April 2011, pp. 2796-2802.
3.	Chandan Singh, **Jaswinder Singh**, Accurate and Fast Algorithm for the Plotting of Contours using Eight-Node Quadrilateral Meshes, *Journal of Brazilian Society of Mechanical Science and Engineering*. **(First revision over)**
4.	Chandan Singh, **Jaswinder Singh**, Algorithms for Accurate and Fast Plotting of Contour Surfaces in 3D using Hexahedral Elements, *The Arabian Journal of Science and Engineering*. **(First revision over)**
5.	Chandan Singh, **Jaswinder Singh**, Algorithm for Accurate Plotting of Contour Surfaces in 3D Using Tetrahedral Elements, *Iranian Journal of Science and Technology Transaction B: Engineering*. **(Communicated)**
Research Project Completed	
1.	"Development of Algorithms and Software Packages for 2D and 3D Contour Plotting Using Non-Linear Interpolation", University Grants Commission (UGC), Government of India, New Delhi, vide UGC Reference No. F. 30-257/2004.

REFERENCES

[1] Russell W. Stineman, Plotting a Function of Three Independent Variables, *Communications of the ACM*, Vol.10, No.7, pp. 425-428, 1967.

[2] Stephen P. Morse, Concepts of Use in Contour Map Processing, *Communications of the ACM*, Vol. 12, No. 3, pp. 147-152, 1969.

[3] D. H. McLain, Drawing Contours from Arbitrary Data Points, *The Computer Journal*, Vol. 17, No. 4, pp. 318-324, 1974.

[4] Hiroshi Akima, A method of Bivariate Interpolation and Smooth Surface Fitting Based on Local Procedures, *Communications of the ACM*, Vol. 17, No. 1, pp. 18-20, 1974.

[5] E. Keppel, Approximating Complex Surfaces by Triangulation of Contour Lines. *IBM Journal of Research Development*, Vol. 19, No. 1, pp. 2-11, 1975.

[6] D. C. Sutcliffe, An Algorithm for Drawing the Curve $f(x, y) = 0$, *The Computer Journal*, Vol. 19, No. 3, pp. 246-249, 1976 a.

[7] D. C. Sutcliffe, A Remark on a Contouring Algorithm, *The Computer Journal*, Vol. 19, No. 4, pp. 333-335, 1976 b.

[8] J. L. Meek and G. Beer, Contour Plotting of Data using Isoparametric Element Representation, *International Journal for Numerical Methods in Engineering*, Vol. 10, pp. 954-957, 1976.

[9] S. Marlow and M. J. D. Powell, A Fortran Subroutine for Plotting the Part of a Conic that is inside a given Triangle, A Report to United Kingdom Atomic Energy Authority, 1976.

[10] J. E. Akin and W. H. Gray, Contouring on Isoparametric Surfaces, *International Journal for Numerical Methods in Engineering*, Vol.11, pp. 1893-1897, 1977.

[11] C. M. Gold, T. D. Charters and J. Ramsden, Automated Contour Mapping using Triangular Element Data Structures and an Interpolant over Each Irregular Triangular Domain, Siggraph '77, July 20-22, San Jose, California, pp. 170-175, 1977.

[12] M. J. D. Powell and M. A. Sabin, Piecewise Quadratic Approximations on Triangles, *ACM Transactions on Mathematical Software*, Vol. 3, No. 4, pp. 316-325, 1977.

[13] William V. Snyder, Contour Plotting, *ACM Transactions on Mathematical Software*, Vol. 4, No. 3, pp. 290-294, 1978.

[14] Thomas Wright and John Humbrecht, ISOSRF - An Algorithm for Plotting Iso-Valued Surfaces of a Function of Three Variables, ACM SIGGRAPH Computer Graphics, Vol. 13, No. 2, pp. 182-189, 1979.

[15] D. H. Faber, E. W. M. Rutten-Keulemans and C. Altona, Computer Plotting of Contour Maps: An Improved Method, *Computers and Chemistry*, Vol. 3, pp. 51-55, 1979.

[16] W. H. Gray and J. E. Akin, An improved method for contouring on isoparametric surfaces, *International Journal for Numerical Methods in Engineering*, Vol. 14, No. 3, pp. 451-472, 1979.

[17] Robin Sibson and Graeme D. Thomson, A Seamed Quadratic Element for Contouring, *The Computer Journal*, Vol. 24, No. 4, pp. 378-382, 1981.

[18] S. Ganapathy and T. G. Dennehy, A New General Triangulation Method For Planar Contours, *ACM Transactions on Computer Graphics*, Vol. 16, No. 3, pp. 69-75, 1982.

[19] J. F. Lyness and L. Asquith, A Simple Contour Plotting Program for Finite Element Output, *Advances in Engineering Software*, Vol. 5, No. 1, pp. 23-31, 1983.

[20] George D. Kontopidis and David E. Limbert, A Predictor-Corrector Contouring Algorithm for Isoparametric 3D Elements, *International Journal for Numerical Methods in Engineering*, Vol. 19, pp. 995-1004, 1983.

[21] S. Antoy, Contour Plotting for Function Specified at Nodal Points of a Mesh Based on a set of Irregular Profiles, *Computers and Geosciences*, Vol. 9, No. 2, pp. 235-244, 1983.

[22] S. Rajasekaran, A Note on Plotting Curves, *Computers and Structures*, Vol. 19. No. 3, pp. 491-500, 1984.

[23] Albrecht Preusser, Computing Contours by Successive Solution of Quintic Polynomial Equations, *ACM Transactions on Mathematical Software*, Vol. 10, No. 4, pp. 463-472, 1984.

[24] G.P. Steven, Use of Minimum Surface Theory to Aid Finite Element Contour Plotting, *International Journal for Numerical Methods in Engineering*, Vol. 20, pp. 1791-1796, 1984.

[25] J. E. Akin, W. H. Gray and Q. D. Zhang, Colouring Isoparametric Contours, *Engineering Computations*, Vol. 1, pp. 36-41, 1984.

[26] M. E. Yeo, An Iteractive Contour Plotting Program, *Engineering Computations*, Vol. 1, pp. 273-279, 1984.

[27] L. S. Chen, G. T. Herman, R. A. Reynolds and J. K. Udupa, Surface Shading in the Cuberille Environment, *Computer Graphics and Applications*, Vol. 10, pp. 33-43, 1985.

[28] G. Farin, Triangular Bernstein-Bezier Patches, *Computer Aided Geometric Design*, Vol. 3, pp. 83-128, 1986.

[29] A. Preusser, Computing Area Filling Contours for Surfaces Defined by Piecewise Polynomials, *Computer Aided Geometric Design*, Vol. 3, No. 4, pp. 267-279, 1986.

[30] M.J. Zyda, A.R. Jones and P.G. Hogan, Surface Construction from Planar Contours, *Computers and Graphics*, Vol.11, pp. 393-408, 1987.

[31] J. F. Stelzer and R. Welzel, Plotting of contours in the natural way, *International Journal for Numerical Methods in Engineering*, Vol. 24, pp. 1757-1769, 1987.

[32] S.R. Yates, Contur: A Fortran Algorithm for Two-Dimensional High Quality Contouring, *Computers and Geosciences*, Vol. 13, No. 1, pp. 61-76, 1987.

[33] W. E. Lorensen and H. E. Cline, Marching Cubes: A High Resolution 3D Surface Construction Algorithm, *Computer Graphics*, Vol. 21, No. 4, pp. 163-169, 1987.

[34] J. D. Boissonnat, Shape Reconstruction from Planar Cross-Sections, Computer Vision, *Graphics, and Image Processing*, Vol. 44, pp. 1-29, 1988.

[35] Michael J. Zyda, A Decomposable Algorithm for Contour Surface Display Generation, *ACM Transactions on Graphics*, Vol. 7, No. 2, pp. 129-148, 1988.

[36] Granville Sewell, Plotting Contour Surfaces of a Function of Three Variables, *ACM Transactions on Mathematical Software*, Vol. 14, No. 1, pp. 33-44, 1988.

[37] Albrecht Preusser, FARB-E-2D: Fill Area with Bicubics on Rectangles-A Contour Plot Program, *ACM Transactions on Mathematical Software*, Vol. 15, No. 1, pp. 79-89, 1989.

[38] Robert R. Dickinson, Richard H. Bartels and Allan H. Vermeulen, The Interactive Editing and Contouring of Empirical Fields, *IEEE Computer Graphics and Applications*, Vol. 9, No. 3, pp. 34-43, 1989.

[39] Albrecht Preusser, Efficient Formulation of a Bivariate Nonic C^2-Hermite Polynomial on Triangles, *ACM Transactions on Mathematical Software*, Vol. 16, No. 3, pp. 246-252,1990.

[40] A. J. Worsey and G. Farin, Contouring a Bivariate Quadratic Polynomial over a Triangle, *Computer Aided Geometric Design*, Vol. 7, No. 1-4, 337-352, 1990.

[41] C. Singh and Debabrata Sarkar, A Simple and Fast Algorithm for the Plotting of Contours Using Quadrilateral Meshes, *Finite Elements in Analysis and Design*, Vol. 7, pp. 217-228, 1990.

[42] C. Singh, Comments on a Simple Algorithm for the Plotting of Contours, *Communications in Applied Numerical Methods*, Vol. 6, No. 3, pp. 191-195, 1990.

[43] K. J. Berry, Parametric Finite Element Contour Mapping, *Computer and Structures*, Vol. 35, No. 3, pp. 249-257, 1990.

[44] A. B. Ekoule, F. C. Peyrin and C. L. Odet, A Triangulation Algorithm from Arbitrary Shaped Multiple Planar Contours, *ACM Transactions on Graphics*, Vol. 10, No. 2, pp. 182-199, 1991.

[45] Ming-Jenq Twu and Bo Ping Wang, A New Method for Contour Plotting in Finite Element Analysis, *Computers and Structures*, Vol. 45, No. 5/6, pp. 1097-1102, 1992.

[46] B. Hamann, Modeling Contours of Trivariate Data, *Mathematical Modelling and Numerical Analysis*, Vol. 26, No. 1, pp. 51-75, 1992.

[47] T.C. Gopalakrishnan and Maha Korttom, An Algorithm for Contouring and Interpolation of Data Using Bilinear Finite Elements, *Finite Elements in Analysis and Design*, Vol. 14, pp. 37-54, 1993.

[48] C. M. Grimm and J. F. Hughes, Smooth Isosurface Approximation, in Proceedings of Implicit Surfaces ’95-First Eurographics Workshop Implicit Surfaces, Grenoble, France, pp. 57-67, April 1995.

[49] S. Rajasekaran and K. G. Venkatesan, A New Contouring Algorithm, *Computers and Structures*, Vol. 54, No. 5, pp. 953-977, 1995.

[50] Sergio deRada and Adel Ali, A High-Resolution Contouring Algorithm, *Computers in Industrial Engineering*, Vol. 31, No. ½, pp. 385-388, 1996.

[51] Gill Barequet and Micha Sharir, Piecewise-Linear Interpolation between Polygonal Slices, *Computer Vision and Image Understanding*, Vol. 63, No. 2, pp. 251–272, 1996.

[52] Bernd Hamann, Issac J. Trotts, and Gerald E. Farin, On Approximating Contours of the Piecewise Trilinear Interpolant Using Triangular Rational-Quadratic Bézier Patches, *IEEE Transactions on Visualization and Computer Graphics*, Vol.3, No. 3, pp.215-227, 1997.

[53] S. de Souza Lima and H. Lima Soriano, A Method for Graphic Stress Representation, *Computers and Structures*, Vol. 63, No. 9, pp. 1223-1228, 1997.

[54] P. Breitkopf, An Algorithm for Construction of Iso-valued Surfaces for Finite Elements, *Engineering with Computers*, Vol. 14, pp. 146-149, 1998.

[55] Siu-Wing Cheng and Tamal K. Dey, Improved Constructions of Delaunay Based Contour Surfaces, Fifth Symposium on Solid Modeling, Ann Arbor, MI, pp. 322-323, 1999.

[56] Y. Kenmochi, K. Kotani and A. Imiya, Marching Cubes Method with Connectivity, in proceedings of International Conference on Image Processing '99, Vol. 4, Kobe, Japan, pp. 361-365, 1999.

[57] Thomas A. Grandine, Applications of Contouring, *SIAM REVIEW*, Vol. 42, No. 2, pp. 297–316, 2000.

[58] A. R. Vasishtha and D. C. Agarwal, Analytical Solid Geometry, Krishna Prakashan Mandir, Meerut, India, 2000.

[59] John Strain, A Fast Semi-Lagrangian Contouring Method for Moving Interfaces, *Journal of Computational Physics*, Vol. 170, pp. 373–394, 2001.

[60] Tao Ju, Frank Losasso, Scott Schaefer and Joe Warren, Dual Contouring of Hermite Data, *ACM Transactions on Graphics* (Proceedings of ACM SIGGRAPH 2002), Vol. 21, No. 3, pp. 339-346, 2002.

[61] D. Hearn and M. P. Baker, Computer Graphics, Prentice Hall, New Delhi, India, 2002.

[62] O. C. Zienkiewicz, The Finite Element Method, Tata McGraw Hill, New Delhi, India, 2002.

[63] D. F. Wiley, H. R. Childs, B. Hamann, K. I. Joy and N. L. Max, Best Quadratic Spline Approximation for Hierarchical Visualization, in Data Visualization 2002, Proceedings of VisSym 2002, pp. 133-140, 2002.

[64] Shigeru Owada, Yoshihisa Shinagawa and Frank Nielsen, Enumeration Of Contour Correspondence, *International Journal of Image and Graphics*, Vol. 3, No. 4, pp. 609–627, 2003.

[65] Gill Barequet, Michael T. Goodrich, Aya Levi-Steiner and Dvir Steiner, Straight-Skeleton Based Contour Interpolation, Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics Philadelphia, PA, USA, pp. 119-127, 2003.

[66] D. F. Wiley, H. R. Childs, B. F. Gregorski, B. Hamann and K. I. Joy, Contouring Curved Quadratic Elements, Joint EUROGRAPHICS - IEEE TCVG Symposium on Visualization, pp. 167-177, 2003.

[67] Emma L. Bradbury and Wayne H. Enright, Fast Contouring of Solutions to Partial Differential Equations, *ACM Transactions on Mathematical Software*, Vol. 29, No. 4, pp. 418–439, 2003.

[68] Philip J. Schneider and David H. Eberly, Geometric Tools for Computer Graphics, Morgan Kaufmann Publishers, San Francisco, CA, 2003.

[69] Gregory M. Nielson, Dual Marching Cubes, 15th IEEE Visualization, Austin, Texas, USA, pp. 489-496, Oct 10th-15th , 2004.

[70] David Hutton, Fundamentals of Finite Element Analysis, McGraw-Hill Education (Asia), Singapore, 2004.

[71] Bo Chen and Harry H. Cheng, Interpretive OpenGL for Computer Graphics, *Computers and Graphics*, Vol. 29, pp. 331-339, 2005.

[72] O. C. Zienkiewicz, R. L. Taylor and J. Z. Zhu, The Finite Element Method, Elsevier Butteworth Heinemann, Burlington, M. A., 2005.

[73] R. Subramanian, Strength of Materials, Oxford University Press, New Delhi, India, 2005.

[74] Timothy S. Newman and Hong Yi, A Survey of the Marching Cubes Algorithm, *Computers and Graphics*, Vol. 30, pp. 854–879, 2006.

[75] Irena Jaworska, An Effective Contour Plotting Method for Presentation of the Postprocessed Results, Computer Vision and Graphics, Computational Imaging and Vision Series, Vol. 32, Springer, Berlin, pp. 1112-1117, 2006.

[76] Jing Jin, Qiang Wang, Yi Shen and Jiasheng Hao, An improved Marching Cubes Method for Surface Reconstruction of Volume Data, in proceedings of the 6th World Congress on Intelligent Control and Automation, Dalian, China, pp. 10454-10457, June 21-23, 2006.

[77] Adam W. Bargteil, Tolga G. Goktekin, James F. O'Brien and John A. Strain, A Semi-Lagrangian Contouring Method for Fluid Simulation, *ACM Transactions on Graphics*, Vol. 25, No. 1, pp. 19-38, 2006.

[78] Ibrahim Zeid, CAD/CAM, Tata McGraw-Hill, New Delhi, India, 2007.

[79] Gregory M. Nielson, Dual Marching Tetrahedra: Contouring in the Tetrahedronal Environment, ISVC 2008, Part I, LNCS 5358, pp. 183–194, 2008.

[80] D. S. Kumar, Heat and Mass Transfer, S. K. Kataria and Sons, New Delhi, India, 2008.

[81] D. L. Logan, Finite Element Method, Cengage Learning India Pvt. Ltd., New Delhi, India, 2008.

[82] Sadhu Singh, Strength of Materials, Khanna Publishers, New Delhi, India, 2008.

[83] D. Hearn and M. P. Baker, Computer Graphics with OpenGL, Pearson Education, New Delhi, India, 2009.