Rochester Institute of Technology
RIT Digital Institutional Repository

Theses

11-3-2009

Investigation of tetrahedron elements using automatic meshing in finite element analysis

Gordon Bae-Ji Tseng

Follow this and additional works at: https://repository.rit.edu/theses

Recommended Citation
Tseng, Gordon Bae-Ji, "Investigation of tetrahedron elements using automatic meshing in finite element analysis" (2009). Thesis. Rochester Institute of Technology. Accessed from

This Thesis is brought to you for free and open access by the RIT Libraries. For more information, please contact repository@rit.edu.

Investigation of Tetrahedron Elements Using Automatic Meshing in Finite Element Analysis

Gordon Bae-Ji Tseng

A Thesis summitted in partial fulfillment
of the
Requirments for the degree of

Master of Science
in
Mechanical Engineering
Rochester Institute of Technology
Rochester, New York
August 1992

Approved by:

Dr. Joseph S. Torok (Advisor)

Dr. Rober Hefner

Dr. Mark Kempski

Dept. Head of Mechanical Engineering
Dr. Charles W. Haines

Acknowledgment

I dedicate this work to my Mother and Father, who have provided support and encouragement to allow me to accomplish this achievement.

I would like to thank Dr. Joseph Torok, my advisor, for guidance , valuable suggestions and for always being available for assistance. He is a great professor and friend.

I would like to thank all the members of the RIT Mechanical Engineering Department who have been very helpful working with me through the school years.

I am deeply grateful to Zexel Gleason USA Inc., whose support and training provided me with all the help I needed during the past two years.

I, Gordon Bae-Ji Tseng, hereby grant permission to Wallace Memorial Library of Rochester Institute of Technology to reproduce my thesis in whole or in part. Any reproduction will not be for commercial use or profit.

Gordon Bae-Ji Tseng
August 3rd, 1992

I, Gordon Bae-Ji Tseng, hereby grant permission to Wallace Memorial Library of Rochester Institute of Technology to reproduce my thesis in whole or in part. Any reproduction will not be for commercial use or profit.

Gordon Bae-Ji Tseng
August 3rd, 1992

Thesis Outline

Abstract

Table of Contents

Chapter 1. **Introduction**

Geometric Modelers

1) Wire frame modelers
2) Surface modelers
3) Solid modelers
   - Constructive Solid Geometry (CSG)
   - Boundary Representation (bRep)
   - Cell Decomposition's and Spatial Enumeration

Approaches to mesh generation

- Laplacian method
- Mapping methods

Automatic mesh generation

1. Point placement followed by domain triangulation
2. Mesh generation based on sub-domain removal
3. Mesh generation by recursive subdividing
4. Spatial decomposition followed by sub-domain mesh

Control of the element distribution

Chapter 2. **Test Problems**

Static State

Modal analysis

Chapter 3. **Analytical Solutions**

- Cantilever (square) beam
- Cantilever (round) beam
- Square flat plate all edges fixed
- Circular flat plate with a edge fixed

Chapter 4. **FEA Results**

- Cantilever (square) beam
- Cantilever (round) beam
- Square flat plate all edges fixed
- Circular flat plate with a edge fixed

Chapter 5. **Discussion of Results**

- Cantilever (square) beam
- Cantilever (round) beam
- Square flat plate all edges fixed
- Circular flat plate with a edge fixed

Chapter 6. **Conclusion**

Chapter 7. **Summary**

References

Abstract

This investigation examines the quality of finite element analysis (FEA) results based on the use of tetrahedron elements. For some classes of problems analyzed by the finite element method (FEM), the use of various polynomial order tetrahedra is considered quite acceptable. However, in other classes of problems, particularly stress analysis, users have a strong bias against these types of elements. Various case studies are performed, comparing results based on several types of three-dimensional elements.

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction

The finite element method is one of the numerical techniques used for the approximate solution of engineering problems. The procedures for FEA can be divided into three stages:

1. Pre-processor
2. Processor
3. Post-processor

The Pre-processor is the stage to make geometry (model), create elements (mesh) and apply the boundary conditions. The processor is a numerical solver which solves the FEA model. The post-processor is a tool which allows the user to see the results. Once the FEA model is finished, the user can only send it to the solver (processor) and check the result (post-processor) after that. The user has no control at the processor and the post processor stages.

The most critical stage is the pre-processor. At this stage, the user must fully understand the problem and use all the tools which are provided by the FEA program. A single mistake can adversely effect the FEA result. If the user fully understands the problem, the only thing that can impede the results are the tools of pre-processor which are provided by the program.

Finite element solvers can give solutions to rather complex problems and have traditionally been used for analysis of components with complicated shapes and boundary conditions (loads and constrains). Various solvers exist which will give reasonably accurate and reliable results for these problems. All the finite element computer packages require an input of a finite element mesh, loads, boundary conditions (constraints) and material properties. The finite element mesh describes to the program, the discrete geometry of the domain to be analyzed. Communicating this mesh to the finite element solver has normally been done manually and thus has been time-consuming and error-prone.

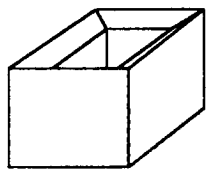
The usefulness of geometric modelers for finite element mesh generation is readily apparent. Since a finite element mesh communicates the geometry of the domain to be analyzed, it seems appropriate that it could be constructed from the geometry of the domain in question. The geometry of parts and assemblies are stored on computers using programs called geometric modelers [1,2].

Geometric modelers

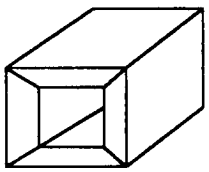
The geometric modelers can be classified into various types according to the way they store the geometry of a object.

1) Wire frame modelers

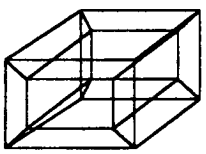
The object is represented by its edges. Imagine putting a wire for every edge in the object. The resulting representation is a wire frame. No information is present regarding the surface. Because it does not completely represent an object with a surface, it could lead to an ambiguous object, a example is given in Figure 1.1.



Top to bottom ?



Front to back ?



Wireframe

Figure 1.1 Wire frame representation

2) Surface modelers

The object may be represented by its surfaces. curved surfaces can be represented and objects are unambiguous. There is no information regarding what is inside or outside the object. Thus it is not possible to compute the volume or other properties of the object.

3) Solid modelers

The object may be represented in various ways but it is defined unambiguously and it should be possible to compute volume, mass, etc. There is information regarding the inside and outside of the object. In other words, the object can be represented completely without

ambiguity. Most solid modelers store the geometry data in a way that can be classified in one of the three categories or their combinations.

- Constructive Solid Geometry (**CSG**)

The object is built by performing *Boolean* operations (union, difference, and intersection) on simple pre-defined objects called primitives, such as blocks, cones, cylinders, tube, spheres, etc. The actual boundary of the object is usually not stored but can be computed at any time. An example is given in Figure 1.2.

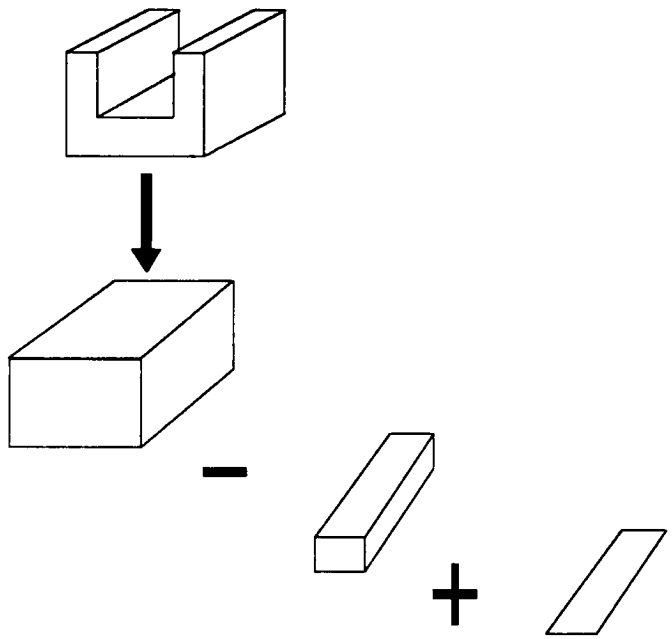


Figure 1.2 Constructive Solid Geometry modeler

- Boundary Representation (**bRep**)

The boundary of the object is stored in what is called boundary file in a structured manner such that it is possible to identify the regions inside or outside the object. The boundary of the boundary of the body will usually be an ordered set of surfaces, faces, loops edges and vertices. An example is given in Figure 1.3.

- Cell Decomposition's and Spatial Enumeration

The object is represented by decomposing it into a bunch of cells, cubes or blocks, which can be thought of as building up the object. The domain of the objects is represented by a tree structure, quadtree in 2-

dimension and octree in 3-dimension, with status of the cells (partial, full and empty), see Figure 1.4. for example. The boundary of an object can be approximated by the cells.

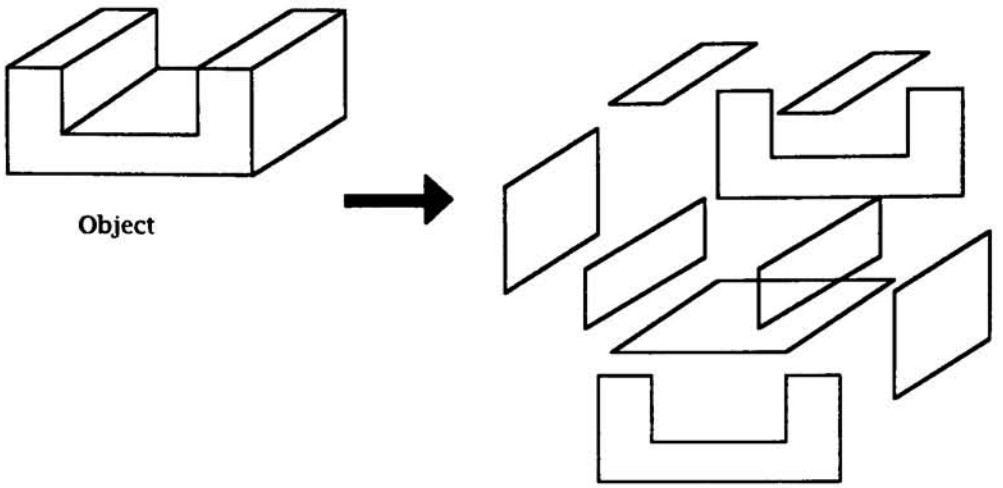


Figure 1.3 Boundary representation modeler

The early systems that simply computerized the drafting process do not contain all the geometric information needed to allow applications to operate automatically. Therefore , the more recent solid modeling systems employ complete and unique geometric representations. These systems contains all the geometric information needed to allow any finite element mesh generation techniques to automated.

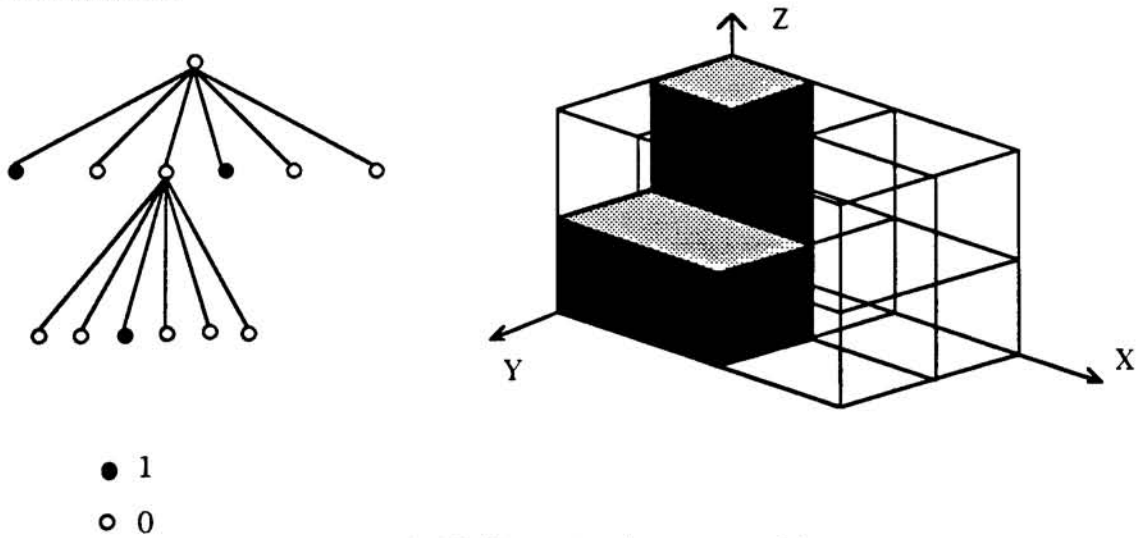


Figure 1.4 3-dimensional octree modeler

The early systems that simply computerized the drafting process do not contain all the geometric information needed to allow

applications to operate automatically. Therefore , the more recent solid modeling systems employ complete and unique geometric representations. These systems contain all the geometric information needed to allow any finite element mesh generation technique to be automated.

Approaches to mesh generation

The problem of mesh generation [3] is to convert the geometry from one of the geometric modelers to a form understood by a finite element solver, in a manner as automatic as possible. The increase in the level of automation will allow the finite element method to be used by engineers to provide reliable analysis results to their problem. There are various popular ways of generating these meshes and they can be classified into the following categories.

1. Laplacian method,
2. mapping methods,
3. point placement followed by triangulation,
4. removal of individual sub-domain,
5. recursive subdivision of the domain, and
6. spatial decomposition followed by sub domain meshing.

An automatic mesh generator is an systematic procedure capable of producing a valid finite element mesh in a domain of arbitrary complexity given no input past the computerized geometric representation of the domain to be meshed. The Laplacian method and mapping method are not the real automatic mesh generation techniques. But both two methods are widely used (especially for quadratic brick (Hex) element or linear brick element) in the finite element method.

• Laplacian Method

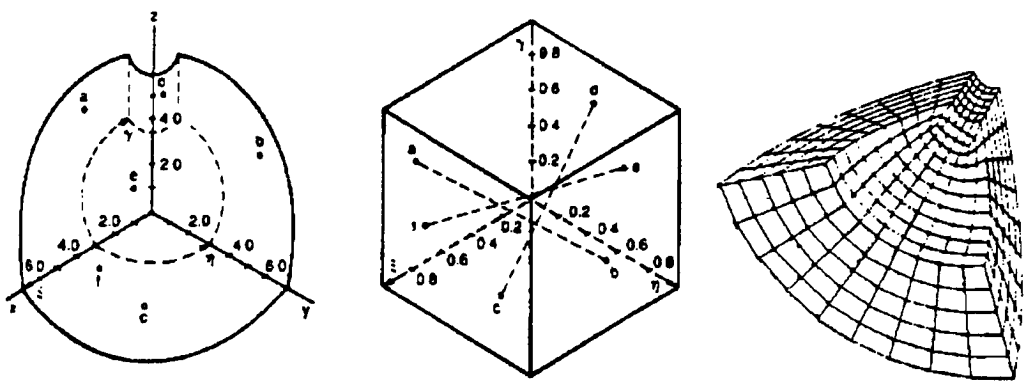
A set of simultaneous nonlinear equations for the position vectors of the interior nodes with respect to the neighboring nodes is solved using iterative techniques. A starting grid is required. The laplacian method iteratively replaces the interior node P_i

$$P_i = \sum_{\substack{j=1 \\ j \neq 1}}^{Nj} P_j \quad \text{..... Eq. 1.1}$$

where N_i is the number of nodes to which P_i is connected and P_j are the node points of the connected nodes. The process has proven useful in mesh generation algorithms and smoothes the mesh into one with better proportioned elements. This may be used to smooth meshes created using other methods.

• Mapping Methods

A function is used to map the given geometry into a simple geometry, usually a square in two dimensions and a cube in three dimensions. This simple geometry is meshed and all the node points are mapped back to the original geometry. Various mapping functions have been used, the trans-finite mapping are most commonly used for mapped mesh generation. Mapping methods do impose a number of restrictions on the geometry of the object. When mapping methods are used, the geometry of the object is constructed by gluing together the individual, fixed topology, mesh patches (see Figure 1.5).
Therefore, the geometric representation is explicitly defined in terms of mesh patches. The user is responsible for defining a valid set of mesh patches, which implicitly define the geometric representation and explicitly provide the geometry necessary for meshing to occur. The mesh generators are, therefore, not concerned with the actual geometry



a. Body coordinate b. Mapped on a unit cube c. Mesh generated

Figure 1.5 A 3-dimensional mapping mesh generation

of the object. This is however, not the case for an automatic mesh generator which is given a complete geometric representation of the domain of interest and is responsible for decomposing it into a valid set of elements. This paper will not discuss this other than to say that the mapping techniques have been found to be rather inadequate for automatically generating meshes on arbitrary complex geometry.

Automatic Mesh Generation Techniques

The other mesh generation methods are fully automatic mesh generation procedures, which are fully 3-dimensional or the extension from 2-dimensional to 3-dimensional mesh generation appears possible. Currently, these mesh generation techniques only use tetrahedron elements in 3-dimension; no other type of elements are available.

1. Point placement followed by domain triangulation

This type of mesh generator involves two independent process [6,7,8,9]. First, node points must be inserted within and on the boundary of the structure to be meshed. Secondly, the node points are automatically triangulated to form a network of well-proportioned elements. The triangulated algorithms function in both 2- and 3-dimensional settings producing meshes of triangular and tetrahedral elements, respectively. These two algorithms function independently of each other, so that there are many ways to triangulate the elements. For ease of exposition, we first discuss a mesh triangulation algorithm.

Recent efforts [6,10,11,12] employ the properties of the geometric constructs of Dirichlet tessellation and more importantly for mesh generation, the Delaunay triangulation of given set of points. consider first the 2-dimensional case. Let $P_1, P_2,...,P_N$ be distinct points in the plane, and define the sets V_i, $1<i<N$, where

$$V_i=\{X: |X-P_i| < |X-P_j| \text{ for all } j\neq i\} \quad \text{Eq. 1.2}$$

where $|.|$ denotes Euclidean distance in the plane. V_i represents a region of the plane whose points are nearer to node P_i than to any other nodes. Thus V_i is an open convex polygon (usually called a *Voronoi polygon*) whose boundaries are portions of the perpendicular bisectors of the lines joining node P_i to node P_j when V_i and V_j are contiguous. The collection of Voronoi polygons is called the *Dirichlet tessellation.* In general, a vertex of a Voronoi polygon is shared by two other polygons so

that connecting the three generating points associated with such adjacent polygons form a triangle, say T_k. This set of triangles $\{T_k\}$ is called the *Delaunay Triangulation.* See Figure 1.6 for example.

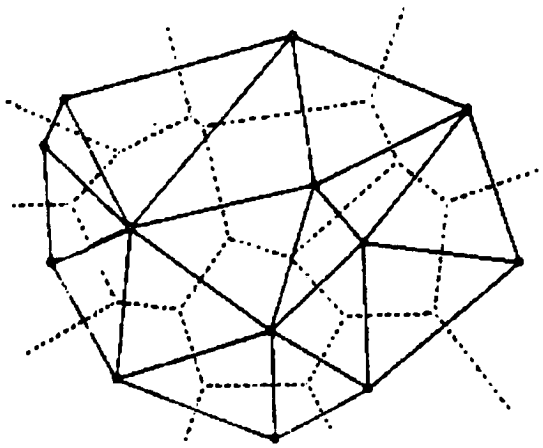


Figure 1.6 Dirichlet tessellation and Delaunay triangulation

The construct can be shown to be a triangulation of the convex hull of the node points. An important property of 2-dimensional Delaunay Triangulation which makes it suitable for use as a finite element that its triangles are as close or equilateral as possible for the given set of nodes. Consequently, ill-conditioned and thin triangles are avoided whenever possible. There are many approaches to the construction of a Delaunay triangulation. A currently popular approach is a version proposed by Watson[8].

In two dimensions, Watson's algorithm turns upon the simple observation that three given node points will form a Delaunay triangle if and only if the circumdisk defined by these nodes contains no other node points in its interior. The algorithm is initialized by calculating the coordinates of three node points which form a triangle T_0 that surrounds all the node points to be inserted. The circumcentre coordinates and circumradius of the circumcircle defined by T_0 are also calculated and recorded. The node points are then introduced one at a time. The algorithm operates by maintaining a list of triplets of node points which represent completed Delaunay triangles. Associated with each such triangles are the coordinates of its circumcentre and circumradius. For each new node point entered, a search is made of all current triangles to identify those whose circumdisks contain the new point. For such disks, the associated triangles are flagged to indicate removal. As shown in Figure 1.7, the union of all such triangles forms what we call an insertion polygon containing the new node point. It can

be shown that no previously inserted node is contained in the interior of the polygon and that each boundary node of the polygon may be connected to the new node by a straight line lying entirely within the polygon. Thus, a new triangulation of the region enclosed by the polygon is formed. Repeated use of this insertion algorithm permits all node points to be entered, while ensures that at each step the triangulation retains its Delaunay properties.

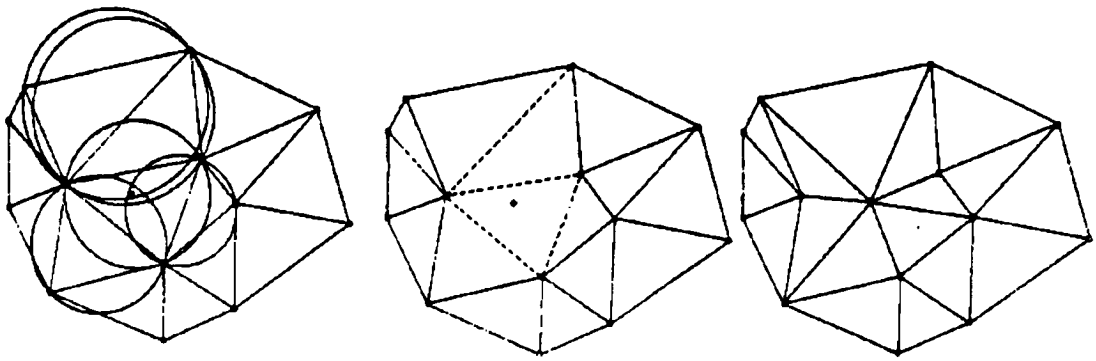


Figure 1.7 Watson's Algorithm

In three dimensions, Watson's Algorithm starts with a *tetrahedron* T_0 containing all points to be inserted, and new internal tetrahedron are formed as the points are entered one at a time. At a typical stage of the process, a new point is tested to determine which circumspheres of the existing tetrahedron contains the point. The associated tetrahedra are removed, leaving an insertion polyhedron containing the new point. Edges connecting the new point to all triangular faces of the surface of the insertion polyhedron are created, defining tetrahedral which fill the insertion polyhedron. Combining these with the tetrahedral outside the insertion polyhedron produces a new Delaunay triangulation which contains the newly added point.

There are a number of ways that node points can be inserted within the domain. For instance, one of these methods used in [6] is to cut planes through the structure, say $P_1,P_2,...,P_N$. It is within each of these cross-sections that node points will be defined. Figure 1.8 illustrates the steps to define the node points in a space. In summary , node points are defined interactively a plane-at-a-time. Within each plane, the user has control over local node densities. It is possible to automate this node insertion process further.

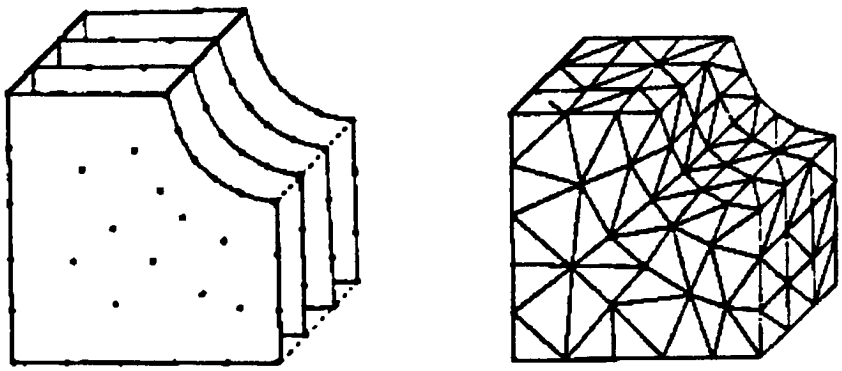


Figure 1.8 A way to define the node points in space

Although Delaunay triangulation performs very well in two dimensions, several problems exist with Watson's approach in 3-D. The first one is the existence of degenerate cases. These occur in practice when a newly inserted node appears to lie on the surface of a circumsphere associated with some existing tetrahedron. The problem becomes apparent whenever the distance from a newly entered nodal point to an existing circumsphere is less than e, where e is the expected accumulated computer truncation error. This in turn produces structural inconsistencies in the triangulation, that is over lapping tetrahedral or gap in the mesh.

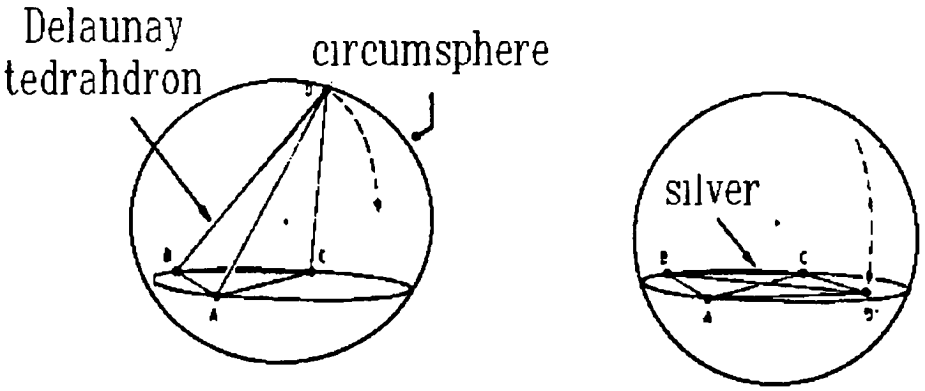


Figure 1.9 The distorted Delaunay tetrahedron, "silver"

Another serious problem which arises in the three dimensions, and which also requires modification of the Delaunay Triangulation, occurs with the creation of tetrahedron we call "silver". In this case (see Figure 1.9), "silver" will defines a badly distorted Delaunay tetrahedron whose faces are well-proportioned triangles, but whose volume can be

made arbitrary small. Therefore, these methods which use Delaunay Triangulation need checking criterion to eliminate those ill-shaped elements after the mesh elements have been generated.

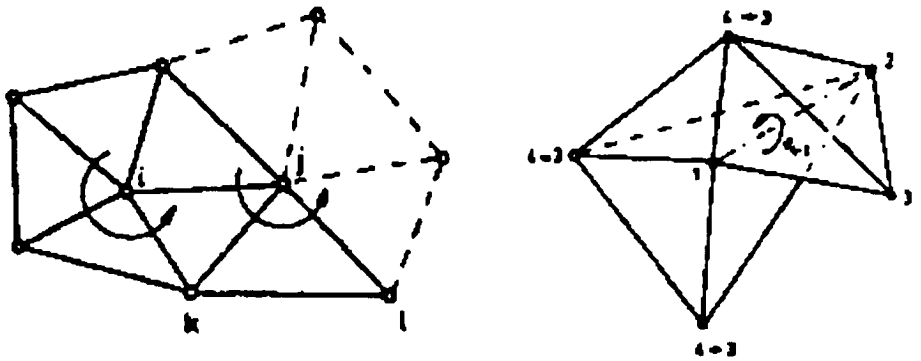


Figure 1.10 Surrounding algorithm

Other rule-based procedures of triangulation have been developed [7,8]. For instance, in two dimensions, a method hinges on the principle of fully surrounded points. Starting with node 1, each node is surrounded in turn with triangular elements. When a node is being surrounded, new triangles are only formed with nodes of higher node number which is registered by a special scanning procedure. This automatically ensures that triangles do not overlap. The process starts by selecting the nearest
point to node 1 and thus establishing a side 1j where j is the node number of the nearest point. When surrounding another point i, the first side is established only after checking whether i is already part of an existing triangle. If this is the case, the existing triangle is used to provide this starting side. The side ij is then used to seek a node k such that the angle i-k-j is maximum and i-j-k is an anti clockwise sequence. This new point k becomes j and the process is repeated and so on. If a node l is found such that triangle ikl already exists this routine is omitted and l becomes the new point i. The process stops when a triangle is obtained containing the side from which the process began and the point is thus completely surrounded (see Figure 1.10 for detail.) The extension to 3-dimension could be in [9]. In this approach, the surrounding a given point with triangular element is replaced with surrounding a line between two points with elements and then move on to another line until the mesh is complete. A flow chart of this routine is shown in Figure 1.11.

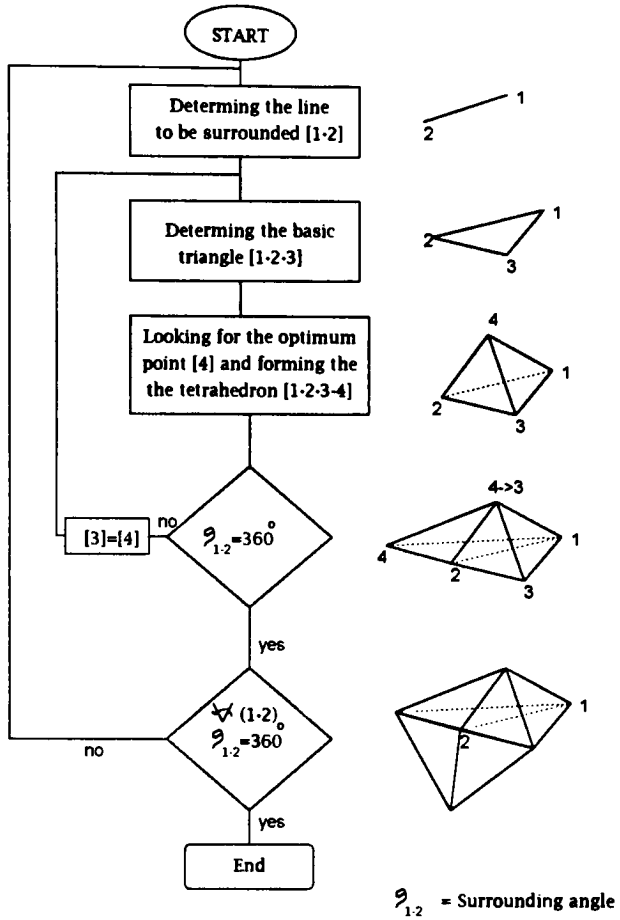


Figure 1.11 The flow chart of the 3-dimensional surrounding algorithm

The node insertion of the surrounding algorithm mentioned above uses a slightly different scheme. The node point number is recorded in sequence by a specific scanning procedure. When all the "true" node points have been recorded, the number of the last node is noted and then start to record "ghost" points outside the boundaries of the structure, see Figure 1.12 The use of "ghost" points removes the need for the identification of nodes on external boundaries. Ghost points must be placed so that each element side which is on an external boundary can form a triangle with a ghost point. Otherwise, the mesh generation program will form a triangle which does not exist on the structure between the boundary node points. These "ghost" triangles will be removed after the meshes have been generated.

Though these algorithms use properly constructed set of rules capable of producing a well-conditioned mesh within a domain, they require extensive searching and large number of checks, such as degenerate and overlapping cases, many more than other mesh generating techniques. In addition, it is difficult to develop a set of

triangulation rules that would insure the elements generated satisfy a given shape criterion.

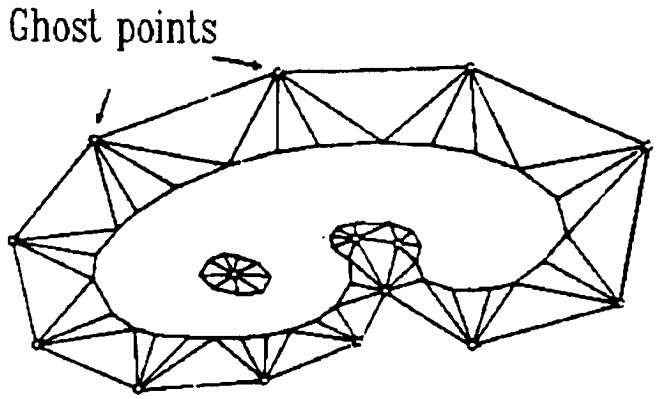


Figure 1.12 The "Ghost" point for boundary definition

2. Mesh generation based on Sub-domain removal

Automatic mesh generation procedures in this method [13,14,15, 16] operate by removing individual pieces from the domain one at a time until the domain is reduced to one remaining acceptable piece. Element removal meshing procedures employ a specific set of element removal operators that are capable of removing a single element from an object. They operate by first examining the topological features of the object testing a specific set of geometric measures to see if any of the element removal operators can be applied. There is a pre-specified hierarchy in which the various operators are applied. They typically employ a boundary representation of the domain and operate by searching for entities of specific type that satisfy a set of conductivity and geometric requirements. Two examples of this kind are represented as followed.

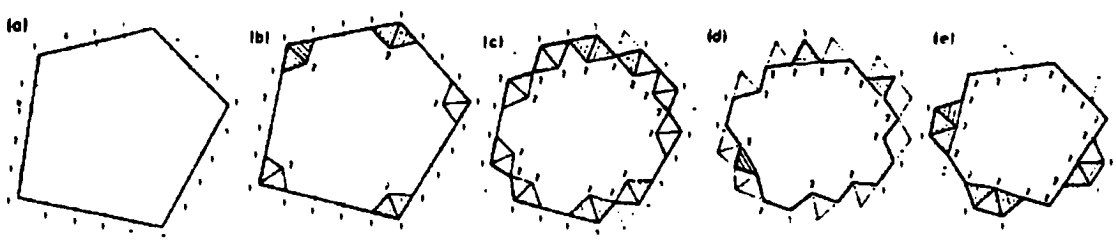


Figure 1.13 Subdivision of a multilateral domain

The first approach is an subdivision of a planar polygon by triangular elements [13]. Given a certain domain with a number of nodes

around its boundary, we start by determining the most well-conditioned elements which can be formed at each corner of the domain. Having finished this step, the elements formed are considered to be cut out from the original area and the same concept is applied to the remaining area and so on. The level equals unity. Subdivision at a corner node of level equals unity generates nodes of level equals to two and so on. The subdivision of domain is per formed in successive stages. In each stage, A continuous boundary layers is cut out. This is shown diagrammatically in Figure 1.13 & 1.14. Starting with the nodes around the original boundary, Triangular elements are generated and a new set of nodes of level equals two are obtained. The first stage is known to be complete when all the nodes bounding the area to be subdivided become of level equals two. Again, starting with these nodes, a new set of nodes of level equals three are generated and so on. The mesh generation process is known to be complete when the number of nodes around the boundary of the remaining area reach three. This means that the remaining area is one single triangular element and hence no more subdivision is required.

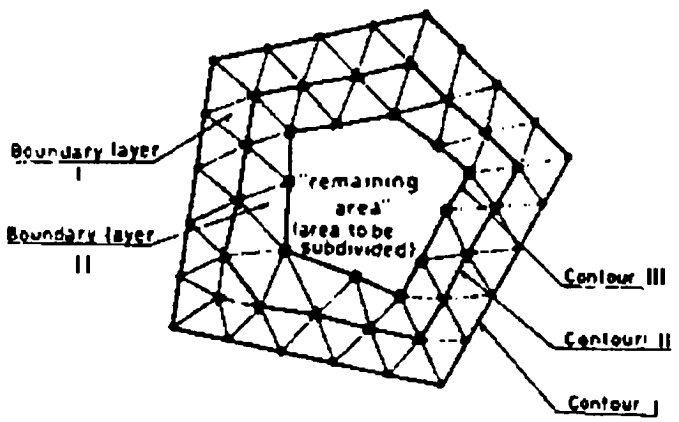
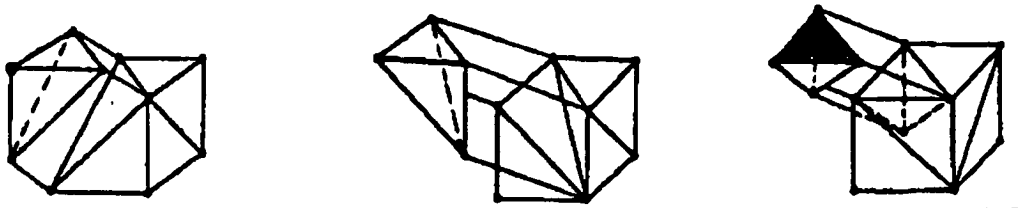


Figure 1.14. Nodal levels

The second approach [14] is a 3-dimensional subdivision of object which is extended from an 2-dimensional procedure. Consider the basic element removal operators used to mesh a 3-dimensional subdivision domain without void. The first operator, VERTEX_REMOVAL is applied by searching the object for vertices with only three edges coming into it. Any such vertex satisfies a set of geometric interference requirements can be validly removed from the object. The removal of vertex carves a tetrahedron from the object. In cases where all vertices have more than three vertices, a second operator, EDGE_REMOVAL, is applied. In this case, tetrahedron containing the selected edge is carved from the object. Since this operation reduces the number of edges connected to two of

the vertices by one of each, it eventually reduces the complexity of the object until the first operator can be applied again. In case where neither of the first two operators can be applied a third operator, FACE_REMOVAL, can be applied. In this case, a tetrahedron containing a face of the object and connecting the three vertices that bound the face being removed. These operators are illustrated in the Figure 1.15.



(A) VERTEX_REMOVAL (B) EDGE_REMOVAL (C) FACE_REMOVAL

Figure 1.15 Element removal operators

A topological-based element by element removal procedure appears ideally suited for the construction of optimal h-p (HEX element) finite element meshes where coarse, exponentially graded meshes are desired. Since the amount of computation required for the application of each removal operation is high, these procedures are not computationally efficient for the creation of a fine mesh. The development of an algorithm that decomposes the domain into large chunks by removing them one at a time is an attractive way to consider the automation of the current methods of mesh generation where the user interactively decomposes the domain of interest into mappable regions and invokes a mapping mesh generator which we have mentioned at the beginning [4,5]. The difficulty in developing such an approach is the identification and implementation of a set of rules that would examine a geometry to determine how to decompose it into mappable regions that will yield the type of mesh generations desired as well as providing a satisfactory mesh topology.

3. Mesh generation by recursive subdivision

The recursive subdivision mesh generators [17,18,19] operate by the repeated splitting of a domain into simpler parts until the individual parts are single elements, or, possibly, simple regions in which elements can be quickly generated. The *Triquamesh* technique is one of the wide used recursive subdivision method in this area. The Triquamesh technique is based on the surface/volume triangulation. It can be used to create triangular or quadrilateral meshes on surface. It can also be broken up into hexahedrons. The whole technique is based on two basic laws of analytic geometry at this time:

- Every polygon is divisible into triangles.
- Every polyhedron is divisible into tetrahedron.

It follows that we will be able to create a mesh consisting of triangles of any surface. Also, we will be able to create a mesh consisting of tetrahedral in any volume. To explain the working of this method, we start with a description in 2-dimensions [18]. The extension to 3-dimension is then straight forward.

Any area can be divided by the user into a certain number of sub-area. Each sub-area needs to be coherent, that is, a closed loop. An area with n-fold incoherence can be made coherent by n-1 cuts. The first step involves obtaining a convex polygon. So, if the polygon is already convex, we move to the next step. Otherwise, This polygon is successively subdivided, until a bunch of convex polygons are obtained. This process of subdividing a concave polygon into more than one convex polygons is controlled by certain heuristics, which have essentially been obtained by trial and error. The aim is to get a split line which will pass through a concavity in the concave polygon and which is likely to yield good elements. These split lines have to be chosen rather carefully because these later become the boundaries of the elements. Once a polygon is divided into two by using a split line, see Figure 1.16, we need to create nodes on the split lines. The problem now reduces to meshing two polygons. This process is continued as long as we have polygons. Once we have convex polygons, they are taken up one by one. An attempt is made to chop off layers from the polygon, where the polygon has a sharp angle. These layers yield elements. These layers are chopped successively, such that each of the layers yields good elements. Again, this is done using certain heuristics. Once no more sharp corners are found, the polygon is again cut into two by use of a split line as

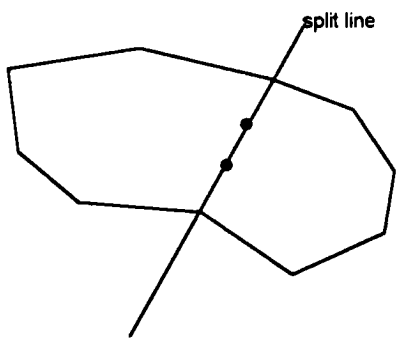


Figure 1.16 Cutting concave polygon using splitting line

explained earlier and this process continues until the mesh completed, see Figure 1.17 for details.

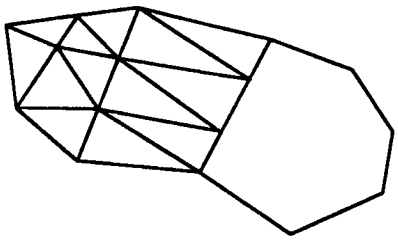


Figure 1.17 chopping off layers yield elements

The same concepts are extended to 3-dimension. A surface in 3-dimension is projected (or transformed) to 2-dimension and the above meshing is done. For solid elements, tetrahedron are generated. These can later be subdivided into hexahedrons. We will now be dealing with convex polyhedra instead of polygons and split planes instead of split lines then same procedures from 2-dimensional mesh applied until the mesh complete.

As in the sub-domain removal procedures, this class of mesh generator typically operates off a boundary representation of the domain to be meshed, looking for candidate topological features meeting specific conductivity and geometric requirements. Some structures which this method cannot split should be transformed or sub-divided to those that can be split.

4. Spatial decomposition followed by sub-domain meshing

The basic idea behind this approach [10,11,14,19,20,21,22,23] is use an efficient procedure to decompose, in a controlled manner, the domain of interest into a set of simple cells and to then mesh the individual cells in such a manner that the resulting mesh is valid. The one spatial decomposition approach that has been applied to mesh generation is the *quadtree* in 2-dimensions and the *octree* in 3-dimensions. The entire structure is stored in a hierarchic tree as shown in figure 1.18.

One algorithm to building a 3 dimensional mesh generator using this basic tree representation is the modified-octree (modified-quadtree in 2-dimension) [21,22,23]. the basic steps in the modified-octree mesh generating processes are :

a. Set up an integer coordinate system that contains the object to be meshed.

b. Generate the modified-octree representation of the object accounting for the mesh gradation information specified with the geometric model.

c. Break the modified-octree up into a valid finite element mesh.

d. Pull the nodes on the boundary of the modified-octree to the appropriate vertices, edges and faces of the original geometry.

e. Smooth the locations of the node points to create a better conditioned finite element mesh.

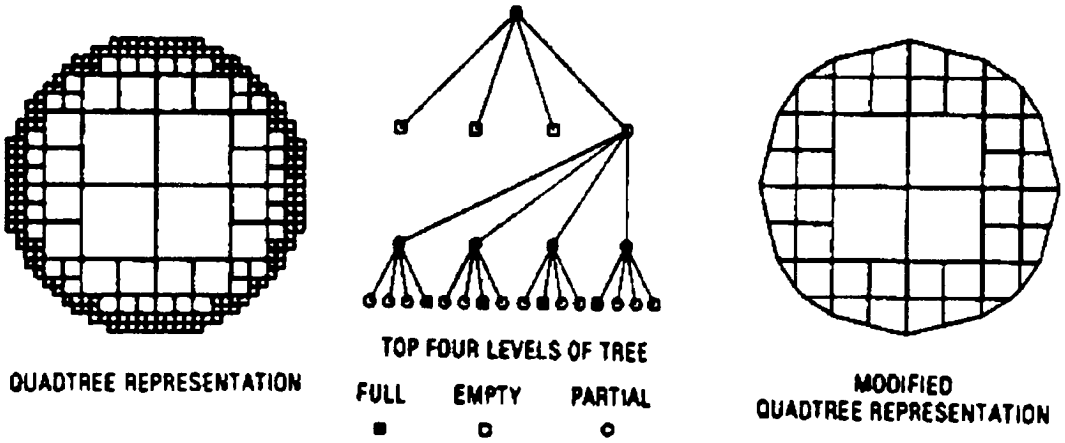


Figure 1.18 The quadtree and modified quadtree representations

The essential difference between an octree and a modified- octree is that the modified-octree allows for the definition of a cut octant. Since the size of octree cubes desired for use in the finite element mesh generation are large with respect to the geometric details of the object, it is necessary to deal in a specific manner with those octree cubes that contain the boundary of the object and are neither fully inside nor outside the object. The cut octants, which is the octants containing the boundary of the object, is completed by qualifying which side of the discrete boundary existing in the octant is inside the object. To maintain the integer tree storage and to limit the number of cut octant cases to a manageable number, only the corners and half-points of an octant cases are used in the cutting process. This operation requires a specific set of geometric checks [23]. With IN/OUT information of these points, the sharp of faces of the octant that are cut by the cutting surface are defined. One approach to define the cutting surface is to allow only single planar cuts. However, for many objects there will be situations

when the cutting surface points are not coplanar. In these situations, the surface representation will have a discontinuity, which will lead to problems when the final mesh modifications are performed. Therefore, it is necessary to include a more extensive set of octant cutting surfaces. The IN/OUT information can be properly represented with an addition of a set of two planar cut octants and a limited number of three planar cuts, see Figure 1.19. After the octant on the boundary are defined, the interior octants within the boundary are then quickly filled by a simple tree traversal process. Once the modified-octree is available, it is broken into a set of valid finite elements. After an integer finite element mesh is available, the final steps include pulling (or assign) nodes to the boundary and smoothing the nodal locations to ensure the best possible element sharps for the given element topology.

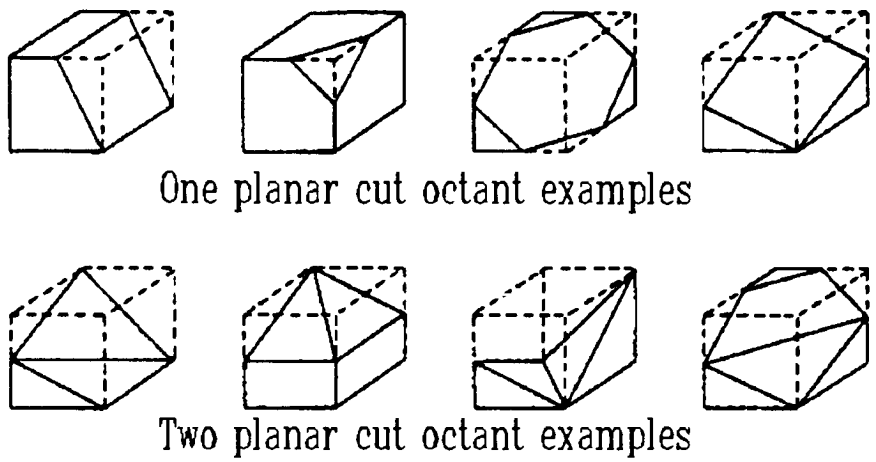


Figure 1.19 Examples of cut octants used in the modified octree algorithm

Another approach based on spatial decomposition uses the combination of *Octree* and *Delaunay Triangulation* [10,11,12]. The basis of the scheme are introduced briefly as follows (see figure 1.20 for details):

a. Tree building. Given a geometric model, generate its octree representation. Each octant is classified inside, outside, or on the boundary of the object. An octant is classified on the boundary when any of its features (vertices, edges, or faces) is classified on or intersects the boundary.

b. Octree triangulation. Each octant classified inside or on the boundary is triangulated using templates or Delaunay procedure.

c. Intersection point triangulation. The octree triangulation serves as the initial triangulation for the intersection point triangulation . In

this step the intersection points generated during tree building are incorporated into the triangulation using the Delaunay property.

d. Classification and compatibility. The geometric triangulation requires that the mesh entities are classified against the model geometry and that topological compatibility is assured. If an incompatibility is identified, it resolved either through a local resolution procedure or by refining the octree.

e. Mesh improvements. A point reposition procedure, such as Laplacian method, may be used to improve the quality of the mesh elements.

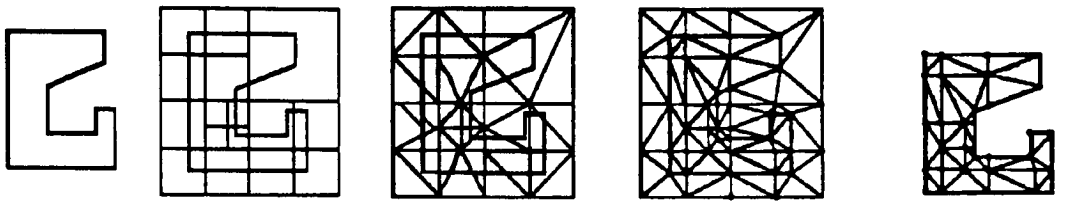


Figure 1.20 A overview of octree/Delaunay triangulation method

Control of the element distribution

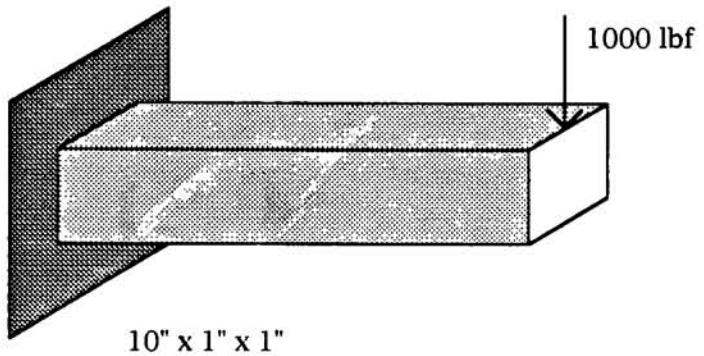
In addition to the ability to generate a valid mesh for any geometry, automatic mesh generators must permit the types of mesh gradations necessary to produce efficient finite element models. Ideally, the mesh control device should allow for the convenient specification of both a priori and a posterior mesh generation information. A priori mesh control device are used to specify the distribution of elements in the initial finite element model. Since the basic input to an automatic mesh generator is a geometric representation., any priori mesh control device must be tied to the geometric representation. This means that a priori mesh control can also be a function of the particular geometric modeling approach used. A posterior mesh control devices are used an adaptive analysis process to improve the mesh as indicated by the results on the overall discretization error in one or more solution norms. The primary function of a posterior error estimators are to provide a convergent and accurate measure of the discretization error of a given finite element solution. To used most effectively, the mesh generation procedures must be coupled with adaptive analysis procedure that can insure that the final mesh yields the requested degree of accuracy. Without adaptive analysis procedures based on reliable a posterior error estimators, the analyst will need to use a priori mesh control techniques to generate the desired element distributions.

Chapter 2. Test Cases

The benchmark is based on two types of analyses (static state and modal analysis). Each type of analysis will given two geometries and meshes by linear and quadratic tetrahedron and quadrilateral elements. All the FEA results will be compared with hand calculation.

Static State

Cantilever (square) beam

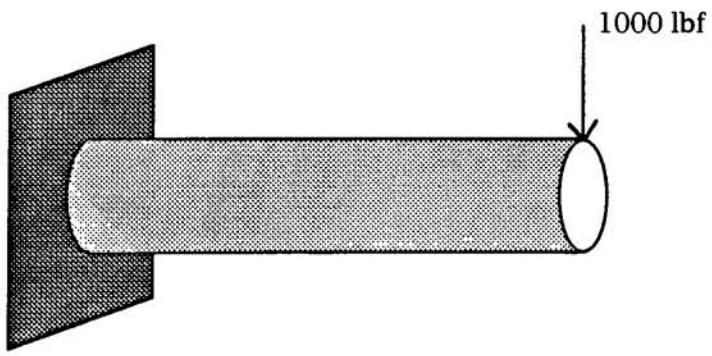


10" x 1" x 1"

Material : Steel E = 3e7 psi υ = .3

Figure 2.1

Cantilver (round) beam



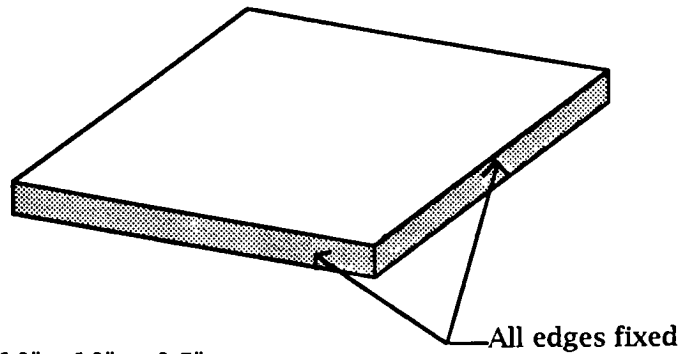
D = 1", L = 10"

Material : Steel - E = 3e7 psi υ = .3

Figure 2.2

Modal Analysis

Square flat plate all edges fixed

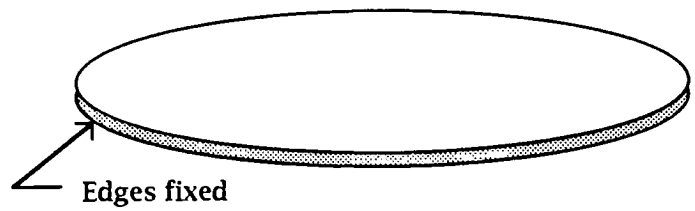


10" x 10" x 0.5"

Material : Steel - E = 3e7 psi υ = .3

Figure 2.3

Circular flat plate with a edge fixed



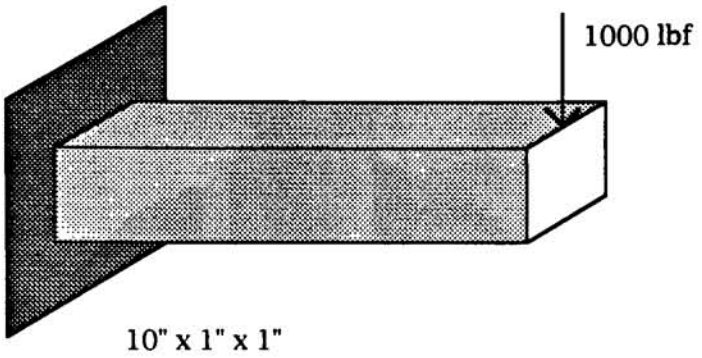
OD = 10" t = 0.5"

Material : Steel E = 3e7 psi υ = .292

Figure 2.4

Chapter 3. Analytical Solutions

Cantilever (square) beam



10" x 1" x 1"

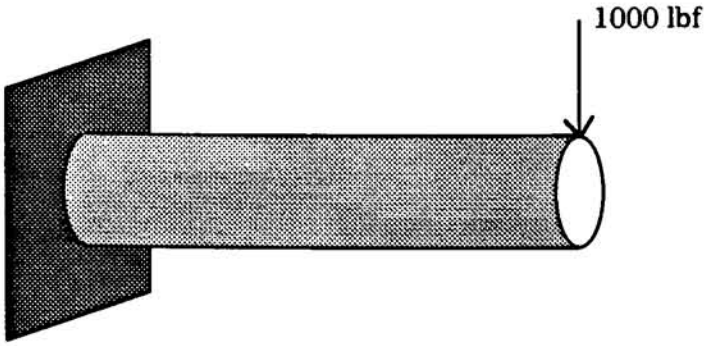
Material : Steel - E = 3e7 psi υ = .3

Figure 3.1

$$I = \frac{bh^3}{12} = \frac{1(1)^3}{12} = 0.08333 \text{ in}^4 \quad \text{..... Eq. 3.1}$$

$$Y_{max} = -\frac{PL^3}{3EI} = -\frac{1000(10)^3}{3EI} = -0.1333 \text{ in} \quad \text{..... Eq. 3.2}$$

Cantilver (round) beam



D = 1", L = 10"

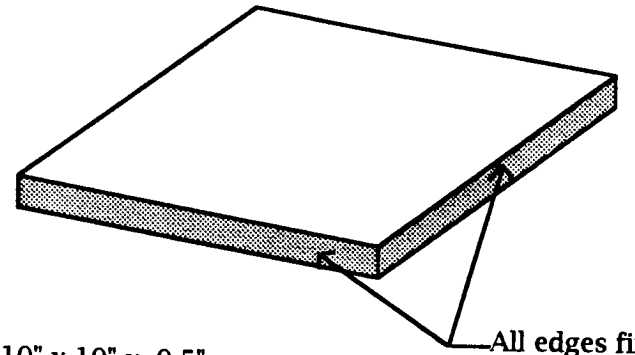
Material : Steel - E = 3e7 psi υ = .3

Figure 3.2

$$I = \frac{\pi r^4}{4} = \frac{\pi (.5)^4}{4} = 0.0491 \text{ in}^4 \quad \text{..... Eq. 3.3}$$

$$Y_{max} = -\frac{PL^3}{3EI} = -0.226354 \text{ in} \quad \text{..... Eq. 3.4}$$

Square flat plate all edges fixed



10" x 10" x 0.5"

Material : Steel - E = 3e7 psi υ = .3

Figure 3.3

$$f_n = \frac{k_n}{2\pi}\sqrt{\frac{Dg}{wa^4}} \quad \text{..... Eq 3.5}$$

where

k1 = 36 fundamental
k2 = 73.4 one nodal diameter
k3 = 108.3 two nodal diameter
$D = Et^3/12(1-\nu^2)$
g = gravity
w = wt./area = density x *thickness*
a = length of edge

So

$$D = \frac{Et^3}{12(1-\nu^2)} = \frac{3e7(0.5)^3}{12(1-(0.292)^2)} = 341628.6229(\text{in-lb.}) \quad \text{..... Eq. 3.6}$$

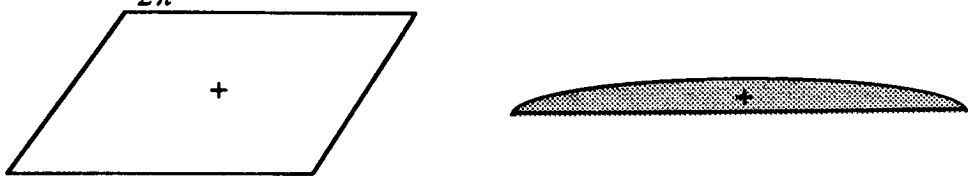
$$w = \rho t = 0.283(0.5) = 0.1415(\text{lb./in}^2) \quad \text{..... Eq. 3.7}$$

Therefore

$$\sqrt{\frac{Dg}{wa^4}} = \sqrt{\frac{341628.6228(386.4)}{0.1415(10)^4}} = 305.434\ (1/\text{in-sec}) \quad \text{..... Eq 3.8}$$

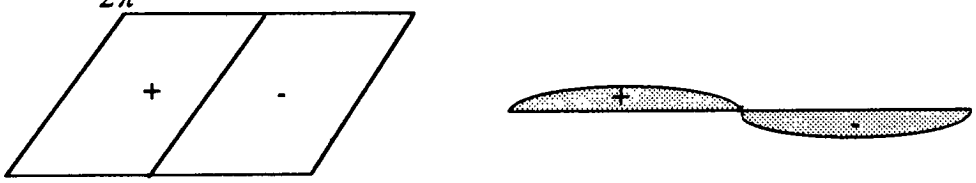
$$f_n = \frac{k_n}{2\pi}\sqrt{\frac{Dg}{wa^4}} = \frac{k_n}{2\pi}(305.434)\ \text{(cycle/sec ---> } H_z\text{)} \quad \text{..... Eq 3.9}$$

$$f_1 = \frac{36}{2\pi}(305.434) = 1750.008\ \text{(cycle/sec)} \quad \text{..... Mode 1}$$



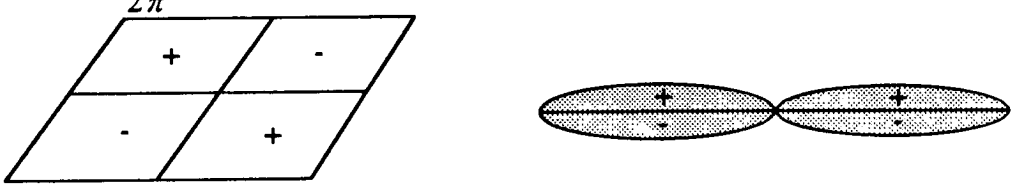
Mode 1
Figure 3.4

$$f_2 = \frac{73.4}{2\pi}(305.434) = 3568.07238\ \text{(cycle/sec)} \quad \text{..... Mode 2\&3}$$



Mode 2 & 3
Figure 3.5

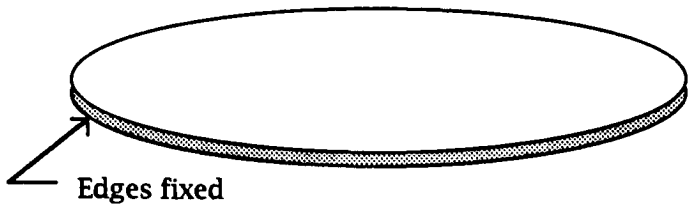
$$f_3 = \frac{108.3}{2\pi}(305.434) = 5264.608\ \text{(cycle/sec)} \quad \text{..... Mode 4\&5}$$



Mode 4 & 5
Figure 3.6

Mode 6
Figure 3.7

Circular flat plate all edges fixed



OD = 10" t = 0.5"

Material : Steel - E = 3e7 psi υ = .292

Figure 3.8

$$f_n = \frac{k_n}{2\pi}\sqrt{\frac{Dg}{wr^4}} \quad \text{..... Eq. 3.13}$$

where

k1 = 10.2 fundamental
k2 = 21.3 one nodal diameter
k3 = 34.9 two nodal diameter
k4 = 39.8 one nodal circle
$D = Et^3/12(1-\nu^2)$
g = gravity
w = wt./area = density x *thickness*
r = radius

So

$$D = \frac{Et^3}{12(1-\nu^2)} = \frac{3e7(0.5)^3}{12(1-(0.292)^2)} = 341628.6229 \text{ (lb.-in)} \quad \text{..... Eq. 3.14}$$

$$w = \rho t = 0.283(0.5) = 0.1415 \text{ (lb./in}^2\text{)} \quad \text{..... Eq. 3.15}$$

Therefore

$$\sqrt{\frac{Dg}{wr^4}} = \sqrt{\frac{341628.6228(386.4)}{0.1415(5)^4}} = 1221.736237 \text{ (1/in-sec)} \quad \text{..... Eq. 3.16}$$

$$f_n = \frac{k_n}{2\pi}\sqrt{\frac{Dg}{wr^4}} = \frac{k_n}{2\pi}(1221.736237) \text{ (cycle/sec ---> } H_Z) \text{ ..... Eq. 3.17}$$

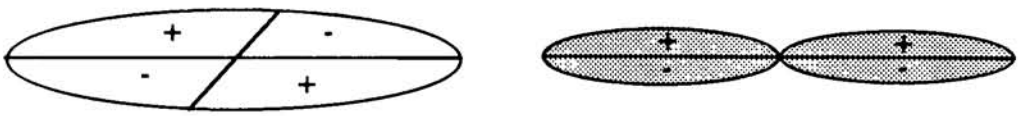
$$f_1 = \frac{10.2}{2\pi}(1221.736237) = 1983.34 \text{ (cycle/sec) ..... Mode 1}$$

Mode 1
Figure 3.9

$$f_2 = \frac{21.3}{2\pi}(1221.736237) = 4141.69 \text{ (cycle/sec) ..... Mode 2\&3}$$

Mode 2 & 3
Figure 3.10

$$f_3 = \frac{34.9}{2\pi}(1221.736237) = 6786.14 \text{ (cycle/sec) ..... Mode 4\&5}$$



Mode 4 & 5
Figure 3.11

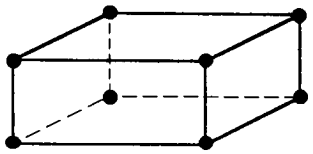
$$f_4 = \frac{39.8}{2\pi}(1221.736237) = 7738.93 \text{ (cycle/sec) ..... Mode 6}$$

Mode 6
Figure 3.12

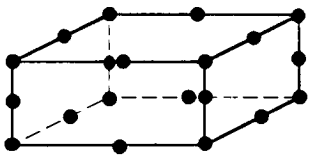
Chapter 4. FEA Results

All the geometries and FEA models are created by Aries Concept software. Ansys is used for the solver. All the tests will use four different types of elements which are shown below:

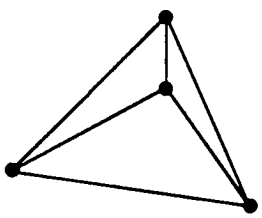
Type of Elements



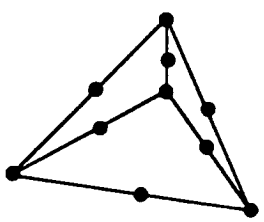
Linear Quadrilateral Element
(LQ)
8 nodes



Quadratic Quadrilateral Element
(QQ)
20 Nodes



Linear Tetrahedron Element
(LT)
4 Nodes



Qudaratic Tetrahedron Element
(QT)
10 Nodes

Figure 4.1 Type of elements

Cantilever (square) beam

Type of Element	# of Elements	# of Nodes	Displacement (inch)
LQ	40	99	0.13148
	320	525	0.13288
QQ	40	321	0.13272
	320	1865	0.13378
LT	71	52	0.03164
	532	207	0.087087
	703	248	0.079731
	4095	1061	0.11817
	7947	1848	0.12262
QT	71	224	0.1318
	532	1129	0.13363
	703	1408	0.13383

Table 4.1

- Figure 4.2 Maximum displacements w/ 40 LQ elements.
- Figure 4.3 Maximum displacements w/ 320 LQ elements.

Missing Page

Missing Page

- Figure 4.4 Maximum displacements w/ 40 QQ elements.
- Figure 4.5 Maximum displacements w/ 320 QQ elements.

Missing Page

Missing Page

- Figure 4.6 Maximum displacements w/ 71 LT elements.
- Figure 4.7 Maximum displacements w/ 532 LT elements.
- Figure 4.8 Maximum displacements w/ 703 LT elements.
- Figure 4.9 Maximum displacements w/ 4095 LT elements.
- Figure 4.10 Maximum displacements w/ 7947 LT elements.

Missing Page

- Figure 4.11 Maximum displacements w/ 71 QT elements.
- Figure 4.12 Maximum displacements w/ 532 QT elements.
- Figure 4.13 Maximum displacements w/ 703 QT elements.

Missing Page

Missing Page

Missing Page

Cantilever (round) beam

Type of Element	# of Elements	# of Nodes	Displacement (inch)
LQ	48	65	0.39421
	192	221	0.26144
	432	432	0.23948
	648	703	0.24373
QQ	48	229	0.22742
	192	841	0.22561
LT	77	53	0.23451
	328	138	0.16722
	659	253	0.16946
	1937	547	0.20261
	5977	1447	0.21293
QT	77	232	0.21433
	328	721	0.2256

Table 4.2

- Figure 4.14 Maximum displacements w/ 65 LQ elements.
- Figure 4.15 Maximum displacements w/ 221 LQ elements.
- Figure 4.16 Maximum displacements w/ 432 LQ elements.
- Figure 4.17 Maximum displacements w/ 703 LQ elements.

Missing Page

Missing Page

Missing Page

Missing Page

- Figure 4.18 Maximum displacements w/ 48 QQ elements.
- Figure 4.19 Maximum displacements w/ 192 QQ elements.

Missing Page

Missing Page

- Figure 4.20 Maximum displacements w/ 77 LT elements.
- Figure 4.21 Maximum displacements w/ 328 LT elements.
- Figure 4.22 Maximum displacements w/ 659 LT elements.
- Figure 4.23 Maximum displacements w/ 1937 LT elements.
- Figure 4.24 Maximum displacements w/ 5977 LT elements.

- Figure 4.25 Maximum displacements w/ 77 QT elements.
- Figure 4.26 Maximum displacements w/ 328 QT elements.

Square flat plate

Type of Element	# of Elements	# of Nodes	Mode 1 (Hz)	Mode 2 (Hz)	Mode 3 (Hz)	Mode 4 (Hz)	Mode 5 (Hz)
LQ	25	72	2078.5	5345.7	5356.5	9313.4	13734
	100	242	1762.6	3685.6	3692.1	5442.2	6956
	225	512	1730.4	3535.2	3537.2	5161.6	6412.4
	400	882	1720.4	3486.1	3487.9	5080.7	6240.1
QQ	25	228	1856	3905.6	3905.6	5825.6	7413.8
	100	803	1747.4	3515	3516.7	5108.2	6204.2
	225	1728	1731.5	3473.6	3475.2	5051.5	6103.3
LT	176	74	8311.9	1463	14507	17360	19556
	626	242	4018.1	7152.8	8398	11372	12308
	2567	914	3049.5	5621.3	6369.5	8833.3	9397.3
	19320	5062	2104.7	4201.8	4228.7	6125.5	7402.3
QT	176	393	2170.7	4731.5	4736	7528.6	8877.9
	626	626	1754.1	3532.8	3563.4	5204	6261.8
	2567	2567	1724.1	3478.9	3481.8	5104.5	6190.4

Table 4.3

- Figure 4.27-4.32 Square flat plate model and mode 1-5 mode shape w/ 25 LQ elements.
- Figure 4.33-4.38 Square flat plate model and mode 1-5 mode shape w/ 100 LQ elements.
- Figure 4.39-4.44 Square flat plate model and mode 1-5 mode shape w/ 225 LQ elements.
- Figure 4.45-4.50 Square flat plate model and mode 1-5 mode shape w/ 400 LQ elements.

- Figure 4.51-4.56 Square flat plate model and mode 1-5 mode shape w/ 25 QQ elements.
- Figure 4.57-4.62 Square flat plate model and mode 1-5 mode shape w/ 100 QQ elements.
- Figure 4.63-4.68 Square flat plate model and mode 1-5 mode shape w/ 225 QQ elements.

- Figure 4.69-4.74 Square flat plate model and mode 1-5 mode shape w/ 176 LT elements.
- Figure 4.75-4.80 Square flat plate model and mode 1-5 mode shape w/ 626 LT elements.
- Figure 4.81-4.86 Square flat plate model and mode 1-5 mode shape w/ 2567 LT elements.
- Figure 4.87-4.92 Square flat plate model and mode 1-5 mode shape w/ 19320 LT elements.

- Figure 4.93-4.98 Square flat plate model and mode 1-5 mode shape w/ 176 QT elements.
- Figure 4.99-4.104 Square flat plate model and mode 1-5 mode shape w/ 626 QT elements.
- Figure 4.105-4.110 Square flat plate model and mode 1-5 mode shape w/ 2567 QT elements.

Circular flat plate

Type of Element	# of Elements	# of Nodes	Mode 1 (Hz)	Mode 2 (Hz)	Mode 3 (Hz)	Mode 4 (Hz)	Mode 5 (Hz)	Mode 6 (Hz)
LQ	20	42	3699.3	12742	13289	15243	15808	17781
	40	82	2412.9	5249.4	5411.9	10177	10533	11580
	100	202	2102.2	4411.8	4418.2	7517.3	7530.8	8949.2
	140	282	2045	4242.1	4248.1	7047.7	7072.9	8194.4
	300	602	1990.6	4094.4	4099.8	6660.8	6669.6	7720.5
	600	1202	1973.6	4042.2	4042.6	6561.3	6563.4	7435.2
QQ	20	143	2158.1	4958.7	4966	8757	8862.3	10613
	40	283	2028.5	4202.5	4205.1	6917.1	6919.2	7823.2
	80	563	2009.2	4128.4	4129.8	6672.6	6675.7	7692.1
	140	983	1977.8	4039.8	4041.3	6502.4	6506.6	7420.9
	600	4203	1957.5	3995.6	3995.6	6434.9	6435.9	7320.2
LT	141	59	7768.4	15123	15434	17693	20043	20876
	209	86	6845.2	14532	15214	15428	15783	19387
	729	241	4531.7	8342.4	9319.9	13765	14440	14689
	2300	784	3606	7058.9	7298	11139	11298	12772
	16088	4133	2422.9	4921.1	4976.1	7956.8	7983.8	9019.8
QT	141	312	2246.1	5108.8	5250.2	9327.8	9562.8	11115
	209	464	2120	4578.1	4643.9	7795.4	7953.5	9095.7
	729	1394	1986	4077.4	4117.1	6638.2	6667.2	7590.7
	2300	4543	1963	4010.8	4023.1	6481.6	6500.4	7392.9
	3055	6248	1963.2	4025.3	4044.6	6555.3	6593.1	7521.7
	11021	18660	1952.3	3998	4007	6505.1	6513.9	7433.6

Table 4.4

- Figure 4.111-4.117 Circular flat plate model and mode 1-6 mode shape w/ 20 LQ elements.
- Figure 4.118-4.124 Circular flat plate model and mode 1-6 mode shape w/ 40 LQ elements.
- Figure 4.125-4.131 Circular flat plate model and mode 1-6 mode shape w/ 100 LQ elements.
- Figure 4.132-4.138 Circular flat plate model and mode 1-6 mode shape w/ 140 LQ elements.
- Figure 4.139-4.145 Circular flat plate model and mode 1-6 mode shape w/ 300 LQ elements.
- Figure 4.146-4.152 Circular flat plate model and mode 1-6 mode shape w/ 600 LQ elements.

- Figure 4.153-4.159 Circular flat plate model and mode 1-6 mode shape w/ 20 QQ elements.
- Figure 4.160-4.166 Circular flat plate model and mode 1-6 mode shape w/ 40 QQ elements.
- Figure 4.167-4.173 Circular flat plate model and mode 1-6 mode shape w/ 80 QQ elements.
- Figure 174-4.180 Circular flat plate model and mode 1-6 mode shape w/ 140 QQ elements.
- Figure 4.181-4.187 Circular flat plate model and mode 1-6 mode shape w/ 600 QQ elements.

- Figure 4.188-4.194 Circular flat plate model and mode 1-6 mode shape w/ 141 LT elements.
- Figure 4.195-4.201 Circular flat plate model and mode 1-6 mode shape w/ 209 LT elements.
- Figure 4.202-4.208 Circular flat plate model and mode 1-6 mode shape w/ 729 LT elements.
- Figure 4.209-4.215 Circular flat plate model and mode 1-6 mode shape w/ 2300 LT elements.
- Figure 4.216-4.222 Circular flat plate model and mode 1-6 mode shape w/ 16088 LT elements.

- Figure 4.223-4.229 Circular flat plate model and mode 1-6 mode shape w/ 141 QT elements.
- Figure 4.230-4.236 Circular flat plate model and mode 1-6 mode shape w/ 209 QT elements.
- Figure 4.237-4.244 Circular flat plate model and mode 1-6 mode shape w/ 729 QT elements.
- Figure 4.245-4.251 Circular flat plate model and mode 1-6 mode shape w/ 2300 QT elements.
- Figure 4.252-4.258 Circular flat plate model and mode 1-6 mode shape w/ 3055 QT elements.
- Figure 4.259-4.265 Circular flat plate model and mode 1-6 mode shape w/ 11021 QT elements.

Chapter 5. Discussion Results

In the following results, the "+" % of error means that the FEA results are stiffer than the analytical solutions. The "-" % of error means that the FEA results are weaker than the analytical solutions.

Cantilever (square) beam

From the FEA results shown, the linear tetrahedron element has most deviation in its performance. The percentage of errors range from -76.26% to -8.01% with 71 to 7947 elements. The quadratic quadrilateral element out-performs other types of elements with -0.44% and 0.36% error with 40 and 320 elements. The linear quadrilateral element has -1.37% and -0.32% error with 40 and 320 elements. The quadratic tetrahedron element gives errors of -1.13%, 0.25% and 0.4% with 71, 532 and 703 elements. See table 5.1 and figure 5.1 for details.

# of Elements vs. % of Error

# of elements	LQ	QQ	LT	QT
40	-1.37%			
320	-0.32%			
40		-0.44%		
320		0.36%		
71			-76.26%	
532			-34.67%	
703			-40.19%	
4095			-11.35%	
7947			-8.01%	
71				-1.13%
532				0.25%
703				0.40%

Table 5.1

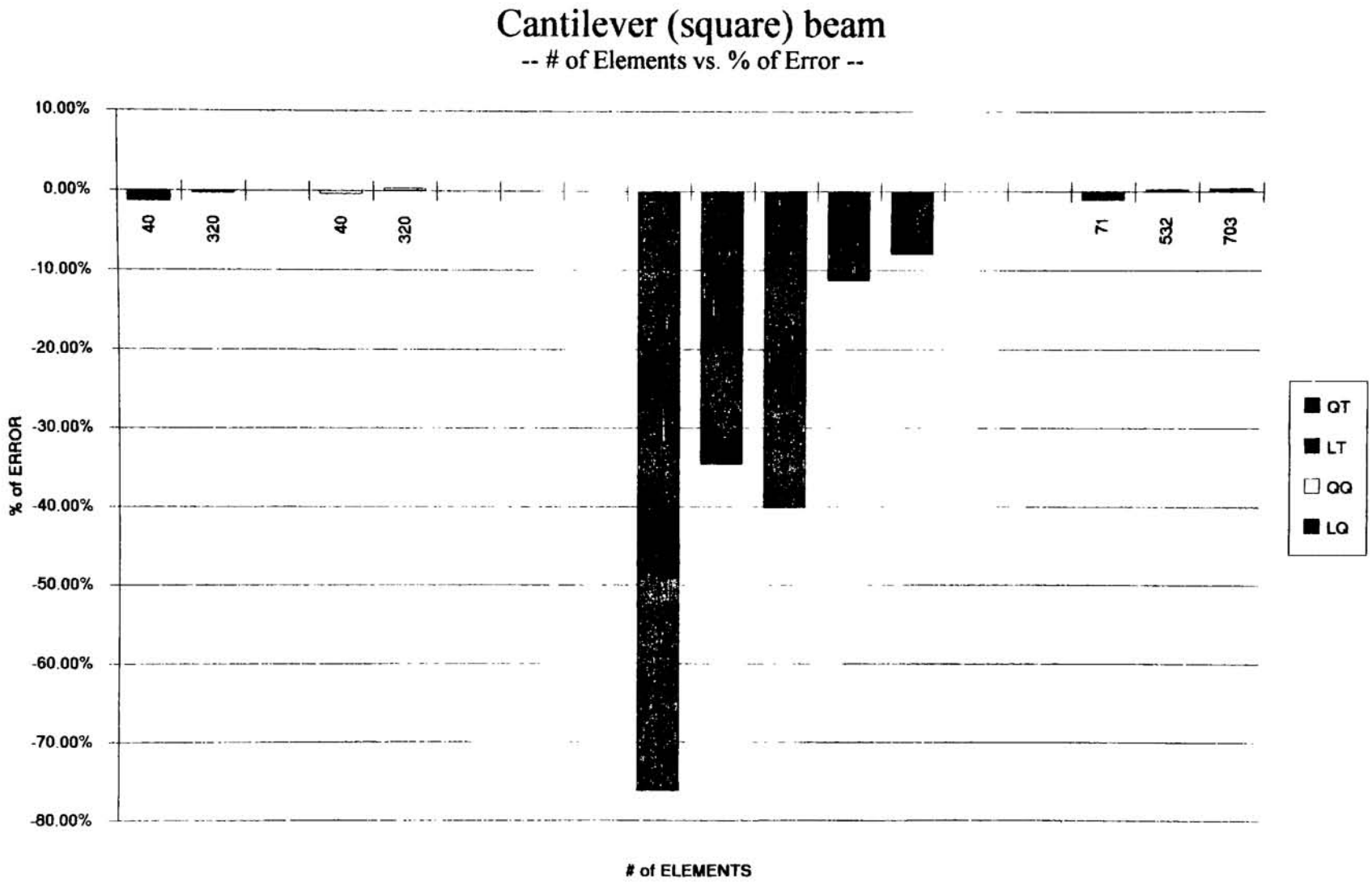

Cantilever (square) beam
-- # of Elements vs. % of Error --
10.00%
0.00%
-10.00%
-20.00%
-30.00%
-40.00%
-50.00%
-60.00%
-70.00%
-80.00%
% of ERROR
40
320
40
320
71
532
703
QT
LT
QQ
LQ
# of ELEMENTS

Figure 5.1

From another point of view, the quadratic quadrilateral element gives errors of -0.44% and 0.36% with 321 and 1865 nodes. The linear tetrahedron has the percentage of errors from -76.26% to -8.01% with 52 to 1848 nodes. The Linear quadrilateral element gives errors of -1.37% and -0.32% with 99 and 525 nodes. The quadratic tetrahedron element gives errors of -1.13%, 0.25% and 0.4% with 224, 1129 and 1408 nodes. See table 5.2 and figure 5.2 for details.

# of Nodes vs. % of Error

# of Nodes	LQ	QQ	LT	QT
99	-1.37%			
525	-0.32%			
321		-0.44%		
1865		0.36%		
52			-76.26%	
207			-34.67%	
248			-40.19%	
1061			-11.35%	
1848			-8.01%	
224				-1.13%
1129				0.25%
1408				0.40%

Table 5.2

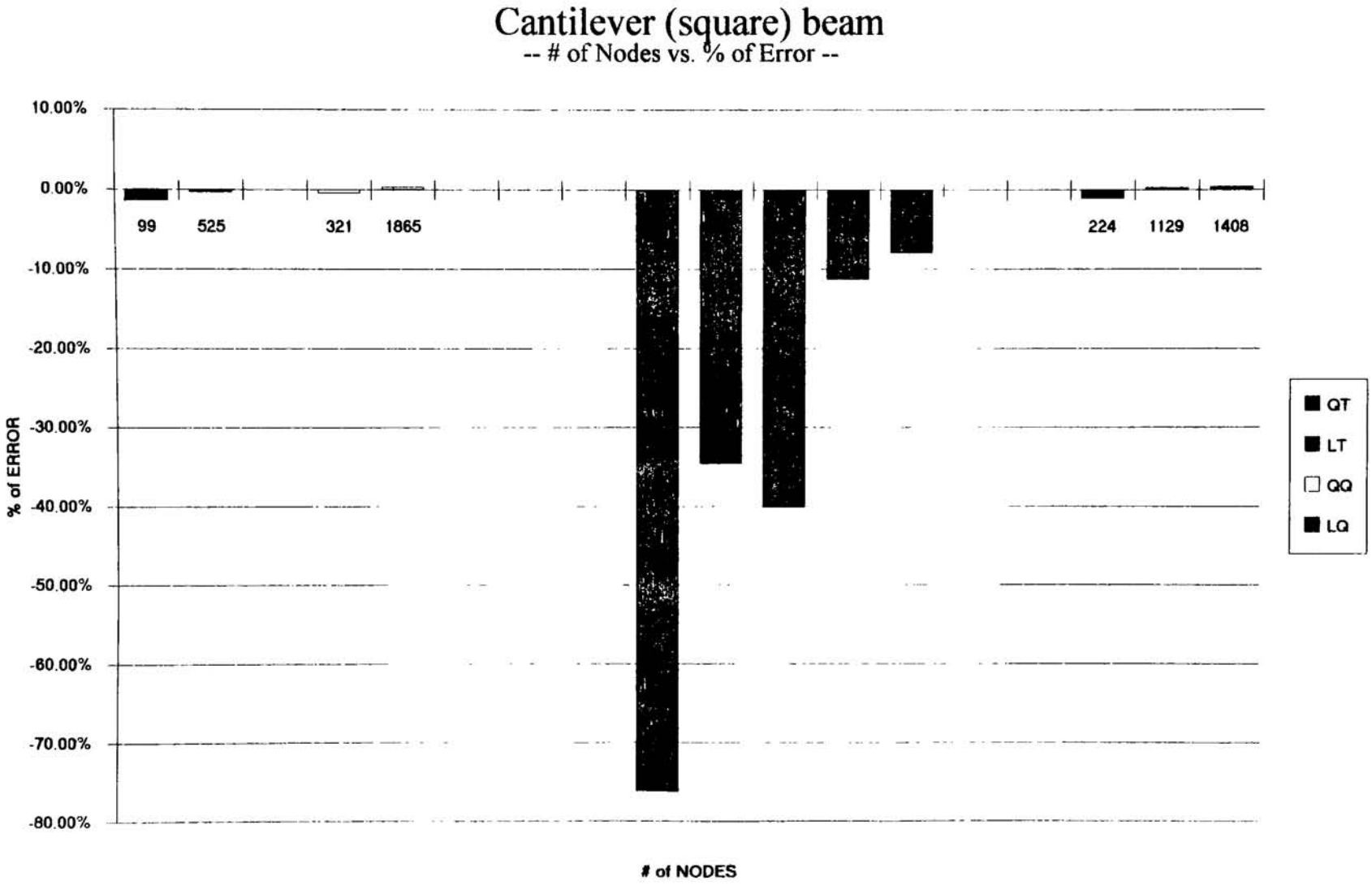

Cantilever (square) beam
-- # of Nodes vs. % of Error --
10.00%
0.00%
-10.00%
-20.00%
-30.00%
-40.00%
-50.00%
-60.00%
-70.00%
-80.00%
% of ERROR
99
525
321
1865
224
1129
1408
QT
LT
QQ
LQ
# of NODES

Figure 5.2

Cantilever (round) beam

From the FEA results shown, there is not much difference between the square beam and the round beam. The linear tetrahedron element has percentage of errors 3.6% and from -26.12% to -5.97% with 77 to 5977 elements. The quadratic quadrilateral element out-performs other types of elements, 0.47% and -0.33% error with 48 and 192 elements. The Linear quadrilateral element gives error range from 74.16% to 7.68% with number of elements from 48 to 648 elements. The quadratic tetrahedron element gives error of -5.31% and -0.33% with 77 and 328 elements. See table 5.3 and figure 5.3 for details.

# of Elements vs. % of Error

# of Elements	LQ	QQ	LT	QT
48	74.16%			
192	15.50%			
432	5.80%			
648	7.68%			
48		0.47%		
192		-0.33%		
77			3.60%	
328			-26.12%	
659			-25.13%	
1937			-10.49%	
5977			-5.93%	
77				-5.31%
328				-0.33%

Table 5.3

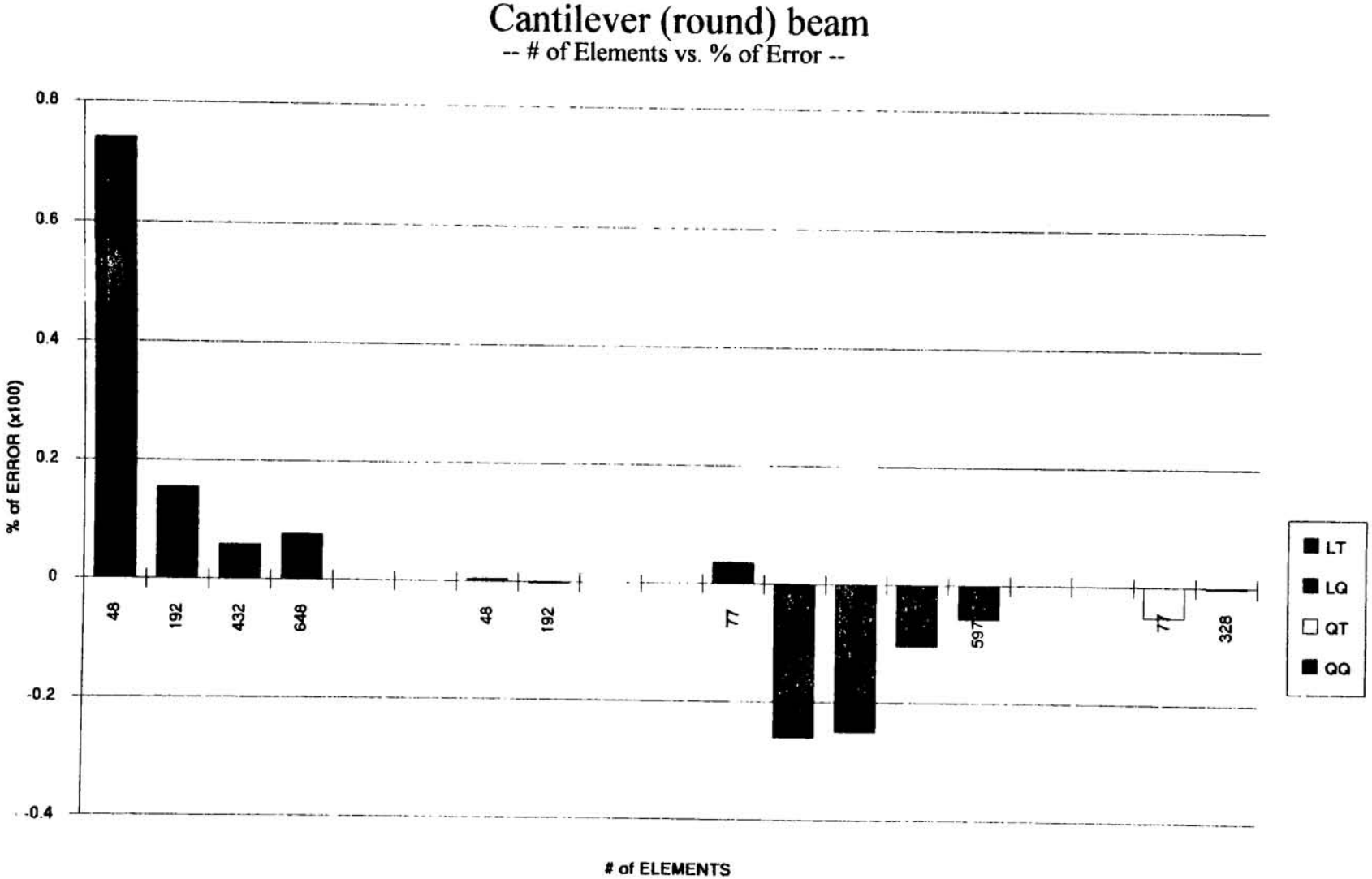

Cantilever (round) beam
-- # of Elements vs. % of Error --
0.8
0.6
0.4
0.2
0
-0.2
-0.4
% of ERROR (x100)
48
192
432
648
48
192
77
597
77
328
LT
LQ
QT
QQ
# of ELEMENTS

Figure 5.3

From another point of view, the quadratic quadrilateral element gives errors of 0.47% and -0.33% with 229 and 841 nodes. The linear tetrahedron has the percentage of errors 3.60% and from -26.12% to -5.91% with number of nodes from 53 to 1447 nodes. The Linear quadrilateral element gives errors of 74.16% to 5.80% with 65 and 703 nodes. The quadratic tetrahedron element has -5.31% and -0.33% deviation with 232 and 721 nodes. See table 5.4 and figure 5.4 for details.

# of Nodes vs. % of Error

# of Nodes	LQ	QQ	LT	QT
65	74.16%			
221	15.50%			
432	5.80%			
703	7.68%			
229		0.47%		
841		-0.33%		
53			3.60%	
138			-26.12%	
253			-25.13%	
547			-10.49%	
1447			-5.93%	
232				-5.31%
721				-0.33%

Table 5.4

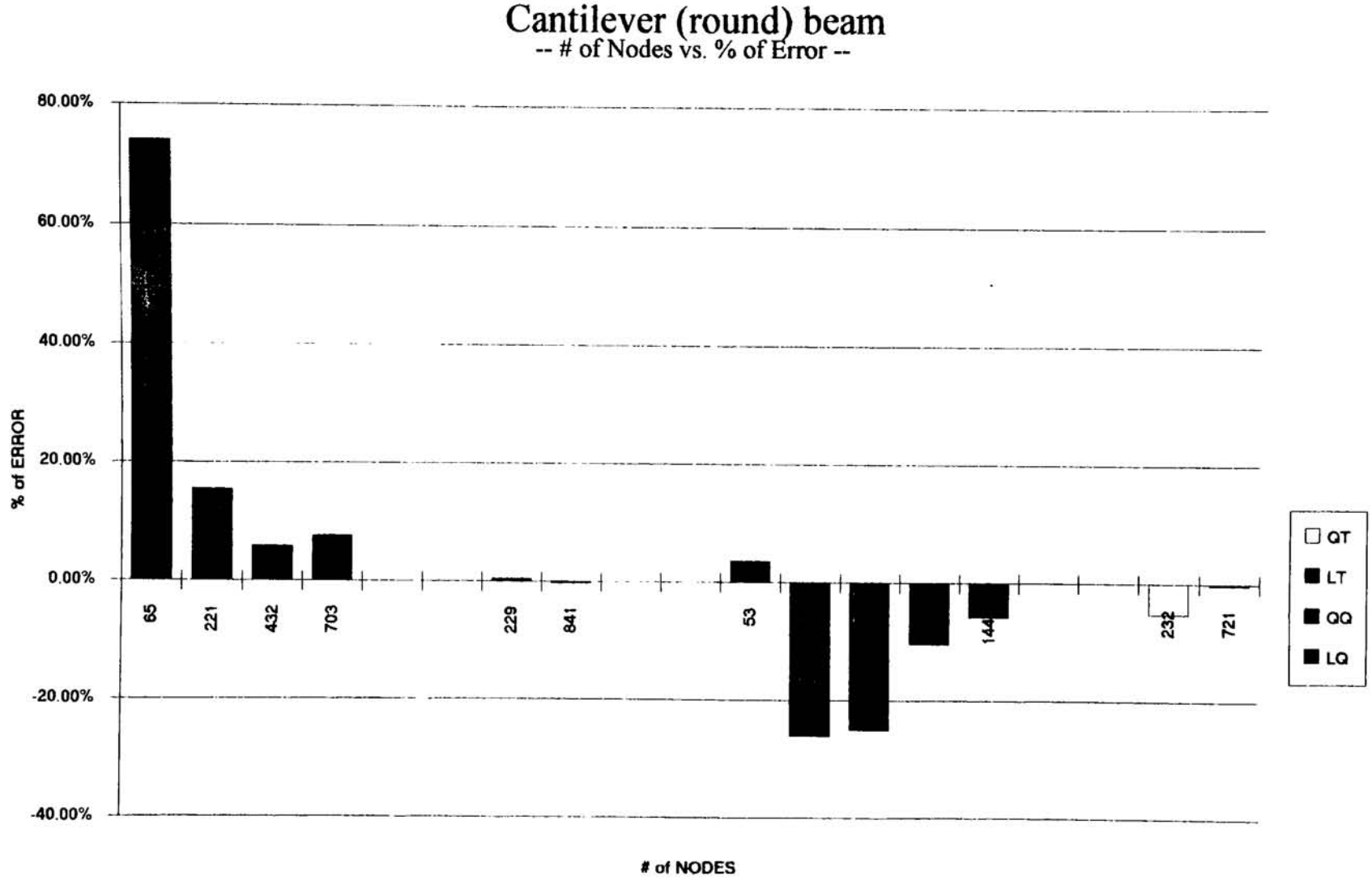

Cantilever (round) beam
-- # of Nodes vs. % of Error --
80.00%
60.00%
40.00%
20.00%
0.00%
-20.00%
-40.00%
% of ERROR
65
221
432
703
229
841
53
144
232
721
# of NODES
QT
LT
QQ
LQ

Figure 5.4

Missing Page

Linear Quadrilateral Element

-- # of Elements vs. % of Error --

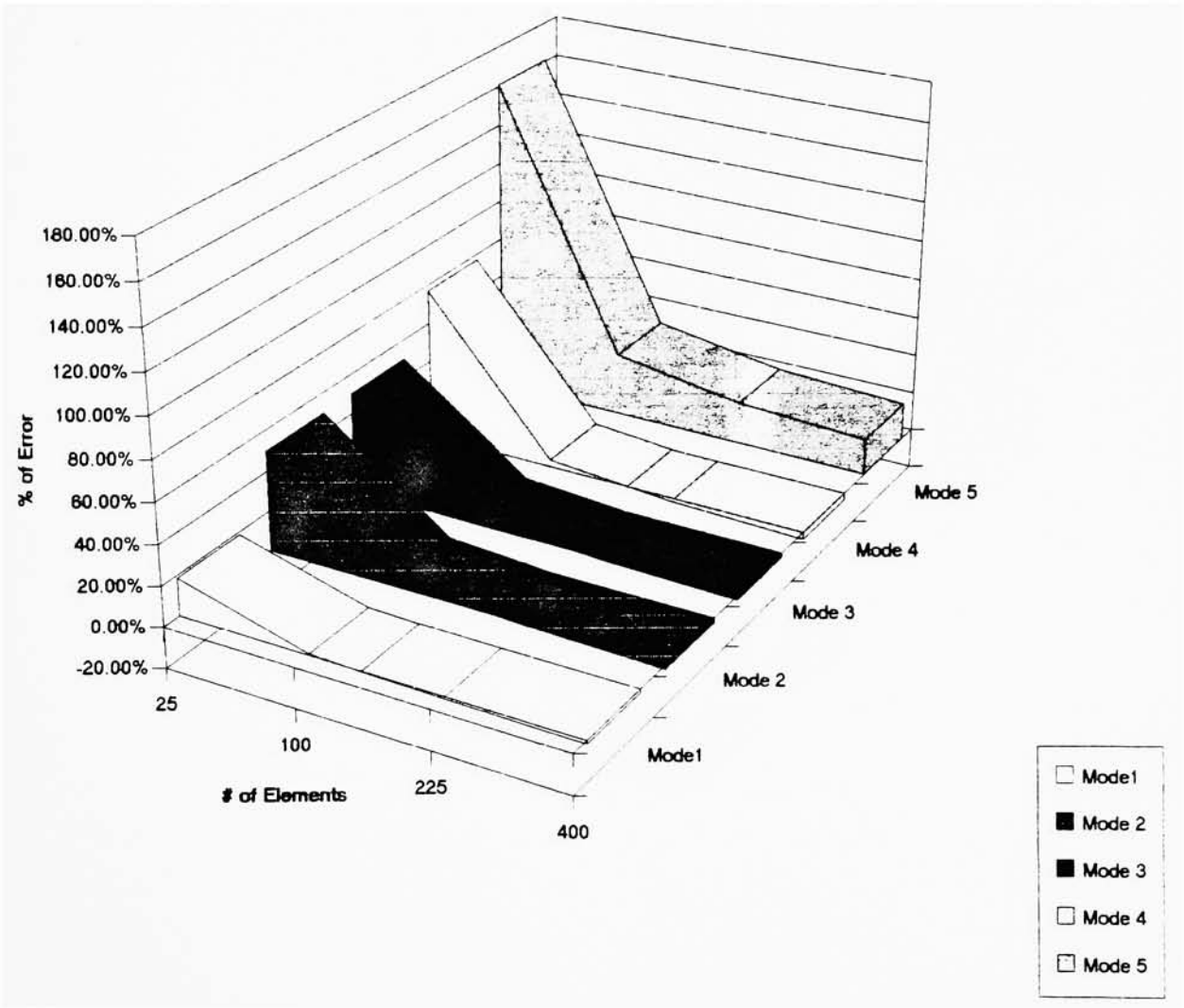


Figure 5.5

Linear Quadrilateral Element

-- # of Nodes vs % of Error --

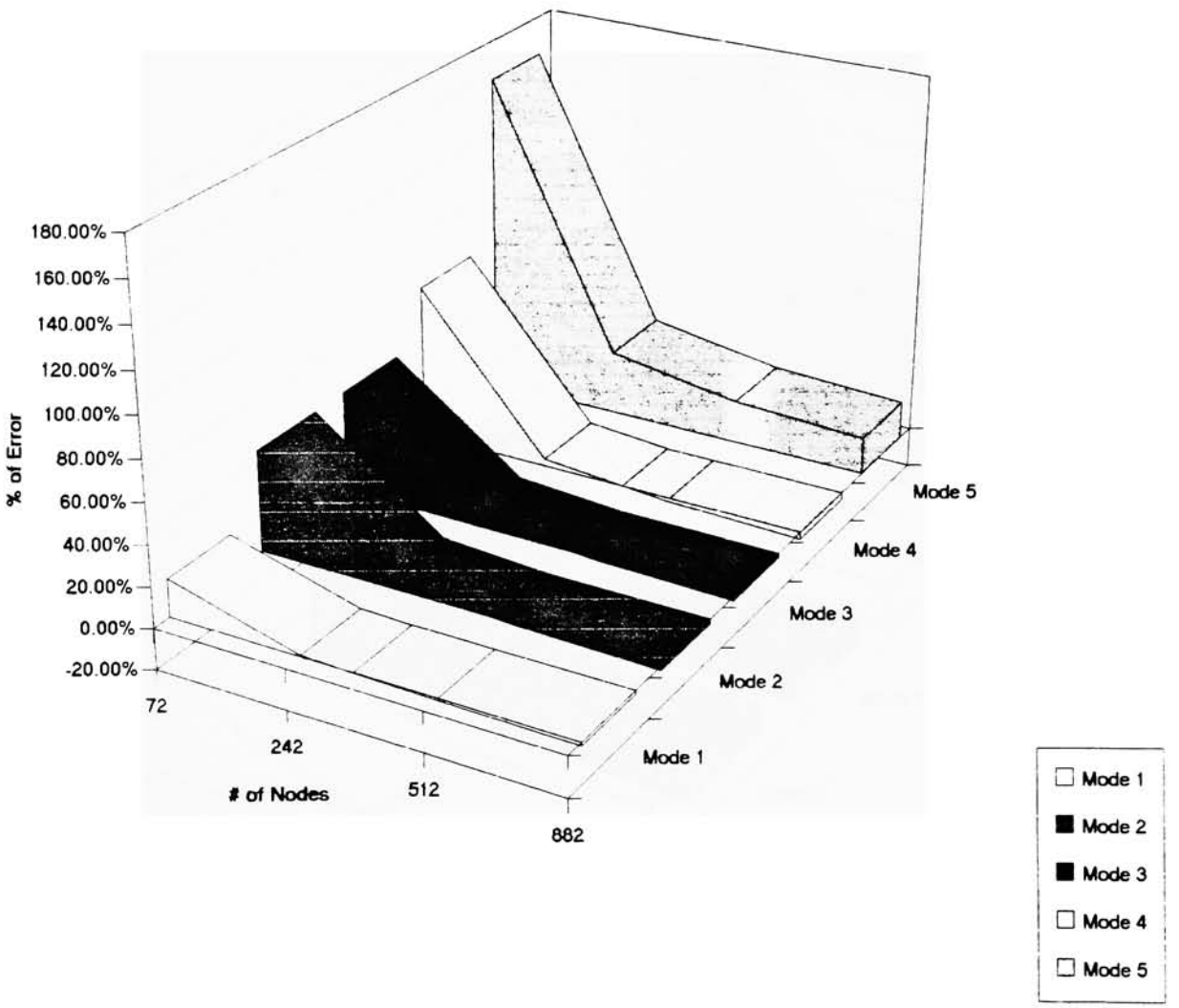


Figure 5.6
Page 324

Quadratic Quadrilateral Element

-- # of Nodes vs % of Error --

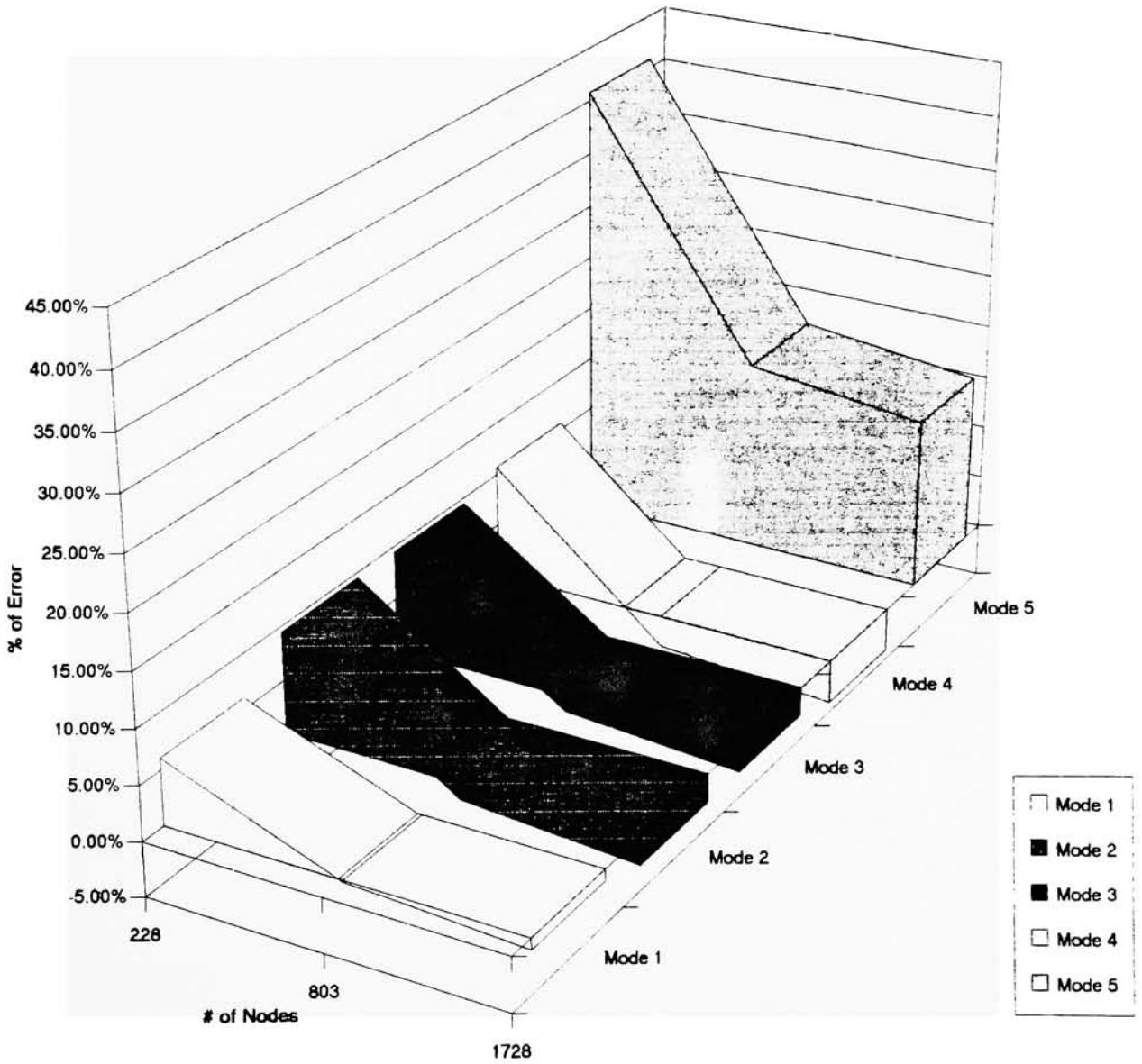


Figure 5.7

Quadratic Quadrilateral Element

-- # of Elements vs. % of Error --

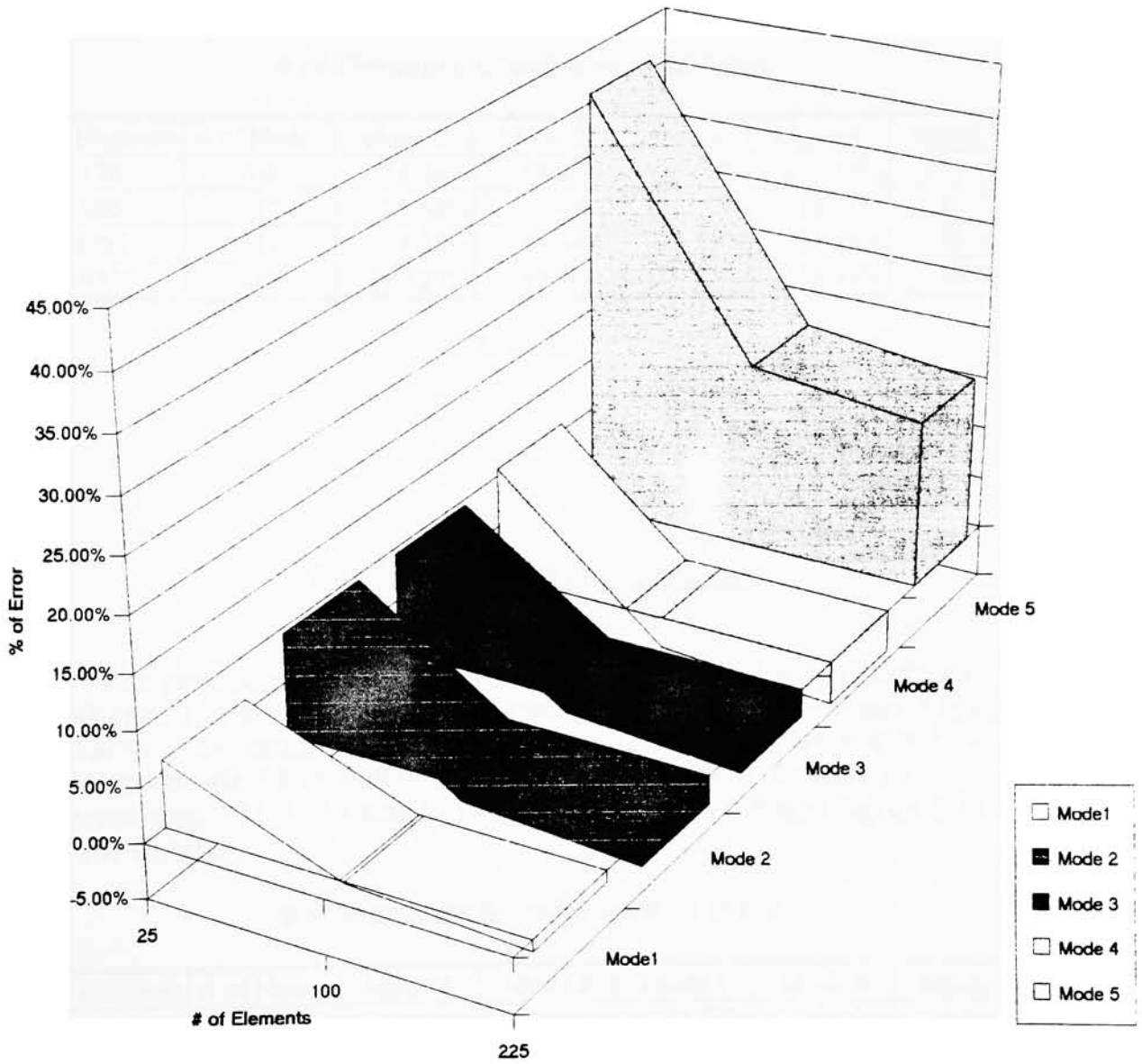


Figure 5.8

Linear Tetrahedron Element

The percentage of errors shown from FEA results by linear tetrahedron element are: mode 1 from 374.96% to 20.27% , mode 2 from 100.47% to -59%, mode 3 from 306.58% to 18.51%, mode 4 from 229.75% to 16.35% and mode 5 from 271.46% to 40.60% with 176, 626, 2567 and 19320 elements and 74, 242, 914 and 5062 nodes. See table 5.7 and figure 5.9 & 5.10 for details.

# of Elements & Nodes vs. % of Error

# of Elements	# of Nodes	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
176	74	374.96%	-59.00%	306.58%	229.75%	271.46%
626	242	129.60%	100.47%	135.37%	116.01%	133.79%
2567	914	74.26%	57.54%	78.51%	67.79%	78.50%
19320	5062	20.27%	17.76%	18.51%	16.35%	40.60%

Table 5.7

Quadratic Tetrahedron Element

The percentage of errors shown from FEA results by Quadratic Tetrahedron element are: mode 1 from 24.04% to -1.48% , mode 2 from 32.61% to -2.5%, mode 3 from 32.73% to -2.42%, mode 4 from 43% to -- 3.04% and mode 5 from 68.63% to 17.59% with 176, 626 and 2567 elements and 393, 1349 and 5274 nodes. See table 5.8 and figure 5.11 & 5.12 for details.

# of Elements & Nodes vs. % of Error

# of Elements	# of Nodes	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
176	393	24.04%	32.61%	32.73%	43.00%	68.63%
626	1349	0.23%	-0.99%	-0.13%	-1.15%	18.94%
2567	5274	-1.48%	-2.50%	-2.42%	-3.04%	17.59%

Table 5.8

Linear Tetrahedron Element

-- # of Elements vs. % of Error --

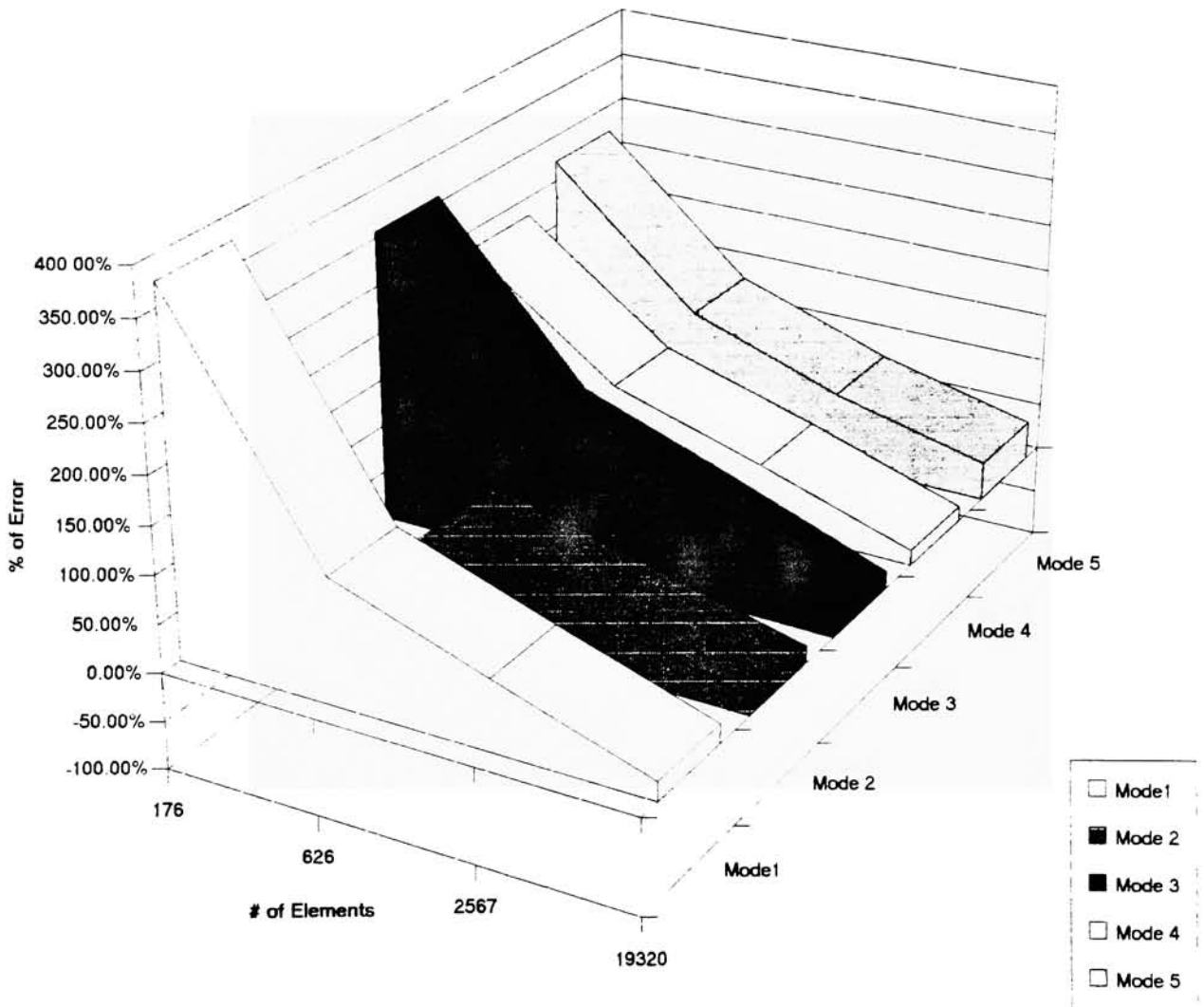


Figure 5.9

Linear Tetrahedron Element

-- # of Nodes vs % of Error --

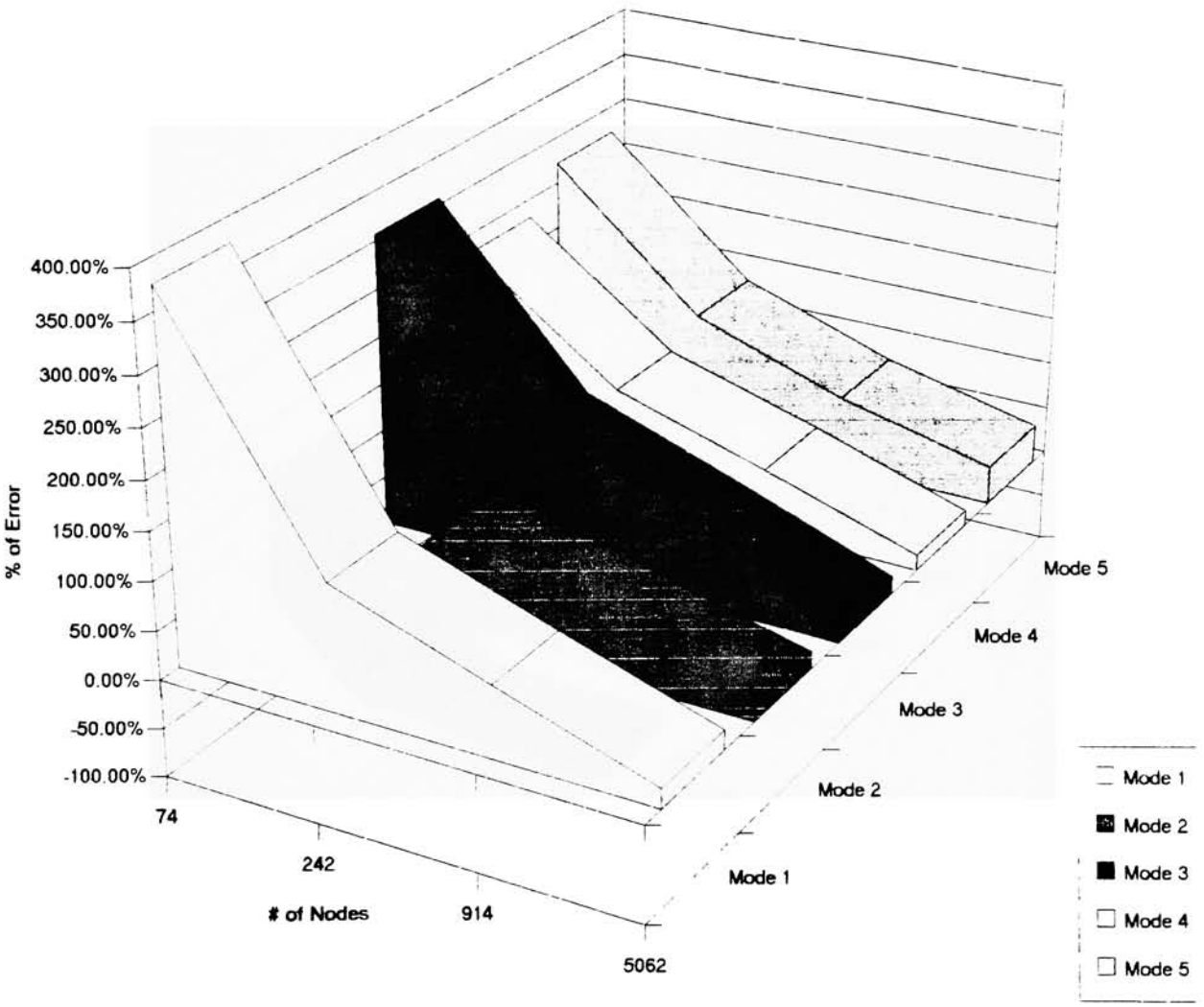


Figure 5.10

Quadratic Tetrahedron Element

-- # of Elements vs. % of Error --

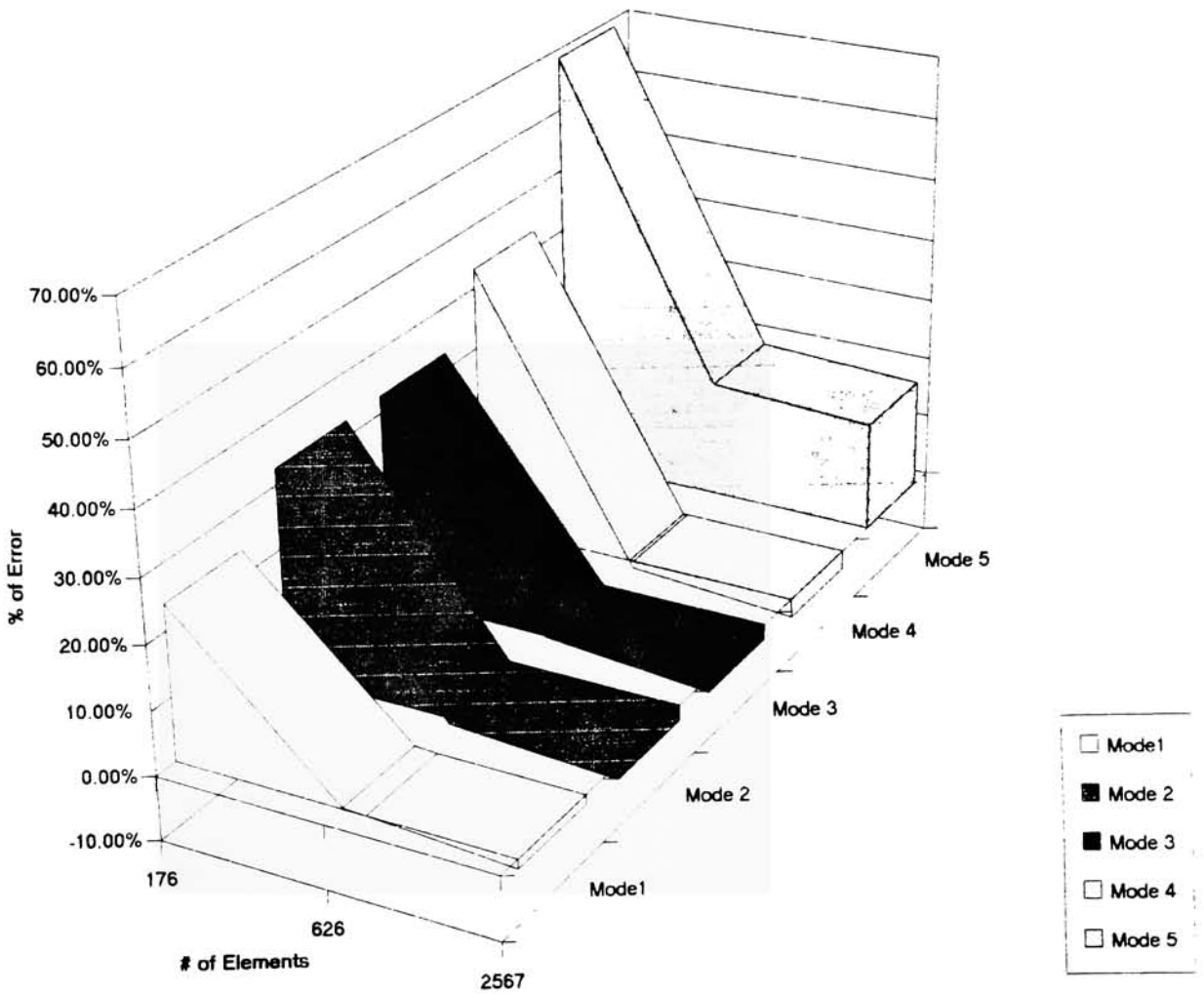


Figure 5.11

Quadratic Tetrahedron Element

-- # of Nodes vs % of Error --

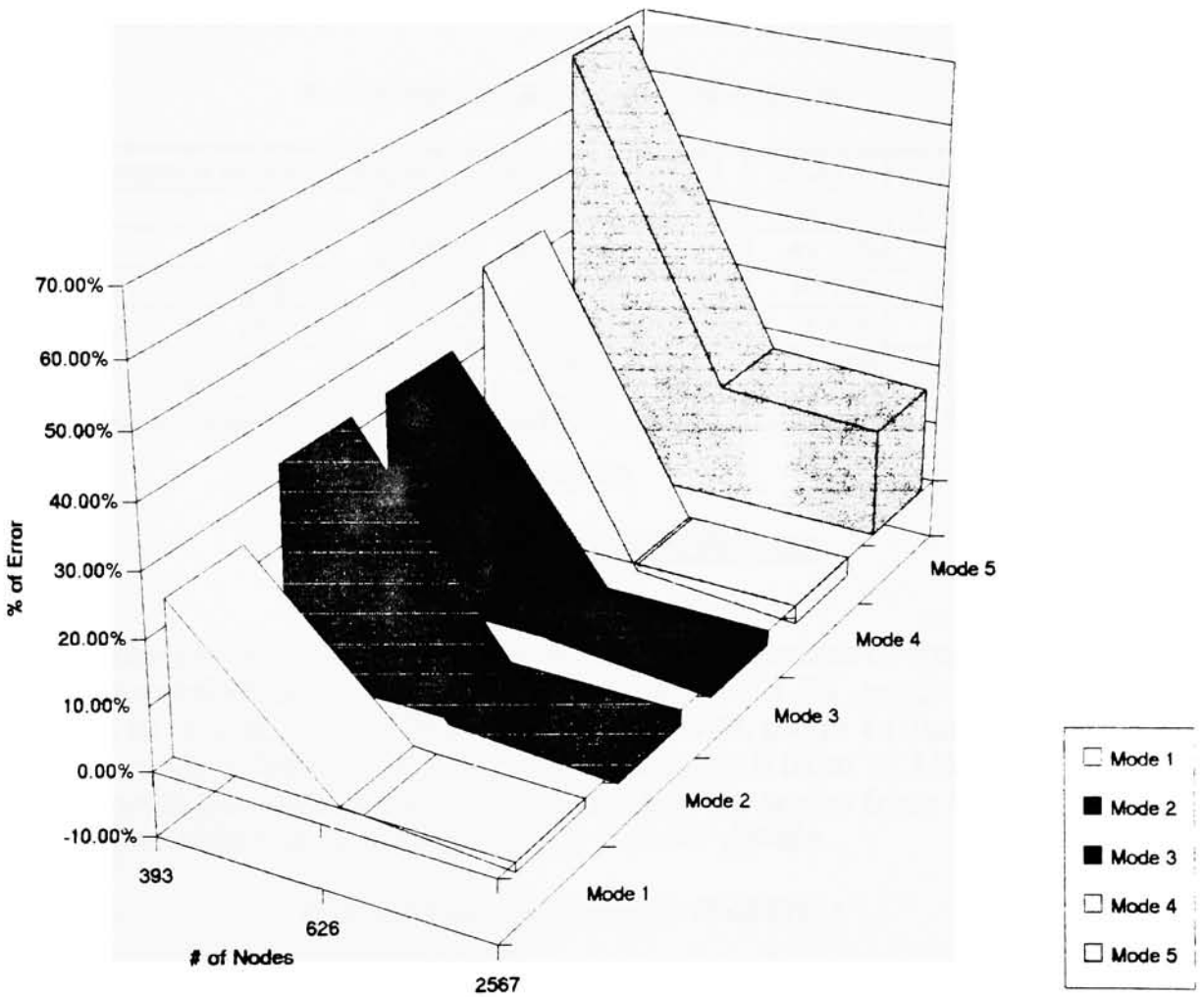


Figure 5.12

Circular flat plate

Linear Quadrilateral Element

The percentage of errors shown from FEA results by linear Quadrilateral element are: mode 1 from 86.52% to -0.49% , mode 2 from 207.65% to -2.4%, mode 3 from 220.86% to -2.39%, mode 4 from 124.62% to -3.31%, mode 5 from 132.95% to -3.28% and mode 6 from 129.76% to -3.92% with the elements from 20 to 600 and the nodes from 42 to 1202. See table 5.9 and figure 5.13 & 5.14 for details.

# of Elements & Nodes vs. % of Error

# of Elements	# of nodes	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6
20	42	86.52%	207.65%	220.86%	124.62%	132.95%	129.76%
40	82	21.66%	26.75%	30.67%	49.97%	55.21%	49.63%
100	202	5.99%	6.52%	6.68%	10.77%	10.97%	15.64%
140	282	3.11%	2.42%	2.57%	3.85%	4.23%	5.89%
300	602	0.37%	-1.14%	-1.01%	-1.85%	-1.72%	-0.24%
600	1202	-0.49%	-2.40%	-2.39%	-3.31%	-3.28%	-3.92%

Table 5.9

Quadratic Quadrilateral Element

The percentage of errors shown from FEA results by quadratic quadrilateral element are: mode 1 from 8.81% to -1.3% , mode 2 from 19.73% to -3.53%, mode 3 from 19.90% to --3.53%, mode 4 from 29.04% to -5.18%, mode 5 from 30.59% to -5.16% and mode 6 from 37.14% to -5.41% with the elements from 20 to 600 and the nodes from 143 to 4203. See table 5.10 and figure 5.15 & 5.16 for details.

# of Elements & Nodes vs. % of Error

# of Elements	# of nodes	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6
20	143	8.81%	19.73%	19.90%	29.04%	30.59%	37.14%
40	283	2.28%	1.47%	1.53%	1.93%	1.96%	1.09%
80	563	1.30%	-0.32%	-0.29%	-1.67%	-1.63%	-0.61%
140	983	-0.28%	-2.46%	-2.42%	-4.18%	-4.12%	-4.11%
600	4203	-1.30%	-3.53%	-3.53%	-5.18%	-5.16%	-5.41%

Table 5.10

Linear Quadrilateral Element

-- # of Elements vs. % Error --

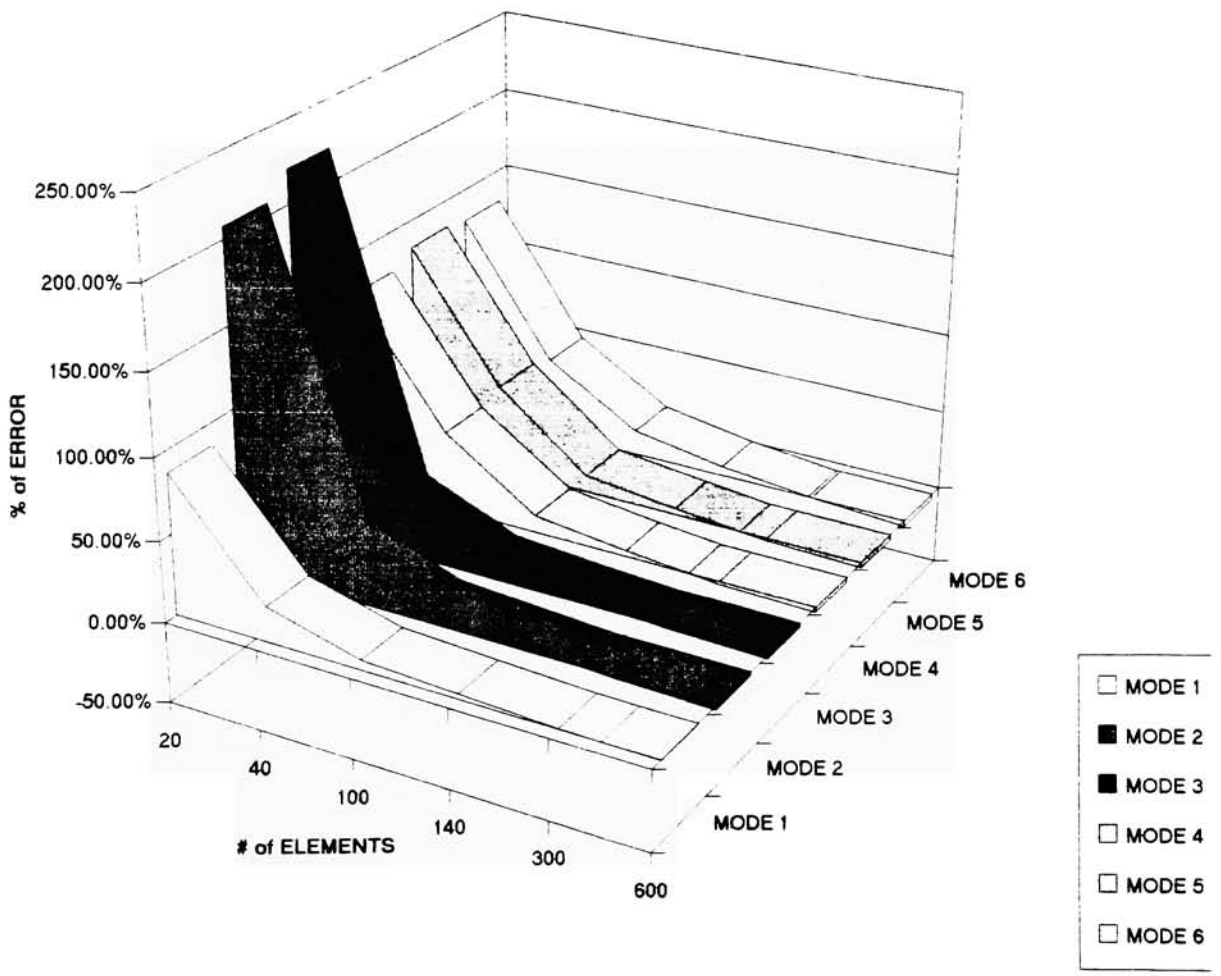


Figure 5.13

Linear Quadrilateral Element
-- # of Nodes vs. % of Error --

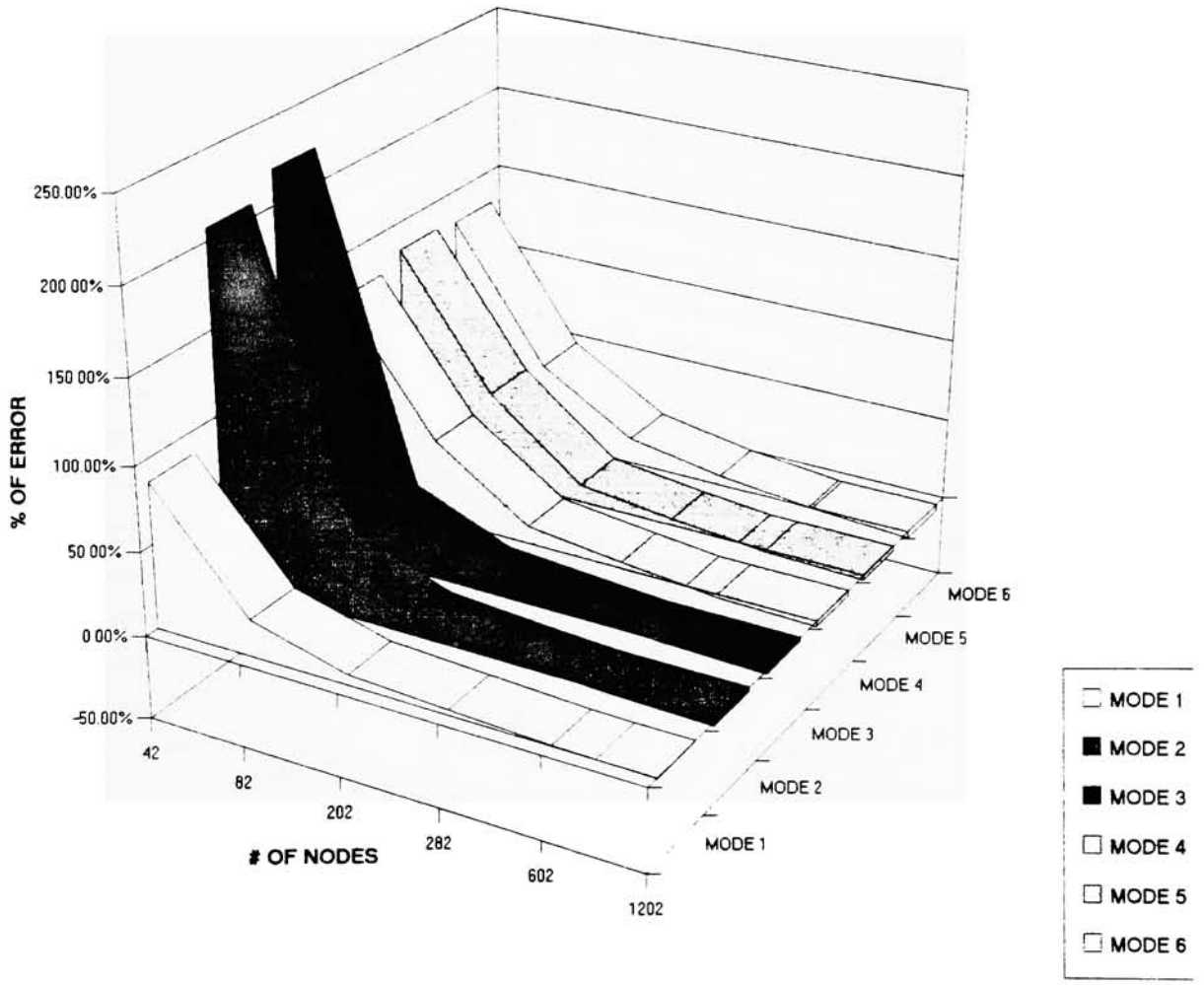


Figure 5.14

Quadratic Quadrilateral Element

-- # of Elements vs. % Error --

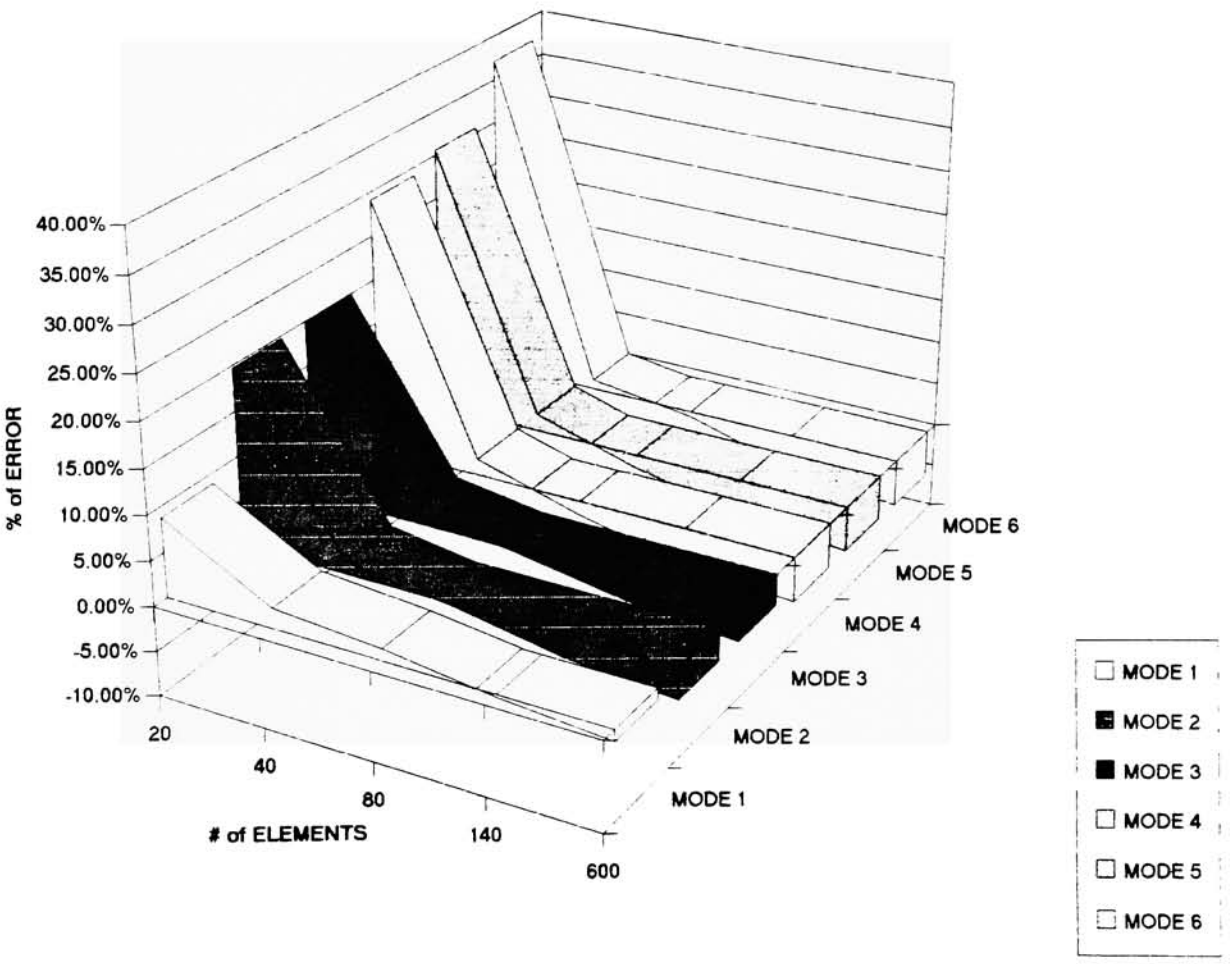


Figure 5.15

Quadratic Quadrilateral Element
-- # of Nodes vs. % of Error --

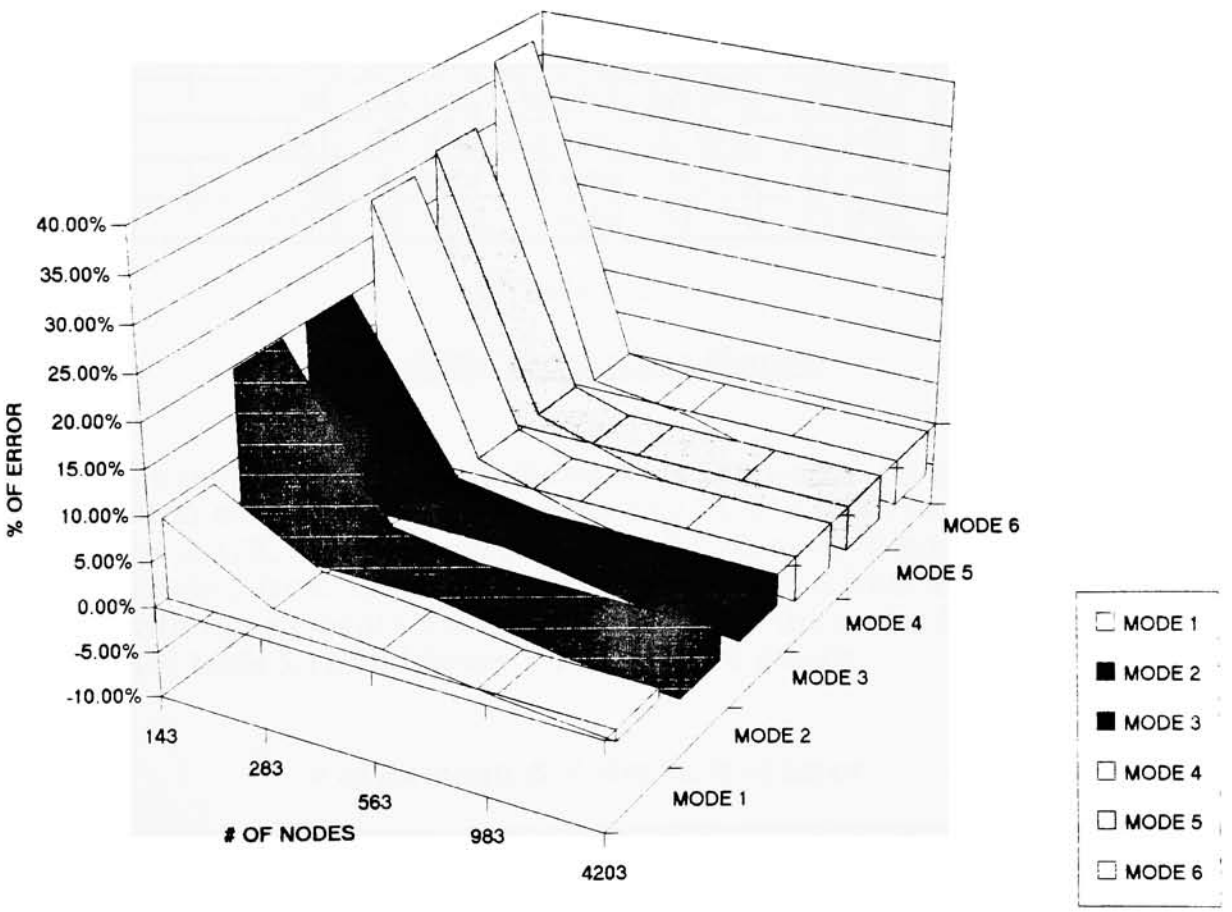


Figure 5.16

Linear Tetrahedron Element

The percentage of errors shown from FEA results by linear tetrahedron element are: mode 1 from 291.68% to 22.16% , mode 2 from 265.14% to 18.82%, mode 3 from 272.65% to -20.15%, mode 4 from 160.72% to 17.25%, mode 5 from 195.35% to 17.65% and mode 6 from 169.75% to -5.41% with the elements from 141 to 16088 and the nodes from 59 to 4133. See table 5.11 and figure 5.17 & 5.18 for details.

# of Elements & Nodes vs. % of Error

# of Elements	# of nodes	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6
141	59	291.68%	265.14%	272.65%	160.72%	195.35%	169.75%
209	86	245.13%	250.87%	267.34%	127.35%	132.58%	150.51%
729	241	128.49%	101.43%	125.03%	102.84%	112.79%	89.81%
2300	784	81.81%	70.44%	76.21%	64.14%	66.49%	65.04%
16088	4133	22.16%	18.82%	20.15%	17.25%	17.65%	16.55%

Table 5.11

Quadratic Tetrahedron Element

The percentage of errors shown from FEA results by quadratic tetrahedron element are: mode 1 from 13.25% to -1.57% , mode 2 from 23.35% to -3.47%, mode 3 from 26.76% to -3.25%, mode 4 from 37.45% to -4.14%, mode 5 from 40.92.35% to -4.01% and mode 6 from 43.62% to -3.95% with the elements from 141 to 11021 and the nodes from 312 to 18660. See table 5.12 and figure 5.19 & 5.20 for details.

# of Elements & Nodes vs. % of Error

# of Elements	# of nodes	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6
141	312	13.25%	23.35%	26.76%	37.45%	40.92%	43.62%
209	464	6.89%	10.54%	12.13%	14.87%	17.20%	17.53%
729	1394	0.13%	-1.55%	-0.59%	-2.18%	-1.75%	-1.92%
2300	4543	-1.03%	-3.16%	-2.86%	-4.49%	-4.21%	-4.47%
3055	6248	-1.02%	-2.81%	-2.34%	-3.40%	-2.84%	-2.81%
11021	18660	-1.57%	-3.47%	-3.25%	-4.14%	-4.01%	-3.95%

Table 5.12

Linear Tetrahedron Element

-- # of Elements vs. % Error --

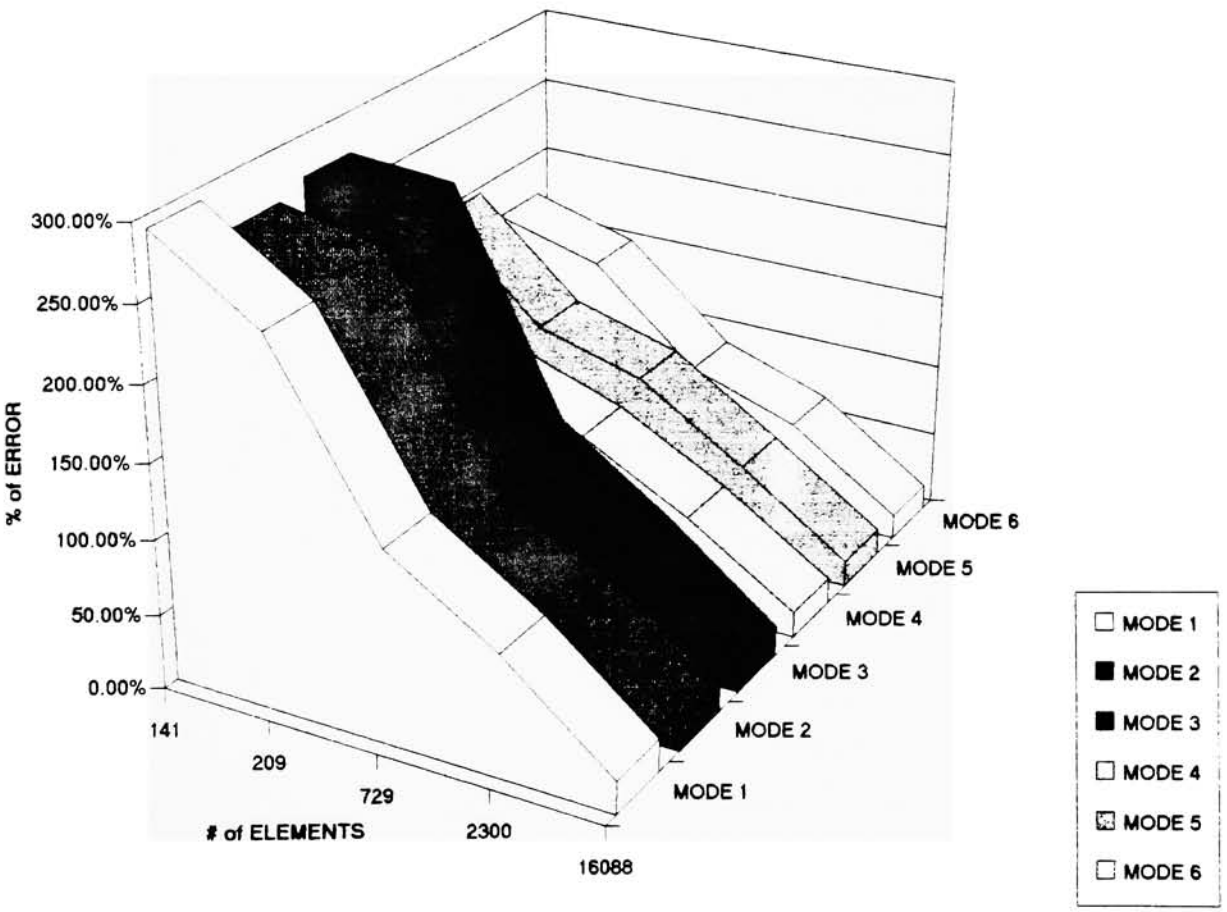


Figure 5.17

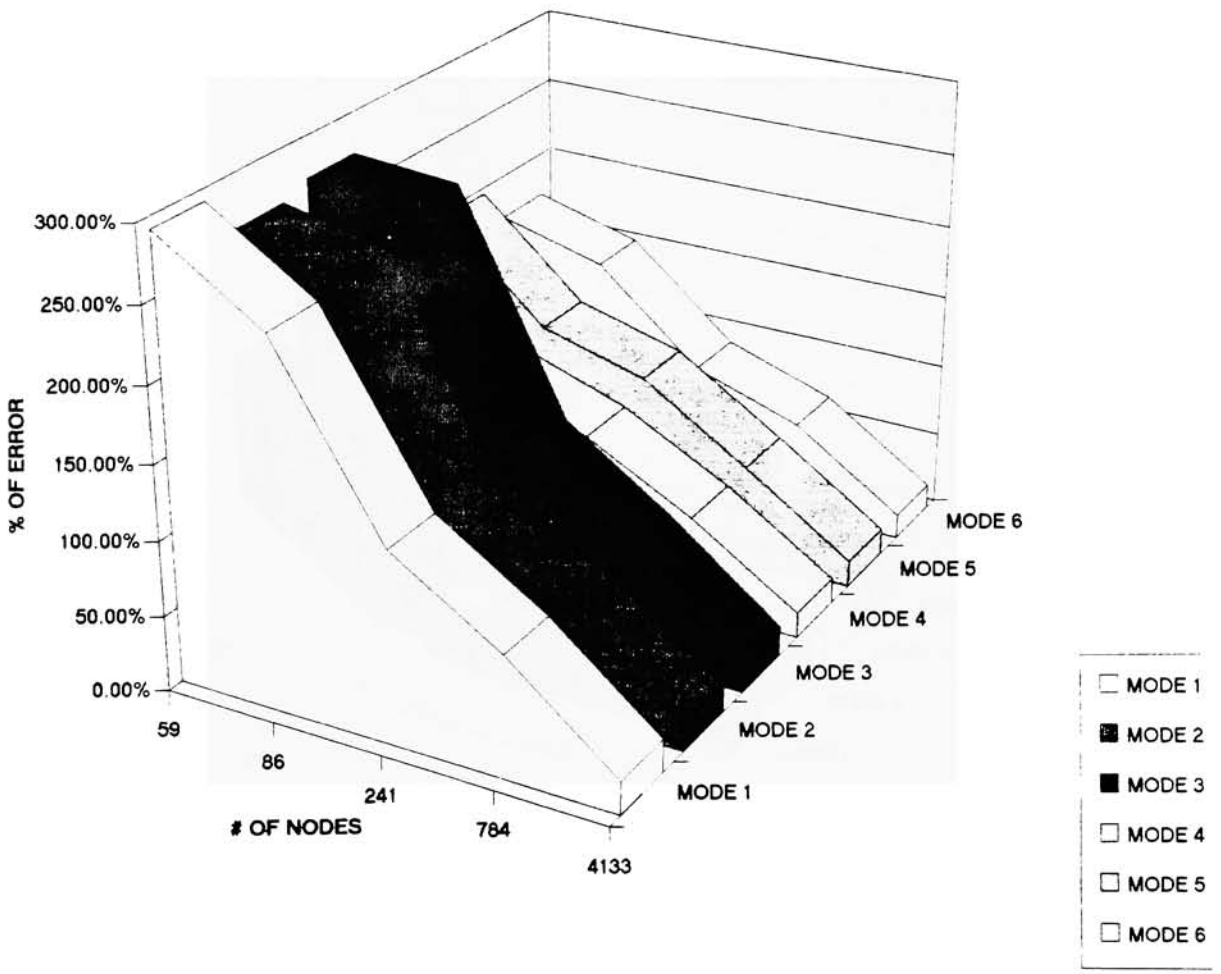

Linear Tetrahedron Element
-- # of Nodes vs. % of Error --
% OF ERROR
300.00%
250.00%
200.00%
150.00%
100.00%
50.00%
0.00%
59
86
241
784
4133
OF NODES
MODE 1
MODE 2
MODE 3
MODE 4
MODE 5
MODE 6
MODE 1
MODE 2
MODE 3
MODE 4
MODE 5
MODE 6

Figure 5.18

Quadratic Tetrahedron Element

-- # of Elements vs. % Error --

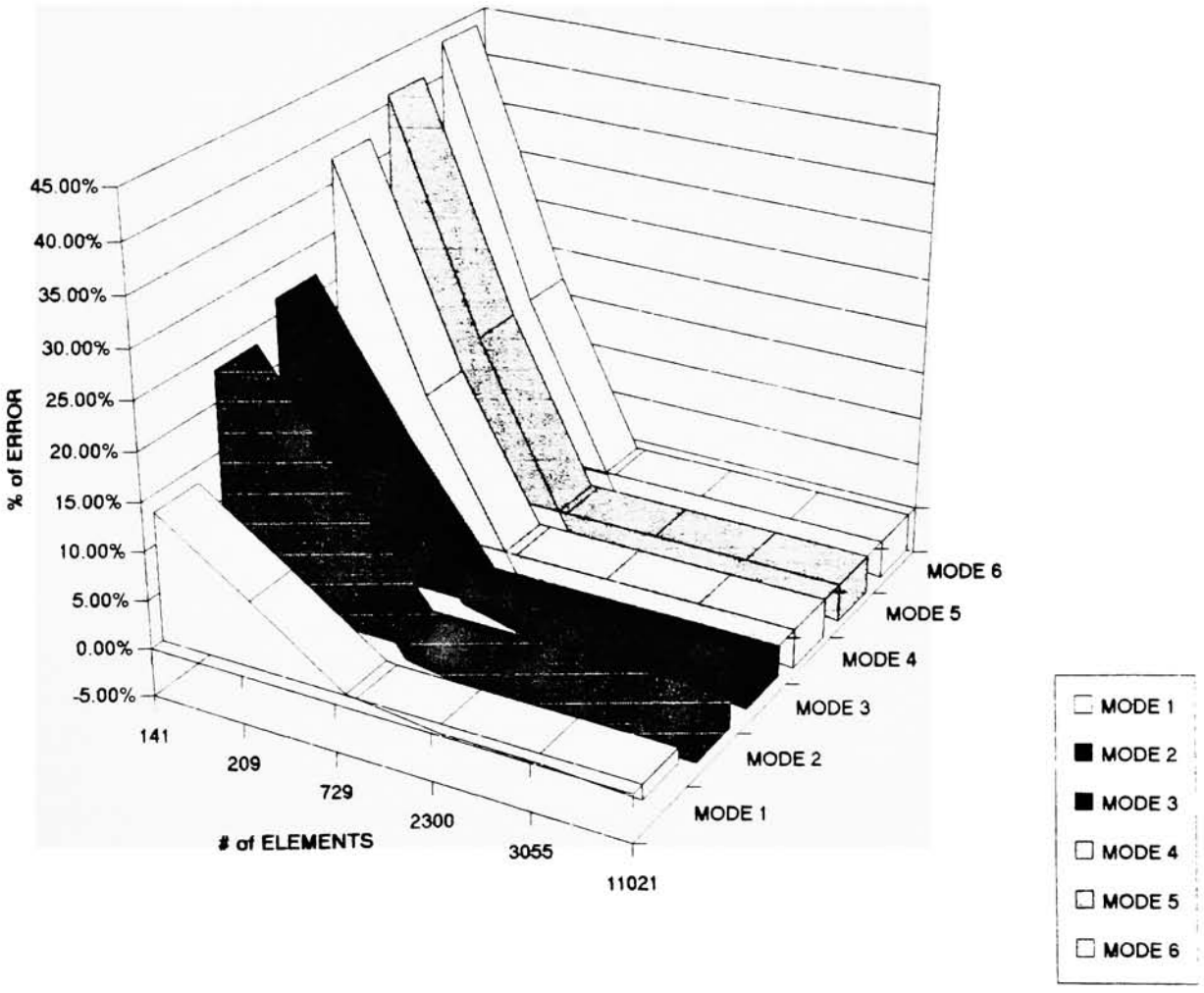


Figure 5.19

Quadratic Tetahedron Element
-- # of Nodes vs. % of Error --

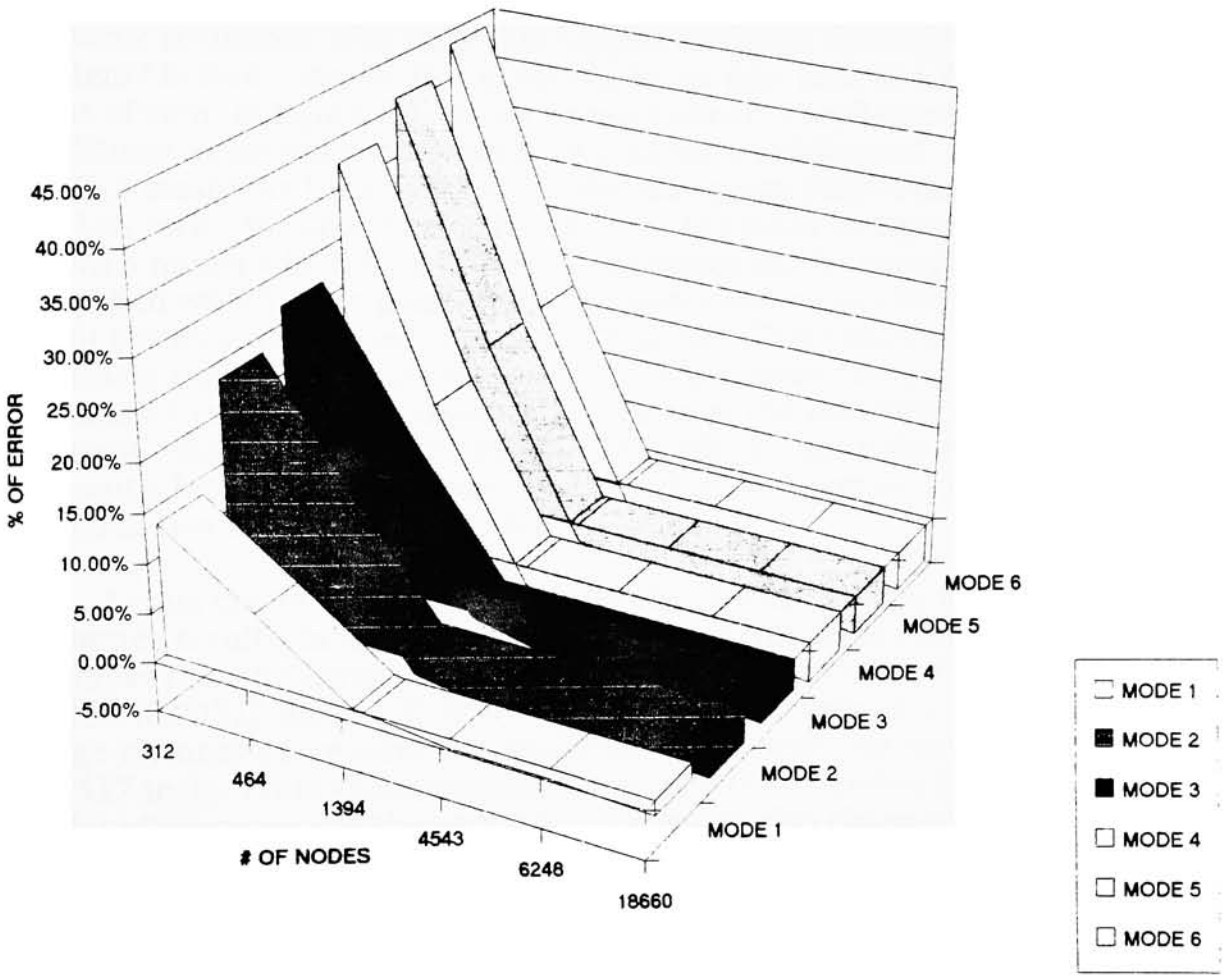


Figure 5.20

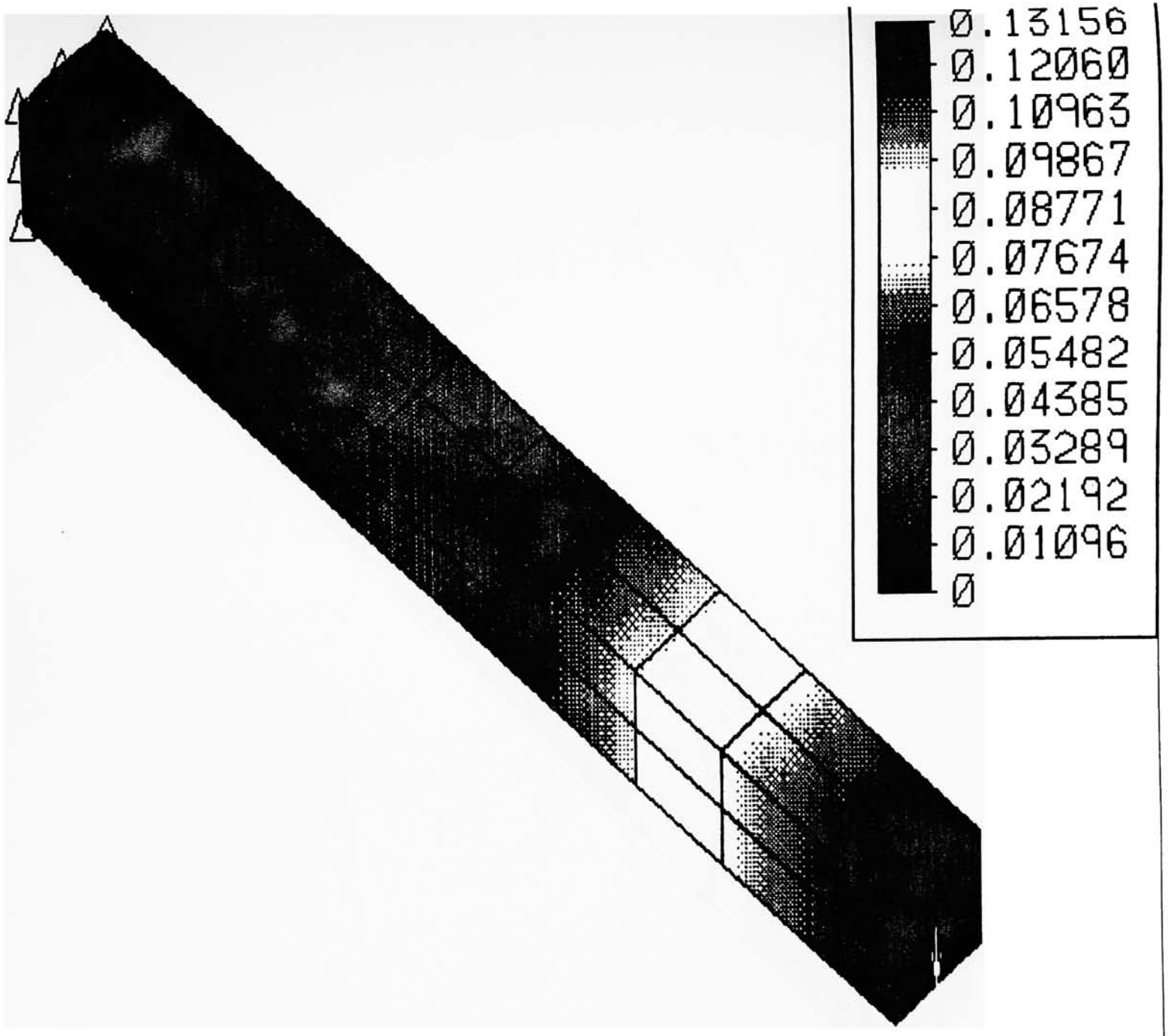
Missing Page

original geometry. An increase in the number of elements in cross-section of the enclosed model is required (close to original geometry). Why does this situation not appear in FEA model created by quadratic elements? The quadratic element has 3 nodes on each element edge unlike the linear element which has only 2 nodes at each element edge. The mid-point node can follow the geometric profile. In other words, this means that the edge of a quadratic element can be curved. That's why the quadratic element does not have such problems associated with linear elements.

The linear tetrahedron element's errors start from a larger negative number, increasing and closing to 0% . This situation is seen in square and round beam models (even quadratic tetrahedron element has the same situation). Why does this happen different than quadrilateral element? Before I answer this question, let us take look at a different point of view. In figure 6.1, which shows a square cantilever beam (same conditions as shown above) mesh with linear quadrilateral elements (10x2x2 mesh). As the results show, the maximum displacement is 0.13156 inch (the analytical solution is 0.1333 inch). In figure 6.2 is shown a model which is the same as a previous model except this model is meshed with 6 node point wedge elements and more elements. The result shows a displacement of 0.09376 inches. This result is stiffer than the linear quadrilateral model in which we increased the number of elements 4 times (same number of nodes) over the previous model and the result did not get better. In figure 6.3, which rotate the wedge element's face from top to side, the result becomes worse than in the previous two models (0.04693 inch).

Let us create a different mesh for the model. We may attempt to get better results. In figure 6.4 the wedge elements were reoriented and it follows that the result is not much different from there of previous models (0.09315 inch vs. 0.09376) in figure 6.3. In figure 6.5, in which the wedge element's face were changed from top to side, the result shown is 0.05517 inch. From these examples, once the cross-section face of FEA model of elements are changed, then the result also changed dramatically. Ideally the quadrilateral element will be the best element to use. Once we change to wedge elements, even the model has more elements than the model with linear quadrilateral elements. The result is stiffer than the actual solution. Also, if we rotate the element face from top to side, the results appear much stiffer than it actually was. Here, we can draw a conclusion which states that the orientation of the element mesh face affects the FEA result.

Let us take a look at an element's shape. The linear quadrilateral element has 8 nodes, the linear wedge element has 6 nodes and the



SVIEWH 3.14 File:beam3 92/03/02 15:20 LC 1/ 1 Vu= 7 Lo= 45 La= 45 R= 0

Figure 6.1

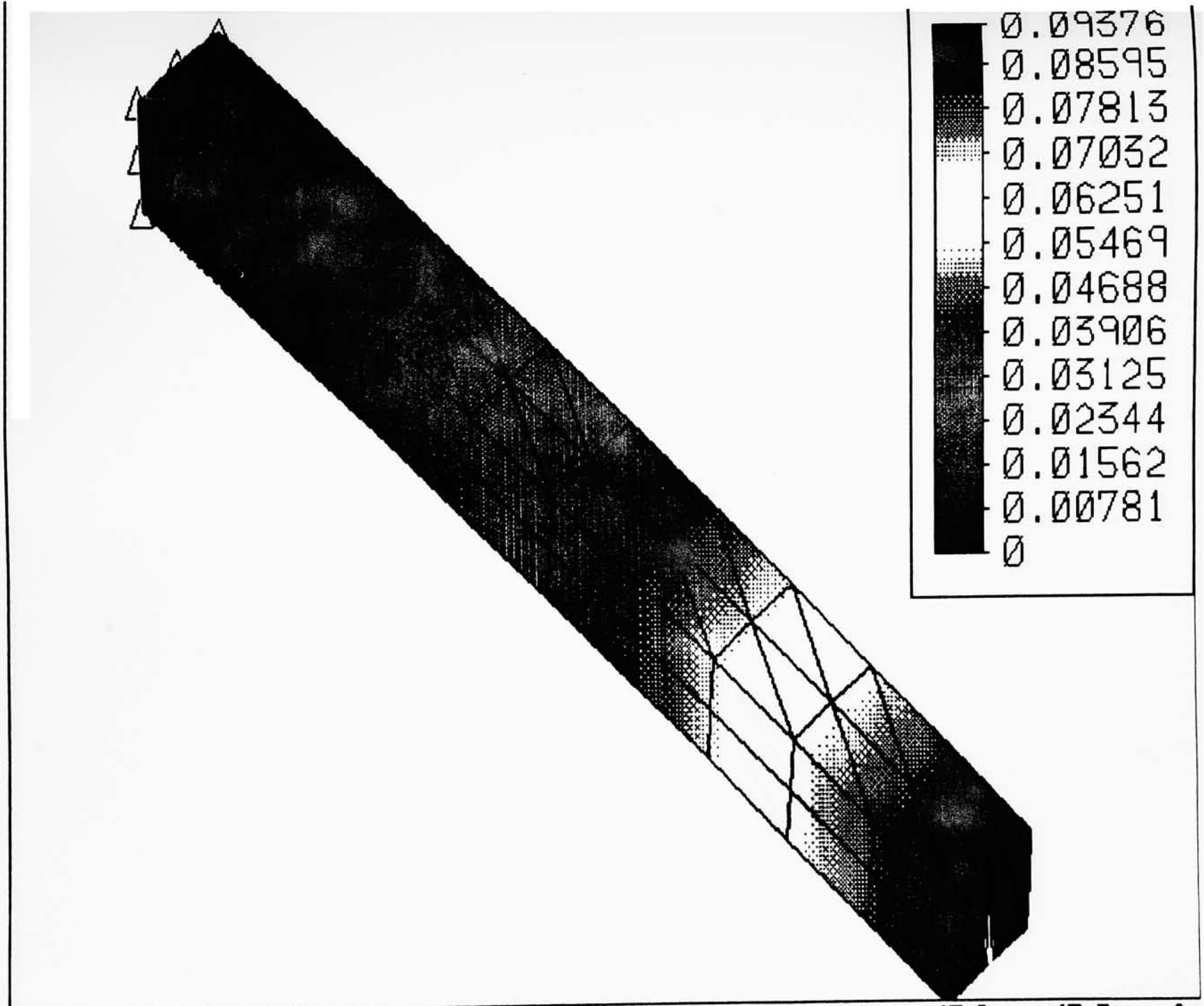

0.09376
0.08595
0.07813
0.07032
0.06251
0.05469
0.04688
0.03906
0.03125
0.02344
0.01562
0.00781
0
SVIEWH 3.14 File:beam1 92/03/02 15:13 LC 1/ 1 Vu= 7 Lo= 45 La= 45 R= 0

Figure 6.2

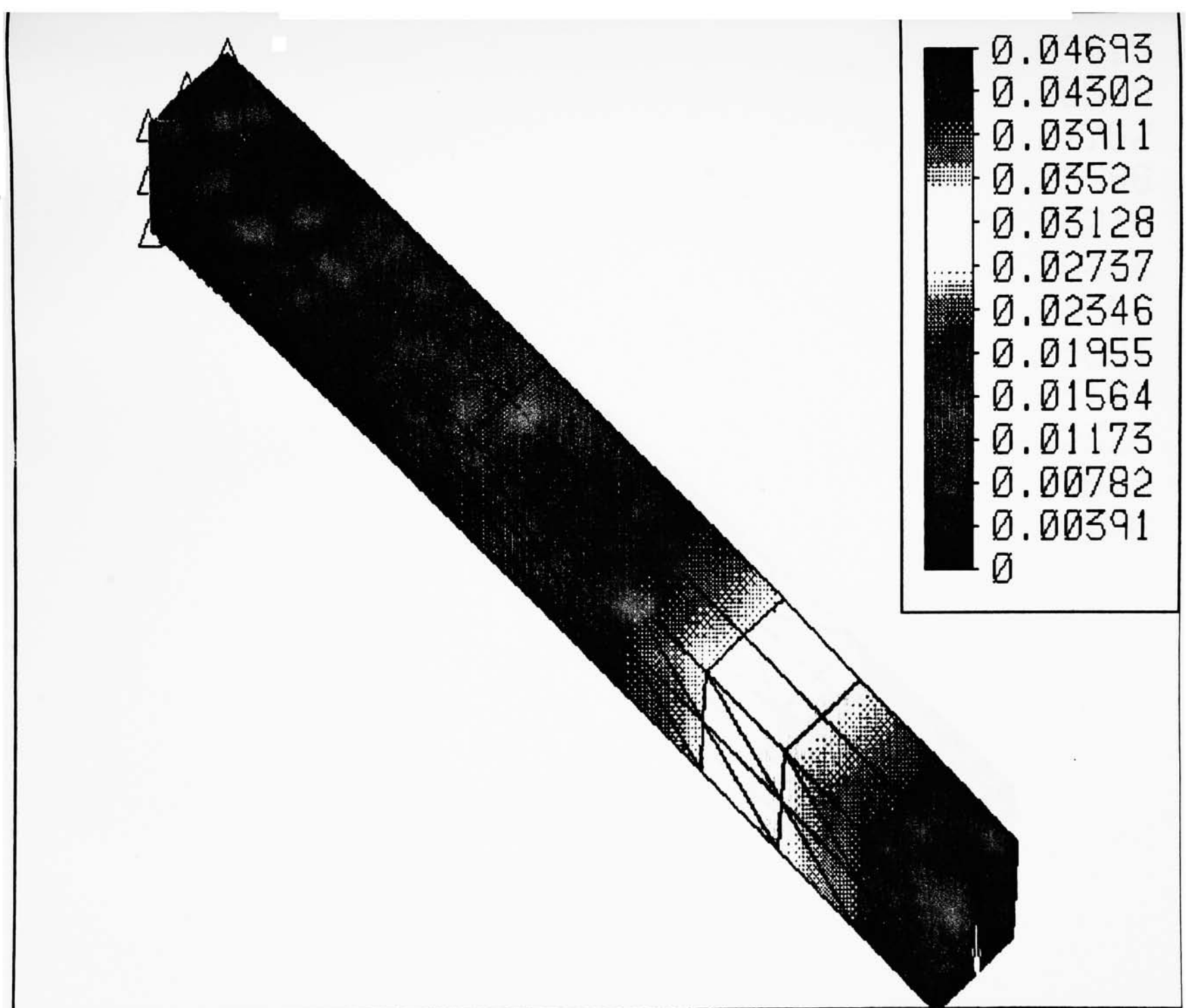
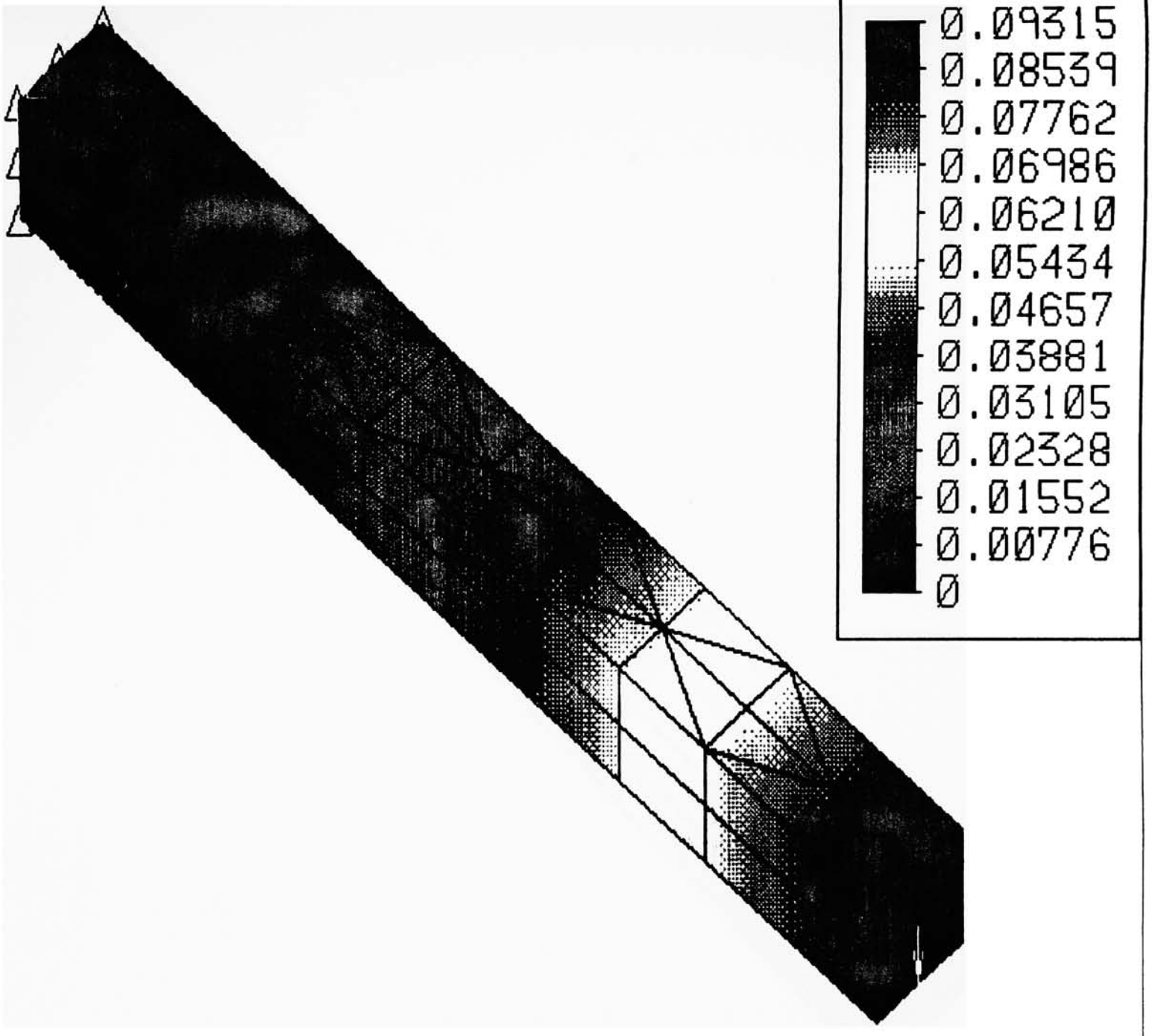

0.04693
0.04302
0.03911
0.0352
0.03128
0.02737
0.02346
0.01955
0.01564
0.01173
0.00782
0.00391
0
SVIEWH 3.14 File:tri2 92/03/02 10:44 LC 1/ 1 Vu= 7 Lo= 45 La= 45 R= 0

Figure 6.3



SVIEWH 3.14 File:beam2 92/03/02 15:03 LC 1/ 1 Vu= 7 Lo= 45 La= 45 R= 0

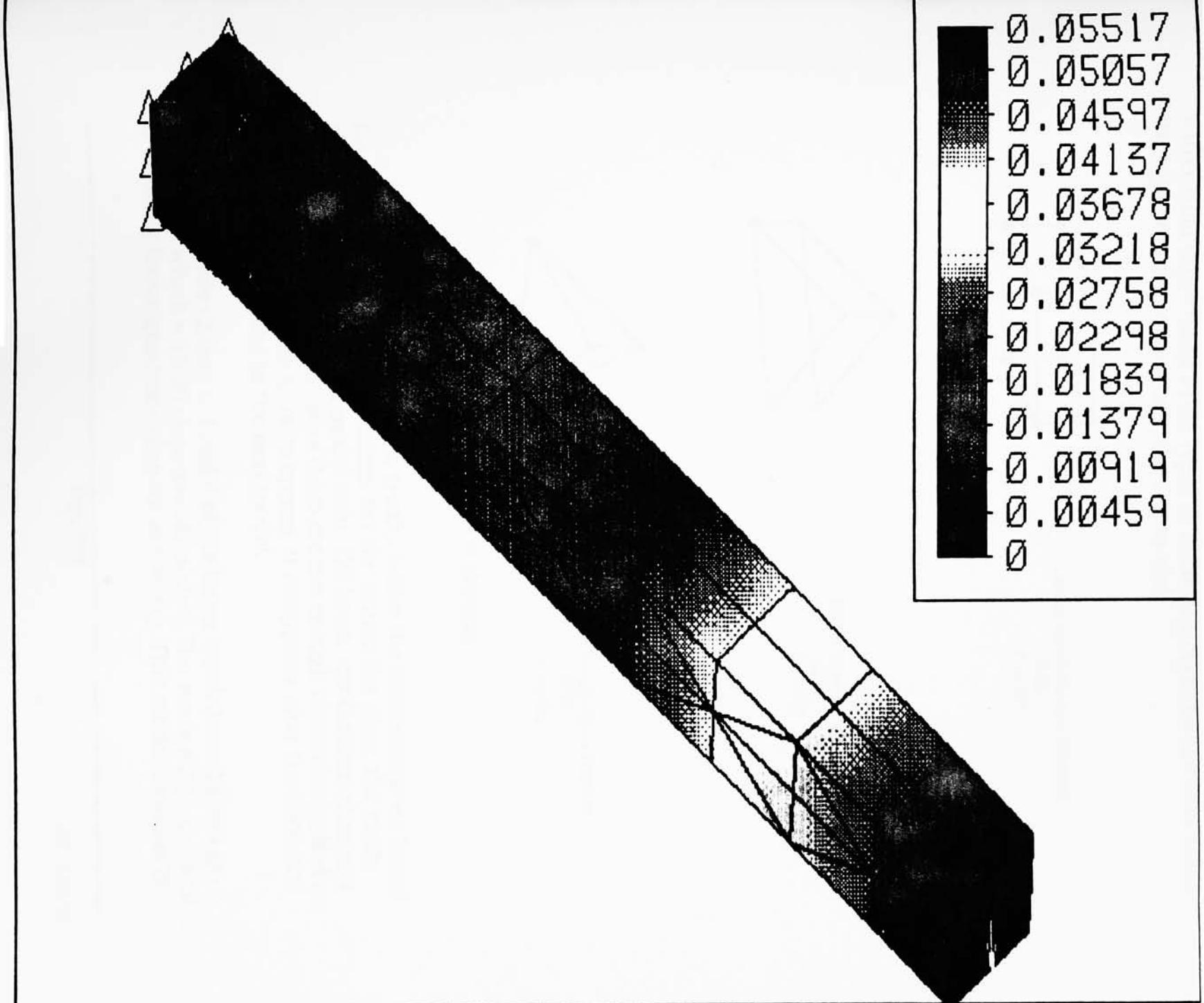
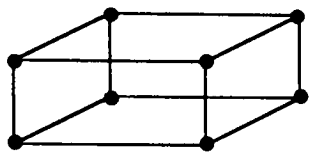
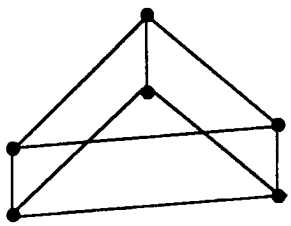

0.05517
0.05057
0.04597
0.04137
0.03678
0.03218
0.02758
0.02298
0.01839
0.01379
0.00919
0.00459
0
SVIEWH 3.14 File:tri1 92/03/02 13:26 LC 1/ 1 Vu= 7 Lo= 45 La= 45 R= 0

Figure 6.4

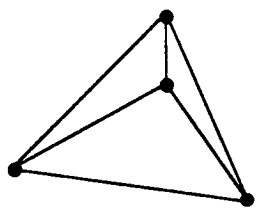
linear tetrahedron element has 4 nodes (see figure 6.6). The wedge element's shape are very much like the tetrahedron element, but it has 2 more nodes than tetrahedron elements. Here, we can conclude that this is the reason why FEA models created by tetrahedron elements are much stiffer than those using other types of elements, even though they utilize the same number of nodes in the FEA model.



Linear Quadrilateral Element
(LQ)
8 nodes



Linear Wedge Element
(LW)
6 nodes



Linear Tetrahedron Element
(LT)
4 Nodes

Figure 6.6 Type of elements

From modal analyses, the results follow the conclusions we found for the cantilever beam problems. For the square flat plate, the mode sharp 2, 3, 4 and 5 of FEA model with 176 linear tetrahedron elements (74 nodes) are wrong. Because there are not enough elements applied on the model causing the error to appear. It disappears after the elements (nodes) are increased in the next model.

The mode shapes 4, 5 and 6 of the linear quadrilateral elements FEA model which with 20 elements (42 nodes), The mode shape 4, 5 and 6 utilizing linear quadratic element are wrong. This model contains 20

elements (42 nodes). Same results using linear tetrahedron elements are also wrong. There models consisted of the mode shapes 2, 3, 4, 5 and 6 of model with 141 elements (59 nodes), mode shapes 4, 5 and 6 with 209 elements (86 nodes) and the mode shape 6 with 729 elements (241 nodes) In all cases, the results are improved after the elements (nodes) are increased. See table 6.1 - 6.12 and figure 6.7 - 6.18 for details.

Mode 1

of Elements vs. % of Error

# of Elements	QQ	QT	LQ	LT
20	8.81%		86.52%	
40	2.28%		21.66%	
80	1.30%			
100			5.99%	
140	-0.28%		3.11%	
141		13.25%		291.68%
209		6.89%		245.13%
300			0.37%	
600	-1.30%		-0.49%	
729		0.13%		128.49%
2300		-1.03%		81.81%
3055		-1.02%		
11021		-1.57%		
16088				22.16%

Table 6.1

of Nodes vs. % of Error

# of Nodes	QQ	QT	LQ	LT
42			86.52%	
82			21.66%	
202			5.99%	
282			3.11%	
602			0.37%	
1202			-0.49%	
143	8.81%			
283	2.28%			
563	1.30%			
983	-0.28%			
4203	-1.30%			
59				291.68%
86				245.13%
241				128.49%
784				81.81%
4133				22.16%
312		13.25%		
464		6.89%		
1394		0.13%		
4543		-1.03%		
6248		-1.02%		
18660		-1.57%		

Table 6.2

Mode 1 of Circular plate
-- # of Elements vs. % of Error --

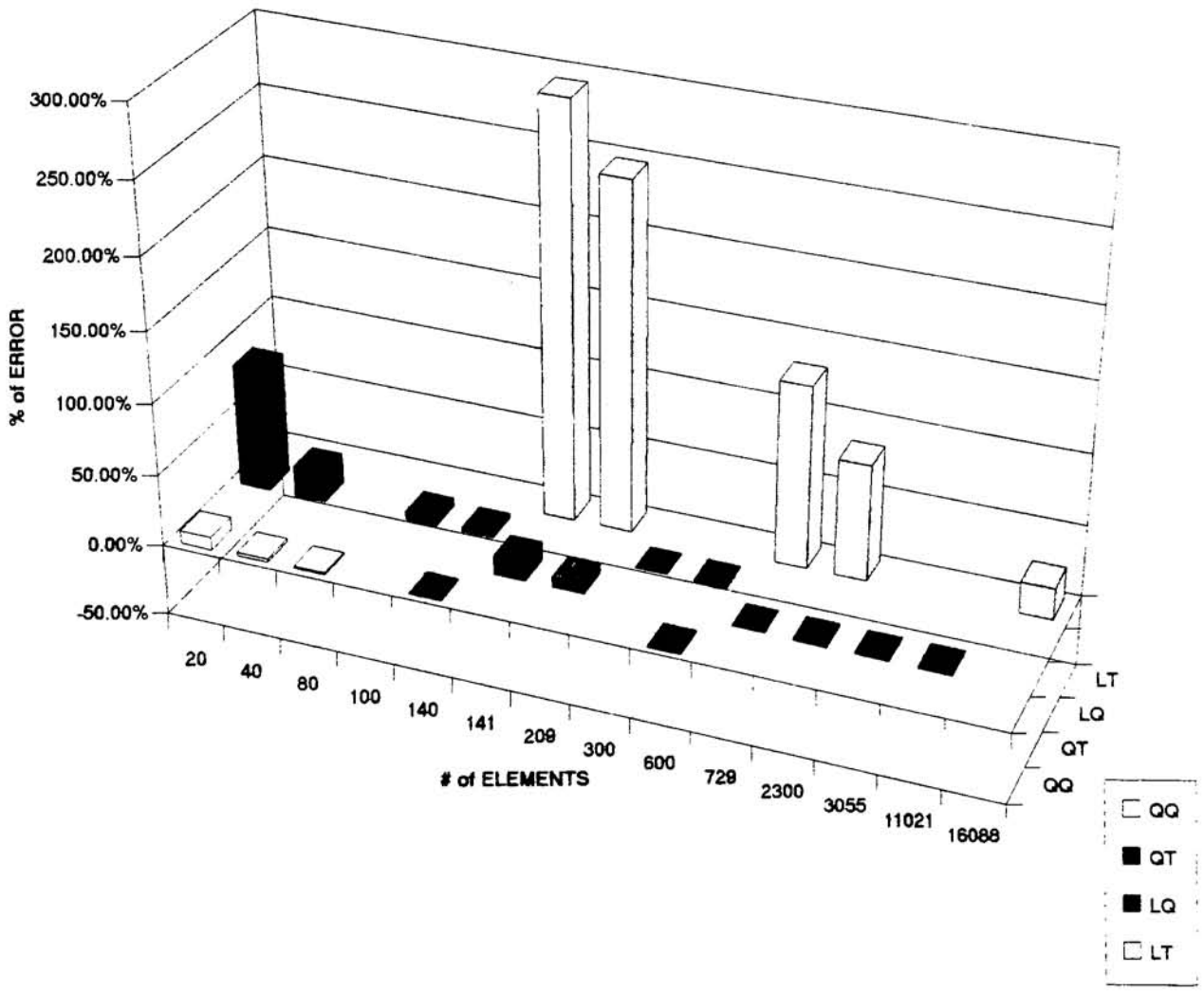


Figure 6.7

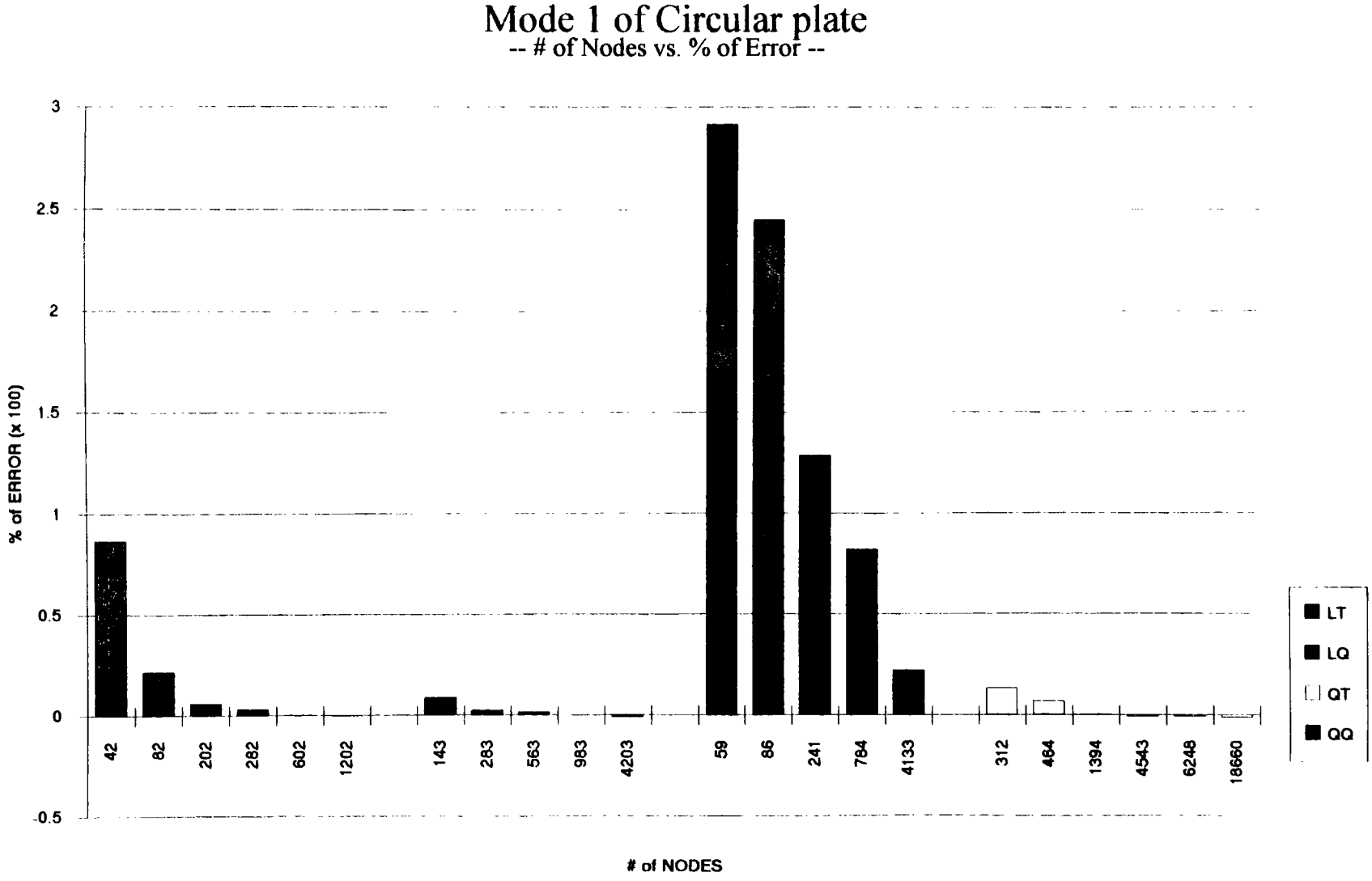

Mode 1 of Circular plate
-- # of Nodes vs. % of Error --
% of ERROR (x 100)
3
2.5
2
1.5
1
0.5
0
-0.5
42
82
202
282
602
1202
143
283
563
983
4203
59
86
241
784
4133
312
464
1394
4543
6248
18660
# of NODES
LT
LQ
QT
QQ

Figure 6.8

Mode 2

of Elements vs. % of Error

# of Elements	QQ	QT	LQ	LT
20	19.73%		207.65%	
40	1.47%		26.75%	
80	-0.32%			
100			6.52%	
140	-2.46%		2.42%	
141		23.35%		265.14%
209		10.54%		250.87%
300			-1.14%	
600	-3.53%		-2.40%	
729		-1.55%		101.43%
2300		-3.16%		70.44%
3055		-2.81%		
11021		-3.47%		
16088				18.82%

Table 6.3

of Nodes vs. % of Error

# of Nodes	QQ	QT	LQ	LT
42			207.65%	
82			26.75%	
202			6.52%	
282			2.42%	
602			-1.14%	
1202			-2.40%	
143	19.73%			
283	1.47%			
563	-0.32%			
983	-2.46%			
4203	-3.53%			
59				265.14%
86				250.87%
241				101.43%
784				70.44%
4133				18.82%
312		23.35%		
464		10.54%		
1394		-1.55%		
4543		-3.16%		
6248		-2.81%		
18660		-3.47%		

Table 6.4

Mode 2 of Circular plate

-- # of Elements vs. % of Error --

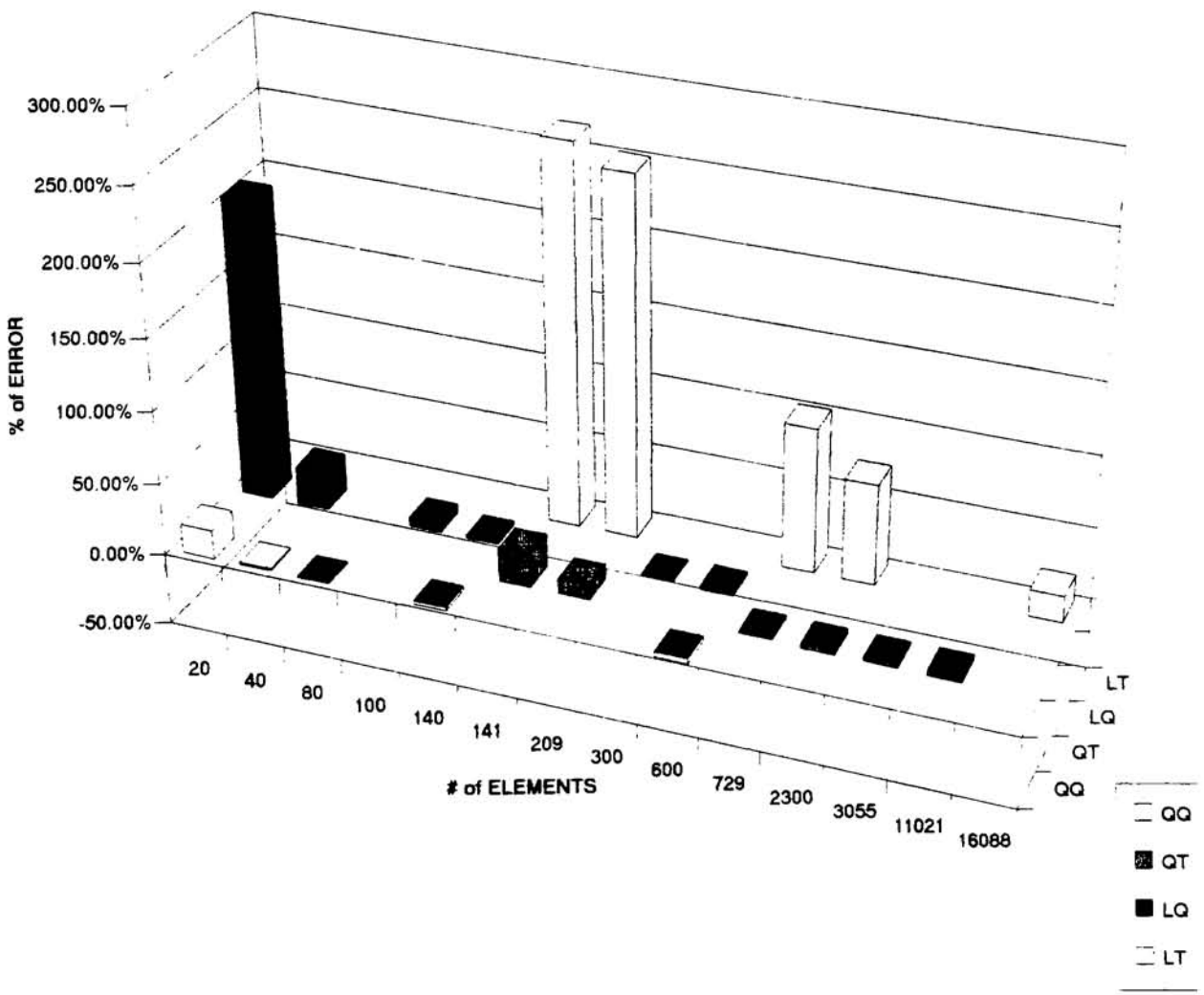


Figure 6.9

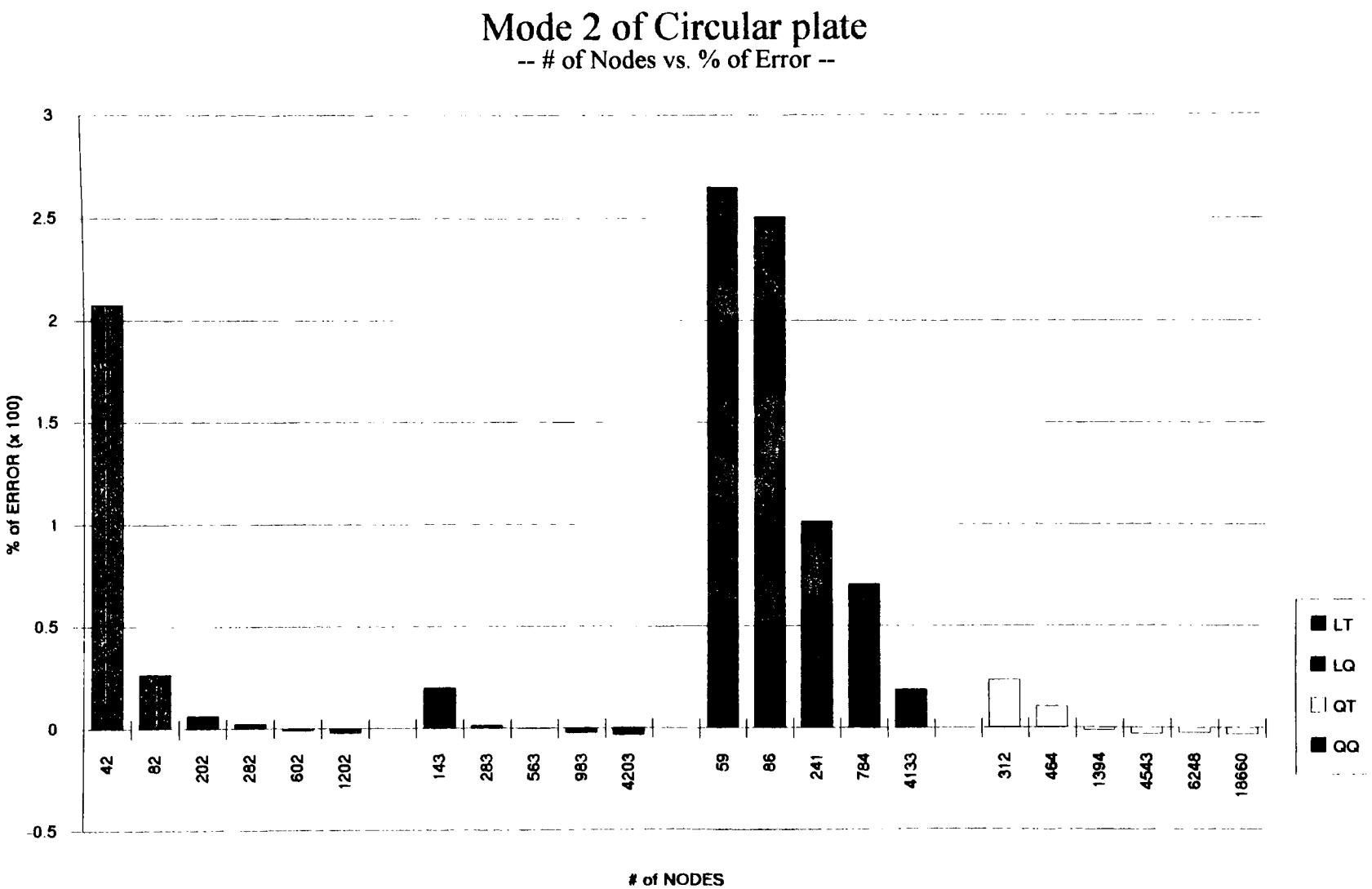

Mode 2 of Circular plate
-- # of Nodes vs. % of Error --
% of ERROR (x 100)
3
2.5
2
1.5
1
0.5
0
-0.5
42
82
202
282
602
1202
143
283
563
983
4203
59
86
241
784
4133
312
464
1394
4543
6248
18660
# of NODES
LT
LQ
QT
QQ

Figure 6.10

Mode 3

of Elements vs. % of Error

# of Elements	QQ	QT	LQ	LT
20	19.90%		220.86%	
40	1.53%		30.67%	
80	-0.29%			
100			6.68%	
140	-2.42%		2.57%	
141		26.76%		272.65%
209		12.13%		267.34%
300			-1.01%	
600	-3.53%		-2.39%	
729		-0.59%		125.03%
2300		-2.86%		76.21%
3055		-2.34%		
11021		-3.25%		
16088				20.15%

Table 6.5

of Nodes vs. % of Error

# of Nodes	QQ	QT	LQ	LT
42			220.86%	
82			30.67%	
202			6.68%	
282			2.57%	
602			-1.01%	
1202			-2.39%	
143	19.90%			
283	1.53%			
563	-0.29%			
983	-2.42%			
4203	-3.53%			
59				272.65%
86				267.34%
241				125.03%
784				76.21%
4133				20.15%
312		26.76%		
464		12.13%		
1394		-0.59%		
4543		-2.86%		
6248		-2.34%		
18660		-3.25%		

Table 6.6

Mode 3 of Circular plate

-- # of Elements vs. % of Error --

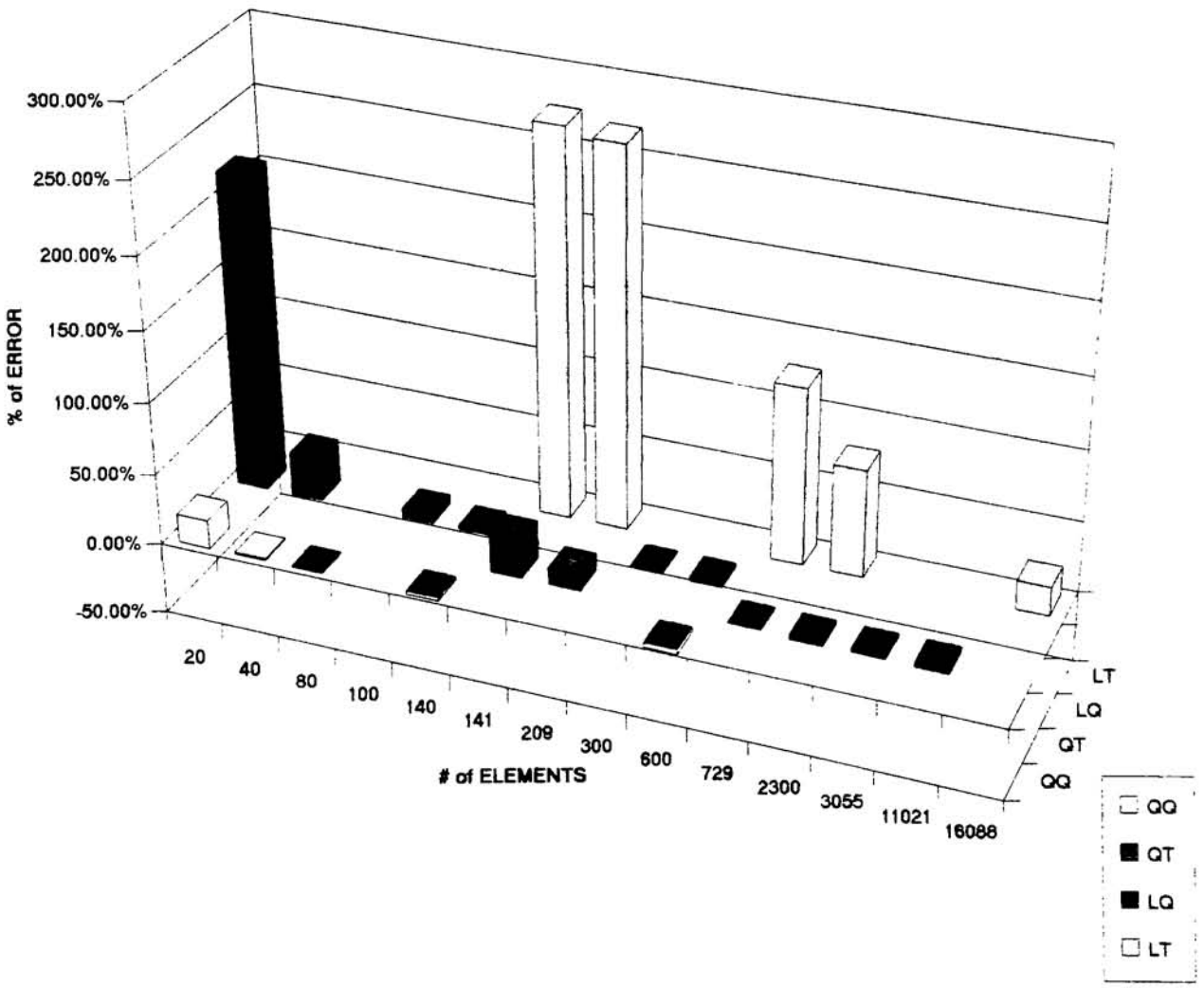


Figure 6.11

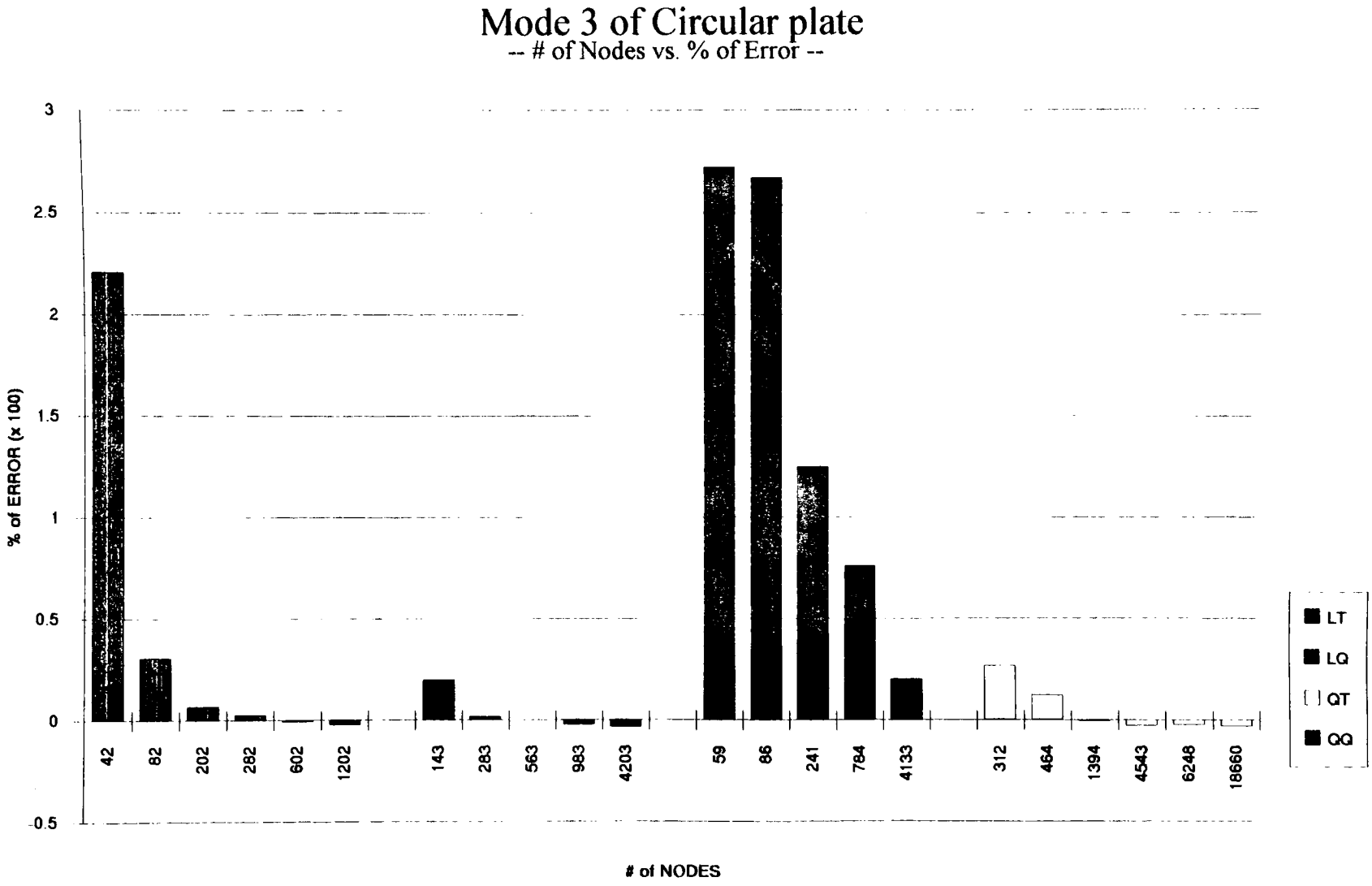

Mode 3 of Circular plate
-- # of Nodes vs. % of Error --
% of ERROR (x 100)
3
2.5
2
1.5
1
0.5
0
-0.5
42
82
202
282
602
1202
143
283
563
983
4203
59
86
241
784
4133
312
464
1394
4543
6248
18660
# of NODES
LT
LQ
QT
QQ

Figure 6.12

Mode 4

of Elements vs. % of Error

# of Elements	QQ	QT	LQ	LT
20	29.04%		124.62%	
40	1.93%		49.97%	
80	-1.67%			
100			10.77%	
141		37.45%		160.72%
209		14.87%		127.35%
300			-1.85%	
600	-5.18%		-3.31%	
729		-2.18%		102.84%
2300		-4.49%		64.14%
3055		-3.40%		
11021		-4.14%		
16088				17.25%

Table 6.7

of Nodes vs. % of Error

# of Nodes	QQ	QT	LQ	LT
42			124.62%	
82			49.97%	
202			10.77%	
282			3.85%	
602			-1.85%	
1202			-3.31%	
143	29.04%			
563	-1.67%			
983	-4.18%			
4203	-5.18%			
59				160.72%
86				127.35%
241				102.84%
784				64.14%
4133				17.25%
312		37.45%		
464		14.87%		
1394		-2.18%		
4543		-4.49%		
6248		-3.40%		
18660		-4.14%		

Table 6.8

Mode 4 of Circular plate

-- # of Elements vs. % of Error --

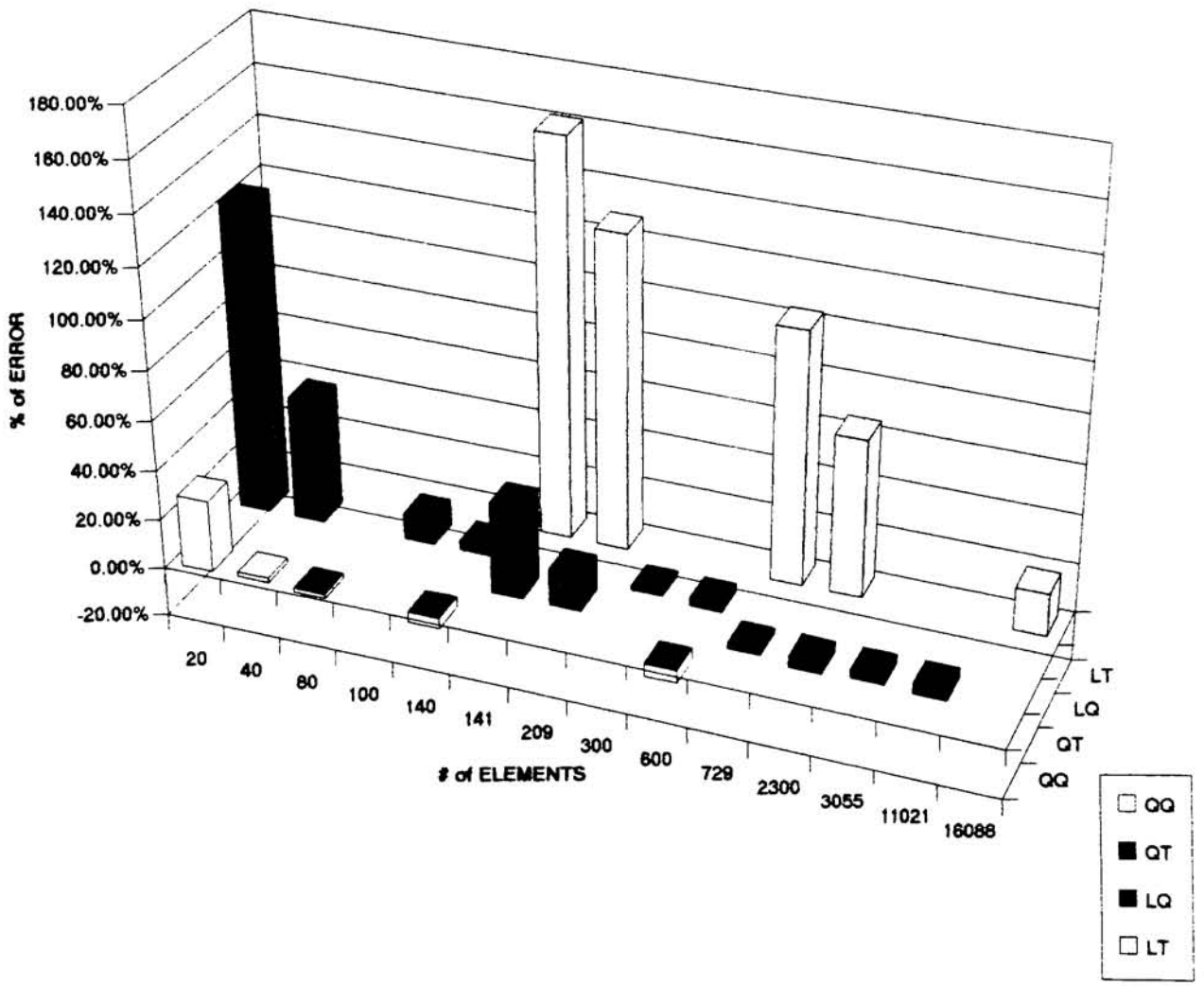


Figure 6.13

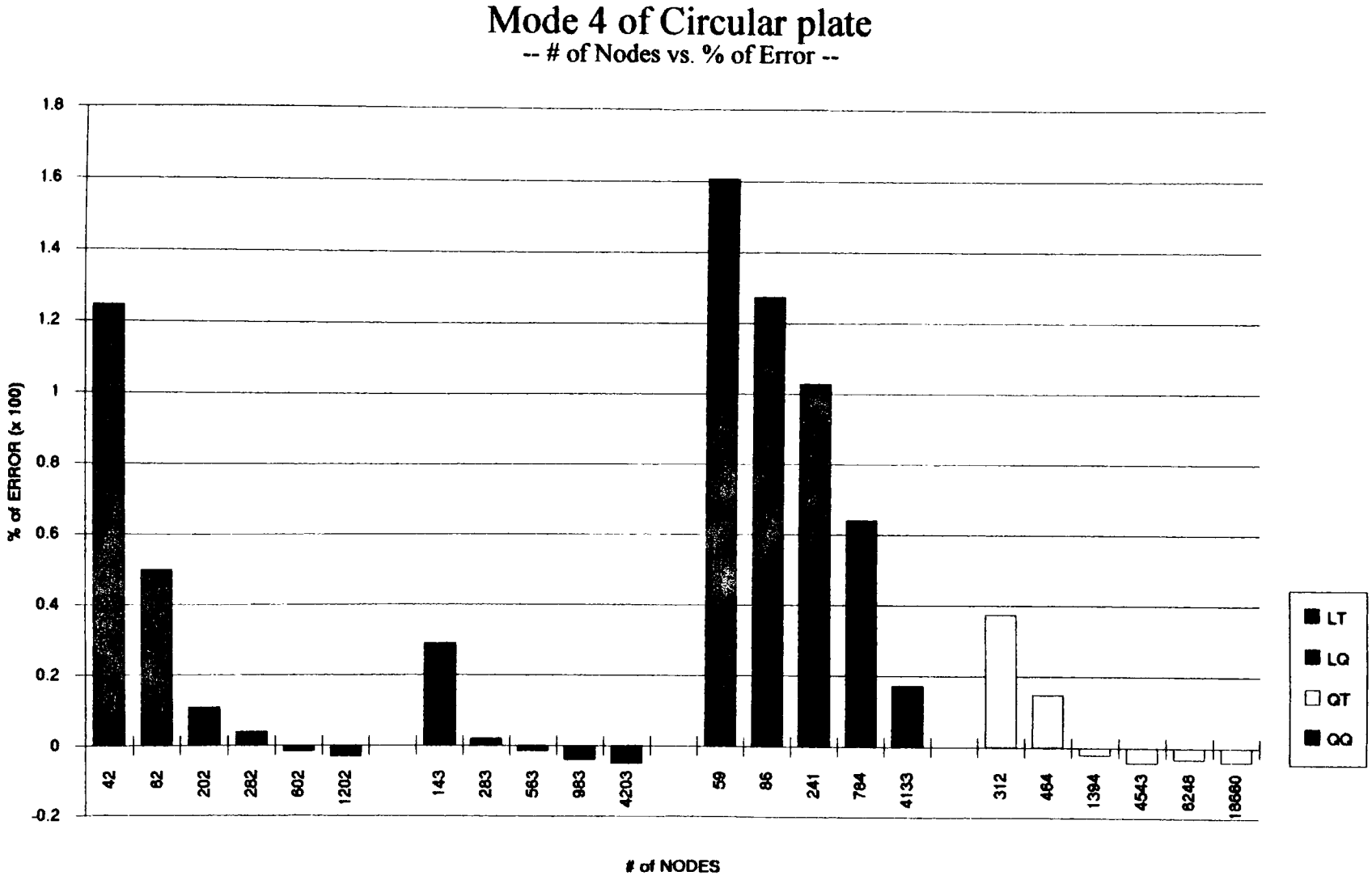

Mode 4 of Circular plate
-- # of Nodes vs. % of Error --
% of ERROR (x 100)
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
42
82
202
282
602
1202
143
283
563
983
4203
59
86
241
784
4133
312
464
1394
4543
8248
18660
# of NODES
LT
LQ
QT
QQ

Figure 6.14

Mode 5

of Elements vs. % of Error

# of Elements	QQ	QT	LQ	LT
20	30.59%		132.95%	
40	1.96%		55.21%	
80	-1.63%			
100			10.97%	
140	-4.12%		4.23%	
141		40.92%		195.35%
209		17.20%		132.58%
300			-1.72%	
600	-5.16%		-3.28%	
729		-1.75%		112.79%
2300		-4.21%		66.49%
3055		-2.84%		
11021		-4.01%		
16088				17.65%

Table 6.9

of Nodes vs. % of Error

# of Nodes	QQ	QT	LQ	LT
42			132.95%	
82			55.21%	
202			10.97%	
282			4.23%	
602			-1.72%	
1202			-3.28%	
143	30.59%			
283	1.96%			
563	-1.63%			
983	-4.12%			
4203	-5.16%			
59				195.35%
86				132.58%
241				112.79%
784				66.49%
4133				17.65%
312		40.92%		
464		17.20%		
1394		-1.75%		
4543		-4.21%		
6248		-2.84%		
18660		-4.01%		

Table 6.10

Mode 5 of Circular plate

-- # of Elements vs. % of Error --

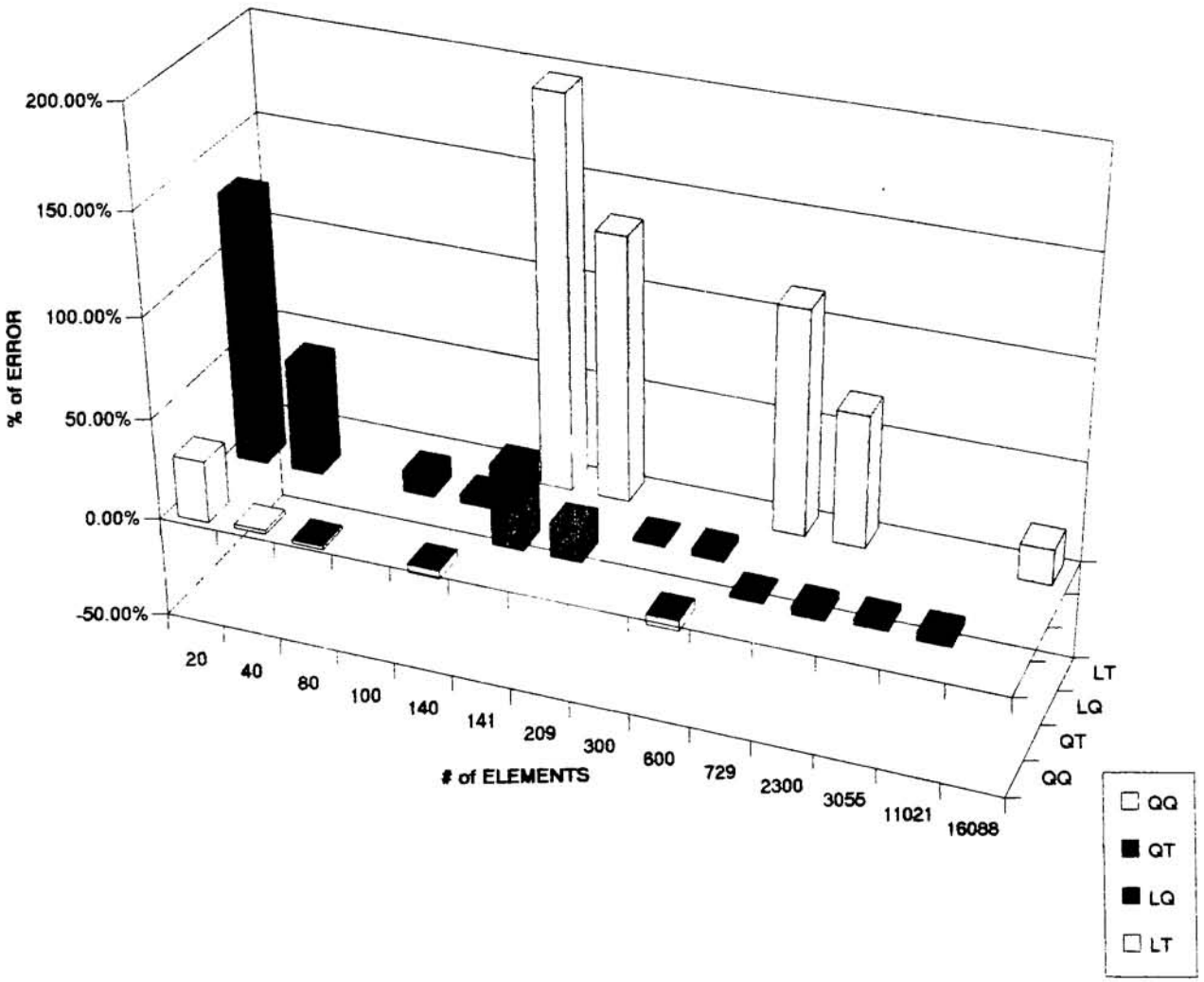


Figure 6.15

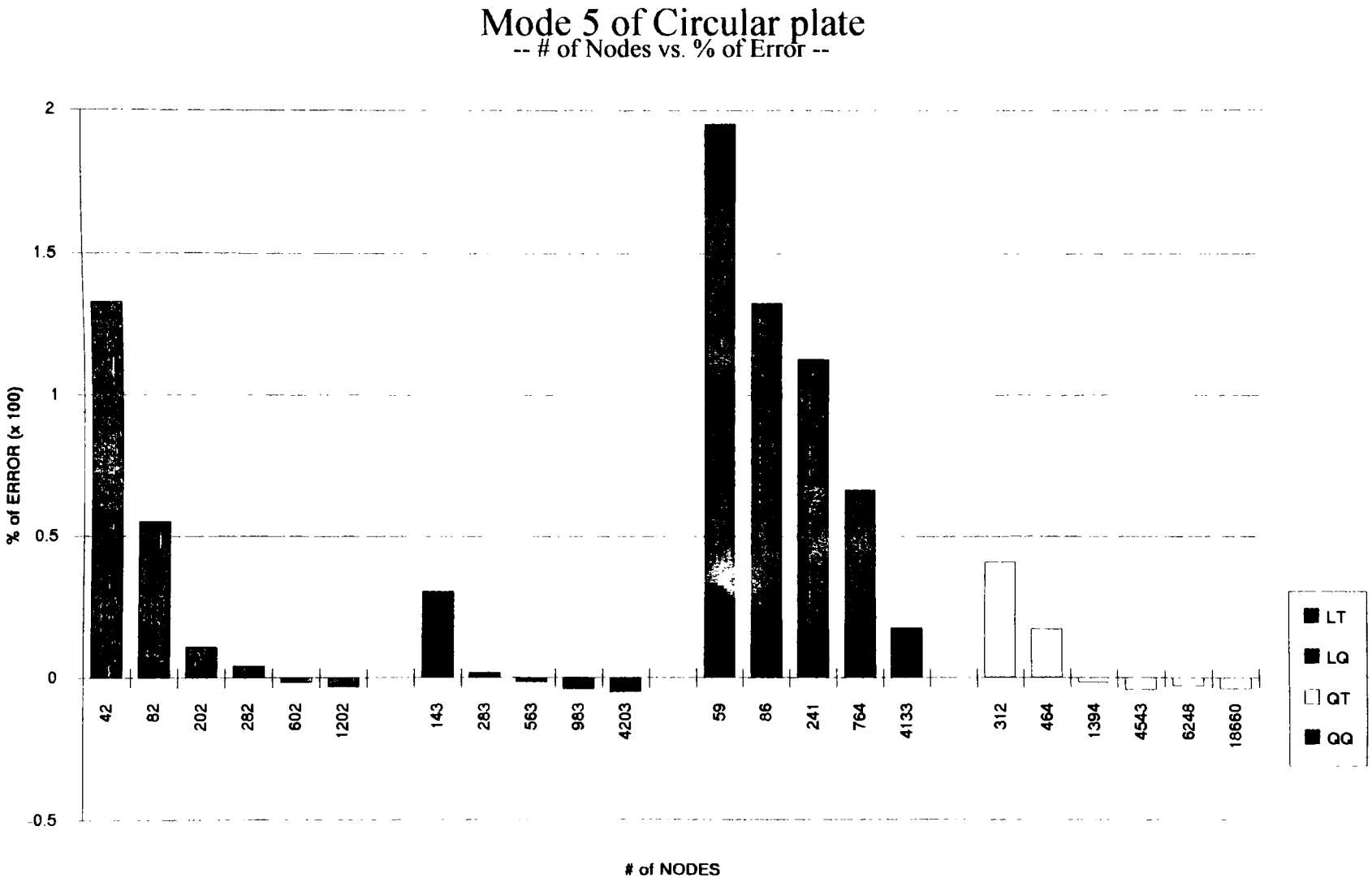

Mode 5 of Circular plate
-- # of Nodes vs. % of Error --
% of ERROR (x 100)
2
1.5
1
0.5
0
-0.5
42
82
202
282
602
1202
143
283
563
983
4203
59
86
241
764
4133
312
464
1394
4543
6248
18660
LT
LQ
QT
QQ
# of NODES

Figure 6.16

Mode 6

of Elements vs. % of Error

# of Elements	QQ	QT	LQ	LT
20	37.14%		129.76%	
40	1.09%		49.63%	
80	-0.61%			
100			15.64%	
140	-4.11%		5.89%	
141		43.62%		169.75%
209		17.53%		150.51%
300			-0.24%	
600	-5.41%		-3.92%	
729		-1.92%		89.81%
2300		-4.47%		65.04%
3055		-2.81%		
11021		-3.95%		
16088				16.55%

Table 6.11

of Nodes vs. % of Error

# of Nodes	QQ	QT	LQ	LT
42			129.76%	
82			49.63%	
202			15.64%	
282			5.89%	
602			-0.24%	
1202			-3.92%	
143	37.14%			
283	1.09%			
563	-0.61%			
983	-4.11%			
4203	-5.41%			
59				169.75%
86				150.51%
241				89.81%
784				65.04%
4133				16.55%
312		43.62%		
464		17.53%		
1394		-1.92%		
4543		-4.47%		
6248		-2.81%		
18660		-3.95%		

Table 6.12

Mode 6 of Circular plate

-- # of Elements vs. % of Error --

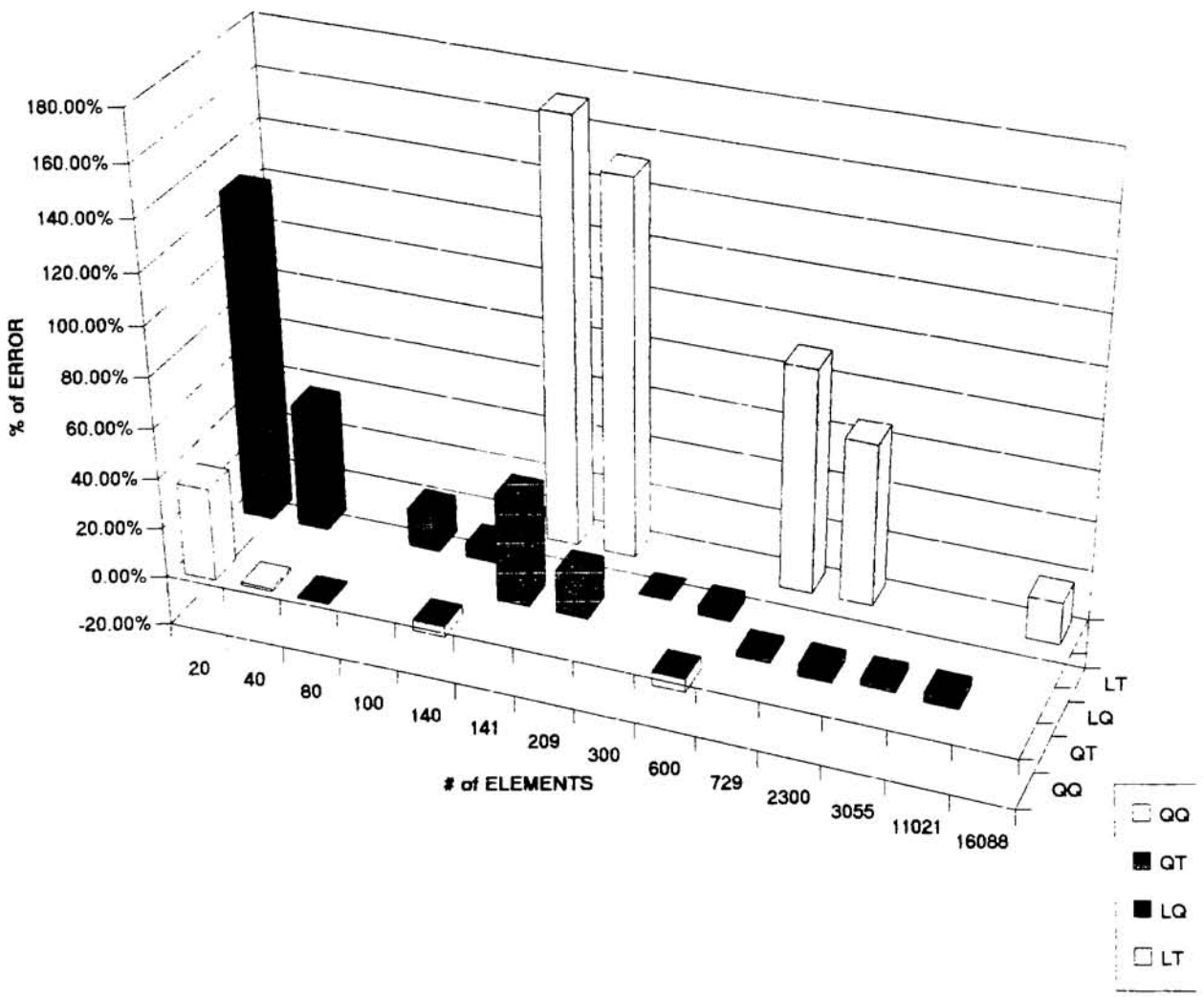


Figure 6.17

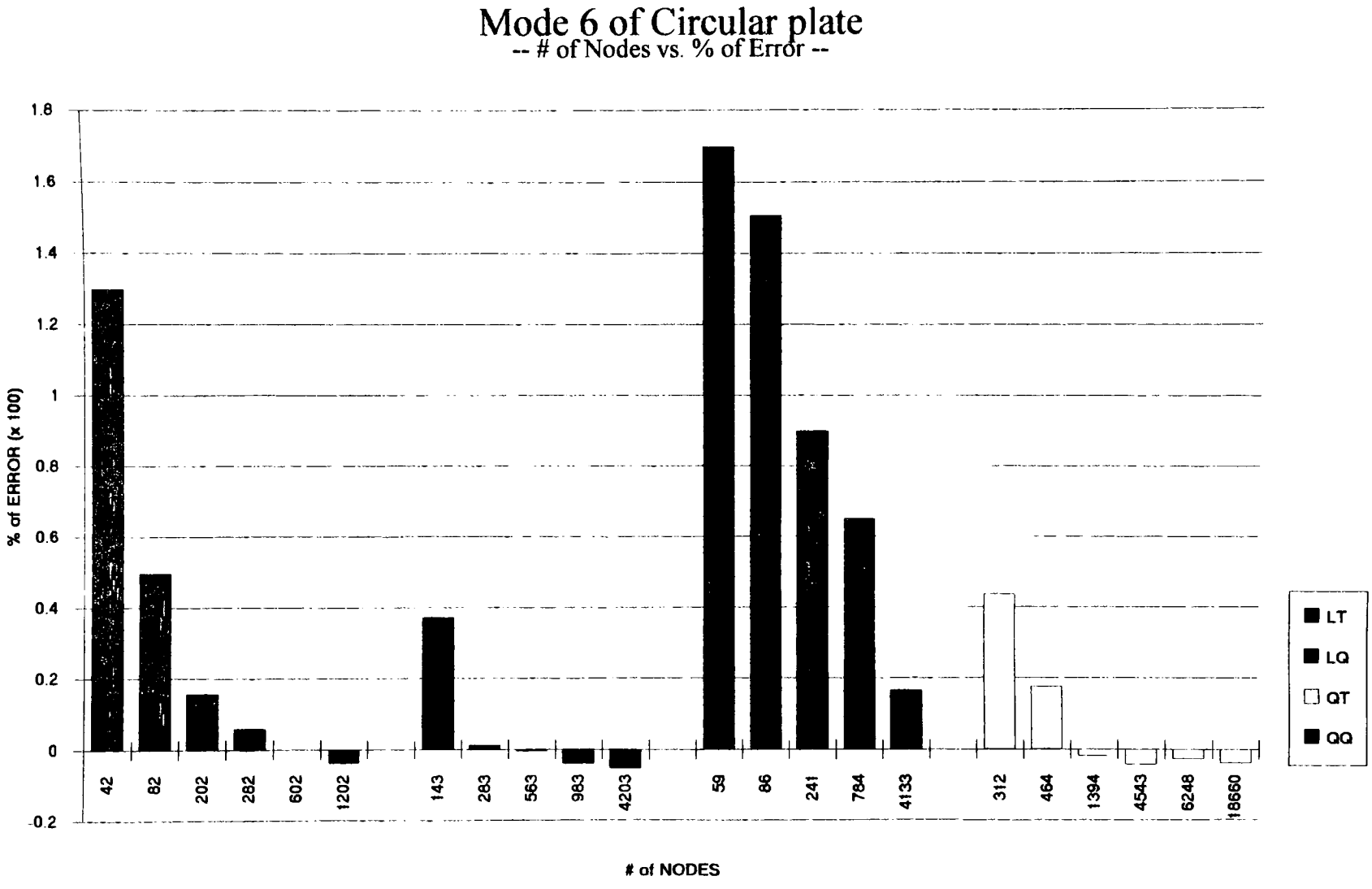

Mode 6 of Circular plate
-- # of Nodes vs. % of Error --
% of ERROR (x 100)
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
42
82
202
282
602
1202
143
283
563
983
4203
59
86
241
784
4133
312
464
1394
4543
6248
18660
# of NODES
LT
LQ
QT
QQ

Figure 618

From the square flat plate modal analysis, the FEA result for mode 5, with all types of elements, are way off from the analytical solution (LQ with 160.87% , 32.87%, 21.80% and 18.53%, QQ with 40.82%, 17.85% and 15.93%, LT with 271.46% , 133.79%, 78.50% and 40.60%, QT with 68.63%, 18.94% and 17.59%). The mode 5's FEA results appear to be at around 6200 Hz and the analytical solution is 5264.6 Hz. This situation did not appear with the circular flat plate. ---- So chances are, either the solution is wrong or the software has a bug in it -----

From the FEA result of the modal analysis, it appears that the FEA solutions are below the analytical solutions (negative percentage of error). The frequency equation for a square flat plate is shown below:

$$f_n = \frac{k_n}{2\pi}\sqrt{\frac{Dg}{wa^4}} \quad \text{.....Eq. 3.5}$$

where

k1 = 36 fundamental
k2 = 73.4 one nodal diameter
k3 = 108.3 two nodal diameter
$D = Et^3/12(1-\nu^2)$
g = gravity
w = wt./area = density x *thickness*
a = length of edge

The frequency equation for a circular flat plate is shown below:

$$f_n = \frac{k_n}{2\pi}\sqrt{\frac{Dg}{wr^4}} \quad \text{.....Eq. 3.13}$$

where

k1 = 10.2 fundamental
k2 = 21.3 one nodal diameter
k3 = 34.9 two nodal diameter
k4 = 39.8 one nodal circle
$D = Et^3/12(1-\nu^2)$
g = gravity
w = wt./area = density x *thickness*
r = radius

The solution for each mode depends on the value of the constant k_n. These constants are given in a handbook, based on the zeroes of Bessel functions. For example, k_1 for a circular plate is 10.2. The estimate for f1 is shown below:

$$f_1 = \frac{10.2}{2\pi}(1221.736237) = 1983.34 \text{ Hz(cycle/sec)} \ldots\ldots\ldots\ldots \text{ Mode 1}$$

If k_1 has a round-off error, let us assume that $k_1 = 10.19$, then the result shows

$$f_1 = 1981.40 \text{ Hz.}$$

It has -0.098% error compared with the solution using the value 10.2. If $k_1 = 10.15$, then the result leads to

$$f_1 = 1973.62 \text{ Hz.}$$

It has -0.5% error compared with the solution using the constant 10.2. These negative percentage of errors appear because of the round-off error. If we have round-off error in the material density, constant of gravity and constant of $k_{n,}$ then this error margin will increase. As long as the solutions are converging, then the difference between the FEA results and analytical solutions can be assumed to be caused by round-off error.

Chapter 7. Summary

This investigation has reviewed various automatic finite element mesh generation techniques along with uses of geometric modelers. The developments in automatic mesh generation reduce the involvement of the user in the tedious mesh generation process. The future efforts are expected to further reduce the time spent in the modeling process, in obtaining better finite element models and modifying the element densities based on results obtained from an initial run.

This paper has examined the quality of tetrahedron elements in various classes of problems. How does the automatic mesh generator and tetrahedron element perform with real problems? The figure 7.1 - 7.8 are shown a housing part and two cases of finite elements analyses which were done by using linear and quadratic tetrahedron elements.

For this part, the two FEA models are almost 100% identical to the actual parts and are almost impossible to create by mapped meshing due to the details required. Each model took less than 4 hours to mesh (due to individual experiences). The linear model took 43.3 minutes to solve. The quadratic model took 5 hours and 33.6 minutes to solve. The linear element model has 9181 LT elements and 2052 nodes. The quadratic element model has 9109 QT elements and 14270 nodes. From these results, the model created by linear elements only has 50% - 40% of the values of stress obtained by using quadratic elements.

The stress distributions (contours) of these two models are almost identical to each other (except the stress numbers) and match the stress coating results. The stresses shown on the quadratic element model are less than 10 % in error compared with actual test results. This suggests that the quadratic tetrahedron element yields a very high quality FEA result. For this portion of the analysis, from creating the model to getting the FEA results, it only takes less than two days to complete. For the entire project, it took two weeks to complete (including four different design parts and twelve FEA models). I have a similar design done by mapped meshing which took a full two weeks to complete one FEA model. It would take the same amount of time to complete for each redesign.

As all the test results show, the linear tetrahedron element is way off the actual solutions. But the stress distributions (contour) of linear tetrahedron element models are very close to other models which were done using other types of elements. The computer time is less than one

fifth of that need for quadratic elements. What if, we use the linear tetrahedron elements for a first try out of the design model. Use the automatic mesh generator to create a FEA model to find out the areas of stress concentration. Thus, we can understand the first design and find out the potential problems which may appear. Then, use this information to redesign and create a FEA model using quadratic elements. At this time, we should have a precise solution from FEA and within a shorter period of time.

This paper has reviewed and examined the various types of elements. I found:

- The tetrahedron elements are stiffer than other types of elements.
- The FEA results done by linear tetrahedron elements are unacceptable, but the stress distributions (contours) are same as these results down by other type of elements.
- The mesh generated by automatic mesh generator is much closer to the actual model compared with those obtained by mapped meshing.
- The finite elements analysis results are dependent on the number of nodes, not the number of elements.
- The time required to mesh a model using the automatic mesh generator is less than 25% of the time required when using mapped meshing.

The other important aspect is that of adaptive mesh generation. Currently, the user has to specify the density of elements at various locations in the model. More elements are placed in the higher stress gradients. If the program does it automatically, the user's interaction in this regard can be eliminated. The program can start with a uniform mesh. This will not give very good results, but will give some indication of regions with high stress gradients. These can be used to decide on the element densities of the next mesh. After a few attempts, it should be possible to get the desired mesh density with minimal effort of the user.

Currently, solid modelers are widely used in industry. A solid model can show the design before making the prototype model and FEA can take the solid model to analyze (only with tetrahedron elements) the design. Once the design is completed, we can use the solid model to perform stereo lithography which can make a plastic prototype over-

night or as input to the CNC machine for machining. All the procedures can work continuously without any paperwork.

References

1. A A G Riquicha and H B Voelcker, Solid modeling: A historical summary and contemporary assessment, IEEE Computer Graphic Appl, 1982.
2. A A G Riquicha and H B Voelcker, Solid modeling: Current status and research direction, IEEE Computer Graphic Appl, Oct. 1983.
3. M S Shephard, Approaches to automatic generation and control of finite element meshes, Applied Mechanics Reviews Vol. 41, No. 4, April 1988.
4. W A Cook and W R Oakes, Mapping methods for generating 3-D meshes, Computer in Mechanical Eng, Aug. 1982.
5. Zienkiewicz and Phillips, An automatic mesh generation scheme for plane and curved surfaces by isoparametric coordinates, Int J numer Methods Eng., Vol. 3, 1971.
6. C O Frderick, Y C Wong, and F W Edge, 2-D automatic mesh generation for structral analysis, Int J Numer Methods Eng. 2, 1970.
7. J C Cavendish, D A field, and W H Frey, An approach to automatic 3-D mesh generation, Int J Numer Methods Eng., Vol. 3, 1971.
8. D F Watson, Computing the n-dimensional Delaunay tessellation with applications to Voronoi polytopes, Computer J 24(2), 1981.
9. Nguyen and Van-Phai, Automatic mesh generation with tetrahedron elements, Int J Numer Methods Eng. 18, 1982.
10. W J Schroeder and M S Shephard, A combined octree/Delaunay method for fully automatic 3-D mesh generation, Center for Interactive Computer Graphics, RPI, Troy NY, 1987.
11. W J Schroeder and M S Shaephard, An O(n) algorithm to automatically generate geometric triangulations satisfying the Delaunay circumsphere criteria, Eng. with Computers 5, 1989.
12. W J Schroeder and M S Shaephard, Gometry-based fully automatic mesh generation and the Delauny Triangulation, Int J Numer Methods Eng. 26, 1988.
13. E A Sadek, A scheme for the automatic generation of triangular finite elements, Int J Numer Methods Eng. 15, 1980.
14. M S Shephard, K R Grice, C E Soechtig, and C M Graichen, Automatic, topologically correct, three-dimensional mesh generation by the finite octree technique, Center for interactive Computer Graphics, RPI, Troy NY, 1987.
15. M S Shephard, K R Grice, and M K georges, Some recent advances in automatic mesh generation, Modern methods for automating finite element mesh generation, K Baldwin, Ed, ASCE, NY, 1986.
16. B Wordenweber, Finite element mesh generation, Computer Aided Design 16, 1984.

17. A J C Schoofs, L H Th M Van Beukering, and M L C Sluiter, A general pupose two-dimensional mesh generator, Computers in Engineering, Vol. 3, ASME, NY, 1982.
18. A Jain, Generating finite element meshes on geometry from a solid modeler, Finite element methods, modeling and new applications, Vol. 1, ASME 1985.
19. M L C Sluiter and D L Hansen, A general pupose two- and three-dimemsional mesh generator, Computers in Engineering, Vol. 3, ASME 1985.
20. P L Baehmann, S L Wittchen, M S shephard, K R Grice, and M A Yerry, Robust geometrically-based automatic two-dimensional mesh generation, Int J Numer Methods Eng. 24, 1987.
21. M A YERRY and M S Shephard, Finite element mesh generation based on a modified-quadtree approach, IEEE Computer Graphics Appl. 3(1), 1983.
22. M A YERRY and M S Shephard, A modified quadtree approach to finite element mesh generation, IEEE Computer Graphics Appl, 1983.
23. M A YERRY and M S Shephard, Automatic three-dimensional mesh generation by the modified-octree technique, Int J Numer Methods Eng. 20 1984.
24. M A YERRY and M S Shephard, Finite element modeling within an integrated geometric modeling environment: part I -- Mesh generation, Eng. with Computers 1, 1985.