5-7-2005

Solution Adaptive Isotropic And Anisotropic Mesh Refinement Using General Elements

Vinoad Senguttuvan

SOLUTION ADAPTIVE ISOTROPIC AND ANISOTROPIC MESH REFINEMENT USING GENERAL ELEMENTS

By

Vinoad Senguttuvan

A Thesis
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Computational Engineering
in the College of Engineering

Mississippi State, Mississippi

May 2005

SOLUTION ADAPTIVE ISOTROPIC AND ANISOTROPIC MESH REFINEMENT

USING GENERAL ELEMENTS

By

Vinoad Senguttuvan

Approved:

Dr. David Thompson
Associate Professor
Department of Aerospace Engineering
(Major Professor)

Dr. Edward Luke
Assistant Professor
Department of Computer Science and Engineering
(Committee Member)

Dr. Hyeona Lim
Assistant Professor
Department of Mathematics and Statistics
(Committee Member)

Dr. J. Mark Janus
Associate Professor
Department of Aerospace Engineering and Graduate Coordinator of Computational Engineering

Dr. Kirk Schulz
Dean
Bagley College of Engineering

Name: Vinoad Senguttuvan

Date of Degree: May 7, 2005

Institution: Mississippi State University

Major Field: Computational Engineering

Major Professor: Dr. David Thompson

Title of Study: SOLUTION ADAPTIVE ISOTROPIC AND ANISOTROPIC MESH REFINEMENT USING GENERAL ELEMENTS

Page in Study: 80

Candidate for Degree of Master of Science

Two refinement techniques to generate solution adaptive meshes have been developed. Both techniques utilize arbitrary polyhedra (general elements) to constrain the propagation of refinement. A face-based approach that produces isotropic refinement and a combined element- and edge-based approach that produces anisotropic refinement are presented. Refinement is triggered through sensors that use a shock detection algorithm or error estimation based on the smoothness of the reconstructed solution variables. The basic algorithms as well as specific implementation issues are presented. The advantages and disadvantages of the different methods are discussed and illustrated through a set of synthetic and realistic test cases. It is shown that general elements can be employed effectively in solution adaptive meshes generated using refinement.

ACKNOWLEDGMENTS

I would like to express my appreciation to my major professor, Dr. David Thompson for his valuable guidance, patience and support. I would like to thank Dr. Edward Luke for his help in strengthening my understanding and testing the validity and usefulness of my work. I would like to thank Dr. Hyeona Lim for her support and being on my committee. My most sincere thanks to Dr. Thompson and Dr. Luke for running and generating various test cases. I am extremely grateful to them for spending their valuable time on the test cases without which this thesis could never have been completed. Thanks to my friend, Dr. Satish Chalasani for working with me and guiding me at various times. Special thanks are also due to the ERC and all the people for providing a great environment and great facilities. The partial support of the National Science Foundation (EEC-0121807 and ITR/ACS-00859669), the National Aeronautics and Space Administration (NCC3-994), and the Army Research Office (DAA D19-00-1-0155) is also gratefully acknowledged.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

FIGURE Page

CHAPTER I

INTRODUCTION

1.1 Meshes

Meshes are generated to discretize the domains employed for computational simulations of physical phenomena and processes. Typically, a mesh is categorized according to its nodal connectivity.

A structured grid [1] consists of a set of coordinates with connectivity that naturally maps into elements of a matrix. Neighboring grid points in physical space are neighboring elements in the matrix. This by no means restricts structured grids to rectangular domains. Any grid that forms a regular lattice (even curvilinear), where neighboring points can be implicitly identified, can be described as a structured grid. Although the simplicity of a structured grid promotes accuracy and efficiency, the resulting topological constraints are too rigid for many complex geometries and it may be difficult to adapt in response to changing solutions. An alternative that is somewhat more flexible is the block-structured grid [1]. A block-structured grid consists of multiple blocks of logically rectangular structured grids that may fit together in an unstructured manner. Unfortunately, generation of block-structured grids for complex configurations is a labor-intensive process.

An unstructured mesh [1], on the other hand, has no such logical structure. It is defined by a set of points and explicit connectivity information. Since the mesh has no local structure, it is possible to add and delete nodes and elements locally. Hence, unstructured meshes have the flexibility to handle complex geometries as well as adapt to changing solutions.

Unfortunately, the memory and computation requirements are higher. Additionally for tetrahedral unstructured meshes, solution accuracy in viscous regions may be inadequate unless isotropic elements are employed [2]. Unfortunately, using isotropic elements near boundaries where flow gradients are much larger in the direction normal to the wall than in the direction parallel to the wall is prohibitively expensive.

Several different approaches have been developed to generate meshes with the advantages of both structured and unstructured topologies. A hybrid mesh [2] exploits features of both structured grids and unstructured meshes by placing anisotropic prismatic elements near boundaries where viscous effects are appreciable and filling the interior with unstructured isotropic elements. Typically, layers of prisms are placed in regions near viscous boundaries and tetrahedra are employed to discretize the interior of the domain. Hybrid meshes have higher memory and computational requirements when compared to structured grids and tetrahedral meshes but are more flexible than structured grids with regard to geometry and are more efficient than isotropic unstructured meshes for viscous flows.

A generalized mesh [5] extends the hybrid mesh concept a little further. A generalized mesh may contain arbitrary polyhedra. That is, an element may have an arbitrary number of faces and each face may have an arbitrary number of edges. The only requirement is that the elements should cover the entire domain without any overlaps or gaps. The flexibility of the general element structure makes an adaptive procedure like refinement easier to perform.

1.2 Solution Adaptive Meshing

The two main factors driving mesh generation for simulation are accuracy and efficiency. Clearly, we would like to get the best accuracy we can achieve using as few mesh points as possible. These seemingly conflicting objectives can be accomplished if we locally

increase the point density of the mesh in regions of the domain in which the error is large without enriching other regions of the domain. In general, since we do not know where this error will occur, there is a need to adapt the mesh as the solution develops.

The result of the adaptation process is that the mesh should be modified such that its characteristics tend to that of the optimal mesh. In this context, the optimal mesh is defined as a mesh that has the minimum spatial discretization error. Alternatively, the optimal mesh can be defined as the mesh in which the minimum number of degrees of freedom is required to achieve a specific level of accuracy. Therefore, a good adaptive meshing technique must improve solution accuracy while preserving mesh quality. It should introduce minimum additional errors and should be automatic and efficient.

In many cases, it is clear that the mesh should be clustered in certain regions because they are known to contain large gradients. The simplest example of this is the boundary layer that develops in the field adjacent to no-slip boundaries. This approach is called pre-adaptation [1] and it works well for steady solutions with well-known characteristics. However, if solution features or their locations are unknown or if the solution is unsteady, pre-adaptation may not be adequate. A technique that adapts the mesh based on the flow solution is required. The resulting meshes are called solution adaptive meshes. The two major steps in solution-based adaptation are estimating regions where the relative error in the solution is large and modifying the mesh based on these estimates.

1.3 Literature Review

McRae [6] lists the following as goals of solution adaptive meshing: (1) reduction of spatial discretization error and reduction of the dependence of the solution on the mesh, (2) quantifiable improvement in solution accuracy, (3) preservation of temporal accuracy and conservation (if required), (4) introduction of additional errors by the algorithm should not

reduce the benefits significantly, and (5) automation and efficiency of the algorithm. McRae further notes that conventional ideas on mesh quality sometimes do not hold for solution adaptive meshes. For example, when aligned properly, skewed high-aspect ratio elements can produce excellent results for shock transitions. Hence, he defines a quality mesh as, "Whatever mesh configuration is required to resolve the solution well locally, with no other restrictions."

The two main types of mesh adaptation are h-refinement, in which mesh nodes are added or deleted, and r-refinement (or redistribution), in which the number of mesh nodes remains constant but the nodes are relocated physically while maintaining mesh connectivity. Each approach has its advantages and disadvantages. For example, there are design optimization applications in which changes in mesh topology will adversely affect the computation of sensitivity derivatives [7]. Since the original mesh connectivity is conserved in r-refinement, it becomes a good choice for such applications. r-refined meshes have topological consistency with the original mesh and reduce mesh-induced errors in the solution. Also, the flow solution computed on the original mesh can be directly used on the adapted mesh as initial flow field without interpolations and other forms of averaging [7].

A popular technique in r-refinement is based on the elastic spring analogy in which element edges are taken to be linearly elastic springs with spring constants inversely proportional to their respective edge lengths. This technique has been extended to improve robustness by employing torsional spring elements [8]. A recent idea models the mesh as an elastic solid. In this approach, the equilibrium equations of the stress field are solved to obtain the deformations on the mesh [7].

A combination of r-refinement and h-refinement is often used in mesh adaptation. This provides an entire array of operations including node movement, mesh refinement, coarsening, and edge swapping [9]. Sometimes different operations are applied to different

regions of the domain such as the boundary layer, interior, etc. Refinement is triggered by various methods. The gradient of a scalar solution property, such as pressure, can be used. The Delaunay criterion or some other geometric constraint can also be utilized [10]. The Hessian matrix can be employed as an alternative method [10] as it identifies regions of curvature in the solution.

There are numerous refinement strategies that have appeared in the literature. In [11], elements in an unstructured tetrahedral mesh are subdivided to give two, four or eight tetrahedra. Since the resulting elements must remain tetrahedra, neighboring elements that contain a refined face must also be subdivided. The neighbor information of the base element, its neighbors, and the neighbors' neighbors must be modified to preserve the validity of the mesh. Hence, refinement of one element has an effect on approximately ten to fifteen elements. This operation increases the number of all the elements, faces, edges and nodes, which, in turn, increases the memory requirements and the run time of the flow solver. Although significant, this effect is local.

In a hybrid mesh [12], subdivision in the void-filling tetrahedral elements is the same as refinement in an unstructured tetrahedral mesh. Care must be taken in the prism layers, however. There are two possibilities when a prism is subdivided:

1. A prism is subdivided parallel to its triangular faces; neighbors that share the split rectangular faces must be subdivided to maintain element validity. All of the neighbor's neighbors must also be subdivided since some faces they contain have been refined. Thus, the effect propagates through the entire layer of prisms. All prisms in the layer must be refined thereby creating an extra layer of prisms.
2. A prism is subdivided by refining its triangular faces; all neighbors sharing a triangular face have to be subdivided in a similar manner to maintain element validity. This

propagates the subdivision across multiple layers and one full stack of prisms is subdivided.

We can see that, in the case of a hybrid mesh, refinement of one element may have affect an entire layer or entire stack of elements, i.e., the effect is not local. Also, additional refinement causes a significant increase in the number of all the mesh elements, faces, edges and nodes. That in turn increases the memory requirements and the run time of the flow solvers.

To summarize, the primary issue related to solution adaptive refinement of unstructured meshes is dealing with refinement of neighboring elements in response to refinement of a given element. Based on its shape, an element can be refined in many different ways. Some refinement strategies leave "hanging nodes" on some of their neighbors' edges. Unless general elements are permitted, the elements with the hanging nodes must then be refined in a suitable manner and so on, until all the hanging nodes are eliminated.

Refinement templates are one strategy to define how a given element is refined based on its hanging nodes [13]. Limiting the number of allowable templates eases programming and cost. Mavriplis [14] presents a technique that can be used for hybrid meshes containing a mixture of tetrahedra, pyramids, prisms, and hexahedra to produce a resulting mesh which contains the same set of element types (no new shapes are created) with no hanging nodes. This is achieved by defining a set of allowable subdivision types for each element and constructing a set of hierarchical subdivision rules. Biswas and Strawn [15] followed a similar approach by developing a set of allowable templates. However, if an insufficient number of templates is allowed, propagation of the refinement through the entire mesh may occur. Instead of using a self-similar refinement, Rivara [16] developed a successive bisection method for triangles, which minimizes propagation without limiting the number of patterns. Liu and Joe [17] extended it to tetrahedral meshes.

One approach to limit propagation of refinement through the mesh is to use buffer elements [18]. These buffer elements patch the interfaces between refined and unrefined elements. For example, a hexahedron sharing a face with a refined hexahedron will be subdivided into five pyramids while a hexahedron sharing an edge with a refined hexahedron will be subdivided into four tetrahedra. The buffer elements may exhibit poor quality and directional refinement characteristics unrelated to flow conditions. Also, as the surface area of an interface increases, the required number of buffer elements increases significantly.

Spragle, et al., [20] take a very different approach. Instead of patching the interfaces between refined and unrefined elements, the newly created hanging nodes are allowed to remain. In other words, they exist on one element but not on its neighbor. This requires maintaining a hierarchical data structure of faces and edges with parents storing pointers to children. That way, elements sharing edges (or faces), when refined, also share the bisecting nodes (and edges). Storage of obsolete elements, faces and edges facilitates efficient de-refinement. Since the adapted mesh contains hanging nodes (nodes), the flow solver must be able to treat the resulting elements. The face-based flux evaluation in the solver developed by Spragle is designed to work with the hanging nodes. Muller [21] described a similar element-node finite-volume method that can be used with hanging nodes. In the solver, faces with hanging nodes are divided into quadrilaterals or triangles based on the distribution of the hanging nodes. Similar approaches are employed in other flow solvers that have been developed to handle general elements [22-24].

Previously, a technique has been described to generate meshes via extrusion that locally employs general elements (arbitrary polyhedra) to improve mesh quality [3-5]. Similar techniques have been used to exploit the flexibility of general elements [18,19] for static meshes.

1.4 Motivation

It can certainly be argued that one of the most effective forms of solution adaptive meshing is Cartesian meshing [25]. There are two primary reasons: (1) the data structure facilitates efficient refinement and de-refinement and (2) the refinement rules contribute to the generation of high quality meshes, e.g., orthogonal, smoothly varying, etc. However, Cartesian meshes generally are refined isotropically and provide no mechanism to align faces with strong features. Additionally, Cartesian meshes are less than optimal for problems with viscous boundaries.

At least in part, the effectiveness of Cartesian meshes can be attributed to the fact that refinement produces what can be termed as general elements. For example, consider two adjacent quadrilaterals in a two-dimensional Cartesian mesh. One of the elements is selected for refinement and the other is not. Rather than subdividing the adjacent element, the "hanging" node effectively produces an element with five edges. This suggests a strategy whereby other element types may be subdivided in a similar manner resulting in general elements.

We have developed an isotropic face-based refinement algorithm and an edge-based anisotropic refinement algorithm. Our general philosophy is similar to Spragle, et al., [20] and Muller [21] in that we permit hanging nodes and maintain a hierarchical data structure of faces and edges. But, at the end of the adaptation process, the hanging nodes (edges and faces) are added to all the elements containing them and thereby producing general elements (elements and faces).

1.5 Thesis Statement

The objective of this thesis is to determine if general elements can be employed effectively in solution adaptive meshes generated using refinement. Generalized meshes of

various complexities are refined using solution adaptive methods and the ensuing meshes and their solutions are analyzed to verify the efficacy of this approach.

1.6 Thesis Outline

A strategy for mesh refinement that exploits general mesh elements, i.e., arbitrary polyhedra, is presented. First, three types of refinement are then discussed – an element-based centroidal insertion that produces pseudo-isotropic refinement, a face-based approach that results in true isotropic refinement, and an edge-based strategy that can be employed for anisotropic refinement. Then, details regarding the implementation of the algorithms are presented. Next error sensors, including a feature detection algorithm for stationary shock waves that can be employed for isotropic and anisotropic mesh refinement, are described Results are then presented that illustrate the strong and weak points of this approach. Conclusions are then drawn based on the numerical results.

CHAPTER II

REFINEMENT FOR GENERALIZED MESHES

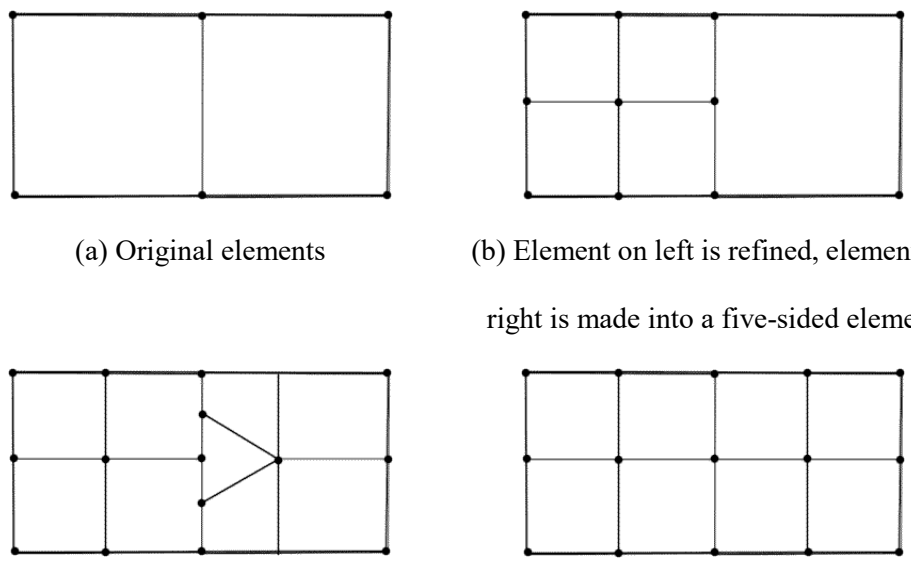
In this section, the procedures employed to refine a mesh are discussed. Isotropic refinement is a well-defined process regardless of the element type. Anisotropic refinement is more challenging and depends strongly on the element type. In some cases, such as when the mesh is aligned with a region of large gradients, anisotropic mesh refinement is sufficient. Additionally, since anisotropic directional subdivision results in a significantly fewer number of elements than isotropic refinement, it may be preferred for reasons of efficiency.

2.1 Advantages of Employing General Elements

In the context of solution adaptive meshing, the primary advantage of employing general elements is the ability to terminate mesh refinement efficiently without the addition of buffer elements. In Figure 1, the quadrilateral on the left is isotropically refined leaving a hanging node on the quadrilateral to the right. The element on the right is simply updated by adding the new node and it becomes a five-sided element. There is no propagation of any sort. Only the elements that require refinement are subdivided.

However, care must be taken while dealing with general elements to maintain mesh quality. For example, again consider a region of a two-dimensional mesh consisting of the two elements shown in Figure 1(a). As shown in Figure 1(b), the element on the left is selected for refinement and the element on the right was updated and converted into a five-sided element. Assuming this element is selected for refinement in the next cycle, it will be

subdivided as a five-sided element as shown in Figure 1(c). However, since that element was actually descended from a quadrilateral, it should be refined as shown in Figure 1(d) thereby creating a higher quality mesh than that shown in Figure 1(c).



(a) Original elements

(b) Element on left is refined, element on right is made into a five-sided element

(c) Right element refined as a five-sided element

(d) Right element refined as a four-sided element with a hanging node

Figure 1. Different ways to refine a generalized mesh

2.2 Element Refinement

In this section, the procedures employed to refine mesh elements are discussed.

2.2.1 Face-based Subdivision: Isotropic Refinement

The isotropic refinement method employed here is termed face based. Each *n*-sided face is subdivided into *n* faces by inserting a new node at the centroid of the face and connecting it to the midpoint of each edge. These newly created edges form faces that determine the refinement in the interior of the element. A node is inserted at the centroid of

the element and new faces constructed to subdivide the element thereby maintaining element quality. The disadvantage is that the newly created nodes and edges increase the size of the mesh quite remarkably. Since all the faces on the element are refined, this routine cannot be used along with the anisotropic edge based refinement described in a later section. Results for some standard polyhedra are shown in Figure 2. Since this approach is well-defined procedure that is independent of the element type, we do not consider it a template.

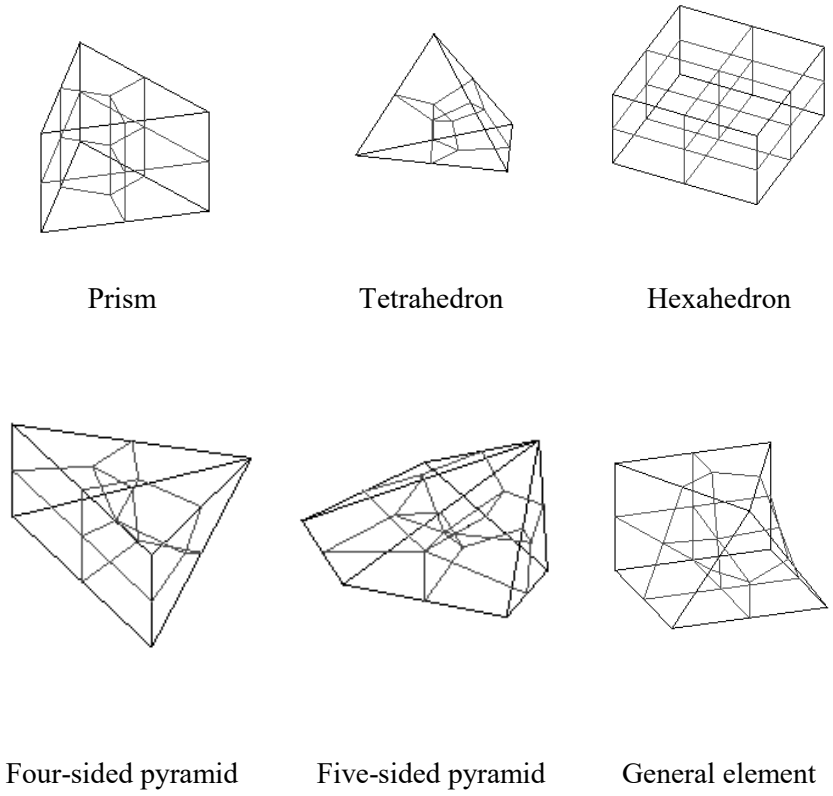


Figure 2. Face based isotropic refinement

2.2.2 Element-based Subdivision: Refinement by Centroidal Insertion

As an alternative to face-based refinement, any polyhedron can be refined by inserting a node at its centroid and introducing new edges by connecting the inserted node to the nodes in the existing element as shown in Figure 3. For an existing element with *nf* faces, *nf* new elements are created and the old element is deleted. A tetrahedron results if the inserted node is connected to a triangular face. If the inserted node is connected to a quadrilateral face, a pyramid is formed. Similarly, if the inserted node is connected to face that is an *n*-gon, a pyramid with an *n*-gon base is formed. The existing element faces remain unchanged. The obvious disadvantage is that, since no new nodes are introduced on existing edges, the element quality degrades after multiple refinement cycles. Additionally, since the refinement is effectively isotropic, it does not represent an efficient approach for highly anisotropic features such as shock waves and shear layers. Centroidal insertion has the advantage of simplicity and it creates a minimum increase in the number of mesh elements. Results for several polyhedra are shown in Figure 3. As with the face-based isotropic refinement, this approach is not considered to be template based because its application is independent of element type.

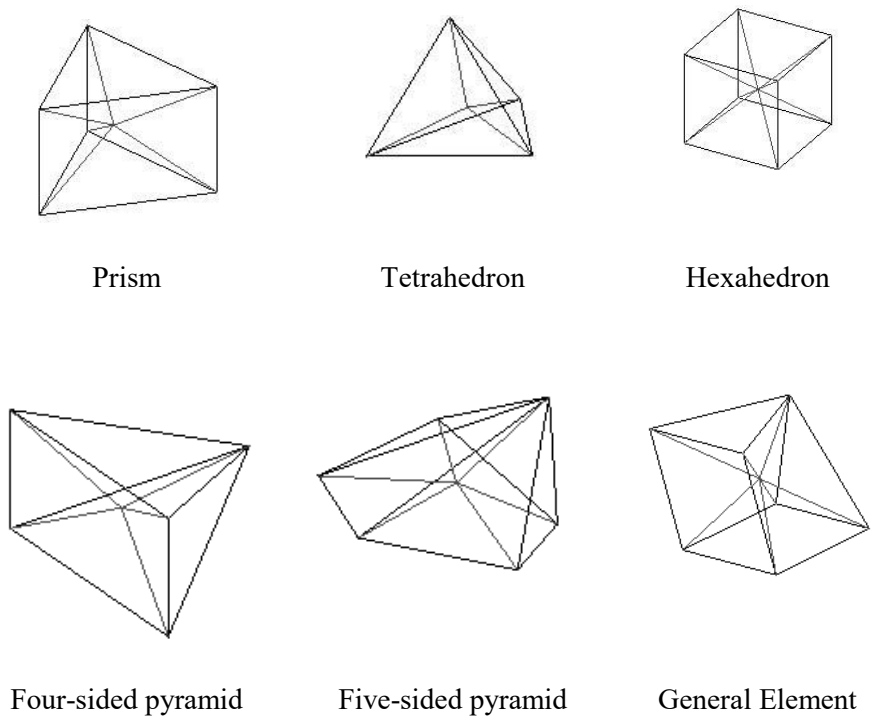


Figure 3. Cell based isotropic refinement

2.2.3 Edge-based Subdivision: Anisotropic Refinement

We now describe template-based, anisotropic refinement strategies for tetrahedra, triangular prisms, hexahedra, and arbitrary pyramids. These strategies are called edge-based because it is the element type and the division of the edges that determines how an element is subdivided. The elements listed below are refined anisotropically based on the following set of templates:

Triangular prism: A triangular prism is defined as an element with three quadrilateral faces and two triangular faces. The triangular faces have no edges in common. This element has six nodes, nine edges, and five faces. There are two possible cases for subdivision:

1. If the chosen edge is on a triangular face, the slicing plane will cut through an edge on the other triangular face. That edge is found and chosen for refinement. The two selected edges are refined by inserting nodes at their midpoints. The two triangular faces are divided by connecting the opposite node to the new node. For the rectangular face, connecting the two new nodes subdivides it and creates a new edge. The new edges created form a new face, which subdivides the element into two prisms.
2. If, however, the chosen edge is not on a triangular face, the slicing plane will cut through all the three axial edges (non-triangular). These three edges are divided by inserting nodes at their midpoints. For each rectangular face, connecting the two new nodes subdivides it and creates a new edge. The new edges created form a new face, which subdivides the element into two prisms.

Tetrahedron: A tetrahedron is defined as an element with four triangular faces, four nodes, and six edges. The selected edge is refined by inserting a node at its midpoint. Each of the two triangular faces that contain the edge is divided into two triangular faces by connecting the opposite node to the new node. The new edges created form a new face, which subdivides the element into two tetrahedra.

Hexahedron: A hexahedron is defined as an element with six quadrilateral faces, eight nodes, and twelve edges. For any selected edge, a cutting plane slices through four parallel edges. Each of the four selected edges is refined by inserting a node at its midpoints. For each of the four rectangular faces containing the chosen edges, connecting the two new nodes subdivides it and creates a new edge. The new edges created form a new face, which subdivides the element into two hexahedra.

Arbitrary Pyramid: An arbitrary pyramid has an n-gonal face as its base and n triangular faces connecting the n edges on the base to a node at the apex of the pyramid. The

number of nodes is equal to the number of faces. Anisotropic subdivision is performed only if the chosen edge is on the base of the pyramid. The chosen edge is divided by inserting a node at its midpoint. The triangular face that contains the edge is divided into two triangular faces by connecting the opposite node to the new node. For the base, there are two possibilities:

1. If the base has an odd number of nodes, the new node of the refined edge is connected to its opposing node thereby subdividing the base.
2. If the base has an even number of nodes. Then there is no unique node opposite to the refined edge. Instead, there is a unique opposite edge. The opposing edge is then divided by inserting a node at its midpoint. The triangular face that contains this edge is divided into two triangular faces by connecting the opposite node to the new node. The new nodes of the two subdivided edges are joined to create a new edge and subdivide the base face into two.

The new edges created form a new face, which subdivides the element. The new elements could either be tetrahedra or arbitrary pyramids based on the number of nodes in the base.

If an arbitrary element is encountered, it is subdivided using centroidal insertion. Since centroidal insertion yields only tetrahedra and arbitrary pyramids, subdivision of arbitrary elements yields elements with defined anisotropic refinement templates. Hence, the number of arbitrary elements in the mesh is always within acceptable limits. This reduces the use of centroidal insertion, which in turn maintains element quality, and decreases the number elements created. Results for a few standard polyhedra are shown in Figure 4.

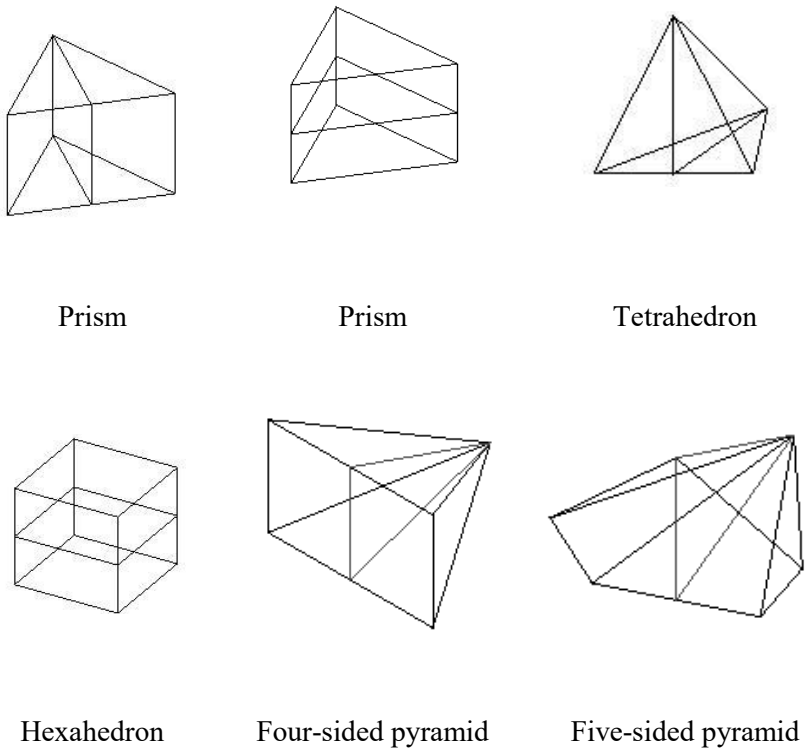


Figure 4. Face based anisotropic refinement

We now illustrate a potential problem with the template-based approach. Consider two neighboring prisms in the mesh. If one of them is axially subdivided and the other is updated after the refinement, we get the elements illustrated in Figure 5. The first prism has been subdivided into two prisms and the other prism is updated with the new faces and once it is updated, it is no longer a prism. Its type is checked and since it doesn't match any of the classified shapes, it is marked as an arbitrary element with six faces. If this element is marked for refinement but assigned a lower priority, it will be subdivided by centroidal insertion. The result is illustrated in the Figure 6.

Although the generated mesh is perfectly valid, it is not desirable. Since one prism is subdivided axially and the other centroidally, there are now adjacent elements of different types. Further, the uniform orientation that existed is lost and more elements than needed are generated.

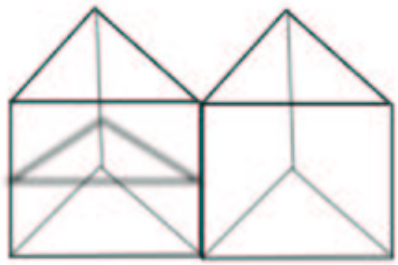


Figure 5. A prism refined axially

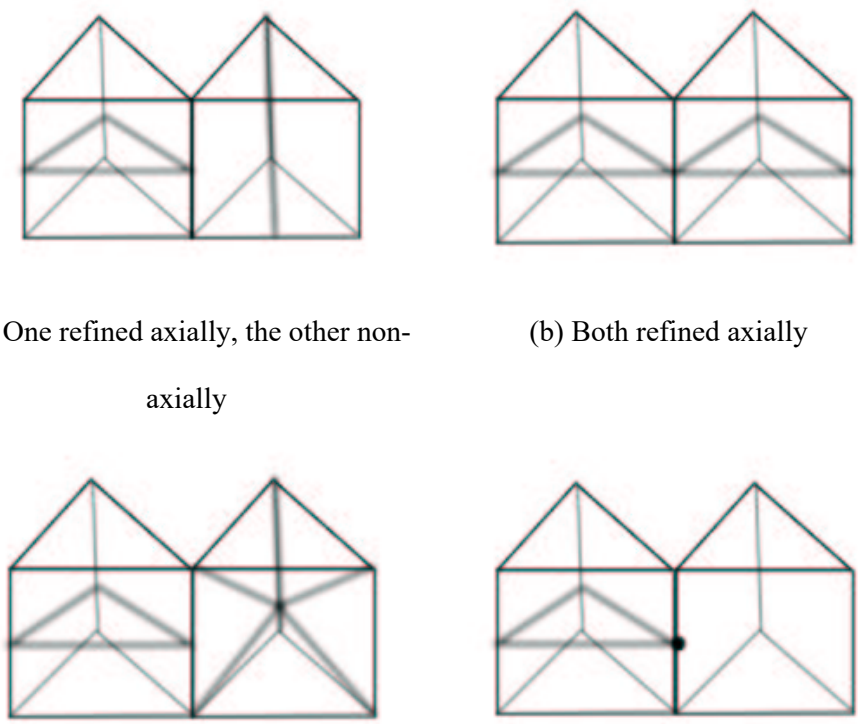
.

Figure 6. A prism refined axially with a neighbor refined centroidally

To avoid this situation, the following approach is employed. When the first prism is subdivided, the second prism is not updated with the new faces. The old face is marked as old and the two new faces created are stored in an old face. If, at a later point, the other prism needs to be subdivided, the following possibilities are encountered:

1. The required subdivision would not refine the old face.
2. The required subdivision would refine the old face in the same manner.
3. The required subdivision would refine the old face in a different manner.

In the first and second cases, the other prism is subdivided in the required manner. In the third case, the required subdivision is not possible and centroidal insertion is used. Once all of the required subdivisions are performed, the elements are checked to see if they contain an old face. Each old face is replaced by two new faces that were created when the old face was refined. Thus, mesh connectivity is maintained and the most optimal mesh is generated, at least in the context of the minimum number of elements, more uniform element size, and more uniform face orientation. The different possibilities and the resulting refinement strategies are illustrated in the Figure 7.



(a) One refined axially, the other non-axially

(b) Both refined axially

(c) One refined axially, the other centroidally

(d) One refined axially, the other updated

Figure 7. Ways of refining two neighboring prisms

2.3 Relative Efficiency of the Refinement Methods

One obvious advantage of anisotropic refinement is that the number of elements created by refinement is significantly reduced in comparison to isotropic refinement. An estimate for the number elements after n adaptive refinement cycles, N^n, can be obtained from:

$$N^n = N^0\left((1-\alpha)+\alpha\beta^n E_f^n\right) \qquad \text{(Eq. 1)}$$

where N^0 is the number of elements in the baseline mesh, is the fraction of elements in the baseline mesh that are selected for refinement, is the fraction of elements that are selected for refinement in subsequent refinement cycles, and E_f is the element refinement factor, i.e., the number of elements that are produced by refinement of a given element. For the face based isotropic refinement presented, E_f is equal to the number of nodes in the element. Table 1 shows values of the refinement factor for selected element types. The estimate given in Eq. 1 is based on the assumption that a feature becomes better defined as the mesh is refined so that the physical region marked for refinement decreases in extent in subsequent refinement cycles. It should be noted that may, in fact, be different for different element types. Additionally, although this estimate could be made more accurate by accounting for the fractions of the different element types present in the baseline mesh, this modification is not considered significant since isotropic refinement tends to produce hexahedral elements.

Eq. 1 can be rewritten as

$$\frac{N^n - N^0}{\alpha N^0} = \beta^n E_f^n - 1 \ . \qquad \text{(Eq. 2)}$$

Thus, we see that the increase in the number of elements relative to the original number of elements marked for refinement increases as the product of the element fraction and element refinement factor to the power n. As expected, we can conclude that isotropic refinement for any element type is significantly more expensive than anisotropic refinement. This equation

also demonstrates that unconstrained isotropic refinement is prohibitively expensive in most situations. Therefore, some type of constraints must be imposed on the isotropic refinement algorithm if it is to be useful for realistic problems.

Table 1. Element refinement factors for selected element types

Element type	E_f(isotropic refinement)	E_f(anisotropic refinement)
Hexahedron	8	2
Triangular prism	6	2
Tetrahedron	4	2
Four-sided pyramid	5	2
Five-sided pyramid	6	2

CHAPTER III

REFINEMENT ALGORITHMS

In the previous chapter, three element subdivision strategies were discussed. In this chapter, the isotropic and anisotropic algorithms are discussed in detail.

3.1 Isotropic Refinement

Isotropic refinement is achieved using a face-based strategy. In a face-based refinement, it is the refinement of the faces that determines how elements are refined. An *n-sided* face is subdivided into *n* faces by inserting a new node at the centroid of the face and connecting it to the midpoint of each edge. These newly created edges form faces that divide the interior of the element. This approach can be thought of as being analogous to refinement for a Cartesian mesh.

3.1.1 Implementation Strategy

For regular elements such as tetrahedra, hexahedra, pyramids, and prisms, this approach is relatively simple to implement. A detailed step-by-step procedure is described below for refinement of a general element:

1. A pre-defined criterion is used to determine if an element should be subdivided. For example, if more than half the nodes of an element are tagged, the element is marked for subdivision. An element is also marked for subdivision if a neighboring element

has two or more additional levels of subdivision. The following steps are done for all the elements marked for subdivision.

2. Each edge in the element is subdivided by inserting a node at its midpoint
3. For each *n*-sided face in the element,
    a. A new node is created at its centroid.
    b. The node at the midpoint of each edge on the face is connected to the node at the centroid to form *n* new edges.
    c. Each node on the face lies exactly on two edges and it forms a four-sided face with the midpoints of those two edges and the centroid of the face. *n* four sided faces are formed.
4. For the *n*-faced element with *e* edges and *v* nodes,
    a. A new node is created at its centroid.
    b. The node at the centroid of each face on the element is connected to the new node to form *n* new edges.
    c. Each edge on the element lies exactly on two faces and the node at the midpoint of the edge forms a four-sided face with the centroid of the two faces and the centroid of the element. *e* four sided faces are formed.
    d. For each node, an element is formed. Faces on the surface containing this node are added to the element. Then, all the newly formed faces are checked. If three of the four nodes of a face (the forth is the centroid of the element) are part of the formed element, the face is added to the element. This creates *v* elements.
5. The desired number of refinement cycles is then performed on the mesh. Finally, if any face contains a refined edge, that edge is removed and is replaced by the subdivided and newly formed edges. Similarly, if any element contains a refined

face, that face is removed and is replaced by the subdivided and newly formed faces. This updates the connectivity of the mesh.

This approach has the advantages and disadvantages of a Cartesian mesh refinement. While the isotropic refinement maintains element quality, the newly created nodes and edges can increase the size of the mesh quite remarkably.

3.1.2 Selecting an Element for Isotropic Refinement

During the feature detection/error estimation process, nodes are tagged for refinement. For face-based isotropic refinement, a procedure is needed to determine whether or not a given element is refined based on the tagged nodes. The following criteria were investigated:

1) If half or more nodes of an element are tagged, the element is marked.

2) If all the nodes in a face of the element are marked, the element is marked.

3) If three non-planar (they could be mutually planar) nodes are tagged.

4) If two nodes from different faces are tagged.

5) An edge is marked if both its nodes are tagged. A face is marked if more than half of its edges are marked and an element is marked if more than half it's faces are marked.

The criteria are increasingly stringent. A combination of the first and second criterion appeared to work best for sharp features such as shock waves.

3.2 Anisotropic Refinement

There are many ways of performing anisotropic refinement. First an element could be subdivided into two or more elements. Even subdividing an element into two children can be

done in different ways. If we take a triangular face for example, it can be refined by dividing an edge and the angle opposite to it or by subdividing two edges.

Here, we use an edge-based approach. First, the edge on an element that most needs subdivision is identified. That edge is subdivided and the element is refined based on the edge refinement. Hence, in a triangle, the angle opposite to the selected edge will be refined. In a quadrilateral, the edge opposite to the chosen edge will be refined. This extends to other polygonal shapes and to tetrahedra, prisms, and hexahedrons in three dimensions.

The advantages of this method are that it is versatile and that it facilitates a wide variety of refinement strategies for elements aligned with flow features. Though each refinement cycle results in a single element subdivision, multiple refinement cycles can be employed to achieve a wide variety of refinement strategies, including pseudo-isotropic refinement. The disadvantage of this method is that when odd-edged faces like triangles and pentagons exist in the element, some angles will be subdivided decreasing the quality of the elements.

3.2.1 Template-based Implementation Strategy

Anisotropic refinement is performed using an edge-based refinement scheme. The template for subdivision for each type of element (tetrahedron, prism, hexahedron, pyramids) is described. If another type of element is encountered, centroidal insertion, as described in Chapter 2, is employed.

1. In addition to tagging nodes, the feature detection process must provide a local direction vector for each node that indicates the direction that refinement is needed. The idea is to refine the element along the edges that are most parallel to the direction vector. The direction vector for each edge is calculated by averaging the direction vectors specified at its two nodes.

2. For each edge that has been previously identified as potentially needing refinement (both nodes are marked), the dot product of the edge vector and the local refinement direction is taken. The absolute value of the dot product is stored as the orientation value. Larger values indicate that the edge is more parallel to the refinement direction.
3. A pre-defined criterion is used to determine if an element should be subdivided. For example, if more than half the nodes of an element are tagged, the element is marked for subdivision. An element is also marked for subdivision if a neighboring element has two or more additional levels of subdivision. The following steps are performed for all the elements marked for subdivision. The subdivision criteria are discussed in more detail in a later section.
4. In an element marked for subdivision, the dot product of the geometric vector and the gradient is calculated for each edge. The larger the value of the dot product, the more aligned the edge is to the gradient.
5. The edges are sorted from most aligned to least (largest dot product to smallest).
6. The sorted edge list is then traversed. To divide the current edge, the two faces (on the element) that contain the edge, must be divided. There are three possible cases:
    a. One of the two faces has already been divided as a consequence of the refinement of another element. If the division is across the same edge, the element is divided in the same manner.
    b. One of the two faces has already been divided as part of the refinement of another element. If the division is not across the same edge, the element is not divided and the next edge in the list is considered.
    c. Neither face has been divided. The face and element are subdivided.

7. Subdividing an element into regular shapes may require the division of more than one edge. In that case, the other edges are considered along with the faces that contain them. An evaluation of the candidate faces is performed to determine if they can be refined in the required manner. If so, the original edge is subdivided.
8. If an edge is selected, refine the element using the appropriate template. If no edge can be selected, use centroidal insertion.

3.2.2 Refinement Directions for Anisotropic Subdivision

In addition to tagging nodes, the feature detection process stores a gradient at each node. For anisotropic refinement, the idea is to refine the element along edges that are parallel to direction of the gradient. The gradient for each edge is calculated by averaging the gradient at its two nodes. The elements are marked for refinement based on the criteria explained earlier. For each edge in the element, the dot product of its geometric edge vector and the selected gradient vector is calculated. The larger the dot product, the more aligned the edge is with the gradient. The edges are sorted from most aligned to least aligned (largest dot product to smallest). Based on this sorted edge list, the element is refined anisotropically.

3.2.3 Achieving Psuedo-isotropic Refinement from Anisotropic Subdivision

In many instances, the gradients are not aligned with element edges. Take for example the case of a mesh with hexahedral elements in the presence of a gradient inclined at 45 degrees with respect to the element faces. Due to minute numerical variations, there will be random horizontal and vertical refinements, which is highly undesirable. Such regions of the mesh should be typically identified and isotropically refined. Unfortunately, using the isotropic and anisotropic refinement strategies described here in a seamless manner requires

additional programming that is beyond the scope of this effort. This idea is discussed in detail in the future work section.

Some uniformity and a pseudo-isotropic refinement can be achieved with the following idea. While sorting the edges of the elements marked for refinement, instead of using only the dot product (of geometric vector and gradient), a multiple check is applied. A comparison of two edges is made based on the first parameter and, if the variation is less than a threshold, the next parameter is used and so on. The following is the order of parameters used:

1) Dot product (of geometric edge vector and gradient vector)

2) Length of the edge

3) Length of projection onto z-axis (dot product of geometric vector and z-axis)

4) Length of projection onto y-axis (dot product of geometric vector and y-axis)

When this method of sorting is used on the case of the hexahedral square mesh with a 45-degree gradient, the following happens. In the first iteration, all the edges have same dot product and size but the horizontal edges get refined because they have projection lengths on y-axis. In the second iteration, since the horizontal edges are refined, their size is smaller and hence the vertical edges get refined. After two iterations, the resulting mesh is the same as the mesh after one level of isotropic refinement. In three dimensions, it takes three iterations.

CHAPTER IV

IMPLEMENTATION ISSUES

In this chapter we discuss practical details that are related to implementation of the algorithms. Lists of classes and functions are included in the appendix.

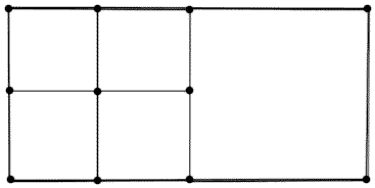
4.1 Data Structure

The characteristic information discussed in the previous chapters is stored in the data structure. We have lists of nodes, edges, faces and elements. A hierarchical data structure is built on these lists. An element contains a set a faces, a face contains a set of edges and an edge contains two nodes.

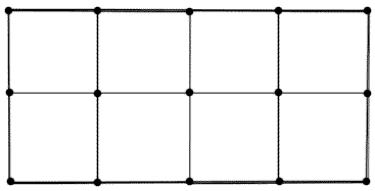
A face, when subdivided, stores pointers to its children. Hence, elements need not be updated after each refinement cycle. To achieve the results shown in Figure 8, the element on the right is maintained as a four-sided element with one of its faces refined. If this element needs to be refined at a later refinement cycle, all unrefined faces are subdivided while refined faces are queried for their children.

Once a refinement cycle is complete, all elements are updated by replacing the refined faces they contain with the children of those faces. These children might themselves be refined. In that case, they are replaced by their children and so on until all faces contained by the elements are tagged as unrefined. This way, the flexibility of general elements is utilized without loss of mesh quality. The process was discussed in two dimensions for ease

of understanding. The same procedure works for three dimensions. In addition to the element-face pair, the face-edge pair is handled in the same manner.



(a) Element on left is refined, element on right is made five-sided



(b) Right element refined as a four-sided element with a hanging node

Figure 8. Refinement of general element with a hanging node

4.2 Mesh State File

Instead of running multiple refinements using the same flow field data, it may be preferable to run one (or two) refinement cycles, execute the flow solver, and then return to perform refinement with a presumably more accurate solution. However, if the mesh file is input to the refinement software, the only information we have are the nodal positions and the mesh connectivity. The refinement levels, parent details, etc., that were contained in the data structure are lost. To overcome this shortcoming, a mesh state file is also written out in addition to the mesh file. The mesh state file has all the information that is contained in the data structure. After the solver is executed, the new solution can be read in along with the state file for further adaptation. This way, the quality is maintained over multiple refinement cycles. The format of the mesh state file is described in the appendix.

4.3 Folded Faces

In three dimensions, a face is called folded if it is non-planar. If an element contains one or more folded faces, a node on the element might not be visible to another node on the element. Some solvers will not run if there are such elements [23]

When there are folded faces in the input mesh, care needs to be taken during refinement to ensure that no invalid elements are created. We use two checks for this. A face is marked as folded if the angle between the normals of two triangles on it is larger than a specified threshold. Since, in many cases a folded face has small edge length, a threshold on the minimum length edge that can be refined eliminates some problems. Additionally, we can impose the constraint that the elements on either side of a folded face must both be refined (or not).

4.4 Balancing Levels of Refinement between Elements

When the sizes of neighboring elements vary greatly, the accuracy of the flow solver decreases. Hence, as elements are refined, their levels of refinement are stored. If an element's neighbor has two levels of refinement more than the element, the element is marked for refinement. In this context, neighbors are defined as two elements that share one or more nodes.

CHAPTER V

TRIGGERING REFINEMENT

The first step of the solution adaptive refinement algorithm is to tag the nodes according to some criteria algorithm. This is essentially a binary process although additional information such as the preferred direction of the feature may also be stored for anisotropic mesh refinement. In the section, we describe to alternative approaches. One approach, which relies on detecting features in the flow field, is based on the premise that features in the flow field need to be resolved adequately to reduce error. The premise of the second approach, refinement based on the smoothness of the solution, is that error occurs in regions where the solution is not smooth. Both approaches will be discussed along with implementation strategies.

5.1 Feature-based Refinement – Shocks

We now describe a feature detection algorithm for stationary shock waves. A shock is characterized by abrupt changes in flow quantities such as pressure, velocity, and density. A physical shock is nearly a singularity – the changes occur over a distance equal to a few mean free paths of the fluid molecules. In the case of numerical data, however, the discontinuity is typically smeared over several elements due to the errors inherent in the discrete approximation. The properties of shocks are explained in more detail in any gas dynamics text [26]. The algorithm employed here is based on the algorithm presented in [27].

The key quantity in a shock detection algorithm is the Mach number (ratio of the velocity magnitude to the local sound speed) normal to the shock wave. For a stationary shock, the normal Mach number changes from greater than unity to less than unity in the direction of the flow. Further, near a shock, the pressure gradient is aligned along the local normal to the shock front. Therefore, the scalar product of the normalized pressure gradient and velocity vector can be used to find the normal Mach number $\boldsymbol{M_n}$ (after division by the local acoustic speed a) as given by

$$M_n = \frac{\nabla p}{|\nabla p|} \cdot \frac{\mathbf{V}}{a} \quad . \tag{1}$$

This scalar product can also be used to identify compression and expansion. If the product is greater than zero, the node is located in a region of compression and if it is less than zero, it is located in an expansion region.

If a node has a normal Mach number greater than one and has a neighbor in the direction of the flow with a normal Mach number less than one, then these two nodes are tagged as part of the shock.

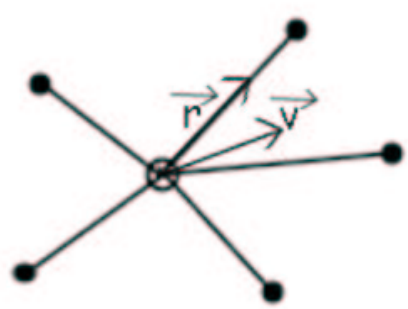


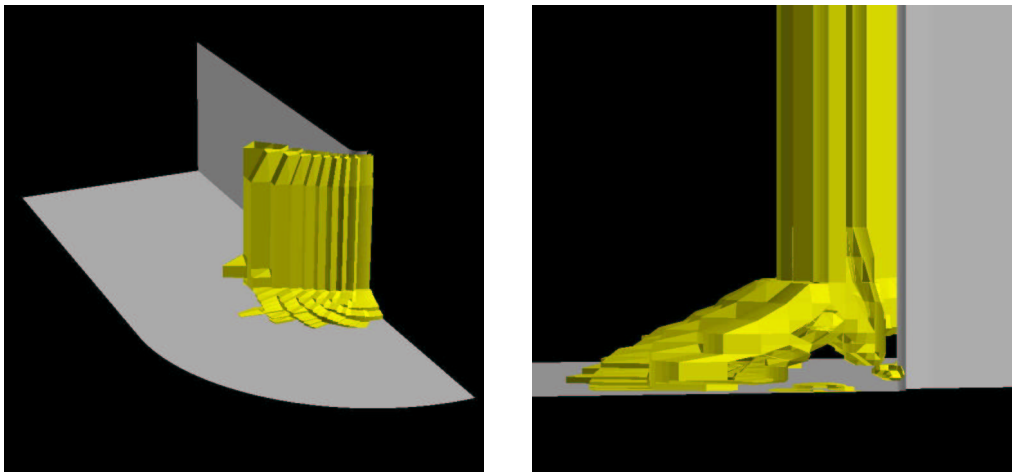
Figure 9. Velocity and position vectors

In the Figure 9, $\boldsymbol{v}$ is the local velocity vector at a node and $\boldsymbol{r}$ is the position vector from the node to one of its neighbors. If the scalar product of $\boldsymbol{v}$ and $\boldsymbol{r}$ is greater than zero, the

neighbor is in the local flow direction. And in addition to that, if the node has a normal Mach number greater than one and the neighbor has normal Mach number less than one, then both these nodes are tagged as a part of the feature (shock). Of course, a correction term must be included if the shock is moving [27].

For the purpose of anisotropic adaptation, the local pressure gradient vector can be saved to provide a preferential direction for refinement. Care must be taken not to divide by zero in regions where the pressure gradient is zero. Small disturbances in the pressure gradient may be caused by numerically induced Mach waves. To avoid detection of these disturbances as a part of a shock, a filter is applied. If the pressure jump between a node and the neighbor node in the direction of flow, the node is discarded.

As an example of our shock detection technique, we show two images for the standard blunt fin/flat plate flow field solution in Figure 10. Figure 10(a) shows a detached shock that wraps around the blunt fin. Figure 10(b) shows the -shock that intersects the symmetry plane. It should be noted that the mesh used here is relatively coarse which contributes to the rapid dissipation of the shock.



(a) Detached shock (b) -shock

Figure 10. Shock detector applied to blunt fin/flat plate flow field

5.2 Error Estimation – Jumps in the Solution

The objective of adaptation is to reduce the variation of error in the solution. An alternative approach to the feature-based approach described above is to estimate the error in the solution based on the jumps in the solution across the element faces. These jumps indicate regions in the domain where the solution is non-smooth. Consider a piecewise linear reconstruction of some flow field variable. In general, the reconstruction is discontinuous at cell boundaries. However, if the variation in the flow field parameter is smooth, i.e., the curvature in the solution is small, the jumps in the reconstructed values at the cell faces will be small. In this approach, the error at each face is taken to be the magnitude of the jump in reconstructed value between the cells on either side of the face. Each node is assigned the maximum error among its surrounding faces.

We identify large variations in error by evaluating the mean error for the problem and finding the nodes that are at least one standard deviation above the mean. These nodes are

considered to contribute significantly to error variation and are marked for refinement. If one refinement does not sufficiently reduce the error, they will be marked in subsequent refinement cycles. The advantage of this approach is that it provides an automatic scaling of error for any particular problem and is effective at using refinement to reduce locally high error concentrations.

CHAPTER VI

DISCUSSION AND RESULTS

The face-based isotropic and edge-based anisotropic refinement techniques described previously were employed in solution-adaptive mesh refinement algorithms and their effectiveness evaluated by application to several test problems. First, as a proof of principle and a demonstration of the methodology, some simple meshes are refined based on synthetic trigger functions. Solution adaptive, face-based isotropic mesh refinement was then performed for the following flow problems: supersonic inviscid flow over a double ramp, supersonic inviscid flow past a notional reentry vehicle, low-speed viscous flow with hydrogen/oxygen combustion in a combustion chamber with a single element injector, and transonic viscous flow over an M6 wing. Anisotropic refinement was performed on for hypersonic flow of a real gas over a sphere and transonic viscous flow over an M6 wing. In some cases, the simulated results are compared to experimental data. The effectiveness of the refinement techniques and their limitations are discussed.

6.1 Proof of Concept and Code Validation

To illustrate the properties of the different mesh refinement strategies, as well as provide a proof of concept, several simple meshes are refined using various synthetic trigger functions.

6.1.1 Cube with Hexahedral Mesh: Isotropic Refinement

We first consider a regular 20 x 20 x 20 hexahedral mesh as shown in Figure 11(a). A trigger function was defined using

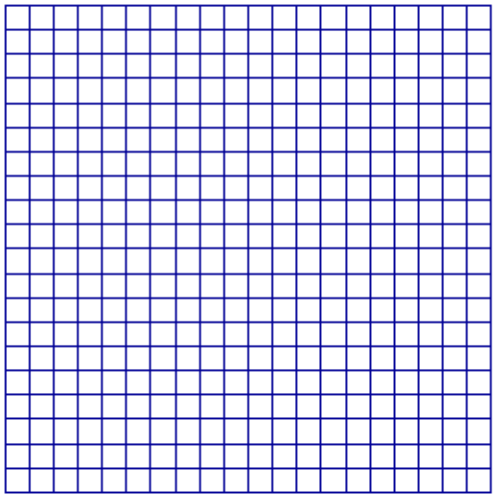
$$\phi = \begin{cases} 1 & y_1 < y < y_2 \quad or \quad y_3 < y < y_4 \\ 0 & otherwise \end{cases} . \qquad \text{(Eq. 2)}$$

Nodes in the regions $6.0 < y < 8.0$ and $12.0 < y < 14.0$ were tagged as features. This trigger function produces two parallel regions of refinement and is denoted as the "parallel" trigger function. Cells with one-half or more of their nodes tagged in this manner were marked and then isotropically refined as shown in Figure 11(b). The parallel trigger function was again applied to the mesh and nodes are tagged. In the second refinement cycle, cells in a smaller region ($6.5 < y < 7.5$ and $12.5 < y < 13.5$) were marked and refined as shown in Figure 11(c). In the third refinement cycle, shown in Figure 11(d), cells in the same region were marked and refined. This simulates a feature that is being better resolved by refinement of the mesh. It should be noted that, since the level of refinement between neighboring cells was not allowed to exceed one, some cells outside the marked region were refined as shown in Figure 11(d).

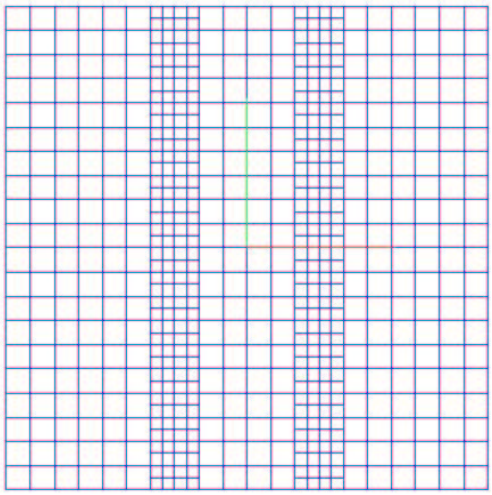
A cylindrical trigger function was then defined using,

$$\phi = \begin{cases} 1 & r_1 < r < r_2 \\ 0 & otherwise \end{cases} \qquad \text{(Eq. 3)}$$

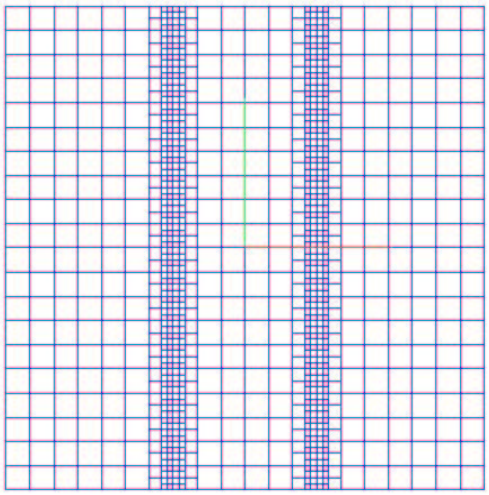
and three cycles of refinement were performed as shown in Figures 12(a)-12(d). As in the previous case, the extent of the region identified as a feature was decreased with each cycle. Thus, the refined region becomes sharper and sharper with continued refinement. Again, a few neighboring cells were refined to ensure that the two-level refinement rule was satisfied.



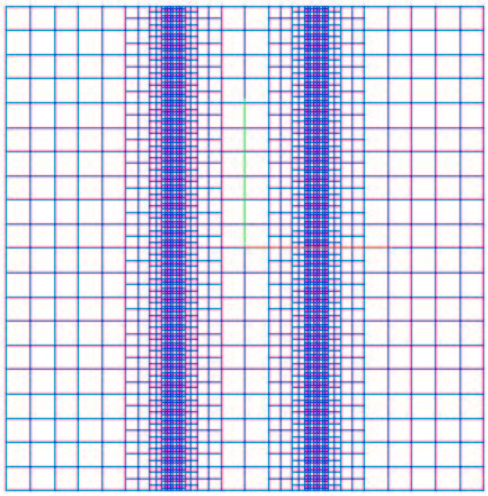
(a) Baseline mesh



(b) One refinement cycle

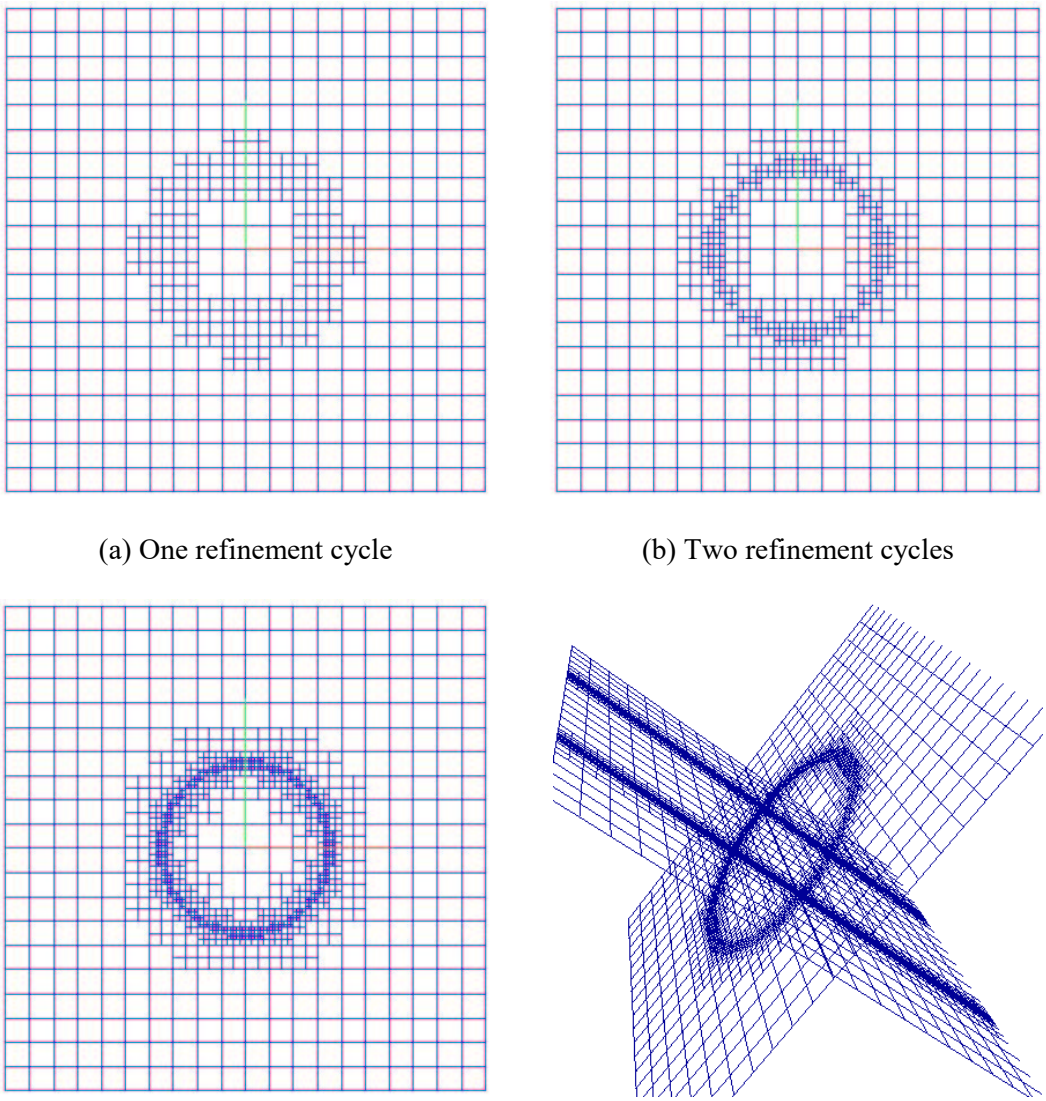


(c) Two refinement cycles



(d) Three refinement cycles

Figure 11. Isotropic refinement of a hexahedral mesh using the parallel trigger function



(a) One refinement cycle

(b) Two refinement cycles

(c) Three refinement cycles

(d) Perspective view after three cycles

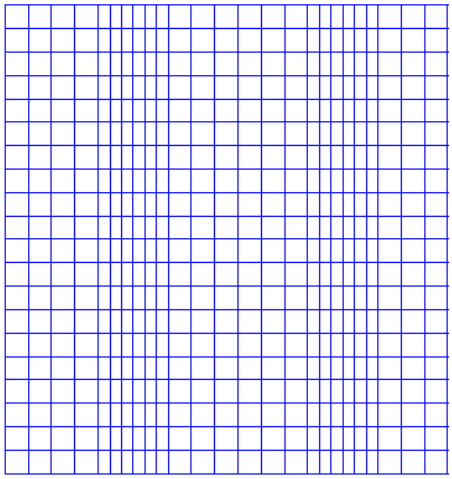
Figure 12. Isotropic refinement of a hexahedral mesh using the cylindrical trigger function

6.1.2 Cube with Hexahedral Mesh: Anisotropic Refinement

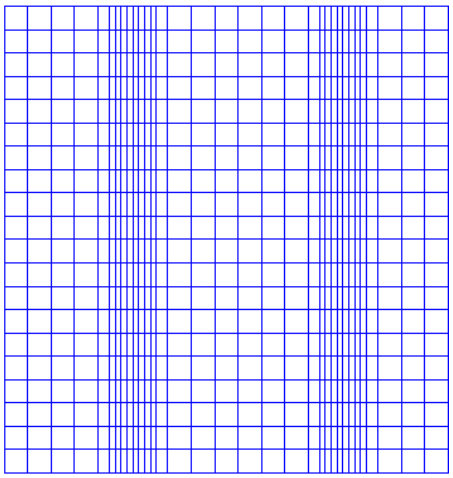
The same regular 20 x 20 x 20 hexahedral mesh was refined anisotropically using the parallel and cylindrical trigger functions. For anisotropic refinement, a direction for the refinement must also be specified. First, the parallel trigger function was applied. Initially, nodes in the regions $4.5 < y < 6.5$ and $13.5 < y < 15.5$ were identified as features and tagged. A horizontal refinement direction was specified. Cells with one-half or more of their nodes tagged were refined anisotropically. The edges that were best aligned with the specified gradient direction were refined by subdivision. Since the specified gradient was horizontal, the horizontal edges were refined as shown in Figures 13(a) and 13(b). The parallel trigger function was again applied to the refined mesh. In the second refinement cycle, cells in a smaller region ($4.5 < y < 6.5$ and $13.5 < y < 15.5$) were marked and refined. In the third refinement cycle, cells in the same region were marked. In addition, since the difference in level of refinement between neighboring cells was not allowed to exceed one, some cells outside the marked regions were refined as shown in Figure 13(b).

The same trigger function was employed with a vertical refinement direction (0,1,0) producing refinement in the same regions. However, in this case, the cells were refined by subdividing their vertical faces. The refined meshes for this condition are shown in Figures 13(c) and 13(d). Since the specified refinement direction was parallel to the orientation of the feature, the size of the feature was not reduced by refinement, as was the case in previous examples.

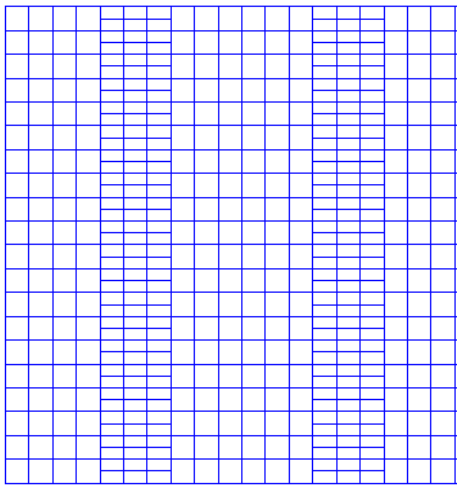
Refinement based on the cylindrical trigger function was then performed. The refinement direction was specified to be horizontal. Three cycles of refinement were performed, the results of which are shown in Figures 14(a)-14(d). After the first cycle, a smaller region of cells was marked and refined during each cycle. Again, a few neighboring cells were also refined to satisfy the two-level refinement rule



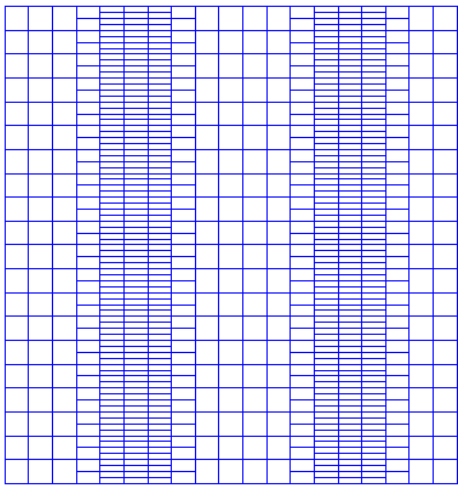
(a) One refinement cycle: Horizontal gradient



(b) Two refinement cycles: Horizontal gradient



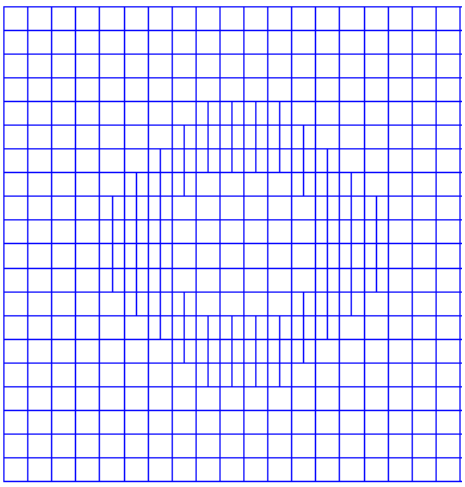
(c) One refinement cycle: Vertical gradient



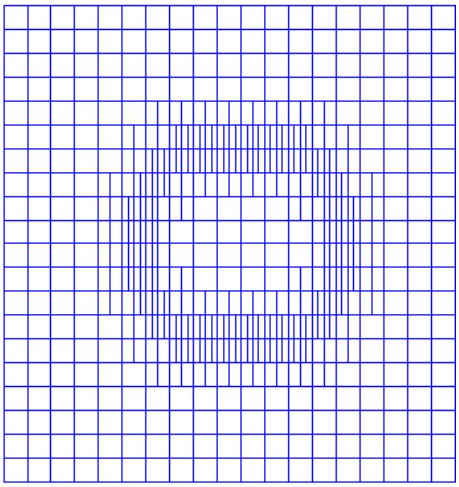
(b) Two refinement cycles: Vertical gradient

Figure 13. Anisotropic refinement of a hexahedral mesh using the parallel trigger function

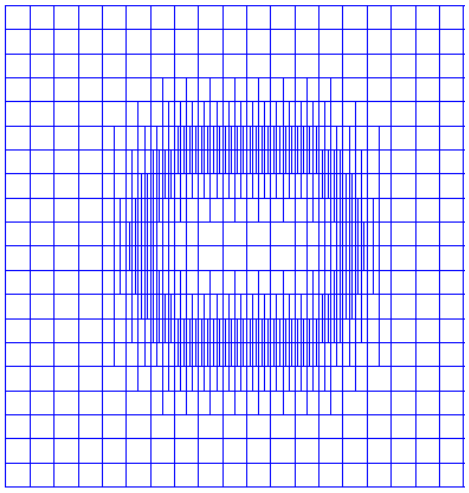
.



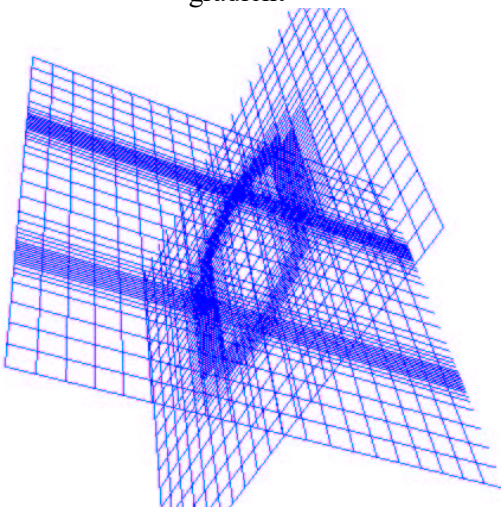
(a) One refinement cycle: Horizontal gradient



(b) Two refinement cycles: Horizontal gradient



(c) Three refinement cycles: Horizontal gradient



(d) Three refinement cycles: Horizontal gradient

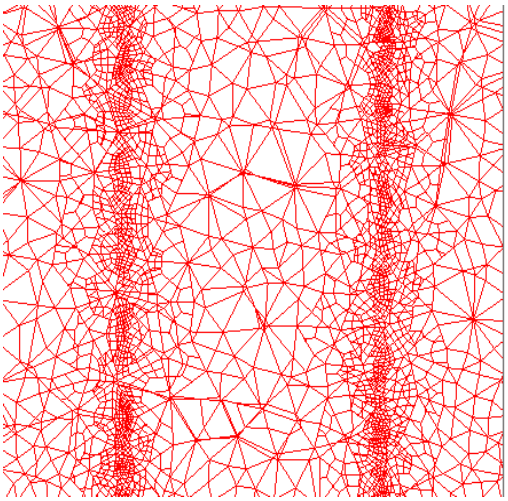
Figure 14. Anisotropic refinement of a hexahedral mesh using the cylindrical trigger function

6.1.3 Cube with Tetrahedral Mesh: Isotropic and Anisotropic Refinement

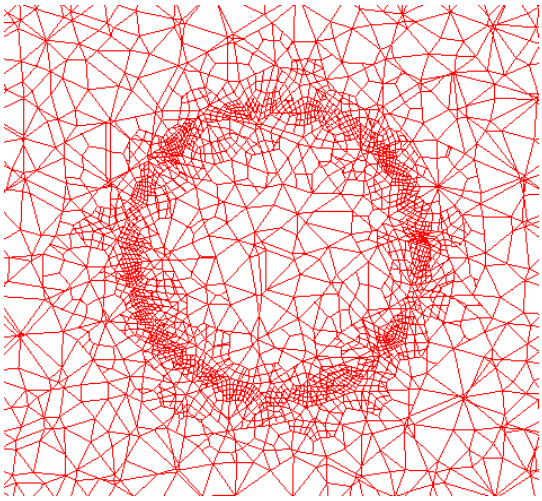
The same cube was discretized by a tetrahedral mesh generated using AFLR3 [28]. Isotropic refinement and anisotropic refinement were performed using the parallel and cylindrical trigger functions. As described in section 6.6.1, the extent of the refined region was decreased after each cycle to simulate a feature that was becoming better resolved. The

meshes resulting from three cycles of isotropic refinement with the parallel and cylindrical trigger functions are shown in Figures 15(a) and 15(b) respectively. It can be seen that isotropic refinement of the tetrahedral mesh was effective and relatively good quality elements were produced. The parallel and cylindrical trigger functions were also used with the anisotropic refinement algorithm. The refinement direction was set to be parallel to the y-axis. The meshes resulting after three cycles of refinement are shown in Figures 15(c) and 15(d). It can be seen that edges parallel to the y-axis were subdivided over multiple refinement cycles. This causes successive subdivision of angles in the tetrahedral cells producing poor quality sliver elements.

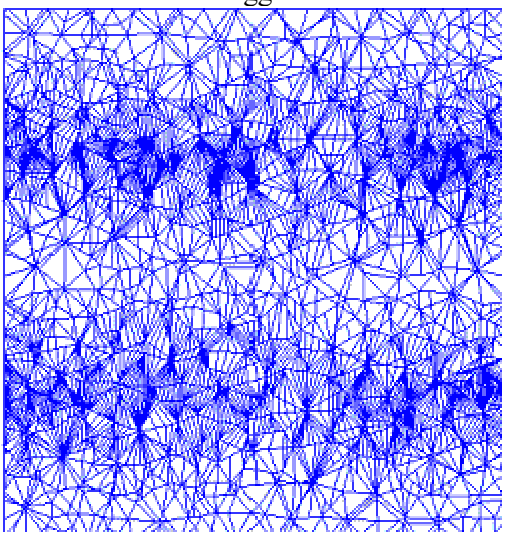
To better illustrate the problem with the anisotropic refinement procedure when applied to tetrahedral meshes, consider Figure 16, which shows a large number of high aspect ratio triangles for anisotropic refinement generated using the parallel trigger function. Since a vertical region is marked for refinement and a horizontal refinement direction is specified, edges that have the least inclination with respect to the *x*-axis are marked for refinement. In triangular faces, this process causes repeated subdivision of interior angles producing poor quality cells. However, the new faces are aligned perpendicularly to the specified direction, as desired. Alternative procedures, such as basing the refinement criteria on other factors such as edge length, may be employed to produce better quality elements. However, since the primary objective of anisotropic refinement is to improve efficiency by refining in a specified direction, these procedures are not particularly useful. A second alternative, which has more potential, is to perform an isotropic refinement first, which produces hexahedral elements, and then apply the anisotropic refinement procedure.



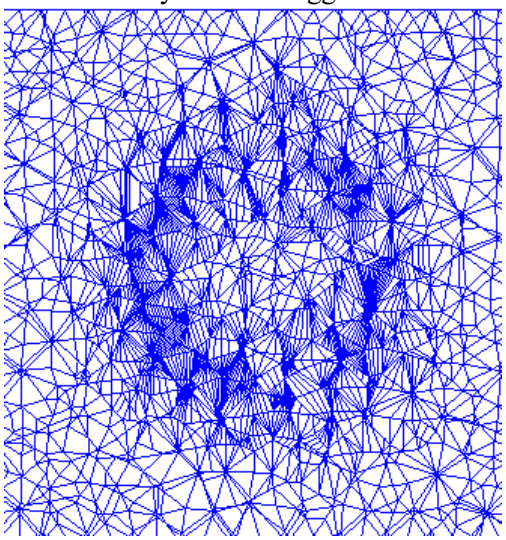
(a) Isotropic refinement using the parallel trigger



(b) Isotropic refinement using the cylindrical trigger



(c) Anisotropic refinement using the parallel trigger: Horizontal gradient



(d) Anisotropic using the cylindrical trigger: Horizontal gradient

Figure 15. Refinement of a tetrahedral mesh

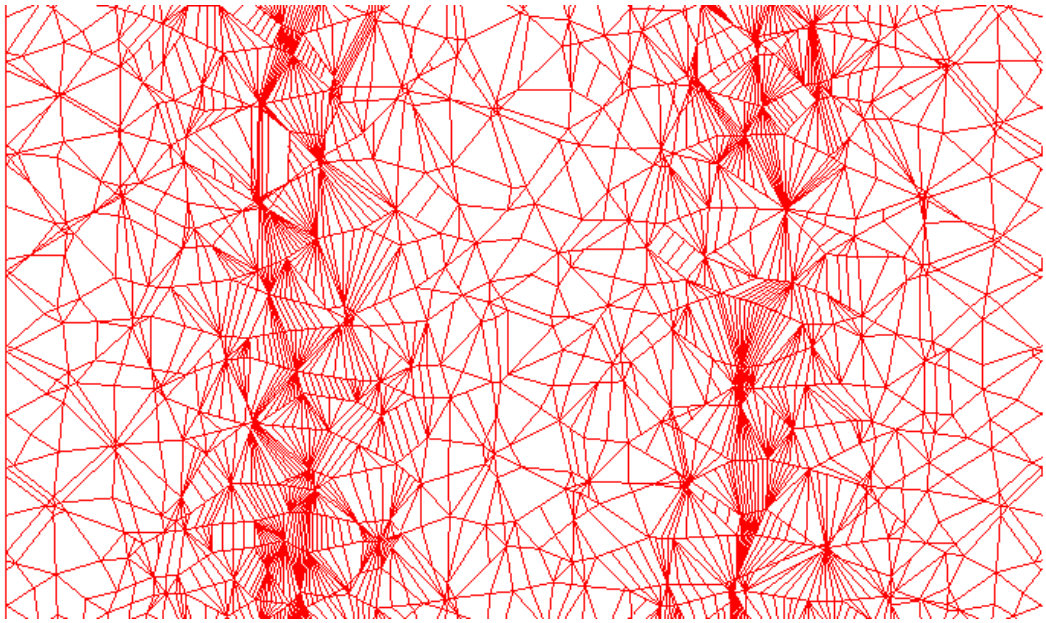


Figure 16. Anisotropic refinement using only the gradient to determine refinement direction: Horizontal gradient

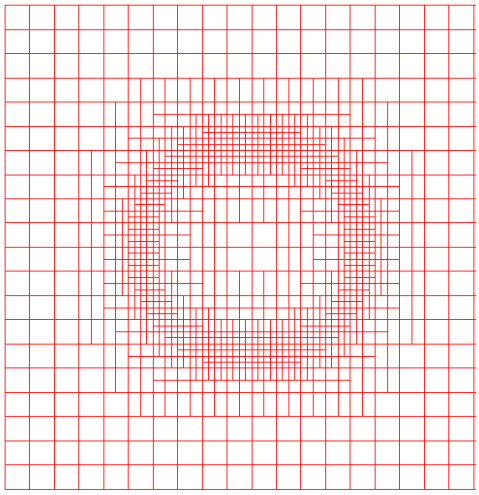
6.1.4 Achieving Psuedo-isotropic Refinement from Anisotropic Subdivision

Anisotropic refinement is very effective when cells faces are aligned with the refinement direction. Consider the refinement of the hexahedral mesh on a cube with cylindrical trigger function shown in Figure 14. Although the trigger was cylindrical, the refinement direction was defined to be horizontal. Hence horizontal edges were refined to produce vertical refinement. However, if the refinement direction were oriented at 45-degrees above the horizontal, uniform refinement in the *x* and *y* directions *and no refinement in the z direction* would be more appropriate. The similarity to isotropic refinement is obvious. Of course, isotropic refinement would also produce unneeded refinement in the *z* direction.

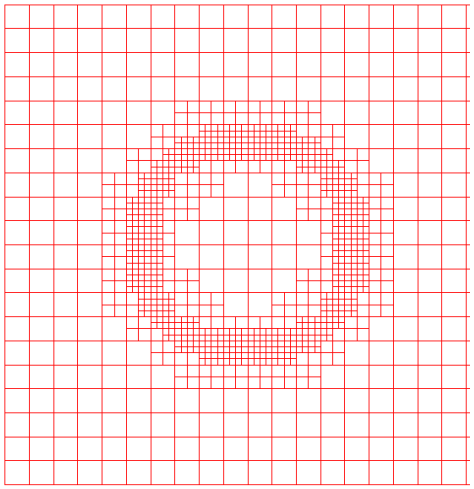
Pseudo-isotropic refinement can be obtained using multiple cycles of anisotropic refinement by including additional sorting criteria. As described previously, anisotropic

refinement of a cell is performed by first choosing an edge to refine. Edges are sorted based on their alignment to the specified refinement direction. Edges whose alignment varies within a specified threshold are further sorted based on their length and those whose lengths vary within a second threshold are further sorted based on direction with priority given to x, y, and z directions respectively.

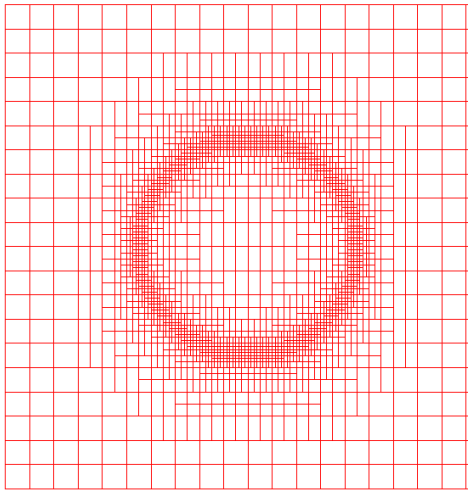
Let us consider what happens in the case described above, i.e., an isotropic hexahedral mesh when the refinement direction is oriented at 45-degrees above the horizontal. In the first cycle, all edges parallel to the x-axis or the y-axis will have the same alignment to the pressure gradient as well as the same lengths. Hence, based on the specified direction, edges parallel to the x-axis are chosen for refinement. In the second cycle, all edges parallel to the x-axis or the y-axis will still have the same alignment to the pressure gradient. However, the edges parallel to the y-axis will be twice as long (since the edges parallel to x-axis were refined) and will be chosen for refinement. Hence, after two cycles of refinement, we will have a mesh that looks similar to the one obtained after one cycle of isotropic refinement as shown in Figure 17. Similarly, four cycles of anisotropic refinement would be equivalent to two cycles of isotropic and six to three and so on. It should be noted that this approach is highly problem specific. However, it does illustrate the flexibility of refinement strategies that allow general elements. If there were an equal component of the gradient in all three directions, three cycles of anisotropic refinement would be equivalent to one cycle of isotropic refinement.



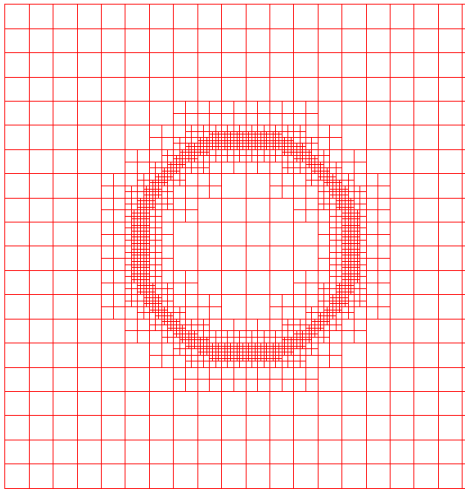
(a) Four cycles of anisotropic refinement



(b) Two cycles of isotropic refinement



(c) Six cycles of anisotropic refinement



(d) Three cycles of isotropic refinement

Figure 17. Pseudo-isotropic refinement obtained using anisotropic refinement

6.2 Supersonic Flow over a Double Ramp

The inviscid supersonic flow past a double ramp, each ramp with a turning angle of 10 degrees with respect to the incoming flow, is the next case considered. Under the specified conditions, an oblique shock forms at the first turn. The flow then encounters the second turn and a second oblique shock forms. The two oblique shocks intersect and a very weak oblique

shock emanates from the shock intersection and intersects the wall. A contact discontinuity is also formed at the intersection. Although the configuration and solution are two dimensional, the problem is solved on a three-dimensional mesh. The objective of this case was to demonstrate that refinement procedure works for a relative simple flow field.

A structured curvilinear, nearly isotropic grid was generated. The Mach number of the flow was 2. The shock sensor of Lovely and Haimes [27] was employed to trigger isotropic refinement. A solution was computed on the baseline mesh and the shock detection algorithm was employed to mark nodes where a shock occurs. An isotropic refinement cycle was then performed and a new solution was computed on the refined mesh. This process was repeated to illustrate how the size of the mesh can be constrained using an edge length criterion. In this case, the minimum edge length was specified to be 10% of the initial edge length. This would permit no more than three subdivisions of a given edge. Multiple refinement cycles were performed to obtain a converged mesh, i.e., a mesh for which no more refinement occurs with additional refinement cycles. Figure 18 shows the mesh size, as measured using the number of nodes, faces, and elements, as a function of the number of refinement cycles. Although a true steady state was not reached in thirteen refinement cycles, very few elements, relative to the total number in the mesh, were actually being refined during the final six iterations. Note that the rapid growth in mesh size was effectively curtailed by the edge length constraint after three refinement cycles.

The baseline mesh and the meshes resulting after two and 13 cycles of refinement are shown in Figures 19 (a), 19(c), and 19(f), respectively. As expected, significant refinement can be observed near the shock waves. However, the refinement region near the initial shock was not uniform as was anticipated. The causes of this nonuniformity are unknown at this time. Figures 19(b), 19(d) and 19(f) show the field pressures for the baseline mesh and after two and thirteen cycles of refinement, respectively. It can be noted that with two cycles of

refinement, the shocks are much better resolved and by thirteen cycles, they become fine lines. Isotropic refinement was very effective in improving resolution of the flow feature in this case.

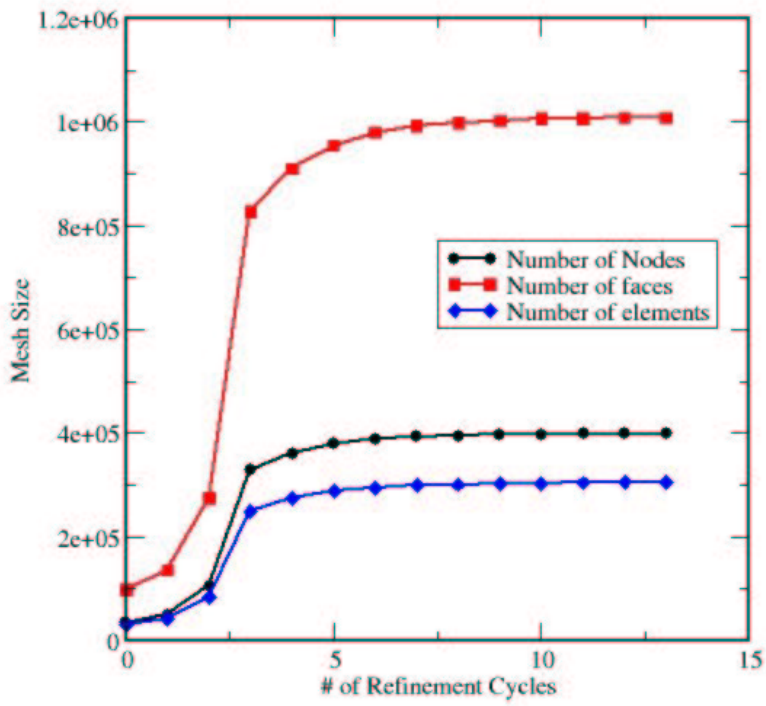
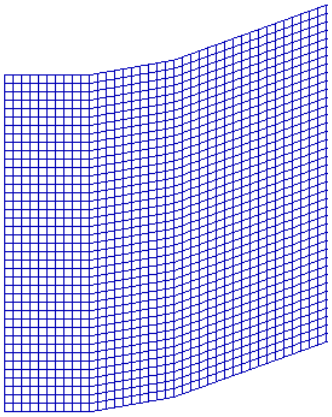


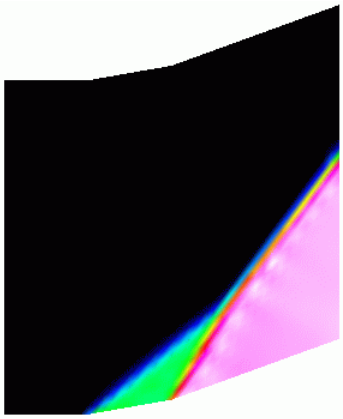
Figure 18. Mesh growth for double ramp problem: constrained isotropic refinement

6.3 Notional Re-entry Vehicle

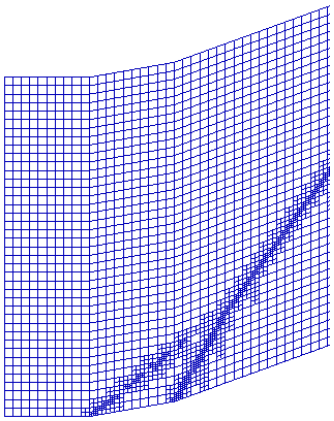
The first realistic configuration considered here is a notional re-entry vehicle similar to an X38. A surface mesh comprised of triangles and quadrilaterals was generated. The extrusion algorithm described in [5] was employed to generate the volume mesh. General elements were used in the extrusion algorithm to improve the quality of the mesh. The following conditions were employed to obtain a flow solution: a Mach number of 3 and an angle of attack of 10 degrees. The AVUS code (the noncommercial version of the Cobalt [23] code) was used to obtain an inviscid solution.



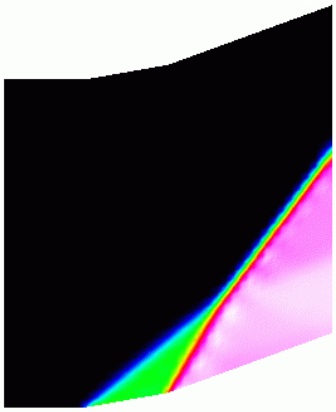
(a) baseline mesh



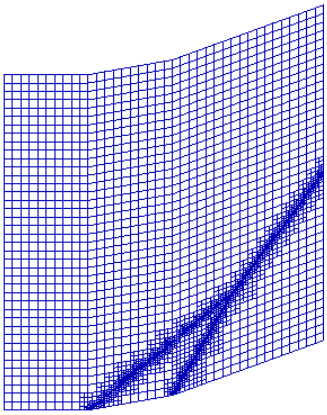
(b) Pressures computed on the baseline mesh



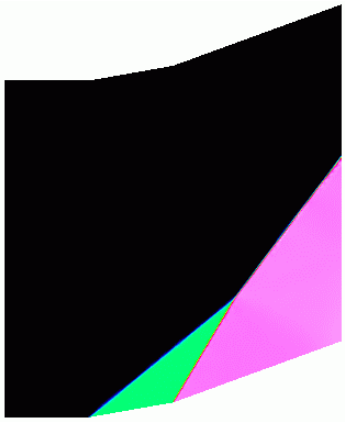
(c) Solution adaptive mesh after two refinement cycles



(d) Pressures computed on the mesh after two refinement cycles

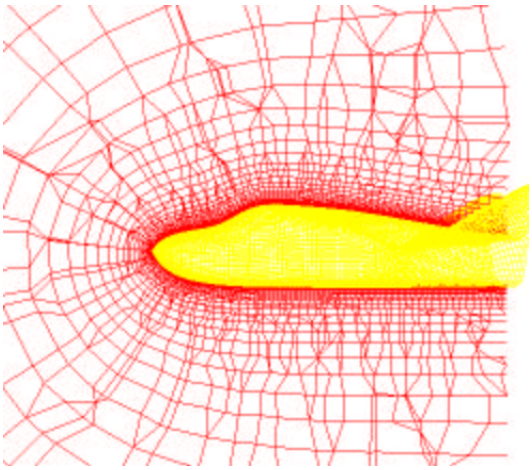


(e) Solution adaptive mesh after thirteen refinement cycles

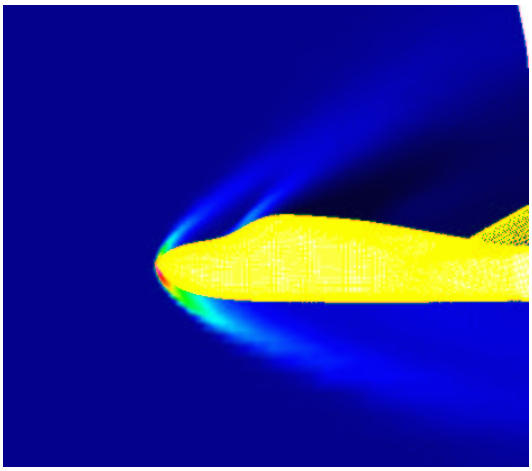


(f) Pressures computed on the mesh after 13 refinement cycles

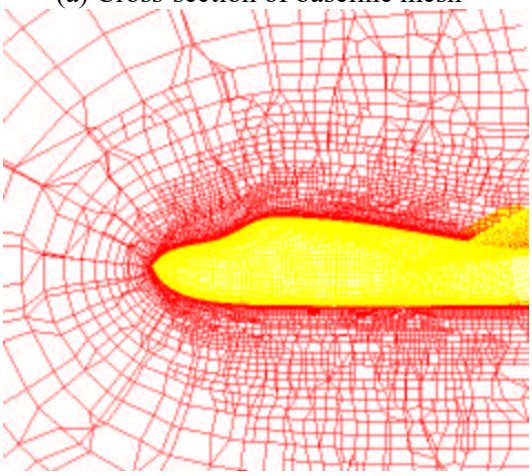
Figure 19. Supersonic flow over a double ramp: Solution adaptive mesh and pressure field



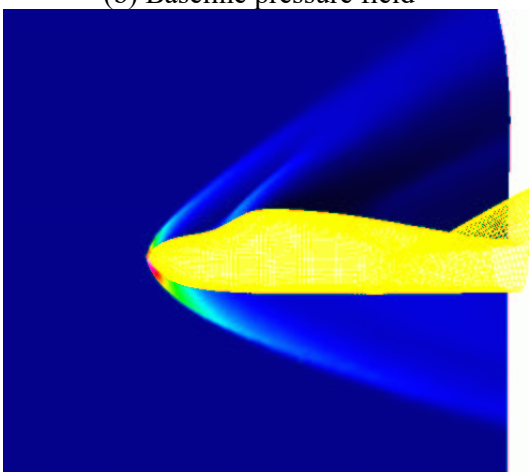
(a) Cross-section of baseline mesh



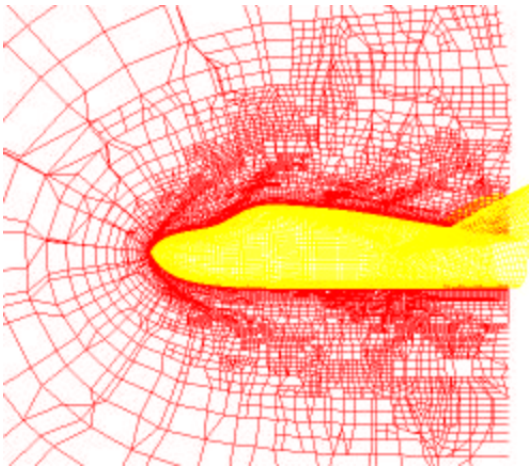
(b) Baseline pressure field



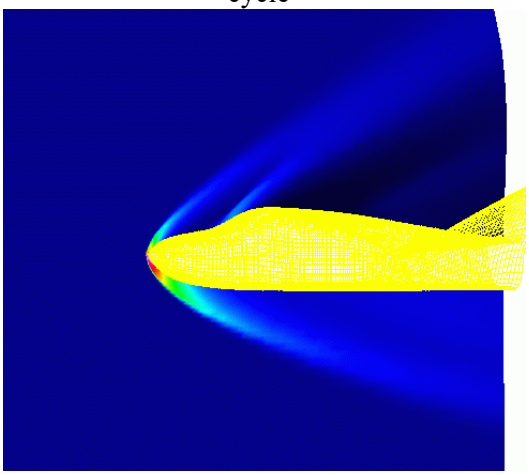
(c) Cross-section of mesh after one refinement cycle



(d) Pressure field after one refinement cycle



(e) Cross-section of mesh after two refinement cycles



(f) Pressure field after two refinement cycles

Figure 20. Isotropic refinement for notional X38 flow field

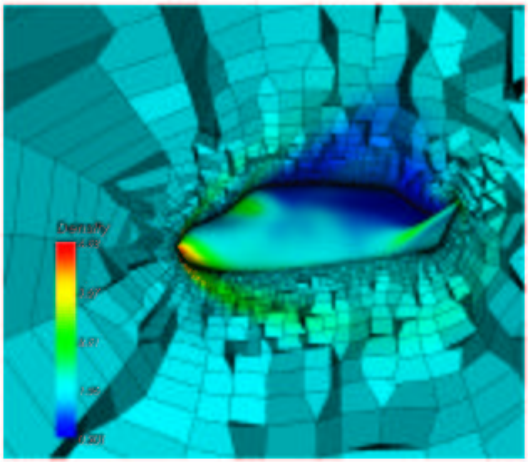
The objective of this test was to capture the shock in the flow field more accurately using isotropic refinement. The shock detection algorithm of Lovely and Haimes [27] was again employed to trigger refinement. Cells with half or more of their nodes tagged were refined isotropically. The flow solver was then run on the refined mesh. The entire cycle, detection, refinement, and flow solution was executed again for a total of two refinement cycles.

Figure 20(a) shows a cross section of the baseline mesh in the symmetry plane. Figure 20(b) shows a shaded surface plot of the pressure field in the symmetry plane. Figures 20(c) and 20(d) show the mesh and the pressure field in the same cross section of the mesh after one refinement cycle. Figures 20(e) and 20(f) show the mesh and pressure field after two refinement cycles. It can be seen that the bow shock becomes better resolved on the refined meshes after only one refinement cycle. Refinement also improves the resolution of the secondary shock on the lee side. On the windward side, relatively weak shock waves due to “ripples” in the geometry are refined.

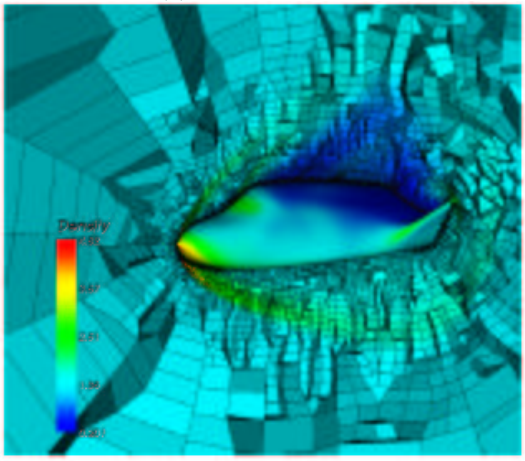
Figures 21(a), 21(b) and 21(c) show cross sections of the baseline volume mesh and the mesh after one and two cycles of isotropic refinement, respectively, in an axial cutting plane and a longitudinal cutting plane. In these images, polygonal faces that intersect the cutting plane are rendered which produces the “crinkled surface effect.” The mesh faces are colored based on the density field. It can be observed that the detached bow shock becomes more prominent with refinement.

When the refined mesh was examined in detail, it was observed there were regions of the mesh that were refined even though they were not marked by the shock detector. The horizontal line of refined cells just above the body surface, shown in Figure 22 is one such example. The cells in this region of the baseline mesh have faces that are twisted (non-planar faces). Their degree of twist is just below the threshold that the flow solver will accept. If a

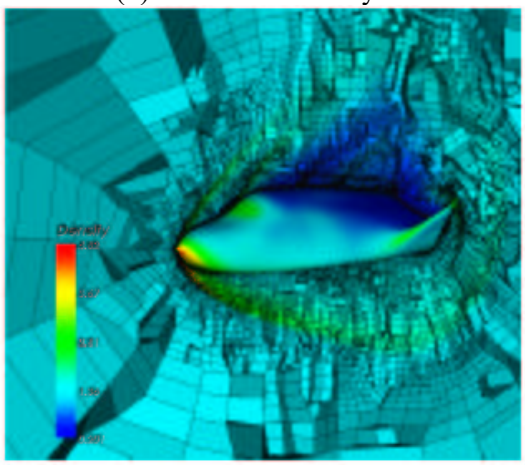
cell on one side of such a face is subdivided while the other is not, the larger cell fails the AVUS “folded faces check.” To avoid this issue, when a folded face is encountered, the two cells containing it are both refined or both not refined. Hence, when a cell on one side of a non-planar face is marked for subdivision, the cell on the other sided is marked too. This in turn may force additional cells to be refined until the condition is satisfied. One of the cells on the row of refined elements shown in Figure 22 was marked for subdivision and, due to the presence of a set of twisted faces, the cells in the layer were refined.



(a) Baseline mesh



(b) One refinement cycle



(c) Two refinement cycles

Figure 21. Mesh at different isotropic refinement cycles colored by density for notional X38

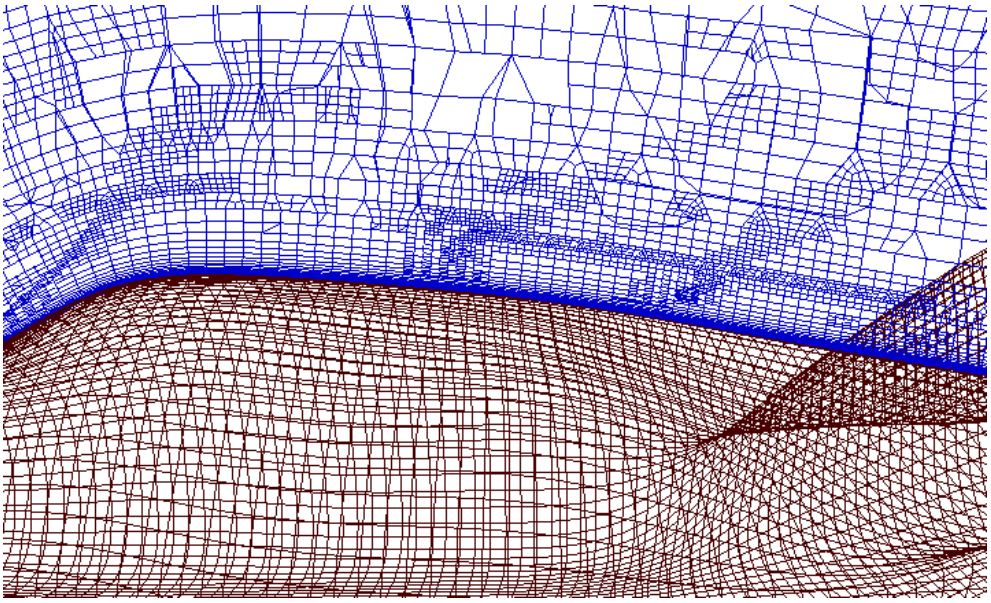


Figure 22. Close-up of notional X38 surface mesh and mesh in the symmetry plane

6.4 Combustion in a Single-element Injector

For the next case, a simulation of hydrogen-oxygen combustion in a single-element coaxial injector of a rocket motor was performed. Pennsylvania State University performed experiments for NASA to create a database that can be used to validate rocket motor simulations. These experiments employed a range of chamber pressures with both ambient injection and injection with pre-burners for the fuel and oxidizer. This experimental data was compared with the results of the simulation.

The objective of this test is to resolve the flame better using isotropic refinement. We chose two cases from the experimental data for comparison with our simulations. In one, hydrogen and oxygen gases were injected at ambient temperatures to maintain a chamber pressure of 300 psia. For the preburner case, the chamber pressure was 750 psia. The CHEM code [29] was used for the simulation. CHEM is a finite-rate chemically reacting flow solver, which supports simulations on meshes with arbitrary connectivity. Fluid transport properties

were provided by the CHEMKIN package. To reduce the computational cost of the simulation, the domain of computation for the axisymmetric injector was modeled as a wedge. The extrusion algorithm described in [5] was used to generate the mesh. The generated mesh was rotated about axis of the configuration to form the required wedge-shaped mesh. When refining the mesh, constraints were applied to ensure no refinement occurred in the circumferential direction.

Figure 23 illustrates the geometry and the computed temperature field. The flow is from left to right and the oxygen is injected along the axis of the injector. Due to the combustion, a flame arises inside the combustion chamber. Therefore, the regions where combustion occurs can be deduced from by the temperature contours. The exhaust gases exit the nozzle on the right. Flow inside the combustion chamber is shown in Figure 24. The background color in Figure 24 is the fluid temperature and the fluid velocity is indicated by the streamline colors. A large re-circulation bubble dominates the flow pattern. There is a characteristic secondary re-circulation region in the upper left corner of the chamber. The heat flux, which is measured on this outer wall, is strongly influenced by these flow features.

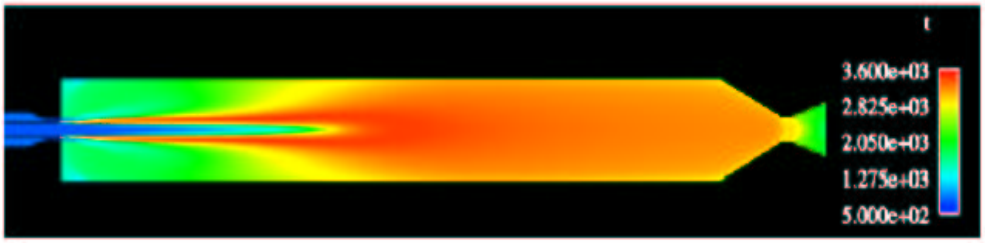


Figure 23. Temperature field for single-element injector

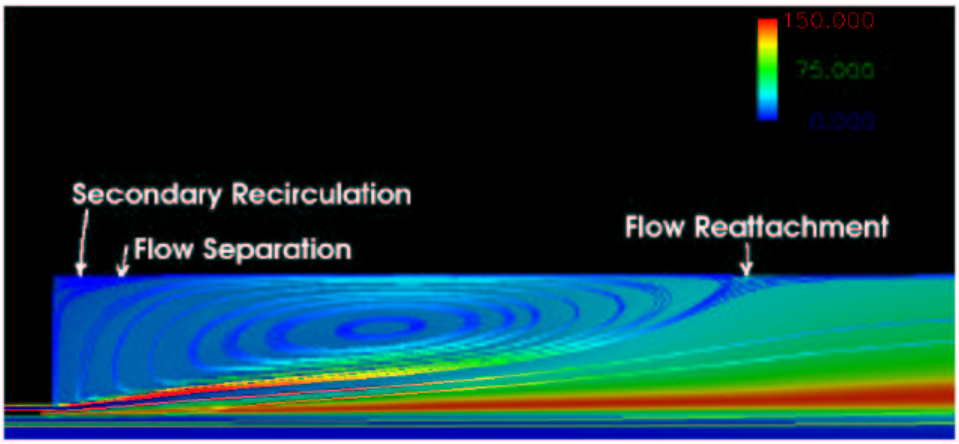
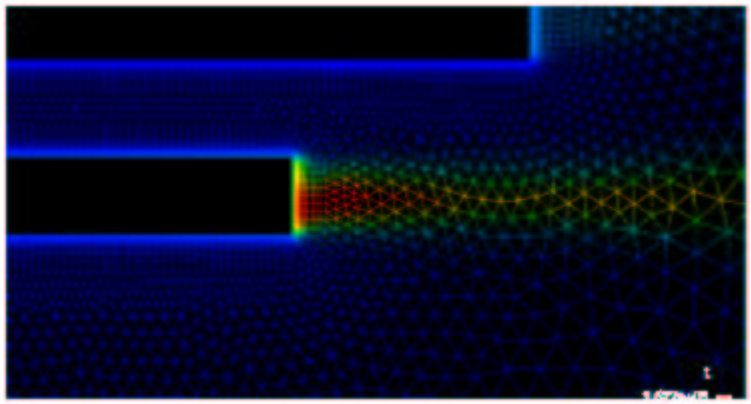


Figure 24. Streamlines colored by velocity magnitude superposed over temperature field

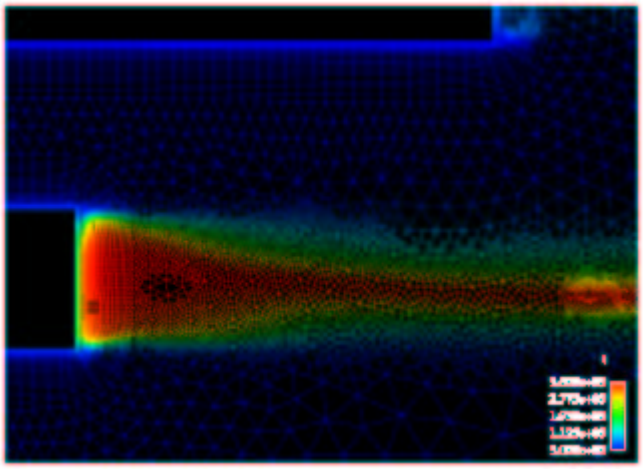
An error estimator based on the jumps in temperature and species mass fractions was used to tag nodes and trigger refinement. When the estimated error exceeded the mean value plus one standard deviation, refinement was triggered. Figures 25(a) and 25(b) show the original mesh and the mesh after three cycles of isotropic refinement along with the predicted temperatures, respectively. Clearly, it can be seen that the flame is better resolved through refinement.

Figures 26(a) and 26(b) show comparisons of simulated results with experimental data. The location of flow reattachment is labeled “R” and the location of flow separation at the secondary re-circulation as “S”. Zero crossings of the computed wall shear stress were used to identify these locations. These positions are also shown in the streamline patterns in Figure 24. In the ambient gas case, the results compared well with the experimental data. However, in the case with a pre-burner, the wall temperatures in the simulated case over-predicted the heat flux in the downstream region by more than 50%. A secondary peak in the simulated results, which is not observed in the experimental data, possibly causes this discrepancy. At this time, the cause of this secondary peak is not known. In general, the heat

flux is found to be insensitive to the mesh refinement. It should be noted, that the flow characteristics in the region where the heat fluxes are measured are strongly dependent on the turbulence model. We do consider the agreement between the prediction and the experimental data to be good considering the difficulty of the problem at hand.

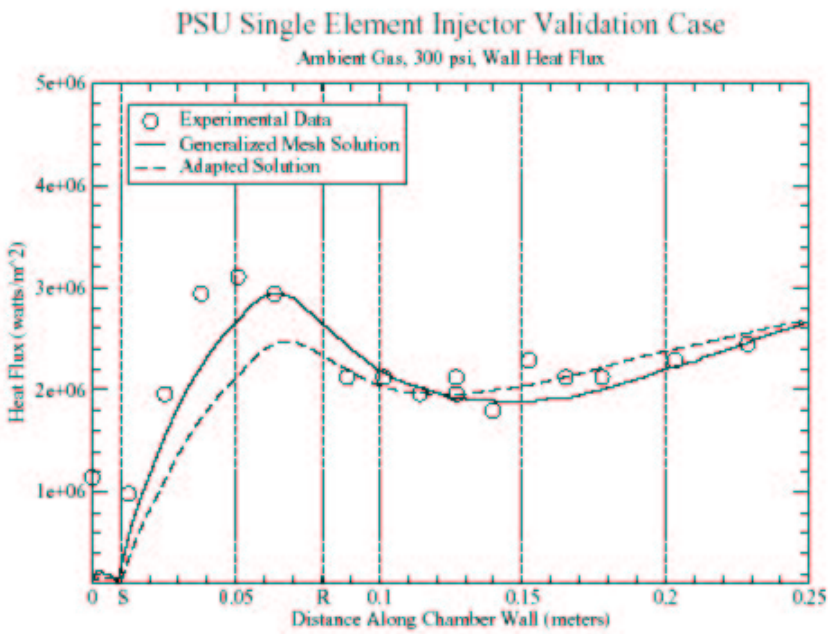


(a) Computed temperature field on the baseline mesh

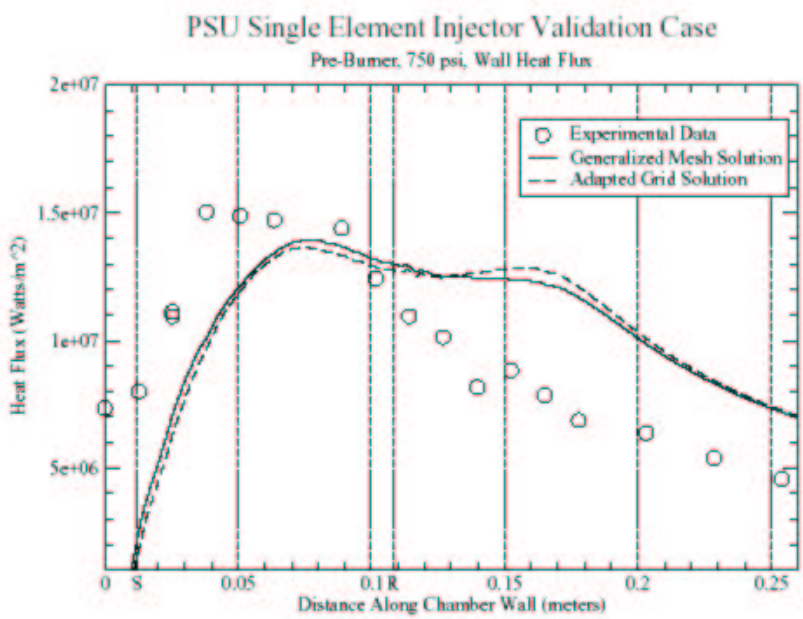


(b) Computed temperature field after three isotropic refinement cycles

Figure 25. Flame attachment for single-element injector



(a) Cold gas injection



(b) Hot gas injection

Figure 26. Comparison of experimental and computed wall heat fluxes for the single element injector configuration

6.5 Hypersonic Flow over a Sphere

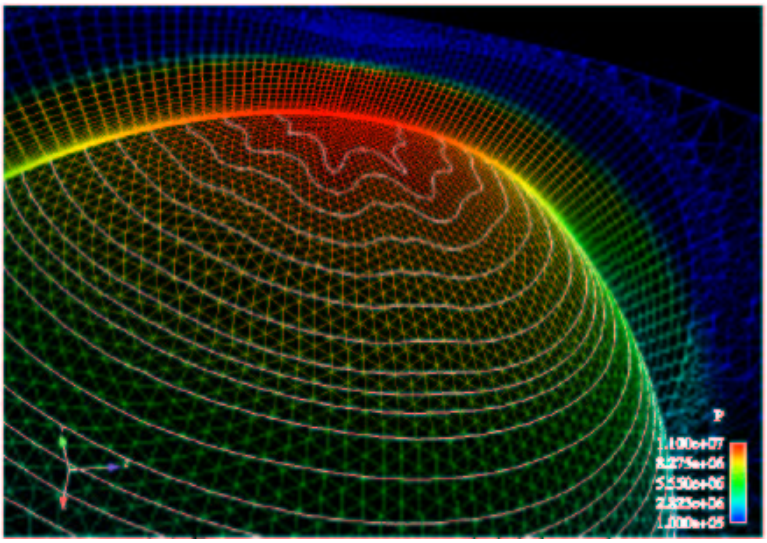
A simulation of a hypersonic flow over a sphere was then performed to illustrate anisotropic refinement. The initial mesh was generated using SolidMesh [28] and contained prisms, pyramids, and tetrahedra. The following conditions were used: air pressure of 1atm, air temperature of 288K, and a free stream Mach number of 9. The simulation was performed using the CHEM code [29]. The five-species air chemistry model of Kang and Dunn [30] was used to account for the dissociation and recombination reactions.

Under these conditions, a strong detached bow shock forms just ahead of the sphere. As with most flow simulations performed using flow solvers that employ a Riemann problem-based flux definition, shocks tend to align with cell faces. Because the shock was not precisely aligned with the shock, the shock exhibits “stair stepping.” The proximity between the sphere and the shock causes the surface pressure contour of the sphere to be adversely affected by the errors in the shock computation as shown in Figure 27(a).

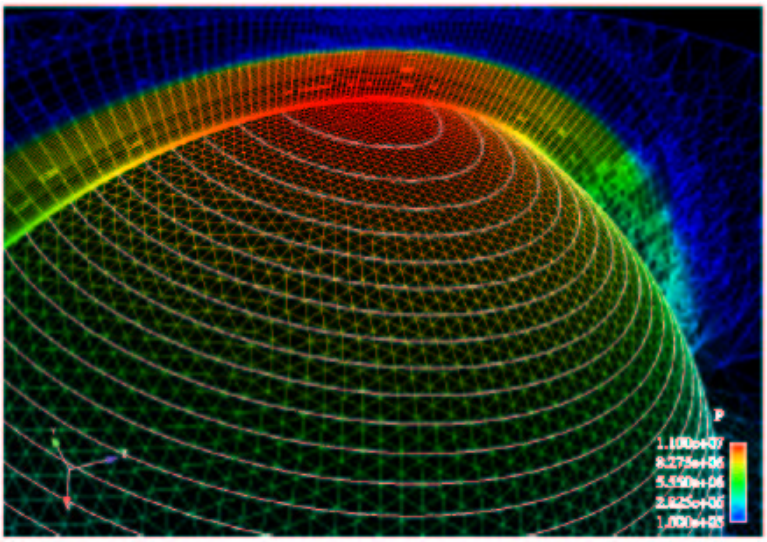
The objective of this test was to reduce the effects of the stair stepping by refining cells anisotropically in the shock region in the direction normal to the shock. For anisotropic refinement to be effective, the flow feature we are attempting to refine should be aligned with the mesh faces.

A simple pressure-based shock sensor was employed for this problem. When the ratio of extrapolated pressures on the two sides of a face exceeded two, the cells sharing the face were tagged as parts of the shock. Further, if the edge vector was parallel to the pressure gradient, it was normal to the shock and tagged for refinement. Eight cells downstream of the shock were identified using the same vector and their nodes were also marked for refinement. The direction of refinement was determined by the pressure gradient at the shock. This vector was advected downstream along the eight identified cells to preserve the direction on any additional nodes downstream of the shock that may have been tagged. Ansiotropic refinement

was performed and the flow solution was computed on the refined mesh. Another cycle of detection, refinement and flow solution was then performed. Figure 27(b) shows the surface pressure contour on the sphere after two cycles of refinement. Considerable improvement can be observed. This case demonstrates that the anisotropic refinement can improve the quality of the flow solution.



(a) Baseline mesh



(b) Solution adaptive mesh

Figure 27. Pressure contours for sphere in hypersonic flow

6.6 Transonic Flow over an M6 Wing

The classic ONERA M6 three-dimensional test case is considered as the final test case. This is a transonic viscous flow problem. The flow conditions used for this case are, a freestream Mach number of 0.8395, an angle of attack of 3.06 degrees, a mean chord length of 0.64607 meters and a wingspan of 1.1963 meters. The free stream pressure and temperature were 256 Kelvin and 80,795 Pascals respectively. The Reynolds number of 11.72x10e6 was obtained. Under the specified conditions, the flow field is characterized by a swept shock just aft of the leading edge and a second recovery shock that is initiated at approximately 60% of the chord and is swept slightly forward. These two shocks merge at a distance approximately 85% of the span from the root. It should be noted that these shocks are much weaker than the shock present in the hypersonic sphere flow field. An edge length constraint was employed to reduce the size of the refined meshes.

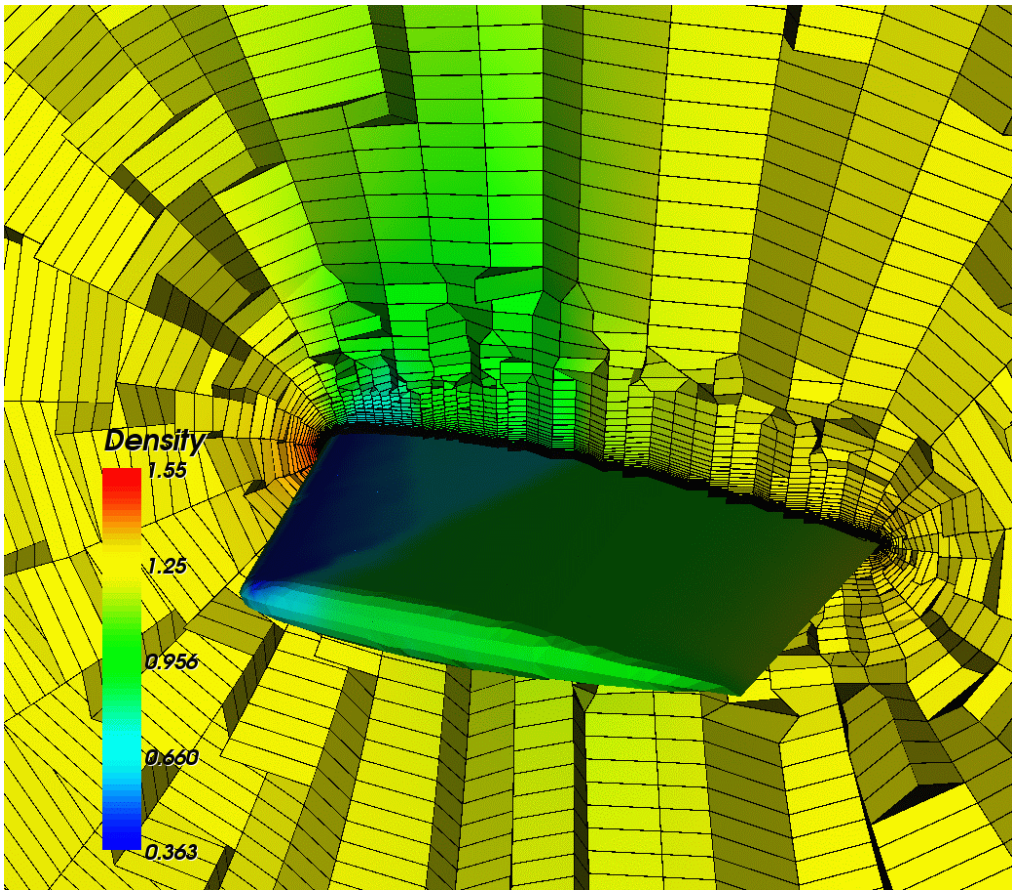
AFLR3 [28] was used to generate the surface mesh and the extrusion algorithm described in [5] with quality improvements was used to build the volume mesh. Because of the simplicity of the configuration, the mesh was extruded all the way to the outer boundary of the domain with no void-filling tetrahedrons.

A cross-sectional view of the baseline mesh taken at y/b of 0.625 is shown in Figure 28(a). The meshes in all the images are colored by the density value. Figures 28(b) and 28(c) show detailed views of the leading edge shock and the recovery shock at this span wise station, respectively.

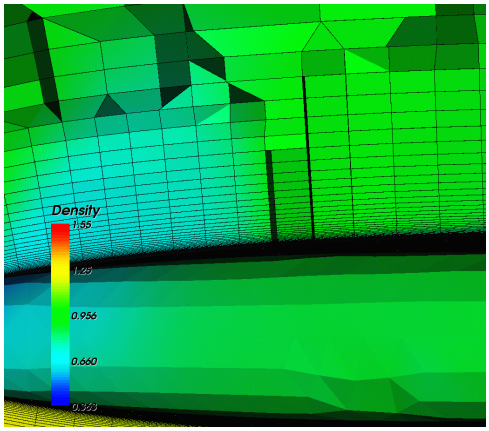
Figure 29 shows the results of isotropic refinement. Figure 29(b) shows a detailed view of the leading edge shock and the refinement pattern near the shock transition after three cycles of refinement. The refinement upstream of the shock is anomalous and is possibly caused by the sensor reacting to noise in the data. Figure 29 (c) shows a detailed view of the recovery shock and the refinement pattern after three cycles of refinement. The requirement

that there be no more than a difference of one level of refinement in adjacent cells causes some cells downstream of the shock to be refined.

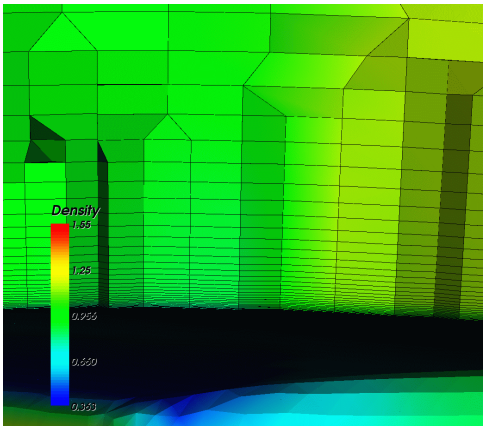
Figure 30 shows the results of anisotropic refinement. A detailed view of the leading edge shock and the resulting refinement pattern is shown in Figure 30 (b). Some vertical refinement can be viewed at the leading edge of the shock and, just as in the prior case, there is some anomalous refinement upstream of the shock. From the orientation of the refinement, the anomalous feature can be deduced to be parallel to the surface. Figure 30 (c) shows a detailed view of the recovery shock and the refinement patterns. Again vertical refinement in the expected region can be noted along with some anomalous horizontal refinement upstream.



(a) Chord-wise section at y/b = 0.625

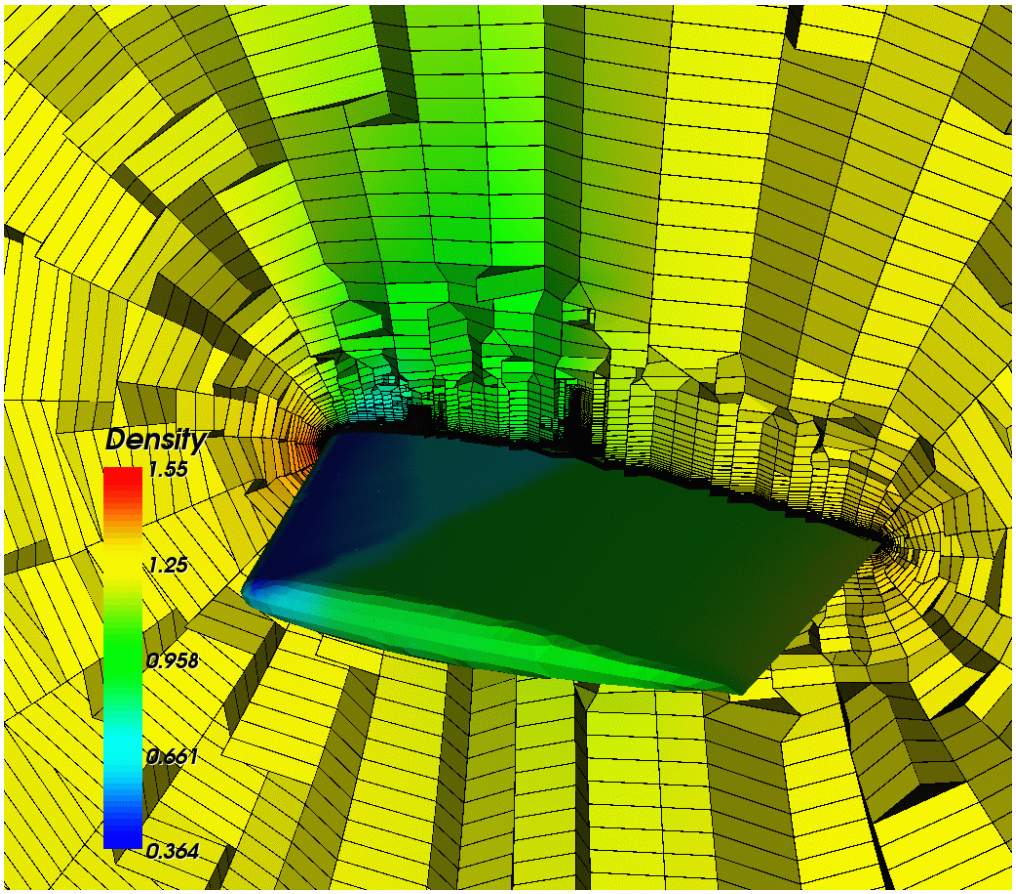


(b) Detail of leading edge shock

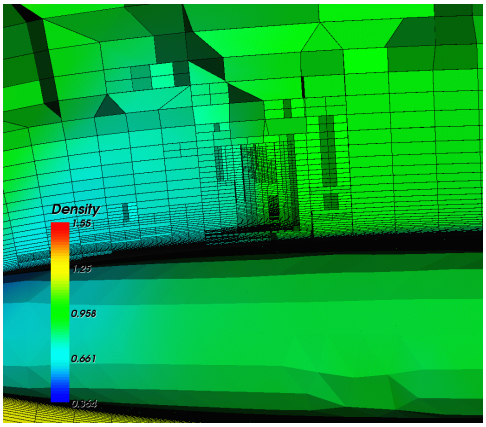


(c) Detail of recovery shock

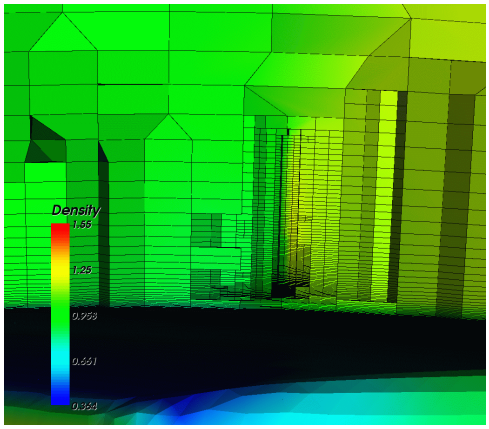
Figure 28. Solution on baseline mesh for M6 wing (Mesh colored by density)



(a) Chord-wise section at y/b = 0.625

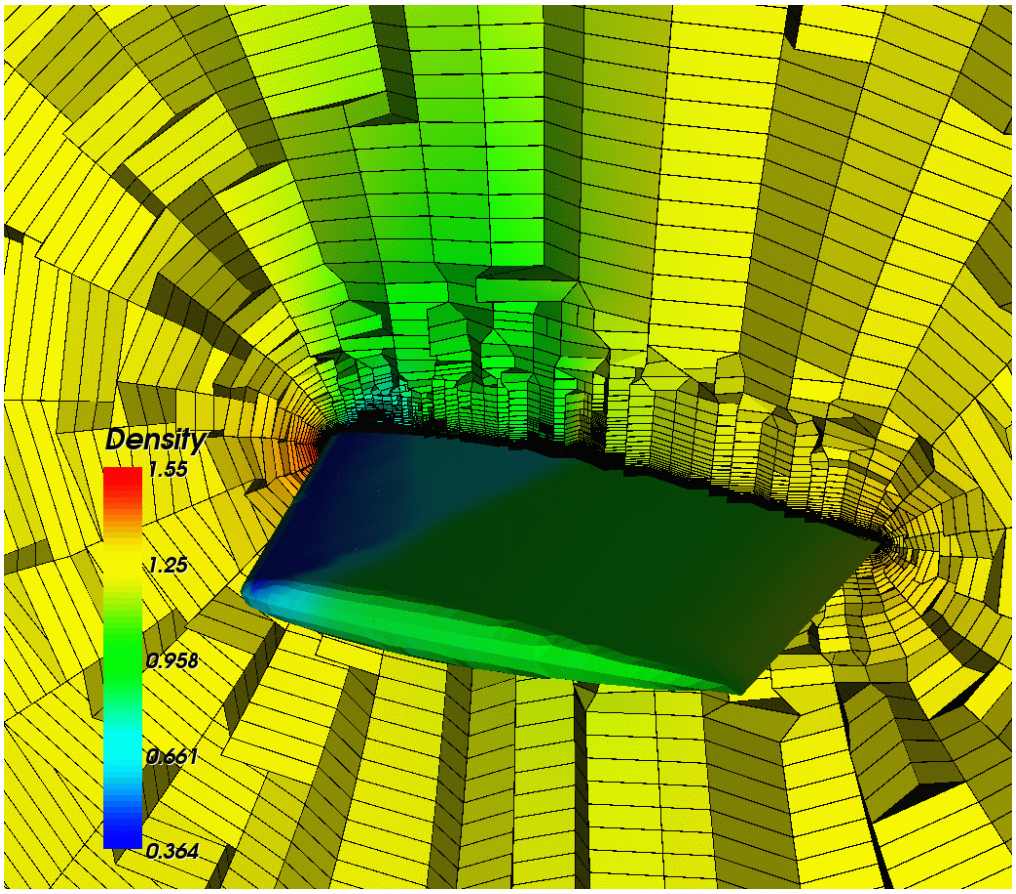


(b) Detail of leading edge shock

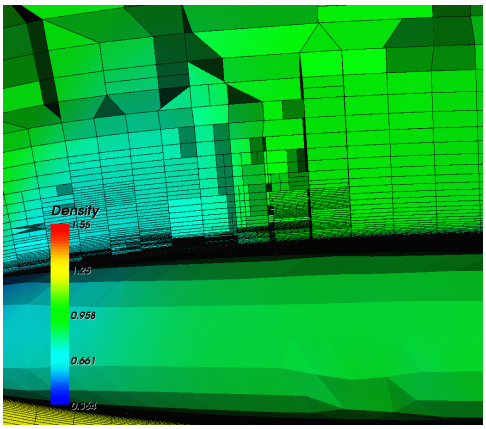


(c) Detail of recovery shock

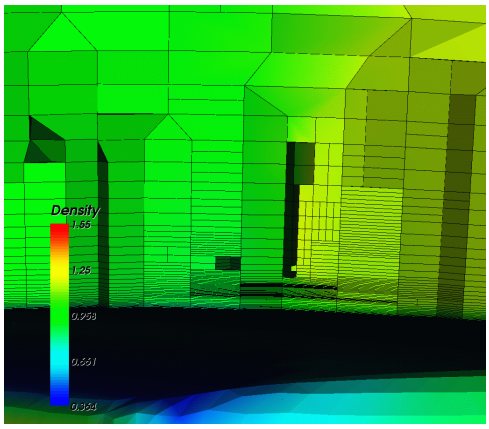
Figure 29. Solution for M6 wing after three cycles of isotropic refinement
(Mesh colored by density)



(a) Chord-wise section at y/b = 0.625



(b) Detail of leading edge shock



(c) Detail of recovery shock

Figure 30. Solution for M6 wing with 3 cycles of anisotropic refinement
(Mesh colored by density)

CHAPTER VII

CONCLUSIONS

Solution adaptive isotropic and anisotropic mesh refinement using general elements has been demonstrated successfully demonstrated for various flow problems. In some cases, refinement improved the quality of the solution significantly while in other cases the improvements were mild. Use of generalized meshes helped to confine the refinement locally. The versatility of general elements made it possible to have different subdivision algorithms such as face-based isotropic refinement and edge-based anisotropic refinement on individual elements without requiring unnecessary subdivision of neighbors and the resultant propagation of refinement.

The study demonstrates that general elements can be employed effectively in solution adaptive meshes generated using refinement thereby validating the thesis statement. Generalized elements provide versatility and locality, which facilitate effective refinement while minimizing the increase in mesh size. While the two techniques of refinement were shown to be effective, they still possess some limitations that must be overcome to increase their utility. The main limitation of isotropic refinement is the significant increase in the mesh size. Even though the generalized strategy does limit mesh refinement somewhat, the number of children produced when an element is subdivided isotropically is equal to the number of nodes of the element. While anisotropic refinement produces a more modest increase in mesh size, the flow features must be aligned with element faces for maximum effectiveness – a relatively stringent requirement.

Development of effective and accurate sensors is critical to the success of any refinement methods. Anomalous refinement patterns were observed in the refinement of the meshes for the double ramp and M6 wing flows. This is due to the inadequacy of the shock sensor. It is likely cause by the fact that the underlying shock definition employed here satisfies a necessary, but not sufficient condition for the presence of a shock wave.

There is still significant work that remains. One possible approach is to employ a combination of both isotropic and anisotropic refinement. In this way, regions with flow features aligned with element faces can be refined anisotropically. In regions where the features are not aligned with the mesh, isotropic refinement may be employed. This would alleviate the limitations of the two approaches and provide more flexibility. Implementation of this approach would require the possibility of two neighboring cells to be subdivided using the different strategies – one isotropically and the other anisotropically. Hence, seamless integration of the two techniques is needed. An alternative strategy is to perform an alignment of the mesh with the feature before anisotropic refinement is applied.

However, neither of these strategies address issues related to anisotropic refinement applied to elements containing triangular faces (or any odd-sided faces). The current approach subdivides interior angles, which degrades mesh quality with multiple refinements. Since isotropic refinement tends to produce hexahedral elements, it may be possible to apply isotropic refinement followed by a face alignment cycle and anisotropic refinement.

Finally, the sensors employed here to identify shocks and tag cells for refinement need to be improved. The sensors appeared to be more effective for stronger shocks. Other types of sensors should be investigated.

REFERENCES

[1] J. F. Thompson, B. K. Soni, N. P. Weatherill, eds, Handbook of Mesh Generation, CRC Press, Boca Raton, 1999.

[2] Khawaja, T. Minyard, and Y. Kallinderis, "Adaptive Hybrid Mesh Methods," *Comput. Methods Appl. Mech. Engrg.*, pp. 1231-1245, 2000.

[3] D. S. Thompson and B. K. Soni, "Semistructured Mesh Generation in Three Dimensions using a Parabolic Marching Scheme," *AIAA J.*, Vol. 40, No. 2, February, 2002, pp. 391-393.

[4] S. Chalasani, D. Thompson, and B. Soni, "Topological Adaptivity for Mesh Quality Improvement," Proceedings of the 8th International Conference on Numerical Mesh Generation in Computational Field Simulations (available on CD), Honolulu, HI, June 2002.

[5] S. Chalasani and D. Thompson, "Quality Improvements in Extruded Meshes using Topologically Adaptive Generalized Elements," *Int. J. Num. Methods Eng.*, (to appear).

[6] D.S. McRae, "Adaptive Mesh Algorithms - A review of progress and future research needs", AIAA J. 2001-2551

[7] D.G.Martineau, J.M.Georgala, "A Mesh Movement Algorithm for High Quality Generalised Meshes", AIAA Paper 2004-0614, Presented at 42th Fluid Dynamics Conference and Exhibit, Reno, NV, January 2004.

[8] C. Farhat, C. Degand, B. Koobus, M. Lesoinne, "Improved Method of Spring Anology for Dynamic Unstructured Fluid Meshes," AIAA Paper 98-2070, April 1998.

[9] C.Y.Lepage, F.S.Gulick, W.G.Habashi, "Anisotropic 3-D mesh Adaptation on Unstructured Hybrid Meshes", AIAA J. 2002-0859

[10] Guoping Xia, Ding Li, Charles L. Merkle, "Anisotropic Mesh Adaptation on Unstructured Meshes", AIAA Paper 2001-0443, Presented at 39th Aerospace Sciences Meeting and Exhibit, Reno, NV, January 2001.

[11] Nakahashi, K., Deiwert, G. S., "Three-Dimensional Adaptive Mesh Method," AIAA Journal, Vol. 24, No. 6, 1986, pp.948-045.

[12] A.Kawaka, T.Minyard, Y.Kallinderis, “Adaptive Hybrid Mesh Methods”, Computer Methods in Applied Mechanics and Engineering 189 (2000) 1231-1245.

[13] W. B. Tauber, “Adaptive Mesh Strategy For Unstructured Meshes On Multi-Mechanics Infrastructure,” AIAA Aerospace Sciences Meeting and Exhibit, AIAA 2004-88.

[14] Mavriplis, D.J. “adaptive Meshing Techniques for Viscous Flow Calculation on Mixed Element Unstructured Meshes”, International Journal For Numerical Methods In Fluids, 34, pp 93-111, 2003.

[15] Biswas, R. and R. C. Strawn, “tetrahedral and Hexahedral Mesh Adaptation for CFD Problems”, Applied Numerical Mathematics, 26, pp 135-151, 1998.

[16] Rivara, Maria-Cecilia, “A Mesh Generator Based on 4-Triangles Conforming Mesh Refinement Algorithms”, International Journal For Numerical Methods in Engineering, Vol. 24, pp, 1343-1354, 1987.

[17] Liu, A. and B. Joe, “Quality Local Refinement Of Tetrahedral Meshes Based on Bisection”, SIAM J. Sci. Comput. Vol. 16, No. 6, pp. 1269-1291, November 1995.

[18] M. Leatham, S. Stokes, J. A. Shaw, J. Cooper, J. Appa, and T. Blaylock, “Automatic Mesh Generation for Rapid-Response Navier-Stokes Calculations,” AIAA 2000-2247, presented at the AIAA Fluids 2000 Conference and Exhibit, Denver, CO, June 19-22, 2000.

[19] Cary and T. Michal, “Generalized Prisms for Improved Mesh Quality,” AIAA Paper 2001-2552, presented at the AIAA 15th Computational Fluid Dynamics Conference, Anaheim, CA, June 11-14, 2001.

[20] G. S. Spragle, W. A. Smith, J. M. Weiss, “Hanging Node Solution Adaptation on Unstructured Meshs,” AIAA Aerospace Sciences Meeting and Exhibit, 1995.

[21] J. Muller, T. Schonfeld, M. Rudgyard, “A Comparison of the Treatment of Hanging Nodes For Hybrid Mesh Refinement,” AIAA Computational Fluid Dynamics Conference, 1997.

[22] R. P. Koomullil and B. K. Soni, “Generalized grid techniques in computational field simulation,” Numerical Grid Generation in Computational Field Simulations, M. Cross, P. R. Eiseman, J. Hauser, B. K. Soni, J. F. Thompson (eds). International Society for Grid Generation: Mississippi State, MS, 1998; 521–531.

[23] W. Strang, R. Tomaro, and M. Grismer, “The Defining Methods of Cobalt60: A Parallel, Implicit, Unstructured Euler/Navier-Stokes Flow Solver,” AIAA, 1999, Paper Number 99-0786.

[24] J. Wu, L Tang, E. A. Luke, X. L. Tong, and P. Cinnella, "Comprehensive Numerical Study of Jet-Flow Impingement over Flat Plates," J. Spacecraft Rockets, Vol. 39, No. 3, 2002, pp. 337–366.

[25] M. Aftosmis, M. Berger, and J. Melton, "Adaptation and Surface Modeling for Cartesian Mesh Methods," AIAA Paper 95-1725-CP, 1995.

[26] J. D. Anderson, Modern Compressible Flow with Historical Perspective, McGraw Hill Book Company, 1982.

[27] D. Lovely and R. Haimes, "Shock Detection from Computational Fluid Dynamics Results," AIAA Paper 99-3285, Presented at the 14th Computational Fluid Dynamics Conference, June 1999.

[28] Marcum, D. L., "Generation of Unstructured Grids for Viscous Flow Applications," AIAA Paper 1995-0212, 1995.

[29] Wu, J., Tang, L., Luke, E. A., Tong, X.-L., and Cinnella, P., "Comprehensive Numerical Study of Jet-Flow Impingement over Flat Plates," J. Spacecraft Rockets, Vol. 39, No. 3, 2002, pp. 337–366.

[30] Kang, S. W. and Dunn, M. G., "Theoretical and Measured Electron-Density Distributions for the RAM Vehicle at High Altitude," AIAA Paper 72-689, January 1972.

[31] Mani, M., Ladd, J. A., Cain, A. B., and Bush, R. H., "An Assessment of One- and Two-Equation Turbulence Models for Internal and External Flows," AIAA Paper 97-2010, 1997.

APPENDIX A

CLASSES

Node: This is a class and each object contains the following,

Index

Location vector

Sensor flag: This flag is used to mark nodes in the region of refinement

Input flow parameters: Density, velocity vector, pressure, etc

Calculated flow parameters: Pressure gradient, etc

Validity: If the node is still in the mesh. During refinement, all nodes are active. For other operations, nodes may be removed from the mesh. They are not deleted from the data structure but are just marked inactive.

Edge: This is a class and each object contains

Pointers to two node objects: The nodes that form the edge.

Edge refinement flag: This flag is used to tag edges in the region of refinement. Usually an edge is marked if its two nodes are marked.

Priority parameter: Based on refinement requirements, a value is assigned to this parameter based on certain criteria being satisfied. For example, for flows with shocks. It is based on the angle between the pressure gradient and the geometric edge vector.

Validity: If the edge is subdivided, it will no longer be on the mesh, then it's marked inactive.

Pointers to two edge objects: If the edge is undivided and active, these would be NULL. When the edge is subdivided, these pointers contain the newly created edges.

Pointer to a node object: If the edge is undivided and active, this would be NULL but when the edge is subdivided, this pointer contains the node created during the subdivision process.

Face: This is a class and each object contains

Number of edges: As a face can contain an arbitrary number of edges.

A vector of pointers to edge objects: The edges that form the face.

Number of nodes

A vector of pointers to node objects

Pointers to two element objects: The two elements that contain this face. Each interior face is contained exactly in two elements. For boundary faces, one of the two pointers is NULL, since only one element contains it.

Boundary flag: A negative value tells which boundary surface the face lies on and a zero means an interior face. This flag cannot have a positive value.

Validity: If the face is subdivided, it will no longer be in the mesh, then it's marked inactive.

Pointer to an edge object: If the face is subdivided, this contains the edge across which it was subdivided. If the face needs to be subdivided across an edge in a later refinement cycle, that edge is compared with this stored object, Refinement is possible only if they match.

Pointers to two face objects: If the face is undivided and active, these would be NULL. But, when the face is subdivided, these pointers contain the newly created faces.

Pointer to an edge object: If the face is undivided and active, this would be NULL but when the face is subdivided, this pointer contains the edge created during the subdivision process.

Element: This is a class and each object contains

Index

Level: The number of subdivisions that formed the element. When an element of level L is subdivided, the resulting elements are of level L+1.

Type: 1 if it is a tetrahedron, 2 for a triangular prism, 3 for a hexahedron, 4 for an arbitrary pyramid and 5 all other types.

Number of faces: As an element can have arbitrary number of faces.

A vector of pointers to face objects: The faces that for the element.

Number of edges

A vector of pointers to edge objects

Number of nodes

A vector of pointers to node objects

A vector of pointers to element objects: This is the list of neighbors. The power and preciseness of generalized girds is demonstrated by the fact that this information is never used during the refinement process.

Element refinement flag: This used to mark elements in the region of refinement. For example if more than half of its nodes are marked then the element is marked.

Validity: If the element is subdivided, it will no longer be on the mesh, then it's marked active.

APPENDIX B

FUNCTIONS

Edge:

Common node: Find the common node with another edge, return NULL if no common node.

Calculate the priority parameter: For shocks, average the pressure gradients at the nodes and find the dot product of it with the geometric edge vector.

Split

Face:

Order Edges: Order the edges of the face so that the list of edges forms a cycle.

Order for Division: Given an edge, the edge list is re-ordered with the given edge at the top of the list and the list of edges forming a cycle.

Is edge: Check if the edge is a member of the face

Common Edge: Find the common edge with another face, return NULL if no common edge.

Set Nodes: Using the edge list, find the list of nodes that form the face.

Split and Split check

Element:

Set Type: Based on the ideas described earlier, determine the type of the element using the number of faces, edges, nodes, and their connectivity information.

Set Edges: Using the face list, find the list of edges that form the face.

Set Nodes: Using the face list, find the list of nodes that form the face.

Order for Division: Based on the priority flag, order the edges.

Split and Centroid Split

Update: When a new element is created, set level, set the number of faces and update the containing element information of all its faces.

APPENDIX C

FORMAT OF THE MESH STATE FILE

```
<begin file>

NV NE NF NC     // num. of nodes, edge, face and elements [int]

(for every node)
  X Y Z       // postion [double]
  VAL  // refinement parameter [double]
      // value in the pix file will over write this
  STEP  // level of refinement [int]

(for every edge)
  V1 V2  // index of it's two nodes [int]
  SPLIT  // = 1 means the edge have been split. = 0 means otherwise [int]
  if SPLIT=1
    E1 E2 V1  // index of it's two child edges and one child node [int]
  end if

(for every face)
  NE BRY SPLIT  // number of edges, boundary flag (=0 if inside) [int]
  E1 E2 (upto) E[NE]  // index of it's edges [int]
  C1 C2  // index of it's containing elements (boundary faces will have a -1) [int]
  if SPLIT=1
    F1 F2 (upto) F[NE]  // index of it's child faces [int]
    E1 E2 (upto) E[NE]  // index of it's child edges [int]
    V1         // index of it's child node [int]
  end if

(for every element)
  NF SPLIT STEP  // number of faces, split flag and level of refinement [int]
  F1 F2 (upto) F[NF]  // index of it's faces [int]

<end file>
```