2011-06-01

Hexahedral Mesh Refinement Using an Error Sizing Function

Gaurab Paudel
Brigham Young University - Provo

Hexahedral Mesh Refinement Using

an Error Sizing Function

Gaurab Paudel

A thesis submitted to the faculty of
Brigham Young University
In partial fulfillment of the requirements for the degree of

Master of Science

Steven E. Benzley, Chair
Paul W. Richards
Steven J. Owen

Department of Civil and Environmental Engineering

Brigham Young University

June 2011

ABSTRACT

Hexahedral Mesh Refinement Using
an Error Sizing Function

Gaurab Paudel
Department of Civil and Environmental Engineering
Master of Science

The ability to effectively adapt a mesh is a very important feature of high fidelity finite element modeling. In a finite element analysis, a relatively high node density is desired in areas of the model where there are high error estimates from an initial analysis. Providing a higher node density in such areas improves the accuracy of the model and reduces the computational time compared to having a high node density over the entire model. Node densities can be determined for any model using the sizing functions based on the geometry of the model or the error estimates from the finite element analysis. Robust methods for mesh adaptation using sizing functions are available for refining triangular, tetrahedral, and quadrilateral elements. However, little work has been published for adaptively refining all hexahedral meshes using sizing functions. This thesis describes a new approach to drive hexahedral refinement based upon an error sizing function and a mechanism to compare the sizes of the node after refinement.

Keywords: hexahedral, meshing, adaptation, refinement, sizing function, error estimates

ACKNOWLEDGMENTS

I would like to thank my graduate advisor, Dr. Steven E. Benzley for his support in this research. I would also like to thank Sandia National Laboratories for providing funding as well as the software necessary to develop this refinement technique and an internship opportunity during the spring/summer term. Specifically, I would like to thank Steve J. Owen for his constant willingness to provide assistance. I would also like to thank Mark Dewey at ETI who helped me with debugging the codes and also like to thank Tim Miller who helped throughout this research. I would also like to thank the Department of Civil and Environmental Engineering for providing the resources and financial support during my undergraduate education. Lastly, I would like to thank my family for their constant support and encouragement, especially my parents, Shanker and Anita Paudel.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

There are several reasons analysts prefer a hexahedral mesh when analyzing three-dimensional models. As indicated by Shepherd[1], these reasons include the fact that a) tetrahedral meshes can require up to 10 times more elements than hexahedral meshes to obtain the same level of accuracy, b) some types of numerical approximations such as high deformation structural finite element analyses[2, 3], tetrahedral elements will be mathematically stiffer due to a reduced number of degrees of freedom associated with a tetrahedral element[4, 5], a condition known as tet-locking, c) there is often a specific requirement imposed by the intended analysis code where only hexahedral elements are acceptable and d) there is often sometimes simply a built in preference by some analysts to utilize a hexahedral mesh.

However, even using hexahedral elements, the initial mesh might not accurately represent the physics of the problem[6]. It is often beneficial to increase element density (i.e. to refine) in the mesh where the error estimates are high or where there are small features in the geometry[7]. Providing a refined mesh reduces the error in the solution, models the geometry accurately, and also increases the resolution in the areas of high stress gradients[8]. Hence, refinement provides more accuracy and efficiency for finite element solutions.

The desired mesh should be refined enough to allow a high level of accuracy in the analysis and coarse enough so that computational time is minimized. Refining the entire model may increase the computational time without sufficiently increasing the accuracy of the results in

some parts of the model. Therefore, there is a desire to be able to refine only in the areas of model that will result in increased accuracy.

To define the element density, sizing functions based on error estimates, geometric features of the model, and stress and strain gradients are often used[8, 9]. Some effective element density distributions can be determined simply, from the analyst's experience using manual mesh refinement tools[10] or from comprehensive geometric reasoning algorithms that automatically compute a sizing function based on features of the model[7]. The resulting sizing function can often be used by sophisticated mesh generation algorithms[10] to compute a well graded meshes. However, element densities based on error estimates and high stress/strain gradients are normally not known before an initial analysis is run. Hence, the optimum size of the elements in a mesh cannot be determined until at least an initial finite element analysis is performed. As a part of the analysis, many computational tools will also produce an approximate error associated with each element of the mesh. To determine an optimum size an analysis can be performed on a coarse mesh and the error approximation results from that analysis used to compute a sizing function to help define the target node density for subsequent refinement operations.

Mesh adaptation based on a sizing function is not a new topic. Procedures that incorporate quadrilateral, triangular, and tetrahedral mesh adaptation that rely on error-based sizing functions are available in the literature[8, 11]. In addition, it is also noted that there are few techniques that generate initial hexahedral mesh using the geometry features of the model to develop a sizing function[12]. However, conformal hexahedral mesh refinement, based on an error-based sizing function, has not been effectively addressed in the literature.

Currently, the use of sizing functions for mesh adaptation is limited to tetrahedral, triangle and quadrilateral meshes. Traditional hexahedral meshing methods have not effectively used a sizing function because of connectivity restrictions imposed by common generation techniques such as mapping and sweeping[13-15]. One way to achieve a graded hex mesh based upon a sizing function is to construct an initial constant size mesh generated with traditional hex methods and then supply subsequent refinement operations to achieve the desired sizing. The method proposed in this work would be effective for such an approach.

This thesis presents a method for a conformal hexahedral mesh refinement procedure based upon a sizing function. This work incorporates the hexahedral mesh refinement techniques developed by Parrish[16] with a sizing function to drive refinement. The sizing function used here is developed from computed error estimates. However, other criteria such as feature size or user specification could be included in the sizing function. To validate the method, comparisons between the refined node size and the target node size are presented. The method provides conformal, all hexahedral locally refined meshes based on a developed sizing function.

The remainder of the thesis is organized as follows: Chapter 2 discusses the background work on hexahedral mesh refinement. Chapter 3 introduces a sizing function based on error estimates, describes the algorithm, sets the criteria used for selecting the nodes for refinement, and gives a comparison of the refined mesh size with the target size. Chapter 4 includes examples generated using this new technique. Chapter 5 provides a brief summary of this work and suggests areas of future work.

2 BACKGROUND

As computational power has increased, the need for refined high quality meshes, to gain more accuracy in the finite element modeling, has also increased. The accuracy of finite element solutions can be improved by adapting the mesh that defines the object being modeled. Meshes can be smoothed to improve the quality, which is known as r-adaptation. Another mesh adaptation method, known as p-adaptation, involves increasing the degree of the elements in the mesh. A third type of adaptation, known as h-adaptation, involves increasing or decreasing the number of elements. Coarsening[17] can be used to reduce the number of elements. Although coarsening, r adaptation, and p adaptation are valid methods, this thesis focuses specifically on h-adaptation i.e. refining by increasing the number of elements locally, to increase accuracy.

2.1 Refinement

This work primarily focuses on splitting the current hexahedron three times along an edge, also known as the 3 refinement in the literature. Another refinement technique available in the literature is 2-refinement, which splits the current hexahedron two times along an edge. There are advantages and disadvantages of the 3 refinement process. If higher mesh density is required in concentrated regions, then 3-refinement is usually the best option as it splits one hex into 27 hexes. However, if the sizing function requires a gradual change in the mesh density over the model, then 2-refinement may be a better choice. Figure 2-1 shows 2-refinement and 3-

refinement performed on a single hex. This work utilizes 3-refinement technique as basis for refinement, as it generally requires fewer refinement iterations to arrive at the target size. The same level of refinement can be obtained with 2-refinement but additional iterations may be required. It is also noted that a robust 3-refinement procedure based on the work of Parrish[16] was available at this writing. A robust 2-refinement procedure based on the work of Edgel[9] was not available. It should be noted however that in many cases, 2-refinement will be more desirable. The procedures described in this thesis, while specifically developed for 3-refinement, can be adapted easily for use using 2-refinement.

Another common approach for adapting hexahedral meshes involves introduction of hanging nodes at edge centers as shown in Figure 2-2. This approach is very straightforward to implement, and no transitions to surrounding course hexahedra are required. However, accuracy of the solution at the hanging node interfaces can often be a problem. For this reason, we limit our method to conformal refinement techniques that do not introduce hanging nodes.

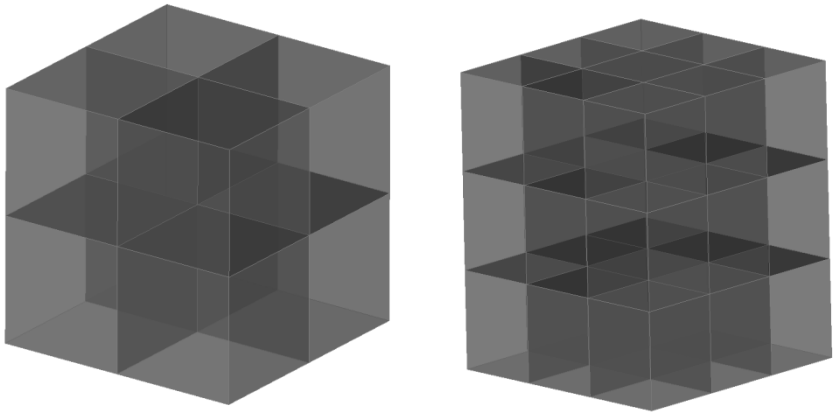

Figure 2-1: 2 refinement and 3 refinement

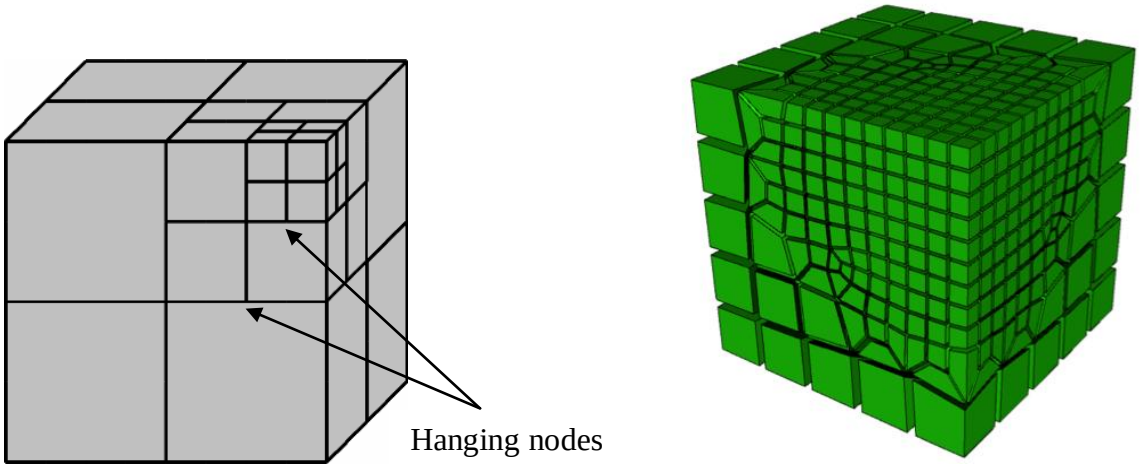

Figure 2-2: Refinement with hanging node and conformal refinement

2.2 Current Methods

Many hexahedral mesh generation techniques have been developed, however fully automatic hexahedral mesh adaptation is the motivation of this research. Adaptation of a hexahedral mesh, by both refining and coarsening, is also an area of interest. A mesh can be adapted before any analysis is run, which for tetrahedral meshing is normally the simplest way to adapt a mesh to a target mesh density. This process requires good knowledge of the physics of problem which, in most cases, is not known a priori. Hence, adaptation for an initial mesh generation may be desirable based on a priori metrics such as geometric features rather a posteriori metrics such as than error estimates or stress strain gradients. As stated previously, however, most initial hexahedral meshing cannot effectively utilize a sizing function. Consequently, grading of a hex mesh may be limited to subsequent refinement operations based upon target element sizes.

There are several methods that use sizing functions to refine the nodes or to adjust the node densities at the time of the initial mesh generation. Quadros, et al.[7], and Zhang and Zhao[12] have introduced mesh refinement using a sizing function based on geometric features

of the model. However, they do not discuss hexahedral mesh refinement based on the error estimates from a finite element analysis. Anderson et al.[8] developed a refining and coarsening technique that uses error estimates as the sizing function, however their method is limited to all quadrilateral elements and does not consider the hexahedral mesh. Zhang and Bajaj[18] introduce hexahedral mesh refinement using volumetric data, but do not consider converting the error estimate from the finite element analysis into a mesh size for refinement. Wada et. al[19] discuss adaptation of hexahedral meshes using local refinement and error estimates, however their method also does not consider comparing the refined size of the mesh to the target size from the error estimates.

As mentioned by Anderson[8], most of the adaptation techniques are limited to triangular and tetrahedral elements. In 2010 Kamenski[20], presented mesh adaptation using the error estimates but his method is limited to triangular elements. De Cougny and Shephard[21], discuss the tetrahedral mesh adaptation but they do not consider a hexahedral technique. Kallinders and Vijayan[11] also discuss tetrahedral and triangular mesh refinement and coarsening but they do not consider hexahedral elements. Babuska et. al[22] have presented a refinement technique based on sizing function derived from the error estimates. Their method is limited to rectangular elements with hanging nodes and do not consider the conformal mesh.

A hexahedral mesh can provide more accurate results and, as mentioned in the introduction section, is often the choice of an analyst. However, hexahedral adaptation techniques are not common. Hexahedral adaptation is a time consuming process, and requires knowledge of physics of the problem so that the generated mesh produces an acceptable error estimate from the finite element analysis. This thesis presents unique and simple criteria to refine

a hexahedral mesh using a sizing function and compares the refined size of the mesh with the target mesh size.

2.3 Sierra Mechanics Refinement Technique

In practice, rather than using a sizing function to drive the refinement, the error measures themselves are often utilized. For example, Sierra[23], an advanced suite of analysis tools, provides three main approaches for driving refinement based upon an error measure. Although these techniques are currently used for tetrahedral and hanging node refinement, they could also be applicable for driving conformal hexahedral refinement in an adaptive analysis. For each of the three approaches, an error metric is computed for each element in the mesh, and the elements are ordered $M_{i=1..N}$ from minimum to maximum error.

1. Percent of Elements: The user provides a threshold, α, which represents a percentage of the total number of elements N in the mesh that will be refined. Starting from the element in M with the highest error and working towards the smallest error, α percent of the elements in the list are identified for refinement.
2. Percent of Max: The user provides a percentage threshold, β that represents the percentage of maximum error in the mesh that will be identified for refinement. For example, if the maximum error of all elements in $M_{i=1..N}$ was 50% with β=90%, then all elements with error > 5% would be identified for refinement.
3. Percent of Total Error: The user provides a percentage threshold, γ, which represents a percent of the total error in the mesh that will be identified for refinement. For example if we represent the total error of all elements in $M_{i=1..N}$ as:

$$\| e \|_{total} = \sum_{i=1}^{N} \| e \|_i \qquad (2\text{-}1)$$

Starting from the element in *M* with the highest error and working to the smallest error, those elements that contribute to a total error of γ. $||e||_{total}$ would be identified for refinement.

For the Sierra Mechanics examples described above, refinement is performed based upon one of the approaches, followed by subsequent analysis iteration. Following each iteration, elements are once again identified for refinement. This procedure continues until a convergence or error threshold has been achieved.

For this work, rather than using the error measure directly, first the error measure is interpreted as a function of element size using the Equation 3-2 as will be shown in section 3. Thus, a sizing function is developed from calculated error estimates. This provides the opportunity to utilize the sizing function as a general field to drive meshing or refinement. It also provides a field for which we can validate the resulting refinement operations to determine the effectiveness of the refinement algorithms at reaching the desired size.

3 HEXAHEDRAL MESH REFINEMENT

For the work of this thesis, it is desired to have the size of a mesh be as close as possible to the sizes provided by the sizing function in order to obtain high computational accuracy in the results without significantly increasing the computation time. This chapter includes information on sizing functions, tools and requirements for the algorithm developed in this thesis, the outline of the algorithm, a discussion and comparison of refined mesh sizes with the target mesh sizes.

3.1 Sizing Functions

Sizing functions are used as the mechanism for refining a mesh. There are several ways to generate sizing functions. Error estimates can be used to define a sizing function. Geometric characteristics, curvature, and sharp features in the model can also be used to define a sizing function[7]. Other bases for sizing functions include: the stress or strain gradients, change in the material properties, points of application of loading, and the location of boundary conditions.

For this thesis, only developing a sizing function based on the error estimate of an initial calculation will be considered. The error estimate should be robust enough to ensure the increase in accuracy of the results, and also steer the adaptation in the desired area of the model. The generation of the error estimate is an important area of study. For this work, error estimates are obtained from the existing finite element code. Physical phenomenon in engineering and sciences can be modeled using partial differential equations. However, complex mathematical

models using the partial differential equations might not have an analytical solution. Fortunately, finite element analysis can provide an approximate solution to these complex models[24]. As these solutions are an approximation to the analytical solution, there are several sources of error. One of the major contributors to the error is the discretization of the model. This error can be minimized using the higher order elements, also known as p adaptation, or increasing the number of elements in the area where the error is high known as h-adaptation. This thesis will be using the h-adaptation technique to reduce the error in the discretization of the model.

In this thesis error estimates are computed using existing methods. As cited by Grastch and Bathe[24], the computation of error estimates and using it as criteria for refining the region where error estimates are high should be computationally cheaper than refining the entire model. The error estimates should be accurate enough to closely represent the unknown actual error. The goal of the error estimates is to steer the mesh adaptation. For this work, the built in error estimates produced by the simulation code, ADINA[25] are used.

3.2 Tools and Requirements

This thesis incorporates a sizing function developed from computed error estimates and incorporates existing hexahedral refinement techniques to adapt a given hexahedral mesh. The technique developed here produces a conformal, locally refined all hexahedral mesh.

The error estimates from the finite element analysis approximate the expected error produced from the numerical model. This error can be reduced using a higher element density in that region. As discussed in the background section, the technique used here uses 3-refinement, which subdivides 1 hex into 27 hexes and applies appropriate templates in the transition zone to ensure that there are no hanging nodes. This 3-refinement was chosen over 2-refinement, (which

subdivides 1 hex into 8 hexes), because it is simple to implement, it takes fewer iterations to obtain the same level refinement with 3-refinement than with 2-refinement, and currently 2-refinement is not as robust as 3-refinement.

The refinement process implemented in this thesis is driven by a sizing function generated from error estimates. The error estimates from the finite element solvers are converted into an Exodus II[26] file, a random access, machine independent, binary file, that is used as a sizing function by the mesh generating toolkit, CUBIT[10]. The Exodus file format stores all the information about the initial mesh. It can be used for input and output of results and can also be used for post-processing of results. The Application Programming Interface (API) to create the exodus file is available in the public domain and a manual to create such file is also available. An example to create an exodus file to drive the refinement process can be found in Appendix B.

3.3 Algorithm

The proposed algorithm refines a hexahedral mesh locally based on a sizing function. The sizing function used here is based on the error estimates generated from a finite element analysis and is applied in order to reduce the error in the model using an adapted mesh and also to decrease the computational time without loss of the accuracy in the solution of finite element analysis. The main goal of this algorithm is to complete the refinement process without the need for a user to determine which area in the mesh to refine. The error estimate, used to define the sizing function, determines whether a node should be considered for refinement or not. Usually, after the mesh has been refined, the quality of the elements degrades. The degradation is usually a result of insertion of templates in the transition zones between the refined and non-refined regions. Hence, smoothing is performed on the elements within and near the refinement region

to improve the element quality. A flowchart of the algorithm is shown in Figure 3-1. The steps are enumerated below and explained in detailed in the next section using an example quarter piston model:

1. The Hexahedral Mesh to be refined is given along with a sizing function generated from error estimates in a binary file format, and desired minimum error threshold are provided as input.
2. Each node is assigned with an error estimate using the scalar values read from the binary file.
3. Nodes with error estimates greater than minimum error threshold, γ, are identified for refinement for the first level of refinement, similarly the nodes requiring second and third level of refinement are identified.
4. Target sizes, computed as a function of the nodal error estimates, and the average of the length of edges attached to the nodes, are computed for each node
5. Step 3 and 4 are repeated, if more refining is required
6. If the mesh is refined to the desired size then the mesh is smoothed.

3.4 Algorithm Example

This section outlines the input, refinement criteria, and comparison of target and current sizes, of the algorithm. An example is used to explain the steps outlined in the algorithm. For simplicity, the algorithm example section is further divided into three sub-sections: input, refinement criteria and comparison of current size and target size.

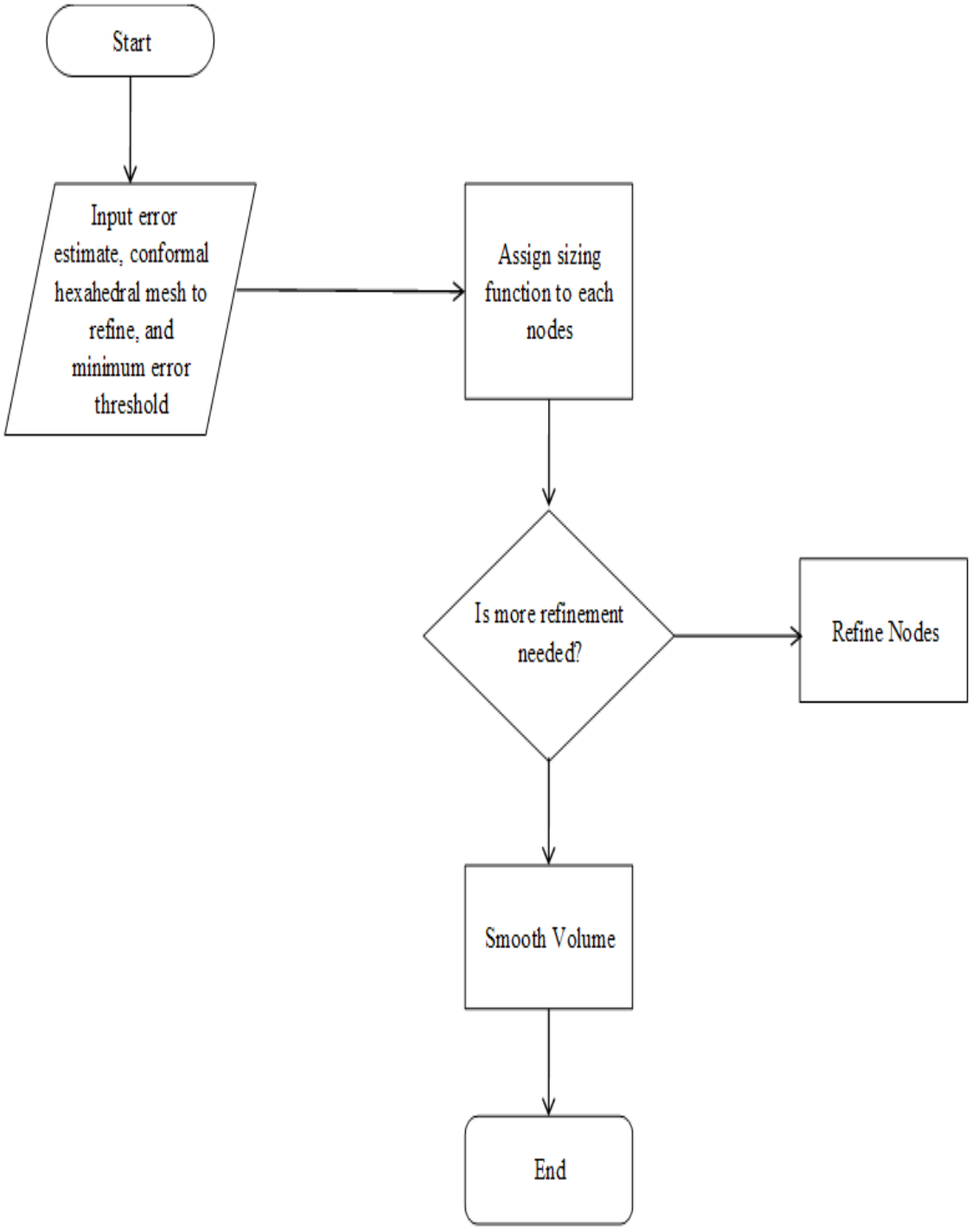

Figure 3-1: Algorithm flowchart

3.4.1 Input

For this example, a quarter piston modeled with a load on its base plate is used. Initially, a coarse mesh, as shown in Figure 3-2, is generated using an all hexahedral meshing technique [10]. Next, the appropriate boundary conditions and loading, are applied. The initial mesh and boundary conditions are then exported to ADINA[25] to perform the finite element analysis. Error estimates are generated as a part of the ADINA analysis[25] and the band plot of the error estimate is shown in Figure 3-3. The error estimate from the finite element analysis is then written in Exodus II, a binary file format, and is used to compute the sizing function to drive the refinement process.

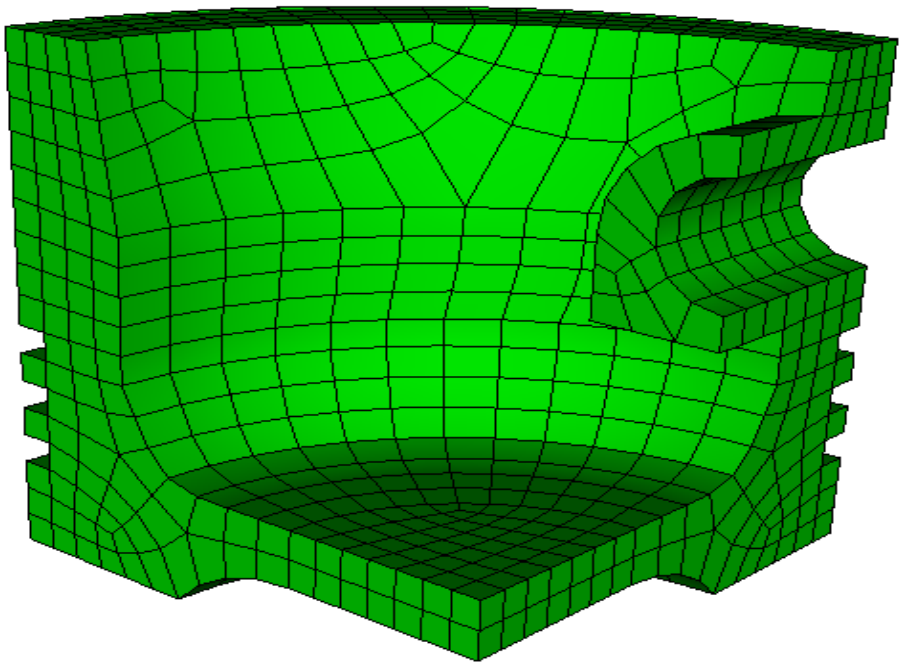

Figure 3-2: Initial coarse mesh of quarter piston

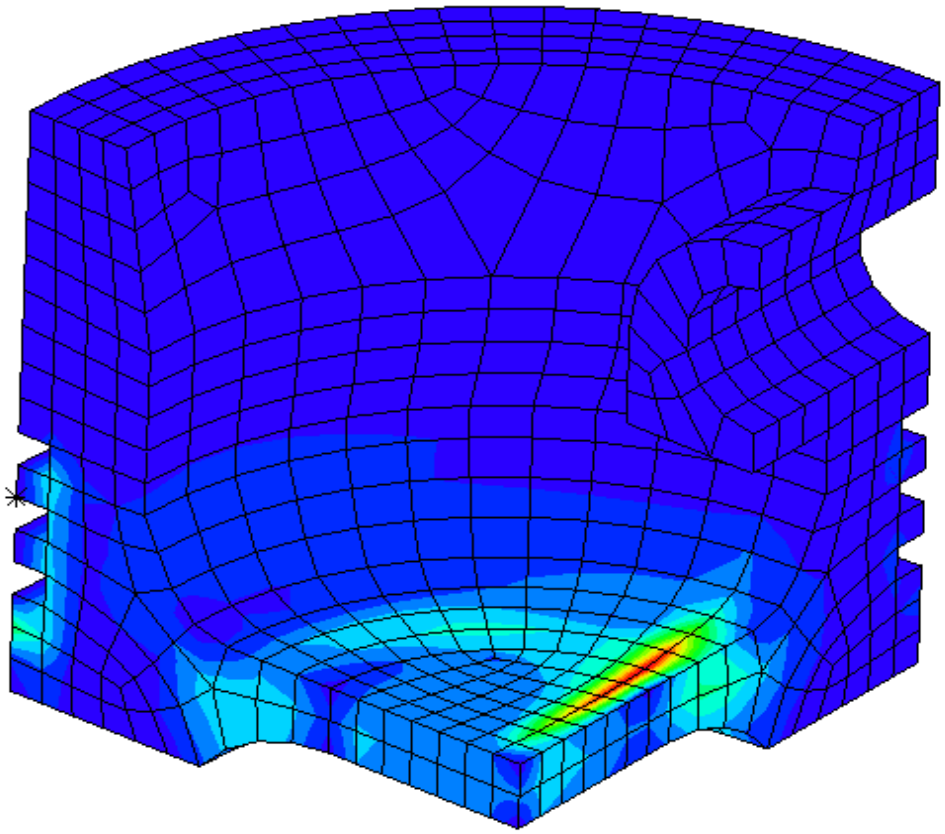

Figure 3-3: Band plot of error estimate to define the sizing function for quarter piston

3.4.2 Refinement Criteria

After each node is assigned a scalar error estimate, it is compared with the minimum threshold allowable error, γ, provided by the user, for the particular problem. Normally the value for γ and the error estimates in the binary file are scalar values between 0 and 100, representing a percent error. Nodes with error estimates greater than the allowable user defined error are identified for further refinement. Nodes with error estimates lower than γ are not refined but may be subsequently smoothed to improve the mesh quality. In this section the specific criteria will be presented, based on the sizing function computed from error estimate, to identify the nodes needed for refinement. The terms basic to this technique are defined below.

The "size" of a node, h_a, is computed using Equation 3-1. h_a is the average of the lengths of edges attached to the node.

$$h_a = \frac{1}{n}\sum_{i=1}^{n} l_i \tag{3-1}$$

where h_a = size of a node

l_i = length of i^{th} edge attached to the node

n = number of edges attached to the node

The relationship between error and mesh size can be approximated, for elasticity and heat problems, from the Poisson heat equation[27] as:

$$|e| = Ch^2 \tag{3-2}$$

where e = error estimate at the node

C = a constant

h = the element edge length

If the error and the element edge length for a node in the mesh are known, then C_n for that node can be computed as:

$$C_n = \frac{e_n}{h_a^2} \tag{3-3}$$

where C_n = a constant for the node

e_n = error estimate at the node

h_a = size of a node as defined in Equation (3-1)

The user provides a threshold error measure for the entire mesh, at which below is acceptable. For this work, target size is computed from the error measure but it can also be computed from the geometric feature or size provided by the user. Then, a target size from the above equation at the node is computed as:

$$t_s = \sqrt{\frac{\gamma}{C_n}} \tag{3-4}$$

where t_s = target node size

γ = minimum allowable error provided by user

C_n = a constant computed from Equation 3-3

The size ratio, S_r, is defined in Equation 3-5 as the ratio of the target size of the node, t_s, determined using the error estimates, to the actual size of the node, h_a.

$$S_r = \frac{h_a}{t_s} \tag{3-5}$$

where S_r = node size ratio

h_a = actual node size as defined in Equation 3-1

t_s = desired node size determined from the error measure described later in equation 3-3

The terms defined in Equation 3-1 through 3-5 are used to identify the nodes that require refinement. It is assumed that the nodes with a size ratio of one or equal to one are acceptable and need no refinement. Similarly, if the size ratio at the node approaches 3, then it indicates that at least one refinement operation should be performed. The value of 3 is assigned because 3-refinement is used as the mechanism for refinement. It is assumed that when the refinement is performed the size of a node will decrease by a factor of three and hence a size ratio of one is obtained after the refinement is performed. Since each split operation will reduce the local element size by a factor of 3, as the size ratio approaches 9, the node will be marked for two split operations. Likewise, a size ratio that approaches 27 will be marked for three splits. Because of the discrete nature of the refinement operations, size ratio thresholds at which additional refinement operations will be performed. For this work, using an average of powers of three for the thresholds seemed to provide acceptable results. Table 3-1 summarizes the approach.

Table 3-1 Size ratio range and number split operation

Size Ratio Range	Number of refinement split operation
0 – 2	0
2 – 6	1
6 -18	2
>18	3

If the size ratio is below the allowable threshold, γ, then no refinement is performed on the node. It is assumed that it is perfect size or it is over refined. The nodes with the size ratio between 2.0 and 6 are identified for the first level of refinement. These nodes are identified for

the first level of refinement because they have size ratio near 3 and less than 9. Hence, after the refinement their size ratio should come close to 1. Similarly, the nodes with size ratio between 6 and 18 are identified for the second level of refinement based on the fact that these nodes have size ratio near 9 and less than 27. Hence, when they are refined the new size ratio should be close to 1. The nodes with error estimates greater than 18 are identified for three levels of refinement. This criteria for refinement is continued until less than 10% of total nodes meet the size ratio criteria. This 10% is chosen to ensure that computation time is not wasted performing a refinement that will not gain a significant level of accuracy in the finite element solution. This criteria serves as the exit criteria for the refinement process. Figure 3-4 shows the refined mesh of the quarter piston with load on its base.

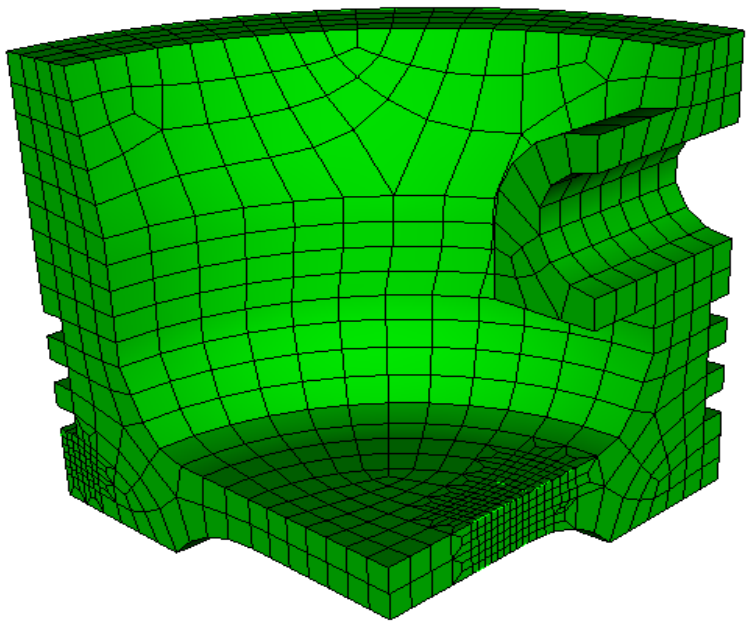

Figure 3-4: Refined mesh of quarter piston

3.4.3 Comparison of Current and Target Size

One of the measures to determine if the algorithm is performing adequately to achieve the desired accuracy in the results is to compare the refined size of node to the target size of the

node. The node size ratio, S_r, is used as the criteria to compare the efficiency of the algorithm. It is assumed that the algorithm should perform such that the size ratio of all the nodes should be less than or equal to 1.0. If a coarsening algorithm were to be implemented, size ratios less than 1 would be minimized and all the size ratios would be close to 1.0. It is recognized that an exact match everywhere where S_r=1 is impossible, however the approach presented here can be validated by statistically examining how close final mesh matches the intended sizing function.

Based on the Equation 3-5, a size ratio for the node before the refinement and after the refinement is computed. The Tables 3-2 and 3-3 provide the percentage of volume falling in the particular size ratio before and after refinement respectively. Figure 3-4 represents the Tables 3-1 and 3-2 in the bar graph. Theoretically, there should not be any change in the volume for size ratio below 1.0 and the volume with size ratio greater 3 should be reduced significantly after the refinement. Also, there should be less than 10% of the volume that fall in the size ratio greater than 2.0. The plot in Figure 3-5, shows that there is not much change in the volume for size ratio below 1.0 and there are not that many volumes with size ratio greater than 3. Although there are few volume greater than 2.0, this condition prevails because the refinement is deemed complete if there are less than 10% of nodes that require refinement. In Figure 3-6, change in the percentage of volume falling in the particular size ratio before and after refinement is shown. It is noticed that most of the change is around size ratio 1.0 and there is negative change in volume for size ratio greater than 2.0. This shows that most of the nodes have size ratio 1.0 after the refinement and refinement is taking place in the node with size ratio greater than 2.0.

Table 3-2: Size ratio of quarter piston before refinement

Size Ratio, S_r	% of Volume	Size Ratio, S_r	% of Volume	Size Ratio, S_r	% of Volume	Size Ratio, S_r	% of Volume
0-0.1	1.493	0.9-1.0	2.62	1.9-2.0	1.420	2.9-3.0	1.632
0.1-0.2	5.874	1.0-1.1	1.39	2.0-2.1	1.539	3.0-3.1	0.573
0.2-0.3	4.654	1.1-1.2	1.740	2.1-2.2	1.173	3.1-3.2	0.395
0.3-0.4	10.876	1.2-1.3	0.79	2.2-2.3	1.902	3.2-3.3	0.214
0.4-0.5	10.837	1.3-1.4	0.467	2.3-2.4	1.248	3.3-3.4	0.385
0.5-0.6	16.694	1.4-1.5	1.124	2.4-2.5	1.921	3.4-3.5	0.361
0.6-0.7	14.019	1.5-1.6	1.004	2.5-2.6	1.083	3.5-3.6	0.442
0.7-0.8	2.518	1.6-1.7	1.162	2.7-2.8	1.063	3.6-3.7	0.274
0.8-0.9	2.299	1.8-1.9	1.325	2.8-2.9	1.075	>3.7	2.384

Table 3-3: Size ratio of quarter piston after the refinement

Size Ratio, S_r	% of Volume	Size Ratio, S_r	% of Volume	Size Ratio, S_r	% of Volume	Size Ratio, S_r	% of Volume
0-0.1	1.490	0.9-1.0	6.864	1.9-2.0	1.438	2.9-3.0	0.000
0.1-0.2	5.720	1.0-1.1	5.109	2.0-2.1	1.545	3.0-3.1	0.000
0.2-0.3	4.659	1.1-1.2	2.206	2.1-2.2	0.192	3.1-3.2	0.000
0.3-0.4	10.775	1.2-1.3	2.775	2.2-2.3	0.000	3.2-3.3	0.000
0.4-0.5	10.845	1.3-1.4	1.563	2.3-2.4	0.000	3.3-3.4	0.000
0.5-0.6	16.931	1.4-1.5	1.236	2.4-2.5	0.000	3.4-3.5	0.000
0.6-0.7	14.435	1.5-1.6	1.945	2.5-2.6	0.000	3.5-3.6	0.000
0.7-0.8	2.842	1.6-1.7	1.779	2.7-2.8	0.000	3.6-3.7	0.000
0.8-0.9	3.758	1.8-1.9	1.887	2.8-2.9	0.000	>3.7	0.000

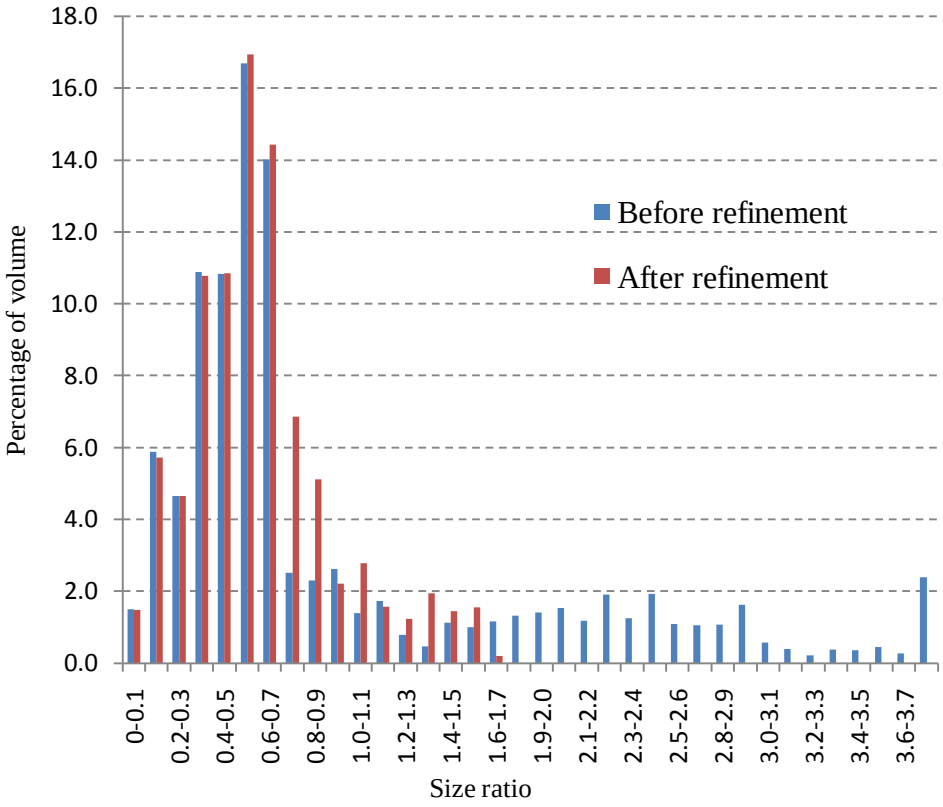

Figure 3-5: Plot of size ratio and percentage of total volume before and after refinement

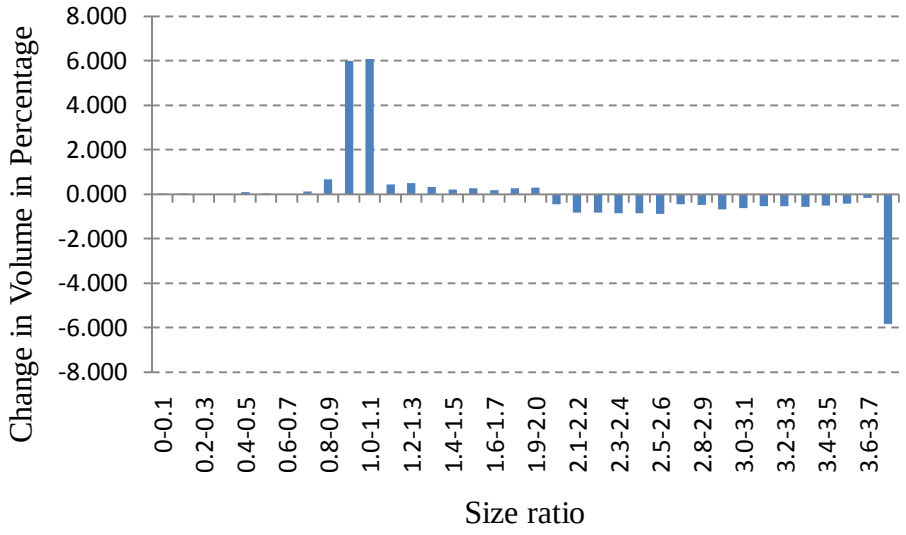

Figure 3-6: Plot of size ratio and change in volume in percentage

4 EXAMPLES

This chapter includes examples of all hexahedral sizing based refinement. Three examples are presented. Each example shows the initial coarse mesh, the band plot of the stress error provided by the finite element analysis, the refined mesh, and a table showing the results of refinement. All initial meshes were generated with CUBIT[10] using the existing meshing techniques. The finite element analysis was performed using the commercial software ADINA[25].

The goal of this algorithm is to obtain size ratios less than 1.0, which achieves the allowable error estimate. It is not possible to obtain all the nodes with size ratio 1.0 but however the goal of this thesis is to get most of the nodes to have size ratio close to 1.0 and 2.0. When a node is refined and three nodes are created, it is difficult to get size ratio exactly 1.0. Also, addition of templates in the transition zones often over refines the mesh.

The results shown in this chapter are promising; most of the nodes have a size ratio less than 1.0. As expected, there are very few nodes that have size ratio greater than 1.0.

4.1 Quarter Piston with Load in the Cylindrical Cavity

This problem is similar to the example presented in the previous chapter. In this case, the loading conditions are different. The cylindrical cavity is pressurized. The initial mesh is shown on the left side of Figure 4-1 and on the right a band plot of the stress error, with red indicating a

high error. The maximum error in this mesh was 22%. Observe that the error is high around the regions where loads are applied. For this example a minimum threshold error was chosen to be 6%. This minimum was chosen so that refinement actually increases the accuracy in the results. This example, again illustrates how the sizing function based on the error estimate can be used to refine the model.

In Figure 4-4 the refined mesh is shown. It can be observed that the refined mesh provides the high density nodes around the area where the error estimates from the finite element analysis are high. The refined mesh and appropriate boundary conditions are again exported to ADINA to perform the finite element analysis on the refined mesh. The Figure 4-5 shows the bland plot of the error estimate after the refinement and it can be observed that the error has been significantly reduced. The maximum error estimate with the refined mesh dropped to 4.7% from 22% with the coarse mesh.

In Table 4-1 the size ratio and the volume of hexes that have fallen in that size ratio before the refinement are presented and in Table 4-2 the size ratio and the volume of hexes that have fallen in that size ratio range after the refinement are presented. It can be seen that most of the volumes are below size ratio 2.0. It can be noted that there is low percentage of the volume that are above 2.0 size ratio. Again, the goal of this algorithm is to maximize the percentage of the volume below size ratio 2.0. In Figure 4-2, the values from Table 4-1 and Table 4-2 are presented in the bar graph. In Figure 4-3, the change in the percentage of volume falling in the particular size ratio before and after refinement is shown. It is noticed that most of the change is around size ratio 1.0 and there is negative change in volume for size ratio greater than 2.0. This shows that most of the nodes have a size ratio of 1.0 after the refinement and refinement is taking place in the nodes with size ratios greater than 2.0

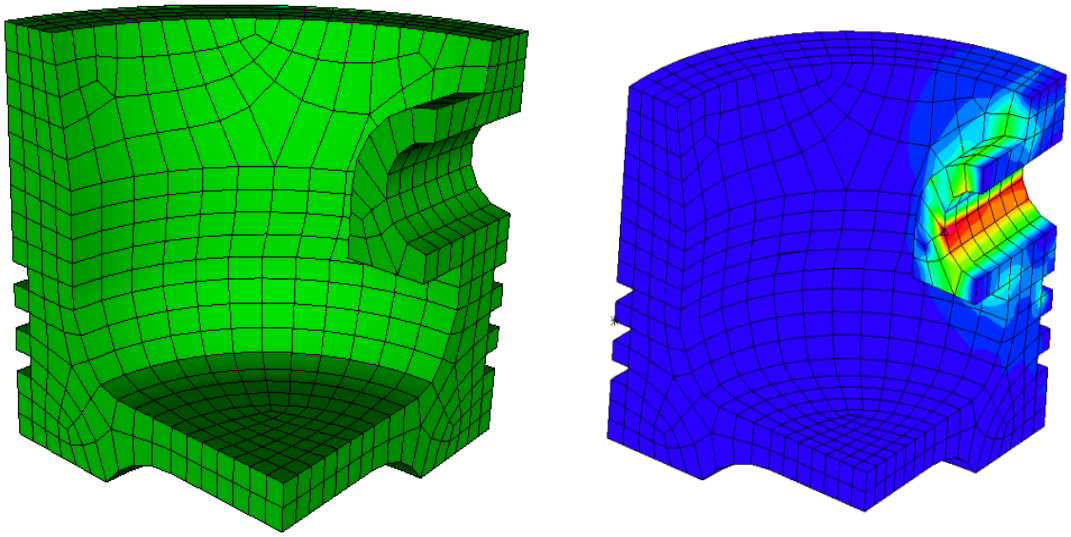

Figure 4-1: Initial Coarse mesh (left) and the band plot of error estimates (right)

Table 4-1: Size ratio for quarter piston before refinement

Size Ratio, S_r	% of Volume	Size Ratio, S_r	% of Volume	Size Ratio, S_r	% of Volume	Size Ratio, S_r	% of Volume
0-0.1	1.623	0.9-1.0	2.154	1.9-2.0	0.861	2.9-3.0	0.851
0.1-0.2	4.189	1.0-1.1	1.562	2.0-2.1	1.286	3.0-3.1	0.424
0.2-0.3	8.112	1.1-1.2	1.624	2.1-2.2	0.249	3.1-3.2	0.367
0.3-0.4	11.152	1.2-1.3	0.856	2.2-2.3	1.254	3.2-3.3	0.214
0.4-0.5	12.865	1.3-1.4	0.957	2.3-2.4	0.784	3.3-3.4	0.524
0.5-0.6	15.264	1.4-1.5	1.029	2.4-2.5	0.896	3.4-3.5	0.865
0.6-0.7	15.119	1.5-1.6	0.995	2.5-2.6	0.981	3.5-3.6	0.962
0.7-0.8	2.689	1.6-1.7	1.037	2.7-2.8	0.784	3.6-3.7	0.663
0.8-0.9	2.321	1.8-1.9	1.121	2.8-2.9	0.953	>3.7	2.399

Table 4-2: Size ratio for quarter piston after refinement

Size Ratio, S_r	% of Volume	Size Ratio, S_r	% of Volume	Size Ratio, S_r	% of Volume	Size Ratio, S_r	% of Volume
0-0.1	1.623	0.9-1.0	7.126	1.9-2.0	1.051	2.9-3.0	0.000
0.1-0.2	4.213	1.0-1.1	6.129	2.0-2.1	0.324	3.0-3.1	0.000
0.2-0.3	8.154	1.1-1.2	1.936	2.1-2.2	0.024	3.1-3.2	0.000
0.3-0.4	11.377	1.2-1.3	1.862	2.2-2.3	0.000	3.2-3.3	0.000
0.4-0.5	12.924	1.3-1.4	1.327	2.3-2.4	0.000	3.3-3.4	0.000
0.5-0.6	15.251	1.4-1.5	1.181	2.4-2.5	0.000	3.4-3.5	0.000
0.6-0.7	15.113	1.5-1.6	1.089	2.5-2.6	0.000	3.5-3.6	0.000
0.7-0.8	2.982	1.6-1.7	1.827	2.7-2.8	0.000	3.6-3.7	0.000
0.8-0.9	2.854	1.8-1.9	1.628	2.8-2.9	0.000	>3.7	0.000

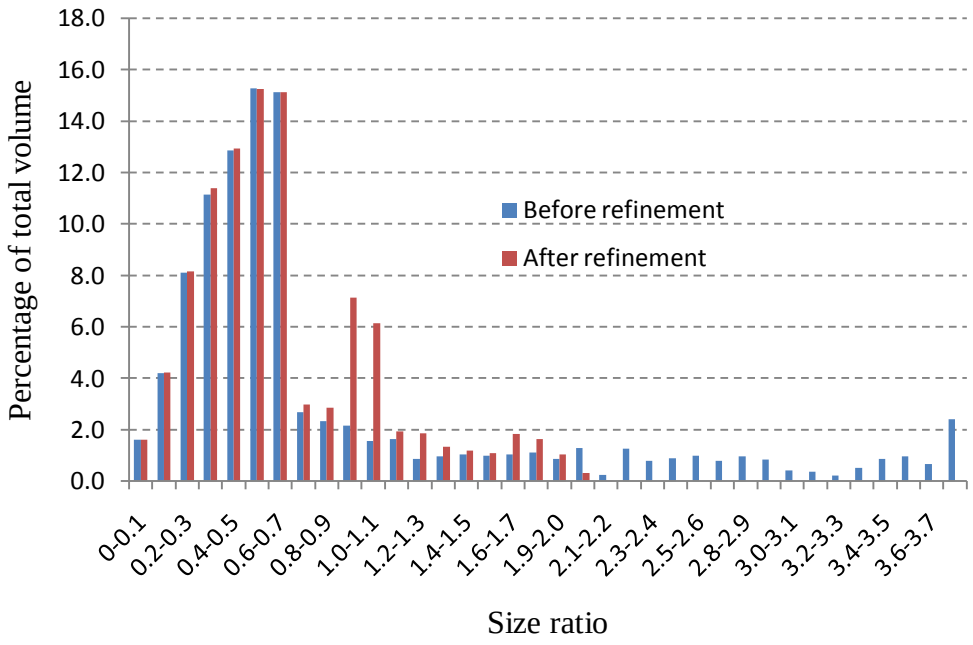

Figure 4-2: Plot of percentage of total volume and size ratio before and after refinement for quarter piston

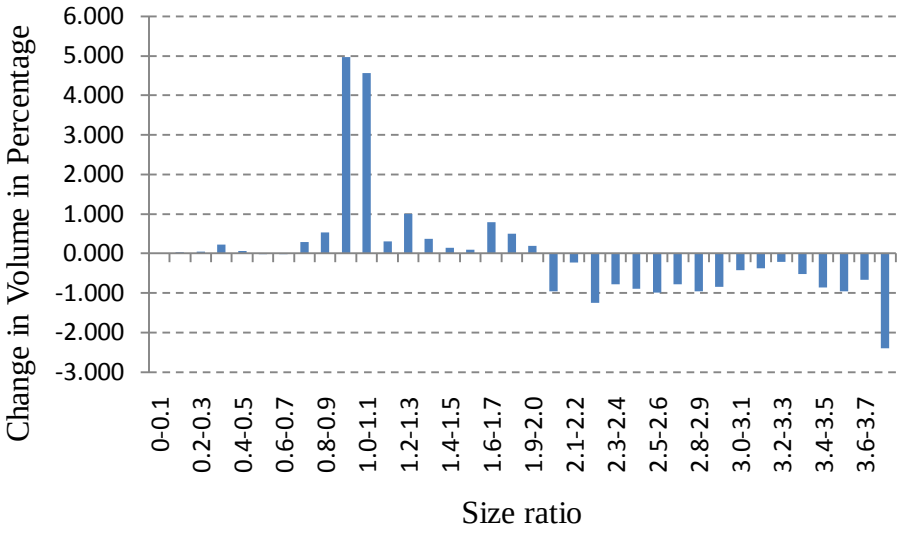

Figure 4-3: Plot of size ratio and change in volume in percentage

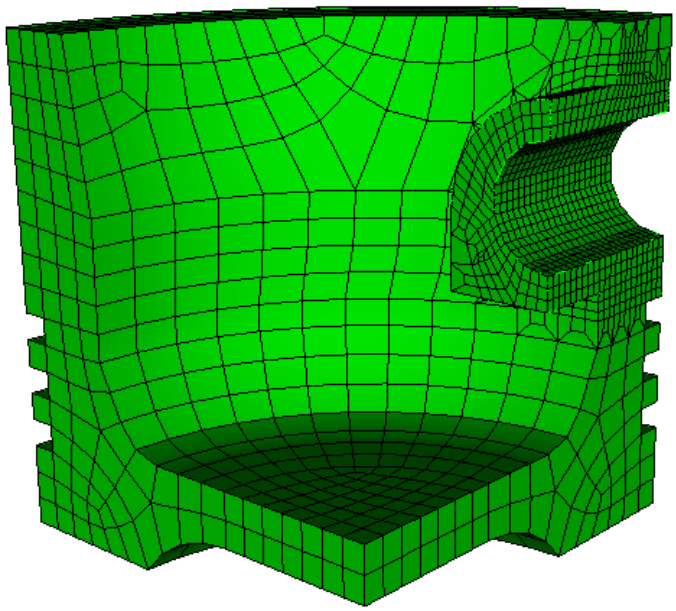

Figure 4-4: Refined mesh of the quarter piston with load in the cylindrical cavity

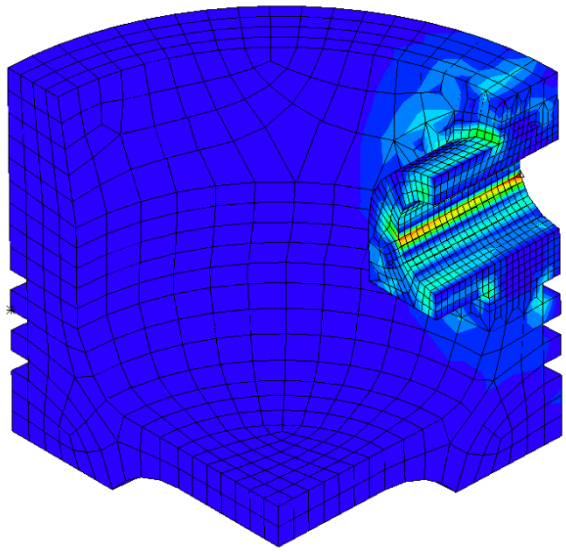

Figure 4-5: The band plot of error estimate with the refined mesh

4.2 Gear

In this example, a gear in rotating about its axis is modeled. Three gear teeth are constrained and a couple is applied at the center of the gear. The Figure 4-6 shows the initial mesh of the gear generated using CUBIT and Figure 4-7 shows the band plot of error estimates. For this example the minimum threshold error used is 8%. Any nodes below this threshold value have not been refined.

Figure 4-10 shows the refined mesh of the gear. It is refined around the teeth, where there is a high error estimate. Table 4-3 presents the size ratio before refinement and Table 4-4 shows the size ratio after refinement. Figure 4-8 is a plot of the size ratio and volume before and after the refinement. Notice that most of the volume is below a size ratio of 2.0. Also the there is not that much of a change in volume for size ratios less than 1.0. In Figure 4-9, the change in the

percentage of volume falling in the particular size ratio before and after refinement is shown. Notice that most of the change is around a size ratio of 1.0 and there is negative change in volume for size ratio greater than 2.0. This shows that most of the nodes have a size ratio of 1.0 after the refinement and that refinement is taking place in the node with size ratio greater than 2.0. Also, it should be noted that when the coarsening algorithm is implemented the volumes should come close to the size ratio 1.0, the ideal element size ratio.

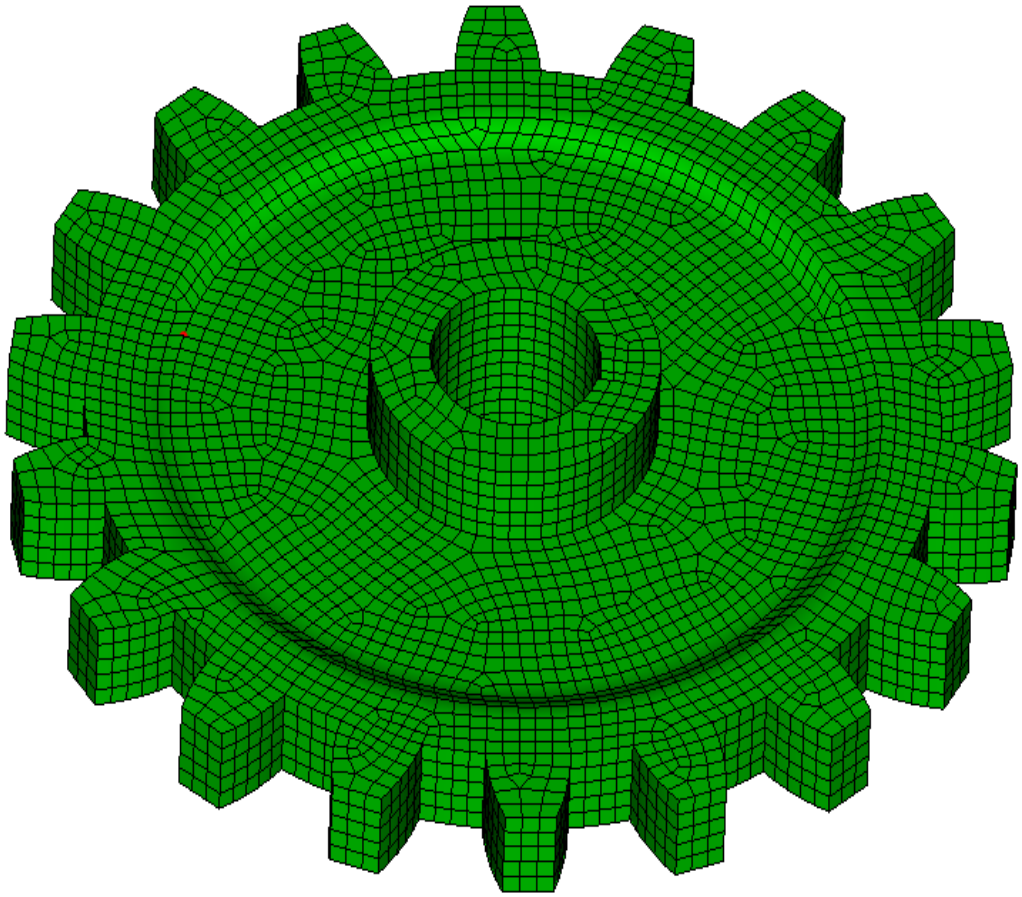

Figure 4-6: Initial mesh of gear

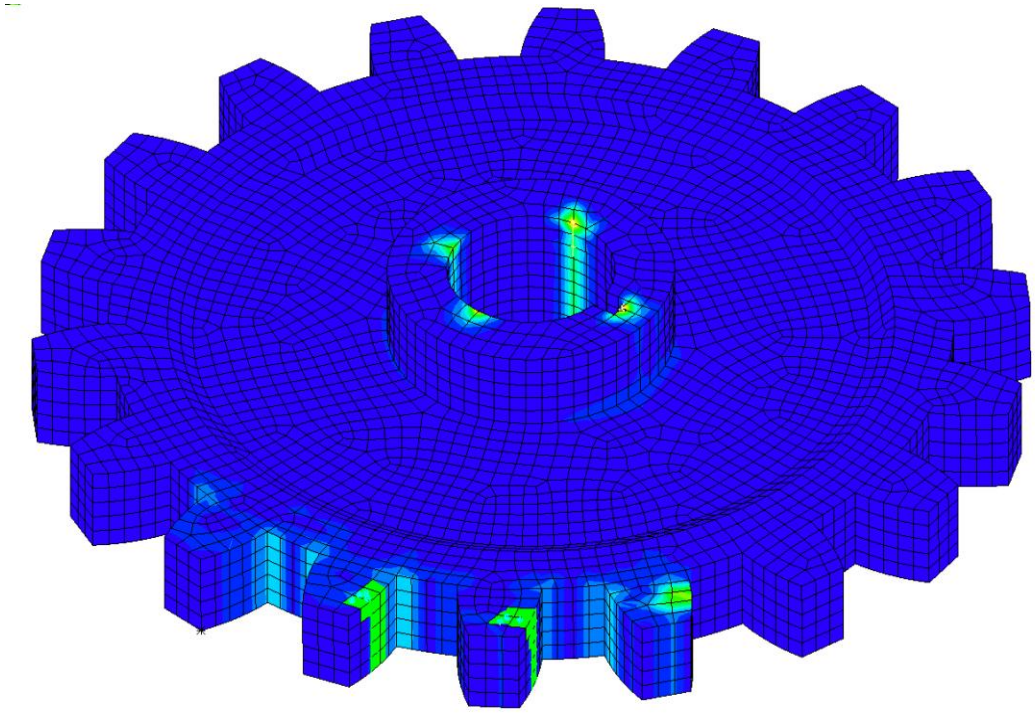

Figure 4-7: Error band plot of the gear

Table 4-3: Size ratio for gear example before refinement

Size Ratio, S_r	% of Volume	Size Ratio, S_r	% of Volume	Size Ratio, S_r	% of Volume	Size Ratio, S_r	% of Volume
0-0.1	1.175	0.9-1.0	1.128	1.9-2.0	0.612	2.9-3.0	0.752
0.1-0.2	3.254	1.0-1.1	1.624	2.0-2.1	0.824	3.0-3.1	0.624
0.2-0.3	10.168	1.1-1.2	0.995	2.1-2.2	0.291	3.1-3.2	0.779
0.3-0.4	12.356	1.2-1.3	0.648	2.2-2.3	0.384	3.2-3.3	0.718
0.4-0.5	12.654	1.3-1.4	0.619	2.3-2.4	0.268	3.3-3.4	0.729
0.5-0.6	17.214	1.4-1.5	0.821	2.4-2.5	0.358	3.4-3.5	0.817
0.6-0.7	19.327	1.5-1.6	0.358	2.5-2.6	0.502	3.5-3.6	0.819
0.7-0.8	1.678	1.6-1.7	0.714	2.7-2.8	0.427	3.6-3.7	0.991
0.8-0.9	1.257	1.8-1.9	0.784	2.8-2.9	0.529	>3.7	2.792

Table 4-4: Size ratio for gear example after refinement

Size Ratio, S_r	% of Volume	Size Ratio, S_r	% of Volume	Size Ratio, S_r	% of Volume	Size Ratio, S_r	% of Volume
0-0.1	1.179	0.9-1.0	6.147	1.9-2.0	0.692	2.9-3.0	0.000
0.1-0.2	3.317	1.0-1.1	5.894	2.0-2.1	0.517	3.0-3.1	0.000
0.2-0.3	10.172	1.1-1.2	2.218	2.1-2.2	0.016	3.1-3.2	0.000
0.3-0.4	12.348	1.2-1.3	1.247	2.2-2.3	0.000	3.2-3.3	0.000
0.4-0.5	12.661	1.3-1.4	0.962	2.3-2.4	0.000	3.3-3.4	0.000
0.5-0.6	17.198	1.4-1.5	0.912	2.4-2.5	0.000	3.4-3.5	0.000
0.6-0.7	19.319	1.5-1.6	0.561	2.5-2.6	0.000	3.5-3.6	0.000
0.7-0.8	1.652	1.6-1.7	0.819	2.7-2.8	0.000	3.6-3.7	0.000
0.8-0.9	1.241	1.8-1.9	0.924	2.8-2.9	0.000	>3.7	0.000

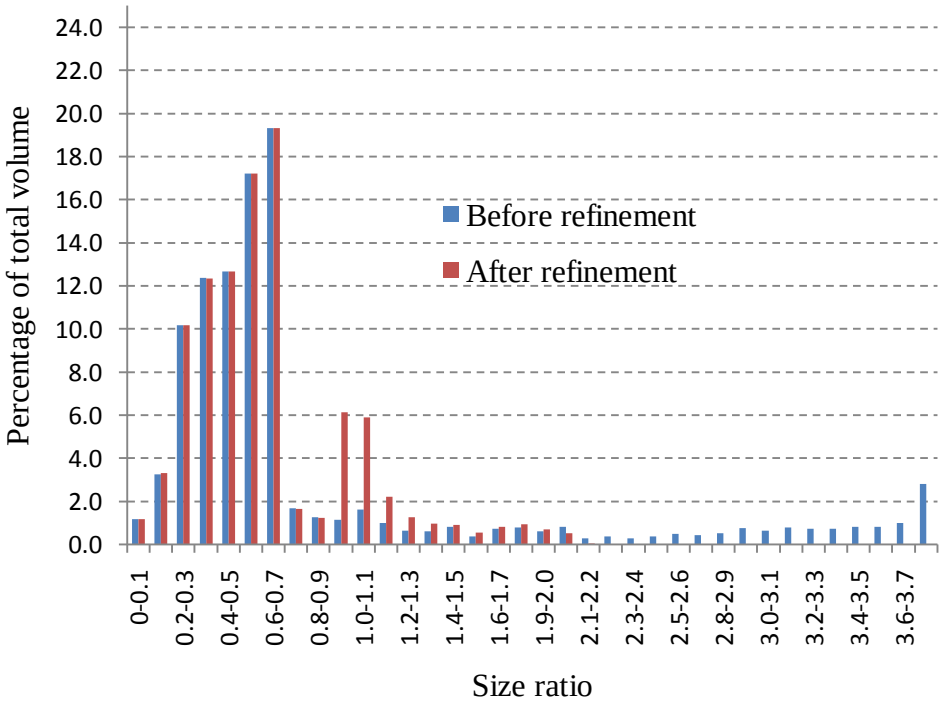

Figure 4-8: Plot of percentage of total volume and size ratio before and after refinement for gear model

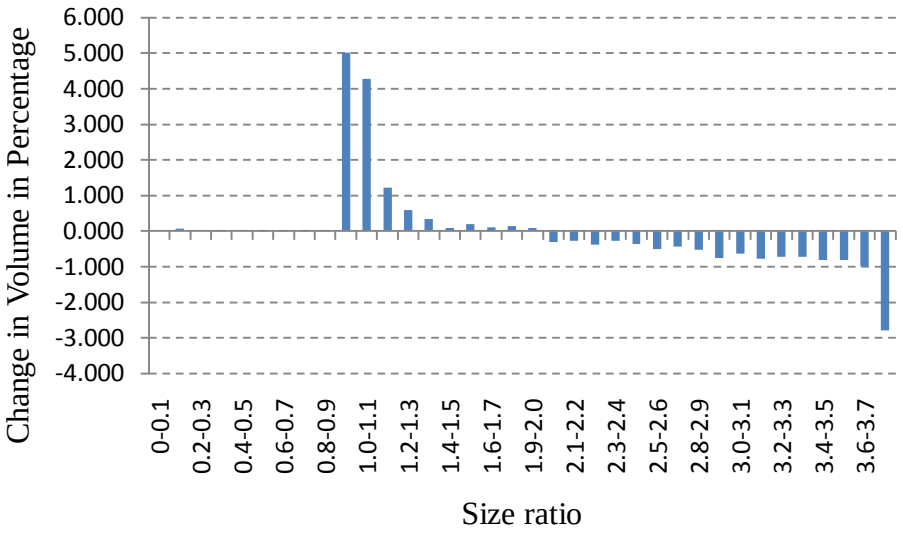

Figure 4-9: Plot of size ratio and change in volume in percentage

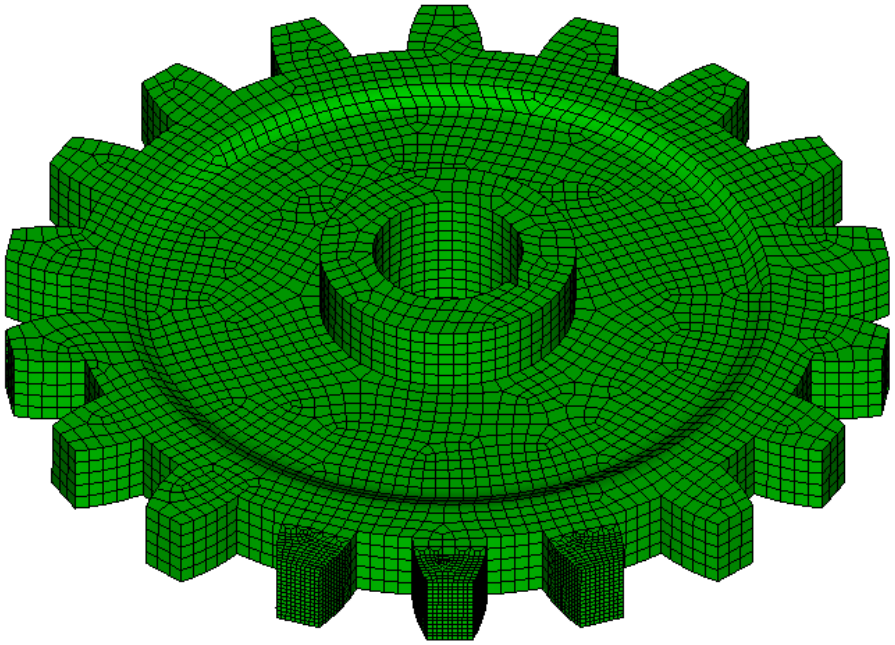

Figure 4-10: Refined mesh of gear

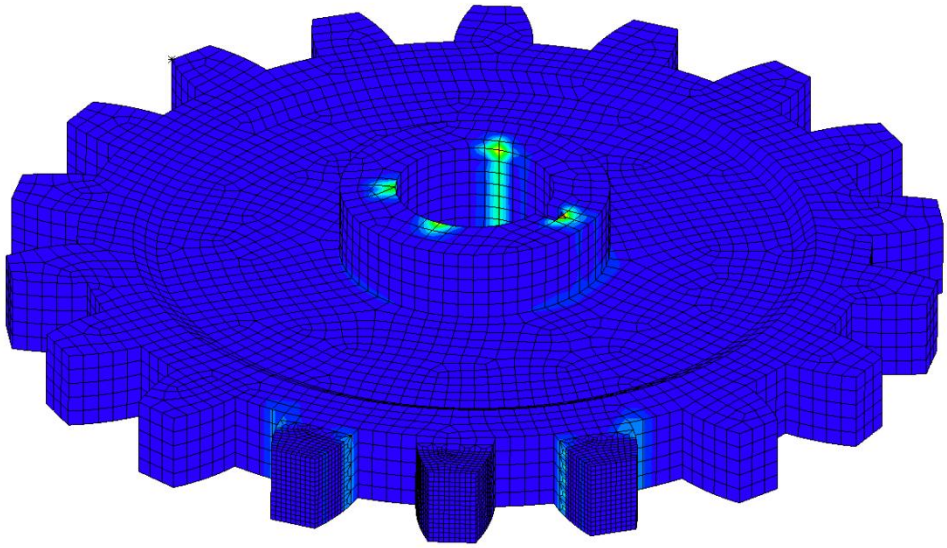

Figure 4-11: Band plot error on the refined gear mesh

4.3 Hook

A coarse mesh of a loaded hook model is shown in Figure 4-12 and also the band plot of error estimate on the coarse mesh is shown next to the coarse mesh. The maximum error with this mesh was 15% which is depicted by the red color in the plot. For this example a minimum threshold error was chosen to be 7%.

In Figure 4-13 the refined mesh of the gear and the error band plot on the refined mesh are shown. The hook is refined around the area where there is high error estimate. Table 4-5 presents the size ratio before the refinement and Table 4-6 shows the size ratio after refinement. Figure 4-14 shows the plot of size ratio and volume before and after the refinement of the model. Notice that most of the volume is below a size ratio of 2.0. Also, there is not much change in volume for size ratios less than 1.0. In Figure 4-15, change in the percentage of volume falling in the particular size ratio before and after refinement is shown. Notice that most of the change is

around a size ratio of 1.0 and there is negative change in volume for size ratios greater than 2.0. This shows that most of the nodes have size ratio 1.0 after the refinement and refinement is taking place in the node with size ratio greater than 2.0.

The error has been reduced significantly on the adapted mesh from 15% to 4.1% which is less than the minimum threshold error.

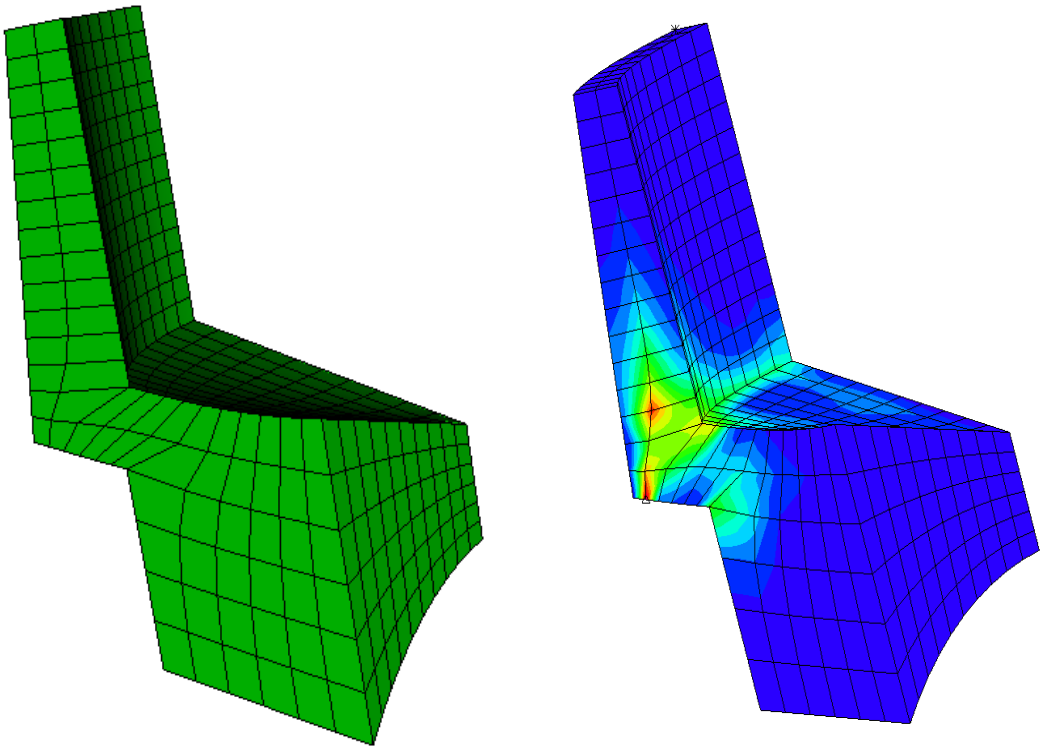

Figure 4-12: The coarse mesh of hook (right) and the band plot of error (left)

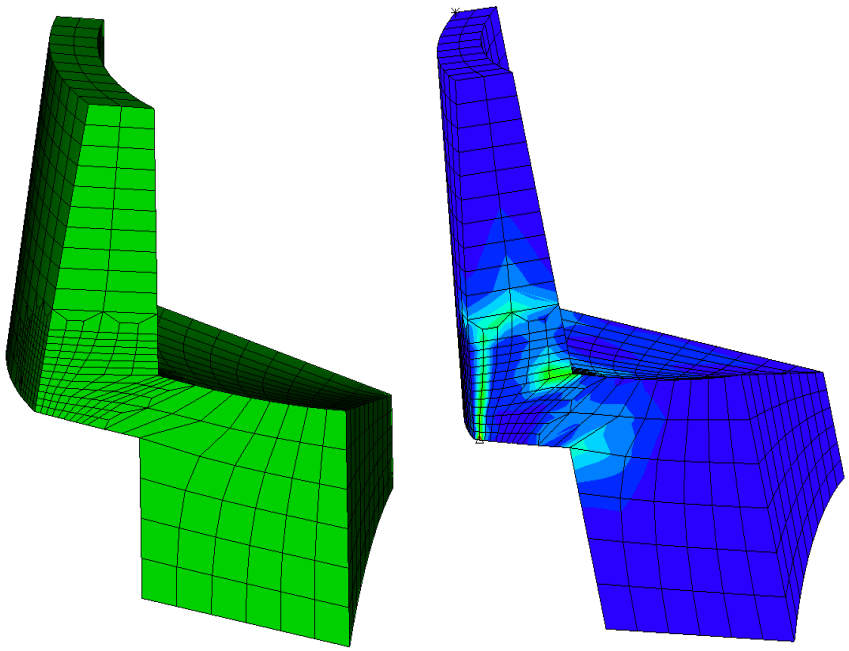

Figure 4-13: Refined mesh and error band plot on the refined hook mesh

Table 4-5: Size ratio before the refinement for hook

Size Ratio, S_r	% of Volume	Size Ratio, S_r	% of Volume	Size Ratio, S_r	% of Volume	Size Ratio, S_r	% of Volume
0-0.1	1.147	0.9-1.0	1.289	1.9-2.0	1.217	2.9-3.0	0.675
0.1-0.2	2.814	1.0-1.1	1.584	2.0-2.1	1.143	3.0-3.1	0.624
0.2-0.3	7.125	1.1-1.2	1.389	2.1-2.2	0.851	3.1-3.2	0.549
0.3-0.4	10.389	1.2-1.3	0.769	2.2-2.3	0.824	3.2-3.3	0.554
0.4-0.5	13.125	1.3-1.4	0.825	2.3-2.4	0.846	3.3-3.4	0.567
0.5-0.6	16.274	1.4-1.5	0.937	2.4-2.5	0.859	3.4-3.5	0.512
0.6-0.7	16.185	1.5-1.6	0.857	2.5-2.6	0.871	3.5-3.6	0.429
0.7-0.8	2.886	1.6-1.7	1.146	2.7-2.8	0.451	3.6-3.7	0.152
0.8-0.9	2.547	1.8-1.9	1.271	2.8-2.9	0.487	>3.7	5.827

Table 4-6: Size ratio after the refinement of hook

Size Ratio, S_r	% of Volume	Size Ratio, S_r	% of Volume	Size Ratio, S_r	% of Volume	Size Ratio, S_r	% of Volume
0-0.1	1.151	0.9-1.0	7.284	1.9-2.0	1.523	2.9-3.0	0.000
0.1-0.2	2.816	1.0-1.1	7.669	2.0-2.1	0.685	3.0-3.1	0.000
0.2-0.3	7.124	1.1-1.2	1.827	2.1-2.2	0.024	3.1-3.2	0.000
0.3-0.4	10.387	1.2-1.3	1.269	2.2-2.3	0.000	3.2-3.3	0.000
0.4-0.5	13.215	1.3-1.4	1.157	2.3-2.4	0.000	3.3-3.4	0.000
0.5-0.6	16.282	1.4-1.5	1.139	2.4-2.5	0.000	3.4-3.5	0.000
0.6-0.7	16.179	1.5-1.6	1.124	2.5-2.6	0.000	3.5-3.6	0.000
0.7-0.8	3.015	1.6-1.7	1.338	2.7-2.8	0.000	3.6-3.7	0.000
0.8-0.9	3.215	1.8-1.9	1.527	2.8-2.9	0.000	>3.7	0.000

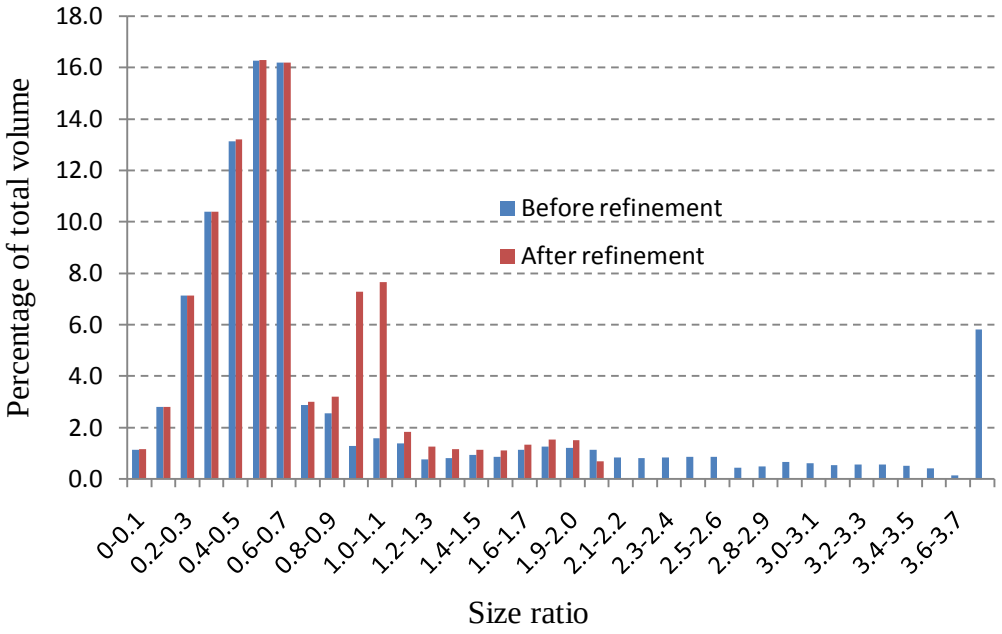

Figure 4-14: Plot of percentage of total volume and size ratio before and after refinement for hook

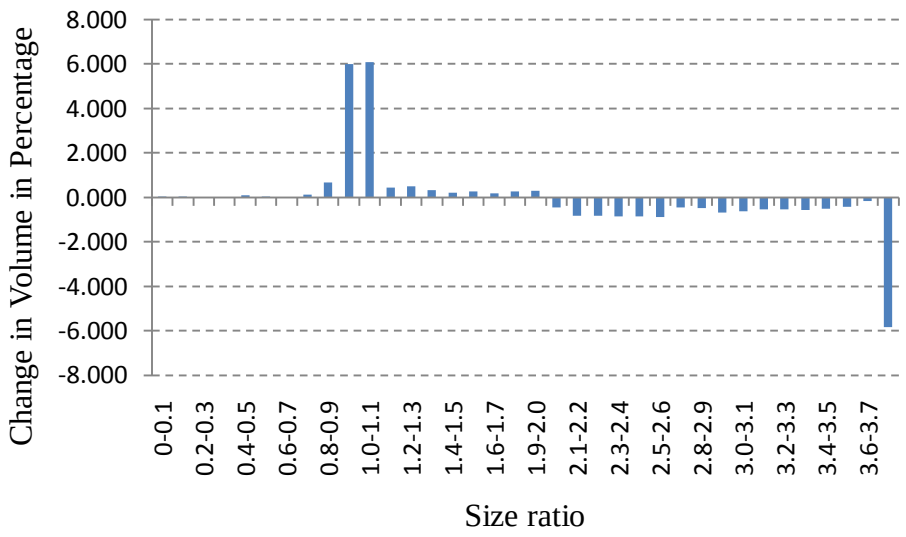

Figure 4-15: Plot of size ratio and change in volume in percentage

5 CONCLUSIONS

This thesis presents a sizing function algorithm that selects the nodes in all hexahedral meshes for refinement and then generates a refined all conformal hexahedral mesh for subsequent finite element analysis. As a result, the locally refined mesh will be able to capture the physics of the problem more accurately, with a minimum increase in the computation time, and provide high efficiency in the finite element solutions.

Combining a sizing function that is obtained from the error estimates with currently available hexahedral meshing techniques, this method refines the regions in the hexahedral mesh where there are high error estimates. Thus, the need for totally re-meshing or manually refining the regions was removed. Hence, it saves the computation time, and also increases the accuracy in the solution of finite element analysis.

5.1 Future Work

This method effectively uses a sizing function to drive the refinement process. The results from the examples shown in this thesis are promising but improvements can be done on this technique to work more efficiently.

This work provides a platform for the total adaptation of hexahedral elements. Only refinement was considered here, but coarsening could be included. There has been recent research on coarsening of hexahedral mesh[17]. Including coarsening would provide a method to

fully adaptive hexahedral meshes. This work showed that there are often nodes with size ratio less than 2.0 when considering only refinement. However, when combined with coarsening, most of the size ratios should fall between 1.0 and 2.0.

Currently, this method uses 3-refinement which sometimes over refines the mesh and does not provide good gradation between refined and coarsened region. When this technique is used with the 2-refinement technique developed by Edgel et al. [8] it should provide more gradation in the refinement.

In this work, sizing function is developed using computed error estimates. However, other criteria such as feature size or user specified field function could be included in the sizing function.

Another area for improvement is that this technique uses Exodus II as an input for the sizing function. Most of the solvers write the error measure as a text file, the user has to create an exodus file from the text file. It would eliminate a step to create Exodus II file if a text file is used.

REFERENCES

1. Shepherd, J.F. and C.R. Johnson, *Hexahedral Mesh Generation Constraints.* Eng. with Comput., 2008. 24(3): p. 195-213.

2. Weingarten, V.I., *The controversy over hex or tet meshing.* Machine Design, 1994: p. 74-78.

3. Cifuentes, A.O. and A. Kalbag, *A performance study of tetrahedral and hexahedral elements in 3-d finite element structural analysis.* Finite Elements in Analysis and Design, 1992. 12(3-4): p. 313-318.

4. S.E.Benzley, et al. *A comparision of all hexagonal and all tetrahedral finite element meshes for elastic and elasto-plastic analysis*. in *4th International Meshing Roundtable*. October 1995. Sandia National Laboratories.

5. Bussler, M.L. and A. Ramesh, *The eight-node hexahedral elements in fea of part designs.* Foundry Management and Technology, 1993: p. 26-18.

6. Lee, N.S. and K.-J. Bathe, *Error indicators and adaptive remeshing in large deformation finite element analysis.* Finite Elements in Analysis and Design, 1994: p. 99-139.

7. Quadros, W.R., et al. *A computational framework for generating sizing function in assembly meshing*. in *14th International Meshing Roundtable*. 2005.

8. Anderson, B.D., S.E.Benzley, and S.J.Owen. *Automated All-Quadrilateral Mesh Adaptation*. in *18th International Meshing Roundtable*. 2009.

9. J. Edgel, S.E.Benzley, and S.J.Owen. *An Adaptive Grid-Based All Hexahedral Meshing*. in *19th International Meshing Roundtable*. 2010.

10. Cubit 12.1 user documentation. Research report, Sandia National Laboratories, Albuquerque, N.M., 2010.

11. Kallinderis, Y. and P.Vijayan, *Adaptive Refinement-Coarsening Scheme for Three-Dimensional Unstructured Meshes.* American Institute of Aeronautics and Astronautics Journal, 1993. 31: p. 1440-1447.

12. Zhang, H. and G. Zhao, *Adaptive hexahedral mesh generation based on local domain curvature and thickness using a modified grid-based method.* Finite Elements in Analysis and Design, 2007. 43(9): p. 691-704.

13. Knupp, P.M. *Next Genration Sweep Tool: A Method For Generating All-Hex Meshes on Two-And-One-Half Dimensional Geometries*. in *7th International Meshing Roundtable*. 1998.

14. Scott, M.A., et al. *Adaptive Sweeping Techniques*. in *14th International Meshing Roundtable*. 2005.

15. Cook, W.A. and W.R. Oaks, *Mapping Methods for Generating Three-Dimensional Meshing* Computers in Mechanical Engineering, 1983. 1: p. 67-72.

16. Parrish, M., et al., *A Selective Approach to Conformal Refinement of Unstructured Hexahedral Finite Element Meshes.* Proceedings, 16th International Meshing Roundtable, 2007.

17. Woodbury, A.C., et al., *Localized Coarsening of Conforming All-Hexahedral Meshes.* 17th International Meshing Roundtable, 2008.

18. Zhang, Y. and C. Bajaj. *Adaptive and Quality Quadrilateral/Hexahedral Meshing From Volumetric Data.* in *13th International Meshing Roundtable,*. 2004.

19. Wada, Y. and H. Okuda, *Effective adaptation technique for hexahedral mesh,*. Concurrency and Computation: Practice and Experience, 2002. 14.

20. Kamenski, L. *A Study on Using Hierarchical Basis Error*. in *19th International Meshing Rountable*. 2010.

21. Cougny, H.L.D. and M.S. Shephard, *Parallel Refinement and Coarsening of Tetrahedral Meshes.* International Journal for Numerical Methods in Engineering,, 99(46): p. 1101-1125.

22. Babuska, I., et al., *The asymptotically optimal meshsize function for bi-p degree interpolation over rectangular element.* Journal of Computational and Applied Mathematics, 1998. 90: p. 185-221.

23. Stewart, J.R. and H.C. Edwards, *A framework approach for developing parallel adaptive multiphysics applications.* Finite Elements in Analysis and Design, July 2004. **40**(12): p. 1599-1617.

24. Grätsch, T. and K.-J. Bathe, *A posteriori error estimation techniques in practical finite element analysis* Computer and Structures, 2005. 83.

25. ADINA-AUI 8.5.2. 2008, ADINA R & D, Inc., http://www.adina.com.

26. Schoof, L.A. and V.R. Yarberry, *Exodus II :Finite Element Data Model.* SAND92-2137, 1995.

27. Malvern, L.E., *Introduction to the Mechanics of a Continuous Medium*1969, New Jersey: Prentice-Hall.

28. Harris, N., S. Benzley, and S. Owen. *Conformal Refinement of All Hexahedral Element Meshes Based on Multiple Twist Plane Insertion*. in *13th International Meshing Roundtable*. 2004.

29. Tchon, K.-F., C. Hirsch, and R. Schneiders, *Octree-Based Hexahedral Mesh Generator for Viscous Flow Simulations*, in *13th AIAA Computational Fluid Dynamics Conference*1997.

30. Schneiders, R., *Octree-Based Hexahedral Mesh Generation.* Int Journal Comput Geom Appl, 2000. 10(4): p. 383-398.

Appendix A. MESH REFINEMENT

There are several refinement techniques are prevalent in the literature. It is one of areas of research for meshing community. As it is desired to have high density nodes in the areas of interest while performing the finite element analysis, which can be achieved by refining the mesh. This appendix will discuss about the few hexahedral mesh refinement techniques available and will discuss in detail about the octree, element by element technique, and sheet refinement technique developed by Harris[28]. These schemes are selected because of their use in the refinement process in this thesis.

Octrees

This technique refines one hex into eight hexes[29]. This is done by spitting each edge at its midpoint. This method is performed until the desired size of the hex is obtained. This method provides local refinement of the hexes and control over the element size. One of the major limitations of this method is that it produces the non-conformal hexahedral mesh. Many finite element solvers cannot handle the non-conformal meshes. Figure A-1 shows the refinement performed on a cube using the octrees.

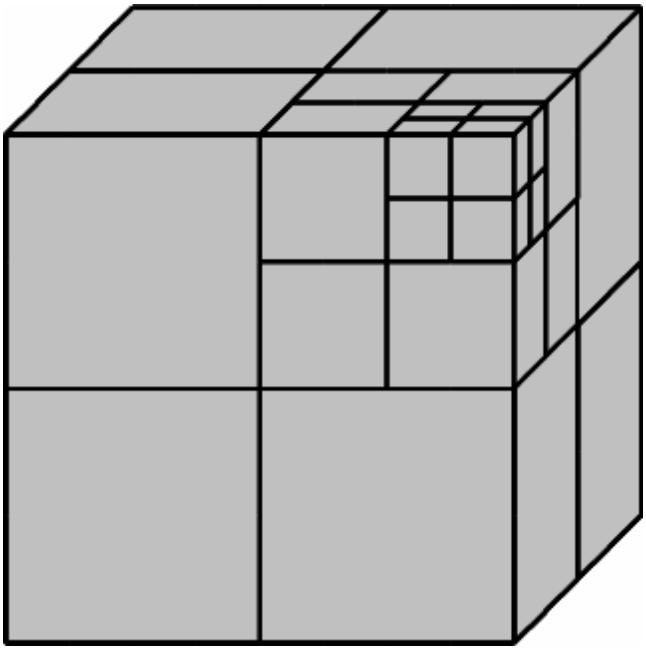

Figure A-1: Refinement using octrees

Element by Element Refinement

This is technique refines all three directions of hexahedron in one step[16]. This method inserts templates in the transition region to avoid the hanging nodes. Schneiders introduced an element by element refinement using the octree-based mesh generator. This method is efficient but cannot handle if the multiply connected transition hexes are present. As defined by Parrish, the multiply connected transition hexes are those hexes that are not selected for refinement but they share more than one faces with the hex that is selected for refinement. The templates to resolve this issue has not been developed yet. Figure A-2 shows the element by element refinement process by refining a single hex and adding the appropriate templates.

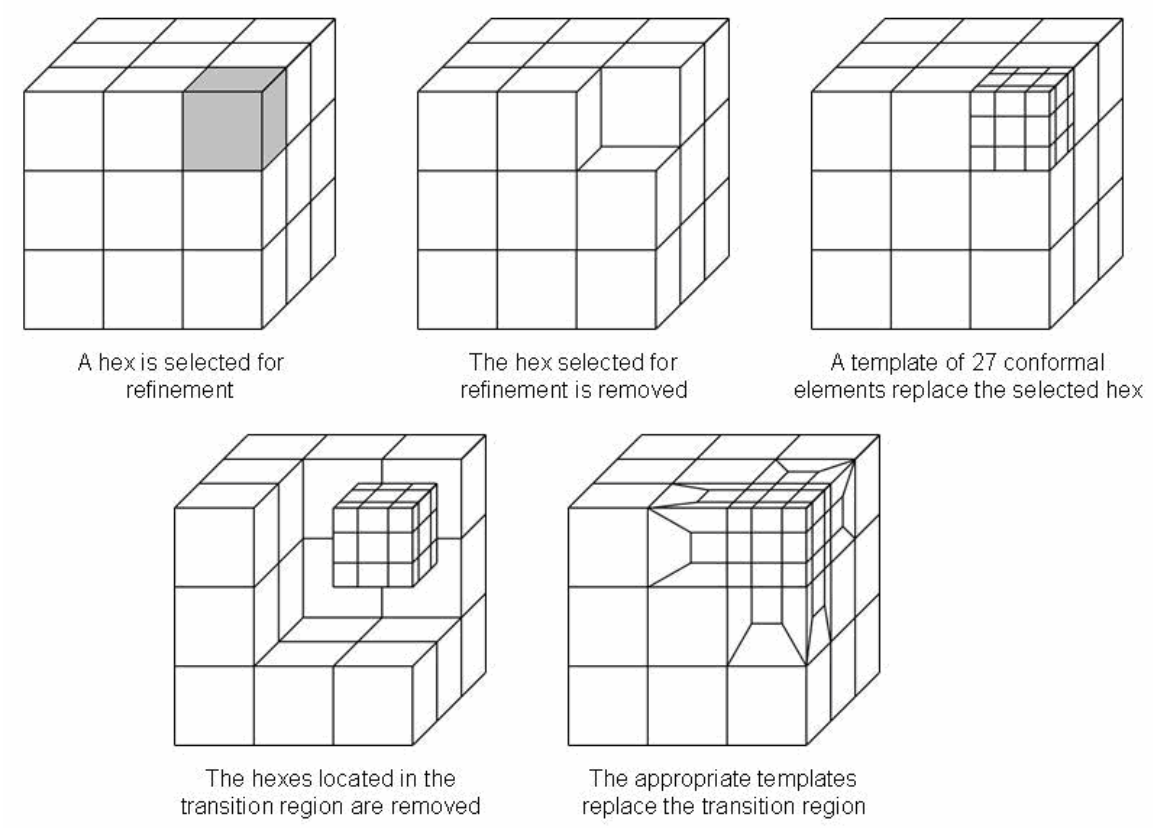

Figure A-2: Element by element refinement process

Sheet Refinement

This method is also developed by Schneiders[30], where it refines a mesh by inserting pillow in each direction of hex. Harris used the templates to refine the hex instead of inserting pillows. The technique developed by Harris is further generalized the refinement technique to include nodes, edges, and faces for hexahedral refinement. This method produces a conformal mesh, refines the hexes locally, and provides excellent user control of the refinement region[16]. This method has some limitations as well this method does not work well when there are self intersecting hexahedral sheets present and computation time is also another issue. Figure A-3 shows the steps of the sheet refinement technique.

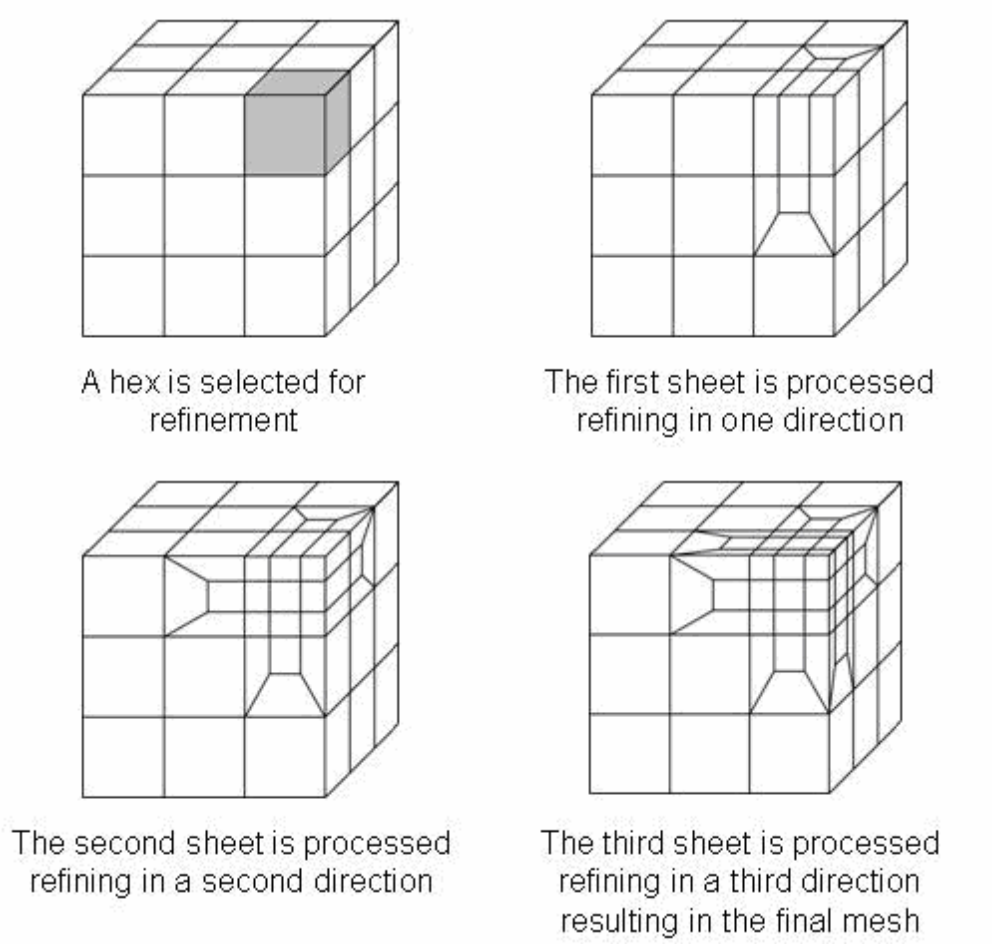

Figure A-3: Sheet refinement process

Selective Approach Algorithm

This method for refinement developed by Parrish is used in the refinement technique discussed in this thesis. As suggested by the name, this method is the implementation of the combination of element by element refinement and sheet refinement technique. As it uses the combinations of two methods it allows removing the limitation of both element by element and sheet refinement.

Appendix B. CREATING EXODUS II BASED SIZING FUNCTION

The manuals to create Exodus II[26] files are available on the internet. This appendix shows a simple example to create the Exodus II based sizing function which is used to drive the refinement process. As mentioned earlier Exodus II in the algorithm section Exodus II is a binary format, machine independent and random access. It is used for both the input and output of the finite element solution. The initial mesh and geometry is saved in 'qtr.e' file. Following is the piece of code that is used to create an Exdous File, to write the nodal variable (the error estimates), for the quarter piston problem shown in the example for this thesis.

```
void create_adapt_exodus_example_test()
{
   int i, j;
   int CPU_word_size,IO_word_size, exoid;
   int num_dim, num_nodes, num_elem, num_elem_blk, num_node_sets,        num_side_sets,
error;
   int id, num_nod_per_el, num_attr, num_el_in_blk, *connect;
   double *attrib;
   int *idelbs;
   char title[100];
   float version;
   double *x, *y, *z;


//Initialization of variables
   CPU_word_size = sizeof(double); /* float or double */
   IO_word_size = 8; /* use what is stored in file */

/*****************************************************************************/
/***                          Open exodus file                             ***/
/***                                                                       ***/
/*****************************************************************************/
```

```
   exoid = ex_open ("qtr.e", /* filename path */
   EX_WRITE, /* access mode = WRITE */
      &CPU_word_size, /* CPU word size */
   &IO_word_size, /* IO word size */
   &version); /* ExodusII library version */


//Get initial paramerters
   error = ex_get_init (exoid, title, &num_dim, &num_nodes, &num_elem, &num_elem_blk,
   &num_node_sets, &num_side_sets);


/*****************************************************************************/
/***                          Get nodal coordinates                        ***/
/***                                                                       ***/
/*****************************************************************************/

   x = (double *) calloc(num_nodes, sizeof(double));
   y = (double *) calloc(num_nodes, sizeof(double));
   if (num_dim >= 3)
      z = (double *) calloc(num_nodes, sizeof(double));
   else
      z = 0;

   error = ex_get_coord (exoid, x, y, z);



/*****************************************************************************/
/***                          Read block info                              ***/
/***                                                                       ***/
/*****************************************************************************/

   idelbs = (int *) calloc(num_elem_blk, sizeof(int));
   error = ex_get_elem_blk_ids (exoid, idelbs);
   char elem_type[MAX_STR_LENGTH+1];
/* read element block parameters */
   id = 1;
   error = ex_get_elem_block (exoid, id, elem_type,
   &num_el_in_blk, &num_nod_per_el, &num_attr);

/* read element connectivity */
   connect = (int *) calloc(num_nod_per_el*num_el_in_blk, sizeof(int));
```

```
   error = ex_get_elem_conn (exoid, id, connect);

/* read element block attributes */
   attrib = (double *) calloc (num_attr * num_el_in_blk, sizeof(double));
   error = ex_get_elem_attr (exoid, id, attrib)

/*******************************************************************************/
/***                          Write Variables                                ***/
/***                                                                         ***/
/*******************************************************************************/

        int num_nod_vars;
        int num_ele_vars;
   char *var_names[2];
/* write results variables parameters and names */
   num_nod_vars = 1;
        num_ele_vars=1;
   var_names[0] = "NSIZE";
        var_names[1]="ESIZE";
        char **var_ptr=var_names;
        error = ex_put_var_param (exoid, "n", num_nod_vars);
   error = ex_put_var_names (exoid, "n", num_nod_vars, var_ptr);

        error = ex_put_var_param (exoid, "e", num_ele_vars);
        error = ex_put_var_names (exoid, "e", num_ele_vars, ++var_ptr);


int *num_elem_in_block=new int[num_elem_blk], n,m,a;
num_elem_in_block[0] = num_elem;
double *n_variable=new double[num_nodes];
//load 1
n_variable    [    0    ]    =    3.48729    ;
n_variable    [    1    ]    =    5.85084    ;
n_variable    [    2    ]    =    2.86935    ;
n_variable    [    3    ]    =    2.28279    ;
n_variable    [    4    ]    =    3.74355    ;
n_variable    [    5    ]    =    6.00442    ;
n_variable    [    6    ]    =    2.77831    ;
n_variable    [    7    ]    =    2.59911    ;
n_variable    [    8    ]    =    1.85277    ;
n_variable    [    9    ]    =    1.70953    ;
n_variable    [    10   ]    =    2.8881     ;
n_variable    [    11   ]    =    2.90265    ;
n_variable    [    12   ]    =    3.9024     ;
n_variable    [    13   ]    =    4.02522    ;
n_variable    [    14   ]    =    4.29425    ;
```

```
n_variable [ 15 ] = 6.2681 ;
n_variable [ 16 ] = 3.46447 ;
n_variable [ 17 ] = 3.62444 ;
n_variable [ 18 ] = 3.94795 ;
n_variable [ 19 ] = 5.74165 ;
n_variable [ 20 ] = 3.9247 ;
n_variable [ 21 ] = 4.69704 ;
n_variable [ 22 ] = 3.43726 ;
n_variable [ 23 ] = 4.42158 ;
n_variable [ 24 ] = 6.19268 ;
n_variable [ 25 ] = 17.19273 ;
.
.
.
n_variable [ 1647 ] = 20.65976054 ;
n_variable [ 1648 ] = 11.27693909 ;
n_variable [ 1649 ] = 14.79968766 ;
n_variable [ 1650 ] = 3.95937012 ;
n_variable [ 1651 ] = 6.284903696 ;
n_variable [ 1652 ] = 0.0023 ;

printf("Number of blocks: %d\n", num_elem_blk);


error = ex_put_nodal_var (exoid, 1, 1, num_nodes, n_variable);

printf("%d\n",error);


/*****************************************************************/
/***                        Close file                         ***/
/***                                                           ***/
/*****************************************************************/

  error = ex_close (exoid);
}//END of function
```

Appendix C. CUBIT COMMANDS

The following illustrate the general procedure for refining hexahedral mesh using the exodus sizing function in CUBIT. Each line indicates commands that are entered into the CUBIT application, either by typing at command line, or using a text-based journal file. <*italics*> indicate required arguments to the commands

import mesh geom <*exodus II filename*>

import sizing function *"<exodus II filename>"* **block** <*block_ids*> **variable** *"<variable_name>"* **time** <*time*>

volume <*volume_ids*> **sizing function type exodus**

surface <*surface_ids*> **sizing function type exodus**

curve <*curve_ids*> **scheme stride**

adapt mesh vol <*volume_ids*> **min_error** <*minimum_error*> **sizing_function**

As an example, the following commands are used to refine the gear model in the Example Section of this Thesis:

import mesh geom 'gear.e'

import sizing function "gear.e" block 1 variable "NSIZE" time 0

volume all sizing function type exodus

surface all sizing function type exodus

curve all scheme stride

adapt mesh vol 1 min_error 8 sizing_function