ANISOTROPIC TETRAHEDRAL MESH GENERATION

By

Rao V. Garimella

A Thesis Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Major Subject: Mechanical Engineering, Aerospace Engineering and Mechanics

Approved by the
Examining Committee:

_____
Dr. Mark S. Shephard, Thesis Adviser

_____
Dr. Joseph E. Flaherty, Member

_____
Dr. Robert L. Spilker, Member

_____
Dr. Kenneth E. Jansen, Member

Rensselaer Polytechnic Institute
Troy, New York

December 1998
(For graduation May 1999)

ANISOTROPIC TETRAHEDRAL MESH GENERATION

By

Rao V. Garimella

An Abstract of a Thesis Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Major Subject: Mechanical Engineering, Aerospace Engineering and Mechanics

The original of the complete thesis is on file
in the Rensselaer Polytechnic Institute Library

Examining Committee:

Dr. Mark S. Shephard, Thesis Adviser

Dr. Joseph E. Flaherty, Member

Dr. Robert L. Spilker, Member

Dr. Kenneth E. Jansen, Member

Rensselaer Polytechnic Institute
Troy, New York

December 1998
(For graduation May 1999)

CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENT

Many people have helped me earn my doctoral degree and more importantly, develop the skills and knowledge required to aspire to do research. I take this opportunity to acknowledge their contributions.

I would like to thank my advisor, Dr. Mark S. Shephard, who took a chance and gave me an opportunity to join his research group although I had no experience in mesh generation or geometric modeling. Mark has taught me the importance of focus and quality in research. He has also taught me by example that hard work and a strong work ethic are essential for success in any field. I would like to thank him for his support and guidance through my stay at SCOREC.

I would like to thank Dr. Kenneth E. Jansen for his invaluable technical advice. Ken helped take this research to a new level of excellence by many hours of insightful advice on the desired characteristics of anisotropic meshes for CFD. Ken also reminded me that main purpose of mesh generation is to be able perform high fidelity finite element simulations.

I thank both the Investment Casting Cooperative Arrangement members and Simmetrix, Inc. for providing financial support for this thesis work. In particular, I would like to thank Dr. Bruce E. Webster for his continued faith in me to deliver a quality mesh generator for his work.

All my colleagues and friends at SCOREC have provided me with a wonderful research environment and I owe my knowledge of mesh generation and geometric modeling to their patient tutoring, answering my endless questions, and long, interesting discussions. In particular, I would like to thank my advisor Mark Shephard and colleagues Marcel Georges, Pascal Frey, Ravi Ramamoorthy, Saikat Dey and Hugues de Cougny and Kaan Karamete for educating me about mesh generation. Also, Kaan Karamete's development of edge recovery procedures came at a timely moment freeing me up to concentrate on other pressing issues in my thesis.

On a personal note, I owe much of what I have achieved to the love and support of my wonderful parents who have endeavored to provide me with everything I

needed and always propelled me to set high standards for myself. I thank my great-uncle, P. Rama Rao, my sister, Aruna Prakash and brother-in-law, N. Prakash, without whose help I would not have been able to pursue my studies in the U.S.A. Also, I owe a great deal to my uncle, Chalam Garimella, who has been a guardian, mentor and above all, a wonderful friend. His friendship and affection have pulled me through some of the more trying times during completion of my degree. In addition, Sita and Chalam Garimella, Lakshmi and Sastry Sreepada, Lalita and Sanjeev Hirve, and Raji and Kumaraswamy Dikshitar, have all provided me a home away from home and I am grateful for their care and affection.

Finally, my humble gratitude to the almighty for giving me a wonderful life full of opportunities and I pray that I make the best use of it.

ABSTRACT

Many physical problems exhibit strong gradients in specific directions compared to other directions. To successfully perform finite element analysis and obtain accurate solutions for such problems, elements in the finite element mesh must be small enough in these directions. Anisotropic meshes with small dimensions in the directions of strong gradients and large sizes along others can significantly reduce solution costs. This research focuses on two classes of problems requiring generation of such anisotropic tetrahedral meshes.

Viscous flow problems exhibit boundary layers and free shear layers in which the solution gradients, normal and tangential to the flow, differ by orders of magnitude. The Generalized Advancing Layers Method is presented here as a method of generating meshes suitable for capturing such flows. The method is designed to reliably generate anisotropic elements in boundary layers for arbitrarily complex non-manifold domains. The boundary layer mesh is created by tetrahedronization of prismatic, transition and blend polyhedra constructed on top of an initial surface mesh. The method includes several new technical advances allowing it to mesh complex geometric domains that cannot be handled by other techniques and is currently being used for simulations in the automotive industry.

Anisotropic meshes are also desirable in problems with a strongly non-linear solution across thin sections of the analysis domain. A procedure has been developed to transform an isotropic mesh with insufficient refinement through thin sections into one with a user defined number of elements through such sections. The method automatically identifies deficient portions of the mesh and anisotropically refines it using local mesh modification tools.

The two mesh generators form components of an overall framework for adaptive analysis in which anisotropic mesh generation and adaptation decrease the computational cost of converging to solutions of the desired accuracy for simulations in general geometric domains.

CHAPTER 1
INTRODUCTION

Simulation of systems is playing an ever increasing role in the engineering design cycle. Increasing use of simulations to test designs is propelled primarily by the high cost of performing tests on physical prototypes and the need to refine design ideas rapidly. The availability of sophisticated numerical techniques and increased accessibility to high performance computing are central to the ease with which engineers can perform simulations to evaluate design ideas. The availability of these resources also feeds the desire to do more extensive analysis of complex coupled multi-physics, multi-scale systems with fewer idealizations. Thus the envelope of the current design and analysis technology is constantly being pushed outward to keep pace with and even outgrow present technological capabilities.

Finite element analysis methods have played a major role in the development of simulation as a viable tool in many engineering fields such as stress analysis, simulation of chemical processes, flow of fluids, electromagnetics etc. The availability of reliable automatic finite element mesh generators is a critical component in the ability of an analyst to harness the power of the finite element method. Automatic mesh generators must be able to mesh arbitrarily complex non-manifold[1] geometric domains derived directly from CAD systems. The important characteristics of a mesh is good mesh quality, proper mesh gradation and proper element sizes which will enable the analyst to capture the desired features of the solution within the available resources. The mesh should be sufficiently refined in regions where solution exhibits sharp gradients without over refinement or propagation of the refinement to parts of the domain where the solution is not changing rapidly. While the optimal distribution of points may best be achieved by adaptive mesh generation based on error estimation, a good point distribution obtained *a priori* by a mesh generator and careful choice of mesh control specification can significantly reduce the number of adaptive analysis loops required for the solution to converge.

[1]Simply put, non-manifold models consist of general combinations of solids, surfaces and wires. For a more rigorous definition see Chapter 3 and ref. [43].

Many physical problems exhibit relatively strong gradients in certain local directions compared to the other directions. Some examples of such situations are thermal and fluid boundary layers, and nonlinear solutions in domains with very thin sections. A certain minimum element size along these directions is necessary to capture the solution in these regions. However, isotropic refinement of the mesh in these parts of the domain leads to prohibitively large meshes and a wastage of degrees of freedom in directions which do not need such fine resolution. Anisotropic meshes with small element sizes in the directions of strong gradients and large sizes along the other directions leads to significant savings in solution costs (often up to several orders of magnitude).

This research focuses on two classes of problems requiring anisotropic refinement:

1. Applications requiring the resolution of boundary and free shear layers such as viscous flow simulations, and

2. Applications requiring resolution of strongly nonlinear variation of field variables in geometrically thin domains.

High Reynolds number fluid flow simulations have boundary layers at the wall and also free shear layers not attached to any model boundary. The relative rates at which the solution variables change in boundary and shear layers, normal and tangential to the flow, may differ by many orders of magnitude in such problems. Use of properly aligned anisotropic meshes in these parts of the flow results in large reductions in the total number of elements.

A generalization of the popular advancing layers method [11, 31, 34, 40, 54] is presented here as the method for generating boundary layer meshes. The method is designed to efficiently and reliably generate good quality anisotropic elements near the boundary layer surfaces for arbitrarily complex non-manifold domains starting from a surface mesh. The method has several improvements over the previous advancing layers techniques reviewed in Chapter 2. It is demonstrated that the common strategy of inflating the surface mesh *as is* to form the boundary layer leads to invalid meshes for some non-manifold models and poor quality elements at

sharp corners in 2-manifold models. Various procedures are described to make the boundary layer elements valid and to ensure that the mesh is not self-intersecting. The improvements incorporated into the method has enabled it to be used successfully to generate large boundary layer meshes for real industrial models that are geometrically very complex.

Another class of problems needing anisotropic refinement of meshes is one in which the solution is strongly non-linear across thin sections of a domain relative to other directions. Use of one linear element across such thin sections is unacceptable for such problems. The simplest and most problematic deficiency of such a mesh is in flow simulations where a "no slip" boundary condition is enforced on the walls of a section spanned by a single element. An analysis with linear finite elements using this mesh incorrectly predicts no flow through these sections. The second part of the research presented here describes a method that addresses this problem. The method is designed to transform an isotropic mesh with insufficient refinement through thin sections into one with a user defined number of elements through such portions of the domain. The procedure uses local mesh modification procedures to effect the refinement. The method is completely automatic, identifying deficient portions of the mesh without any user input. It is designed to handle arbitrarily complex geometric domains. It functions in conjunction with isotropic automatic mesh generators [13, 17, 65] and can use input from any mesh generator capable of providing the necessary information about the initial mesh [5].

The two procedures described in this research form parts of the overall framework for adaptive analysis in which anisotropic mesh generation and adaptation serves to decrease the computational cost of converging to solutions for complex problems in general geometric domains (Figure 1.1).

The rest of this thesis is organized in the following manner. A review of the previous efforts in anisotropic mesh generation is presented in Chapter 2. Definitions and notations used in the following chapters are described in Chapter 3. Chapter 4 discusses the motivation for specialized boundary layer meshing techniques and presents an overview of the Generalized Advancing Layers method used here. Chapter 5 discusses at length the issues in point placement for boundary

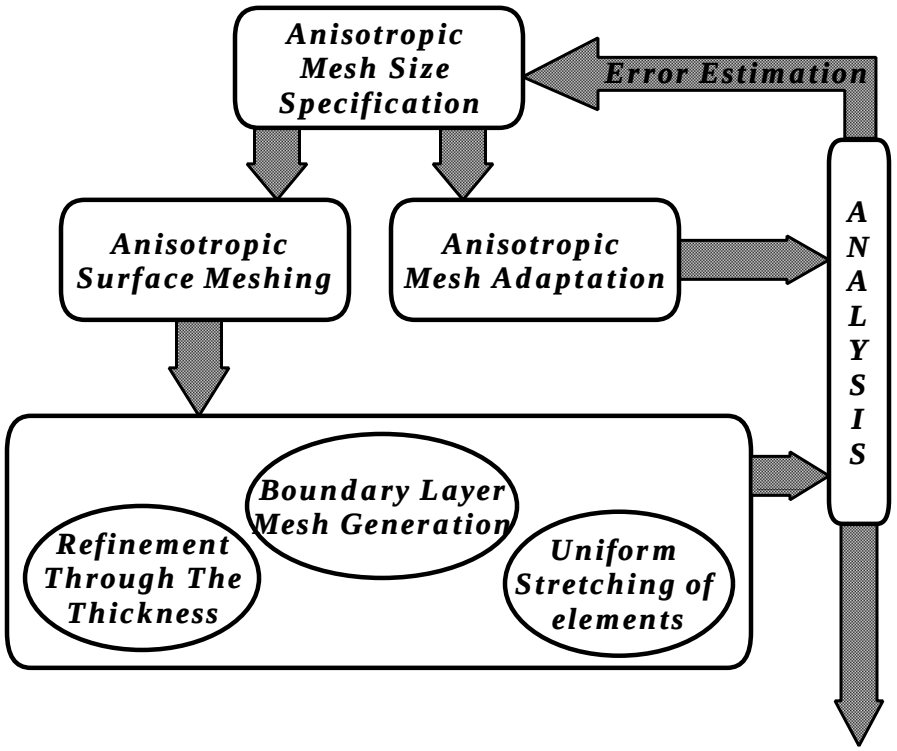


Figure 1.1: Framework for anisotropic mesh generation and refinement.

layer meshing of arbitrarily complex non-manifold geometric domains. Chapter 6 describes techniques to ensure that the boundary layer elements generated will be valid and the creation of boundary layer elements is presented in Chapter 7. Chapter 8 discusses the method used to guarantee that the boundary layer mesh is not self intersecting. The discussion of boundary layer meshing concludes with results and discussion in Chapter 9.

The need for anisotropic refinement of meshes for thin section domains and the methodology for achieving it are outlined in Chapter 10. Chapter 11 discusses the procedure for automatically identifying deficient portions of the domain for general geometric models. Chapter 12 describes the creation of multiple layers of elements through the thickness. This chapter discusses refinement of the deficient mesh by edge splitting and elimination of short paths by edge swapping. Uniform refinement of the mesh to eliminate any remaining deficiencies in the mesh is also discussed in Chapter 12. Matching of the mesh on opposite model faces and other pre- and post-

processing steps to improve the quality of the final mesh is discussed in Chapter 13. Results and discussion follow in Chapter 14.

Closing remarks and future work for a complete anisotropic mesh generation capability are presented in Chapter 15.

CHAPTER 2
SURVEY OF PREVIOUS EFFORTS ON ANISOTROPIC MESH GENERATION

Anisotropic meshes for capturing boundary layers in viscous flows have been generated by three techniques:

1. Direct creation using anisotropic mesh control information often combined with transformation of meshing space

2. Modification of an initial isotropic mesh by node reposition and/or local mesh modifications

3. Creation of the anisotropic mesh by a special method followed by the generation of an isotropic mesh in the rest of the domain.

Direct generation of unstructured anisotropic meshes has been attempted with both Delaunay and Advancing Front methods.

The Delaunay method ([2, 12, 22–25, 29, 30, 71] are just a few of the extensive list of references on this subject) for generating simplex meshes in n dimensions starts with a discretization of the boundary of the domain. To mesh the domain, an extremely coarse mesh of a convex polyhedron consisting a few simplices is constructed such that it completely encloses the boundary discretization. The points of the boundary mesh are then inserted into this coarse mesh according to the Delaunay criterion. The Delaunay criterion dictates that no vertex in the mesh can be contained in the circumsphere (circumcircle in 2D) of a simplex not connected to the vertex. Therefore, when a new point is to be inserted, the elements whose circumsphere contains the new point are deleted to form a cavity. The new point must be visible from every point on the boundary of the cavity. Then the new point is connected to the boundary of the cavity to create a new mesh containing the new vertex. The resulting mesh satisfies the Delaunay criterion. The distribution of the points is chosen such that the edges in the mesh have a satisfactory length

according to the mesh size specification. The simple Delaunay algorithm guarantees that all vertices of the boundary are present in the mesh. However, it is not guaranteed that all the connecting boundary entities between the boundary vertices are represented in the mesh. The Constrained Delaunay algorithm [23] uses local mesh modifications to recover these missing edges to preserve boundary integrity.

By its nature, satisfaction of the Delaunay criterion tends to produce isotropic triangulations. In fact, even if the point distribution is anisotropic, the Delaunay method tries to create low aspect ratio elements resulting in elements of widely varying sizes [48]. Therefore, various researchers have proposed methods to transform the meshing space such that an isotropic mesh created by the Delaunay method in the transformed space is anisotropic in the real space.

Mavripilis [48] has described a method for anisotropic adaptation of triangular meshes constructing a metric based on two independent stretch vectors at each point. Using this metric the local space is mapped to a control surface in a transformed higher dimension space in which a Delaunay triangulation is performed. The triangulation so generated is isotropic in the mapped space but stretched when mapped back to the real space. The control surface dimensionality is reduced by assuming local planarity.

M. G. Vallet, F. Hecht and B. Mantel [70] have proposed a similar idea for the initial mesh generation process as well as adaptation. Researchers P. L. George, H. Borouchaki, F. Hecht et.al. have generalized the ideas of generating anisotropic mesh generation by the Delaunay method using metric specifications in recent works [7, 8, 23]. M. J. Castro-Diaz, F. Hecht and B. Mohammadi, in their work [9, 10], have added to the existing ideas of anisotropic grid adaption by recognizing that, for viscous flow simulations, it is desirable to have the near-wall mesh as orthogonal to the wall as possible and to maintain a certain minimum distance of the first node from the wall nearly. The metrics used for generation of the anisotropic mesh are modified near the wall to account for these factors.

Borouchaki, George et.al. [7] have presented a generalization of the Delaunay method that encompasses isotropic and anisotropic mesh creation. In its basic form, the method bears close resemblance to the classic Constrained Delaunay algorithm.

However, the method has the ability to use a modified Delaunay criterion for point insertion. Consider a domain Ω discretized by an initial mesh in which one or more metrics are specified at each vertex. Each metric is a positive definite symmetric tensor of as many dimensions as the dimension of the domain being meshed. In two dimensions, the metric is written as

$$\mathcal{M}(X) = \begin{bmatrix} a(X) & b(X) \\ b(X) & c(X) \end{bmatrix} \tag{2.1}$$

where X is any point in Ω, $a(X), c(X) > 0$ and $a(X)c(X) - (b(X))^2 > 0$.

When the metric values from the vertices are interpolated over the entire domain, a Riemannian space is defined by the pair $(\Omega, \mathcal{M}(X))$. In the case where the metrics are identical over all points over the domain, the Riemannian mapping simplifies to a Euclidean mapping (the transformations in this case are only scaling, translation and rotation; skewing is not allowed).

The length of a line segment from $\overline{PQ} = (P + t\overline{PQ})_{0 \leq t \leq 1}$ in Ω is given by

$$l(P,Q) = \int_0^1 \sqrt{a(t)x_1^2 + 2b(t)x_1x_2 + c(t)x_2^2}dt \tag{2.2}$$

where

$$\overline{PQ} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \text{ and } \mathcal{M}(P + t\overline{PQ}) = \begin{bmatrix} a(t) & b(t) \\ b(t) & c(t) \end{bmatrix} \tag{2.3}$$

If the space is Euclidean, this simplifies to

$$l(P,Q) = \sqrt{ax_1^2 + 2bx_1x_2 + cx_2^2} \tag{2.4}$$

With the above relationships for transforming lengths between the real and the mapped space, a modified Delaunay criterion may be devised. If the metric is a true Riemannian metric, it is very difficult to define the equivalent of a circumcircle in the transformed space since the metric varies continuously from point to point. Therefore, the problem is simplified by assuming that the space is Euclidean in the local neighborhood of the point. Given a triangle K with points P_1, P_2 and P_3 with circumcenter O and a point to be inserted P into the triangulation, the point violates the Delaunay criterion with respect to the triangle in the transformed space if $l^{\mathcal{M}}(O,P) < l^{\mathcal{M}}(O,P_i), i = 1,2,3$. It is possible to refine this criterion by assuming that each vertex of the triangle and the point to be inserted have different locally Euclidean metrics associated with them.

In their work, the authors also discuss details of smoothly interpolating the metrics between endpoints of the segments. Given a field of metrics, relationships for transforming length measures from one space to another, a generalized Delaunay kernel and a method for interpolating the metrics smoothly, it is now possible to generate a mesh with edge lengths of unity in the Riemannian space so that the mesh will have the desired sizes in the real space. If the metric is the identity matrix, I, then an isotropic mesh with edges of unit length is generated. If the metric is hI where h is the size specification defined on the background or initial mesh, then an isotropic mesh satisfying this size specification may be generated. On the other hand, if the metric is more general then an anisotropic mesh is created.

The authors derive the metric specification for generating the anisotropic mesh from solution variables in an adaptive analysis. Since one may require more than one solution variable to be taken into account, the authors also provide mechanisms for combining multiple metrics.

This method has been shown to work well for generating anisotropic meshes in two dimensions based on solutions from an existing mesh [8–10, 70]. The extension to three dimensions seems natural and is proposed in the papers but no results are presented. The method possesses the following complexities:

- The anisotropic mesh must be regenerated from scratch at every adaptive step. This is due to the fact that it is easy to refine a mesh based on Delaunay

criterion but it is not straightforward to coarsen an existing mesh while maintaining the Delaunay property. One recent work has been published which takes a step in this direction for two dimensions [77]. Therefore, to avoid the cost of carrying the refinement in uninteresting portions of the domain, the mesh is regenerated at each step. The regeneration of the mesh can potentially become an expensive step in itself for complex domains in three dimensions.

- The quality (large angles) of elements degrades rapidly with increasing anisotropy in this method. It is expected that this characteristic is more pronounced in three dimensions where the large dihedral angles will degrade quickly with increasing anisotropy of the Riemannian mapping. Therefore the method is not very well suited for generation of meshes capable of capturing strong gradients in the boundary layer, where desired aspect ratios of 1000 to 10000 are quite common. It must be noted, that in a recent work [10], this problem is indirectly addressed in two dimensions by using a special near wall metric. The metric is designed to create edges orthogonal to the wall and to maintain a predefined offset of the first layer of nodes from the wall. This characteristic improves element quality implicitly.

Like the Delaunay method, the advancing front method [28, 42, 50] starts with a mesh of the model boundaries. The boundary triangles are incorporated into a front. The mesh is then grown from the boundary inwards by forming elements using each of the front faces in turn. To form an element using a front face, one or more candidate locations are chosen in the domain for the fourth vertex of the tetrahedron. An obvious choice for the location of the fourth point is along the normal to the front face originating from its centroid. The distance of the candidate point from the face is designed to generate a well shaped isotropic element respecting the mesh size specification in the domain. The mesh size distribution may be specified in a number of ways, common ones being using a user generated background mesh [53] and a tree structure [13, 17, 64]. For each candidate location considered, a number of checks must be performed to ensure that the mesh will be valid. The most important of these is the check to ensure that the new element will not intersect any other front entities. Since the front can be quite large, it is necessary to localize the search for

intersections using a search tree such as an octree [13, 39, 56, 59] or an alternating digital tree [6]. Once a candidate location passes all the checks, an element is formed using the front face and a vertex at the chosen location. The front face is removed from the front and the new faces of the element are added to the front. Element creation continues until the front is empty indicating that the domain is meshed. The advancing front, like the Delaunay method, can produce good quality isotropic meshes quite efficiently. In addition, it has the advantage of concentrating the best shaped elements near the boundaries of the domain.

Hassan, Probert, Morgan and Peraire [27] have used a modified advancing front method to generate anisotropic meshes where a layer of elements is generated from a front using isotropic criteria and then the layer compressed as a whole to the desired thickness by node repositioning. Points are constrained to move along element sides until they lie at a user specified offset from the model boundary. The user can specify the number of such layers desired. The thickness of subsequent layers increases by a geometric progression such that the final layer thickness is half of the isotropic mesh size. Once the special elements are generated, the usual isotropic advancing front method is used to fill up the rest of the domain. While this method worked well in 2D, it is prone to problems in 3D [26]. One difficulty in the method stems from ambiguities in the direction of movement of the nodes to achieve good quality of elements. Also, isotropic advancing front methods typically generate more points for the upper surface of the first layer than there are on the surface triangulation. Compression of the layer then leads to some points coming too close to each other in the tangential direction to the model boundary.

Most of the work in generating meshes for viscous flow simulations has been in the direction of generating an anisotropic mesh next to surfaces where a boundary layer is expected and then filling the rest of the domain by an isotropic mesh generator. A popular set of such methods are the advancing layers and advancing normals methods. The basic strategy in both methods is to use a special method to generate the anisotropic layers of elements next to model boundaries and then hand the task of meshing the rest of the domain to one of the common isotropic mesh generation methods.

Kallinderis et.al. [31–34, 37] developed a hybrid prismatic/tetrahedral mesh generators for viscous flow simulations by enclosing the body around which the flow is to be simulated with layers of prisms and then filling the rest of the domain using a combination of octree and advancing front methods. The height of the prisms increases away from the wall according to user specifications. The procedure incorporates an algorithm to ensure that the interior nodes of the prisms are "visible" from all the relevant faces of the previous layer [34]. "Visibility" of a node from a face is a necessary condition for the face and the node to form a positive volume element. Included in this method is a procedure to automatically recede and smoothly grade layers in confined regions of the model based on ray tracing methods although no explicit check for interference of boundary layer prisms is described [37]. Sharov and Nakahashi [60] have described a similar method with some modifications for generating better elements and for generating all tetrahedra. They use a Delaunay method for generating the interior tetrahedra. The use of prisms to capture the boundary layers leads to fewer elements in the mesh. The method is capable of handling 2-manifold geometries [32] but may create poor quality meshes at sharp corners (where boundary layer nodes are not "visible" to mesh faces). The method also cannot handle non-manifold geometries.

Löhner [40] describes a similar method for by combining a semistructured grid consisting of layers of anisotropic tetrahedronized prisms grown on some model boundaries with an unstructured isotropic mesh in the rest of the domain. The method starts from a mesh of the surface, grows the tetrahedronized prisms on mesh faces on selected boundaries with nodes placed along directions derived from surface normals and then fills the rest of the domain with isotropic elements created by an advancing front mesh generator. A Laplacian smoothing procedure is used to smooth the directions of node placement. The procedure detects poorly shaped, improperly sized and intersecting elements, and deletes them from the mesh. A search tree is used to speed up the detecting of intersecting elements. The possibility of impermissible diagonal combinations in prism tetrahedronization is recognized and an iterative procedure to correct such configurations is introduced. (A recent paper by Löhner [41], however, advocates the use of anisotropic refinement of an

isotropic mesh using the Delaunay criterion to generate boundary layer meshes. Inability of the advancing layers method to mesh complex models is cited as the reason for switching to the new method.)

Pirzadeh [54] described a similar approach called the Advancing Layers Method (ALM) for the generation of anisotropic meshes for viscous flow calculations. The significant features of this work are:

1. Introduction of prism templates.
2. A non-iterative procedure for obtaining valid diagonals for the prisms.
3. An iterative procedure for obtaining valid directions for placement of points based on maximization of the minimum angle the direction makes with the faces connected to the surface vertex.
4. A procedure for avoiding interference between layers based on a virtual springs connecting front vertices.

Connell and Braaten [11] have described an implementation of the advancing layers procedure with many enhancements to deal with general situations. They advocate the use of surface normals for node placement, since they assert that it gives smoother distribution of nodes in the standard advancing layers method. The paper details an algorithm to ensure that all prisms have a valid set of diagonals. The goal of the procedure is to ensure that no prism has all diagonals in the same direction. In this method, each vertex on the model boundary is visited and the edges coming into the vertex are assigned a direction pointing into the vertex if they do not already have one assigned. The direction assigned to the edge uniquely determines the direction of the diagonal of the prism face associated with the edge. It can be shown [11] that it is impossible to assign a cyclic set of directions to the edges of any triangle using this method. Also, discussed is a technique for grading the boundary layer mesh to avoid exposing highly stretched faces to the isotropic mesh generator when elements are deleted. If it is seen that the faces of a prism will be exposed to the isotropic mesh generator at any edge, the number of layers at that edge are reduced to zero. A recursive procedure then ensures that the number nodes at neighboring

vertices are also trimmed so that no two adjacent surface mesh vertices have more than a one layer difference. Then the stretched faces are sealed off using diagonal faces which are isotropic (See Chapter 7 for a discussion of transition elements that are a generalization of this concept). The authors check explicitly for interference between the prisms based on the advancing front method reported to be $O(n^2)$ with a proposal to reduce the time using a search tree. Their technique for finding valid directions at model edges and vertices, however, is not general and is not guaranteed to always work. Connell and Braaten's work discusses many of the fundamental issues with viscous flow simulations and mesh generation for these problems using the advancing layers methods. They discuss issues of exposing stretched faces to the advancing front mesh generator, devising assurance algorithms for valid prism tetrahedronization, the interference of layers, varying thickness boundary layers and resolution of wakes. They also demonstrate the capabilities of the mesh generator on a number of complex configurations.

Hassan et.al. [26] have also devised another variation of the advancing front method to circumvent the difficulties with their earlier work. In this method, a new vertex is generated with respect to each front vertex and all the connected front faces are combined with the new vertex to form elements of the next layer. The new vertex is placed at a user specified distance from the previous vertex along a direction normal to the surface at that vertex. At model edges and vertices, a special procedure is used to prevent invalid elements from being created. This procedure tries to compute a direction which is "visible" to all mesh faces using the base mesh vertex. This procedure may not always succeed in such situations and when it does, it may produce barely valid elements. An important feature in the procedures of Hassan et.al. is the ability to merge two directions if they intersect or two points in the layer being generated come too close to each other. This allows the procedure to eliminate points in the upper layers on concave boundaries. An equivalent procedure for adding additional directions when the convexity of the surface is too high is absent and is evident in the large elements created in some parts of the example meshes shown. Once the anisotropic elements are generated, the isotropic advancing front mesh generator takes over and fills the rest of the

domain. The major drawback of this, and other methods based on direct variations of the advancing front methods, is that they must check for intersections for every anisotropic element created.

Marcum and Weatherill [45–47] have described an approach for unstructured grid generation for viscous flows using iterative point insertion followed by local reconnection subject to a quality criteria. The point distribution for the anisotropic mesh is generated along "normals" according to user specifications or error estimates at model boundaries and stream surface based wake surfaces. The isotropic point distribution in the far field is generated using a standard advancing front method. The quality measure used is a Delaunay in-sphere criterion followed by a min-max criterion. The most interesting aspect of this work is that they account for sharp "discontinuities" at edges and vertices and generate points along additional directions in such cases. "Discontinuities" are identified based on the deviation between the average normal at a mesh vertex and its deviation from the individual mesh face normals. When this deviation exceeds a certain angle tolerance, an additional direction for point placement is created that is the weighted mean of the average normal and the face normals. This ensures that elements are not very poorly shaped near sharp corners. This has some parallels to the more general idea of multiple growth curves presented later in this thesis. Once the anisotropic elements are created, the same procedure is used to do isotropic point insertion and local reconnection in the rest of the domain.

Marchant and Weatherill [44] discuss the creation of the boundary layer mesh by the advancing normals method and using the outer envelope of the boundary layer mesh as the starting point for a constrained Delaunay type procedure. The problem of interference between layers is addressed by terminating insertion of points along a certain direction if they are closer to another surface than the originating surface. The issue of boundary layers interacting with adjacent model faces with no boundary layer is dealt with by tapering off the boundary layers at such interfaces.

The research described herein is a generalization of the advancing layers method mentioned above combined with an isotropic mesh generator based on a combination of advancing front and delaunay methods [13, 17].

CHAPTER 3
DEFINITIONS AND NOTATION

In this chapter definitions and notations of quantities used in the thesis are introduced. The notation given below expresses mesh and model entity relationships in a concise form [5]. Other definitions are introduced in the relevant chapters as necessary.

3.1 Notations

3.1.1 Set notation

$\{\}$	Unordered set
$\lfloor\rfloor$	Ordered set
$[]$	Cyclically ordered set
$\langle\rangle$	Set in which ordering of elements is unspecified and may be unordered, ordered or cyclic

3.1.2 Geometric model notations

G	Geometric model
G_i^d	i^{th} geometric model entity of order d ($d = 0, 1, 2, 3$ for vertices, edges, faces and regions respectively)
$(g_i^d)_{,j}$	j^{th} use of i^{th} geometric model entity of order d
∂G_i^d	Boundary of model entity G_i^d
$\overline{G_i^d}$	Closure of G_i^d, $G_i^d \cup \partial G_i^d$

3.1.3 Mesh notations

M	Mesh or discretization of the geometric model
M_i^d	i^{th} mesh entity of order d ($d = 0, 1, 2, 3$ for vertices, edges, faces and regions respectively)
$(m_i^d)_{,j}$	j^{th} use of i^{th} mesh entity of order d

∂M_i^d	Boundary of entity M_i^d
$\overline{M_i^d}$	Closure of M_i^d, $M_i^d \cup \partial M_i^d$
$\sqsubset$	Classification of M_i^d on G_j^D which is the unique association of M_i^d with G_j^D if M_i^d forms all or part of the discretization of G_j^D. Classification of M_i^d on G_j^D is written notationally as $M_i^d \sqsubset G_j^D$. It follows that $d \leq D$.

3.1.4 Adjacencies

$\varphi^{d_1}\langle T_i^d \rangle$	The set of entities in model T of dimension d that are adjacent to an entity or set of entities ϕ of dimension d_1
$(T_1^{d_1}, T_2^{d_2}, \cdots, T_n^{d_n})\langle T_i^d \rangle$	The set of entities of dimension d that are adjacent to the entities $T_1^{d_1}, T_2^{d_2}, \cdots, T_n^{d_n}$

3.2 Definitions

The basic input for any automatic mesh generator is a properly defined geometric model and a set of meshing attributes prescribed on the model. All models are expected to contain a definition of their topology, and a definition of the geometry underlying the topological entities (points, curves and surfaces).

3.2.1 Geometric model definitions and concepts (Also see [43, 51])

Geometric models may, in general, be *2-manifold* or *non-manifold*. Informally, non-manifold models are models which are general combinations of solids, surfaces and wires. A more formal definition is given below. The difference between 2-manifold and non-manifold models is illustrated in Figure 3.1.

Definition 3.1 *2-manifold models are geometric models in which the local neighborhood of every point on the model boundary is topologically equivalent or homeomorphic*[2] *to a disk [43, 72–74].*

[2]Intuitively, this means the local neighborhood of every point on the boundary may be transformed into a disk without any cutting, tearing or otherwise making points that were separate, coincident in the new form. See [49] for a discussion of topology and homeomorphism.

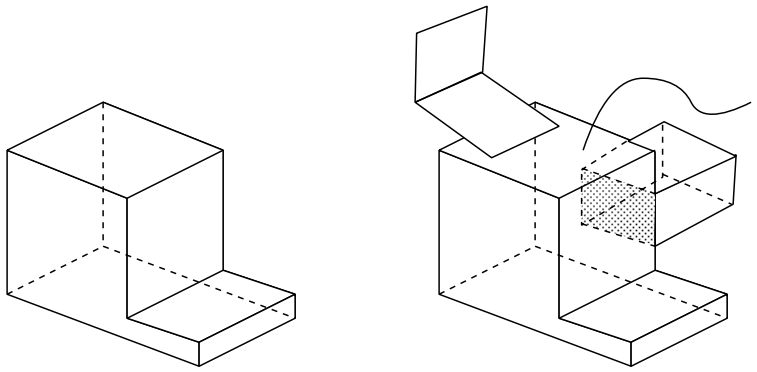


Figure 3.1: (a) 2-manifold model. (b) non-manifold model.

Definition 3.2 *All models that are not 2-manifold are non-manifold.*

Geometric models may have two types of non-manifold faces - embedded faces and interfaces (See Figure 3.2).

Definition 3.3 *Interface faces in geometric models are faces that partly bound two different model regions, one on each side.*

Definition 3.4 *Embedded faces in geometric models are faces that are connected to the same region on both sides.*

The data structure used to represent the model in this work is based on the *Radial Edge Data Structure* developed by Kevin Weiler [72–74]. The Radial edge data structure presents the idea of *uses* to represent how topological entities are used by others in a non-manifold model. Every face in the model has two face uses, one on each side of the face. An edge carries as many pairs of uses as there are pairs of face uses coming into it. Each edge use has an *edge use mate*. A vertex carries as many uses as there are edge uses coming into it and each vertex use has one *vertex use mate*. This is illustrated in Figure 3.3 (adapted from [72]).

The radial edge data structure is more detailed than the minimum amount of information required to represent many common types of non-manifold model. The representation is therefore reduced by fusing edge use mates together to form a single "edge use" connected to two face uses. Similarly, vertex uses are condensed so

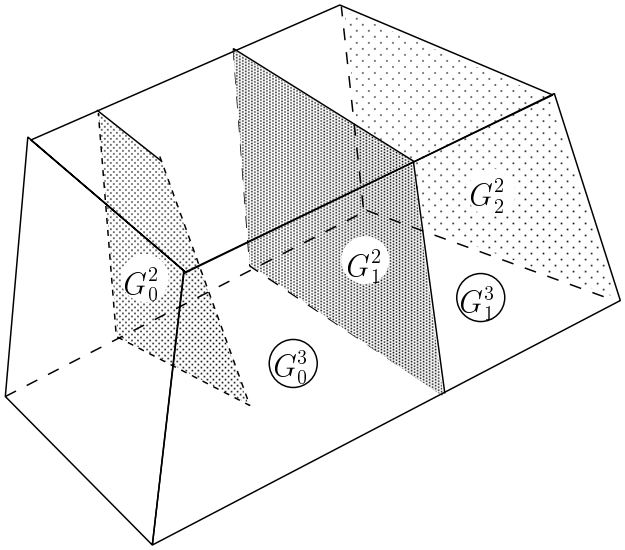


Figure 3.2: Model face types - G_0^2 is *embedded face*, G_1^2 is an *interface* and G_2^2 is a *2-manifold face*.

that the minimum number of uses are present at any vertex. Such a data structure is referred to as the *Minimal Use Data Structure* [4]. Conceptually, the minimal use data structure builds a representation of the non-manifold topology such that the connected face uses locally form a 2-manifold at any point. For example, every edge use in a minimal use is connected to two or no faces uses. This is even true for embedded faces in the model. For example, a rectangular face completely embedded in a region, has two faces with common edge and vertex uses (like a pillow case). Similarly, the use topology at any vertex use can always be represented locally as a 2-dimensional disk. Lastly, by its very nature the use topology at any face use is always a 2-dimensional disk.

Definition 3.5 *Given an entity use,* $(g_i^d)_{,j}$, $d = 0, 1, 2$, *the collection of its connected face uses,* $\{(g_k^2)_{,m} \mid (g_i^d)_{,j} \subset \partial((g_k^2)_{,m})\}$ *in a minimal use representation of a non-manifold model is called a* ***manifold****.*

At any point on a model face there are always two manifolds of face uses. One, both or none of these manifolds may partly or completely bound a model region. Points on edges and vertices may have multiple face use manifolds connected to them. As with faces, each of these manifolds may or may not form part of the

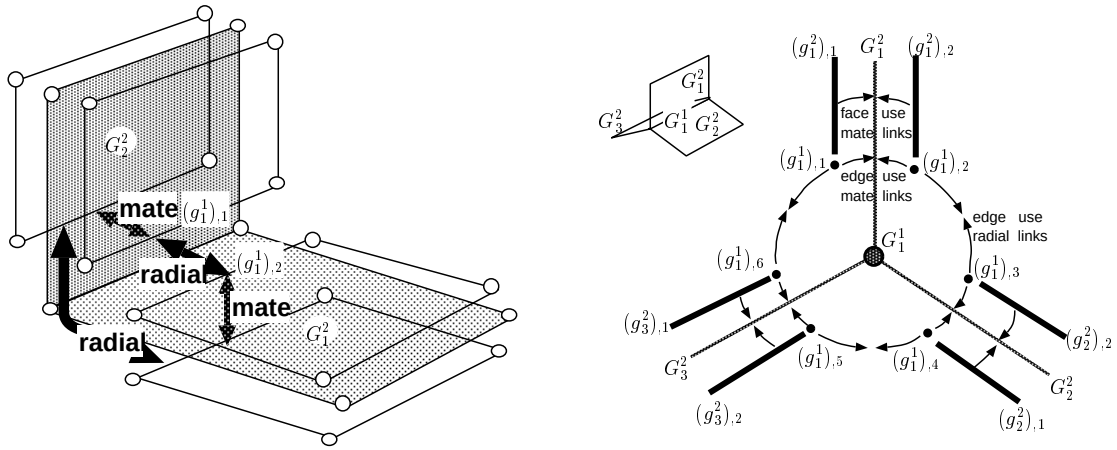


Figure 3.3: Radial edge representation of a non-manifold boundary (Adapted from [72]).

boundary of model region. *At any point it is not possible to travel from one of its mesh manifold to another without penetrating a boundary model entity.* The concept of a face use manifold is similar to the idea of a *separation surface* defined by Weiler [72]. Weiler describes a separation surface as "a complete surface formed by the juncture of faces around a vertex that effectively separates the space immediately around the vertex into two half-spaces, distinguishable from each other because the surfaces are orientable." Separation surfaces may be made up of one or more model faces as long as they form a continuous surface at the vertex. One or more separation surfaces may exist at any vertex. 2-manifold models are characterized by the presence of only one model region and only one manifold at each point on the model boundary connected to the region.

With the help of the minimal use representation and its collection of face uses into manifolds, dealing with a non-manifold boundary becomes equivalent to dealing with a set of 2-manifold boundaries. Figure 3.4 shows the minimal use representation equivalent of the non-manifold situations shown in Figure 3.3. For geometric modelers using a purely 2-manifold representations, non-manifold models may still be built up in a non-manifold data structure from multiple 2-manifold components if the appropriate additional information is specified [63, 67].

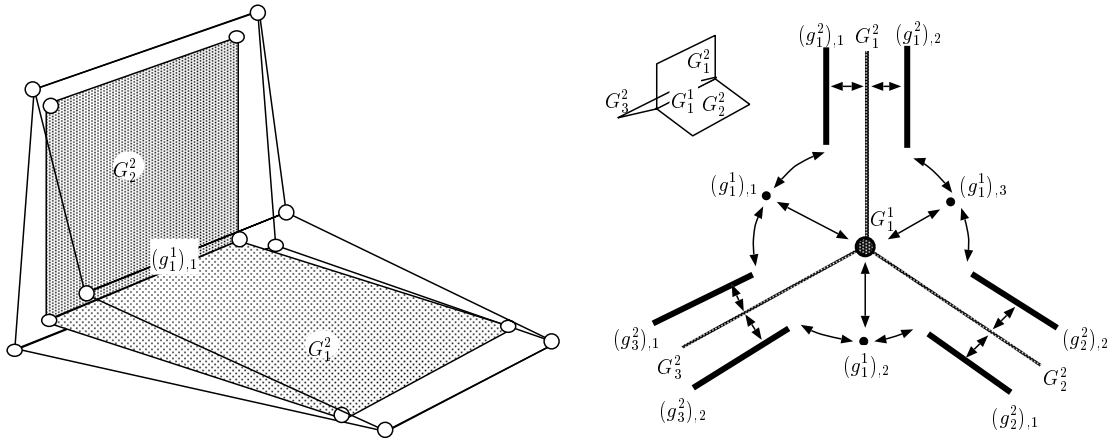


Figure 3.4: Minimal use representation of non-manifold boundaries shown in Figure 3.3.

3.2.2 Mesh definitions and concepts

The representation for the mesh [5, 57, 58, 66] used in this research is based on concepts from geometric modeling. The mesh consists of mesh vertices, edges, faces and regions. If necessary, the mesh may also represent vertex, edge and face uses. Each entity in the mesh has a unique *classification* with respect to the model.

Definition 3.6 ***Classification*** *is the unique association of a mesh entity, $M_i^{d_i}$, to a geometric model entity, $G_j^{d_j}$, $(d_i \leq d_j)$ to indicate that $M_i^{d_i}$ forms part or all of the discretization of $G_j^{d_j}$ but not $\partial G_j^{d_j}$. The classification operator is denoted by $\sqsubset$ and $M_i^{d_i} \sqsubset G_j^{d_j}$ is used to denote the classification of $M_i^{d_i}$ on $G_j^{d_j}$.*

The geometry of mesh vertices is described by points associated with them. The geometry of mesh edges and faces is not stored for linear elements. For higher order elements, the geometry of edges and faces is a polynomial or other interpolation implicitly defined through additional information stored with the mesh entities or with the geometric model in terms of the shape of the model entities they are classified on.

Definition 3.7 *The connectivity of a mesh or model entity to other mesh or model entities respectively is called* ***Topological Adjacency*** *or simply* ***Adjacency****.*

Definition 3.8 *A valid mesh is one that correctly approximates the geometry of an object.*

The implication of this definition is that the mesh should topologically and geometrically equivalent or congruent to the geometric model. Schroeder and Shephard [57, 58] lay down the following conditions for validity of a mesh:

1. The mesh should be topologically compatible with the geometric model.
2. The mesh should be geometrically similar to the geometric model.

Definition 3.9 ***Topological Compatibility****: Given a mesh consisting of mesh entities M_i^d with boundary entities M_i^{d-1} classified on the closure of a geometric model entity G_j^d with boundary entities G_j^{d-1}, the mesh is topologically compatible with the model if*

1. *Each $M_i^{d-1} \sqsubset G_j^d$ is connected to two and only two $M_i^d \sqsubset G_j^d$.*
2. *Each $M_i^{d-1} \sqsubset G_j^{d-1}$ is connected to as many number of $M_i^d \sqsubset G_j^d$ as the number of times G_j^{d-1} is used by G_j^d.*

A mesh is topologically compatible with a geometric model if it is compatible with each of the model entities.

Geometric similarity [57, 58] is the relationship of the mesh geometry to the model geometry and is a way of expressing the condition that in the limit of refinement the geometry of a mesh should exactly match that of the geometric model. This idea may be expressed more practically in a number of ways, one of which is expressed by Schroeder and Shephard [57, 58] as follows:

Definition 3.10 ***Geometric Similarity:*** *A mesh of order d, comprised of N entities M_i^d is geometrically similar to a model entity, also of order d (G_k^d), if $M_i^d \sqsubset G_k^d$, $\forall i = 1, \cdots, N$ and the parametric intersection of any two entities of the mesh is $\emptyset$, i.e. $M_i^d \sqcap^* M_j^d = \emptyset$, with the parametrization being with respect to some appropriately defined space.*

Since the mesh is required to topologically compatible with the geometric model, one can define a concept for meshes that is analogous to a face manifold or separation surface in geometric models.

Definition 3.11 *A **mesh manifold** is a set of mesh face uses around a vertex, connected by edge uses, that locally separate the three dimensional space into two halves.*

Some examples of mesh face use manifolds are shown in Figure 3.5. In Figure 3.5a, mesh manifolds for a mesh vertex classified on a model face, $M_v^0 \sqsubset G_0^2$, is shown. Since the topology of a model face by itself is 2-manifold, the mesh vertex has two mesh manifolds (S_0^v and S_0^v), one for each side of the model face. In Figure 3.5b, mesh manifolds are shown for two mesh vertices classified on model vertices in a non-manifold model. In the figure, G_1^2 is an embedded face making edge contact with two model faces G_0^2 and G_2^2. The mesh manifolds in the picture are depicted only with respect to the model region common to all three faces. The local topology at M_a^0 is non-manifold and two mesh manifolds, S_0^a and S_1^a exist at the vertex with respect to just one side of the model faces G_0^2 and G_1^2. At M_b^0, only one mesh manifold, S_0^b, exists in the model region under consideration. Note how this mesh manifold wraps around the free edge of the embedded face and uses both sides of mesh faces classified on the embedded model face. In Figure 3.5c, a non-manifold topological situation at a model vertex is depicted. Assuming that each mesh face shown is classified on a model face embedded inside a model region the mesh vertex has 3 mesh manifolds connected to it as shown in the figure.

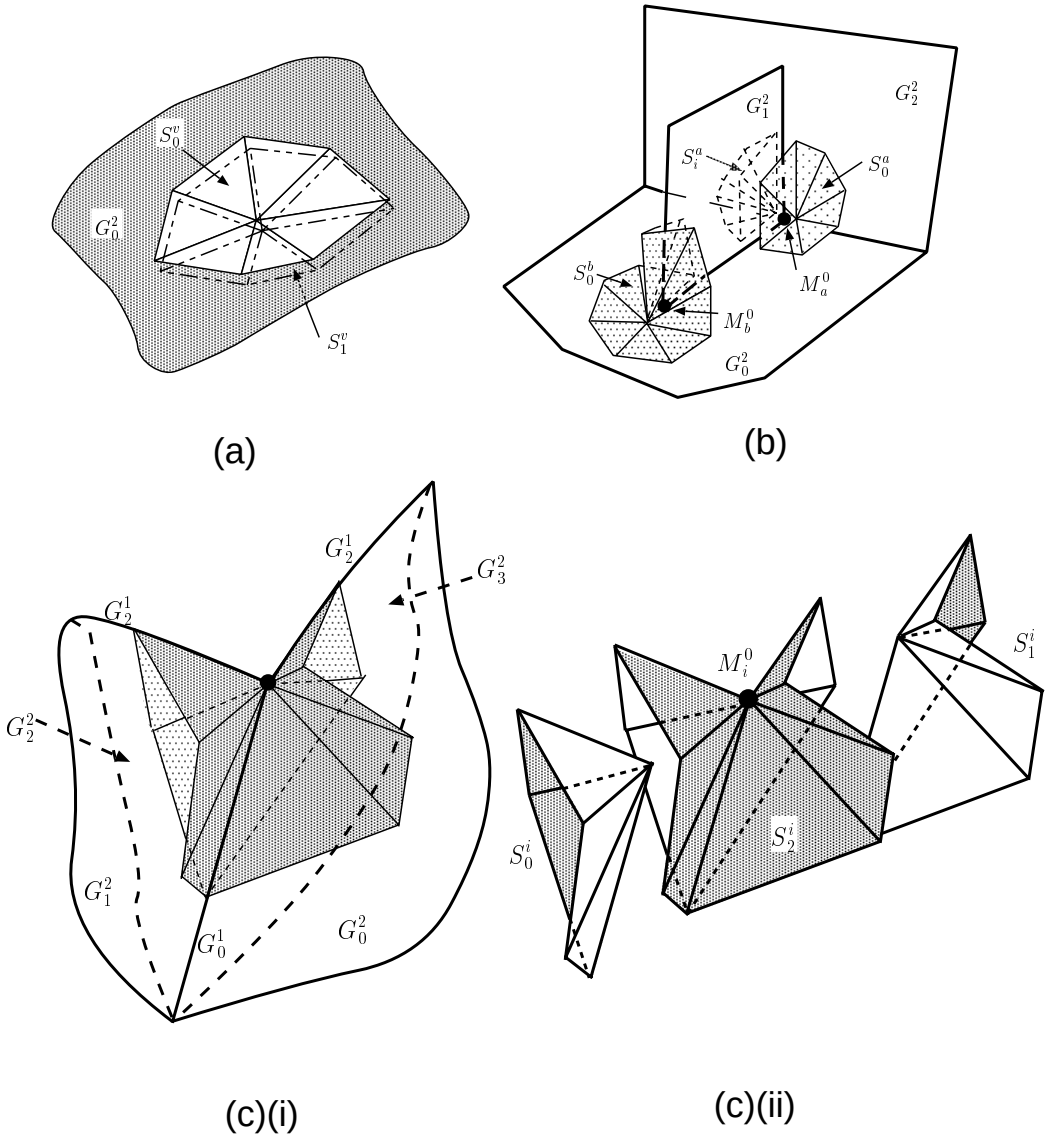


Figure 3.5: Examples of mesh face use manifolds.

CHAPTER 4
BOUNDARY LAYER MESHING - INTRODUCTION

4.1 Motivation

High Reynolds number flows exhibit high gradients near walls and in free shear layers requiring fine resolution of the solution normal to the boundary layer and across the free shear layers. The gradients in the direction of the flow (that is, tangential to the walls) are much weaker in comparison. Refining a finite element mesh isotropically to capture such strongly directional gradients results in excessive refinement along the other directions. Anisotropic elements which are small in the directions requiring fine resolution but stretched along others are often employed to keep the mesh size manageable. The mesh is typically allowed to gradually become isotropic in parts of the domain where the solution is isotropic.

The requirements on a mesh generator capable of producing such meshes are severe. Some of the qualities of anisotropic meshes capable of effectively capturing the solution in high Reynolds number flows are:

1. Elements in the anisotropic region of the mesh must be highly stretched often with aspect ratios of 1,000 to 100,000 or more. Such high aspect ratios impose severe constraints on advancing front type methods which must perform intersections under constraints of a finite geometric tolerance.

2. The elements in the anisotropic region must have good dihedral angles. Dihedral angles are harder to control with increasing anisotropy as they can degrade very quickly with slight repositioning of nodes.

3. The anisotropic mesh must be graded smoothly into the isotropic mesh since sharp changes in the sizes of elements typically cause inaccuracies in the solution and even lead to instabilities in the solver.

4. The anisotropy in the mesh must be variable over surfaces in order to minimize the number of elements in the mesh while capturing the solution accurately.

A popular strategy for generating boundary layer meshes is the Advancing Layers Method (also called advancing normals method) [11, 31, 34, 40, 54]. In this method the anisotropic mesh is generated on surfaces using a special procedure that explicitly considers the requirements of the boundary layer mesh. The remainder of the domain is meshed using one of the common isotropic mesh generation techniques.

The advancing layers method starts from a triangulation of the surfaces on which the boundary layer mesh must be grown. From each surface node a direction is picked for placing the nodes of the anisotropic mesh. This direction is typically either the true surface normal or an average discrete normal (an average of the normals of surface triangles sharing the node). Nodes are placed along these directions according to user specification. These nodes form the basis for constructing layers of prisms on top of each surface triangle. For hybrid prismatic-tetrahedral meshes, the rest of the domain is tetrahedronized by an isotropic mesher [31, 34]. For fully tetrahedral meshes, each prism is converted into three tetrahedra forming layers of stretched elements before isotropic meshing. These elements have their short direction aligned normal to the surface and their long directions tangential to surface. Care is taken to prevent crossover of the directions along which nodes are placed which leads to invalid elements. This is done by smoothing of the directions or deletion of elements when necessary. Also, interference of the boundary layer mesh on different surfaces has been accounted for in some versions of the algorithm and it is eliminated by deletion of elements.

The restriction of growing a single set of nodes[3] from each surface node constrains the method to 2-manifold models and also limits the quality of elements that can be created by this method.

The boundary layer meshing approach described here employs the advancing layers approach as its basis and generalizes it for meshing arbitrarily complex geometric domains with non-manifold topology with good quality anisotropic elements near the surface. The method is therefore referred to as the ***Generalized Advancing Layers Method***.

[3]Some researchers have allowed multiple sets of nodes to emanate at sharp corners like the trailing edge of airfoils [45].

4.2 Overview

The Generalized Advancing Layers Method also uses the surface mesh as the basic structure on which to grow the anisotropic boundary layer mesh. However, unlike other methods, the anisotropic mesh is not constrained to purely be an inflation of the surface triangles into triangular prisms and their tetrahedronization. It allows for multiple sets of nodes to emanate from each surface mesh vertex thereby facilitating the creation of:

1. valid meshes for non-manifold models, and
2. good quality elements near boundaries with sharp corners or high curvature.

The presence of multiple sets of nodes originating from a single surface node eliminates the restriction that boundary layer prisms sharing a surface mesh edge or vertex must be joined along their sides. This allows the prisms on the mesh faces to be much better shaped naturally leading to good tetrahedral element quality. The gaps created between the prisms are abstracted as more general polyhedral shapes called *blends* which are tetrahedronized. Filling the gaps between the prisms is an important part of the algorithm since failure to do so will expose the highly anisotropic faces to the isotropic mesher.

In the Generalized Advancing Layers method, curves along which the boundary layer nodes must be placed are chosen for all surface mesh vertices. Such curves are referred to as *Growth Curves*. Growth curves may be boundary growth curves (all nodes lie on the model boundary), interior growth curves (all nodes except the originating boundary node lie in the model region) or partly both. Growth curves have an arbitrarily general geometry on the boundary due to surface geometry and are initially straight lines on the interior. It is to be noted that the method itself places no restriction on the shape of the interior growth curves. The number of growth curves at a mesh vertex is dependent on the number of mesh face manifolds present at the vertex and on the geometry of the mesh at the vertex.

First, growth curves are chosen at mesh vertices classified on the model vertices. If any of these growth curves lie partly or fully on a model edge, the boundary layer mesh entities corresponding to this growth curve are incorporated into the

model edge discretization. Next, growth curves are chosen for mesh vertices classified on model edges. The growth curves that lie on model boundaries are smoothed, shrunk or pruned to avoid crossover and self-intersection. Boundary layer mesh vertices and edges are created for these growth curves. After that, neighboring growth curves are joined to form abstract quadrilaterals and triangles. These boundary polygonal shapes are triangulated and the triangles incorporated into the model face discretization. Growth curves are then chosen at mesh vertices classified on model faces. These growth curves are smoothed, shrunk and pruned to ensure creation of valid elements. Mesh vertices and edges are created along these growth curves. Entities from adjacent growth curves are then connected to form quadrilateral and triangular constructs which are then broken up into smaller triangles. Then, triangular prisms are grown from the surface mesh faces classified on the model face and tetrahedronized. Whenever the growth curves of a mesh face use have unequal number of nodes, transition elements are created to seal off high aspect ratio faces. As a final step in the creation of anisotropic elements, the gaps created between prisms due to the presence of multiple growth curves are abstracted as more general polyhedral shapes, called blends, and tetrahedronized. Once the anisotropic mesh is created, the inner boundary of the boundary layer mesh is checked for any self intersection. Self intersections are fixed first by shrinking the layers locally and then by deletion of elements, if necessary. This completes the boundary layer mesh which is then handed over to isotropic mesher for completing the meshing process. The steps in the process are illustrated in Figure 4.1.

In Chapters 5, 6, 7 and 8, the details of each step of the boundary layer meshing procedure are described. The salient points that are emphasized and described at length are:

1. The need for multiple growth curves for arbitrarily complex models.
2. Criteria for choosing growth curves in the Generalized Advancing Layers Method.
3. Ensuring topological compatibility of the mesh with the model.
4. Techniques for ensuring the geometric validity of the mesh.

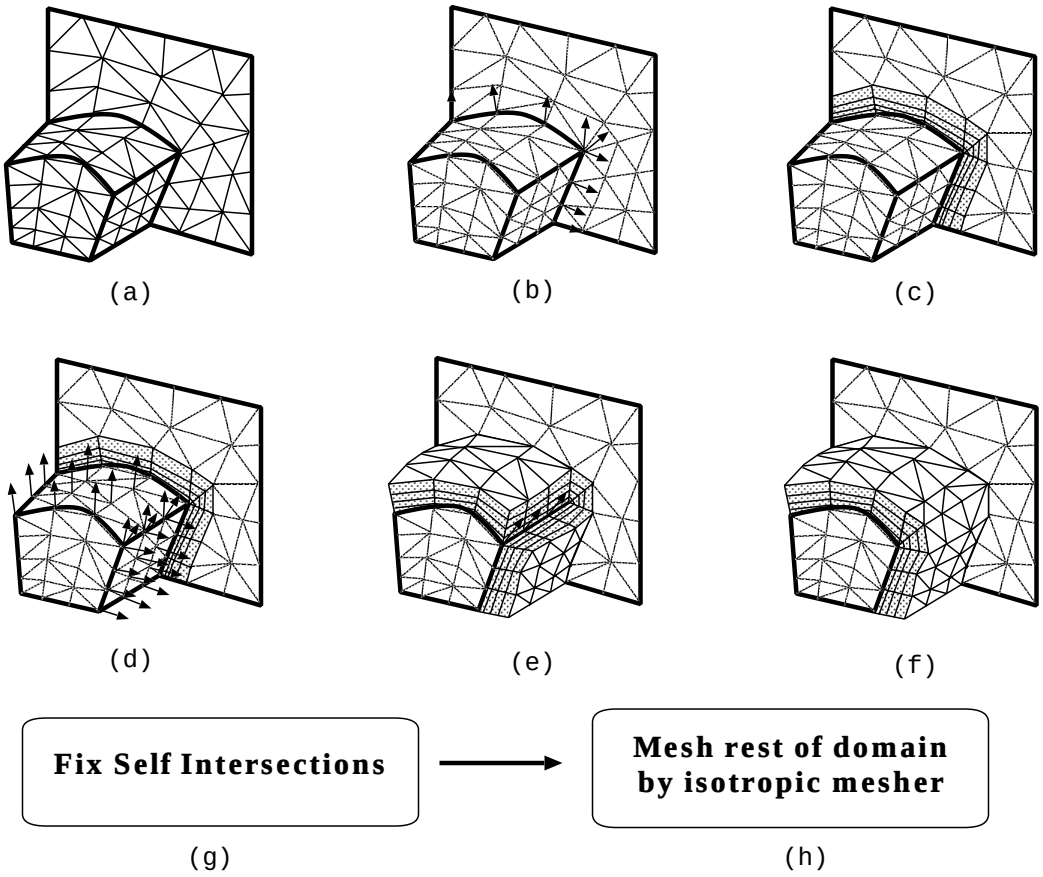


Figure 4.1: Steps in creation of mesh for viscous flow simulations. (a) Surface mesh. (b) Growth curves on model vertices and model edges. (c) Boundary retriangulation. (c) Growth curves on model faces. (d) Prism creation. (e) Blend creation. (f) Fixing self-intersection. (g) Meshing remaining portion of domain by an isotropic mesher.

5. Concepts of transition elements, their necessity and their creation procedure.

6. Concepts of blend meshes, their necessity and their creation procedure.

7. The method for detecting and fixing self-intersections in the boundary layer mesh.

Results of the mesh generation and simulations with these mesh are also presented in Chapter 9 to demonstrate the capabilities of the mesh generator.

CHAPTER 5
BOUNDARY LAYER MESHING - GROWTH CURVES

5.1 Boundary Layer Meshing Notations

C_j^i	j^{th} growth curve originating from M_i^0
$p_{j,k}^i$	k^{th} node of the j^{th} growth curve C_j^i at M_i^0
$M_{i,j,k}^0$	Mesh vertex associated with $p_{j,k}^i$
$\{p_j^i\}$	Set of all nodes of the growth curve C_j^i at M_i^0
$c_{j,k}^i$	k^{th} straight line segment between $p_{j,k}^i$ and $p_{j,k+1}^i$
$M_{i,j,k}^1$	Mesh edge associated with $c_{j,k}^i$
$\{c_j^i\}$	Set of all straight line segments between successive nodes of growth curve C_j^i at M_i^0
$\{\mathcal{M}_j^i\}$	j^{th} mesh manifold at vertex M_i^0
$\{\mathcal{F}_j^i\}$	set of mesh face uses referencing C_j^i
$\partial\{\mathcal{F}_j^i\}$	Boundary entities (edges, vertices) of mesh faces belonging to $\{\mathcal{F}_j^i\}$
$Q_{i,j}^{a,b}$ or Q	Boundary layer quadrilateral between C_i^a and C_j^b

5.2 Introduction

A critical component of any mesh generator is the point placement procedure. A good point distribution in the mesh greatly improves the mesh quality and substantially reduces the burden of element creation and optimization routines. For this reason, point placement in the boundary layer mesh generator described here is done very carefully. The rationale and procedures for point placement are explained in this chapter.

Point placement in the boundary layer mesh occurs along general curves while respecting user requested sizes for the boundary layers. These curves are called *growth curves*. Growth curves may be boundary, interior or both. All nodes of an interior growth curve except the first are classified on a region of the model. All nodes of a boundary growth curve are classified on the boundary of the model. It

must be noted that only straight line growth curves are used in the interior with the present capabilities of the mesh generator. Boundary growth curves may still take an arbitrary shape depending on the surface that the nodes of the growth curves are classified on. Growth curves may also be partly boundary and partly interior. The definition of growth curves allows them to start off with the nodes on the boundary of the model, and then separate from the model surface and grow into the interior of the model. They may even re-attach to the surface again. In principle, separation and re-attachment is allowed an arbitrary number of times. An important point to note here is that the separation of growth curves from surfaces has no relation to the actual phenomenon of boundary layer separation. Rather, it is dictated purely by considerations of element quality in the boundary layer mesh as explained below. In the current implementation, growth curves are permitted to be either interior or boundary. As will be shown in Chapter 7, even with this restriction, boundary layer quadrilaterals may be partly boundary and partly interior under special circumstances. Figure 5.1 illustrates the types of growth curves.

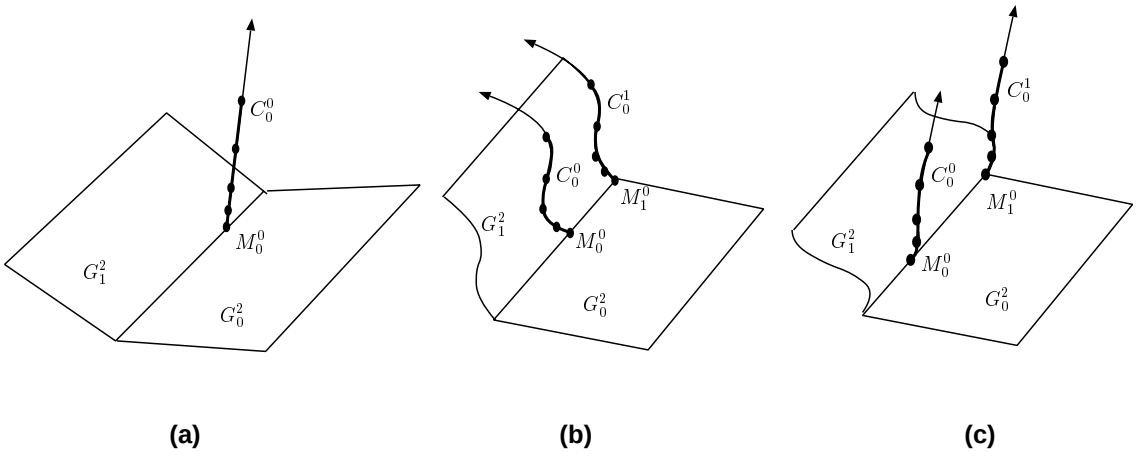


Figure 5.1: Types of growth curves. (a) Interior Growth curve. (b) Boundary growth curves. (c) Partly boundary and partly interior growth curves.

Elements in the anisotropic boundary layer mesh are created by tetrahedronization of triangular prisms, blend polyhedra and transition polyhedra grown from the surface mesh. In particular, the quality of tetrahedra resulting from prisms is heavily influenced by the deviation of the sides of the prism from the normal di-

rection to the base triangle. Therefore, nodes of growth curves growing from mesh vertices classified on model edges and vertices are allowed to lie on the boundary under the following conditions:

- The normal direction of the growth curve is close to the adjacent model surfaces.
- The quality of the elements will be good with the nodes on the boundary.

The Generalized Advancing Layers Method permits multiple growth curves to originate into a single region from any mesh vertex classified on the model boundary. Each growth curve is associated with the mesh face uses that will form a prism using nodes of this growth curve. This information facilitates subsequent connection of nodes of different growth curves unambiguously. Nodes of growth curves from mesh vertices of a mesh edge are connected to form an abstraction called a boundary layer quadrilateral (Figure 5.2a) *if they reference a common mesh face use* (These will be referred to simply as boundary layer quads in the rest of the discussion). Nodes of growth curves from vertices of a mesh face use are connected to form a boundary layer prism (Figure 5.2b) *if they reference a common mesh face use*. In other words, three boundary layer quads from three edges form a boundary layer prism if they are referenced by a common mesh face use. If growth curves that can form a quad as per the previous definition have unequal number of nodes, transition triangles are formed on top of the quad. Similarly, transition elements are formed on top of a prism if its component growth curves have unequal number of nodes. Nodes from multiple growth curves of a mesh vertex use are connected to form a boundary layer triangle (Figure 5.2c). The connectivity between nodes of a multiple boundary layer quads at mesh edge uses and multiple growth curves at mesh vertex uses is more general. These constructs will be connected using more general procedures to create blend meshes (Figure 5.2d). It is to be noted that boundary layer quads, triangles, prisms and blends are abstract constructs that never actually exist in the mesh. They are only useful for the design of algorithms and their discussion. In reality, triangles and tetrahedra of the individual layers are created directly in the mesh generator.

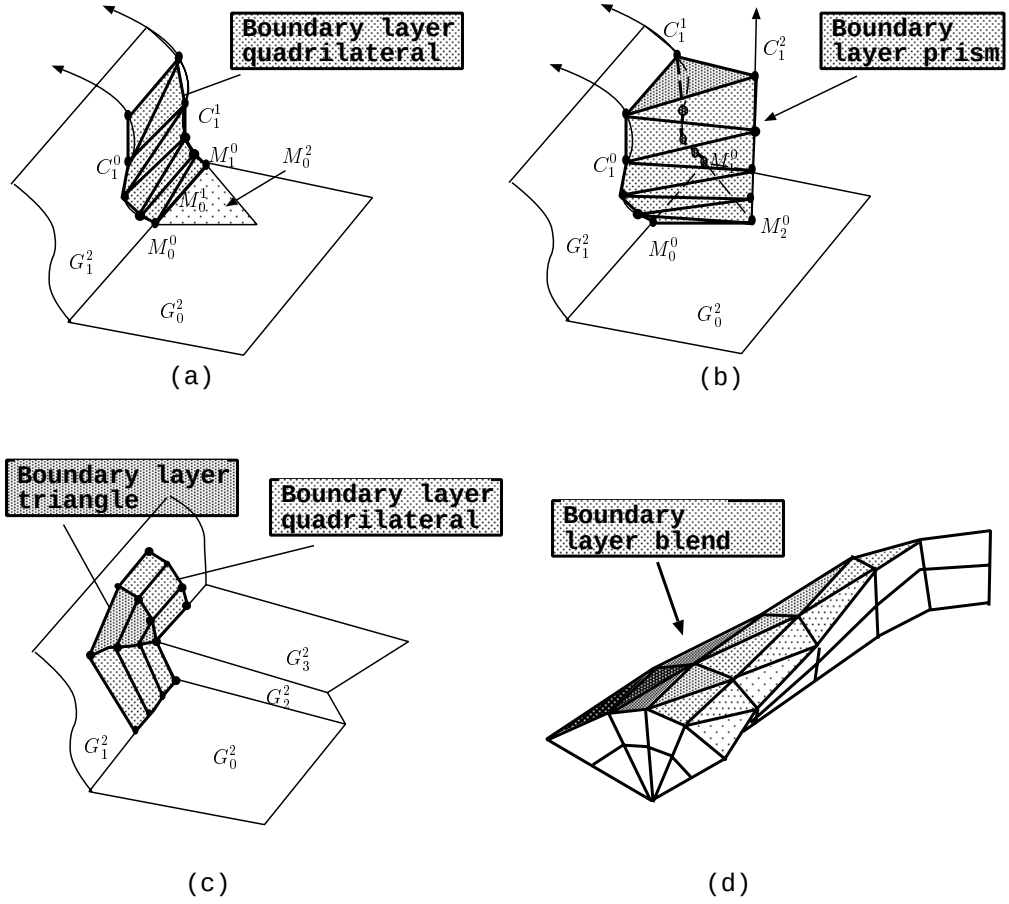


Figure 5.2: Abstract polygonal and polyhedral constructs in the boundary layer mesh.

5.3 Calculating the Number of Growth Curves at a Vertex

The number of growth curves at any mesh vertex with respect to a model face use depends on the local model topology and mesh geometry. The topological requirement for multiple growth curves at a mesh vertex with respect to a single face use arises only at some non-manifold boundaries. At these boundaries, multiple growth curves are necessary for generating a valid mesh.

Axiom 5.1 *The minimum number of growth curves at any boundary mesh vertex required to produce a topologically valid mesh is equal to the number of mesh manifolds at the vertex that include at least one mesh face use classified on a model face with a boundary layer.*

The above assertion can be easily demonstrated to be true with the help of an example shown in Figure 5.3a,b. Here, the embedded face G_1^2 is incident on vertex G_1^0 along with two other 2-manifold faces, G_2^2 and G_3^2. It is assumed that a boundary layer mesh is being grown on all three faces (note G_1^2 has two uses for the boundary layer to grow on). It can be seen from Figure 5.3a that use of only one growth curve at $M_i^0 \sqsubset G_1^0$ and $M_i^0 \sqsubset G_1^1$ will lead to a topologically invalid mesh as some of the quads will intersect G_1^1 or penetrate G_1^2. The correct solution in this example is to have two growth curves at the vertex, one for each of the two mesh manifolds at the vertex (Recall Figure 3.5 in Chapter 3). Also, the nodes of each of these growth curves must lie within the respective mesh manifold (Figure 5.3b). Similarly, in 3-D, interior edges may penetrate model faces if the right number of growth curves as dictated by the mesh and model topology are not present at each vertex.

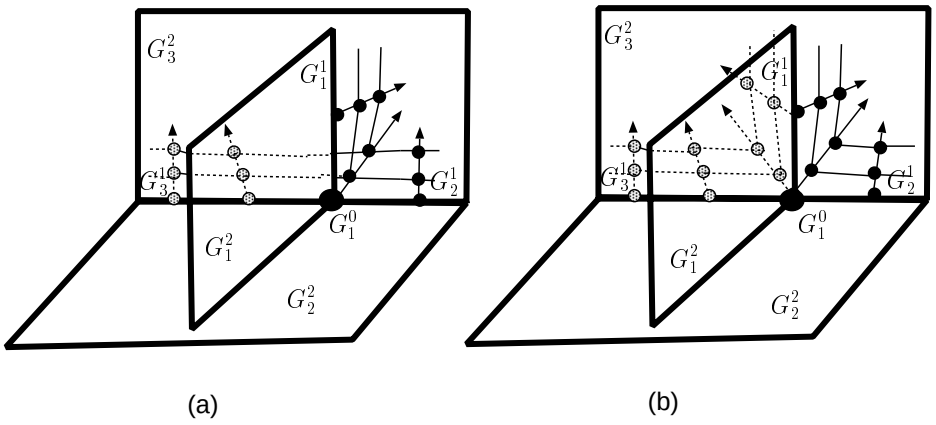


Figure 5.3: Need for multiple growth curves at non-manifold boundaries. (a) Single growth curve along G_1^1. (b) Two growth curves along G_1^1.

At some mesh vertices, multiple growth curves may become necessary due to the geometry of the model faces or the coarseness of their discretization. This is because creation of valid prisms requires that the nodes of a growth curve at any mesh vertex be "visible" from any mesh face connected to the mesh vertex. *Visibility of a node from a mesh face means that an element formed by connecting the mesh face to the node must have positive volume.* If the surface discretization is very coarse or the model geometry itself changes drastically, the normals of the mesh faces may

vary so much that it may not possible to find a valid common node that is visible from all the faces (even with the methods described in [31, 54]). Such impossible situations are the limit of the case where the growth curve deviates greatly from the mesh face normal leading to large dihedral angles in elements. Therefore, in general, it is desirable to have multiple growth curves at mesh vertices where the normals of the connected mesh faces are very disparate. The concept of visibility of growth curves is illustrated in Figure 5.4 in which a set of mesh faces uses connected to a mesh vertex classified on a model edge are shown. In Figure 5.4a, the growth curve from the mesh vertex makes nearly equal angles with the plane of the two model faces using the model edge. This makes the growth curve visible to all mesh face uses connected to the mesh vertex. On the other hand, in Figure 5.4b, the growth curve is skewed to one side making the growth curve visible to mesh faces classified on G_1^2 and invisible to those classified on G_0^2.

An example showing a combination of the topological and geometric need for multiple growth curves is shown in Figure 5.5. The situation illustrated is borrowed from a real mesh. In this figure, the different face uses form a total of three manifolds and therefore at least three growth curves are necessary to create topologically valid connections in the boundary layer mesh. In addition, the "convex" mesh manifold, S_2^i, requires multiple growth curves so that the nodes of each growth curve are visible from the mesh face uses that reference them and geometrically valid boundary layer elements may be formed.

In keeping with the necessity of creating a topologically valid mesh and desirability of creating well shaped prisms, mesh manifolds are first found at each vertex and these are then divided up based on geometric criteria into subsets of mesh face uses. Each of these subsets of mesh face uses then share a common growth curve to be used in their prisms.

5.4 Finding Mesh Manifolds For Mesh Vertices

The procedure to find mesh face use manifolds at a boundary mesh vertex is described in this section. The algorithm does not require explicit use information for the mesh entities. The only requirement is that the mesh faces be orientable. The

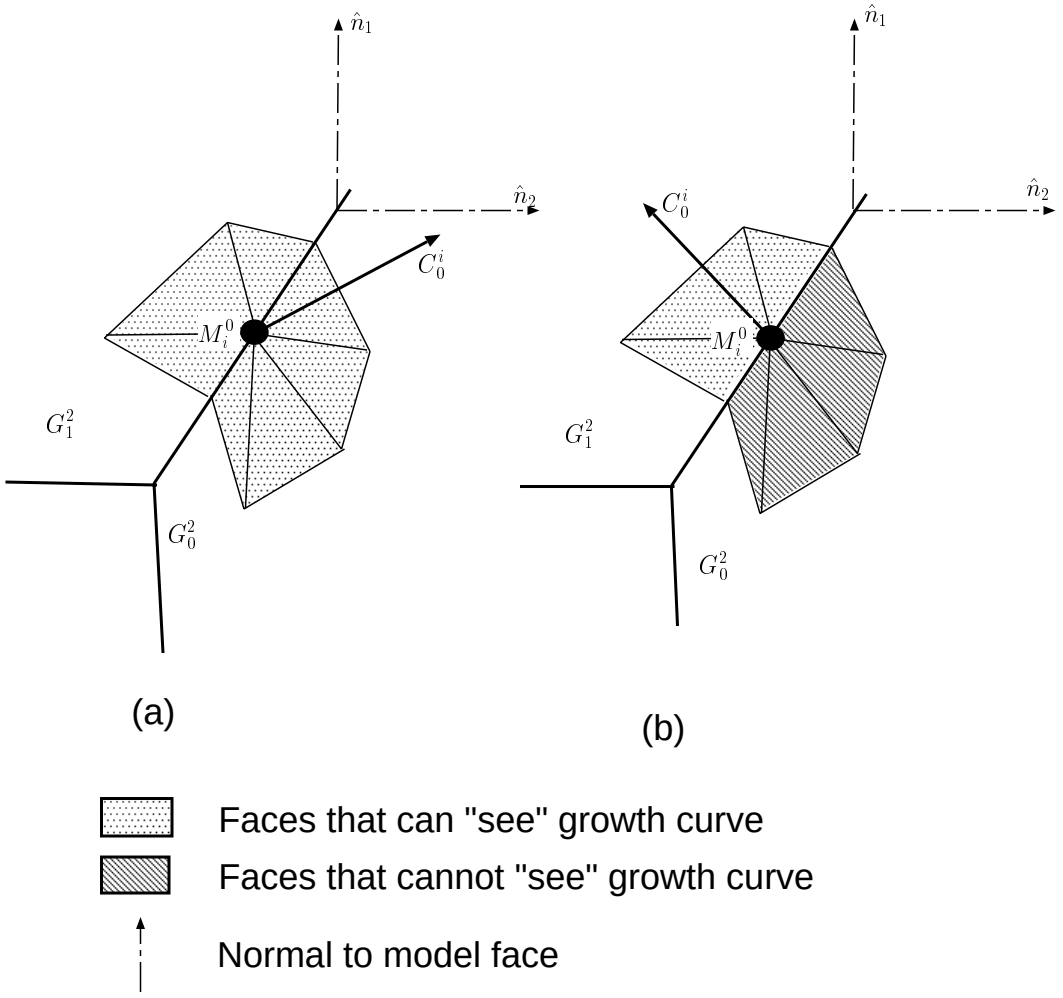


Figure 5.4: Visibility of growth curves. (a) Example where growth curve is visible to all mesh faces shown. (b) Example where growth curve is visible to only some of the faces in the set shown.

central idea used in finding mesh face use manifolds is that *a pair of adjacent face uses in a mesh manifold must use the common edge in opposite directions.* In case of multiple choices, the candidate mesh faces uses are ordered around the common edge and the radially closest ones are paired together based on the dihedral angles. The steps of the procedure are (refer to Figure 5.6):

1. Given a mesh vertex M_i^0, find the connected set of mesh face uses (as pairs of mesh faces and associated directions) $\{\mathcal{S}_i\} = \{(M_j^2, s) \mid s = \pm 1, M_i^0 \subset \partial M_j^2\}$, where s represents the side of the face.

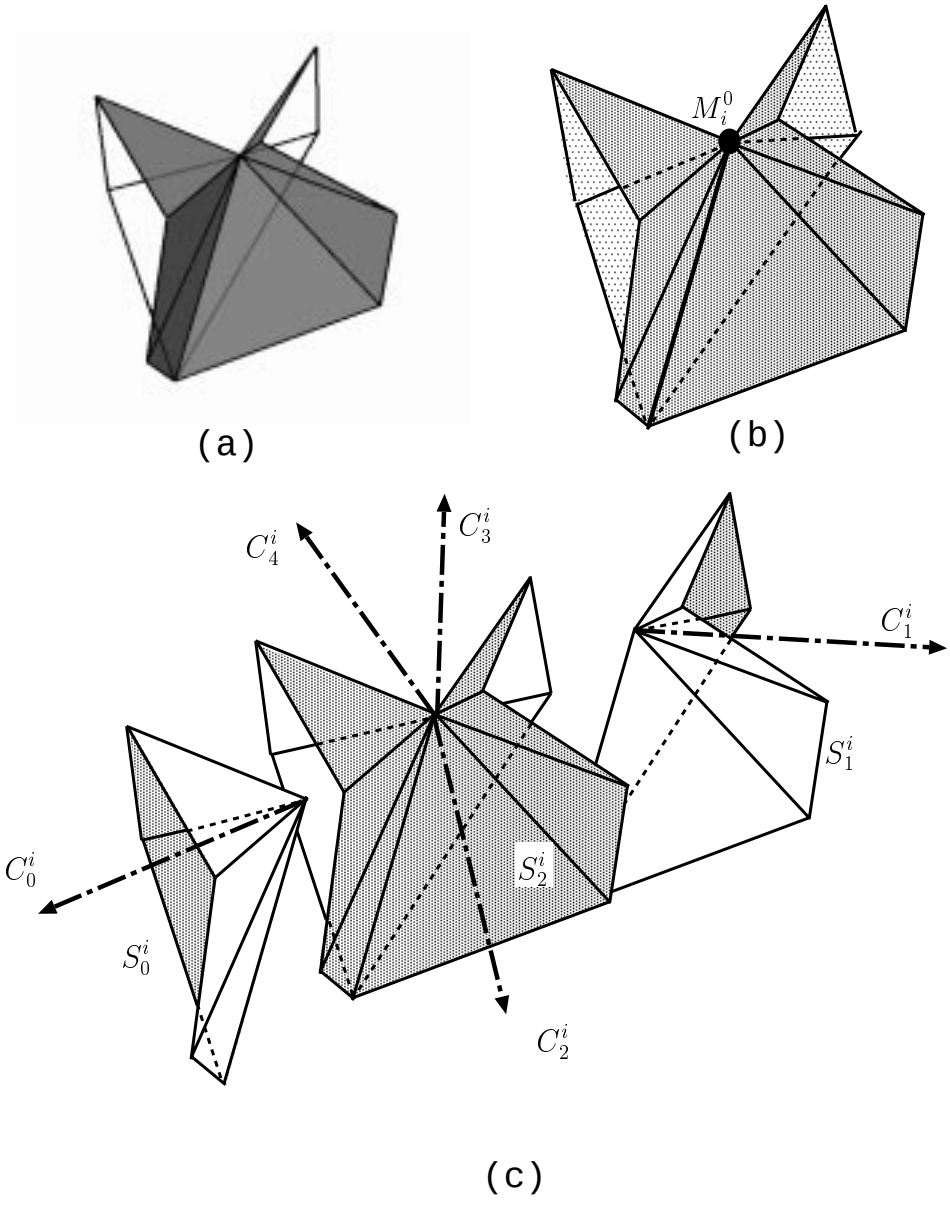


Figure 5.5: Mesh topology and geometry for which multiple growth curves are necessary. (a) Local view of real surface mesh. (b) Schematic of local mesh shown in 'a'. (c) Exploded view of the mesh manifolds with growth curves.

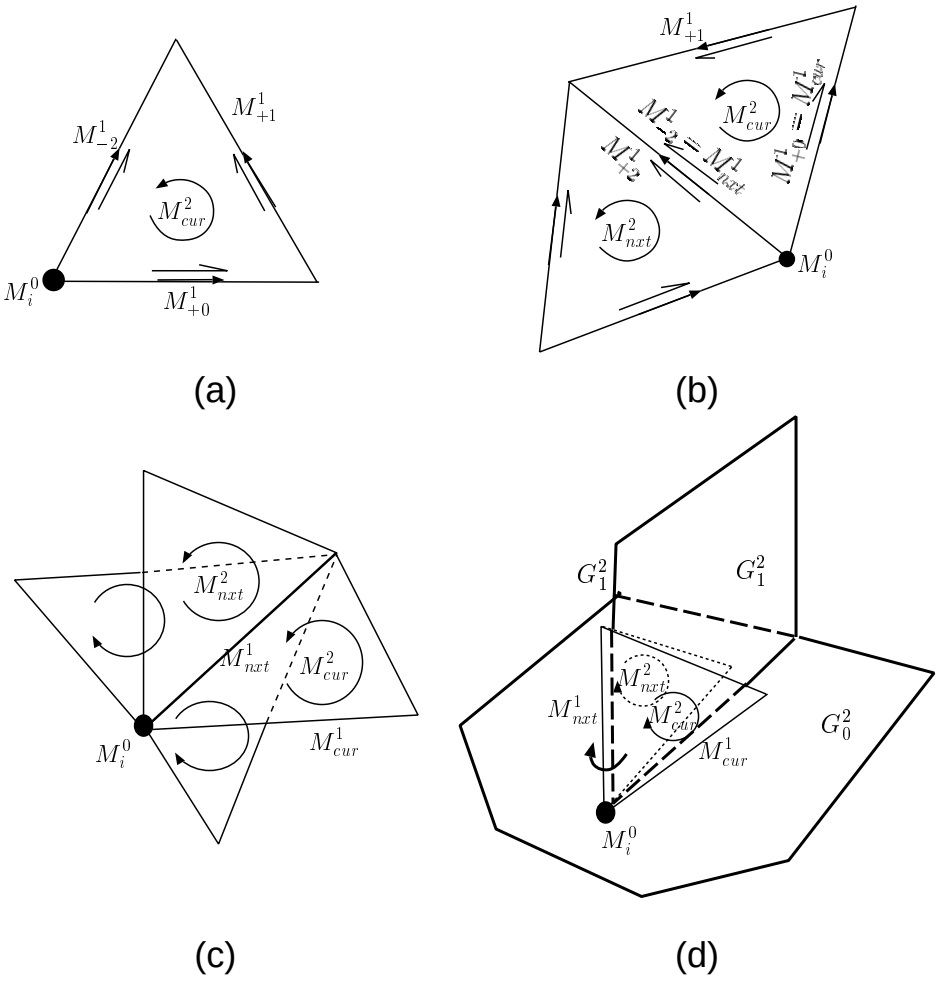


Figure 5.6: Finding mesh manifolds. (a) Face use with its edge uses. (b) Getting the next face use on a model face. (c) Getting the next face use at non-manifold model edge. (d) Getting the next face use at free edge of an embedded face.

In the absence of explicit face use information in the mesh, the set of candidate "face uses" is determined using the classification of mesh faces and modeler queries. A face-side pair is considered as a mesh face use if the mesh face is classified on a model face and the particular side of the model face is connected to a model region.

2. Choose a face use of the set $\{\mathcal{S}_i\}$ as a starting face use, $M^2_{cur} \sqsubset G^2_k$.

3. Initialize a new mesh manifold set, $\{\mathcal{F}_i\}$ and add M^2_{cur} to it.

4. Find the loop of edges, $[M^1_{\pm 0}, M^1_{\pm 1}, M^1_{\pm 2}]$, $M^1_{i=0,1,2} \subset \partial M^2_{cur}$ of the face in the direction s starting with vertex M^0_i. Note that $M^0_i \subset \partial M^1_0, \partial M^1_2$, and $M^0_i \not\subset \partial M^1_1$ (Figure 5.6a).

5. Denote M^1_0 as M^1_{cur} and M^1_2 as M^1_{nxt}.

6. The next face use in the mesh face use manifold is found as follows:

   - if $M^1_{nxt} \sqsubset G^2_k$, then there is only one other face connected to M^1_{nxt}. Pick this as the next face, M^2_{nxt}. The direction in which the face is used is s (Figure 5.6b).
   - if $M^1_{nxt} \sqsubset G^1_k$ and G^1_k is a 2-manifold edge, there is only one other model face connected to G^1_k and equivalently, only one other boundary mesh face connected to M^1_{nxt}. Pick this as the next face, M^2_{nxt}.

     The direction in which the M^2_{nxt} is used in the current mesh manifold is determined by the relative directions in which M^2_{cur} and M^2_{nxt} use M^1_{nxt}. If M^2_{cur} and M^2_{nxt} use the edge M^1_{nxt} in opposite directions, then the direction of M^2_{nxt} continues to be the same as that of M^2_{cur}. if M^1_{nxt} is used in the same direction, the direction of the face is reversed (same as Figure 5.6b).
   - if $M^1_{nxt} \sqsubset G^1_k$ and G^1_k is a non-manifold edge, there are multiple choices for the next face in the manifold set of mesh face uses. The choice of which face use to pick next, is based on radially ordering the face uses around the edge. This is done by finding the dihedral angle between the different face uses and the current face use (Figure 5.6c). Note that the radial ordering is of face uses and not just faces. This implies that just picking the smallest angle between two faces using the natural orientation of the faces is not sufficient.
   - if $M^1_{nxt} \sqsubset G^1_k$ and G^1_k is the free edge of an embedded face then there are no other faces connected to G^1_k. In this case the mesh manifold wraps around the free edge and goes from one side of the face to the other. Thus the next face in the manifold for this situation is the current face

used in the opposite direction, or in other words, the mesh face use mate of the current face use (Figure 5.6d).

7. If the next face is the same as the starting face then the mesh manifold is complete. Continue at step 3 above if any mesh face uses are left to be put into the manifold set.

   If not, make the next face the current face ($M^2_{cur} = M^2_{nxt}$) and the next edge the current edge ($M^1_{cur} = M^2_{nxt}$). Continue adding face uses to the current manifold set, i.e., continue at 4 above.

The procedure to find the unique angle (and not just the principal angle) between two face uses $M^2_{s_1,1}$ and $M^2_{s_2,2}$ ($s_1, s_2 = \pm 1$) at the mesh vertex M^0_i and sharing an edge M^1_j is as follows (See Figure 5.7a,b). Get the ordered set of vertices of each of the faces, $\{V_1\} = [M^0_i, M^0_j, M^0_k] \subset \partial(M^2_1)$ and $\{V_2\} = [M^0_i, M^0_l, M^0_j] \subset \partial M^2_2$. Note that $\{V_1\}_{,0} = \{V_2\}_{,0} = M^0_i$ is the first vertex in each of the two adjacency sets and the vertices of the common edge M^1_j are $\{V_1\}_{,0} = M^0_i$ and $\{V_1\}_{,1} = M^0_j$ or $\{V_2\}_{,2} = M^0_j$ and $V_{2,0} = M^0_i$. Say the geometric point associated with each vertex $\{V_k\}_{,m}$ is $\{P_k\}_{,m}$.

$$
\begin{aligned}
\vec{v}_{11} &= P_{1,1} - P_{1,0} \\
\vec{v}_{12} &= P_{1,2} - P_{1,0} \\
\hat{n}_1 &= \frac{\vec{v}_{11} \times \vec{v}_{12}}{\|\vec{v}_{11} \times \vec{v}_{12}\|}
\end{aligned}
\qquad\qquad
\begin{aligned}
\vec{v}_{21} &= P_{2,1} - P_{2,0} \\
\vec{v}_{22} &= P_{2,2} - P_{2,0} \\
\hat{n}_2 &= \frac{\vec{v}_{21} \times \vec{v}_{22}}{\|\vec{v}_{21} \times \vec{v}_{22}\|}
\end{aligned}
$$

$$\begin{aligned} \cos\theta' &= \hat{n}_1 \bullet \hat{n}_2 \\ \\ \vec{v}_{e_1} &= P_{1,1} - P_{1,0} \\ \vec{v}_{e_2} &= \hat{n}_2 \times \hat{n}_1 \\ \alpha &= \vec{v}_{e_1} \bullet \vec{v}_{e_2} \\ \\ \theta &= \theta' \qquad \text{if} \qquad \alpha \geq 0 \\ &= \pi + \theta' \qquad \text{if} \qquad \alpha < 0 \end{aligned}$$

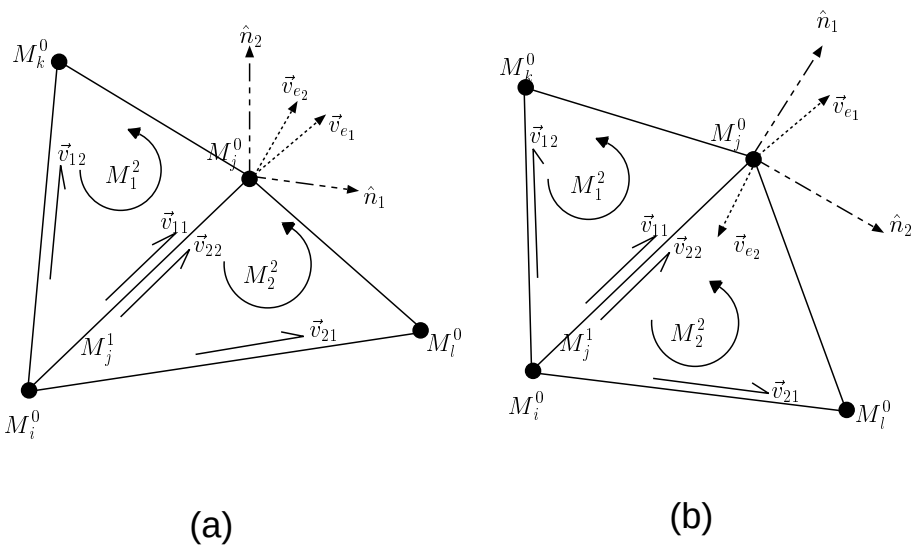


Figure 5.7: Dihedral angle between face uses. (a) Dihedral angle less than π. (b) Dihedral angle greater than π.

5.5 Finding Mesh Face Use Subsets Sharing a Common Growth Curve

The determination of subsets of mesh face uses in a manifold sharing a common growth curve is based on the dihedral angle between pairs of face uses. The

procedure starts with finding a convex edge in the manifold of mesh face uses. *A convex edge is defined as one where the dihedral angle between the two connected face uses in the manifold of mesh face uses is greater than some tolerance, ϵ, where* $\pi < \epsilon \leq 2\pi$. If no such edge can be found in the manifold, then all the mesh face uses share a common growth curve (Figure 5.8a). If such an edge is found, then the procedure traverses the face uses around the common mesh vertex as described above until another convex edge is found or the starting face is reached. The face uses between the starting convex edge and the next convex edge are put into a set of face uses that will share a common growth curve, a new set is initialized and process continued (Figure 5.8b,c). If there is only one convex edge in the manifold set (for example, at the interface between a free edge of an embedded face and a 2-manifold face or as shown in Figure 5.8d), the set is split into two subsets. The bisection is performed such that face uses of the manifold on one side of the convex edge is one subset and the face uses on the other side form the other subset. The procedure ensures that all face uses in a subset will always have a growth curve whose nodes are visible to every face use in that subset.

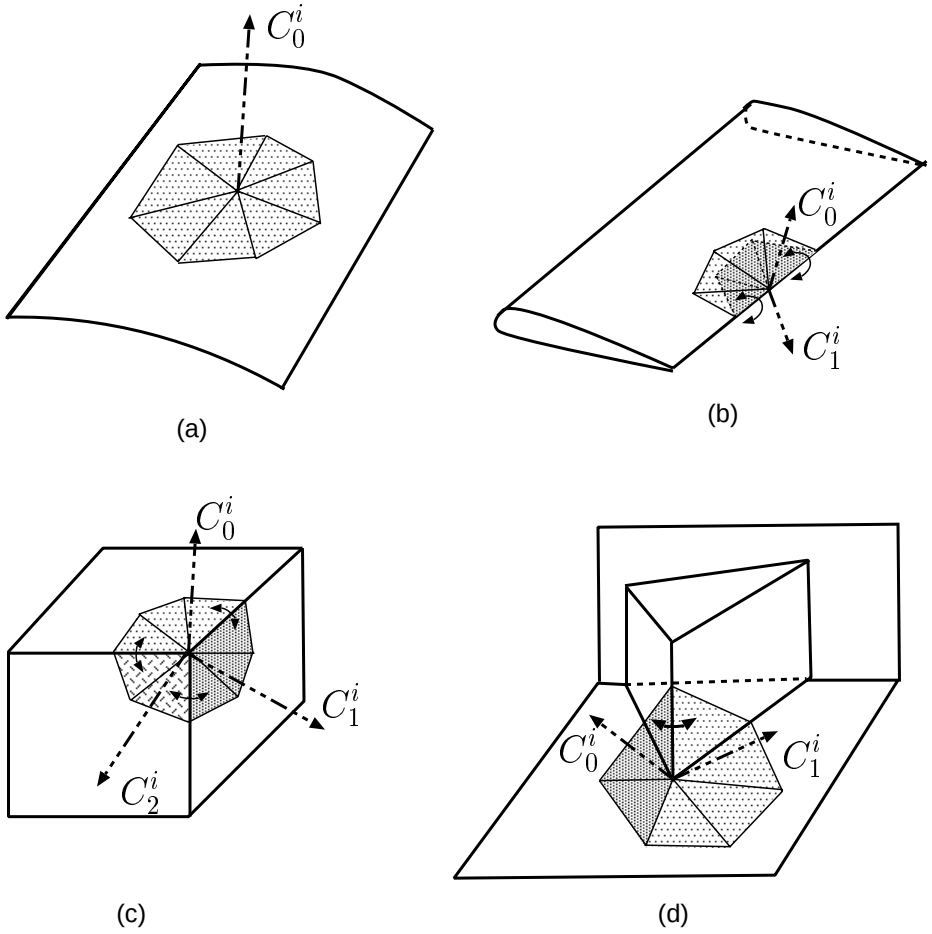


Figure 5.8: Mesh face use subsets in mesh manifolds. (a) All mesh face uses share common growth curve. (b) Two convex edges, shown by curved double headed arrows, in mesh manifold. (c) Three convex edges in mesh manifold. (d) Only one convex edge in mesh manifold which is subdivided into two subsets. Note that although all the convex edges shown in figure are classified on model edges, convex edges may occur on model faces as well and are not dependent on the classification of the edge.

5.6 Growth Curves at Model Vertices and Model edges

In the interest of creating a good quality boundary layer mesh, growth curves from mesh vertices classified on model vertices and model edges are first attempted to be grown as boundary growth curves. In doing so, the growth curves must respect topological compatibility of the mesh with the model and *estimated* geometric validity of mesh. If creating a boundary growth curve violates any of these requirements, the growth curve is grown into the interior.

Consider a mesh vertex, $M_i^0 \sqsubset G_j^d, \quad d = 0, 1$, and a set of mesh face uses, $\{\mathcal{F}\}$, sharing a growth curve, C_j^i. The set of mesh face uses is determined as described above. Then the determination of C_j^i is first carried out by a simple procedure which assumes a single classification for all the nodes of the growth curve. If this causes problems with geometric or topological validity that cannot be resolved then a more general procedure is proposed in which the nodes of a growth curve can have different classifications. In both cases an attempt is made to place the nodes of the growth curve on the lowest order model entity possible. For example, when constructing a growth curve from a mesh vertex on a model vertex, the lowest order model entity that can carry the growth curve is a connected model edge. Since model edges and faces may be curved, a straight line approximation of the growth curve (obtained from an average normal of the given mesh face uses) is used to find locations on the model entity close to the initial positions of the nodes. The detailed procedure to find a growth curve on the boundary is given below.

As before consider a mesh vertex, $M_v^0 \sqsubset G_1^{d_1}, \quad d_1 = 0, 1$, and a set of mesh face uses, $\{\mathcal{F}_j^v\}$, which will share a growth curve, C_j^v at this vertex. Say the growth curve is being grown onto the model entity $G_2^{d_2}$. Since the $\{\mathcal{F}_j^v\}$ is known, the model region into which the growth curve is being grown is fixed, i.e., $G_2^{d_2} \subset \partial(G_r^3)$ and $p_{j,k}^v \sqsubset \overline{G_r^3}, \quad k = 1, n-1$ where n is the number of nodes the growth curve has (indexed from 0 to $n-1$). The steps in constructing a boundary growth curve are:

1. Find $\{G_i^{d_2} \mid G_1^{d_1} \subset \partial G_i^{d_2}, \quad G_i^{d_2} \subset \partial(G_r^3), \quad d_1 < d_2 < 3\}$ in ascending topological order. These are candidate model entities on to which the boundary layer growth curve may be classified provided it meets all the topological and geometric criteria.

2. Perform a series of checks for each entity, $G_i^{d_2}$ until a suitable entity to grow the boundary layer growth curve onto is found. The selected entity must pass the following checks:

   (a) Check if any model face connected to the boundary entity or the boundary entity itself are to have a boundary layer mesh growing into this region or into a region which has already been processed. In this case the candidate entity is rejected since the boundary layer from this set of mesh faces will interfere with the boundary layer on the connected model face.

   (b) Find the closest point to the node $p_{j,n-1}^v$ where n is the number of nodes to grow on the growth curve. If the closest point is coincident within geometric tolerance of the originating node $p_{j,0}^v$, reject the entity. This happens when the model entity is locally nearly perpendicular to the growth curve.

   (c) Check if $\{\mathcal{F}_l^v\}$ or $\partial\{\mathcal{F}_l^v\} \sqsubset G_i^{d_2}$, where $\{\mathcal{F}_l\} \subset \{\mathcal{F}_j^v\}$. If not, reject $G_i^{d_2}$.

   (d) Make sure the connections that will result between the growth curve and any adjacent boundary growth curves that have already been created do not violate topological compatibility; if not, reject this entity. The adjacent boundary growth curves are determined by finding the vertices of the mesh face uses in the set that are classified on a model edge or vertex and checking the boundary growth curves at those vertices (See Chapter 6 for a precise definition of adjacent growth curves).

   (e) Make sure that the boundary layer node placement on this entity will not inevitably lead to large dihedral angles. If it does lead to large dihedral angles, then reject the entity.

   Note that the actual boundary layer entities from the growth curve under investigation and its adjacent growth curve are not yet formed at this stage and only the local geometry at the potential locations of the growth curve nodes is being considered. Therefore, it is only possible to estimate the worst dihedral angles of elements connected to this growth curve and not accurately calculate them. The estimation of the dihedral angles that

will be formed is done by calculating the dihedral angles between the half planes represented by:

i. the tangent plane of the model face on which the boundary layer is being grown, evaluated at the location of the base node, and

ii. the tangent plane of the model face or each of the model faces adjacent to the model entity on which the boundary layer node (other than the base node) is classified, evaluated at the location of that node.

Since evaluation of the normals (or tangent planes) of model faces yields only infinite planes, the dihedral angles between them is not uniquely defined. The unique dihedral angle of interest is calculated by determining the two half planes that represent the planes of two faces that will share an edge during element creation (Figure 5.9). The half tangent plane of the model face on which the boundary layer is being grown can easily be determined using a mesh face use of the manifold that is classified on the model face. The half plane of the model face on the closure of which the boundary layer node will be classified on can be determined by its relation to the first half plane. To do this, a mesh edge is found which is (i) on the boundary of a mesh face using the growth curve and (ii) is classified on the closure of the two model faces. In this calculation, the direction of use of the model faces in the mesh manifold must be accounted for in the calculation of the normals of the two model faces. Directly adopting the natural normal of the model face without consideration of the uses may yield the complement of the dihedral angle instead of the angle itself.

(f) Make sure that no node of the growth curve (except the originating or base node) is forced to be on the boundary of the candidate model entity. If so, this simple approach cannot handle it and one must switch to a more general approach. The general approach does not attempt to find a single classification for all the nodes of the growth curve. Rather, it proceeds layer by layer attempting to find a good and valid location and classification for the node based on the properties of the previous node.

This is a computationally more expensive option and is not used for now. Instead an interior growth curve is grown in its place.

(g) Sometimes two growth curves from a mesh vertex in a non-manifold model may lie on the same model face. For example, consider two model faces, G_i^2 and G_j^2, in the same plane sharing a model edge, G_l^1. Also, let another planar model face, G_k^2, perpendicular to the first two, be incident on the edge. If the first two model faces have a boundary layer mesh and the perpendicular face does not, nodes from the two sets of growth directions at $M_i^0 \sqsubset G_l^1$ will lie on G_k^2. Such growth curves are first checked to see if they are coincident; if they are, the growth curves and the respective sets of mesh face uses referencing them are merged. If the growth curves are not coincident, it must be verified that the boundary layer quads that will be formed using these growth curves will not intersect each other thereby creating an invalid mesh. The check for interference of the quads using these growth curves is performed as follows (see Figure 5.10). Consider two growth curves C_0^v and C_1^v at the mesh vertex $M_v^0 \sqsubset G_l^1$. Let all nodes of the two growth curves be classified on the model face G_k^2. For each growth curve, $C_g^v, \quad g = 0, 1$, find a mesh edge $M_e^1 \sqsubset G_l^1$ of a mesh face use referencing the growth curve C_g^v. Form an abstract triangle T_g with the vertices of M_e^1 and the top node of C_g^v. If $C_{g'}^v, \quad g' = 1 - g$ intersects T_g in any place other than M_v^0, then the two growth curves are incompatible. The intersection is done in real space by projecting the triangles onto a tangent plane of the face, G_k^2. The parametric space is not used as it is not guaranteed that it is well behaved locally. In case of interference, one of the growth curves is deleted and the other is used in its place.

If two growth curves from a mesh vertex lie on the same edge, then they are checked to see if they overlap and merged if they are. Note that the only case two such growth curves will not interfere is when they originate from the vertex of a closed edge and grow in opposite directions on the edge, a feasible but rare situation.

3. If all the above checks are passed for a model entity then the growth curve nodes are placed onto this entity. If not, the procedure returns without calculating a boundary growth direction and an interior growth curve must be created instead.

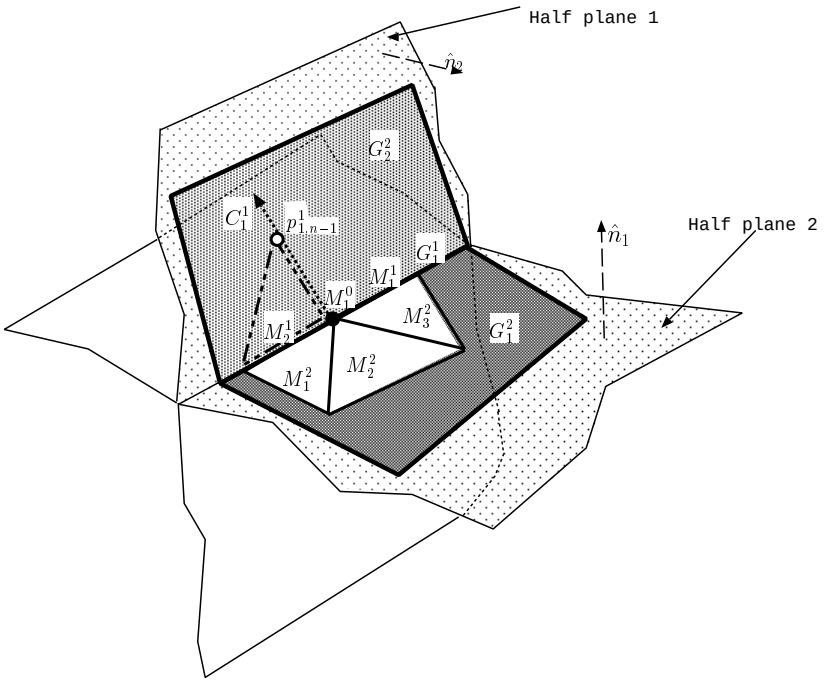


Figure 5.9: Estimation of dihedral angles of elements during growth curve creation.

5.7 Growth Curves on Model Faces

Growth curves from the mesh vertices classified on model faces are always classified in the interior of the model. The model region on which the entities of the growth curve are classified is determined by the model face use on which the boundary layer is being grown. Growth curves in the interior are straight lines although the mechanism to represent them as curved (either during their creation or later by node repositioning) exists fully in the design of the mesh generator. The direction of a growth curve from a mesh vertex classified on a model face is determined by the average of the normals of the face uses sharing the growth curve.

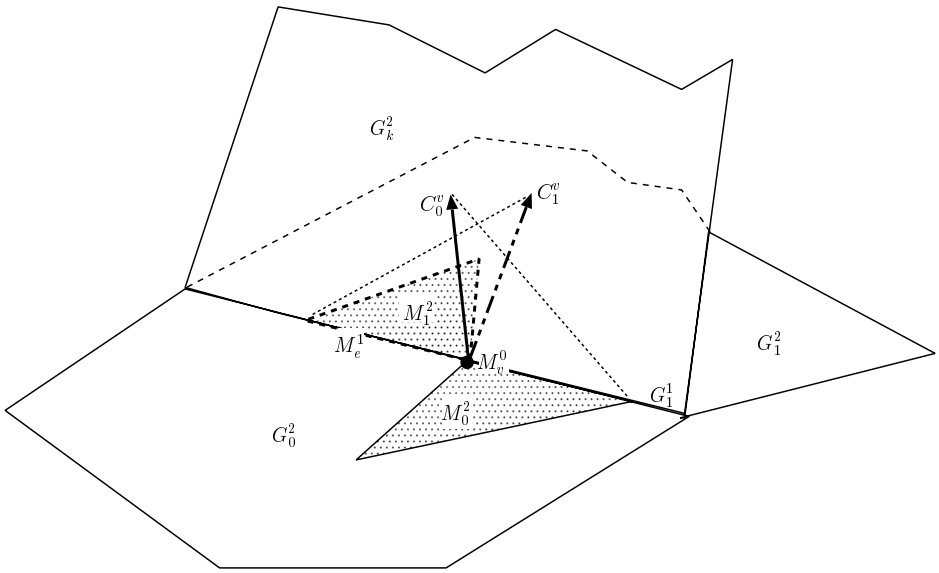


Figure 5.10: Check for interference of boundary layer quads connected to two growth curves of a single mesh vertex.

Since the topology on the model faces is simple (it is 2-manifold) multiple growth curves on the model faces occur solely due to geometry.

5.8 Node Spacing Along Growth Curves

Node spacing for growth curves may be specified in one of three ways - *geometric*, *exponential* or *adaptive*. In all three cases, the first layer thickness must be specified by users (or a calling application if the procedure is incorporated into an adaptive analysis and remeshing loop). In the geometric method of specifying the node spacing, the two additional parameters that are prescribed are the number of layers and the total thickness of the boundary layer mesh. Using this, the thickness of the individual layers and the node spacing along the growth curves is calculated (Figure 5.11a). Given the thickness of the i^{th} layer in the mesh to be t_i, the total thickness of the boundary layer mesh to be T and the number of layers to be n, the following relation holds for geometric growth:

$$
\begin{aligned}
t_{i+1} &= r t_i && (5.1)\\
T &= t_0 + t_0 r + t_0 r^2 + t_0 r^3 \cdots t_0 r^{n-1} && (5.2)\\
&= t_0 \frac{(r^n - 1)}{(r - 1)} && (5.3)
\end{aligned}
$$

where r is the growth factor between the layers. Knowing t_0, T and n, r can be calculated using any method to solve a non-linear equation like the Newton-Raphson method.

For exponential growth, only the first layer thickness and number of layers is specified for calculation of the node spacing (Figure 5.11b). For this type of specification, the following relationships hold:

$$
\begin{aligned}
t_{i+1} &= t_0^i && (5.4)\\
T &= t_0 + t_0^2 + t_0^3 + \cdots + t_0^{n-1} && (5.5)\\
&= t_0 \frac{(t_0^n - 1)}{(t_0 - 1)} && (5.6)
\end{aligned}
$$

In the adaptive method of boundary layer thickness specification, the first layer thickness t_0 and the number of layers, n, are specified. The growth of the boundary layer thickness is still geometric but the layer thickness growth factor r is calculated to ensure a smooth gradation of the boundary layer mesh into the isotropic mesh (Figure 5.11c). The isotropic mesh size in the neighborhood is taken to be the same as the average mesh size at the base node of the growth curve. To see how the layer thicknesses are calculated, consider that the average mesh size at the base node is αh_{ave} where $0 < \alpha \leq 1$. α is chosen as 0.5 in the current implementation. A value for r must be found such that the thickness of the last layer, $t_{n-1} \approx \alpha h_{ave}$. Since $t_{n-1} = t_0 r^{n-1}$, r may be calculated as $(\alpha h_{ave}/t_0)^{(1/n-1)}$.

The attribute specification system [52, 61, 76] used for prescribing boundary layer mesh parameters allows spatial variation of all the variables, t_0, T and n while maintaining the geometric growth rate of layer thicknesses (Figure 5.11d). In the

case of a varying number of layers on a single topological entity, the number of layers is evaluated by rounding of the distribution function to the nearest value. Figure 5.11e shows the boundary layers when the boundary layer thickness and the number of layers both vary on a model entity. The boundary layers parameters may also vary from one model face use to another.

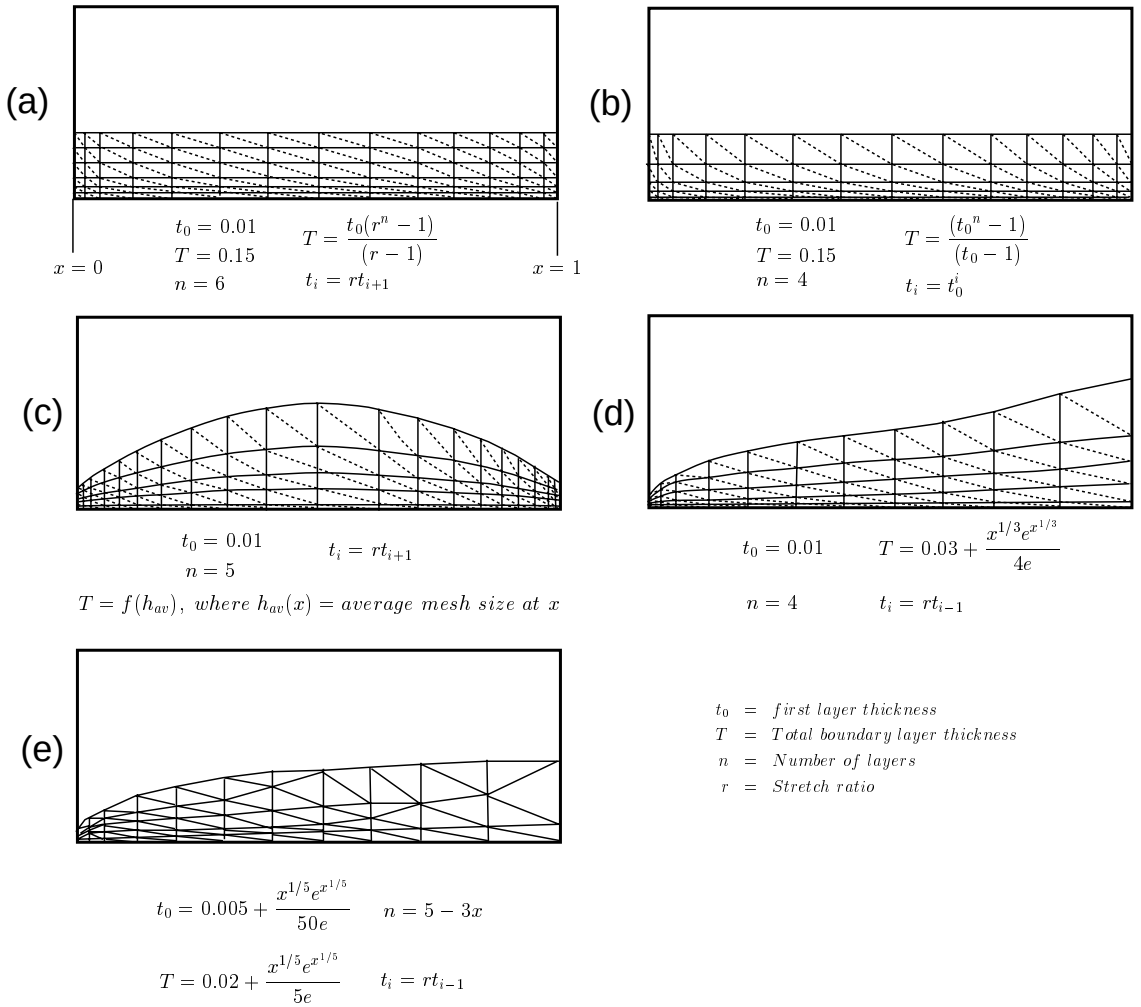


Figure 5.11: Methods of specifying boundary layers. (a) Geometric variation of layer thickness. (b) Exponential variation of layer thickness. (c) Adaptively varying boundary layer thickness. (d) Prescribed variation in boundary layer thickness. (e) Prescribed variation of boundary layer thickness and number of layers.

CHAPTER 6
BOUNDARY LAYER MESHING - ENSURING ELEMENT VALIDITY

Growth curves are created with maximum consideration for topological validity and topological compatibility with the model. However, only preliminary consideration is given to geometric validity of future elements during the point placement phase. This is because geometric invalidities in the generalized advancing layers method arises mostly due to interactions between entities of neighboring growth curves connected to form tetrahedral elements. For this reason, it is only viable to check for geometric validity after creation of all growth curves. Invalidity of boundary layer elements originates from two sources (See Figure 6.1,6.2):

1. Invisibility of nodes of a growth curve from a mesh face use

   This typically occurs at vertices where the mesh is very coarse relative to sharp changes in the model face normals, i. e., where there are sharp convexities in the model/mesh boundary representation (Figure 6.1). This is handled by recognizing these situations properly and utilizing multiple growth curves where necessary, as discussed in Chapter 5.

2. Crossover of growth curves

   This typically occurs at concavities in the model boundary either at an edge where two model faces form an acute angle (Figure 6.2a) or where the model face is curved sufficiently relative to the boundary layer thickness that the growth curves cross over (Figure 6.2b). This situation is not accounted for in the creation of the growth curves and is specifically dealt with in a separate procedure described in this chapter.

The Generalized Advancing Layers Method attempts to correct growth curve crossover by three different methods, Smoothing, Shrinking and Pruning applied in that order. In the smoothing step, a weighted laplacian smoothing procedure is

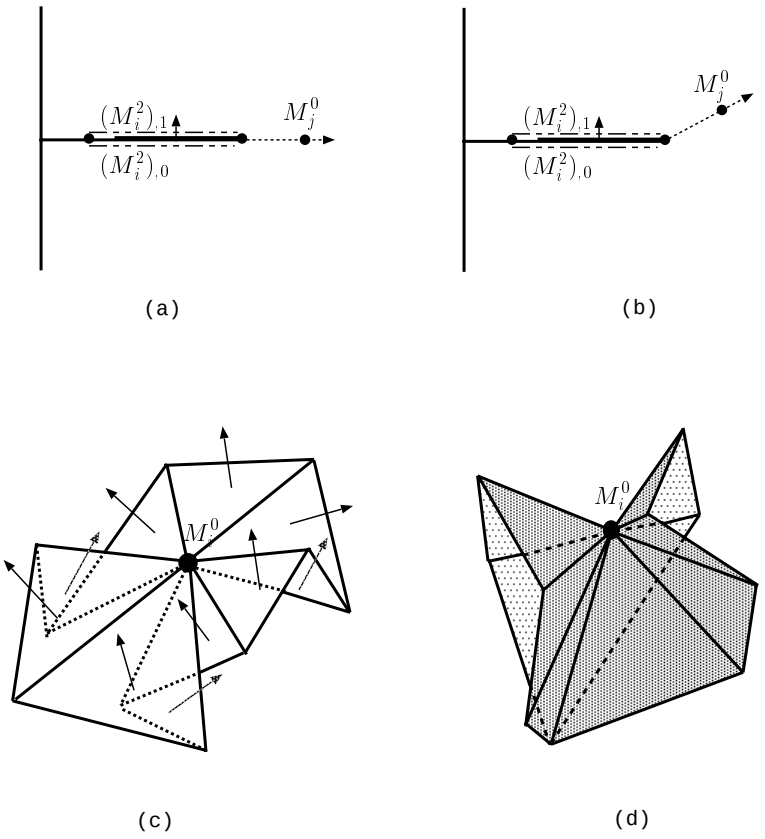


Figure 6.1: Invalid boundary layer elements due to invisibility of node. (a) 2D illustration of minimum visibility of nodes - leads to zero area elements. (b) 2D illustration of invisibility of node from surface mesh elements - leads to inside out elements. (c),(d) 3D illustration of invisibility of growth curves.

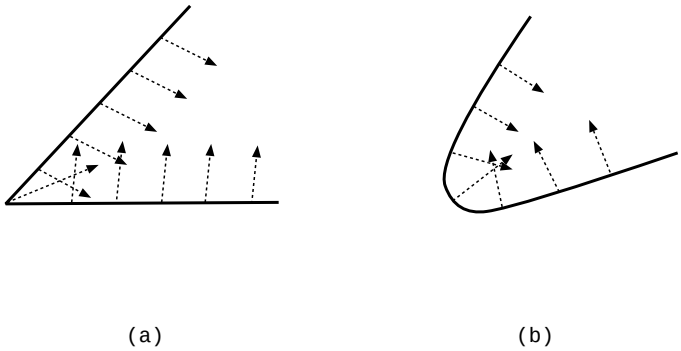


Figure 6.2: 2D illustrations of invalid boundary layer elements created by growth curve crossover due to (a) acute angle between model faces, (b) high curvature in model face geometry.

applied to growth curves to eliminate crossover. Although this distorts previously well shaped elements, it also corrects crossover in many cases. In addition it helps even out shape and size variations in the boundary layer mesh. The shrinking procedure is based on the principle that crossover often occurs because the thickness of the boundary layer is high relative to the curvature of the model face or relative to the acuteness of the angle between model/mesh faces. Therefore, the shrinking process locally reduces the thickness of the boundary layers if it will make the affected elements valid. In some very severe or degenerate case, neither smoothing nor shrinking can fix the invalidity of the elements. In such cases the growth curves of affected elements are pruned, i. e., some of their nodes are deleted, so that only valid elements are remaining. Figure 6.3 illustrates the three methods for fixing growth curve crossover in a 2D boundary layer mesh.

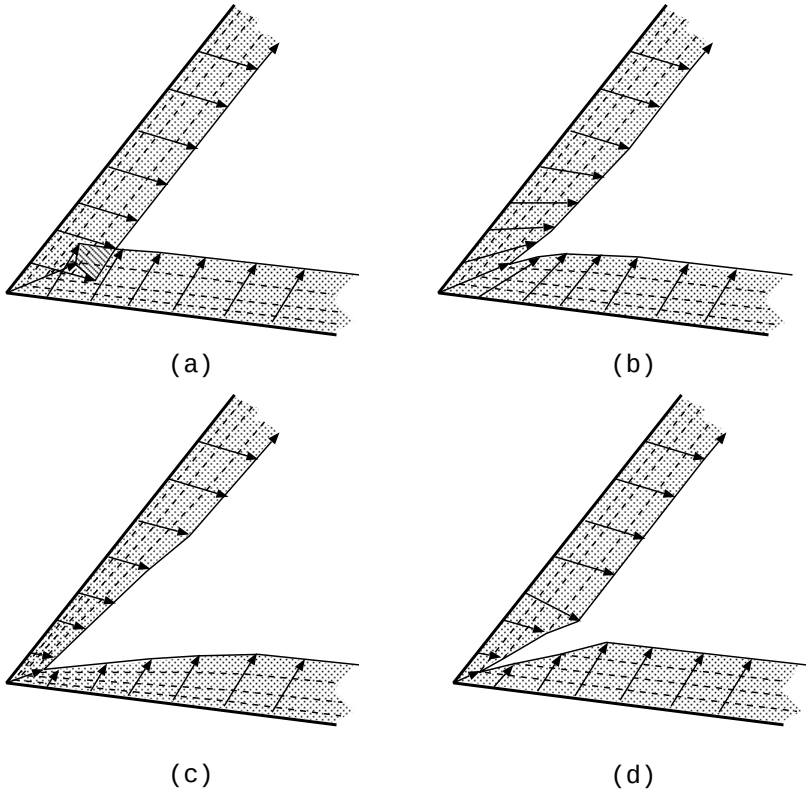


Figure 6.3: Fixing growth curve crossover in 2D mesh. (a) Invalid mesh. (b) Mesh fixed by smoothing. (c) Mesh fixed by shrinking. (d) Mesh fixed by pruning.

6.1 Adjacent Growth Curves

Adjacent growth curves of a growth curve are those that can be connected to the growth curve through a boundary layer construct such as a boundary layer quad, prism or some other general polygon or polyhedron. Given below are definitions for adjacent boundary growth curves and general adjacent growth curves. Recall that the j^{th} growth curve at a vertex, M_i^0, is C_j^i comprised of the set of nodes $\{p_j^i\}$ and segments $\{c_j^i\}$, the mesh face use set referencing the growth curve is $\{\mathcal{F}_j^i\}$ and that $\{\mathcal{F}_j^i\} \subset \{\mathcal{M}_k^i\}$ where $\{\mathcal{M}_k^i\}$ is the k^{th} mesh manifold at M_i^0.

Definition 6.1 *An adjacent boundary growth curve of a boundary growth curve* $C_{n_1}^i$, $\{c_{n_1}^i\} \sqsubset G_{k_1}^{d_1}$, $d_1 = 1, 2$ *(denoted by* $\mathcal{A}_\partial(C_{n_1}^i)$*) is either:*

1. *a growth curve* $C_{n_2}^i$ *(also at* M_i^0*) under the following conditions:*

$$
\begin{aligned}
n_1 &\neq n_2 \\
\{c_{n_2}^i\} &\sqsubset G_{k_2}^{d_2} \\
\text{if } d_1 = 1 \text{ and } d_2 = 1 &\begin{cases} \text{then } G_{k_1}^1 \cap G_{k_2}^1 = G_{k_4}^0, \text{ and} \\ \quad \exists\, G_{k_3}^2,\ G_{k_1}^1, G_{k_2}^1 \subset \partial G_{k_3}^2 \end{cases} \\
\text{if } d_1 = 1 \text{ and } d_2 = 2 &\text{ then } G_{k_1}^1 \subset \partial G_{k_2}^2 \\
\text{if } d_1 = 2 \text{ and } d_2 = 1 &\text{ then } G_{k_2}^1 \subset \partial G_{k_1}^2 \\
\text{if } d_1 = 2 \text{ and } d_2 = 2 &\text{ then } k_1 = k_2 \\
\{\mathcal{F}_{n_1}^i\}, \{\mathcal{F}_{n_2}^i\} &\subset \{\mathcal{M}_k^i\}
\end{aligned}
$$

2. *a growth curve* $C_{n_2}^j$, *under the following conditions:*

$$
\begin{aligned}
\{c_{n_2}^j\} &\sqsubset G_{k_2}^{d_2}, \quad d_2 = 1, 2 \\
\exists\, M_f^2,\ M_i^0, M_j^0 &\subset \partial(M_f^2)\,, \quad M_f^2 \subset \{\mathcal{F}_{n_1}^i\}, \{\mathcal{F}_{n_2}^j\} \\
\exists\, G_{k_3}^2,\ G_{k_1}^{d_1}, G_{k_2}^{d_2} &\subset \overline{G_{k_3}^2}
\end{aligned}
$$

The above definition means that given a fully boundary growth curve $C^i_{n_1}$, its adjacent boundary growth curves must have a classification conforming to the rules described below. If $C^i_{n_1}$ is classified on a model face, the adjacent boundary growth curve must be classified on the closure of that model face. If it is classified on a model edge, the adjacent boundary growth curve must be classified on the closure of a connected model face. In addition, the adjacent boundary growth curve must be another growth curve at the same mesh vertex or must be at another mesh vertex on the boundary of the mesh face uses referencing $C^i_{n_1}$. The concept of adjacent boundary growth curves is illustrated in Figure 6.4. In the figure, the adjacent boundary growth curves of C^1_1 are $\mathcal{A}_\partial(C^1_1) = \{C^1_2, C^3_1\}$. Note that C^2_1 is not considered adjacent to C^1_1 since G^1_1 and G^1_3 do not share a common model face.

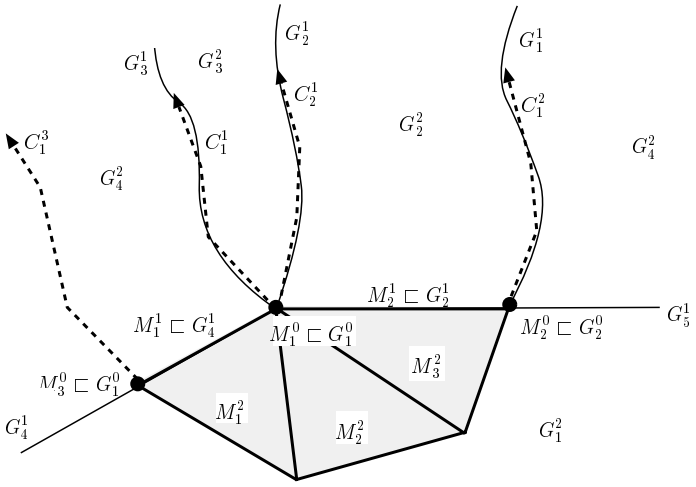


Figure 6.4: Adjacent boundary growth curves.

Definition 6.2 *An adjacent growth curve of a growth curve* $C^i_{n_1}$, $\{c^i_{n_1}\} \sqsubset G^3_{k_1}$ *(denoted by* $\mathcal{A}(C^i_{n_1})$*) is one which satisfies the following conditions:*

1. *a growth curve* $C^i_{n_2}$, *also at* M^0_i *used by the mesh face use set* $\mathcal{F}^i_{n_2}$ *under the following conditions:*

$$\begin{aligned} n_1 &\neq n_2 \\ \{c^i_{n_2}\} &\sqsubset \overline{G^3_{k_1}} \\ \{\mathcal{F}^i_{n_1}\}, \{\mathcal{F}^i_{n_2}\} &\subset \{\mathcal{M}^i_k\} \end{aligned}$$

2. *a growth curve* $C^j_{n_2}$ *under the following conditions:*

$$\begin{aligned} \{c^j_{n_2}\} &\sqsubset G^{d_2}_{k_2}, \quad d_2 = 1,2,3 \\ \exists M^2_f, \quad M^0_i, M^0_j &\subset \partial(M^2_f) \subset \{\mathcal{F}^i_j\} \\ G^{d_2}_{k_2} &\subset \overline{G^3_{k_1}} \end{aligned}$$

The above definition states that given a growth curve, $C^i_{n_1}$, its adjacent growth curves may be boundary or interior growth curves conforming the following rules. The adjacent growth curve may be another growth curve at the same mesh vertex or at another vertex. If it is at the same vertex, the mesh face uses referencing this growth curve and mesh face uses referencing $C^i_{n_1}$ must belong to the same mesh manifold. If it is at another vertex, the two vertices must be connected by an edge on the boundary of the mesh face use set of $C^i_{n_1}$. In addition, the segments of the adjacent growth curve must be classified on the closure of the model region that the first growth curve is classified on. Note that at model edges and vertices, $\mathcal{A}_\partial(C^i_n) \subset \mathcal{A}(C^i_n)$. The concept of adjacent growth curves for a general growth curves is clarified with the help of examples in Figure 6.5a,b. In Figure 6.5a, the adjacent growth curves of C^1_1 are $\mathcal{A}(C^1_1) = \{C^2_1, C^3_1, C^4_1, C^5_1, C^6_1, C^7_1\}$. Similarly, $\mathcal{A}(C^5_1) = \{C^6_1, C^1_1, C^4_1\}$. In Figure 6.5b, the vertices classified on model edge G^1_1 have two growth curves each. At M^1_0, the mesh faces uses referencing C^1_1 are indicated in a lighter shade than those that reference C^1_2. From the figure, it can be seen that

$\mathcal{A}(C_1^1) = \{C_2^1, C_1^3, C_1^4, C_1^5, C_2^5, C_1^6, C_1^7\}$. Note how the growth curves C_2^3, C_1^2 and C_2^7 are not considered to be adjacent the growth curve C_1^1.

6.2 Validity Checks for Boundary Layer Quads and Prisms

6.2.1 Validity of boundary layer quadrilateral

The validity of boundary layer quadrilaterals is checked by a series of tests performed in real and parametric spaces. First, the individual triangles of the boundary layer quad are checked in real space for zero area (within some tolerance)[4]. Then adjacent triangles are checked to see if the dihedral angle along their common edge is greater than an assumed tolerance α (taken to be $90°$). This is to measure if the discretization of the surface is excessively distorted thereby violating the principle of geometric similarity. Finally, the lateral edges of the boundary layer quad are checked for intersection in parametric space to verify that the growth curves are not crossed over. In this check, every growth curve segment is checked for intersection with all the segments of the adjacent growth curve unless an intersection is detected during the process. Since the boundary layer thickness is typically small, it is assumed that the parametric space is well behaved locally. This permits the use of straight line approximations of growth curve segments in the parametric space. For boundary layer quads which are only partially on the model face only those mesh entities that will be classified on the model boundary are tested as described above. In the future, the parametric space checks for crossover is proposed to be replaced by equivalent real space checks. In this method, the quad points will be projected onto a suitable plane in real space and the intersection checks performed there.

6.2.2 Validity of boundary layer triangles

The checks for the validity of an abstract boundary layer triangle follow the same rules as those for boundary layer quads except that the first layer in this construct is only one triangle and that an intersection check is not performed between the first segments of the two adjacent growth curves (since they already share a common vertex).

[4]Note that negative area does not have any meaning for triangles on a general surface in 3-space.

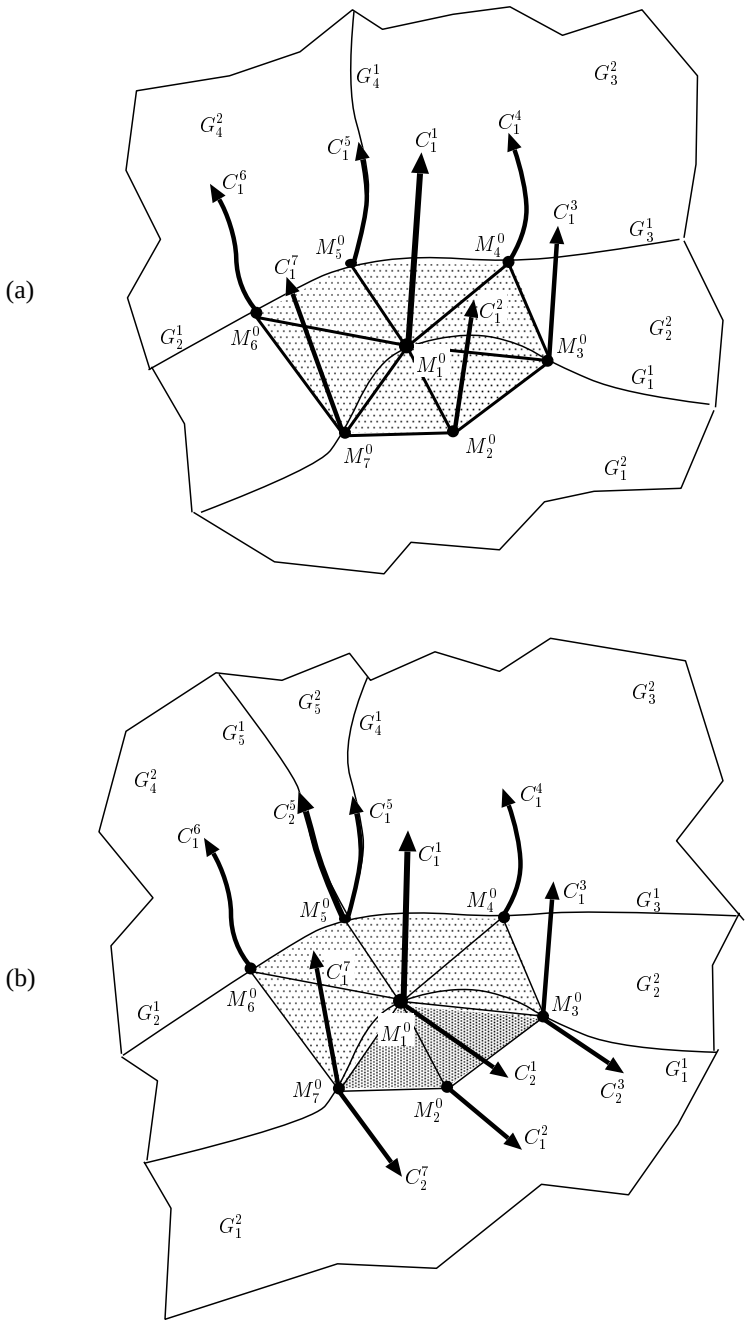


Figure 6.5: Adjacent growth curves of a growth curve, (a) without multiple growth curves. (b) with multiple growth curves.

6.2.3 Validity of boundary layer prisms

The validity of boundary layer prisms is done by checking the validity of all its component tetrahedra, i. e., all tetrahedra are checked to ensure that the worst dihedral angle is lesser than some angle, α. While minimum element validity is satisfied if $\alpha < 2\pi$, α is taken to be lesser than that value $(2\pi - \pi/30)$ in the interest of creating elements of good quality.

6.3 Smoothing Growth Curves

Smoothing of growth curves is the first tool applied in the process of eliminating growth curve crossover. The smoothing procedure applied here is a weighted laplacian smoothing procedure applied to the growth curves. It is the preferred method of eliminating growth curve crossover since it respects the original spacing of nodes along the growth curves.

Smoothing of boundary and interior growth curves is done separately on a model edge-by-edge and model face-by-face basis respectively. The reasons for this are twofold. Firstly, all boundary quads must be made valid and the model faces retriangulated with the boundary quads incorporated before growth directions are determined on the mesh vertices classified on the model face. Secondly, boundary growth directions may be general curves on model surfaces preventing a straightforward application of the laplacian smoothing technique used for straight line interior growth directions.

6.3.1 Smoothing interior growth curves

Smoothing of interior growth curves from mesh vertices classified on the closure of a model face is done by multiple passes (typically 10 passes) of a weighted laplacian scheme.

Given a growth curve C_j^i, $\{c_j^i\} \sqsubset G_r^3$ at a mesh vertex $M_i^0 \sqsubset \overline{G_f^2}$, the steps in calculating a new direction for the growth curve are as follows:

1. Get the vector, $\boldsymbol{v}_j^i$ from the first node $p_{j,0}^i$ to the top node p_{j,n_j-1}^i, where n_j is the number of nodes in C_j^i.

2. Get the adjacent growth curves, $\{\mathcal{A}(C_j^i)\}$.

3. Get the weighted average vector, $\boldsymbol{v}_{ave}^i$ of the adjacent growth curves as follows:

   (a) For each adjacent growth curve C_k^a, find the vector, $\boldsymbol{v}_k^a$ from $p_{k,0}^a$ to p_{k,n_k-1}^a, where n_k is the number of nodes in C_k^a.

   (b) For each C_k^a, find $|\boldsymbol{e}_k^a| = |\boldsymbol{p}_{n_k-1}^0 - \boldsymbol{p}_a^0|$, where $\boldsymbol{p}_i^0$ is the position vector to a point, p_i^0, of a mesh vertex M_i^0.

   (c) Calculate $\boldsymbol{v}_{ave}^i$ as

$$\boldsymbol{v}_{ave}^i = \frac{\sum_{a=1}^{n} |\boldsymbol{e}_k^a| \boldsymbol{v}_k^a}{\sum_{a=1}^{n} |\boldsymbol{e}_k^a|}$$

4. The visibility of the $\boldsymbol{v}_{ave}^i$ is checked for all the mesh faces uses referencing C_j^i. If the newly calculated direction is not visible from all the mesh face uses, it is discarded and the existing growth curve is retained.

5. The new positions of the nodes of the growth curve are then found by rotating each vector from the first node to the i^{th} by an angle equal to the angle between $\boldsymbol{v}_j^i$ and $\boldsymbol{v}_{ave}^i$:

6.3.2 Smoothing boundary growth curves

Smoothing of boundary growth curves is done differently from interior ones as the former are general growth curves whose nodes are constrained to lie on the model boundary. A boundary growth curve is allowed to be influenced only by its adjacent boundary growth curves as boundary quads must be valid before retriangulation of the face they lie on. Naturally, smoothing of boundary growth curves only applies to those growth curves that lie partly or fully on a model face and not those that lie on a model edge.

Given a growth curve C_j^i, $\{c_j^i\} \sqsubset G_f^2$ at a mesh vertex $M_i^0 \sqsubset \overline{G_e^1}$, the steps in calculating a new direction for the growth curve are as follows:

1. Get the vector, $\boldsymbol{v}_j^i$ from the first node to the $p_{j,0}^i$ to the top node p_{j,n_j-1}^i, where n_j is the number of nodes in C_j^i.

2. Get the adjacent boundary growth curves, $\mathcal{A}_\partial(C_j^i)$ as described above.

3. Get the weighted average vector, $\boldsymbol{v}_{ave}^i$ of the adjacent boundary growth curves as follows:

   (a) For each adjacent boundary growth curve C_k^a, find the vector, $\boldsymbol{v}_k^a$ from $p_{k,0}^a$ to p_{k,n_k-1}^a, where n_k is the number of nodes in C_k^a.

   (b) For each C_k^a, find $|\boldsymbol{e}_k^a| = |\boldsymbol{p}_{n_k-1}^0 - \boldsymbol{p}_a^0|$, where $\boldsymbol{p}_i^0$ is the position vector to a point, p_i^0, of a mesh vertex M_i^0.

   (c) Calculate $\boldsymbol{v}_{ave}^i$ as:

$$\boldsymbol{v}_{ave}^i = \frac{\sum_{a=1}^{n} |\boldsymbol{e}_k^a| \boldsymbol{v}_k^a}{\sum_{a=1}^{n} |\boldsymbol{e}_k^a|}$$

4. The visibility of the $\boldsymbol{v}_{ave}^i$ is checked for all the mesh faces uses referencing C_j^i. If the new direction is not visible from all the mesh face uses referencing it, it is discarded and the old direction is retained.

5. This new direction is then used to determine a new boundary growth curve as described in Section 5.6 since the new positions of the nodes cannot be directly calculated using simple transformations as done for interior growth curves.

6.4 Shrinking Growth Curves

Most growth curve crossovers in the boundary layer mesh are fixed by the smoothing procedure described above. If any crossover still remains then the growth curves are shrunk to fix this problem. Shrinking of growth curves is based on the idea that the thickness of the boundary layer mesh may be too large relative to the local curvature of model boundary or the acuteness of the angle between two adjacent model faces. In this procedure, the local thickness of the boundary layer mesh is reduced in an attempt to correct the crossover. This is accomplished by progressively reducing the node spacing of the growth curves which are connected to invalid prisms or in the case of of boundary growth curves, invalid quads. The reduction in the height of the growth curve is always accompanied by a recursive adjustment

of neighboring growth curve heights to ensure a smooth gradation of boundary layer thickness. Also, the shrinking process is constrained so that previously valid elements are not allowed to become invalid. Multiple passes (typically 5 passes) of the shrinking procedure are carried out on the boundary layer mesh to maximize the possibility of the fixing invalid quads and prisms. However, if an iteration does not produce any improvement in the prisms on that model face, then further iterations are not carried out.

6.4.1 Shrinking interior growth curves

Given a prism that is invalid, it is attempted to be fixed by shrinking each of its interior growth curves. If this does not fix the prism then pairs of interior growth curves are shrunk. If this too does not work then all the interior growth curves are shrunk at once to try to fix the prism. The shrinking of each growth curve itself is done in multiple passes using a larger reduction factor with each pass. The decrease in height at each pass is 75% and a total of 15 attempts are made with decreasing node spacing. To be acceptable, the new growth curve(s) must not only fix the prism under consideration but also maintain the status of any previously valid prisms connected to the growth curve. Also, the height of any layer is not allowed to become less than a chosen tolerance. This tolerance must be at least as large as the tolerance used for intersection checks by the isotropic volume mesher. In the implementation, it is conservatively chosen to be 10 times the intersection tolerance. If the prism cannot be made valid by shrinking, the nodes of its growth curves are restored to their original position.

Once a growth curve curve has been shrunk, a recursive procedure to adjust the heights of neighboring growth curves is applied. The ratio of the heights of two adjacent growth curves is not allowed to be greater than a factor, γ, which is designed to control the increase in the dihedral angles formed due to the change in heights of the growth curves. The procedure recurses out and adjusts as many growth curves as necessary. The recursion procedure is convergent since it is controlled by the factor γ. The actual procedure to shrink the neighboring growth curves is similar to those used for shrinking itself (Figure 6.6).

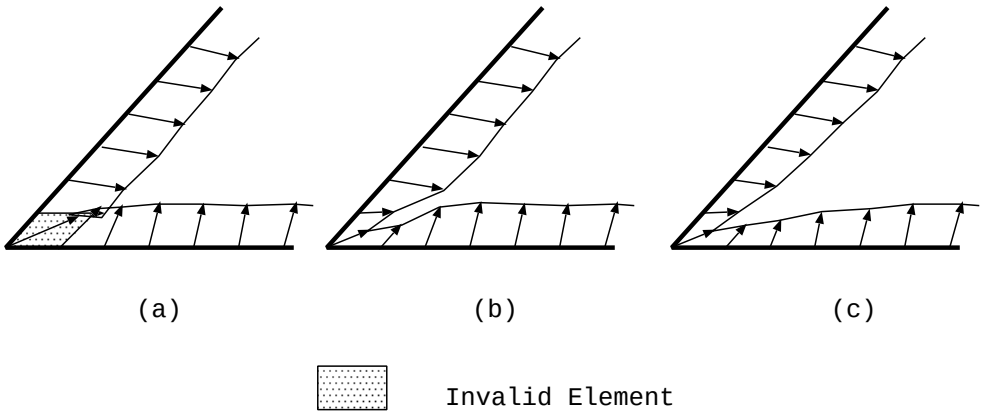


Figure 6.6: Recursive adjustment of neighbors in two dimensions to ensure smooth gradation and better element quality. (a) Invalid element formed by growth curves. (b) Element corrected by shrinking growth curves. (c) Neighboring growth curves recursively adjusted.

The model used to determine γ is a simplified two-dimensional situation shown in Figure 6.7. Consider two adjacent straight line growth curves perpendicular to the edge connecting the two vertices. Let the heights of the two growth curves be T_1 and T_2 respectively and the length of the connecting edge be L_e as shown in Figure 6.7. If we want to limit the angle θ to some value θ_{max}, then we have to control the ratio of the growth curve heights, $\gamma = \frac{T_2}{T_1}$. From the figure we can see that:

$$\begin{aligned} T_2 &= T_1 + L_e \tan\theta, \text{ or} \\ \gamma &= \frac{T_2}{T_1} \\ &= 1 + \frac{L_e \tan\theta}{T_1} \end{aligned}$$

Thus, to limit θ to θ_{max}, we must limit $\gamma \leq \gamma_{max}$; e.g., if $\theta_{max} = \frac{\pi}{4}$, then $\gamma \leq 1 + \frac{L_e}{T_1}$. For computational efficiency, an approximate value of γ is calculated at the vertex using only the root mean square value of edge lengths.

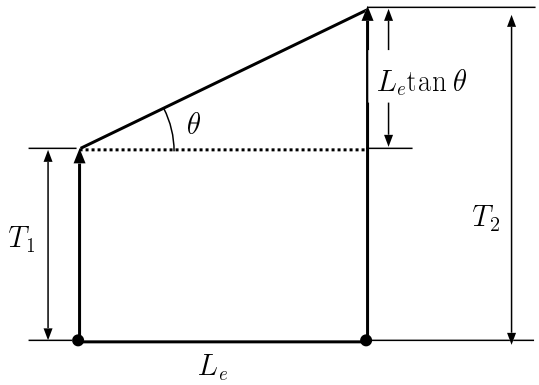


Figure 6.7: Simplified model used for calculating scale factor between adjacent growth curve heights.

6.4.2 Shrinking boundary growth curves

Shrinking of boundary growth curves is similar to the shrinking of interior growth curves except that in this case the primary concern is the validity of quads. The new growth curves are calculated as in the smoothing procedure by calculating a straight line approximation of the existing growth curve, calculating the new heights of the layers along this direction and then recomputing a new boundary growth curve as described in Section 5.6. Once one or both growth curves of a boundary quad are reduced, their neighboring boundary growth curves are recursively shrunk if necessary to obtain a smooth gradation in boundary layer heights.

6.5 Pruning Growth Curves

If both smoothing and shrinking of growth curves fail to fix all the crossover situations due to strong constraints or degenerate growth curve direction, then the nodes of the growth curve are pruned. The process of pruning growth curves eliminates as many nodes as necessary starting from the top of the growth curve (since growth curve crossover is more likely to occur in the top layers of the boundary layer mesh). As with shrinking, adjacent growth curves are also recursively pruned so that adjacent growth curves are allowed to differ by only one node. This is done to facilitate closing off the anisotropic faces of the boundary layer mesh that will

get exposed due to local deletion of some layers which will later cause difficulty for the isotropic mesher invoked to mesh the rest of the domain (See Chapter 7). At the end of the pruning procedure, the boundary layer mesh has no invalid elements. Pruning of interior growth curves is done until the remaining prisms connected to the growth curves are all valid and pruning of boundary growth curves is done until the connected boundary quads are all valid (Figure 6.8).

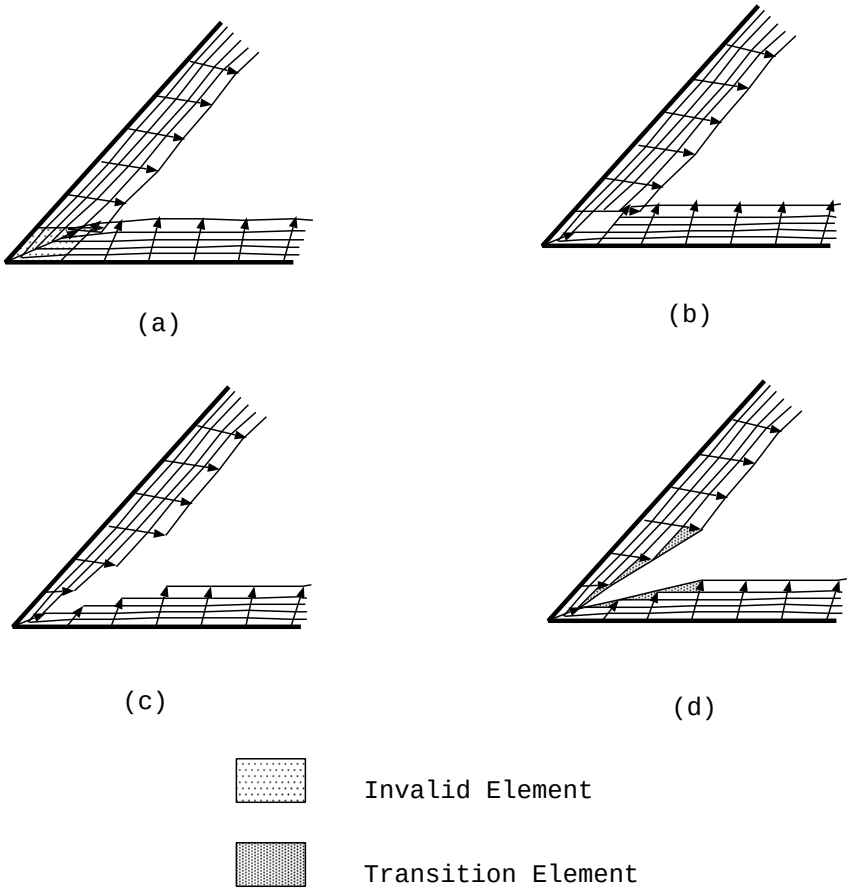


Figure 6.8: Recursive pruning of neighboring growth curves in two dimensions to ensure smooth gradation and better element quality. (a) Invalid element formed by growth curves. (b) Mesh with invalid elements deleted. (c) Neighboring growth curves recursively pruned to ensure one level difference. (d) Mesh with transition elements added to bridge steps in growth curves.

CHAPTER 7
BOUNDARY LAYER MESHING - ELEMENT CREATION

The Generalized Advancing Layers method attempts to grow anisotropic elements on all requested model faces while maintaining good element quality and shielding the isotropic mesh generator from the highly stretched faces of the boundary layer mesh. The primary construct in the creation of the boundary layer mesh is the triangular prism formed by connecting the nodes of three growth curves from the vertices of a single mesh face. Other constructs are boundary layer transition elements and boundary layer blends.

As explained in Chapter 5, multiple sets of nodes are allowed to emanate from a single mesh vertex. The presence of multiple growth curves allows adjacent prisms to be separated from each other. This reduces the distortion of the prisms, a necessary condition for good quality of the tetrahedra formed by subdividing the prism. The separation of adjacent boundary layer prisms leads to the formation of gaps in between the prisms. The walls of these gaps consist of highly stretched faces of the anisotropic mesh and must be shielded from the isotropic mesh generator; otherwise the isotropic mesher tends form poorly shaped elements in their neighborhood and also suffers in reliability. It is proposed that these gaps will be filled by constructs referred to as boundary layer blends similar to the blends used in geometric modeling to round off sharp corners (See Figure 7.1 which is reproduced here from Chapter 5 for easy reference). Boundary layer blends may occur at mesh edges or at mesh vertices. In principle, boundary layer blends may or may not have fixed number of elements along the model edge. Variable blend constructs typically occur at model edges where the dihedral angle between the connected mesh faces is changing along the edge. While boundary layer blends at mesh edges are easy to mesh using templates since only two prisms contribute to their boundary, boundary layer blends at mesh vertices are harder since an arbitrarily large number of prisms and blends incident on the mesh vertex may contribute to the polyhedral cavity

to be meshed. Therefore, such cavities must be meshed by more general meshing procedures tailored for this purpose (Figure 7.1d).

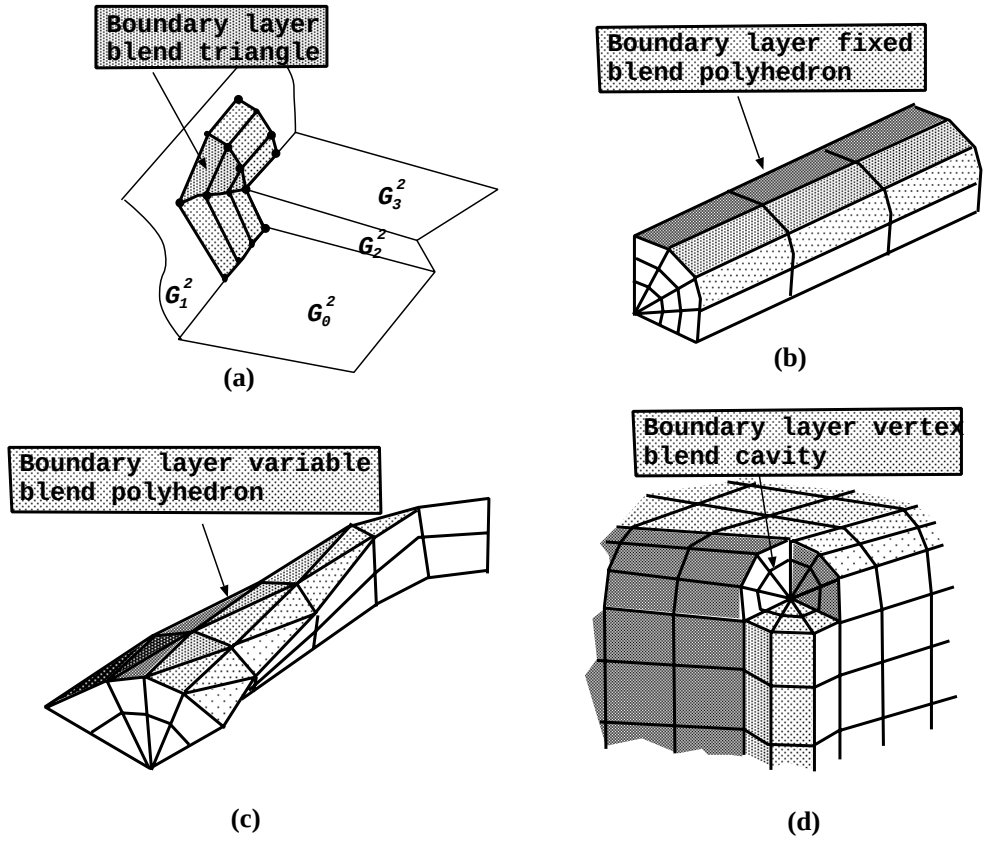


Figure 7.1: Boundary layer blend elements.

When the growth curves of a mesh face forming a prism have different number of nodes, a step is formed in the boundary layer mesh. This too exposes stretched faces to the isotropic mesh generator. The difference in the number of nodes in the growth direction comes from user requested mesh attributes, deletion of nodes due to invalidity of elements or deletion of elements to avoid intersection of boundary layers (Chapter 8). The step in the boundary layer mesh is formed since boundary layer quads and prisms can be formed only by connecting nodes of growth curves at equal levels. To avoid leaving highly stretched faces of the step exposed to the volume mesher, tetrahedra are created to bridge the the growth curves with more nodes to those with less nodes (See Figure 7.2). Since the process of recursively adjusting the number of nodes on growth curves after pruning them tries to enforce a one layer

difference between adjacent growth curves, transition tetrahedra generally span only one layer. However, as will be described in Chapter 8, multiple level difference may be created between adjacent growth curves during pruning of growth curves to fix self-intersections. Therefore, the capability to create multiple level transition elements also exists to be used when necessary. The idea of a one level transition element is similar to the procedure used in the work of Connell and Braaten [11] to phase out the boundary layer at some edges.

All of the boundary layer constructs described above are allowed to abut a model face and therefore modify the model face triangulation. The equivalent boundary layer constructs for the boundary layer prisms are boundary layer quads, boundary layer blend triangles and boundary layer transition triangles.

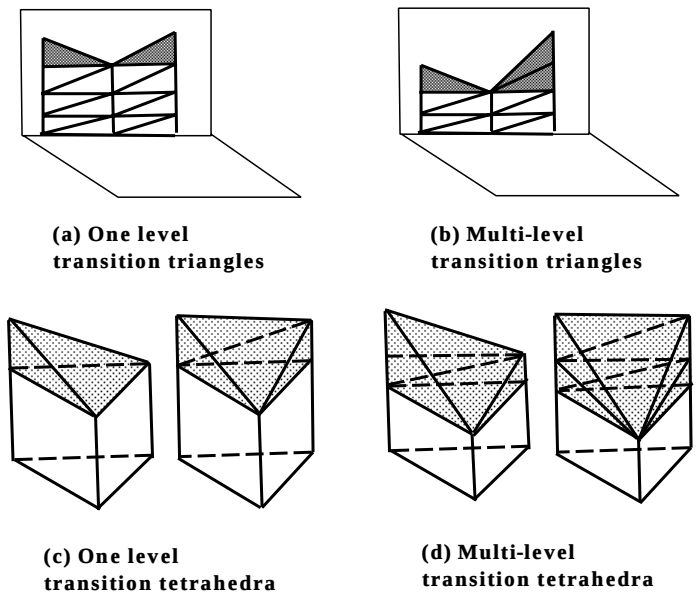


Figure 7.2: Boundary layer transition elements.

Element creation in the generalized advancing layers procedure is done in the following steps:

1. Growth curves are first determined at mesh vertices classified on model vertices. If any of these growth curves lie partly or fully on a model edge, the boundary layer entities classified on the model edges are created.

2. Boundary layer mesh entities classified on model edges are incorporated into the model edge discretization.

3. Growth curves are determined at mesh vertices classified on model edges. Boundary layer entities from these growth curves, and growth curves at model vertices that are classified on model faces are created.

4. Growth curves on the model boundary are combined to form boundary layer quads, boundary layer transitions and boundary layer blends. First, corresponding nodes of adjacent boundary growth curves from neighboring vertices are combined to form quads lying partially or fully on model faces. If a level difference exists between the growth curves, transition triangles are formed on top of the quads. Finally, blends are formed between appropriate multiple boundary growth curves at mesh vertices.

5. Boundary layer quads, transitions and blends lying on model faces are incorporated into the triangulations of model faces.

6. Growth curves are determined at mesh vertices classified on model faces. The entities of these growth curves and growth curves from model vertices and model edges that are classified in the interior are created.

7. The interior growth curves are connected up to form prisms, transitions and blends. First, adjacent growth curves from neighboring vertices are connected up to form prisms. If a level difference exists between growth curves forming a prism, then transition tetrahedra are formed atop the prisms to bridge the step. Finally, blends are created between the multiple growth curves at mesh vertices.

7.1 Conversion of Growth Curves into Boundary Layer Mesh Entities

The creation of boundary layer mesh entities from growth curves is done in one step if all the nodes of the growth curve are classified on only one entity. If the growth curves that are partly boundary and partly interior are permitted, then

the parts of the growth curve classified on a model edge, model face and model region can be converted into mesh entities during model edge retriangulation, face retriangulation and region triangulation respectively.

Nodes of growth curves are directly converted into mesh vertices with their classification being derived from the growth curve node classification. On the other hand, classification of the growth curve segment is not explicitly stored. Therefore, when converting growth curve segments to mesh edges, their classification has to be derived from the classification of the mesh vertices. Given two mesh vertices of a boundary layer mesh edge representing a growth curve segment, the classification of the mesh edge is determined by finding the lowest order model entity common to the two entities that the mesh vertices are classified on. In case there is more than one common entity that the edge can be classified on, an additional check is performed. In this additional step, the midpoint of the straight line approximation of the mesh edge in parametric space is checked if it maps to a point in the interior of one the candidate model entities. The edge is then classified on the entity that satisfies this condition.

7.2 Model Edge Retriangulation

The insertion of boundary layer mesh edges and vertices classified on model edges is carried out through local mesh modification operators [35] (Also see Appendix A). Given an edge to be inserted into the discretization of a model edge, existing mesh edges overlapping the edge to be inserted are identified by examining the one-dimensional parametric space of the model edge. The end vertices of the edge to be inserted are introduced into the model edge discretization by an edge split if coincident vertices do not already exist in the mesh. Then all edges overlapping the edge to be inserted are collapsed out into a single edge. Finally the edge to be inserted is merged with the duplicate edge in the mesh.

7.3 Triangulation of Boundary Layer Quads

Boundary layer quads are formed by connecting nodes of adjacent growth curves *not* originating from the same mesh vertex. Given two adjacent growth

curves, $C^{i_1}_{j_1}$ having n_1 nodes and $C^{i_2}_{j_2}$ having n_2 nodes, $n-1$ boundary layer quads are formed ($n = min(n_1, n_2)$). For each layer l, a boundary layer quad is formed by connecting the nodes $p^{i_1}_{j_1,l}$, $p^{i_1}_{j_1,(l+1)}$, $p^{i_2}_{j_2,l}$ and $p^{i_2}_{j_2,(l+1)}$. In converting these boundary layer quads to triangles the choice of the diagonal is dictated by the future validity of the connected prisms. For reasons explained in the section describing prism tetrahedronization (Section 7.7 below), the diagonal is made so that it connects node l of the growth curve at the mesh vertex with a lower identifying number (vertex ID) to node $l+1$ of the growth curve at the other vertex. If one of the growth curves has more nodes than the other, the transition triangles are formed on top of the quads as explained below.

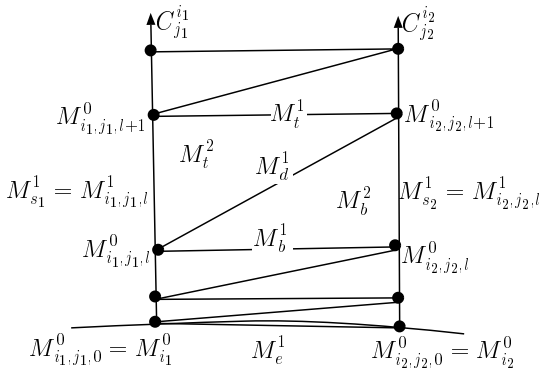


Figure 7.3: Boundary layer quad triangulation template.

The algorithm for constructing a boundary layer quad between the n nodes of the growth curves $C^{i_1}_{j_1}$ and $C^{i_2}_{j_2}$ is given below (See Figure 7.3). In the algorithm, it is assumed that mesh entities have been created from the two growth curves and that $ID(M^0_{i_1}) < ID(M^0_{i_2})$ (where $ID(M^0_i)$ is the identification number assigned to each mesh vertex) which implies that for a layer l, the diagonal of the quad goes from $p^{i_1}_{j_1,l}$ to $p^{i_2}_{j_2,(l+1)}$. The steps of the algorithm are:

1. The bottommost edge of the boundary layer quad is M^1_e, where $M^0_{i_1}, M^0_{i_2} \subset \partial M^1_e$. Make this the bottom edge of the bottommost quad.

2. For each layer l, $l = 1, n-1$, do the following (Refer entities of layer l in Figure 7.3):

(a) Get the bottom mesh edge of the quad, M_b^1, between mesh vertices $M^0_{i_1,j_1,l}$ and $M^0_{i_2,j_2,l}$.

(b) Get the side mesh edges, $M_{s_1}^1 = M^1_{i_1,j_1,l}$ and $M_{s_2}^1 = M^1_{i_2,j_2,l}$ of the quad from the two growth curves $C_{j_1}^{i_1}$ and $C_{j_2}^{i_2}$ respectively.

(c) Create the diagonal edge, M_d^1 between vertices $M^0_{i_1,j_1,l}$ and $M^0_{i_2,j_2,(l+1)}$ for this layer. The classification of the edge is derived from its end nodes.

(d) Figure out the classification of the lower triangular face M_b^2 (where $M_b^1, M_{s_2}^1, M_d^1 \subset \partial M_b^2$) of the quad from its 3 edges.

(e) If $M_b^2 \sqsubset G_r^3$, then the orientation of the face is not important as long it is taken into account by the regions that use it. On the other hand, if $M_b^2 \sqsubset G_f^2$, then the face must be oriented in the same direction as the model face[5]. In this case, the orientation of the mesh face is determined by examining M_b^1.

The natural definition of the face is such that its edges are ordered as M_b^1, $M_{s_2}^1$, M_d^1 used in directions defined by the fact that the vertices around the face starting from $M^0_{i_1,j_1,l}$ are the ordered cyclic set $[M^0_{i_1,j_1,l}, M^0_{i_2,j_2,l}, M^0_{i_2,j_2,(l+1)}]$.

- If $M_b^1 \sqsubset G_f^2$, then the other mesh face, M_p^2 connected to the edge and classified on G_f^2 is found, if it exists. If such a face exists, then M_b^2 uses M_b^1 in a direction opposite to the direction in which M_p^2 uses M_b^1 (See Figure 7.4a). If the direction of use conflicts with the selected direction of use of M_b^1 then the orientation of the face is reversed and the ordered set of edges and vertices around the natural direction of the face becomes $[M_b^1, M_d^1, M_{s_2}^1]$ and $[M^0_{i_1,j_1,l}, M^0_{i_2,j_2,(l+1)}, M^0_{i_2,j_2,l}]$ (Figure 7.4b).
- If $M_b^1 \sqsubset G_e^2 \subset \partial G_f^2$, then it is assumed that the mesh edge has the same direction as the model edge (i. e. the first vertex of the

[5]This is an assumption that is enforced throughout all the mesh generation tools described or referenced here. While this is not a necessary condition for successful mesh generation, it make the procedures much simpler without adversely affecting the reliability of the mesher or the quality of the generated meshes

edge has a lower parameter with respect to the model edge than the second except when it spans a periodic boundary). In this case the direction of use of G_e^1 by G_f^2 is found and M_b^2 must use M_b^1 in the same direction (Figure 7.4c,d).

In the special case when G_f^2 uses G_e^1 both ways, first a check is made for a mesh face $M_p^2 \sqsubset G_f^2$ already connected to the edge. If such a face exists, then the orientation of the current face is checked in the same way that it was done for mesh edges classified on the model faces. If not, a geometric check is performed to see if the direction of the face is right. To do this a mesh face use using the two growth curves of the boundary layer quad is found and a virtual region constructed using the vertices of the M_b^2 and the node of the mesh face use opposite the base edge M_e^1, $M_{i_1}^0, M_{i_2}^0 \subset \partial M_e^1$ of the boundary layer quad. If the volume of this region is positive, the orientation of M_b^2 is correct; if not, it must reversed (See Figure 7.4e).

(f) Create the top edge of the boundary layer quad for this layer, M_t^1, $M_{i_1,j_1,(l+1)}^0, M_{i_2,j_2,(l+1)}^0 \subset \partial M_t^1$, deriving its from the classification from its end nodes.

(g) Figure out the classification of the upper face M_u^2, $M_t^1, M_{s_1}^1, M_d^1 \subset \partial M_u^2$ of the quad from its 3 edges. Reverse its orientation, if necessary, as was done for the lower face using the diagonal edge as a reference edge. Thus in its natural orientation, the ordered set of edges and vertices of M_u^2 are $[M_t^1, M_{s_1}^1, M_d^1]$ and $[M_{i_2,j_2,(l+1)}^0, M_{i_1,j_1,(l+1)}^0, M_{i_1,j_1,l}^0]$ respectively. In the reversed orientation, these sets become $[M_t^1, M_d^1, M_{s_1}^1]$ and $[M_{i_2,j_2,(l+1)}^0, M_{i_1,j_1,l}^0, M_{i_1,j_1,(l+1)}^0]$ respectively.

(h) Label M_t^1 as M_b^1.

An important point to note here is that although the current implementation has precluded the existence of partially boundary and partially interior growth curves, a special type of quad with some faces classified on the boundary and others in the interior may still exist in the boundary layer mesh. This happens under the

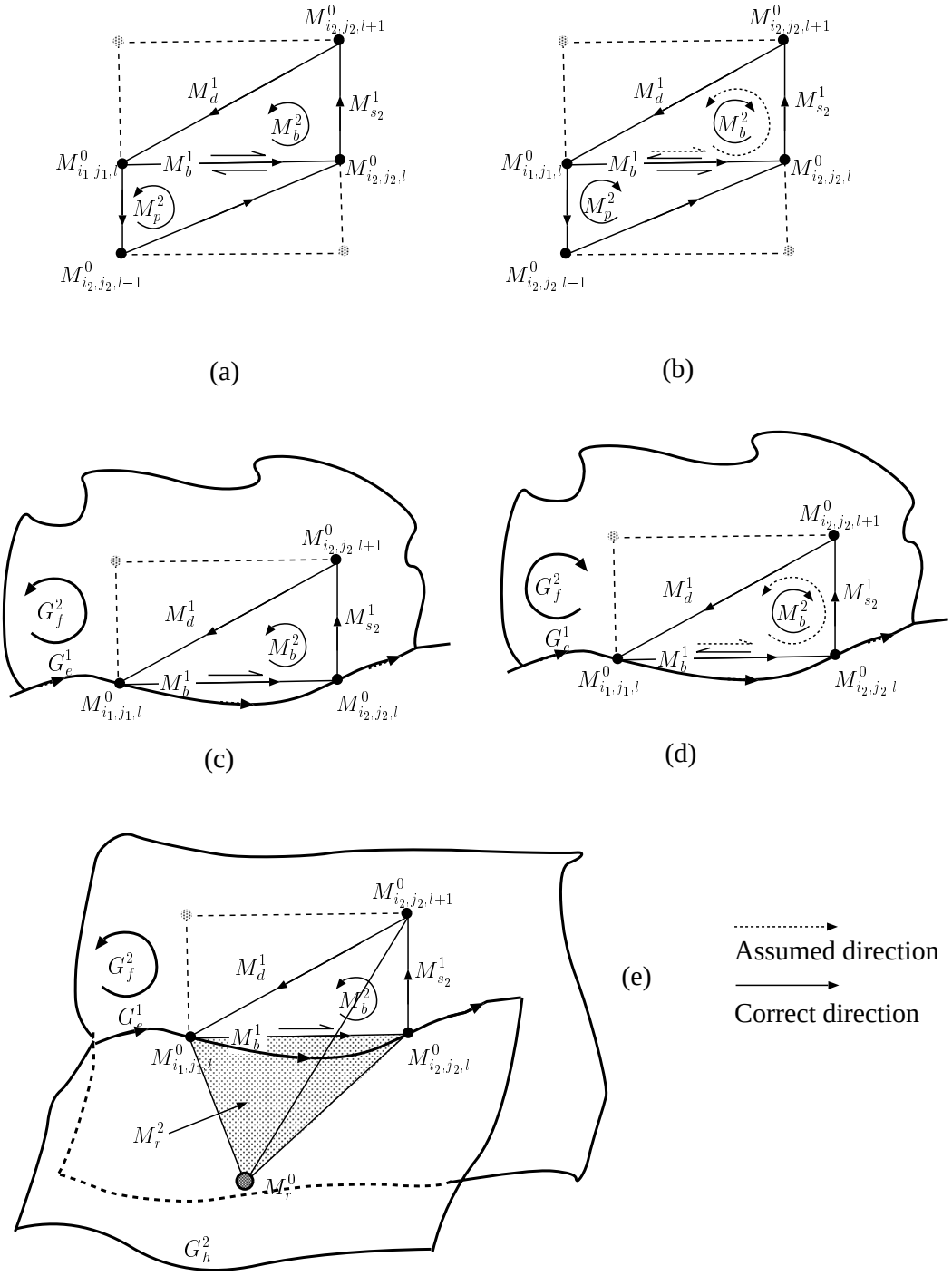


Figure 7.4: Determining face directions for boundary layer quad triangles. (a) Bottom edge classified on model face and assumed face direction is correct. (b) Bottom edge classified on model face and face direction must be reversed. (c) Bottom edge classified on model edge and assumed face direction is correct. (d) Bottom edge classified on model edge and face direction must be reversed. (e) Geometric check when bottom edge classified on model edge used twice by the model face.

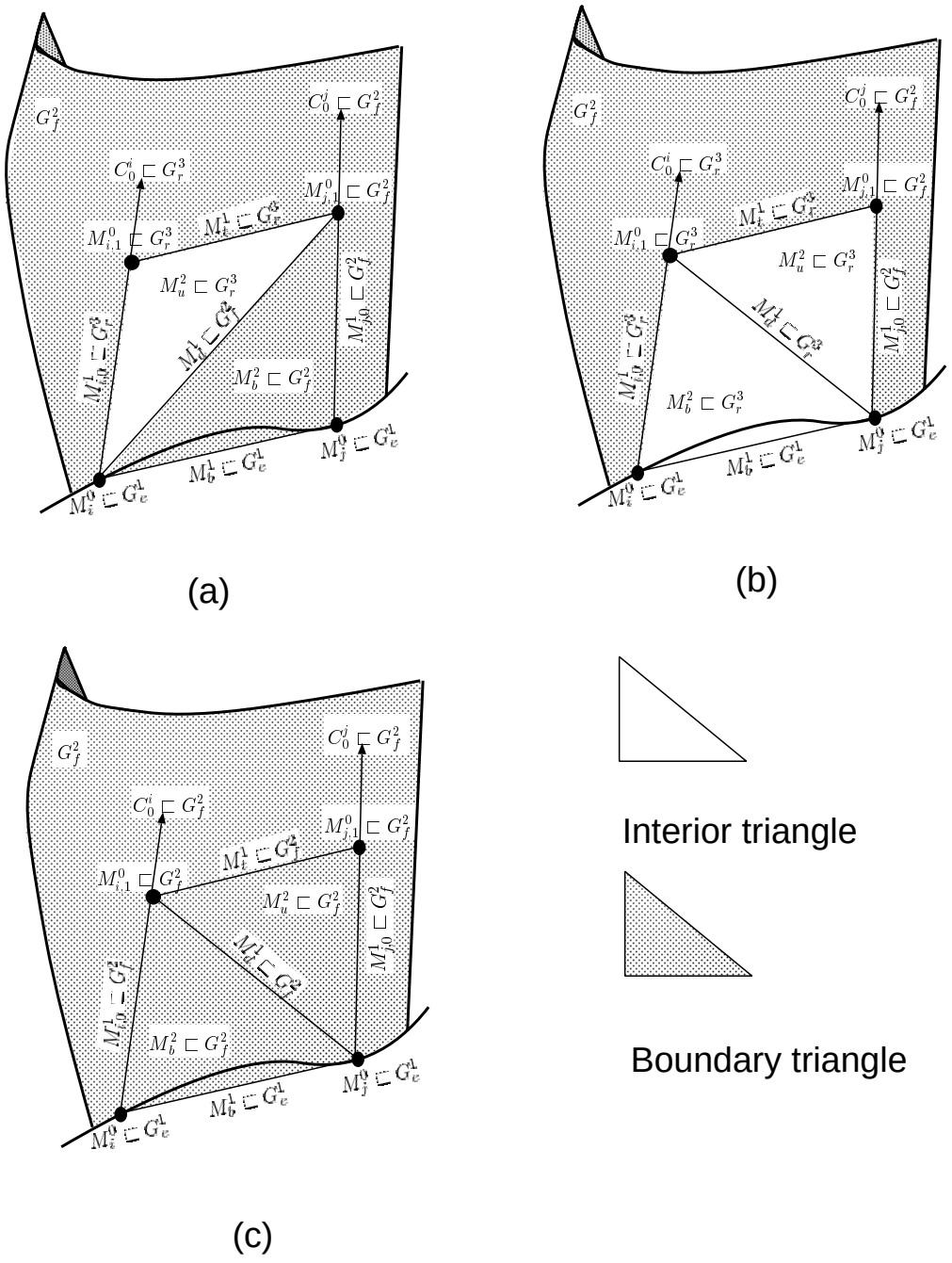


Figure 7.5: Types of quads at model edges. (a) Quad with partly boundary and partly interior classifications. (b) Quad with all interior classification. (c) Quad with all boundary classification.

particular circumstance in which the one growth curve is fully interior and the other is fully boundary and the diagonals of the quad have their lower vertices on the interior growth curve. This is illustrated in Figure 7.5. In the figure, $M_i^0, M_j^0 \sqsubset G_e^1$ and $C_0^i \sqsubset G_r^3$, $C_0^j \sqsubset G_f^2$. In the Figure 7.5a, $ID(M_i^0) < ID(M_j^0)$. Therefore, by the methods described above, in the first layer $M_d^1 \sqsubset G_f^2$ and $M_t^1 \sqsubset G_r^3$. As a result of the classifications of their component edges, the bottom face of the lower triangle in the first layer is classified on model face G_f^2 and the top triangle on model region G_r^2. Note that only the first layer triangles of such boundary layer quads have different classifications and the rest of the triangles are classified interior. In Figure 7.5b, $ID(M_j^0) < ID(M_i^0)$ and therefore, all the triangles are classified as being interior. In Figure 7.5c, both growth curves are classified on the same model face and therefore it is immaterial what the IDs of the vertices are - all triangles are classified on the model face.

When quads are partly boundary and partly interior, the boundary triangles are created first and inserted into the surface mesh and the interior triangles are completed later when other fully interior quads are being created. Although it is true that the procedure can avoid dealing with the quads with multiple classifications in most cases by simply exchanging the ID numbers of vertices, it is not guaranteed that this type of quad will never be needed since neighboring quads get affected by the change in IDs. In addition, the availability of such mechanisms in the mesh generator provides the basis for handling more general types of quadrilateral abstractions in future implementations.

7.4 Creation of Boundary Layer Transition Triangles

Transition triangles are formed atop boundary layer quads with a level difference between the two growth curves. This is done by simply connecting the top node of the growth curve with fewer nodes with nodes of the other growth curve which are at a higher level. It is important to note that the mesh vertex with the lower ID may have the growth curve with more nodes and therefore, diagonal edge of the transition triangle may go in an opposite direction to the diagonals of the boundary layer quad (Figure 7.2a,b).

7.5 Creation of Boundary Layer Blend Triangles

Creation of boundary layer blend triangles is similar to the creation of boundary layer quads. The difference is that boundary layer blend triangles establish connections between nodes of two growth curves originating from the same mesh vertex. The first layer of a blend triangle is made up of a single triangular mesh face and the rest of the layers contain triangulated quads as in the case of the boundary layer quad. The direction of the diagonal for the quads in a boundary layer triangle is arbitrary and may be based solely on the quality of the triangles (Figure 7.1a).

7.6 Model Face Retriangulation

The choice of the method used for the incorporation of mesh entities classified on model boundaries is very important to the reliability of the boundary layer mesh generator. While the use of an advancing front type method to retriangulate the model faces is attractive, it requires the use of the parametric space of the model faces to check for intersections. This is an error prone procedure if the parametric space is highly distorted since the underlying assumption is that straight line approximations of the parametric curves representing mesh edges can be used to check for intersections. In fact, examples exist of commercial modelers using highly distorted parametric spaces for model faces constructed from a existing set of facets. An alternative is to create the boundary layer mesh entities on model edges and then complete the triangulation of the model edges, create entities classified on the model faces and then complete the triangulation of the faces and finally create the interior boundary layer mesh entities before handing the mesh over to a volume mesher. This may be convenient or difficult to do depending on the approach of the isotropic mesher and may require tight integration with the sub-components of the surface mesh generator. To avoid the use of the parametric space for intersection checks and to keep the surface meshing, boundary layer meshing and volume meshing independent and modular, a third alternative is adopted here. This method uses local mesh modification operators combined with checks for the smoothness of the surface discretization to incorporate boundaries of the boundary layer mesh

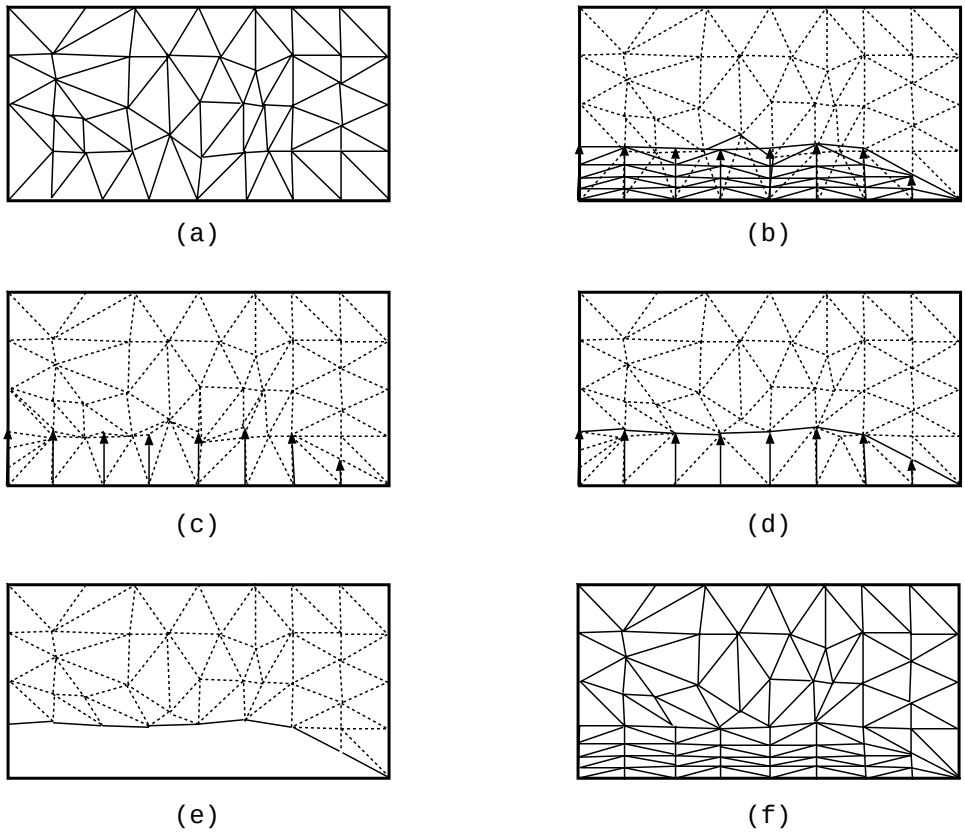


Figure 7.6: Model face retriangulation by local mesh modifications. (a) Initial surface mesh. (b) Surface mesh with boundary layer elements overlayed. (c) Insertion of outermost boundary layer vertices into surface mesh. (d) Recovery of outermost boundary layer edges by edge swapping and edge collapsing. (e) Deletion of surface mesh triangles overlapping the boundary layer mesh. (f) Incorporation of boundary layer mesh into surface mesh.

into the existing surface triangulation. It then replaces overlapping triangles of the underlying mesh with the boundary layer triangles [35].

The model face retriangulation procedure is done for each model face that the growth curves of each model edge affect. Given a model edge and a model face on which growth curves from mesh vertices of the model edge lie, the following steps are carried out to create and incorporate the boundary layer mesh into the surface mesh triangulation:

1. Boundary layer quads and triangles classified on the the model face are created as described above (Figure 7.6b).

2. The boundary layer quads and triangles are temporarily disconnected from the mesh entities of the original face triangulation. This is done to prevent the procedures recovering the boundary layer edges from being misled by the presence of the boundary layer mesh entities in the discretization of the model face.

3. Each boundary layer mesh entity that forms the outer boundary of the set of boundary layer mesh faces classified on the model face is incorporated into the surface mesh by the edge recovery procedure. The recovery procedure is briefly described below and discussed in full detail in [35]. Once all the necessary edges have been recovered, the outer boundary of the set of faces to be inserted into the mesh exactly matches the outer boundary of a set of faces in the underlying surface mesh (Figure 7.6c,d).

4. Mesh faces of the existing surface triangulation overlapping the boundary layer mesh faces are deleted (Figure 7.6e). The set of faces overlapping the boundary layer mesh faces are found using only topological checks and searches.

5. The boundary layer faces are incorporated into the surface mesh in place of the deleted elements (Figure 7.6f).

The edge recovery procedure first identifies where to insert the vertices of the edge to be recovered into the existing mesh. The insertion point may be an existing vertex (in which case no explicit insertion of the vertex is required), a mesh edge or and mesh face. The mesh entity to be modified for insertion of the vertex or in other words, the mesh entity "containing" the vertex to be inserted is first localized using the parametric space. However, the final decision is made with real space checks since the results of containment checks in the parametric space can be misleading for highly distorted spaces. The criterion for finding the mesh entity to split for insertion is based on the observation that a good choice for an insertion entity is one that will preserve or improve the approximation of the true geometry by the surface mesh since it is guaranteed that the point to be inserted will lie on the true geometry. Therefore, the methodology used projects the insertion point onto a local set of mesh faces along their respective normals and among the mesh faces containing

the projected point, the one that maintains or improves the approximation of the true geometry is chosen. The dihedral angles between mesh faces before and after insertion of the new point is a good measure for assessing the choice of the mesh entity to be split, the alternative being to use one of many forms of comparison between the discrete normals (normals of mesh faces) before and after the split. The point is inserted into the mesh either by merging with an existing vertex, by an edge split or by a face split operation (See Appendix A).

Once the vertices to be recovered are inserted into the mesh, a path of edge connected mesh faces is found from one vertex to another. Once again this is done by projection of the edge to be recovered onto planes of the mesh faces in the neighborhood. If any mesh vertices lie in the path of the edge to be recovered, they are either collapsed along a connected edge or perturbed. If the collapses have not already recovered the edge, it is recovered by successive swaps of mesh edges of the faces in its path.

The edge split and face split operations used to insert the vertices of the edge to be recovered often cause poorly shaped elements to be formed. The nearly flat elements then cause numerical imprecision in the remaining steps of the recovery algorithm thereby reducing its reliability. Two strategies may be used to eliminate the creation of such poorly shaped elements. The first is to use an increased point tolerance for checking if point lies on a vertex or an edge. This prevents points from being inserted too close to an existing vertex or an edge. Care has to be taken in this approach to ensure that use of the increased tolerance does not cause a single mesh vertex to capture both endpoints of the edge. Also, if the point is declared to be coincident with a mesh vertex then the vertex must be repositioned within the real point tolerance of the insertion point. The second approach is to insert the point using the real point tolerance of the modeler and then perform edge swaps and edge collapse in the immediate neighborhood to eliminate the poorly shaped mesh faces. This method has the disadvantage that it is harder to estimate how well the mesh will approximate the true geometry after the insertion and modification operations.

Once the edge is recovered successfully, a local mesh optimization is performed to eliminate any other poorly shaped faces created during the edge recovery process. The optimization procedures are constrained to not modify any existing boundary layer edges already recovered or any edges classified on model edges. If the edge cannot be recovered then the procedures split edges of the mesh in its path and return a list of edges forming a path of edges from one vertex of the original edge to be recovered to the other. In such a case, the boundary growth curve is deleted and replaced with an interior growth curve, in effect peeling the boundary layer away from the adjacent wall in the neighborhood. This requires deletion of some existing boundary layer quads, recalculating one or two growth curves, insertion of newly exposed edges and updating of data structures. Alternately, it is possible to go back to the triangulation of the face on which the boundary layer is grown, split the edge from which the boundary layer quad originates and regrow the boundary layer from the new vertices. This way the exposed edges of the new quads will match the mesh edges that the edge recovery algorithm actually created in place of what was earlier requested.

Finally, after the model face retriangulation procedure is completed, the resulting mesh is subjected to a series of checks to ensure that it is not self intersecting [13]. If an intersection is found it is fixed by edge splits of unconstrained entities (i. e. entities that do not belong to the boundary layer mesh and are not constrained in any other way). This step is necessary to ensure that the final mesh that is handed over to the volume mesher is not self-intersecting.

7.7 Creation of Boundary Layer Prisms

The bulk of the elements in the boundary layer mesh are comprised of tetrahedronized layers of boundary layer prisms. Boundary layer prisms are grown on mesh face uses by connecting the three growth curves at the face vertices which share each mesh face use. The tetrahedronization of each boundary layer prism in a layer gives rise to three tetrahedra. The number of layers of such elements grown atop any mesh face use is determined by the growth curve forming the smallest number of nodes.

Boundary layer prisms can be thought of as being formed by three quadrilaterals that are grown from the edges of the mesh face. There are eight possible combinations of diagonals for the quads of a prism. Of these only six are configurations which can be tetrahedronized without the insertion of any new points inside the prism (Figure 7.7a). Therefore, in assigning directions for the diagonals of the quads in the boundary layer mesh, care must be taken not to assign directions such that some prisms cannot be tetrahedronized. This is done by a simple algorithm based on numbering of the surface mesh vertices. Given a surface mesh with any arbitrary assignment of unique numbers (IDs) for the mesh vertices, the IDs of vertices of a face in either clockwise or counterclockwise direction cannot be strictly increasing or strictly decreasing. In other words, the ID of one vertex in each mesh face has to be lesser than the IDs of the other two vertices. Using this notion, the diagonals of boundary layer quads are constrained to go from the lower node of the growth curves of vertices with a lower ID to the upper node of growth curves of vertices with a higher ID. This is shown in Figure 7.7b.

The tetrahedronization of boundary layer prisms is done using templates. The six configurations for the boundary layer templates are reduced to just two by always tetrahedronizing the prism using the face vertex with the lowest ID as a reference vertex. The face vertex with the lowest ID is also the face vertex which has two prism diagonals connected to it. The two templates are shown in Figure 7.8. Wherever necessary, boundary layer prisms utilize the boundary layer quads previously incorporated into the model face triangulation.

The process of creating boundary layers prisms in the interior is comprised of three steps. First all growth curves classified as interior are converted into mesh entities (edges and vertices). Also, the interior portions of growth curves classified partly on the boundary and partly interior must converted into mesh entities. Next, boundary layer quads are grown from all mesh edges classified on model faces as done for mesh edges classified on model edges. In this step partially created boundary layer quads are also fully converted into mesh entities. Finally, the boundary layer quads from mesh faces are suitably combined according to the two templates described above into boundary layer prisms and their component tetrahedra.

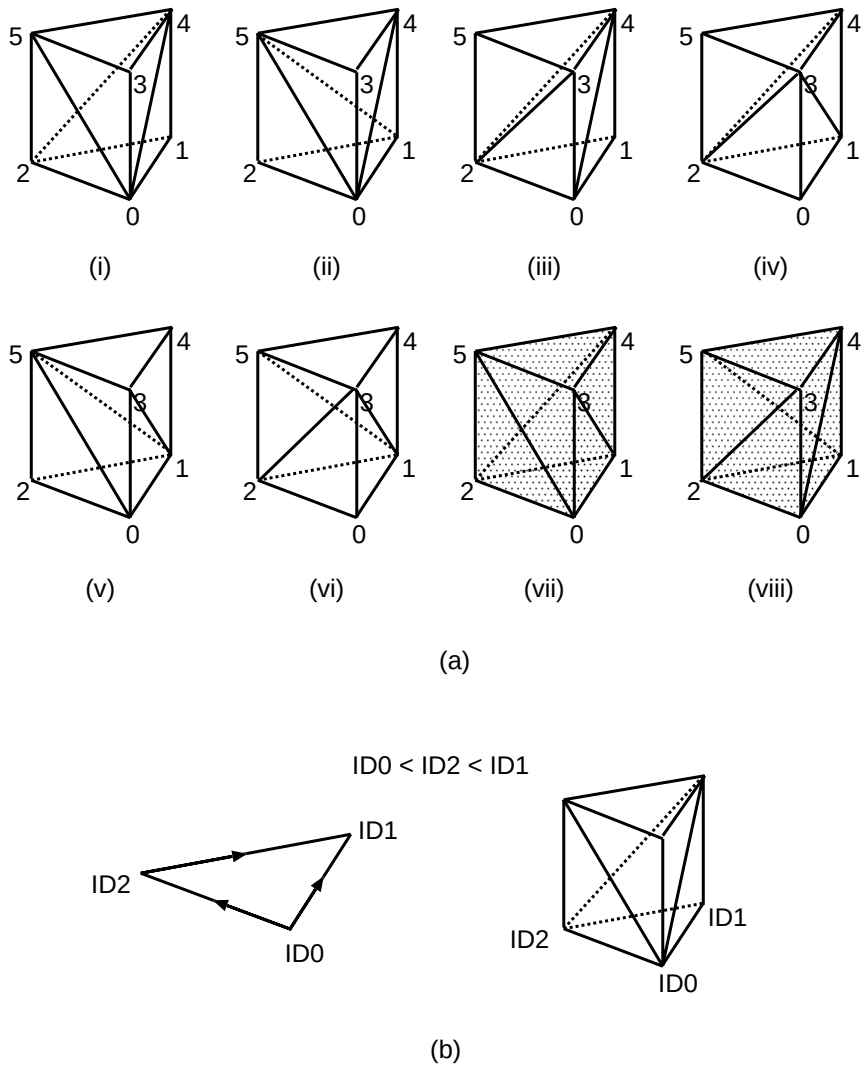


Figure 7.7: (a) (i)-(vi) Valid prism triangulations. (vii)-(viii) Invalid prism configurations. (b) Choice of prism diagonals based on Base Vertex IDs to assure validity of prism configurations.

7.8 Creation of Transition Tetrahedra

Atop a prism with one level difference between its component growth curves, there may be one or two transition tetrahedra depending on whether one or two growth curves have fewer nodes than the others. Similar to the transition triangles for the boundary layer quads, the diagonal faces of the transition tetrahedra may run counter to the pattern that is used for the diagonal faces of the underlying prisms. If the level difference between the growth curves is more than one, then

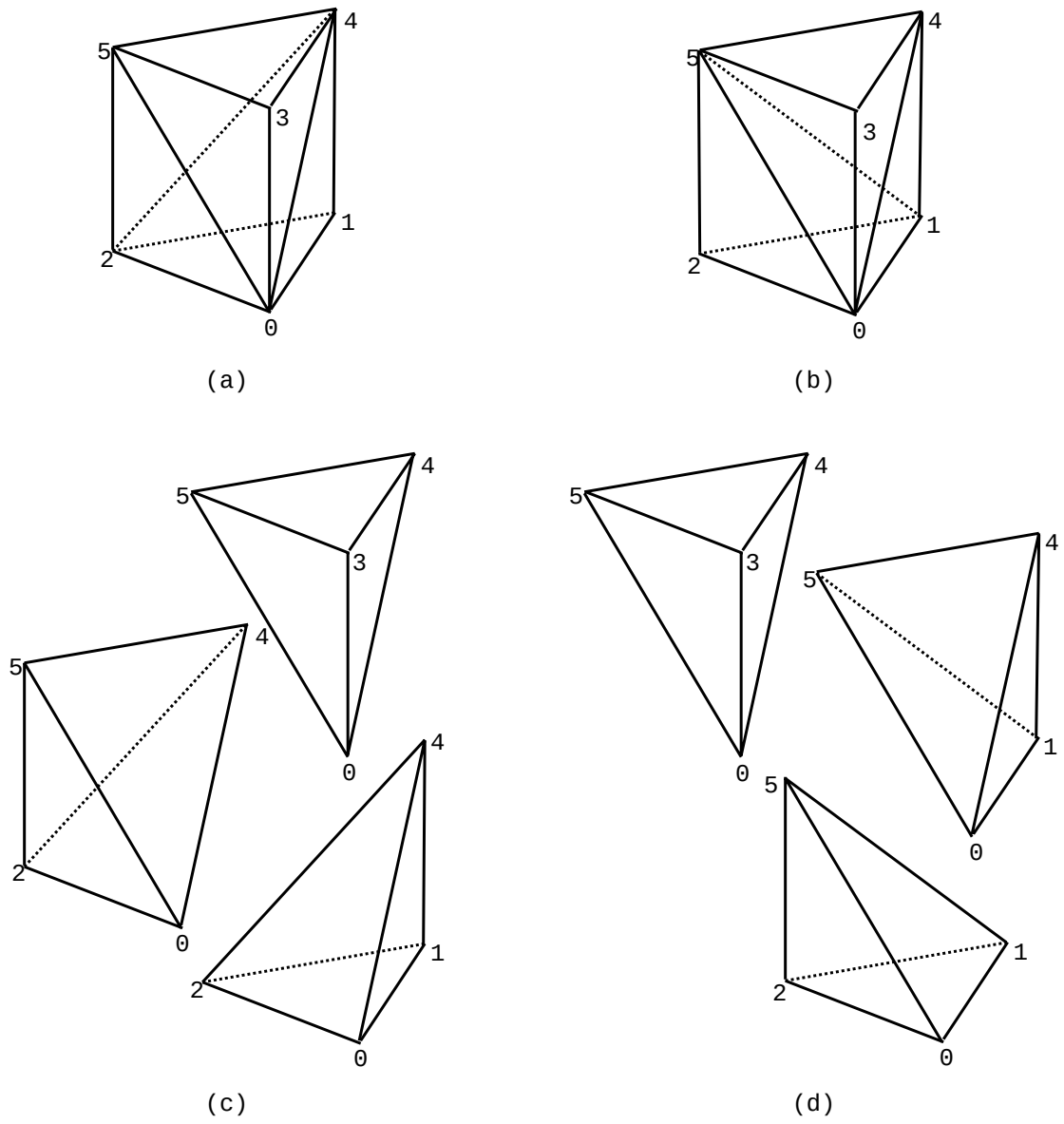


Figure 7.8: Boundary Layer Prism Templates.

several layers of transition elements are created on top of the prism. Consider the case when one of the growth curves has one more node than the other two which have n nodes. Then the topmost nodes of the two shorter growth curves (nodes n), the topmost node (node $n+1$) of the tall growth curve and its previous node (node n) form one transition tetrahedron (*Type I transitions*, Figure 7.9a). On the other hand, if two growth curves have one more node ($n+1$) than the third growth

curve, then two transition tetrahedra are formed on top of the base prism (*Type II transitions*, Figure 7.9b). These two transition tetrahedra together use the nth node of the short growth curve, and the nth and $(n+1)$th nodes of the taller growth curves. The diagonal connection between the two tall growth curves is chosen to be in the same direction as the rest of the quad below it, i. e., it is based on the IDs of the base nodes. Multiple transition layers may be completely *Type I* (Figure 7.9c), completely *Type II* (Figure 7.9d) or may start off as *Type II* transitions and then switch to *Type I* transitions (Figure 7.9e). It is not possible to switch from *Type I* transition to *Type II*.

It is worthwhile to mention here that the checks for the validity of elements to be formed in the boundary layer mesh that are performed in the smoothing, shrinking and pruning operations take into account the presence of transition triangles and tetrahedra.

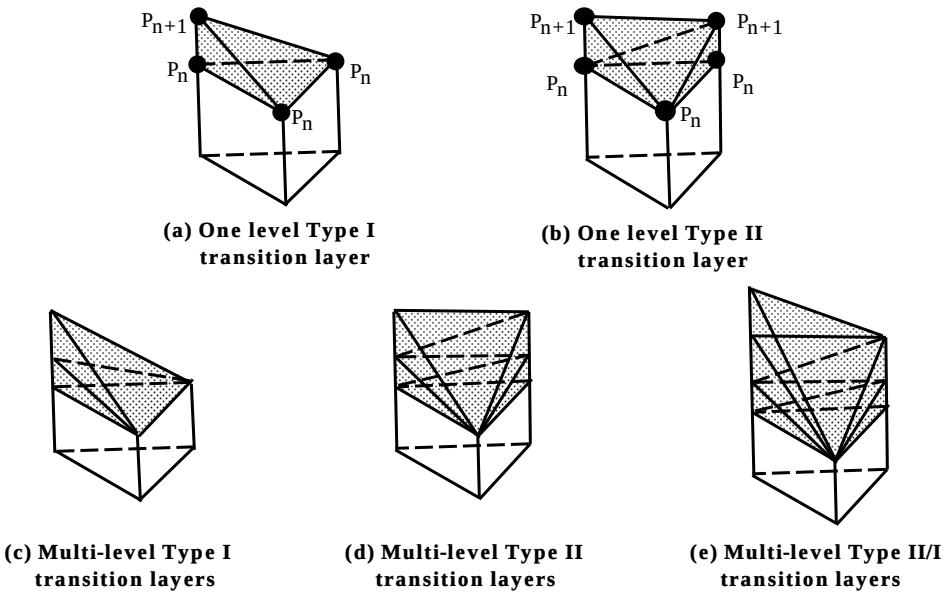


Figure 7.9: Transition Elements.

7.9 Creation of Boundary Layer Blend Polyhedra

As mentioned before, the introduction of multiple growth curves at mesh vertices due to surface mesh geometry introduces gaps between prisms. These gaps

are made up of highly stretched faces present on the sides of prisms. If left as they are, the highly anisotropic faces of the gaps cause problems for the isotropic volume mesher. Therefore, the concept of blend meshes is introduced in the generalized advancing layers procedure. The blend meshes are designed to fill the gaps between prisms while maintaining a good mesh gradation in the boundary layer mesh.

Consider a situation in two dimensions where there are two growth curves at a model vertex representing a convex corner (Figure 7.10a,b). The gap in the two dimensional boundary layer mesh formed at this vertex must be filled by blend triangles (Figure 7.10c). If the angle between the two growth curves is too large, bridging the gap with a single set of blend elements results in disparate mesh sizes on the outer surface of the boundary layer mesh exposed to the isotropic mesh. Therefore, to maintain a good mesh gradation on this outer surface, additional growth curves may have to be introduced at a mesh vertex in between the growth curves used by the standard boundary layer elements (quads in 2D and prisms in 3D) as shown in Figure 7.10d.

The number of additional growth curves required at a mesh vertex can be calculated quite simply by using the model shown in Figure 7.11. Consider a mesh vertex M_i^0 at which there are two original growth curves separated by an angle θ. Let the average mesh size at the vertex be h_{ave}. Assuming that all the growth curves forming quads are nearly perpendicular to their respective base edges, the average mesh size on the outer boundary of the boundary layer mesh is also taken as h_{ave}. Therefore, n additional growth curves must be introduced between the two original growth curves such that the outer entities of the blend mesh also have a mesh size of h_{ave}. It is assumed that the n growth curves will be equispaced in the gap. Also, the average boundary layer thickness in the neighborhood is taken to be H. The arc length between the outermost nodes of the two original growth curves is $L = H\theta$

Therefore the arc length between the outermost nodes of any two consecutive growth curves in the blend is:

$$l = \frac{L}{n} = \frac{H\theta}{n} \tag{7.1}$$

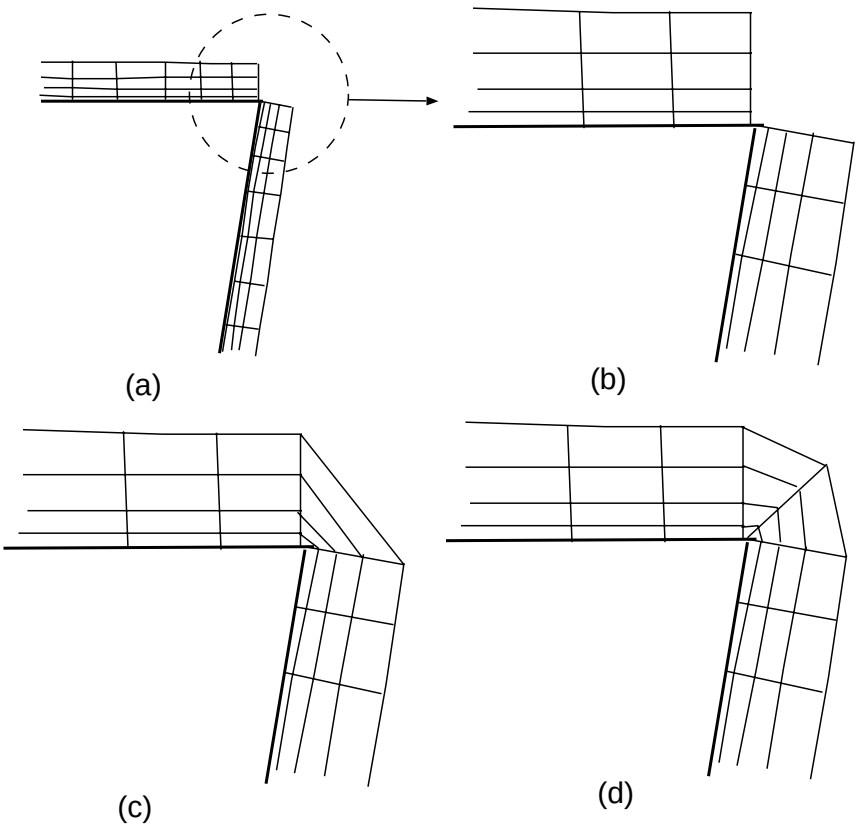


Figure 7.10: Two dimensional illustration of the need for blends and multiple growth curves within blends. (a) Boundary layer mesh on two surfaces with convex corner. (b) Gap between the corners shown in greater detail. (c) Blend mesh directly bridging the two existing growth curves. (d) Blend mesh with introduction of additional growth curve.

Since the aim is to make $l \approx h_{ave}$, it can be deduced that

$$\begin{aligned} h_{ave} &= \frac{H\theta}{n}, \quad \text{or} \\ n &= \frac{H\theta}{h_{ave}} \end{aligned} \tag{7.2}$$

rounded off to the nearest integer.

In three dimensions, gaps may occur at mesh edges or mesh vertices. For simplicity of discussion, it is assumed for now that the need for blend meshes occurs only at mesh edges (and vertices) classified on model edges and at mesh vertices

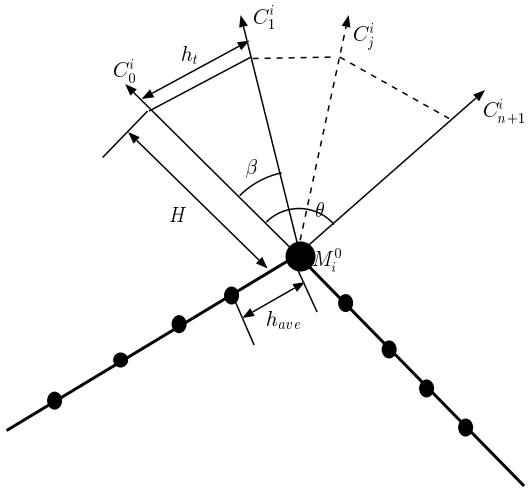


Figure 7.11: Calculating number of additional growth curves for blend meshes.

classified on model vertices. At model edges, the dihedral angle between two model faces connected to the model edge may be constant or vary continuously. If the dihedral angle is constant, then the same number of growth curves are created at each mesh vertex classified on the model edge. A blend mesh on such a model edge will be termed a *fixed or constant blend* since its structure does not vary along the model edge (Recall Figure 7.1b earlier in the chapter). If the dihedral angle does vary along the model edge then it is clear that some mesh vertices may have more growth curves than the others. In this case the topology of the blend mesh must also change along the model edge. Such blend mesh are referred to as *variable blends*. In the discussion below, a further simplification is introduced by assuming that no additional growth curves are introduced in a blend and that the gap is directly bridged by connecting the original growth directions used to create prisms in the mesh. Such blends are referred to as *simple blends*. As noted above, this may result in poor mesh gradation. Therefore, this is compensated for by reducing the angle between multiple growth curves by 25%.

Simple blends require only two sets of templates to mesh. The first case is the *simple fixed blend* in which the vertices of a mesh edge have two growth curves each and the mesh edge has two quads which are joined to form a blend polyhedron.

A simple fixed blend between two quads at a mesh edge M_i^1 with vertices M_j^0 and M_k^0 is shown in Figure 7.12a. As seen in the figure, the first layer of simple fixed blend consists of a prism while subsequent layers are hexahedra stacked on top of each other (in the figure, only one hexahedron is shown since the mesh has only two layers). A triangulation of the simple fixed blend is shown in Figure 7.12b and an individual view of the triangulated prism and hexahedron are shown in Figure 7.12c,d. The salient points of the triangulation are:

1. The sides of prisms and hexahedra formed by the quads *always* have matching diagonals since the quads contributing to them arise from the same mesh edge.

2. It is impossible to obtain an invalid triangulation of the bottom prism since the two matching diagonals from the quads will always meet at a vertex. The third diagonal (forming the base of the hexahedron can be chosen arbitrarily and is selected to provide the best element shapes.

3. Since the shape of the tetrahedra resulting from a regular hexahedron is optimal when all its opposite diagonals are matching, the free diagonals on the blend hexahedron are chosen to match each other if possible. The diagonals that are fixed are the ones from the quads (which have already been shown to be matching) and the bottom face of the hexahedron (which is inherited from the bottom prism). The diagonal of the top face of the hexahedron is unconstrained and is chosen to match the bottom diagonal unless this causes a geometrically invalid triangulation. The diagonals on the side faces of the hexahedron are also unconstrained unless one or both of their faces lie on the model boundary and their diagonals have already been fixed. Even if both the side diagonals are fixed and are not matching, the hexahedron can still be triangulated if the top and bottom diagonals match. No tetrahedronization is possible without introducing an interior vertex if two sets of diagonals do not match for a hexahedron. Naturally, if two blend polyhedra share a common interface originating from a vertex their diagonals are expected to be conforming.

The two prism templates used before can be used as is for meshing the bottom prism. Accounting for the possibility of a mismatch between the top and bottom diagonals, and side diagonals, the blend hexahedron can be triangulated by three distinct templates. The number of different possible templates is reduced to only three by assuming that the hexahedron will always be considered with a quad diagonal and the "bottom" face diagonal meeting in the front lower left corner. Note that to satisfy this condition and apply one of the templates, the hexahedron may have to viewed upside down.

The second case is when one vertex of a mesh edge has one growth curve and the other has two (the edge still has two quads joined at one end and separated at the other). This is called the *simple variable blend* and is shown Figure 7.13a. The simple variable blend is comprised of a 5-vertex wedge and a number of 6-vertex wedges stacked on top of each other. The 5-vertex and 6-vertex wedges are shown in Figure 7.13b(i) and Figure 7.13b(ii) respectively. As with the simple fixed blend, the diagonals of the side faces of the blend polyhedra are constrained to match each other. The 5-vertex wedge may yield two or one tetrahedra depending on the direction of the diagonal of the quads as shown in Figure 7.13c(i) and Figure 7.13c(ii) respectively. In the latter case the triangulation of each quad from the base mesh edge results in an edge between vertices M_j^0 and $M_{k,0,1}^0$. Therefore, there are two coincident edges between the two vertices M_j^0 and $M_{k,0,1}^0$, and two coincident faces between the vertices M_j^0, M_k^0 and $M_{k,0,1}^0$, one from each quad. These coincident entities are merged to prevent the mesh from becoming topologically invalid. This leaves only one tetrahedron $\{M_j^0, M_{j,1,1}^0, M_{j,0,1}^0, M_{k,0,1}^0\}$ to be created from the 5-vertex edge. The two tetrahedra that result from the other 5-vertex wedge are easy to see. The 6-vertex wedge can be triangulated by two templates accounting for the various symmetries in the polyhedron.

Using the above concepts for simple blends, the gap between prisms at edges can be closed off from the isotropic mesh generator. These templates may also be used to build up general fixed and variable blend meshes.

As mentioned in earlier discussions, gaps between the various boundary layer constructs at model vertices are much more general since an arbitrary number of

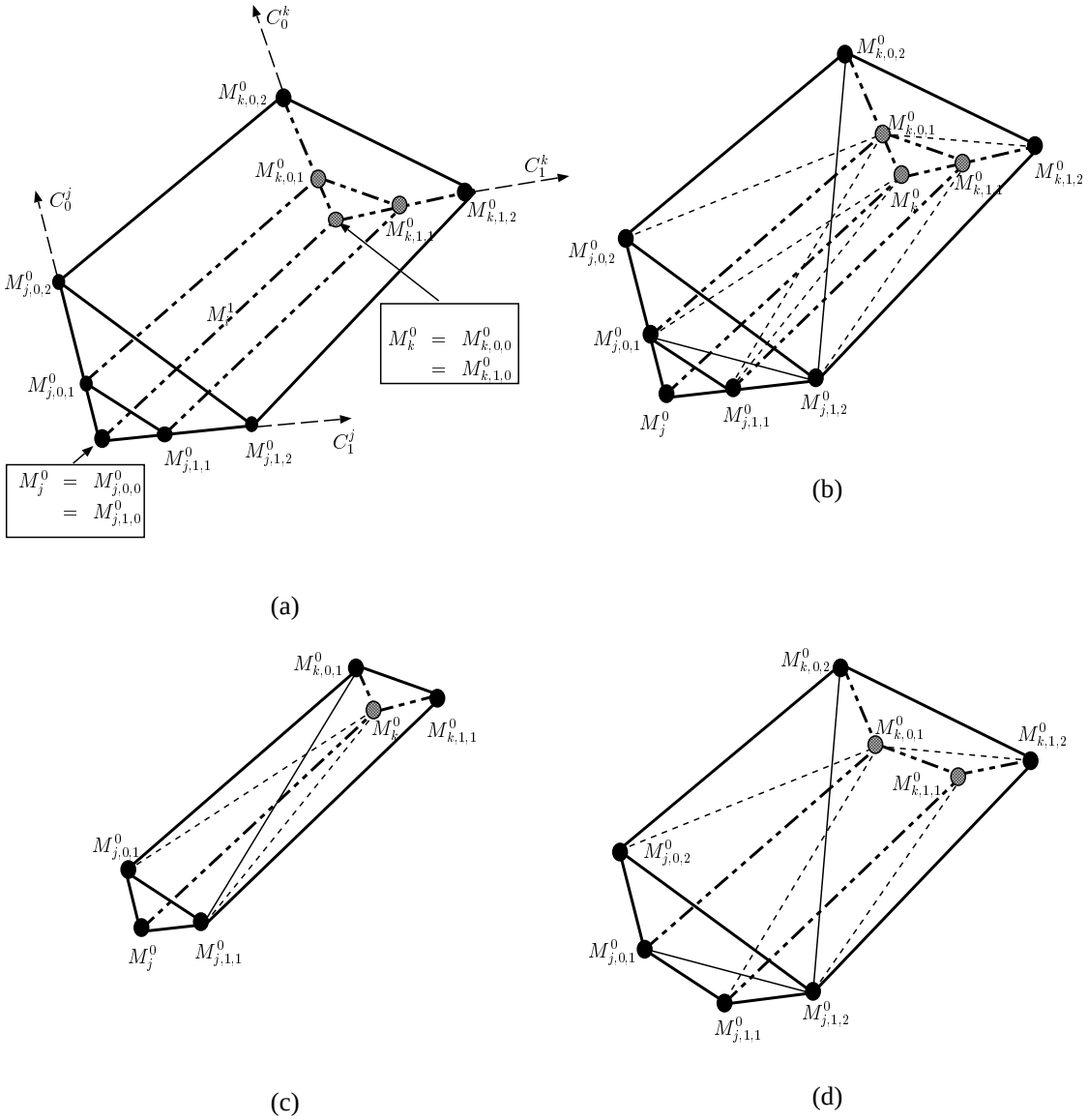


Figure 7.12: Simple fixed blend.

prisms and blends may can contribute to them. However, their anisotropy is also much lesser than that of edge blends making them easier to handle using less specialized procedures. It is proposed that the vertex blends be created using general mesh generation techniques tailored for the purpose of filling these gaps.

The above discussion has focused on filling gaps between boundary layers at model edges and model vertices. In fact, multiple growth curves and therefore,

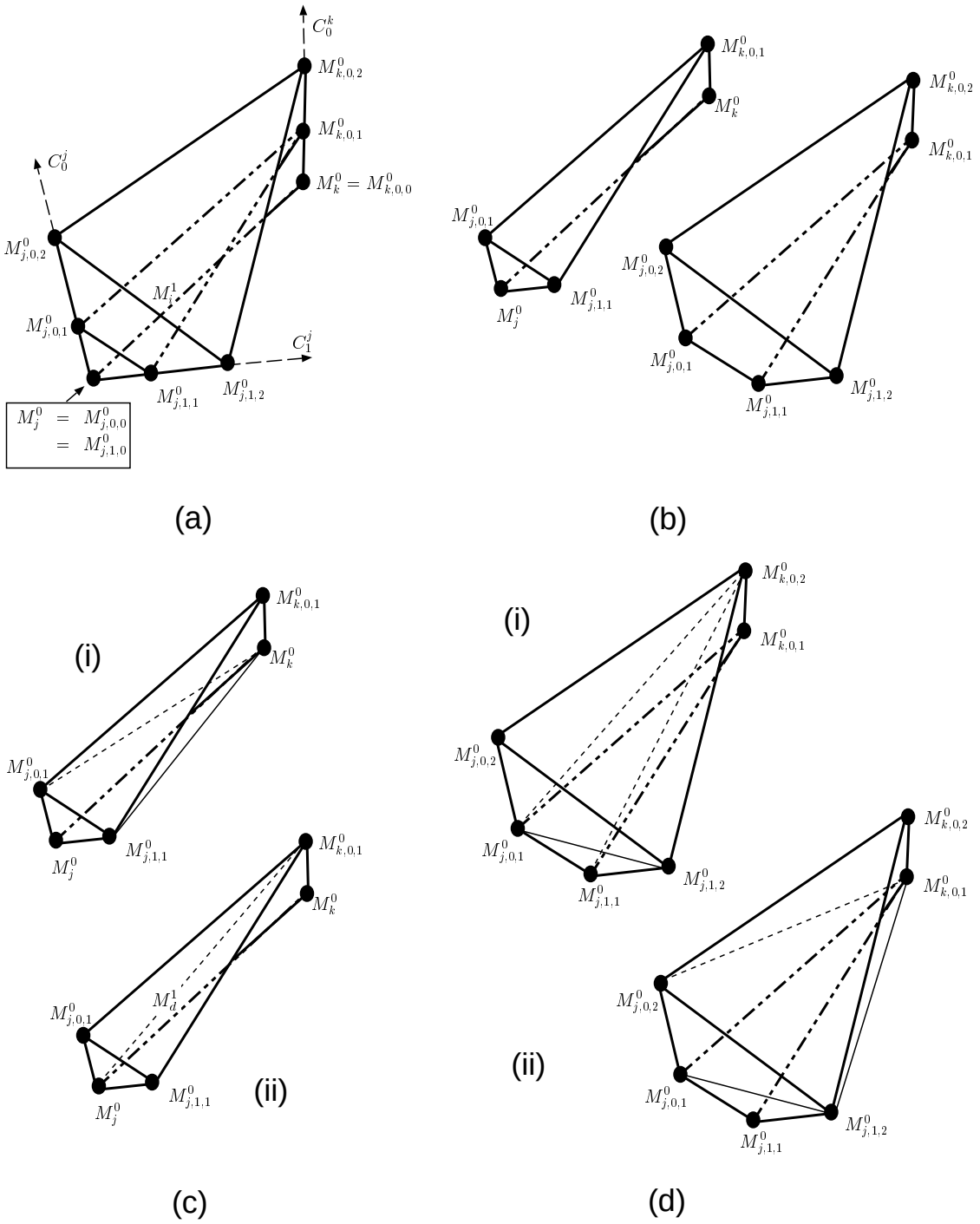


Figure 7.13: Simple variable blend.

gaps between prisms may occur at any mesh edge or mesh vertex classified on the boundary. Therefore, a complex blend structure may be constructed on a model face with sharp corners as shown in Figure 7.14. The procedures and templates to build blend meshes simply proceed on a mesh entity by entity basis and can be used throughout the mesh. It must be noted here, that blend meshes are not present in the current implementation but are expected to be incorporated into the procedures as described above.

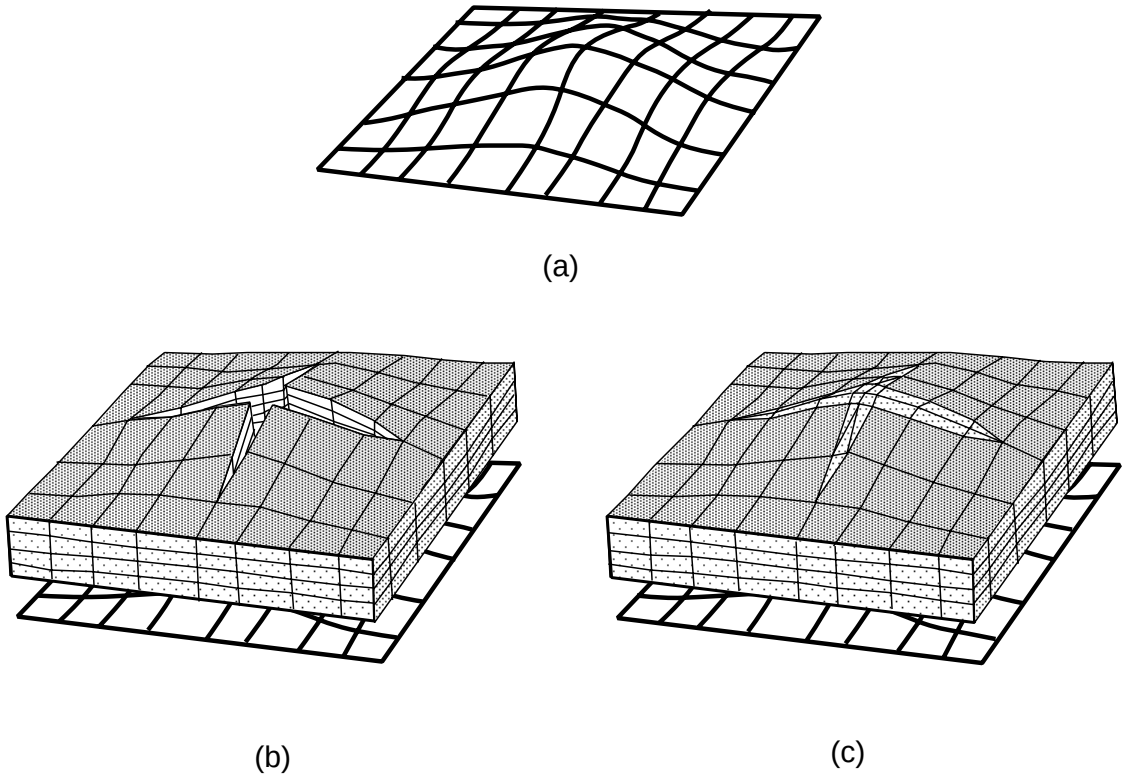


Figure 7.14: Blend meshes on model faces. (a) Model face discretization. (b) "Prismatic" boundary layer mesh on model face with gaps between "prisms". (c) Boundary layer mesh with gaps filled in by edge and vertex blends.

The combination of prism, transition and blend constructs presented in this chapter may be used effectively to construct a good anisotropic mesh for capturing boundary layers while shielding the isotropic mesh generator from most stretched faces. However, there may still be some situations where the boundary layer mesh ends abruptly at a sharp corner. To absolutely prevent the isotropic volume mesher

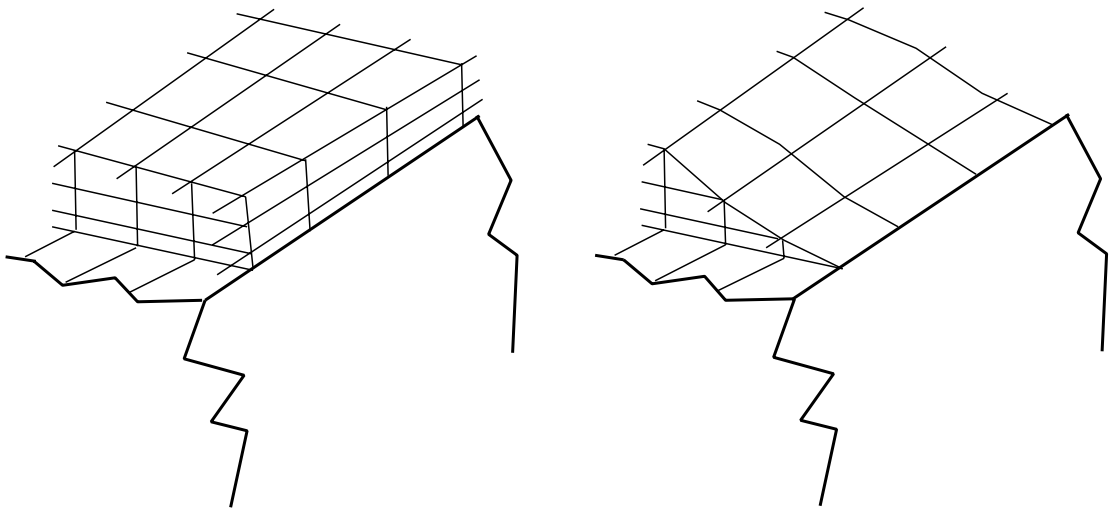


Figure 7.15: Transitioning of boundary layers at model edge. (a) Boundary layer without elimination of exposed faces. (b) Boundary layer with elimination of exposed faces by transitioning.

from seeing the stretched faces of the boundary layer mesh, the number of nodes along the sharp corner edge are reduced to zero and the boundary layer transitioned out from the edge (as described in [11]). This is shown in Figure 7.15.

CHAPTER 8
BOUNDARY LAYER MESHING - FIXING BOUNDARY LAYER INTERSECTIONS

When boundary layer elements are generated on model faces that are too close to each other the layers may run into each other. When boundary layers run into each other the polyhedral cavity that remains to be meshed is self-intersecting. Since the isotropic mesher expects a polyhedral cavity with no self-intersections, this situation must rectified. Boundary layer intersection is fixed by local shrinking of layers and if that fails, then by local pruning of growth curves leading to deletion of elements. However unlike the procedures to ensure validity of elements, correction of self intersections is done after element creation. This is because it is simpler to find the entities of the boundary layer mesh that make up the polyhedral cavity when actual boundary layer entities exist in the mesh and because the checks for self intersections can be more localized.

The technique used to detect self intersections in the mesh draws upon concepts from advancing front methods for mesh generation. After creation of the boundary layer elements, mesh faces that have fewer regions connected to them than is necessary for filling the domain are considered to be on a front. Front faces may be faces classified on model faces with no boundary layer, and top and side faces of the boundary layer prisms, blends and transition polyhedra. Internal faces with no mesh region attached to them are not permitted. The front faces are referred to as *exposed* faces since they are exposed to the volume mesher. Exposed faces are checked for intersection with other exposed faces in the neighborhood. If an intersection is found, its connected prisms are shrunk to fix the intersection. Since the algorithm only checks and fixes the intersection between exposed faces and not of entire prisms, it is possible that shrinking of two prisms to correct interference between their exposed faces results in new intersections of other faces in the neighborhood. Therefore, the algorithm is iterative and continues until all intersections are fixed. The intersections that cannot be fixed by shrinking are dealt with by the pruning of

growth curves. *Given that the initial surface mesh was not self-intersecting and that the face retriangulation did not cause any new self-intersections, pruning of growth curves must eventually fix all self intersections that could not be fixed by shrinking.* The process of fixing boundary layer run-in is illustrated in Figure 8.1 using a 2D example.

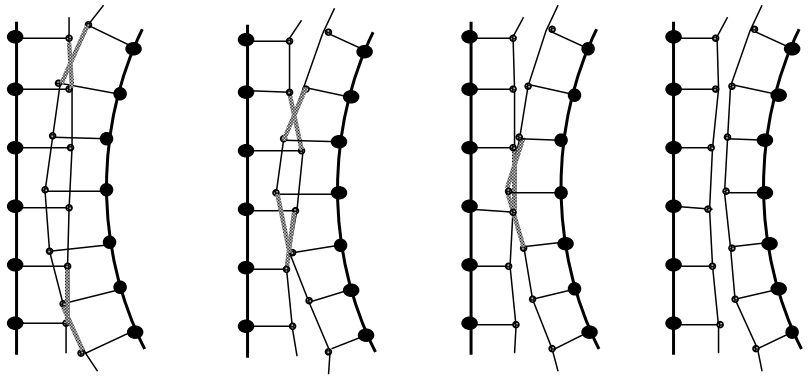


Figure 8.1: Iterative procedure for fixing boundary layer intersections - 2D example.

8.1 Localization of Search for Intersections Using an Octree

Since the front is typically quite large, the search for intersections between exposed faces is made more efficient by using an octree. An octree is built in the domain to serve as a localization structure for intersection checks. The tree is built using a rough estimate of the average size of the exposed faces as a criterion for subdivision of the tree. Specifically, at each mesh vertex the mesh size is approximated by the root mean square length of the edges of exposed faces. This size is used to define the level of refinement of an octant containing this vertex. This approximation is made to make the subdivision process less precise and more efficient since the tree is only used as a search structure. Exposed faces lying partly or fully in a terminal octant are attached in a list to the octant to avoid repetitive search for frequently required information.

Intersection checks in the Generalized Advancing Layers algorithm are performed between exposed faces of the boundary layer constructs on surface mesh face uses. Therefore, the definition of neighborhood for intersection checks is based

on face uses of the surface mesh on which the boundary layer elements have been built. Given a surface mesh face use, its growth curves are first found. The bounding box of the three growth curves together is found and expanded by 10 percent. The set of octants intersecting this bounding box is quickly found using coordinate checks. The exposed faces lying in these octants are found and their base faces in the surface mesh are determined through appropriate links. Knowing the other surface mesh faces that lie in the neighborhood of a particular mesh face then sets the stage for fixing any intersection between the respective elements by shrinking or pruning of the component growth curves. This idea is illustrated in 2D in Figure 8.2.

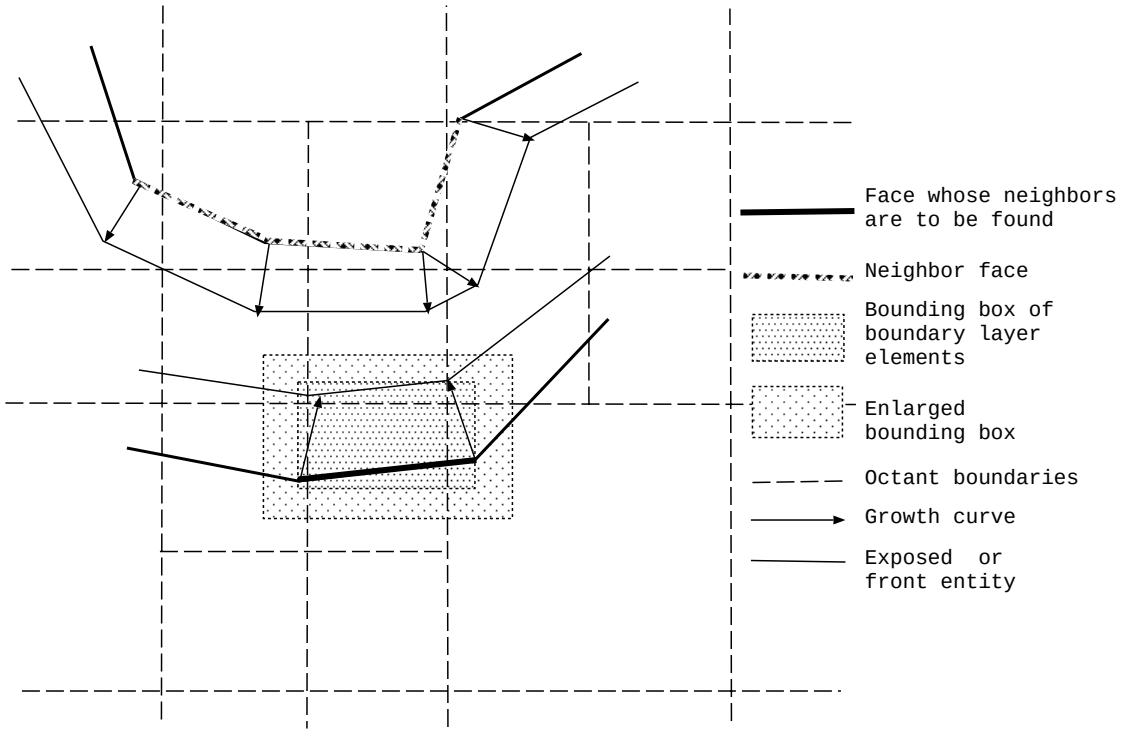


Figure 8.2: Finding neighborhood faces for intersection checks.

8.2 Intersection Checks

Given two surface mesh face uses, exposed faces of the boundary layer constructs on the first face are checked against exposed edges of the boundary layer constructs on the second face. Although it is possible that the reverse may be true,

that is, an exposed edge of the second face use's boundary layer entities may intersect an exposed face of the first, this is not explicitly checked since the intersection check will be invoked later (or has already been invoked) with the order of the face uses reversed. In addition to the normal intersection check, an additional check is performed for a rare but possible situation. Consider a mesh face use whose vertices are all classified on the closure of the model face the mesh face is classified on. If all the growth curves from the vertices of this face classified on a single model face, then a mesh face may exist connecting the top nodes of the three growth curves and classified on the same model face. However, the prism template creates a mesh face classified on the model region between these three nodes. This creates two coincident mesh faces which are detected and merged.

8.3 Fixing Intersections by Shrinking and Pruning Growth Curves

The procedure to detect and fix intersections proceeds on a model face use by model face use basis excluding those model face uses that do not carry a boundary layer. For a given model face, the list of mesh faces classified on the model face are obtained. For each mesh face in this set, its set of neighboring mesh face uses is obtained and intersection checks performed between the exposed faces of the boundary layer constructs of the mesh face use and each mesh face use in the neighborhood as explained above. If an intersection is found it is recognized that the exposed faces of the boundary layer elements from the face use may intersect the exposed faces of more than just the one neighboring mesh face use. Therefore, at this stage the intersections of the exposed faces from the primary mesh face use are checked with the exposed face uses of each of the neighboring mesh face uses and attempted to be fixed by shrinking the growth curves. At the end of this step any intersections between exposed faces from the mesh face use and any mesh face uses in the neighborhood are resolved, if this is possible to achieve through shrinking of growth curves. Shrinking of the growth curves is constrained to keep elements valid. Also, when growth curves are shrunk, vertices associated with their nodes move and as a result, exposed faces may move out of some octants and partially or fully enter

others. This is accounted for by finding the octants containing the face in its old and new positions, and updating both sets.

The algorithm is made more efficient by recognizing that if an intersection between two faces has been detected and fixed there is a good possibility that the fix has caused new intersections or that there are already more intersections in the neighborhood. Therefore, when an intersection is found in the global iteration described above, a locally recursive intersection detection and correction is applied to try and fix the problems in the entire neighborhood. When an intersection is detected and fixed while iterating through the global list of front faces, its adjacent exposed faces are pushed into a local queue. Each member of the local queue is then popped, checked and fixed if an intersection is present. Only if an intersection is detected for a face in the local list are its neighbors also pushed into the queue. The local intersection correction procedure ends when the queue is empty. If a face has been checked in a local step and then it is not checked in that global iteration through the front faces.

Again, due to the fact that only front face intersections are checked, the local correction procedure is not guaranteed to detect all prism interferences in the neighborhood. That is why multiple iterations of the global iteration through the front face list are carried out. The global iterations are performed until no more intersections remain or the procedures can fix no more iterations by shrinking prisms. In practice, it has been observed that one global iteration fixes all the intersections that can be fixed by shrinking and a second one is necessary to verify that.

If all the intersections between front faces cannot be fixed by shrinking prisms, then the growth curves contributing to the intersections are pruned using a similar algorithm as described above. Pruning of growth curves introduce some additional complexities in the algorithm to fix intersection. Firstly, pruning of a growth curve introduces differences between the number of nodes in growth curves of surface mesh faces connected to the base vertex. To avoid exposing the anisotropic faces on the sides of prisms, transition elements must be created to bridge the difference in the number of nodes. Also, deletion of elements introduces new exposed faces and the octants containing the old and new exposed faces must be updated.

CHAPTER 9
BOUNDARY LAYER MESH GENERATION - RESULTS

9.1 Introduction

The boundary layer mesh generator described in the previous chapter has been tested against a wide range of complex models for generation of multi-million meshes for fluid flow simulations. The mesh generator has been used for various applications such as capturing thermal boundary layers for complex automobile configurations with complete under-the-hood and under-carriage detail, climate control simulations in automobile interiors, blood flow simulations, crystal growth, RANS simulation of turbulent flow and aeroacoustics. In this chapter some of the meshes and results of simulations are presented to demonstrate the capabilities of the mesh generator. Also, results of analysis on two problems, one involving laminar flow and the other involving turbulent flow with shear layers are presented to validate the method.

9.2 Example meshes for general models

Figure 9.1 shows the boundary layer mesh for the ONERA-M6 wing model. Figure 9.1a shows the surface mesh for the wing and the Figure 9.1b shows the boundary layer mesh for the wing. Figure 9.1c(i),(ii) show two close-ups of the surface mesh, Figure 9.1d(i),(ii) show close-ups of the boundary layer mesh on the wing only and Figure 9.1e(i),(ii) show close-ups of the boundary layer mesh on the wing and wing tip. This mesh was constructed for illustrative purposes only and does not reflect the mesh sizes and gradations required for performing a reliable simulation on the wing.

The boundary layer mesh generator has been used to generate meshes for simulation of climate control systems in the interior of automobiles. Several cut-away views of the boundary layer mesh on all the interior surfaces of such a model[6] are shown in Figure 9.2. The boundary layer mesh in the example shown here has deliberately been made very thick and also the behavior of the mesh generator (while

[6]Courtesy: Simmetrix Inc.

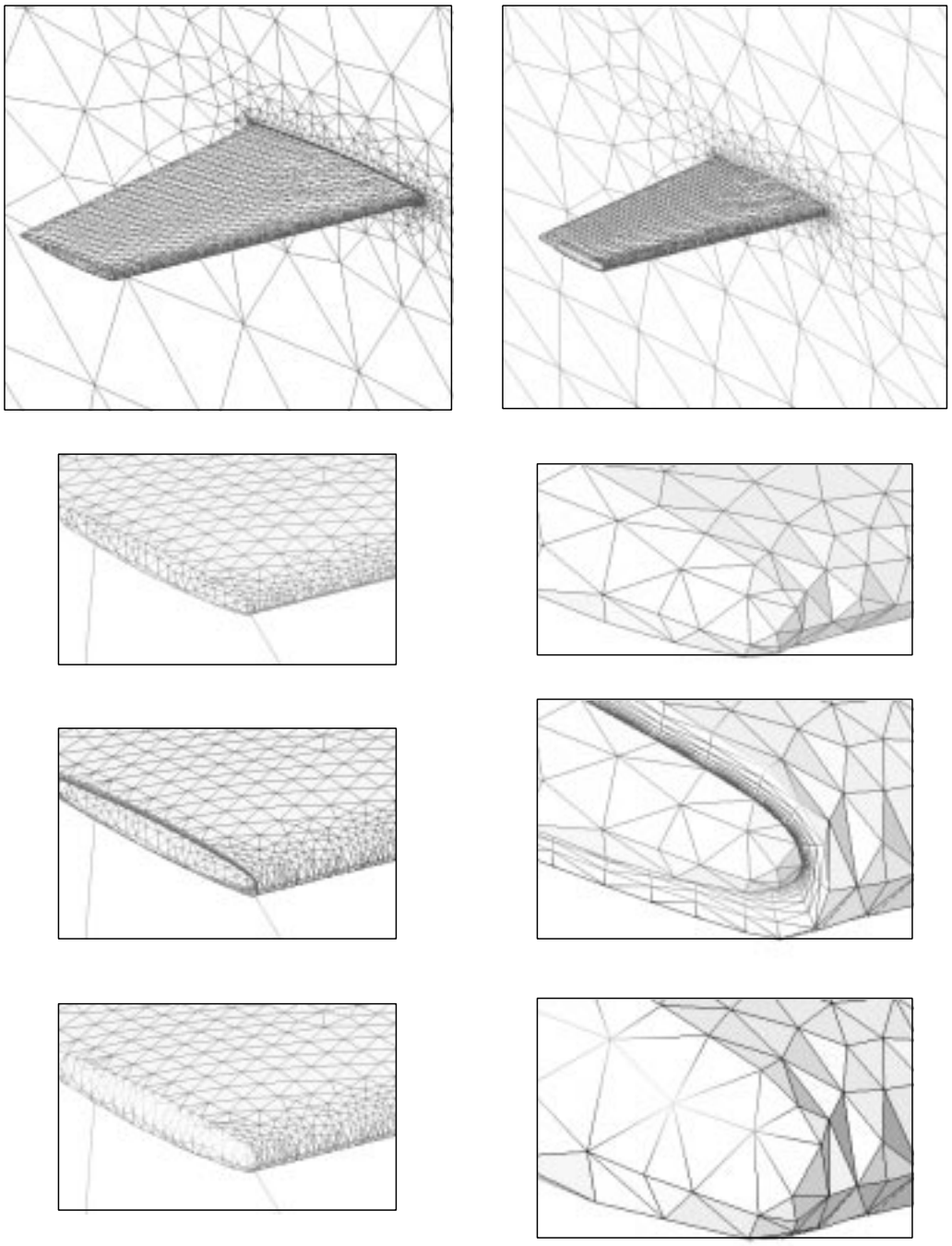


Figure 9.1: Boundary layer mesh for ONERA-M6 wing. (a) Surface mesh. (b) Boundary layer mesh. (c)(i) Close-up 1 of surface mesh near leading edge wing tip intersection. (ii) Close-up 2 of wing tip and leading edge. (d)(i) Close-up 1 of boundary layer mesh on wing. (ii) Close-up 2 of boundary layer mesh on wing. (e)(i) Close-up 1 of boundary layer mesh on wing and wing tip. (ii) Close-up 2 of boundary layer mesh on wing and wing-tip.

shrinking layers) has exaggerated to reveal the features of the mesh. In Figure 9.2a, the boundary layer mesh is shown on the entire car while the rest of the insets show zoomed in views of the boundary layer mesh. Figure 9.2b shows the mesh near the windshield, Figure 9.2c shows the mesh on a seat, two registers and the floor and finally Figure 9.2d shows the boundary layer mesh in the gap between the rear seat bottom and back rest. In all three close-up views, the shrinking of the boundary layers to avoid self-intersection is clearly visible in the example along with the gradation introduced by the recursive adjustment of growth curve heights. The boundary layer in this mesh has half a million tetrahedra and the complete mesh has approximately a million tetrahedra.

The generalized advancing layers method described here has been used extensively to generate boundary layer meshes for thermal management simulations of complex automobile configurations with complete under-the-hood and under-carriage detail. The boundary layer mesh on the under-body of one such vehicle[7] is shown in Figure 9.3a with part of the boundary layer cut away to show the complexity of the surface. The model is a non-manifold model with 11 model regions and 1236 model faces of which 71 are embedded faces. In the mesh actually used for simulations, the first layer thickness was $1.0E-02$ and the total thickness of the boundary layer was $5.0E-01$ with 4 layers of elements in the boundary layer. The largest requested element size was 25.0. In the mesh shown here, the total thickness of the boundary layer was increased by 2 orders magnitude to 25.0 for clarity of visualization. The original surface mesh has 168,000 mesh faces, the boundary layer mesh has 1.7 million tetrahedra and the complete mesh has 3.1 million tetrahedra. The *aspect ratio* (i. e. the longest edge length to shortest height ratio) of elements in the first layer are approximately 2500 on the most coarsely refined surfaces of the automobile. Figure 9.3b shows a close-up of the surface mesh and boundary layer mesh under the hood and near the front wheels while Figure 9.3c shows the under-carriage at the rear. The largest meshes generated for these types of models have been of the order of 4.5 million elements. It can be seen from the figures, that the mesh generator has successfully created a boundary layer mesh for this complex

[7] Courtesy: Simmetrix Inc.

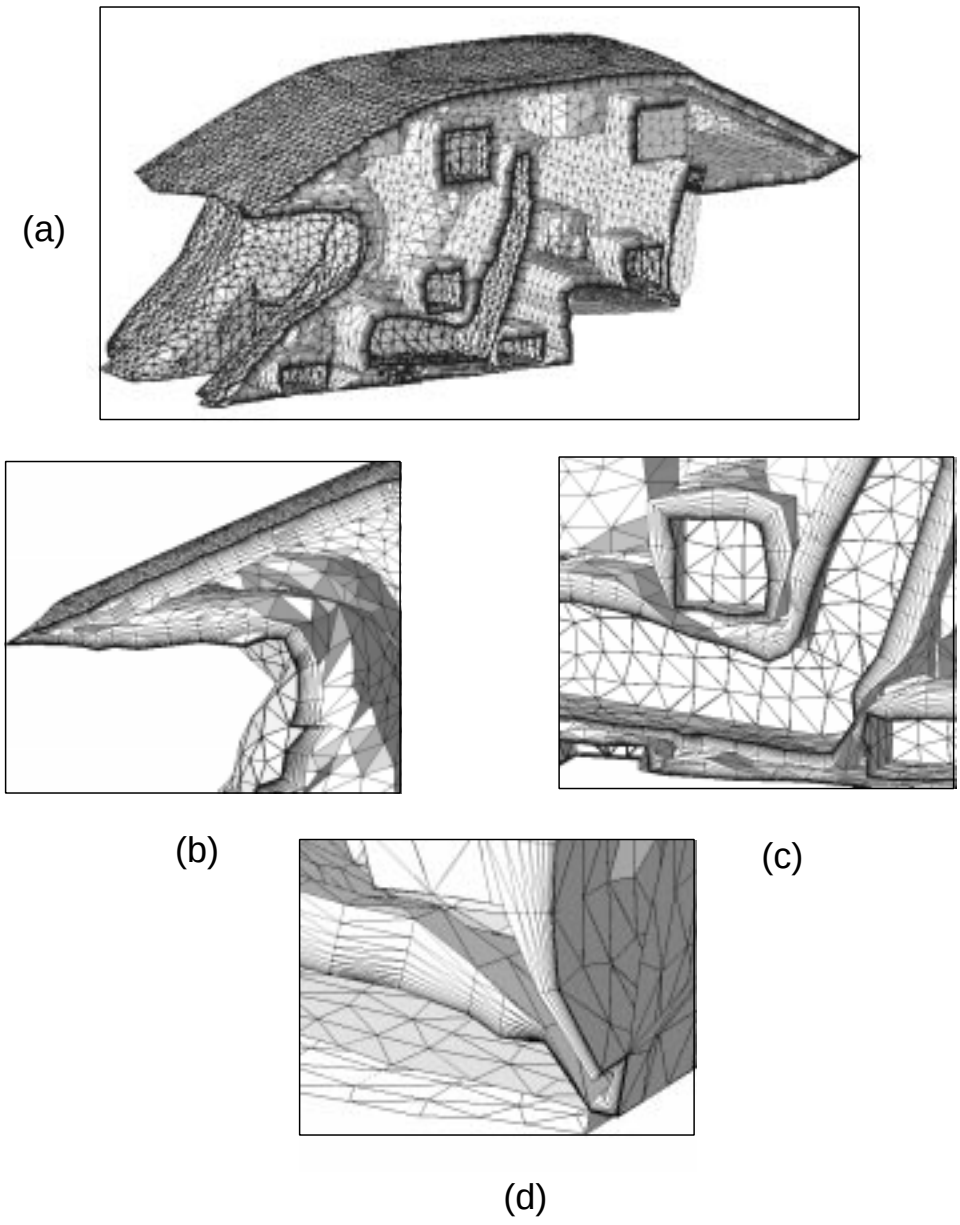


Figure 9.2: Boundary layer mesh for interior of car. (a) Cut-away of boundary layer mesh. (b) Close-up of mesh at juncture of windshield and dashboard. (c) Close-up of seat, floor and two registers. (d) Close-up of gap between rear seat bottom and back rest.

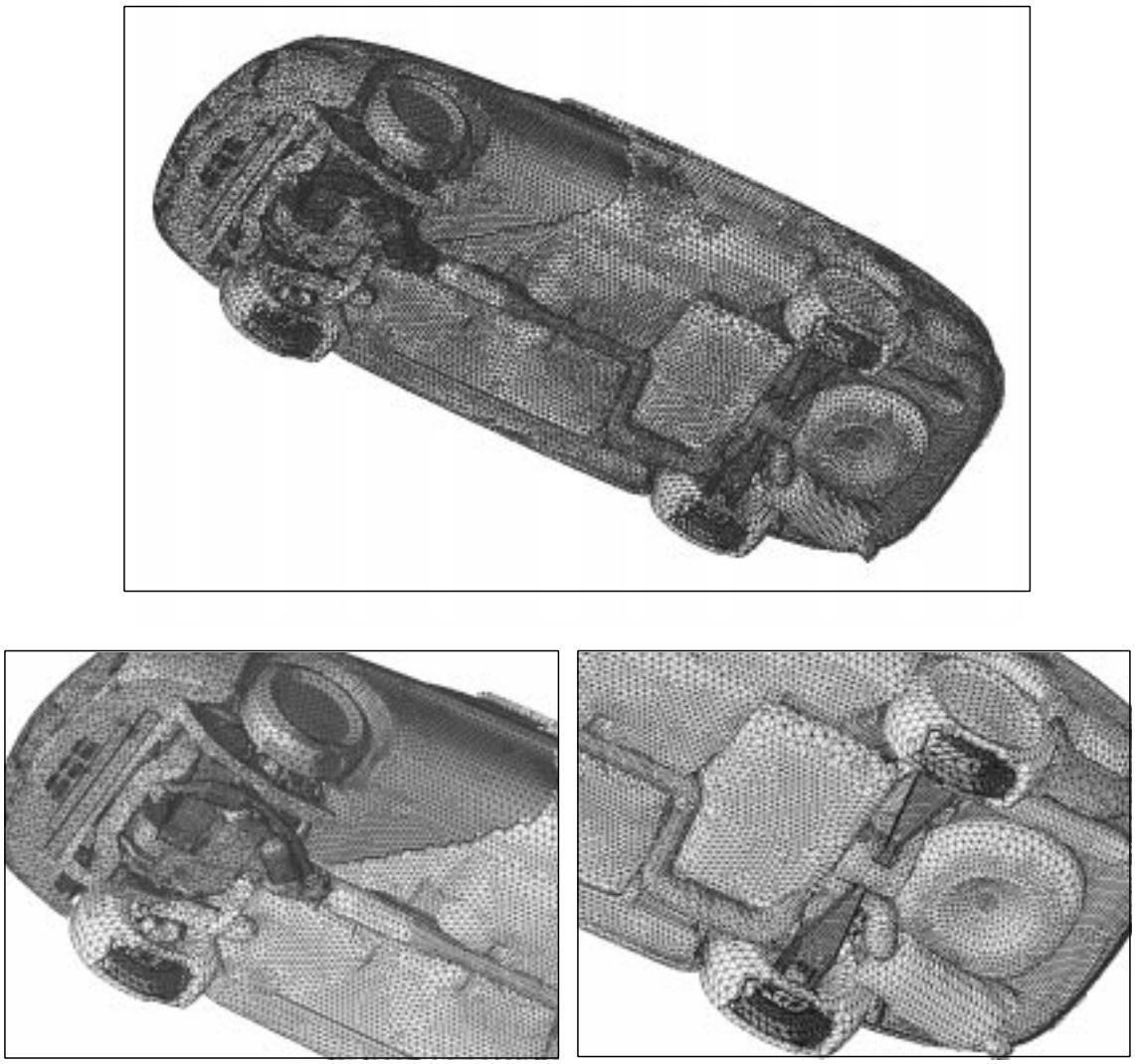


Figure 9.3: Boundary layer mesh for under-carriage of car. (a) Complete boundary layer mesh on all surfaces of car. (b) Cut away of boundary layer mesh revealing under-the-hood detail. (c) Zoom in of front end of under-carriage. (d) Zoom in of rear end of car.

geometry and has resolved self-intersections even in the most constrained portions of the domain.

The next example shows the use of the boundary layer mesh in simulations of flow in blood vessels for surgical planning [68, 69]. Figure 9.4a shows the model [8] of the arteries while Figure 9.4b shows a zoom in of the surface mesh which has a total

[8] Courtesy: Dr. Charles Taylor, Assistant Professor, Dept. of Surgery and Department of Mechanical Engineering, Stanford University.

44,000 triangles. Figure 9.4c,d display various cuts through the mesh showing the boundary layer and volume mesh inside the arteries. The boundary layer mesh has 5 layers with a first layer thickness of 0.002 units. The total boundary layer thickness is determined adaptively in the mesh based on the surface mesh size. Ideally, the boundary layer mesh thickness should be a function of the vessel diameter. Currently there is no mechanism to determine boundary layer mesh parameters on a pointwise basis *based on an arbitrary user-defined function*. Therefore, the available adaptive method is used under the assumption that the surface mesh size reflects changes in the diameter of the blood vessel. The number of boundary layer tetrahedra are 650,000 while the total number of tetrahedra are 800,000.

The example shown next is a model of the space shuttle with center tank and booster rockets. Only half the shuttle is modeled to take advantage of the symmetry. Figure 9.5a shows the geometric model (without the boundaries of the enclosing domain). Shown in Figure 9.5b,c and d are the retriangulated surface mesh on the symmetry plane, a cut-way of the boundary layer mesh and a close-up of the boundary layers showing the element anisotropy. The boundary layer mesh in this model has 810,000 elements while the complete mesh has 1 million elements. The first layer thickness is $1.0E-05$ while the total thickness of the boundary layer is $2.5E-02$ with a total of 10 layers. With the requested surface mesh sizes aspect ratio of the boundary layer elements is on the order of 20,000.

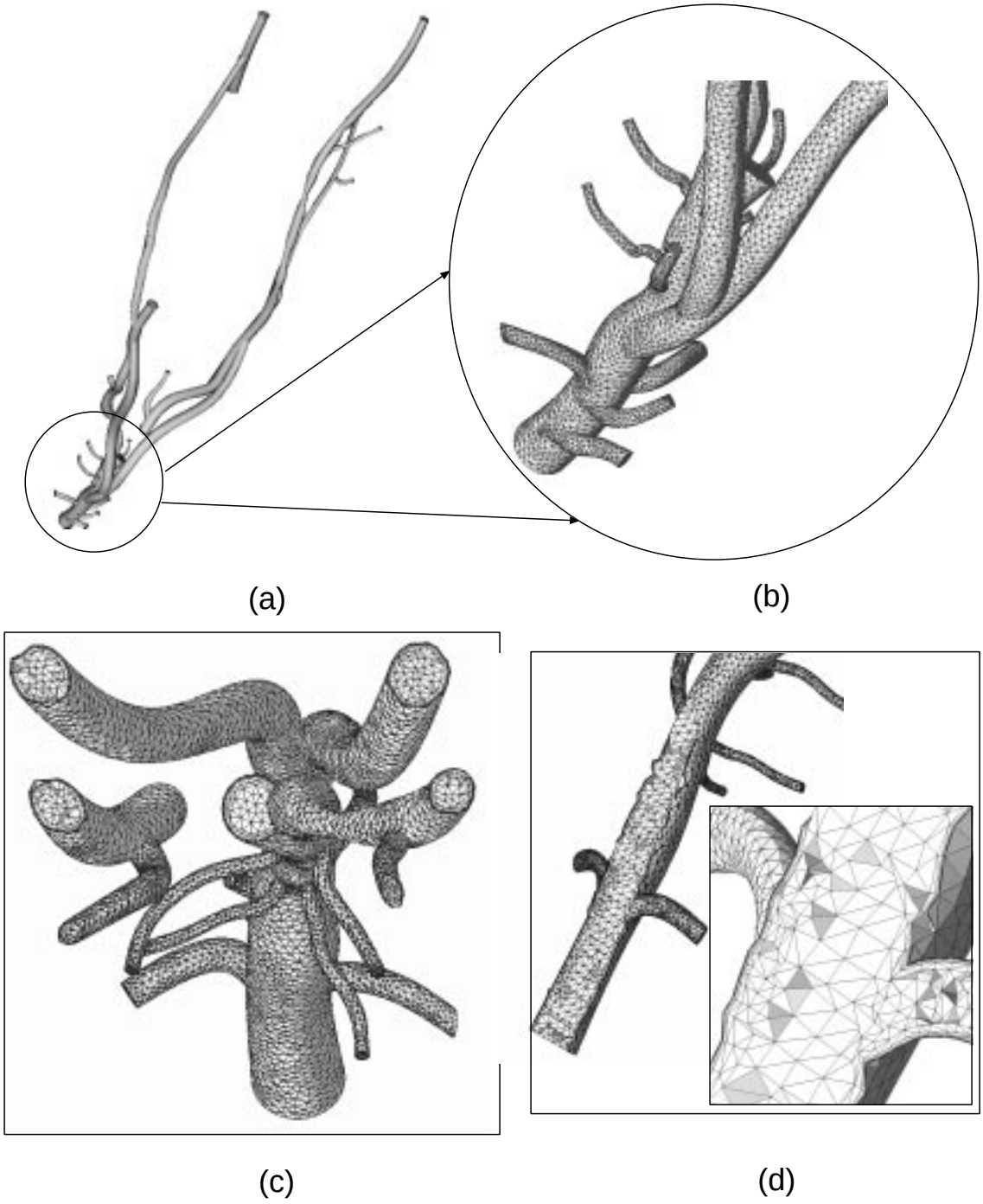


Figure 9.4: Boundary layer mesh for simulation of flow in blood vessels. (a) Geometric model. (b) Zoom in of surface mesh in the encircled region. (c),(d) Cross sections showing the boundary layer and isotropic meshes.

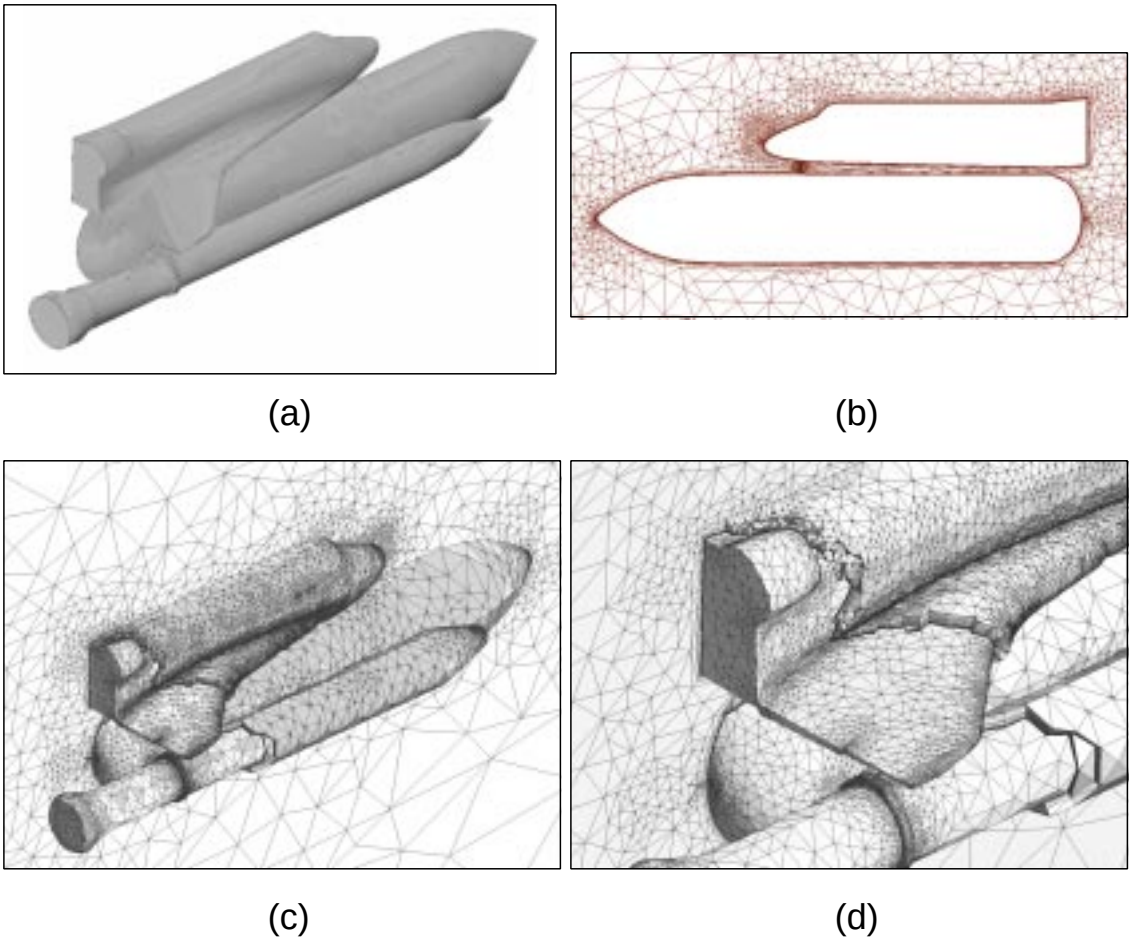


Figure 9.5: Boundary layer mesh for space shuttle, (a) Model geometry. (b) Retriangulated surface mesh. (c) Cut away of boundary layer mesh. (d) Close-up of boundary layer showing anisotropic elements.

9.3 Validation

9.3.1 Laminar flow over flat plate

The first example used to demonstrate the capabilities of the mesh generator to appropriately control element sizes and mesh gradations so as to capture the solution accurately is simulation of laminar flow of an incompressible viscous fluid over a semi-infinite flat plate. The domain used for the simulation is a rectangular plate, which is very thin in the z-direction and is taller in the y-direction than in the z-direction (Figure 9.6a). The flow is in the direction of the positive x axis and is uniform at the inlet with a Reynolds number of 10000. The length of the plate is chosen to be 1.0 unit. The domain in which the flow is simulated itself starts 0.3 units ahead of the plate to capture the flow characteristics around the singular point or the stagnation point at the lead edge of the plate. The domain is 5.0 units in length perpendicular to the plate so that the upper wall is well beyond the boundary layer. The thickness of the domain in the simulation is chosen to be 1.0 unit. The geometric model used for meshing is only $2.0E-04$ units thick so as to restrict the mesh to have only one element through the thickness. Therefore, the mesh is scaled prior to the analysis. The reason for this scaling in the thickness direction is to maintain the influence of the stabilization term in the finite element method (Stabilized Galerkin method) which depends on element size.

The boundary conditions applied to the analysis model are as follows:

1. Uniform flow on the inlet face G_i^2.
2. Symmetry boundary conditions on G_s^2, i. e., $v = 0$.
3. No-slip boundary conditions on the plate, G_p^2, i. e. $u = v = w = 0$.
4. Symmetry boundary conditions on the top face, G_t^2, i. e., $v = 0$.
5. Zero cross flow on front and back faces, G_b^2 and G_f^2, i. e. $w = 0$.
6. Constant pressure on outflow face, G_o^2.
7. Zero viscous traction and pressure on outflow face.

The expected solution in the domain is shown in Figure 9.6b [75]. The fluid shears against the plate due to the no-slip boundary condition and the velocity distribution $u(y)$ at any point downstream of the leading edge shows a smooth reduction from the free stream velocity to zero at the wall. The boundary layer profile as given by Blassius equation is given by:

$$\frac{\delta_{99\%}}{x} \approx \frac{5.0}{\sqrt{Re_x}} \tag{9.1}$$

where $Re_x = \rho U x/\mu$, ρ and μ being the density and viscosity of the fluid respectively.

The first layer thickness in the boundary layer mesh is required to be [75]:

$$\frac{\delta_0}{x} \approx \frac{0.1}{\sqrt{Re_x}} \tag{9.2}$$

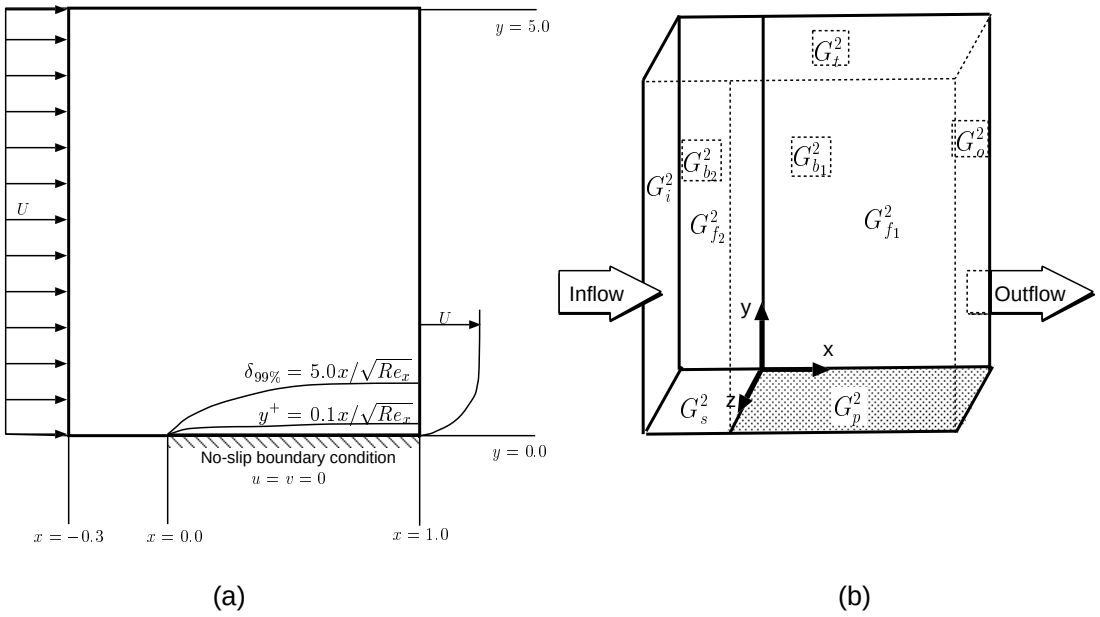


Figure 9.6: Schematic diagrams of setup for simulation of laminar flow over flat plate. (a) Schematic description of domain and important boundary conditions. (b) schematic diagram of geometric model (Figures not drawn to scale).

The surface mesh generated as a starting point for the boundary layer mesh generator is shown in Figure 9.7a,b,c. In the geometric model, the front and back walls ($z = 0$ and $z = 2.24\times10^{-4}$) are each split into two faces for better mesh control. The maximum size of entities in the surface mesh is 0.2 which is achieved only far out in the domain. Since a singularity is expected at the leading edge of the plate, a very small mesh size is requested around that point. The mesh size requested is close to the thickness of the boundary layers at the singular point (note that since the singularity is at $x = 0$ where the boundary layer thickness is zero, a threshold value of $x = 0.04$ is used for calculating the minimum boundary layer thicknesses). All surfaces but the inflow, outflow and top face have an imposed mesh size distribution on them as follows:

1. $G^2_{f_1}, G^2_{b_1}$: $\quad e^{3000(1-\frac{x}{2})y^{2.3}}(5\times10^{-4} + 0.04x\sqrt{x})$
2. $G^2_{f_2}, G^2_{b_2}$: $\quad 5\times10^{-4}e^{(50\sqrt{x^2+y^2})}$
3. $G^2_p : 5\times10^{-4} + 0.04x\sqrt{x}$
4. $G^2_s : max(e^{-x}, 5\times10^{-4})$

The boundary layer mesh size parameters requested are as follows:

1. 20 layers of elements in the boundary layer mesh.
2. First layer thickness of $\max(3\times10^{-4}, 0.00015\sqrt{x})$.
3. Total boundary layer thickness of $\max(0.006, 0.052\sqrt{x})$.

The boundary layer mesh and a zoom-in of the mesh around the singular point is shown in Figure 9.8a,b,c. The mesh has a total of 47040 elements. Note the smooth transition of the boundary layer mesh into the isotropic mesh above the boundary layer mesh and in front of the boundary layer mesh at the singular point.

The solution obtained for this problems is shown in Figures 9.9 and 9.11. Figure 9.9a(i) shows the constant u-velocity contours on the front face while Figure 9.9a(ii) shows a zoom-in of the domain near the singularity. It can be seen that the boundary layer has been captured very well, and so has the behavior of

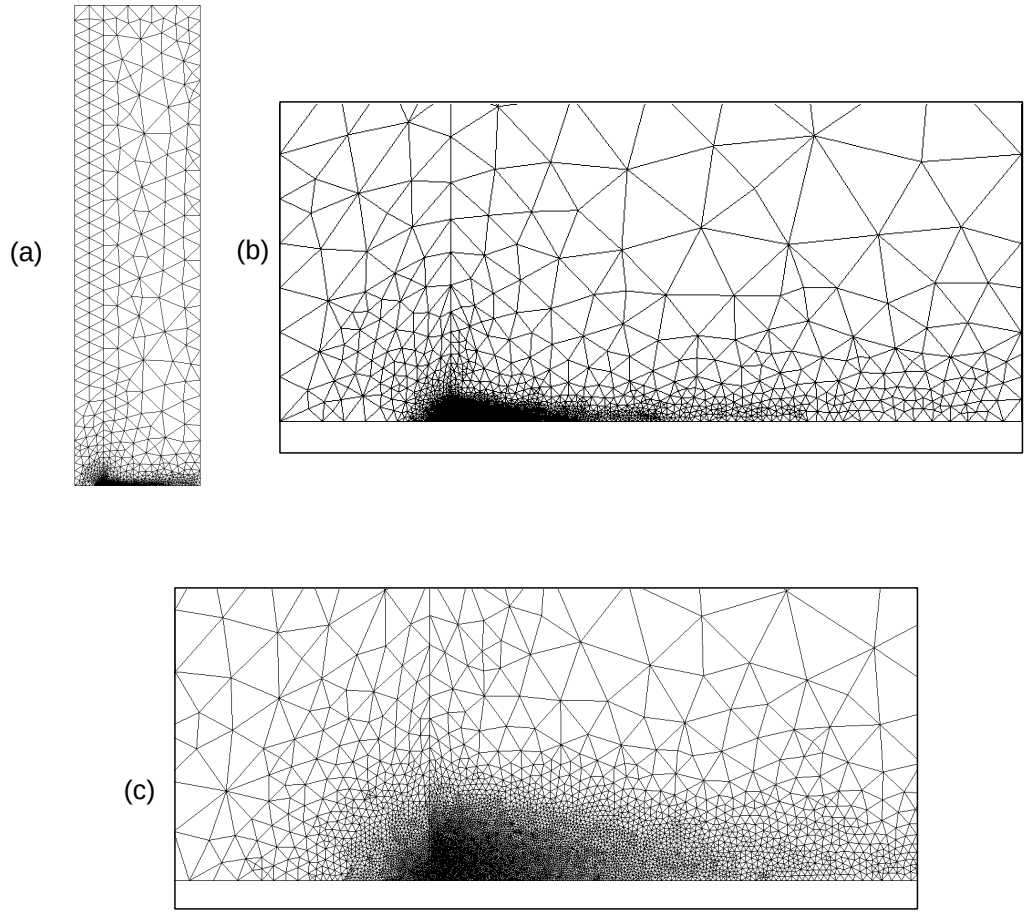


Figure 9.7: Initial surface mesh for flat plate mesh at various zoom factors.

the solution near the singularity. In Figure 9.9c a close-up of the outflow boundary along with u-velocity profile at $x = 1.0$ is shown and is in very good agreement to the expected profile. The results of the simulation are validated using the similarity solution [75] as shown in Figure 9.10 at various points along the flat plate. As seen in the figure the similarity solution of the flow at each of the points matches very well as predicted by theory. Figures 9.11a,b and 9.11c,d show the v-velocity and pressure contours on the front face along with close-ups of the singular point.

9.3.2 Turbulent flow in sharply expanding pipe

The second example used to demonstrate the capability of the boundary layer mesh generator to generate meshes capable of accurately capturing the solution is simulation of turbulent flow in a sharply expanding pipe. A schematic of the domain

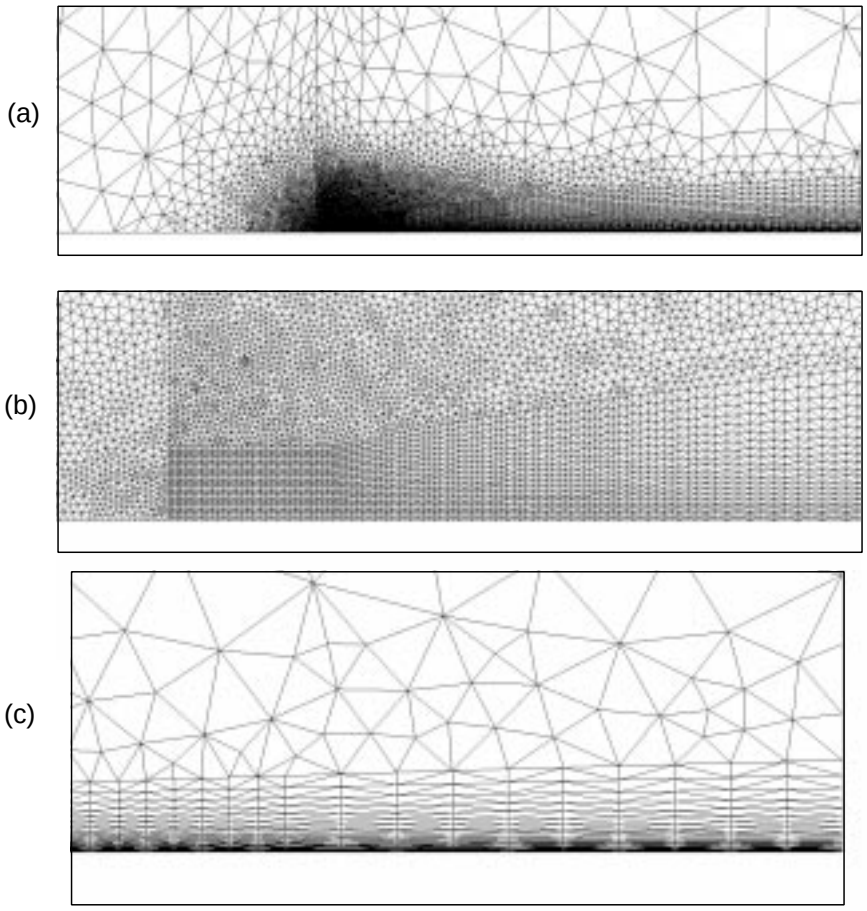


Figure 9.8: Close-up views of Boundary layer mesh for laminar flow over flat plate simulation.

is shown in Figure 9.12a. Fluid enters the narrow pipe which is joined to a large pipe without transition. For this problem, in addition to the boundary layers on the pipe walls, a free shear layer is expected in the flow which leaves the walls at the junction of the two pipes and reattaches to the walls of the large pipe further downstream. A recirculation region is expected behind the shear layers as shown in the figure. It will be shown that with an appropriate model definition the flow can be captured accurately for this problem. The diameter of the small pipe is 1.0, the diameter of the large pipe is 2.0, the length of the small pipe is 2.5 and the length of the large pipe is 15 units. The junction of the two pipes is at $x = 0$ and the axis of the pipe coincides with the x-axis.

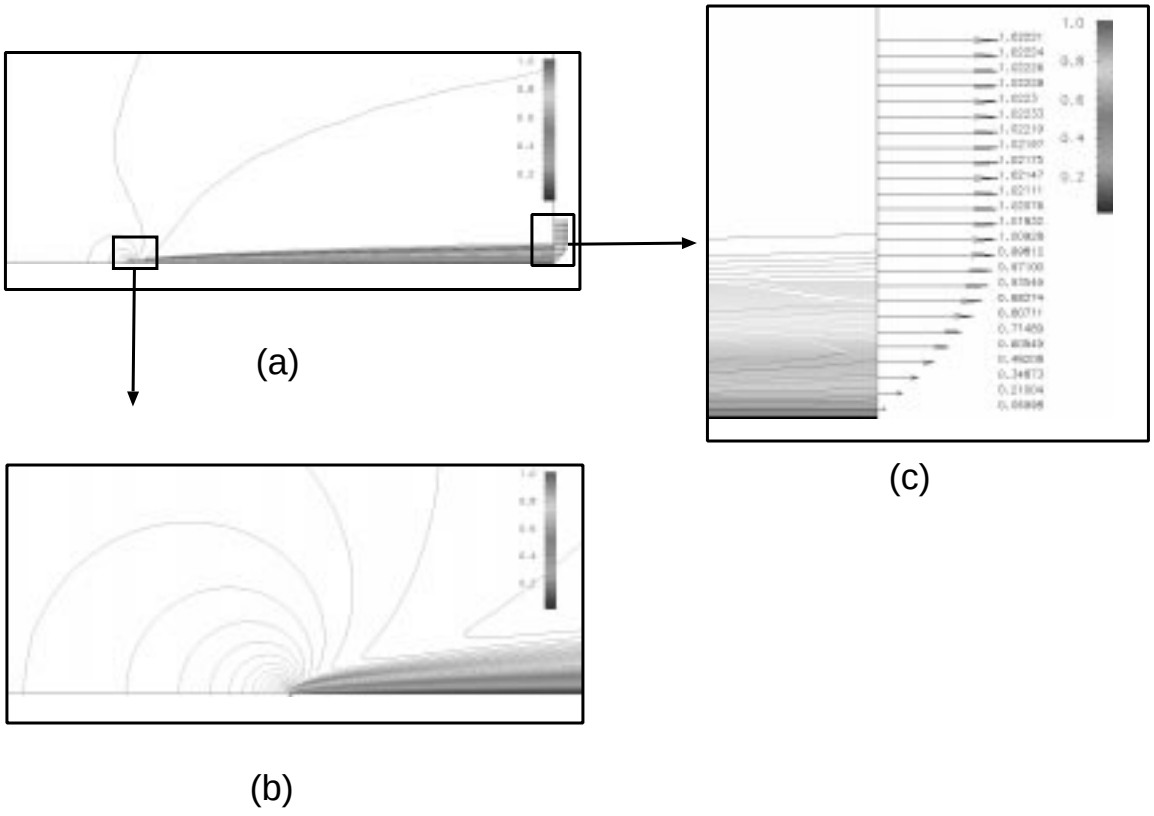


Figure 9.9: u-velocity contours and profile for laminar flow over flat plate. (a) u-velocity contours. (b) Close-up view at singular point. (c) Profile of u-velocity at outflow.

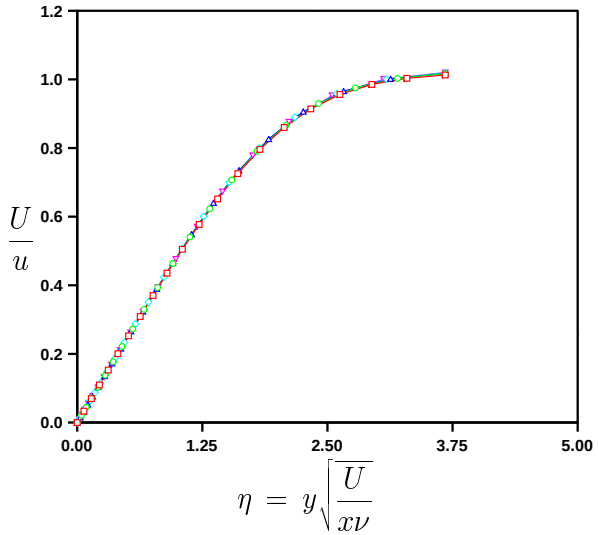


Figure 9.10: Similarity solution of flow over flat plate at various $x = 0.25$, 0.5 0.75 and 1.0

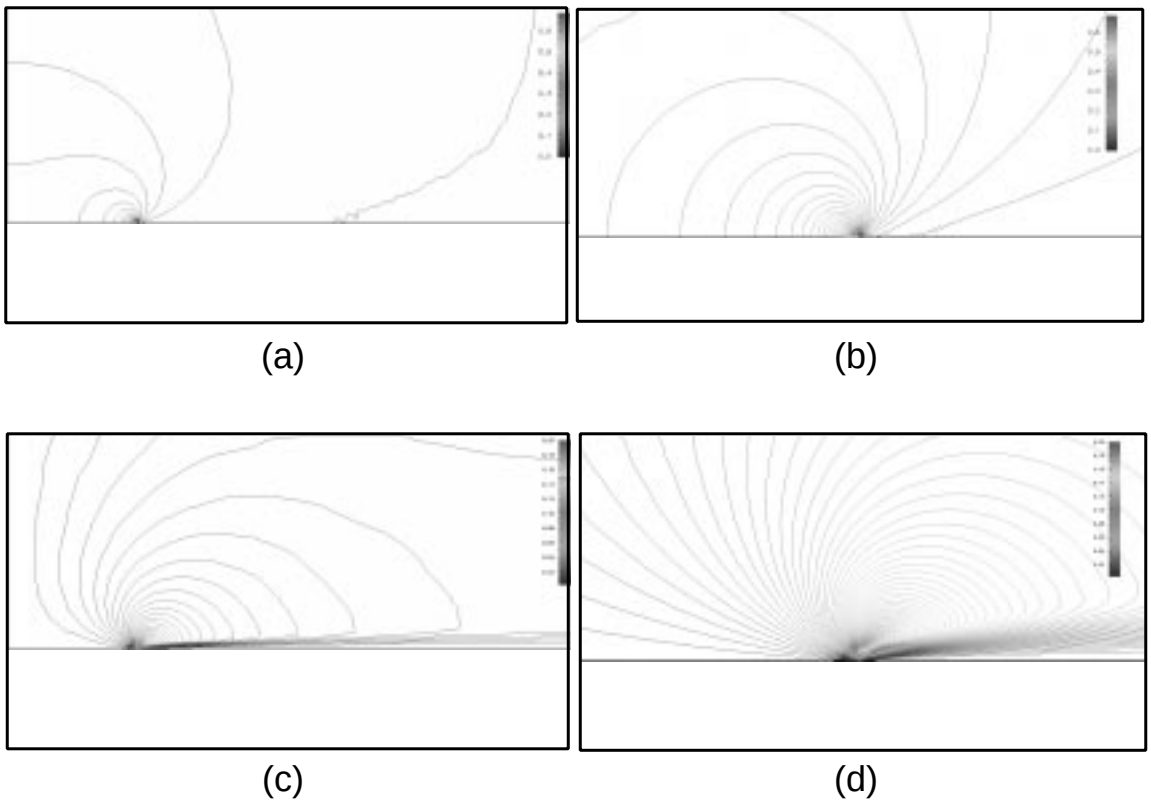


Figure 9.11: Pressure and v-velocity contours for laminar flow over flat plate. (a) Pressure contours. (b) Close-up view of pressure contours at singular point. (c) v-velocity contours. (d) Close-up of v-velocity contours at the singular point.

A schematic of a vertical cross section of the geometric model is shown in Figure 9.12b. Since the generalized advancing layers can build boundary layer meshes on from model boundaries, an artificial surface is defined in the larger pipe based on an estimate of the shear layer path. The two pipes are made into different model regions to maintain better control over the mesh sizes in the interior of the pipes. Furthermore, the shear layer surface divides the larger pipe into two model regions. The large pipe in the geometric model is only a third of the required length and the mesh that is generated in this part is stretched to match the original domain definition. This is done to keep the size of the mesh low and make use of the inherent anisotropy of the whole solution.

The surface mesh (without any stretching) on the original geometric model is shown in Figure 9.13a. The element size is 0.1 in the small pipe, 0.15 between the

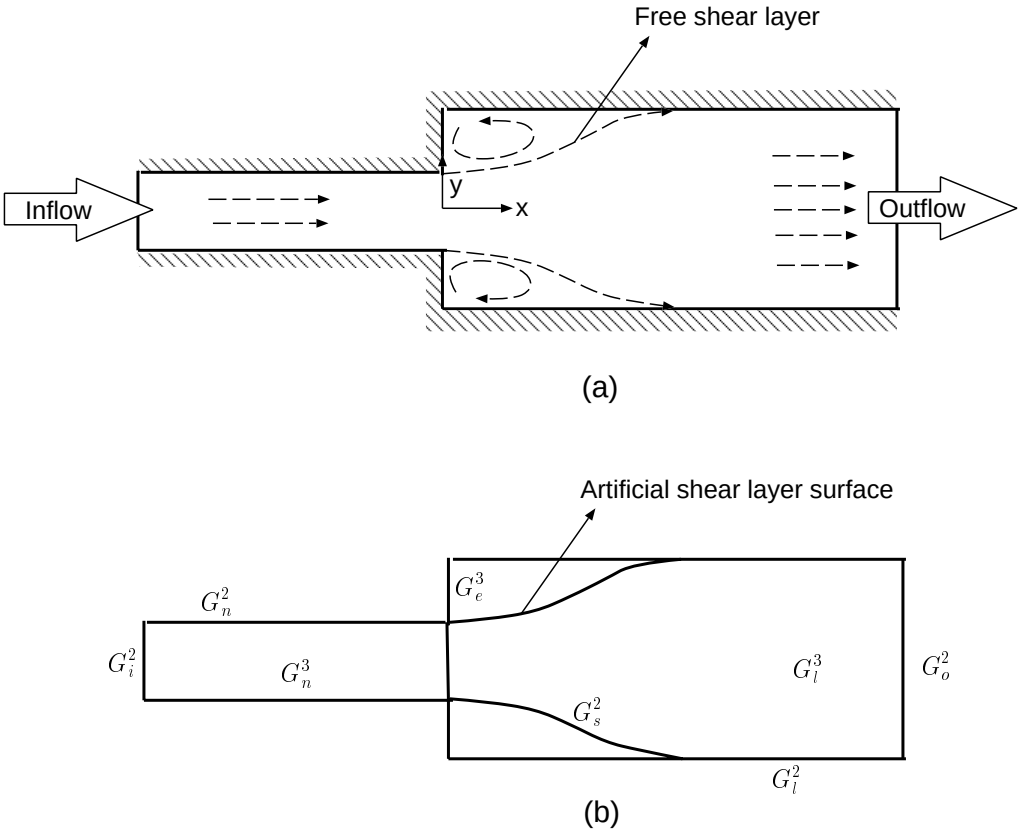


Figure 9.12: Schematic diagram of expanding pipe model. (a) Problem domain. (b) Geometric model cross section.

shear layer and the large pipe and 0.25 in the large pipe. Figure 9.13b shows a cross section of the solid mesh with the different boundary layers clearly visible. Boundary layer elements are created on walls of the small pipe, on both sides of the shear layer surface and on the wall of the large pipe downstream of the reattachment point. The boundary layer thickness and number of nodes are different on the various model faces (and on each side of the shear layer face). The thickness of the boundary layer also varies as a function of the x coordinate. The effect of this is to reduce the total number of elements required for the simulation since the refinement required in the small pipe and on the shear layer need not be carried all the way to the outflow. The specific boundary layer parameters used are (t_0: first layer thickness, T: Total thickness, N: number of layers)

1. Small pipe:

$$t_0 = (1 + 1.6(x + 2.5))\times 10^{-4}, \qquad T = 0.2, \qquad N = 18$$

2. Side 1 of shear surface (towards the interior of large pipe):

$$t_0 = 5\times 10^{-4}, \qquad T = 0.2 + 0.08(x - 2.5), \qquad N = 17$$

3. Side 0 of shear surface:

$$t_0 = 5\times 10^{-4}, \qquad T = 0.2, \qquad N = 12$$

4. Large pipe (downstream of reattachment point):

$$t_0 = (5 + 4(x - 2.5))\times 10^{-4}, \qquad T = 0.4 + 0.08(x - 2.5), \qquad N = 16$$

With these parameters the solid mesh has 311000 elements and largest aspect ratio of elements is approximately 3000.

The boundary conditions for the model are described below given that Reynolds number is $Re = 10^6$, $u_\tau = 0.038$, and the parameters in the similarity solution for the turbulent boundary layer, $\kappa = 0.4$ and $B = 5.5$.

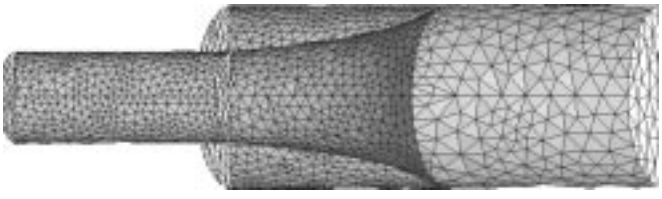
The distance of a point from the wall is $d = 0.5 - \sqrt{y^2 + z^2}$ and the parameter $y^+ = u_t Red$.

1. An inlet velocity profile on the inflow face described as follows:

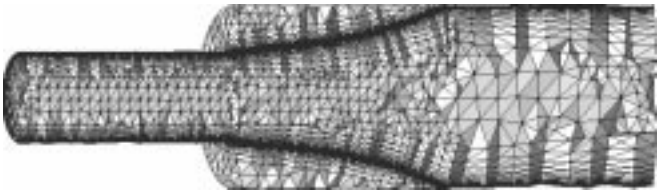
$$u_\tau \min(y^+, B + \frac{\ln y^+}{\kappa})$$

2. No-slip boundary conditions on all the solid wall.

3. Natural pressure of zero on the outflow face.



(a)



(b)

Figure 9.13: (a) Surface and (b) boundary layer meshes for simulation of flow in expanding pipe.

4. An **eddy viscosity** term on all faces as follows:

$$\max(0.0, \min(0.018\times10^{-6}\kappa(e^{\gamma} - 1.0 - \gamma - \frac{\gamma^2}{2}), 1.24\times10^{-3}))$$

where $\gamma = \kappa \min(y^+, B + \frac{\ln y^+}{\kappa})$

The results of the simulation are shown in Figure 9.14. It can be seen that the overall features of the solution have been captured well with the free shear layer clearly captured. However, it is clear that the mesh must have more refinement near the singularity to capture the solution better.

9.3.3 Crystal growth simulation

Figure 9.15[9] shows the results of the simulation of the Czochralski process of bulk crystal growth of indium-phosphide [1] using a boundary layer mesh with approximately 600,000 elements. Finer meshes of 1.5 and 3 million elements have

[9]Courtesy: Dr. Slimane Adjerids, Formerly at Dept. of Computer Science, Rensselaer Polytechnic Institute

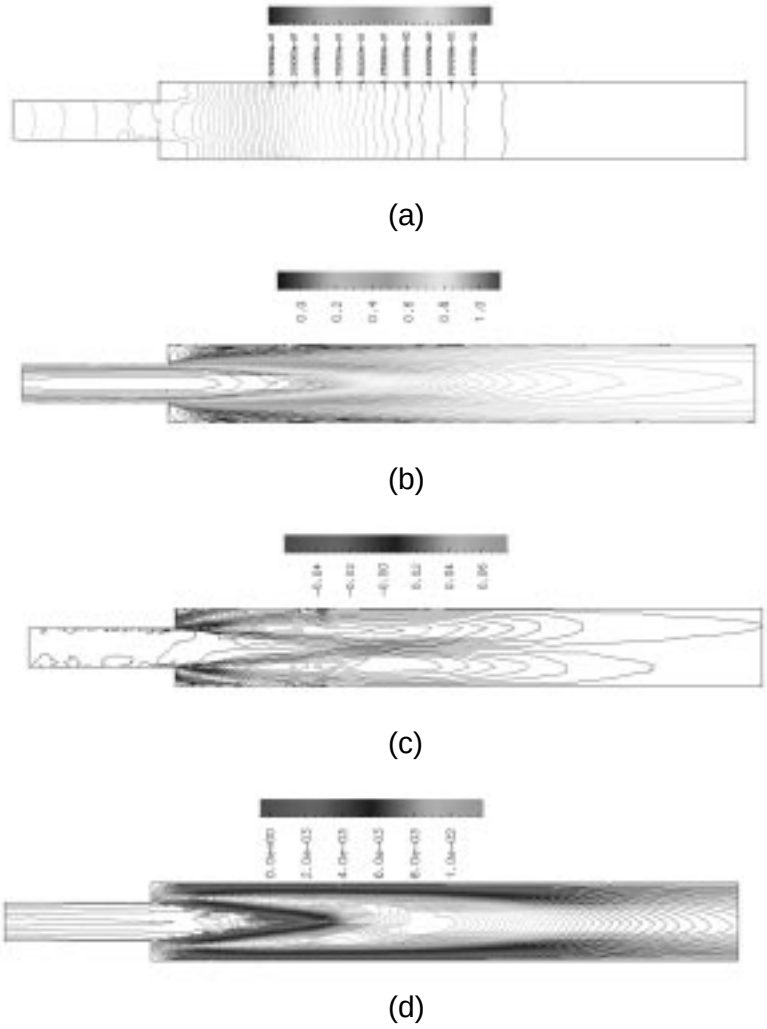


Figure 9.14: Results of simulation for turbulent flow in expanding pipe. All isocurves shown on a vertical cross section through the domain. (a) Pressure distribution. (b) Velocity in x direction. (c) Velocity in y direction. (d) Turbulent or eddy viscosity distribution.

been generated for capturing the solution better. The figure shows velocity vectors and temperature distribution in the horizontal and vertical planes in the domain.

9.4 Timing statistics

The Generalized Advancing Layers method has been observed to produce elements at an average rate of 1000 tetrahedra per second or 3.6 million elements

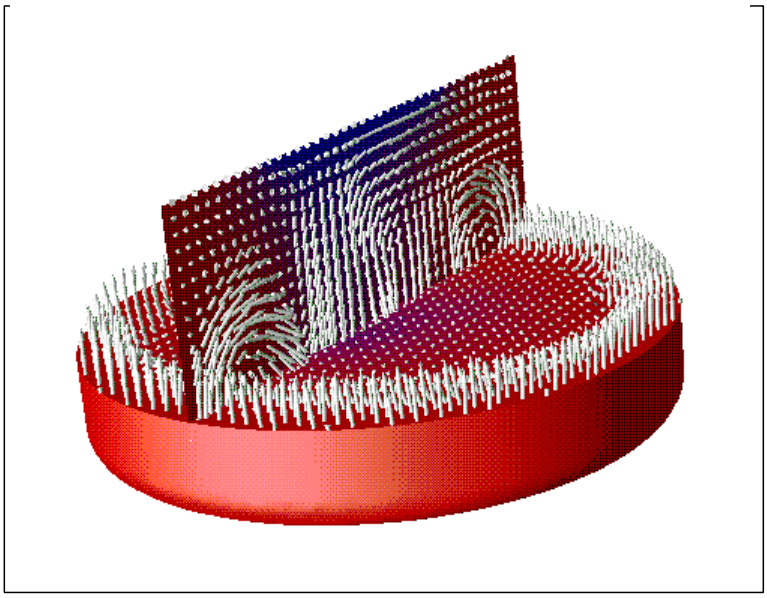
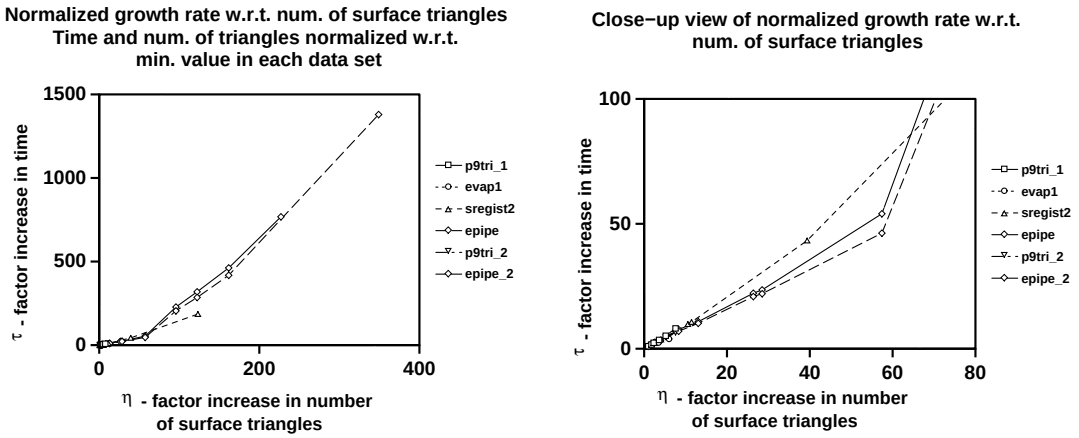


Figure 9.15: Crystal growth simulation - velocity vectors and temperature distribution on the horizontal and vertical planes through a crucible for the simulation of the Czochralski process of bulk crystal growth of indium-phosphide.

an hour on SUN Ultra Sparc 2 workstation. The maximum obtained rate of mesh generation is 2200 tetrahedra per second or 7.9 million elements per hour.

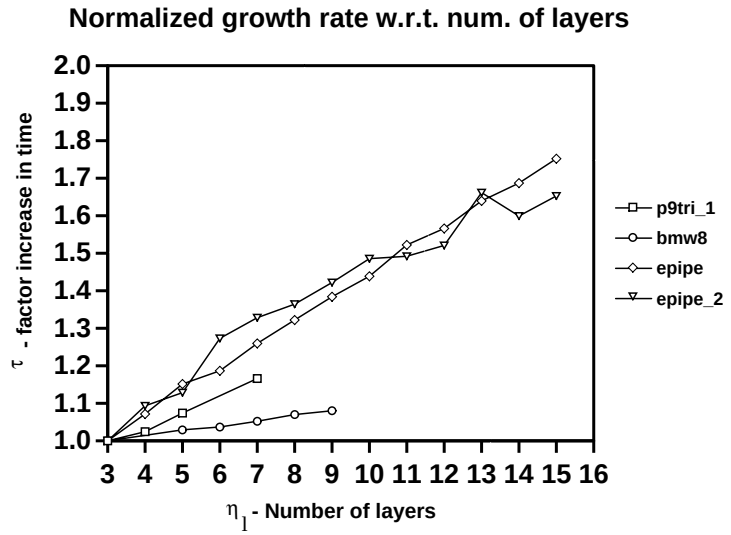
The growth rate of the algorithm with respect to the num of layers and with respect has been observed to be $O(NlogN)$ as shown in Figures 9.16 and 9.17. Figure 9.16a shows the factor increase in time to generate the boundary layer mesh versus the factor increase in the number of surface mesh faces. A zoom-in near the origin is shown in Figure 9.16b. Figure 9.17 shows the factor increase in the time to generate the boundary layer mesh versus the requested number of layers in the mesh. The $O(NlogN)$ behaviour is clearly evident in both graphs and is expected due to the use of a search tree to resolve intersections.



$$\tau = \frac{Time\ to\ create\ mesh}{Time\ to\ create\ first\ mesh\ in\ dataset}$$

$$\eta = \frac{Num\ of\ surface\ triangles}{Num\ of\ surface\ triangles\ in\ first\ mesh\ in\ dataset}$$

Figure 9.16: (a) Growth rate of boundary layer mesher with respect to number of surface triangles (b) Close-up view of graph near the origin



$$\tau = \frac{Time\ to\ create\ mesh}{Time\ to\ create\ first\ mesh\ in\ dataset}$$

Figure 9.17: Growth rate of boundary layer mesher with respect to number of layers

CHAPTER 10
TETRAHEDRAL MESH GENERATION WITH MULTIPLE ELEMENTS THROUGH THE THICKNESS - INTRODUCTION

10.1 Motivation

Finite element simulations with linear elements in domains with thin sections present a special challenge to mesh generation algorithms. A few examples of such models are shown in Figure 10.1. If the solution has strong gradients across the thin section, the mesh must have multiple elements through the thin section to capture these gradients. Flow simulations are particularly susceptible to this problem since prescribing a no-slip essential boundary condition on the walls of a thin section spanned by one linear element incorrectly precludes any flow through the element in the solution. Even when the problem is not as severe, obtaining a mesh with a certain number of elements through the thickness is an important consideration in many areas such as heat transfer, structural mechanics, electro-magnetics and biphasic analysis of soft tissue. If the solution gradients are not as strong in the tangential direction then isotropic refinement to introduce multiple elements through the thickness leads to too many elements along the other direction. Therefore, it is desirable to generate anisotropic meshes with multiple elements through the thickness that are stretched in the tangential direction.

The challenges of this mesh generation problem lie in:

- automatically identifying thin sections of arbitrarily complex models which do not necessarily have matching topology on either side of the thin sections.
- ensuring that the final unstructured tetrahedral mesh has the requested number of elements through these sections without over-refining in other directions.

This is a very practical problem faced by analysts dealing with automatic mesh generators who often revert to using semi-automatic mesh generation techniques to obtain a usable mesh.

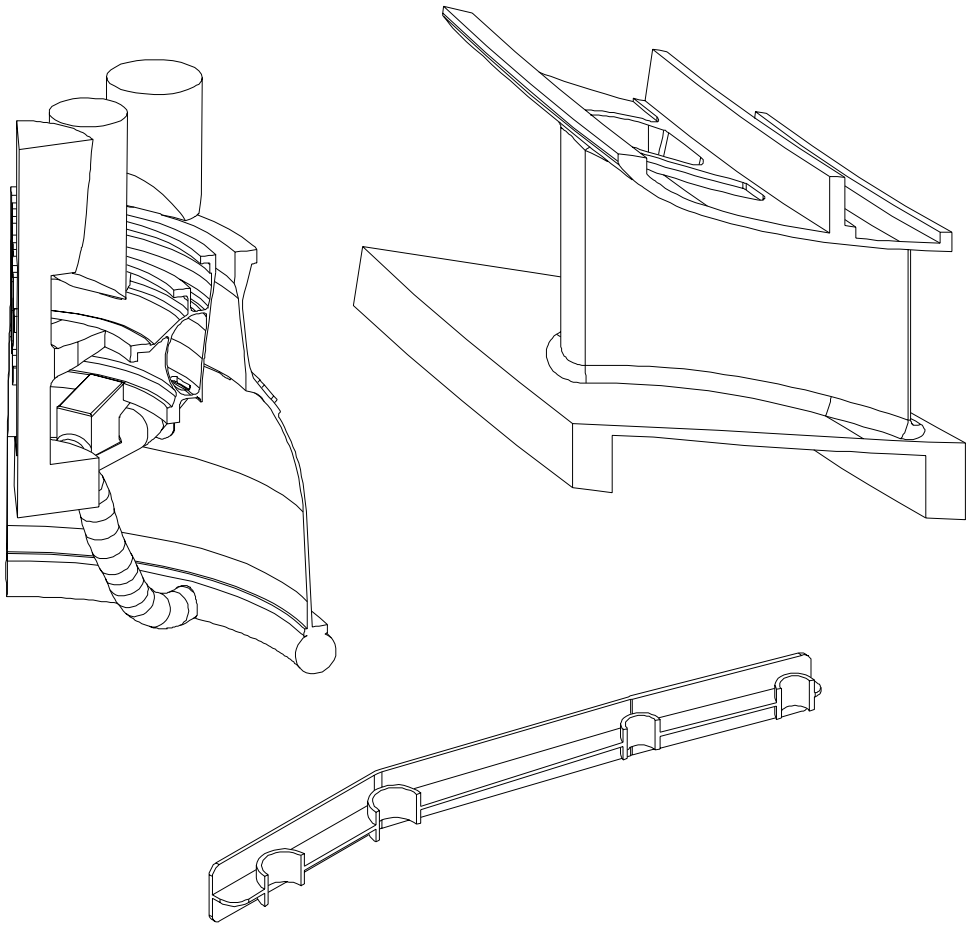


Figure 10.1: Examples of models with thin sections.

10.2 Review of Previous Work

From the literature, there appears to be no automatic mesh generator that specifically addresses this important problem. There are, however, a number of general isotropic and anisotropic mesh generation techniques that may be attempted to be used for generating such meshes.

Although not automatic, mapped mesh generators available in many CAD systems merit mention in the context of creating meshes with multiple elements through the thickness. Thin sections of a model may be isolated during the process of decomposing a model into simpler subdomains. Then mesh sizing information can be prescribed on these sections with more points across the thin sections that

along them. This naturally produces an anisotropic mesh with a guaranteed number of elements across the thin sections of interest. Although very attractive in its guarantee of the number of elements through the thin sections, this method is impractical for meshing arbitrarily complex and large domains. Not only is the process time consuming, the model may not even have clearly demarcated thin sections or matching topology on either side of the thin sections.

General automatic mesh generation and mesh refinement procedures are attractive options since they require no special procedures to generate the mesh. Also with isotropic refinement [15, 55] the quality of the refined mesh can be controlled even in three dimensions [3, 38]. However, in the absence of any definition of thin sections, general refinement procedures are impossible to apply to obtain meshes with the requested number of elements through the thickness. Also, isotropic refinement leads to considerable over-refinement (one or more orders of magnitude) in directions with relatively weaker gradients resulting in greater computational costs.

Of the anisotropic mesh generators discussed in Chapter 2 the Delaunay method for generating anisotropic meshes and the advancing layers methods may be of limited use in generating meshes with multiple elements through the thickness (See Chapter 2 for a brief description of the Delaunay method). To use the Delaunay method, an anisotropic metric must be constructed from the model definition so that elements are stretched in the tangential direction of the thin section and compressed in the thickness direction. To be automatic, this still requires a definition of thin sections in a geometric model. Identification of thin sections in a general geometric model and the construction of an anisotropic metric using modeler enquiries can both be computationally expensive.

The advancing layers method may be considered for generating multiple elements through the thickness. Use of this method would require *a priori* specification or automatic recognition of surfaces forming the opposite walls of thin sections. While this method can be used for generating layers of elements on surfaces forming the walls of the thin section, it is likely to be inefficient since the gap between the opposite walls of the thin section is narrow by definition. Therefore, the layers of elements on opposite walls are more than likely to interfere with each other. Although,

the generalized advancing layers procedure is capable of resolving this interference (See Chapter 8), the computational cost is expected to be high.

Since the existing methods are not well suited to address the issue of generating a mesh with a user defined number of elements through thin sections, a new method to address the issue is proposed here. The method is based on anisotropic refinement of an isotropic mesh followed by local reconnection procedures to generate meshes with multiple elements through the thickness. A definition of thin sections is devised based on deficiency of the starting isotropic mesh. Therefore, the problem of identifying thin sections is recast as a problem of identifying portions of the mesh which do not have sufficient elements through the thickness. After automatically identifying deficient portions of an initial mesh, the necessary number of points are introduced in the thickness directions by performing edge splits. Splitting edges to introduce points does not necessarily eliminate all deficiencies in the mesh and even creates new ones due to the new connections created in the mesh. These deficiencies are eliminated using local reconnection procedures (local edge swapping). Local mesh modification and node repositioning procedures are also used to improve the quality of the final mesh. Results will be presented to show that the method works well for generating meshes with anisotropic refinement through thin sections for arbitrarily complex models.

CHAPTER 11
MULTIPLE ELEMENTS THROUGH THE THICKNESS - IDENTIFYING THIN SECTIONS

11.1 Definition of Thin Sections

Definition 11.1 *An ordered set of mesh edges between two mesh vertices is defined as an edge path. The edge path with the least number of edges among all edge paths between a pair of vertices is called the shortest edge path.*

Definition 11.2 *A mesh is considered to be locally deficient in the thickness direction if the shortest edge path between mesh vertices on opposite model faces has less than a user requested number of edges. Such an edge path is referred to as a deficient path.*

Figure 11.1 shows examples of locally deficient meshes in two types of domains. In Figure 11.1a, the domain has varying thickness and the mesh is uniform. Assuming that three elements are desired through the thickness, the mesh is locally deficient in the thickness direction in indicated portions of the domain. In Figure 11.1b, the domain is of uniform thickness but due to non-uniform refinement of the mesh, only some parts of the domain are considered to be locally deficient.

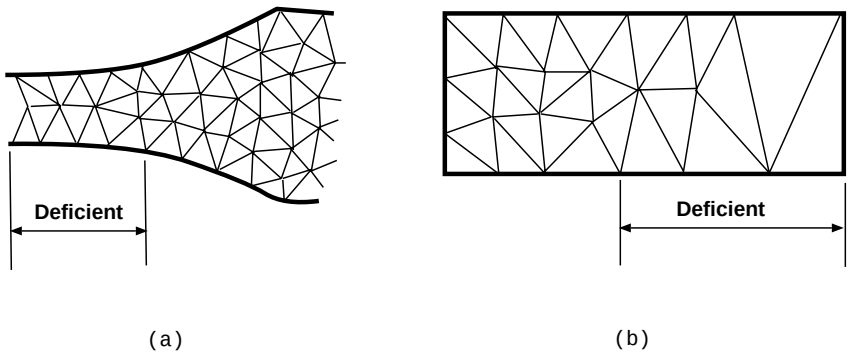


Figure 11.1: Examples of deficient meshes. (a) Deficiency in uniform mesh due to changing cross section. (b) Deficiency in uniform cross section due to mesh gradation.

Definition 11.3 *A mesh vertex, $M_p^0 \sqsubset \overline{G_o^2}$, is defined as an opposite vertex of another mesh vertex, $M_v^0 \sqsubset \overline{G_f^2}$, with respect to model face G_f^2 if*

- *it is the closest to M_v^0 among all $M_p^0 \sqsubset \overline{G_o^2}$.*
- *all edges of the shortest edge path are classified on one model entity $G_e^d \not\subset \overline{G_f^2}$.*

11.2 Determination of Opposite Vertices

The determination of the opposite vertex of a boundary mesh vertex, M_v^0, is a 3 step procedure (refer Figure 11.2) consisting of:

1. *Forward search*: Using the normal of the model face at the mesh vertex M_v^0 as a guiding direction, the forward search attempts to find a candidate opposite mesh vertex, M_c^0 on another (or the same) model face (referred to as the *opposite model face*). The search is conducted along mesh edges.

2. *Boundary search*: The boundary search refines the result of the forward search by finding the closest vertex classified on the opposite model face to the start vertex. This vertex is the *opposite vertex*, M_p^0. The search is conducted along mesh edges and mesh faces on the closure of the model face.

3. Reverse search: The reverse search determines the shortest edge path between the opposite vertex M_p^0 and the start vertex M_v^0. The search is conducted along mesh edges.

Each of these steps are discussed in detail below and it is shown that the computational cost of the search procedure is a constant for each mesh vertex under certain conditions.

11.2.1 Forward search

The forward search assumes the thickness direction at a mesh vertex $M_v^0 \sqsubset \overline{G_f^2}$ with respect to the model face G_f^2 to be the normal to G_f^2 at M_v^0. The true model face normal is preferred over the average discrete normal[10] since it gives more consistent

[10]The average discrete normal at a boundary mesh vertex is average of the normals of all the boundary mesh faces connected to the mesh vertex.

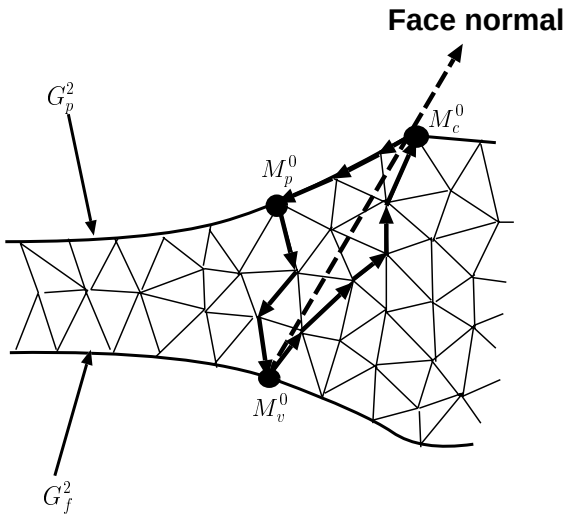


Figure 11.2: Detection of locally thin sections: M_v^0 - start vertex, M_c^0 - candidate opposite vertex, M_p^0 - opposite vertex, G_f^2 - reference model face, G_p^2 - opposite model face.

results. Under this assumption, the forward search attempts to find a potential opposite vertex, M_c^0, approximately in the direction of the normal and classified on the closure of a model face.

Assume that N_t elements are requested through the thickness of a geometric model. Starting from a vertex M_v^0 the forward search proceeds from one mesh vertex to another along mesh edges until a potential opposite vertex is reached. At a given vertex, M_i^0 in the forward search, the following steps are carried out to find the edge along which the search should proceed:

- All mesh edges, $\{M_e^1 | M_i^0 \subset \partial(M_e^1)\}$, connected to the vertex are found.
- The other vertices of the edges, $\{M_j^0 | M_j^0 \subset \partial M_e^1,\ M_j^0 \neq M_i^0\}$, are found.
- For each edge, the vector v_e from M_i^0 to M_j^0, is found.
- Find the vector v_i that is best aligned with the search direction v_f, i. e., find the vector that minimizes the expression $1 - v_e \cdot v_f$.
- The mesh edge $M_i^1 \subset \{M_e^1\}$ corresponding to v_i is the next edge to traverse in the forward search.

The other vertex of M_i^1, $M_{i+1}^0 \subset \partial M_i^1, M_{i+1}^0 \neq M_i^0$ is the next vertex in the forward search.

The forward search ends when a potential opposite mesh vertex is found or sufficient number of edges have been traversed without reaching the closure of a model face. In the true thickness direction it would be sufficient to end the search as soon as $(N_t - 1)$ edges are found in the edge path. However, since the forward search direction is not necessarily aligned with the true thickness direction, the forward search is not ended until $(N_t + 1)$ edges have been found without encountering an opposite model face. A vertex, M_i^0, in the forward search is considered to be a potential opposite vertex, M_c^0, when the following topological conditions are met:

$$M_i^0 \quad \sqsubset \quad G_p^d, \qquad d \neq 3 \tag{11.1}$$

$$d < d' \quad \text{where} \quad M_{i-1}^1 \sqsubset G_e^{d'} \tag{11.2}$$

In addition, given the reference model face normal at the start vertex, $\hat{nv}$ and normals of the N model faces connected to G_p^d at the candidate opposite vertex, $\hat{ni}$, $i = 1, N$ respectively, the following geometric criteria must be satisfied for a model face to be considered an opposite model face.
If

$$\theta_i = \cos^{-1}(\hat{nv} \cdot \hat{ni}), \qquad 0 < \theta_i < 2\pi \tag{11.3}$$

then

$$\pi - \alpha \; < \; \theta_i \; < \; \pi + \alpha, \qquad \forall i = 1, N \tag{11.4}$$

$$\theta_p \; = \; \min\left(\mid \pi - \theta_i \mid\right), \qquad i = 1, N \tag{11.5}$$

where α is an empirical angle tolerance for determining if the two model faces are opposite to each other or not. In practice, the value of α is taken to be $\frac{\pi}{12}$. The check on the dot product of the two normals is necessary to ensure that a forward search proceeding on a boundary does not end at a vertex classified on the boundary

of a model face not really opposite to the reference face. Therefore if none of the model faces satisfy the above condition, the vertex reached is considered to be a spurious opposite vertex and the search for the real opposite vertex continues. The geometric criteria for evaluating opposite model faces is illustrated in Figure 11.3. In the figure, the forward search from $M^0_{v_1}$ first reaches M^0_1 which satisfies the topological conditions for being an opposite vertex. However, it is rejected based on geometric criteria and the search continues on to find $M^0_2 \sqsubset \overline{G^2_{p_1}}$ as a more appropriate opposite vertex. In the figure, the forward search from $M^0_{v_2}$ ends at $M^0_3 \sqsubset G^1_e$ which has two model faces connected to it, $G^2_{p_1}$ and $G^2_{p_2}$. Of the two, $G^2_{p_1}$ is chosen as the opposite model face since its normal at M^0_3 is best opposed to the normal of G^2_v.

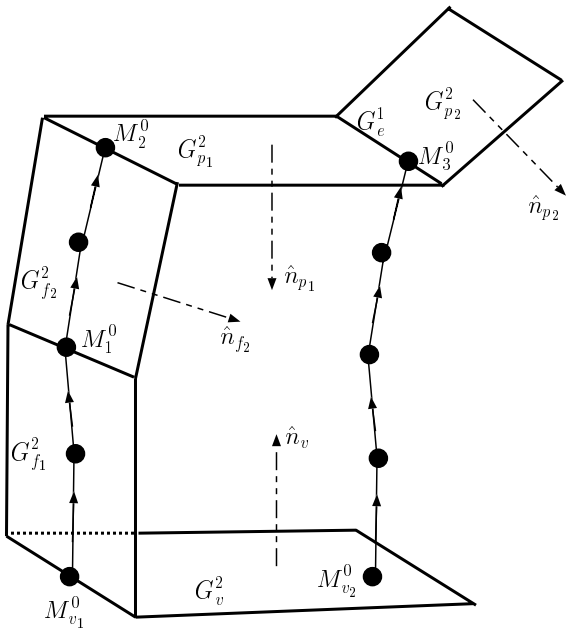


Figure 11.3: Need for geometric check in forward search to avoid termination at spurious opposite vertex.

If the search terminates because $(N_t + 1)$ edges have been traversed, the vertex is labeled as not having an opposite vertex.

As the forward search is terminated when $(N_t + 1)$ edges have been traversed, its computational complexity is $O(N_t)$.

11.2.2 Boundary search

The boundary search is a discrete closest point search on the opposite model face, G_p^2, with respect to M_v^0. Therefore, the closest vertex classified on the closure of G_p^2 to M_v^0 is found in this search. The search starts at the candidate opposite vertex, M_c^0, found in the forward search. From the current vertex, say M_i^0, the search goes to an "adjacent" vertex M_j^0 which is closer to M_v^0 than M_i^0. Given an edge M_e^1 or a face M_f^2 connected to M_i^0, an "adjacent" vertex is an edge-connected or face-connected vertex defined as:

$$M_j^0, \quad \text{such that} \quad M_i^0, M_j^0 \subset \partial M_e^1, \quad \text{and} \quad M_i^0, M_j^0, M_e^1 \sqsubset \overline{G_p^2}, \tag{11.6}$$

or

$$\begin{gathered} M_j^0, \quad \text{such that} \quad M_j^0 \subset \partial M_{f'}^2, \quad M_i^0 \not\subset \partial M_{f'}^2, \\ M_f^2 \cap M_{f'}^2 = M_{e'}^1 \quad \text{and} \quad M_i^0, M_j^0, M_{e'}^1, M_f^2, M_{f'}^2 \sqsubset \overline{G_p^2} \end{gathered} \tag{11.7}$$

To find the edge connected adjacent vertex, find all edges classified on the closure of the model face connected to the current vertex and find their other vertices. To find the face connected neighbors, find all faces classified on the model face connected to the current vertex. For each of these faces, find the edge opposite to the current vertex and check for another face connected to this edge and classified on the model face. If such a face exists, then the vertex of the face opposite the common edge of the two faces is a face connected neighbor of the current vertex. Edge and face connected neighbors are shown in Figure 11.4a. Face neighbors must be accounted for in the discrete point search since it is possible for all the edge neighbors to be farther away from the start (or reference) point yet have a face neighbor closer to the reference point than the current vertex (See Figure 11.4b).

The procedure is repeated until no adjacent vertex is closer to the reference vertex than the current vertex. If the boundary search ends on a vertex classified on the boundary of the original model face, G_f^2, the forward search is repeated ignoring that M_c^0. The boundary search is then repeated with the new result of the forward search. It is reasonable to assume that the number of edges traversed

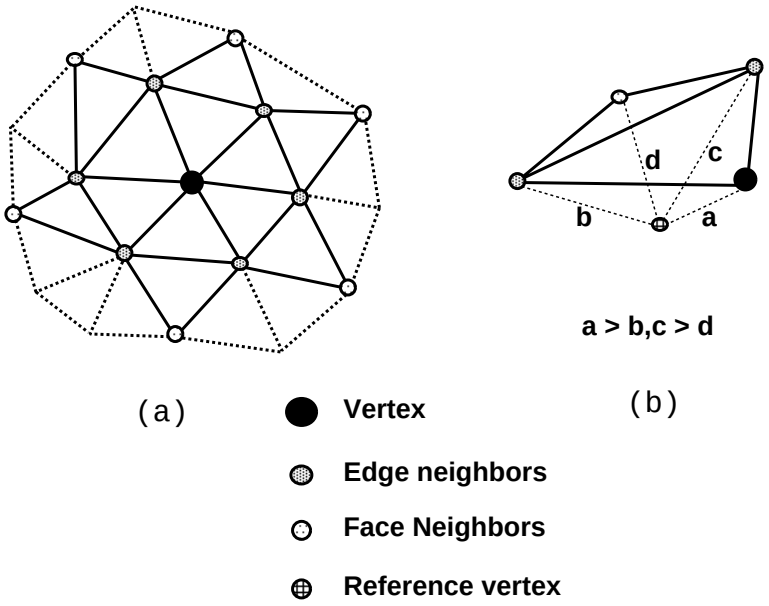


Figure 11.4: Edge- and face-connected neighbor vertices of a vertex.

in the boundary search is on the order of N_t since it is expected that the forward search direction is not too far off from the true direction in most cases. Therefore the computational cost of the boundary search can be taken to be $O(N_t)$.

11.2.3 Reverse search

The reverse search first tries to find a path between the opposite vertex, M_p^0 and the start vertex M_v^0 using the same search method as the forward search. The reverse search is required since the forward search yields a path between start vertex and the candidate opposite vertex, which may not be the actual opposite vertex. The search direction of the reverse search is the vector from M_p^0 to M_v^0. The search is along mesh edges as in the forward search. If this fails to reach the start vertex, a greedy shortest path algorithm is applied between the opposite vertex and the original vertex. The greedy reverse search begins with M_v^0 as the target and M_p^0 as the current vertex. From the current vertex, the search goes to an adjacent edge connected vertex which minimizes the distance to the target vertex. This is repeated until the target vertex or a local minimum is reached. If a local minimum is reached, the direction of the search is reversed and attempted again from the start vertex to

the opposite vertex. If the shortest path still cannot be found, M_v^0 is assumed to have no opposite vertex.

Since the reverse search is subject to a limit on the number of iterations and the steps of the reverse search and the forward search are similar, the computational complexity of the reverse search may be assumed to be on the order of the forward search. Therefore, the computational cost of the reverse search can be taken as $O(N_t)$.

Thus, the total cost of finding an opposite vertex for any boundary mesh vertex may be considered to be $O(N_t)$ or a constant, since N_t is a constant for a given problem. If there are N_b boundary mesh vertices, the total cost of finding their opposite vertices is $O(N_b)$.

Once the search for opposite vertices of all the mesh vertices on the closure of model face is complete, an additional check is made to verify if any logical choices for opposite vertices may have been missed during the search process. If a mesh vertex does not have an opposite vertex and all its edge connected neighbors have opposite vertices, it is assumed that the vertex must also have an opposite vertex. Therefore, the closest of the opposite vertices of the adjacent vertices is taken as an opposite vertex to such a vertex.

The algorithm to determine opposite vertices is designed to efficiently find opposite vertices for creation of multiple elements through the thickness. Although the algorithm is not guaranteed to always find the absolute best opposite vertex, it has been found to do quite well in finding the best or nearly the best opposite vertex.

CHAPTER 12
MULTIPLE ELEMENTS THROUGH THE THICKNESS - ELEMENT CREATION

12.1 Point Creation

The introduction of the required number of elements in the mesh through the thickness is done by splitting edges of paths that have fewer edges than necessary (Figure 12.1). The edges of a path to be split are picked in decreasing order of their lengths. If there are fewer edges in the path than vertices to be added, edges may be split more than once. Edge split points are determined by bisection and subsequent snapping of the calculated point to the boundary, if the edge is a boundary edge. If snapping to the boundary results in the new regions being invalid, the edge is not split.

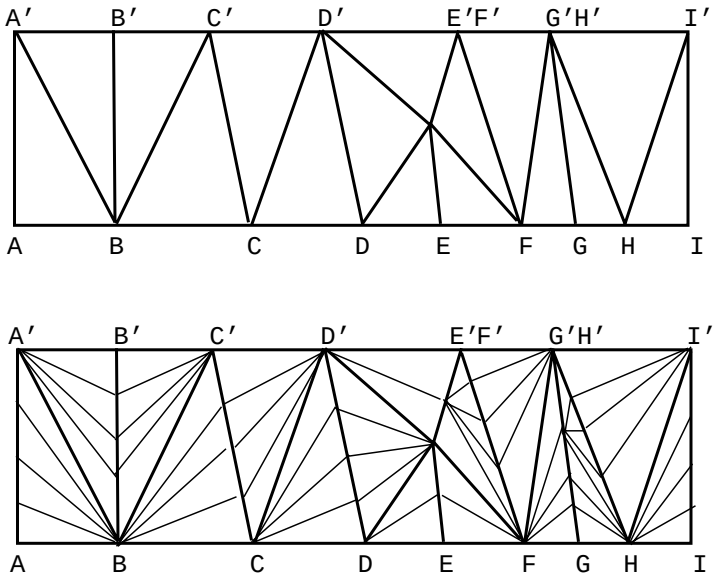


Figure 12.1: Creation of multiple nodes through the thickness by edge splitting - 2D example.

Since splitting of edges changes the paths determined in the initial opposite vertex search, the path information is updated as its edges are being split. This is more efficient than redoing the opposite vertex search.

12.2 Realignment of Edges

From Figure 12.1 it can be seen that splitting of path edges:

- does not eliminate all deficient paths through the mesh,
- creates new deficient paths through the thickness,
- creates large number of connections at some vertices, and
- results large face angles in 2D and large dihedral angles in 3D.

Therefore, realignment of the diagonal mesh edges along and perpendicular to the thickness direction is necessary as shown in Figure 12.1. The realignment of edges is accomplished through the use of edge swapping. The process of edge swapping is equivalent to retriangulation of the polyhedron formed by deletion of all regions connected to the edge such that the new triangulation does not contain any new points and does not contain the edge being swapped (Appendix A).

Definition 12.1 *A mesh edge, $M_i^1 \sqsubset \overline{G_k^2}$, is defined as an opposite edge of another mesh edge, $M_j^1 \sqsubset \overline{G_l^2}$, if each of the vertices of M_i^1 are opposite to one of the vertices of M_j^1.*

Definition 12.2 *A mesh face, $M_i^2 \sqsubset G_k^2$, is defined as an opposite face of another mesh face, $M_j^2 \sqsubset G_l^2$, if each of the edges of M_i^2 is opposite to one of the edges of M_j^2.*

The idea of opposite edges and faces is illustrated in Figure 12.2. In the figure, M_1^0, M_3^0, M_5^0, M_7^0 and M_9^0 are opposite to M_0^0, M_2^0, M_4^0, M_6^0 and M_8^0 respectively. Also, the edge M_1^1 is opposite to edge M_0^1 and the face M_1^2 is opposite to M_0^2. However, the edge M_2^1 connecting vertices M_4^0 and M_6^0 does not have an opposite edge since an edge does not exist between the opposite vertices, M_5^0 and M_7^0. Face

M_2^2 does not have an opposite face because two of its edges do not have opposite edges. On the other hand, in spite of all vertices and edges of face M_3^2 having opposite vertices and edges respectively, the face still does not have an opposite face since no common face connects the opposite edges.

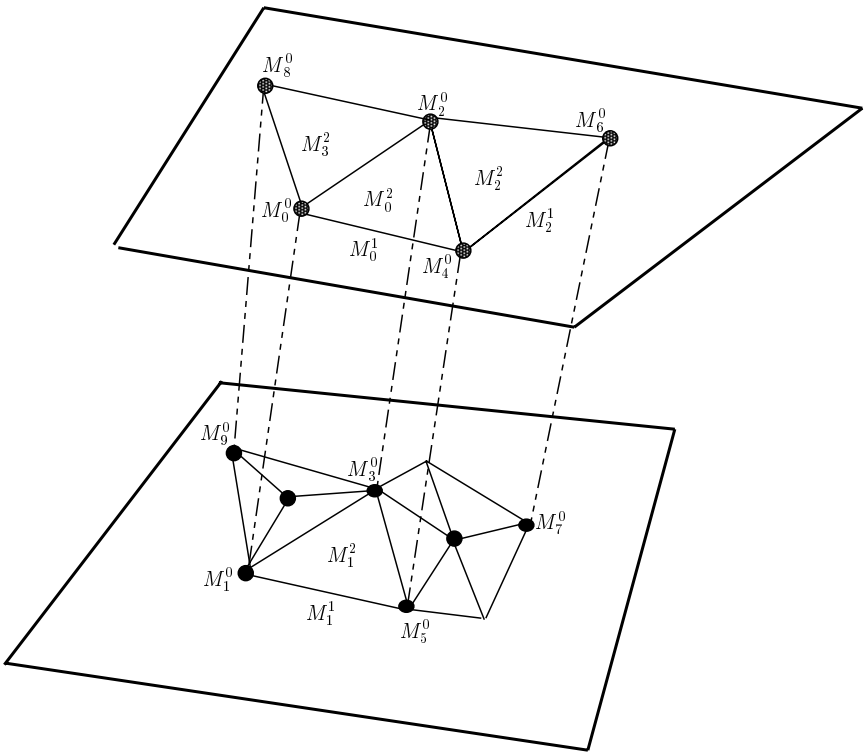


Figure 12.2: Illustration of opposite edges and faces.

12.2.1 Conversion of quads from diagonal to zigzag configurations

The knowledge of the special mesh topology created by splitting is used in the identification of mesh edges to swap and the sequence in which to swap them. The identification of edges to swap is facilitated by abstracting portions of the mesh between opposite faces as wedges (triangular prisms). The lateral faces of such wedges are abstracted as triangulated quadrilaterals. For example, shown in Figure 12.3 is a portion of a mesh in which multiple elements have been introduced through the thickness. In the figure, M_1^2 is the opposite face of M_0^2. One of the 3

abstracted quadrilaterals shown is formed by the vertex set $M_0^0, M_1^0, M_4^0, M_3^0$. The wedge shown is formed by the vertex set $M_0^0, M_1^0, M_2^0, M_3^0, M_4^0, M_5^0$.

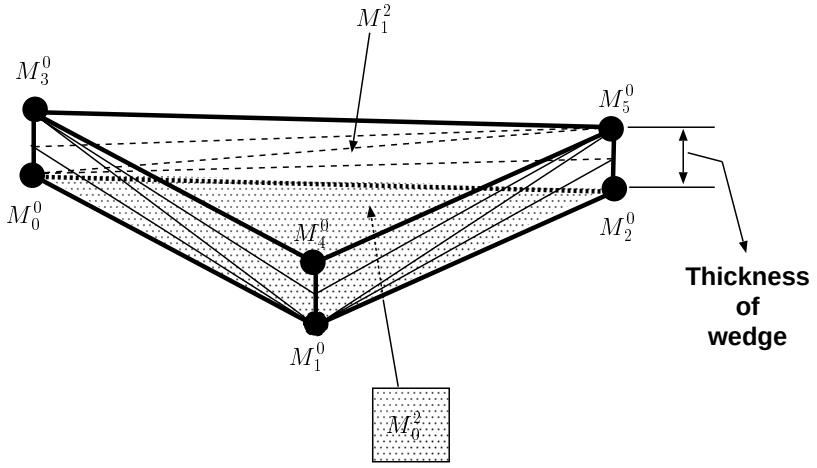


Figure 12.3: Abstraction of mesh between opposite faces as a wedge.

The triangulation on a quadrilateral resulting from edge splitting is shown in Figure 12.4a(i) while the triangulation desired after realignment is depicted in Figure 12.4a(ii). The former triangulation is called a *diagonal triangulation* while the latter a *zigzag triangulation*. A wedge has a diagonal or a zigzag configuration if all of its quadrilateral faces have a diagonal (Figure 12.4b(i)) or a zigzag triangulation (Figure 12.4b(ii)) respectively. If some of the quadrilateral faces have a diagonal triangulation and others have a zigzag triangulation, the wedge is said to be partially zigzag (Figure 12.4b(iii)). Any other configuration is a general configuration that cannot be classified and is dealt with by general mesh modification procedures. Note that the sides of the quadrilaterals in the thickness direction are edge paths between opposite vertices.

With this abstraction defined, a major component of the process of realigning edges along and perpendicular to the thickness direction can be viewed as a conversion of all triangulated quadrilaterals in the mesh from a diagonal to a zigzag configuration. This allows the edge realignment process to be driven largely by topological considerations and is therefore more efficient than using geometric criteria.

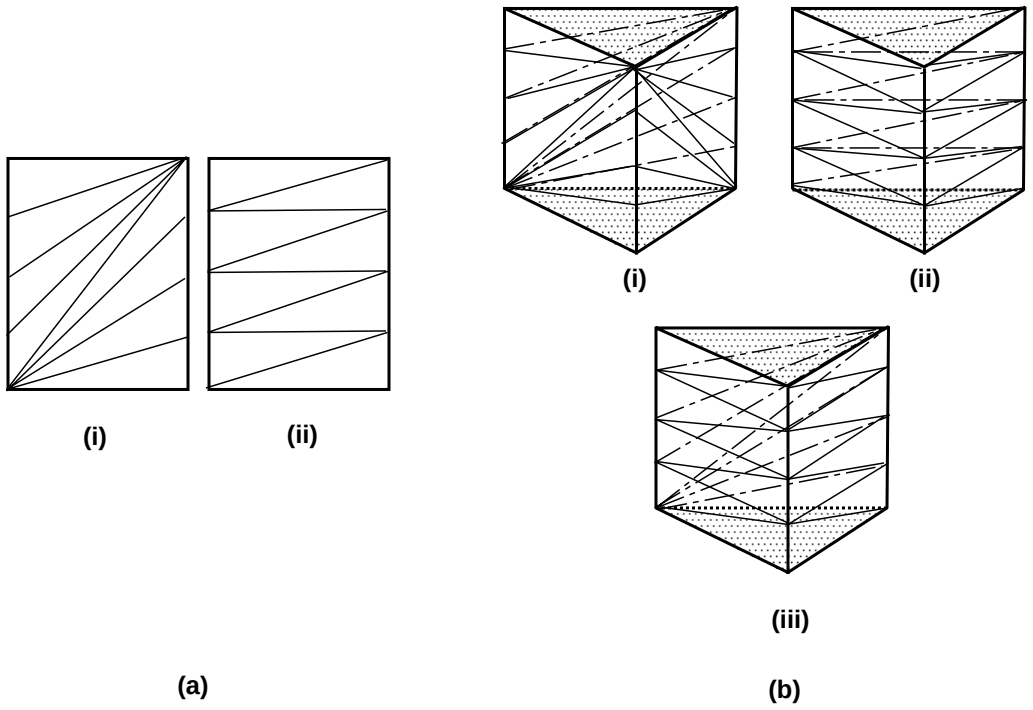


Figure 12.4: (a) Diagonal and zigzag configurations for quadrilaterals. (b) Diagonal, zigzag and partially zigzag configurations for wedges.

The conversion of the diagonal triangulation on each quadrilateral to zigzag is done in a templated sequence of swaps illustrated in Figure 12.5. This sequence can be easily generalized for a quadrilateral with any number of edges through its thickness. Consider a diagonal quadrilateral with N_v vertices along its lateral sides. Let the vertices of the two sides of the quadrilateral be labeled as $\{M^0_{0,0}, M^0_{0,1}, M^0_{0,2}, \ldots, M^0_{0,N_v-1}\}$ and $\{M^0_{1,0}, M^0_{1,1}, M^0_{1,2}, \ldots, M^0_{1,N_v-1}\}$. Then the edge swapping sequence for converting a diagonal quad configuration to a zigzag one can be written as follows

---

for $i = 0$ to $N_v - 1$ **do**
  for $j = N_v - 1$ to $i + 2$ **do**
    Swap edge between $M^0_{0,i}$,$M^0_{1,j}$ to edge between $M^0_{0,i+1}$,$M^0_{1,j-1}$
  end for
end for

If the triangulation created by splitting is such that a quadrilateral is not completely diagonal or completely zigzag then multiple iterations of the above of swaps

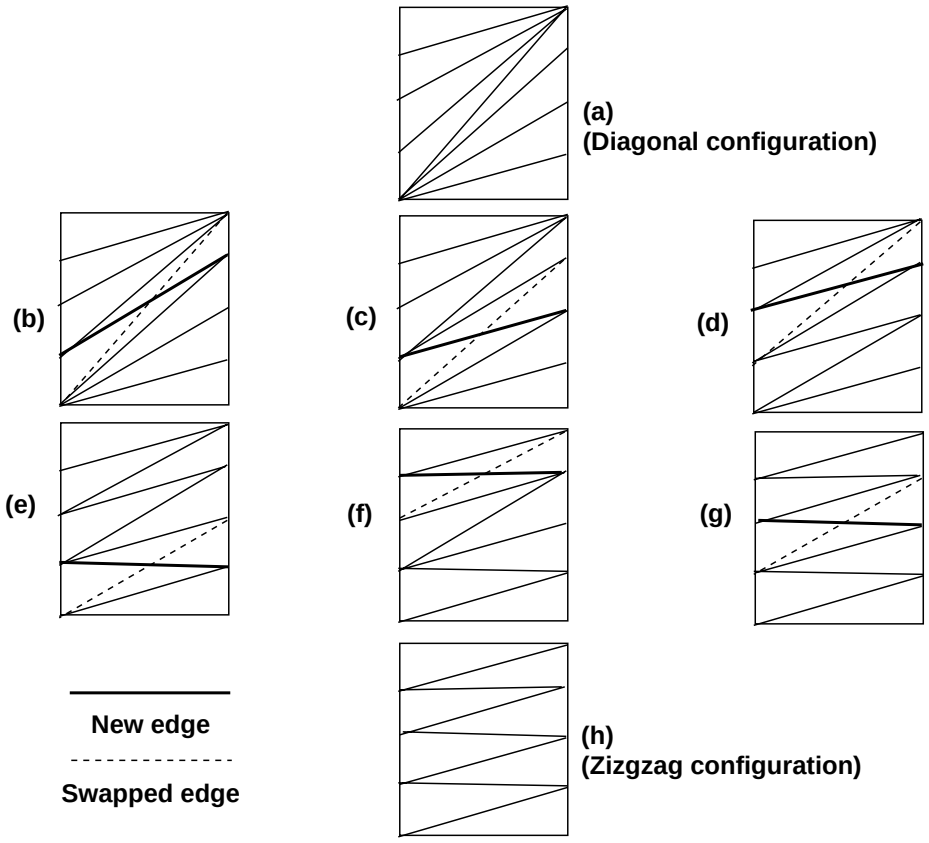


Figure 12.5: Sequence of swaps to convert diagonal quad to zigzag.

is applied to the edges of the quad so that it may be converted into a zigzag configuration. Before each swap it is checked whether the edge actually exists between the vertices $M^0_{0,i}$ and $M^0_{1,j}$.

If the wedge topology is not present in a portion of the mesh, it is still possible to introduce multiple elements through the thickness and realign mesh edges to eliminate deficient paths through the thickness. For general mesh topology through the thickness, edges may be realigned using geometric criteria such as the thickness direction in the local neighborhood. Also, more complex topology of the mesh requires general refinement methods. However, a small number of other cases which can be dealt with are handled specifically in the code and are described below.

12.2.2 Triangle and tetrahedral configuration

A special situation dealt with in the algorithm is two adjacent mesh vertices having the same opposite mesh vertices (Figure 12.6a). In this case, the edge connected to the opposite vertex in each path is ignored and the remaining edges used for the quadrilateral abstraction. Swapping of edges then proceeds as usual with this topology. Similarly, three vertices of a mesh face may have a common opposite vertex forming a master tetrahedron. The three edge paths are split and the edges are swapped on the faces of the master tetrahedron to eliminate any deficient paths in the tetrahedron.

12.2.3 Unswappable diagonal quad

If a diagonal quad's edges cannot be swapped to convert it into a zigzag configuration, then an alternative approach is adopted to eliminate deficient paths in the quad. In this approach, the main diagonal of the quad is also split as many times as the sides of the quadrilateral (See Figure 12.6b). This creates two diagonal configuration "triangles" which can be converted to a zigzag configuration as described above (Figure 12.6b). Further, the direction of the diagonals of the two "triangles" is switched, if possible, since this leads to better face angles. It has been seen that this approach often succeeds when conversion by swapping alone fails.

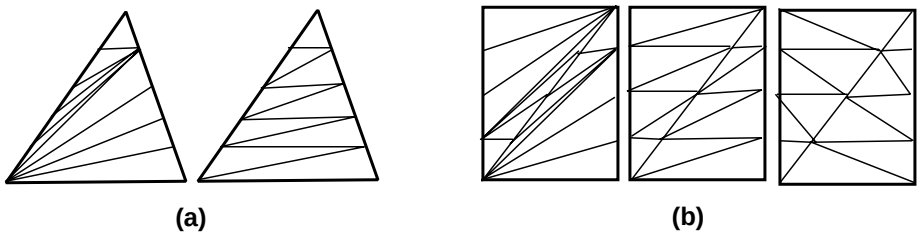


Figure 12.6: (a) Conversion of triangle from diagonal to zigzag configuration. (b) Conversion of quad into two zigzag triangles.

12.2.4 V-triangulation

In the V- quad configurations, a pair of vertices have opposite vertices but the edge between them does not have an opposite edge (or vice-versa). This is illustrated

in Figure 12.7a(i). In this case the diagonal edges comprising the V-configuration are split as many times as the lateral edges of the quad giving rise to three diagonal configuration quads which are then converted to zigzag (Figure 12.7a(ii-iv)).

12.2.5 Star configuration

In the star configuration, the main diagonal of a regular quad is split into two as shown in Figure 12.7b-(i). As before, the original diagonal edges of the quad are split to form a total of six diagonal configuration triangles (Figure 12.7b-(ii)). Thus the star configuration quad can be viewed as a V-quad and an inverted V-quad together. Conversion of the diagonal configuration "triangles" to zigzag follows the same procedure as before (Figure 12.7b(iii-iv)).

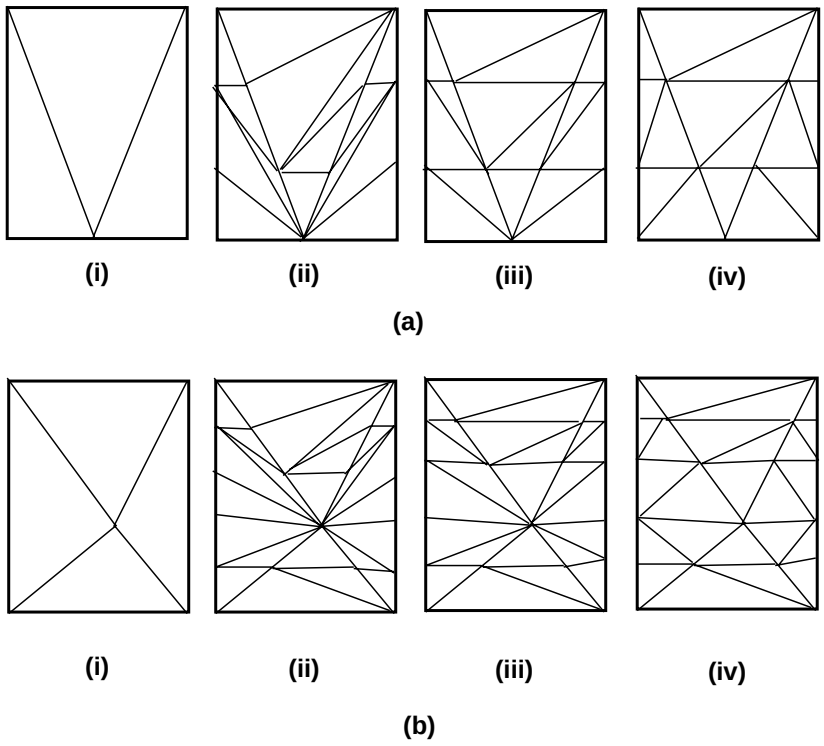


Figure 12.7: Special quad configurations. (a) V-configuration and its conversion. (b) Inverted V-configuration and its conversion (c) Star configuration and its configuration.

The removal of deficiencies in a general configuration mesh using the procedures described up to this point is illustrated in Figure 12.8.

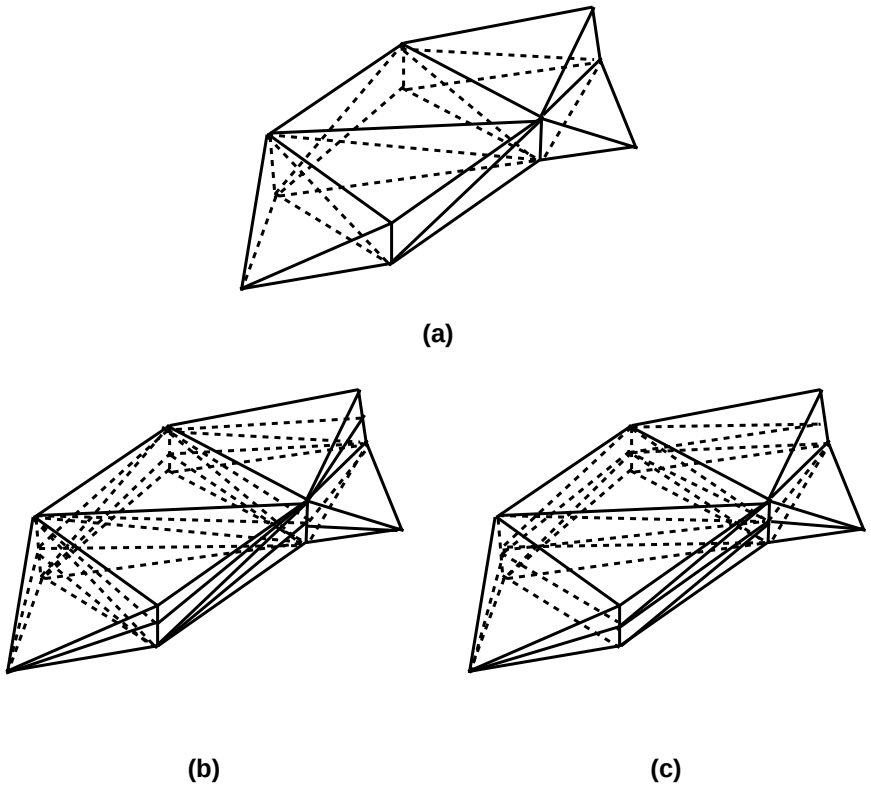


Figure 12.8: Elimination of deficiencies in a general mesh configuration. (a) Initial mesh. (b) Mesh after splitting. (c) Mesh after swapping.

12.3 Constraints in Reconfiguring Wedges using Local Mesh Modifications

The description of the realignment procedure until now assumed that multiple elements were introduced into a mesh which had only one element through the thickness. If the initial mesh had more than one element through the thickness and additional elements were introduced, then multiple wedges, stacked on top of each other, will exist between opposite faces. The procedures for recognition of quadrilaterals and wedges accounts for this. Once the individual wedges through the thickness have been identified, the swapping sequence can be applied to the quadrilateral faces of the wedge as before.

When converting a quadrilateral triangulation from diagonal to zigzag, care must be taken that wedges on either side of the quadrilateral are not forced into a configuration for which a tetrahedronization does not exist. The following rules

may be stated about the validity of wedge configurations with 2 elements through the thickness (See Figure 12.3):

- *Rule 1*: If the directions[11] of triangulations of all 3 quadrilaterals of a wedge is the same, then the wedge cannot be tetrahedronized, regardless of the type of the triangulations.
- *Rule 2*: If 2 of the triangulations are diagonal, their directions must be opposite for the wedge to have a valid tetrahedronization. The direction of triangulation of the third quadrilateral is immaterial.
- *Rule 3*: If 2 of the triangulations are zigzag, at least their directions must be the same for the wedge to have a valid tetrahedronization. In addition, if the third triangulation is zigzag, its direction must not violate Rule 1; if it is diagonal, its direction is immaterial.

Corollary: *If the number of edges through the thickness of the wedge is more than 2, then no valid tetrahedronization is possible with 1 diagonal and 2 zigzag triangulations (i.e. Rule 3 no longer holds). Rule 1 and Rule 2 are still valid.*

The invalidity of all wedge configurations with two zigzag and one diagonal triangulation for wedges with more than two elements through their thickness places an important restriction on the mesh enrichment process. This restriction is that multiple elements must be introduced iteratively with edges through the thickness being split only once before the realignment procedure is applied to the modified mesh. Therefore, if an initial mesh has one element through the thickness everywhere and N_t elements have been requested through the thickness, the splitting and realignment process has to be performed $log_2 N_t$ times rounded off to the next integer. To lift this restriction, the local nature of the diagonal to zigzag conversion process must be sacrificed and propagated out into the mesh.

If a quadrilateral triangulation could not be converted to zigzag due to one of its adjoining wedges becoming invalid, it is revisited later to account for the possibility that other quadrilaterals in the local neighborhood may have been changed

[11] The direction of triangulation of a quadrilateral indicates which end of the diagonals in the triangulation are at a higher level. It is indicated schematically by an arrow on the base edge pointing towards the side of the quadrilateral with the higher end of the diagonal

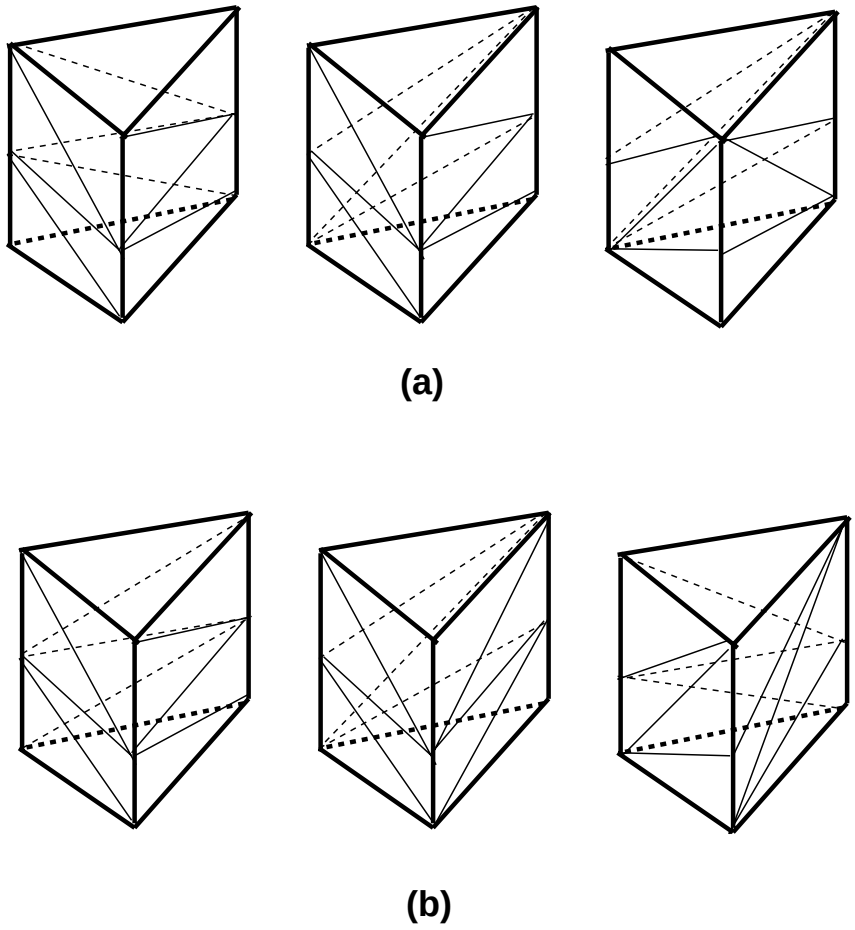


Figure 12.9: Wedge configurations with 2 elements through the thickness. (a) Valid wedge configurations. (b) Invalid wedge configurations.

to zigzag allowing for successful conversion. Still, in their current form, the realignment procedures may be prevented from converting all quadrilateral triangulations to zigzag by the invalidity of certain wedge configurations, even with two edges through the thickness.

12.3.1 Elimination of remaining deficient paths

When the topology of the mesh between thin sections is more general than that described before, it is necessary to use general refinement isotropic techniques to introduce multiple elements through the thickness. Several refinement techniques by means of edge bisection such as Rivara bisection, Bansch's method and alternate bisection are reviewed in [15]. Many of these techniques have been described to be stable in three dimensions (the refined mesh quality is bounded from above or below by the initial mesh quality). Although most of these techniques over-refine due to the propagation of non-conformity, a careful selection of a refinement technique adapted from the above can be used to obtain isotropic refinement with sufficient elements through the thickness. It can be shown that an important component of obtaining a good quality mesh by these refinement methods is the ability to modify the initial surface triangulation.

12.3.2 Creation of multiple layers by local remeshing

While the above procedures to eliminate deficient paths using local mesh modifications do a good job of eliminating the deficient paths in the mesh, they suffer from the following shortcomings:

- They cannot deal well with portions of the mesh with general topology through the thickness, i.e., portions which do not have a wedge or quadrilateral structure.
- Even if wedge topology exists in portions of the mesh, they cannot guarantee conversion of all wedges from a diagonal to zigzag configuration.
- They are less efficient than direct creation of elements knowing the final topology of the mesh.

Of the above, the second shortcoming is very restrictive and directly prevents the procedures from achieving the goal of eliminating all deficient paths through the mesh. Even though the initial and final configurations are valid, it can be shown that a step-by-step conversion process by local mesh modifications cannot convert

all wedges from a diagonal to a zigzag configuration. This can be illustrated by a simple example shown in Figure 12.10. In the figure, the base triangulation for a set of four wedges is shown with the arrows representing the diagonal directions for the wedges. Also, "**D**" or "**Z**" next to the edge indicates that the quad growing on top of the edge is a diagonal or a zigzag configuration respectively. Assume that the triangulations of the four outer quads are constrained. It can be seen from *Rule 2* above (Section 12.3) that all the wedges of the initial configuration are valid. Also, by *Rule 1*, the final configuration is valid. However, to go from the initial to the final configuration, the four interior quads must be made zigzag one after another. However, by *Rule 3*, the configuration will be invalid since the two zigzag quads of the wedge have opposite diagonal directions. Therefore, it can be seen that incremental conversion of the mesh from one configuration to the other is not always possible with the above procedures in the presence of constraints.

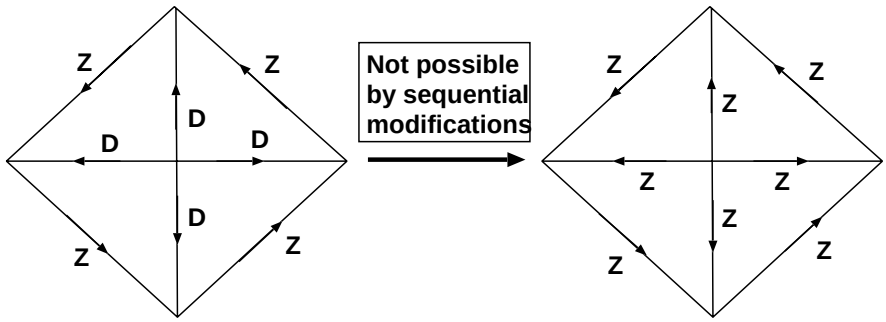


Figure 12.10:]
Illustration that step-by-step modifications of wedge triangulations from diagonal to zigzag is not always possible.

Hence, it is proposed that multiple elements through the thickness be created by deleting the portion of the mesh that is deficient and creating an anisotropic mesh with sufficient number of elements through the thickness in its place. In the following discussion the conditions under which a deficient portion of a mesh can be deleted and replaced with a sufficiently enriched set of wedges is investigated.

Consider a set of edge connected triangles upon which wedges with 1 edge through the thickness and connected to each other along the quadrilateral faces are to be built. It is known that a wedge with 1 edge through the thickness and all its

diagonals in the same direction cannot be tetrahedronized without the introduction of any new points. Therefore, given such a configuration it must be determined if *it is possible to find a combination of directions for the diagonals of the connected set of wedges such that all the wedges have a valid triangulation.*

As before if we think of the wedges in terms of the base triangulations and diagonal directions associated with them, the above question can rephrased as whether *it is possible to find a combination of diagonal directions on the edges of the base triangular mesh representing the connected set of wedges such that all the triangles have a valid combination of diagonal directions.*

The answer to the above question is shown below to depend on:

- whether the direction of the diagonals on the quadrilaterals bounding the set of wedges (referred to henceforth as *bounding quadrilaterals*) is constrained.
- whether the surface triangulation (upon which the wedges are to be constructed) can be altered or not.

If the edges of the triangles are assigned "diagonal directions" arbitrarily, then some of the triangles may end up with an invalid configuration. The methods for correcting these invalid configurations without altering the base triangular mesh are:

1. Flipping the "diagonal direction on an edge of the invalid triangle if does not make an adjacent triangles configuration invalid (Figure 12.11a). Lohner [40] has proposed an iterative scheme using this method but this method is not proved to guarantee correction of all triangles.
2. Propagating the invalid configuration through the mesh until another invalid configuration is reached at which point the diagonal direction for the common edge may be flipped. The flip makes both triangle configurations simultaneously valid (Figure 12.11b,c). The process of propagating an invalid configuration involves successively making a neighboring triangle configuration invalid to make the current one valid.

3. Propagating the invalid configuration through the mesh to the boundary of the set of triangles where the diagonal direction of a bounding edge may be flipped (if permitted) to make the triangle configuration valid (Figure 12.11c).

If the diagonal directions on the bounding edges are not constrained, then it is *always* possible to find a valid combination of directions on the edges of all the triangles under consideration. *In fact, it is possible to prove that only one bounding edge diagonal direction of the connected set of triangles needs to be unconstrained to always get a valid combination of directions everywhere in the connected set.* To see this, consider a set of connected triangles with a certain number of invalid configurations. Since this is a connected set of triangles, there must exist a path connecting any pair of triangles. Every pair of invalid configurations can be propagated towards each other and nullified as described in method 2 above. Therefore, if the set has an even number of invalid configurations, they must nullify each other out. If on the other hand, it has an odd number of invalid configurations, then after nullifying every pair of invalid configurations, one invalid triangle configuration remains. This invalidity can be propagated out to the boundary where it becomes necessary to flip the direction on one of the boundary edges of the triangle. Therefore, only one edge on the boundary of a set of connected triangles must have no constraints on its diagonal direction.

If all the bounding edges of the set of triangles is constrained, then it may not be possible to always obtain a valid set of triangles. This can be easily seen by considering a single triangle in an invalid configuration with all its edges constrained. The only way to obtain a valid configuration in such a case is to refine the surface triangulation in specific ways.

Consider a triangle with an invalid combination of diagonal directions associated with its edges (Figure 12.12a). This situation may be rectified by one of several ways shown in Figure 12.12. In Figure 12.12b, the face is split at the centroid and the new edges assigned diagonal directions appropriately so as to form 3 valid triangle configurations. This form of refinement is not the most desirable since it leads to large face angles on the surface and large dihedral angles in the volume mesh. In Figure 12.12c, the large angles have been bisected by bisecting the original edges of

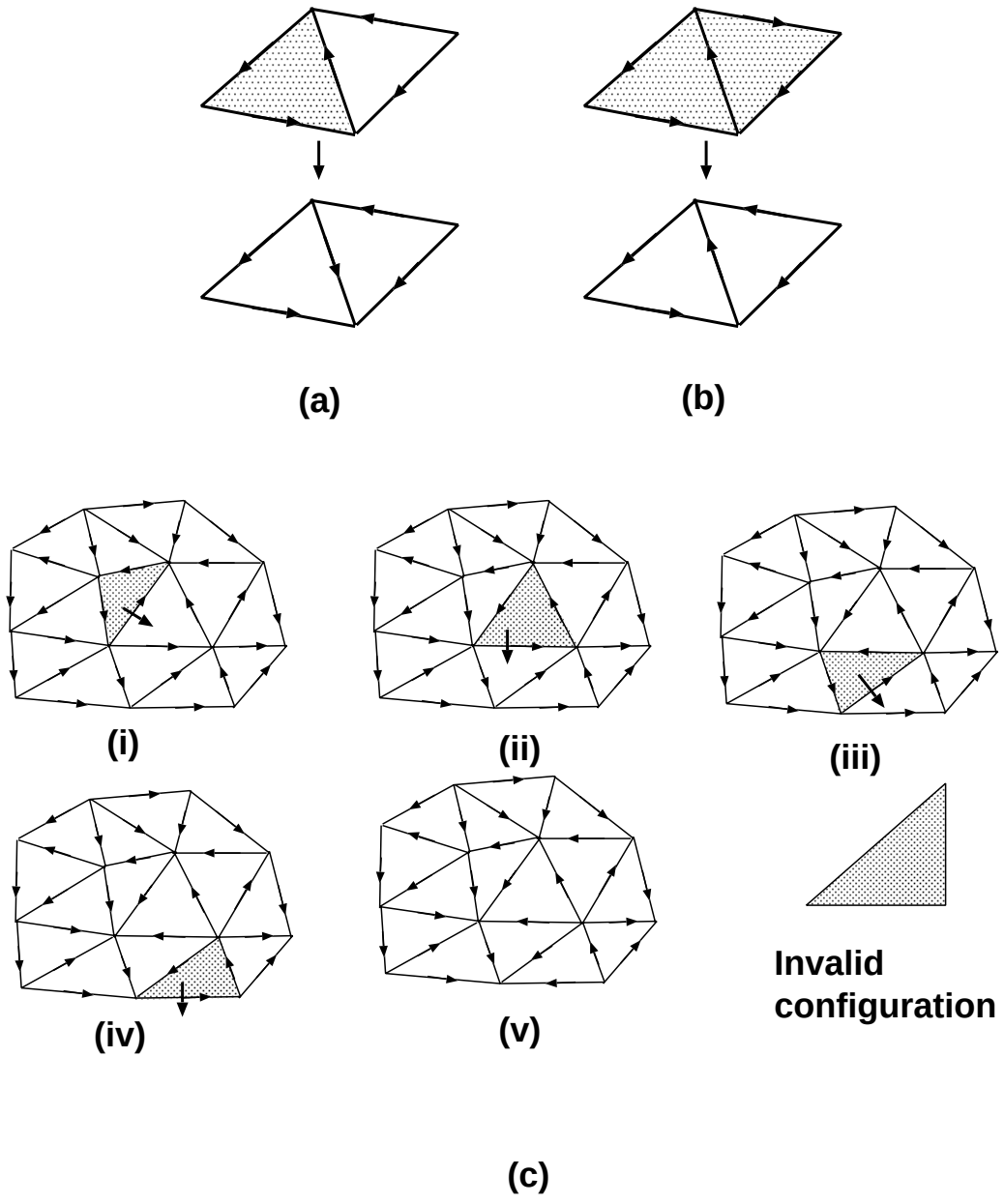


Figure 12.11: Fixing invalid wedge configurations by edge swapping. (a) Fixing one invalid triangle by swapping the diagonal direction on an edge. (b) Fixing a pair of invalid triangles by swapping the diagonal direction of their common edge. (c) Fixing an invalid configuration by propagation of the triangle to the boundary.

the triangle. In Figure 12.12d, only one of the edges is bisected and the direction on one of the split edges is flipped, Note that the last two techniques make adjacent triangles non-conforming (having more than three vertices) which can be fixed by a bisection of the non-conforming triangles. If the adjacent triangle was invalid to begin with, it can be further split in the same way as the first to make it valid. On the other hand, if it was valid to start with, then regardless of the configuration, a direction for the bisection edge can be found such that the resulting two triangles will always be valid. Thus it can be seen that the described method of refinement is always guaranteed to generate a combination of diagonal directions that will all be valid.

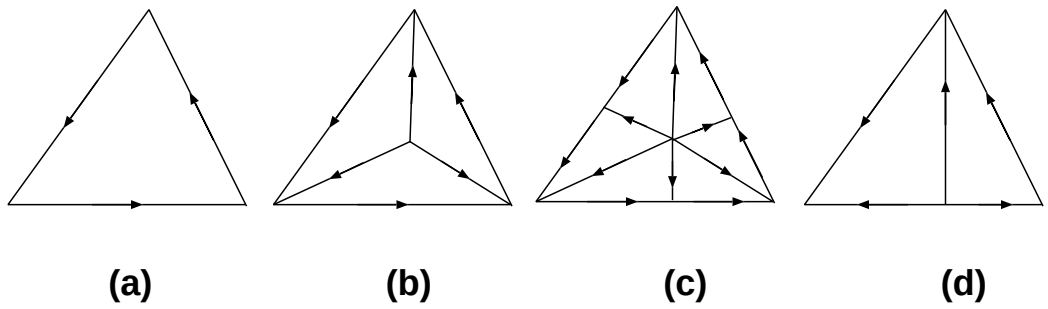


Figure 12.12: Edge bisection patterns to fix an invalid configuration.

If the quality of the surface triangulation is to be preserved, it is not sufficient to terminate the refinement process in the adjacent triangle arbitrarily by bisection. This is because bisecting an edge at the midpoint may result in creation of new poorly shaped elements and the refinement process is no longer stable (the angles are not bounded from below or above by the worst angle in the initial mesh). The process of alternate bisection or longest edge bisection may be applied wherein the refinement is propagated until all the triangles shapes are acceptable. Both methods tend to to over-refine due to propagation of non-conformity and must be used only when all other methods for obtaining a valid set of diagonal directions on the edges have failed.

CHAPTER 13
MULTIPLE ELEMENTS THROUGH THE THICKNESS - PRE- AND POST-PROCESSING

13.1 Pre-processing

A number of pre-processing steps are incorporated into the procedures to generate multiple elements through the thickness. These steps are designed to maximize the number of wedges present through the thickness since subdivision of the wedges provides the desired topology and quality in the final meshes. The pre-processing steps consist of a combination of local mesh modifications and node repositioning. All of these tools are used to match the meshes on locally opposite model faces as closely as possible.

13.1.1 Node repositioning

After the initial search for opposite vertices, nodes on locally opposite model faces are repositioned so that opposite nodes and edge paths between them are more closely aligned with the local thickness direction. This helps reduce the distortion of the wedges and other constructs which are further subdivided into multiple layers. For example, it can be demonstrated that the more the distortion of an isotropic wedge, the greater the largest dihedral angle in the subdivided elements will be.

To align opposite vertices as closely as possible along the local thickness direction, the opposite vertex of a vertex is moved as close as possible on the entity it is classified on. Since the opposite vertex may be classified on a model edge or a face, special procedures are required so that the movement of the vertex is constrained to be on the model entity. Procedures for repositioning vertices on a model edge, on a model face and in a model region are described in Appendix A. The target location is determined by querying the geometric modeler for the closest point on the opposite model entity to the reference vertex. If the closest point search or the movement of the vertex does not succeed, then the reference vertex itself is attempted to be moved. After alignment of the reference vertex and the opposite vertex, any vertices

in the path between the reference vertex and the opposite vertex are moved to be in alignment with the straight line between the reference and opposite vertices.

13.1.2 Matching edges and faces on opposite model faces

13.1.2.1 Mesh matching by edge swapping

Consider an edge classified on a model face. It is possible that the vertices of the edge have respective opposite vertices but an edge does not exist between the opposite vertices. In such a case, due to the absence of quad topology on top of the reference edge (and therefore the absence of wedge topology on top of the two boundary faces connected to it), the quality of elements in the local neighborhood will be poor. In fact, the creation of large dihedral angles is inevitable in such a situation. This is illustrated in Figure 13.1. In the figure, M_0^1 and M_1^1 do not have an opposite edge even though the vertices of the edge M_0^0 and M_1^0 have opposite vertices (M_4^0 and M_5^0 respective). Given that all the neighboring edges and vertices of M_0^0 and M_1^0 have opposite edges and vertices respectively, the mesh configuration can be considered to be a hexahedron which must be broken into tetrahedra. From the individual tetrahedra resulting from this construct, it can be clearly seen that the configuration is prone to large dihedral angles as the height of the hexahedron (or in other words the distance between the opposite vertices) decreases. In particular, the tetrahedron formed by the vertices $\{M_0^0, M_6^0, M_5^0, M_7^0\}$ has a large dihedral angle even when the aspect ratio of the hexahedron is not very large.

The above situation can be greatly improved by swapping the edge between M_6^0 and M_7^0 to form a new edge between the vertices M_4^0 and M_5^0 (Figure 13.2). In this case, the hexahedron can be split into two well shaped wedges which in turn will result in good quality tetrahedra (with respect to large dihedral angles) even if the aspect ratio of the hexahedron decreases.

Therefore, the pre-processing procedures try to match the mesh edges on opposite model faces by edge swapping. If an edge *classified on a model face* does not have an opposite edge but its vertices do, then the two adjacent faces of the edge classified on the same model face are found. The vertices of each of these faces opposite to the edge under consideration are found and then *their* opposite vertices

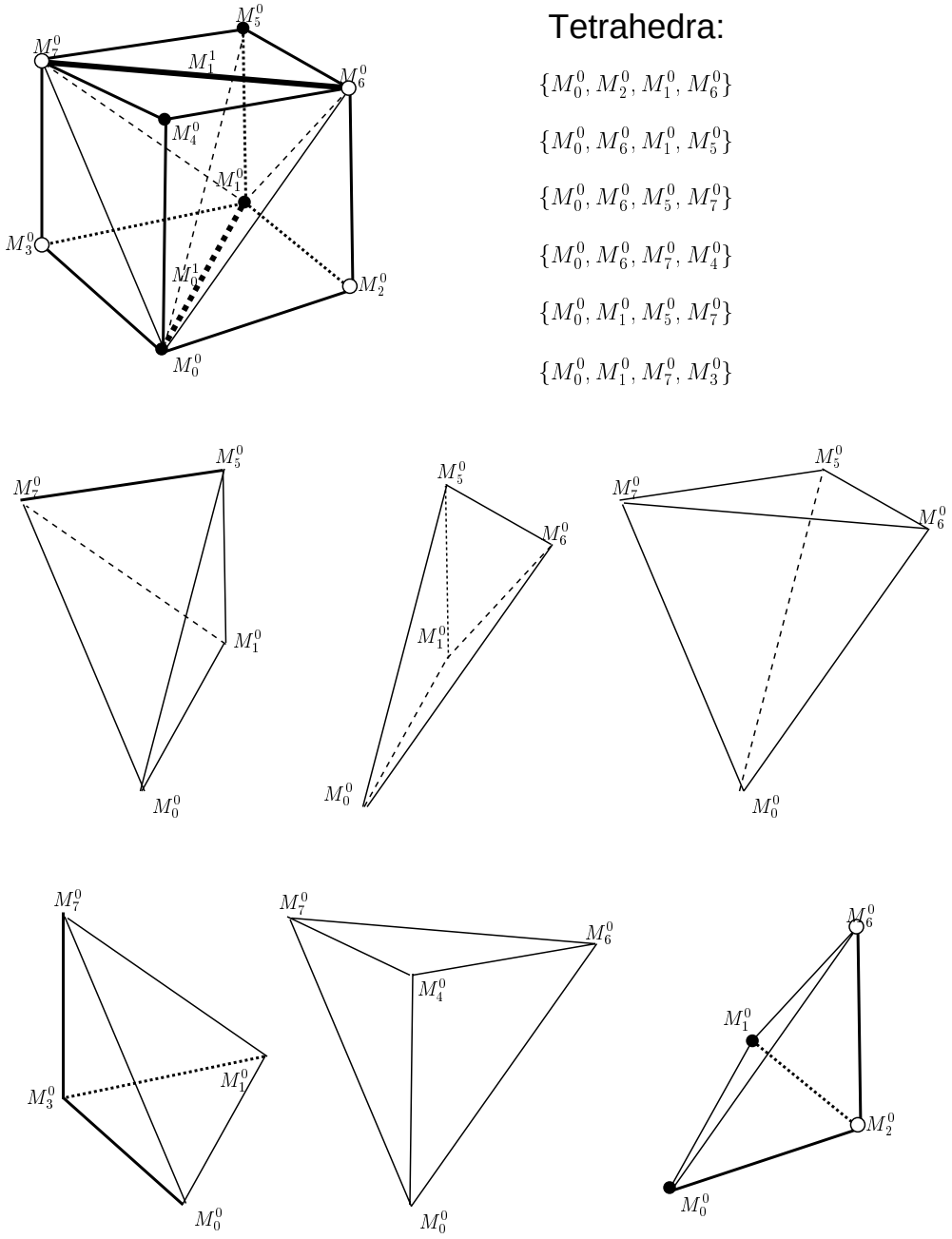


Figure 13.1: Mesh configuration and resulting tetrahedra where one edge does not have an opposite edge. Such mesh configurations are prone to large dihedral angles with decreasing height.

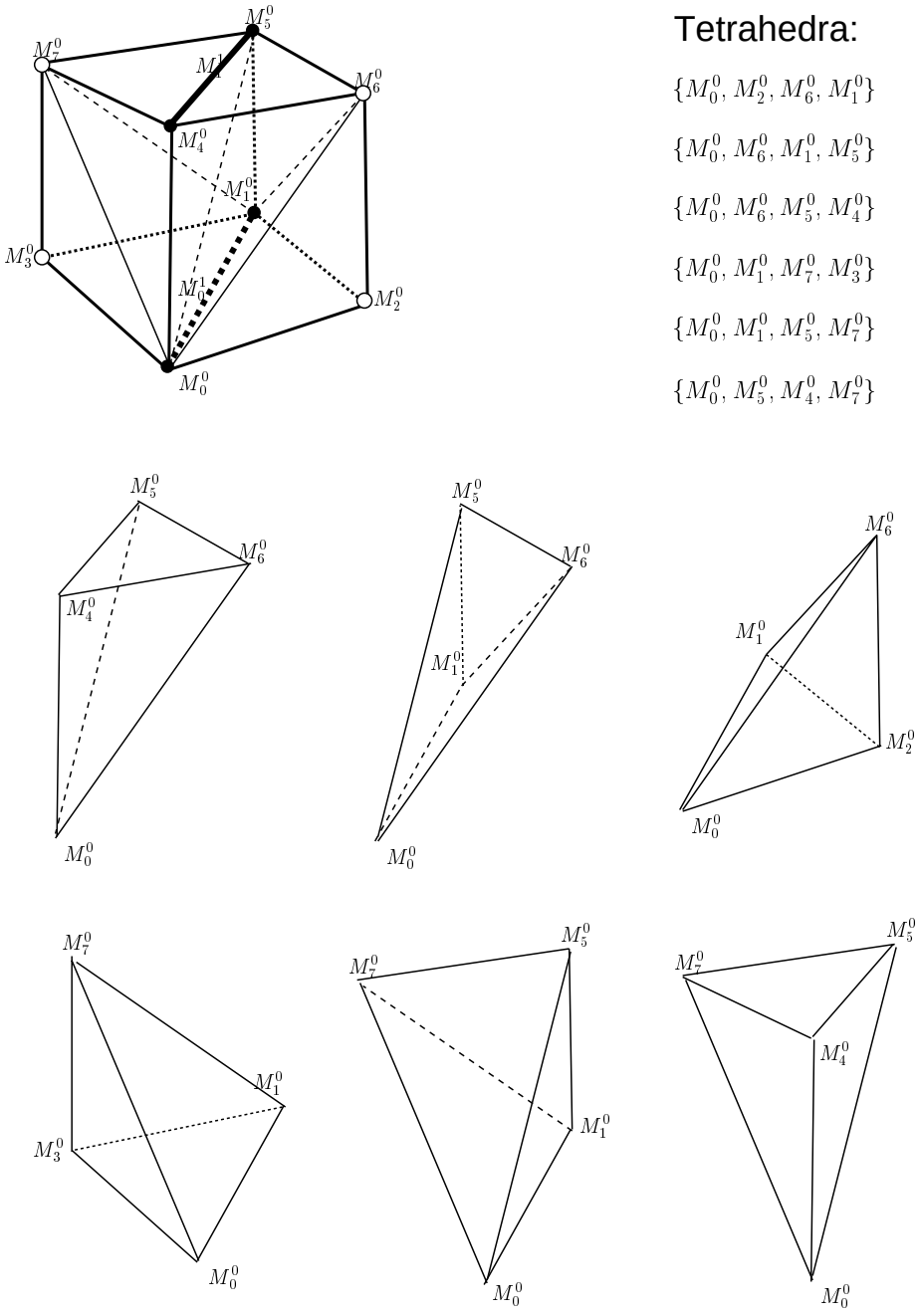


Figure 13.2: Mesh configuration and resulting tetrahedra with matching mesh entities on opposite model faces. Large dihedral angles are well controlled in such configurations even with decreasing height.

are found. If an edge exists between them, then it is attempted to be swapped to create a new edge between the opposite vertices of the original edge. This swapping is subject to standard geometric and topological constraints [58, 66].

13.2 Post-processing

Once multiple elements are generated through the thickness, post-processing of the mesh is performed to match user requirements and improve mesh quality. Node repositioning is used to smooth nodes along edge paths (Also see Appendix A). Although the current procedures attempt to equidistribute the nodes along the edge path, any criterion may be used for this process. For example, if the gradients are stronger near the walls, a smoothing function that clusters the nodes towards the walls may be used instead. Finally, a generalized mesh optimization procedure is applied to the entire mesh to improve mesh quality, in particular to improve large dihedral angles. The procedures once again use a combination of local mesh modification and node repositioning techniques to effect this improvement. During the mesh optimization phase, the newly inserted nodes and the realigned edges by the main procedures to eliminate deficiencies in the mesh are constrained from being affected.

CHAPTER 14
GENERATION OF MULTIPLE ELEMENTS THROUGH THE THICKNESS - RESULTS

In this chapter, results are presented to demonstrate the capabilities and utility of the procedures to generate multiple elements through the thickness.

Figure 14.1a shows the initial solid mesh for a simple rectangular plate for which 4 elements have to be introduced through the thickness. The mesh after mesh matching on opposite faces and splitting edges through the thickness is shown in Figure 14.1b. The mesh after swapping is completed is shown in Figure 14.1c. The largest dihedral angle is 150 degrees in the initial mesh and 96 degrees in the final mesh. The initial mesh has 96 elements while the final mesh has 384 elements.

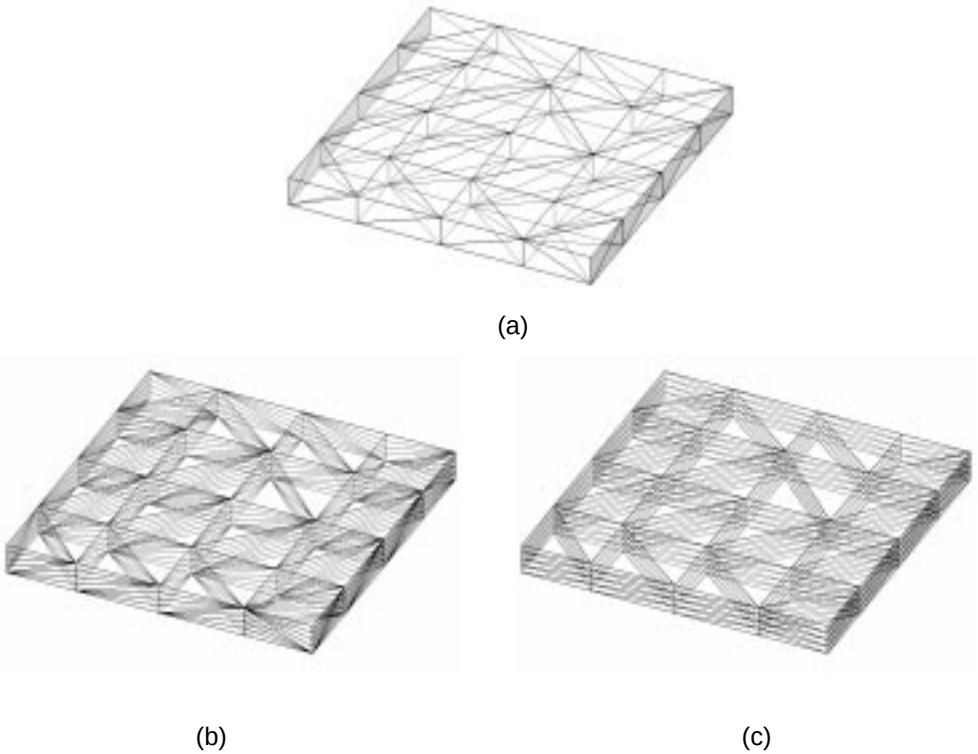


Figure 14.1: Refinement through the thickness for a simple plate. (a) Initial mesh. (b) Mesh after splitting of edges. (c) Mesh after swapping to realign edges. Maximum dihedral angle in final mesh is 96 degrees.

Another illustrative example is shown below in Figure 14.2. The initial mesh is shown in Figure 14.2a and the anisotropically refined mesh is shown in Figure 14.2b. The largest dihedral angle in the initial and final mesh are 144 degrees and 146 degrees respectively.

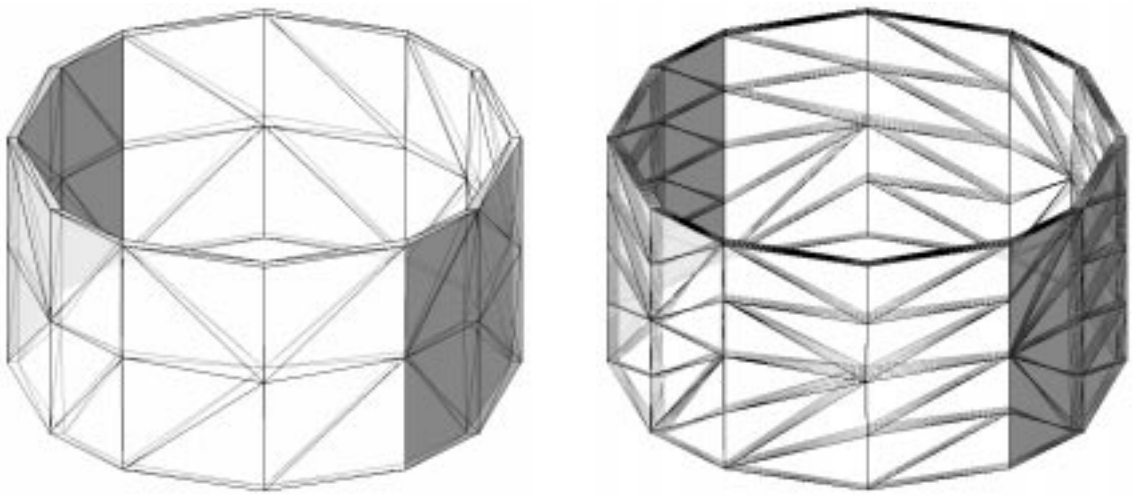


Figure 14.2: Refinement through the thickness of a ring. (a) Initial mesh with 145 elements and largest dihedral angle of 144 degrees. (b) The refined mesh with 1179 elements and largest dihedral angle of 146 degrees.

This demonstrates that when the topology and geometry of the model and mesh permit it, the algorithm generates high quality elements while introducing multiple elements through the thickness. Naturally, geometric models of any practical interest and their meshes do not have this perfect structure throughout and some reduction in quality is expected due to constraints in mesh modification procedures.

Figure 14.3 and Figure 14.4 show the initial and refined meshes with close-up views of two models with general topology and no single thickness direction. It can be seen from the figures, that the procedures have correctly captured the local thickness directions in the various sections of the model. In Figure 14.4 the close-up pictures show a transparent view of the initial and final mesh of the base plate verifying that refinement and the structure of the mesh is as desired even in the interior of the model. 99.98% of the elements in the final mesh in Figure 14.3

have large dihedral angles less than 170 degrees. All elements in the final mesh in Figure 14.4 have large dihedral angles less than 161 degrees.

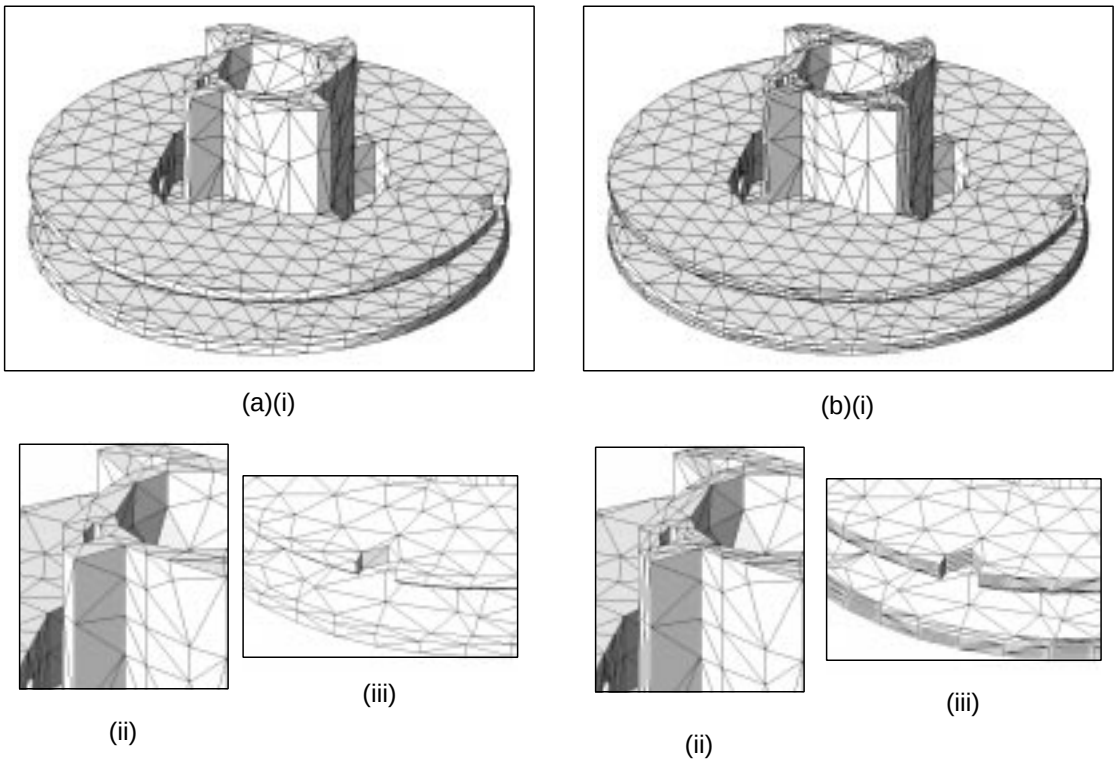


Figure 14.3: Refinement through the thickness for a general model, "asm107". (a)(i) Initial mesh. (ii)(iii) Close-up views of initial mesh. (b) Refined mesh with 4 elements through the thickness. (ii)(ii) Close-up views of refined mesh. 99.98% of elements in final mesh have large dihedral angles less than 170 degrees.

Figure 14.5 shows a simplified airfoil platform in which the thin sections not very clearly demarked and the sections vary in thickness with the result that the initial mesh (Figure 14.5a) has varying number of elements through the thickness along the length of the platform top. Figure 14.5b shows the refined mesh with a close-up view shown in Figure 14.5b. From the figures it can be seen that the procedures have identified the deficient parts of the mesh well and refined correctly through the thickness. For example the smaller "leg" of the platform initially had two elements through the thickness and two more were added to it. On the other hand the top of the platform is of varying thickness and the initial mesh had one elements in some parts and two in others. The procedures correctly recognize this and

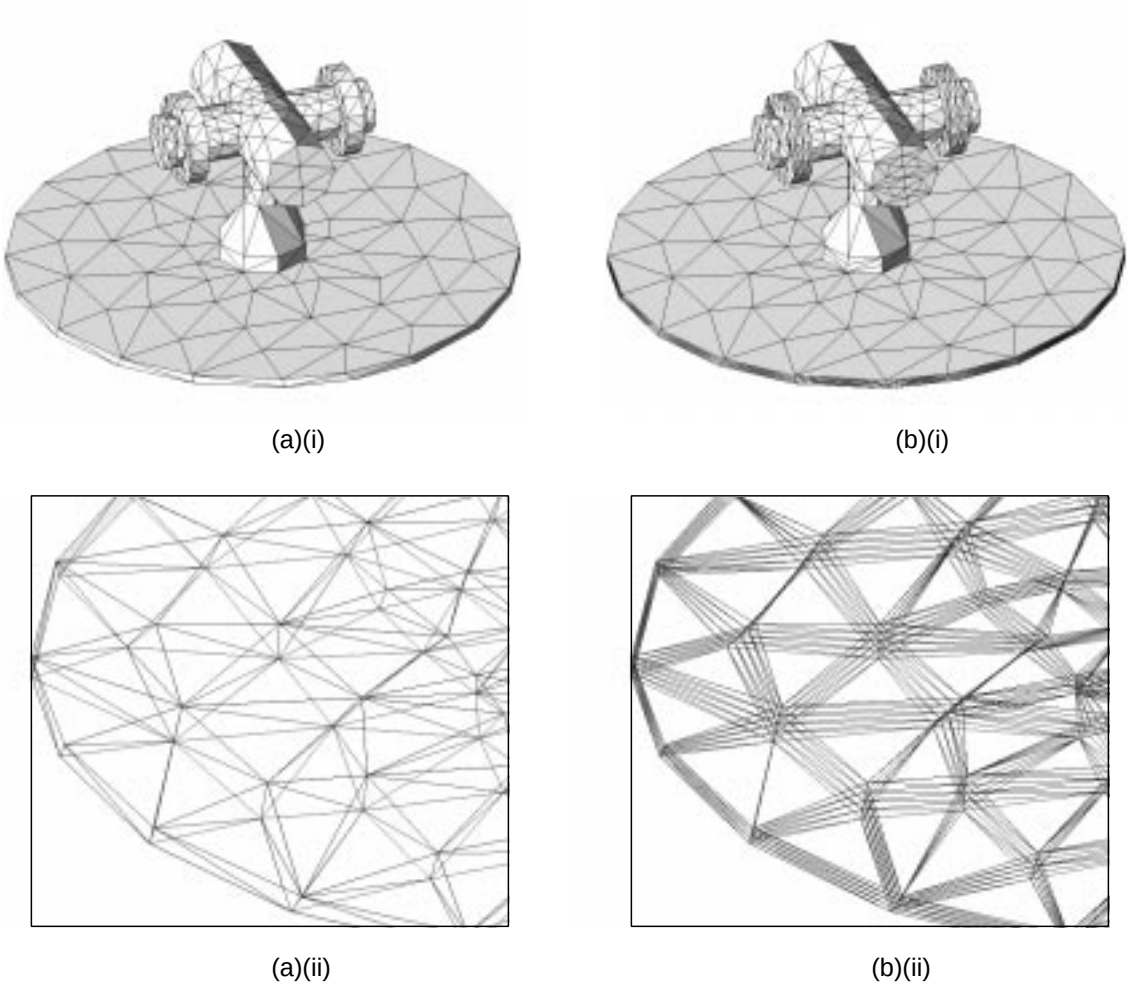


Figure 14.4: Refinement through the thickness for a general model, "asm110". (a)(i) Initial mesh. (ii) Transparent close-up view of initial mesh of base plate. (b)(i) Refined mesh with four elements through the thickness. (ii) Transparent close-up view of refinement in base plate. 100% of elements in final mesh have large dihedral angles less than 161 degrees.

refinement has been performed so that there are 4 elements through the thickness throughout. The example also illustrates that not only the conversion of diagonal quads to zigzag works but that the procedure is able to identify the other types of configurations and realign their edges as well.

The various capabilities and features of the procedures to introduce multiple elements through the thickness are also demonstrated in the following example which is a simplified setup for investment casting of an airfoil (Figure 14.6)

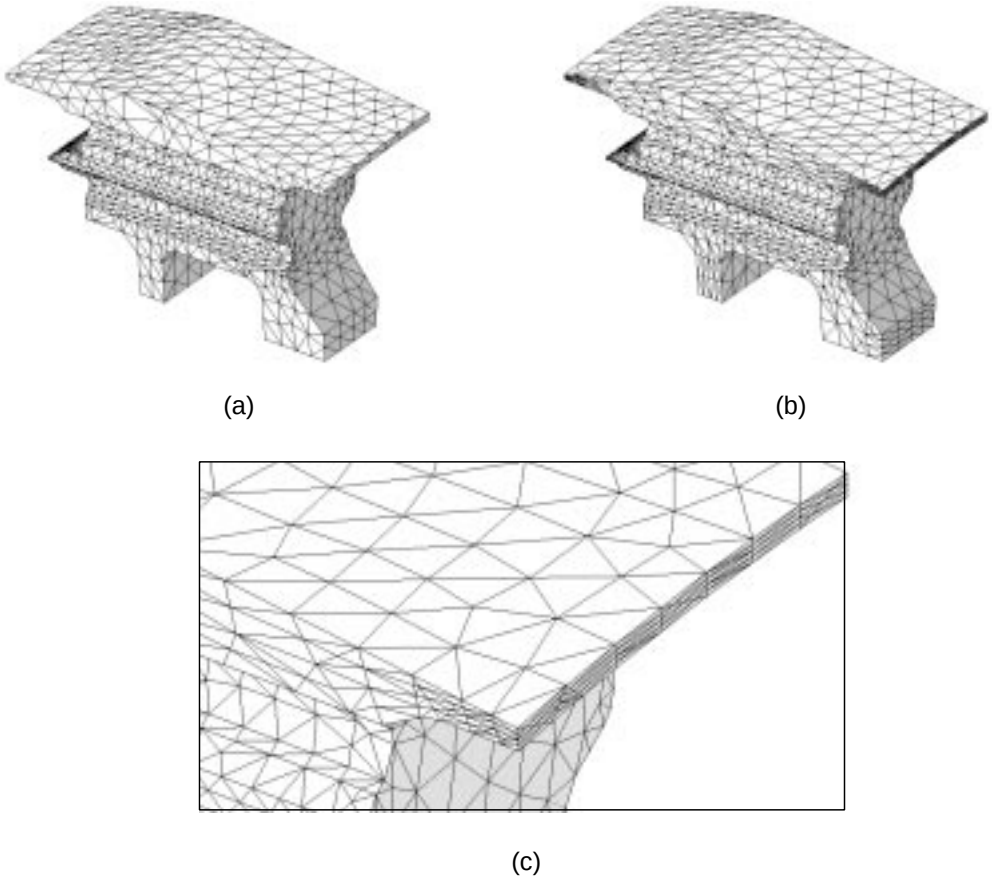


Figure 14.5: Refinement through the thickness for airfoil platform. (a) Initial mesh. (b) Refined mesh with 4 elements through the thickness. (c) Close-up view of refined mesh in one of the thin sections.

Finally, the results[12] of a transient heat conduction analysis with radiative heat transfer in a crucible for crystal growth simulation using a mesh refined by this method are shown in Figure 14.7. The schematic model is shown in Figure 14.7a while shows the mesh with 4 elements introduced through the thickness. The mesh has 32,221 elements compared to an equivalent isotropic mesh, i.e., uniformly refined to have 4 elements through the thickness, which has 317,841 elements, a reduction of an order of magnitude. The temperature distribution near the top of the crucible and through a vertical section (expected to be exponential) are shown in Figures 14.7b,c.

[12]Courtesy: Hongwei Li, formerly at SCOREC, RPI

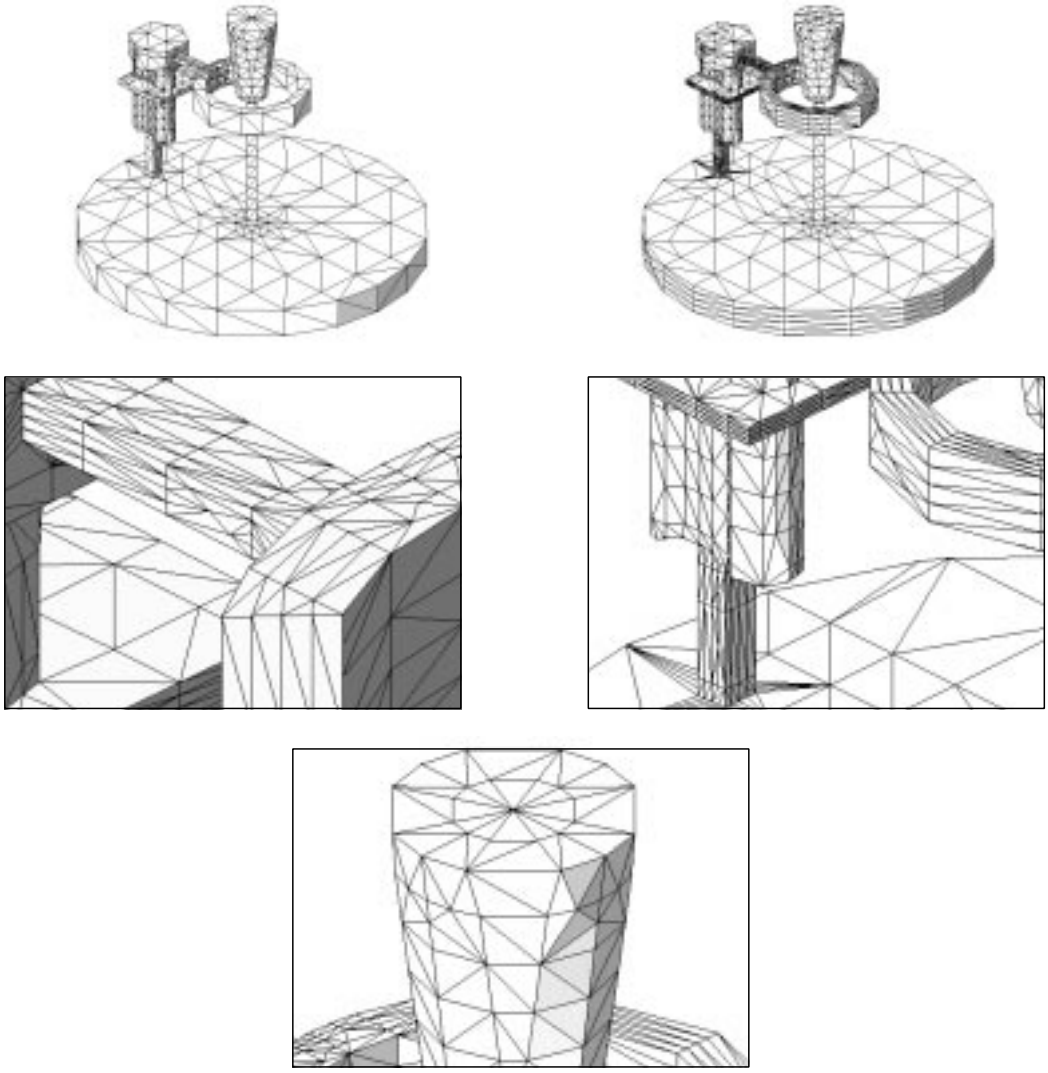


Figure 14.6: Refinement through the thickness for model representing the setup for investment casting of an airfoil.

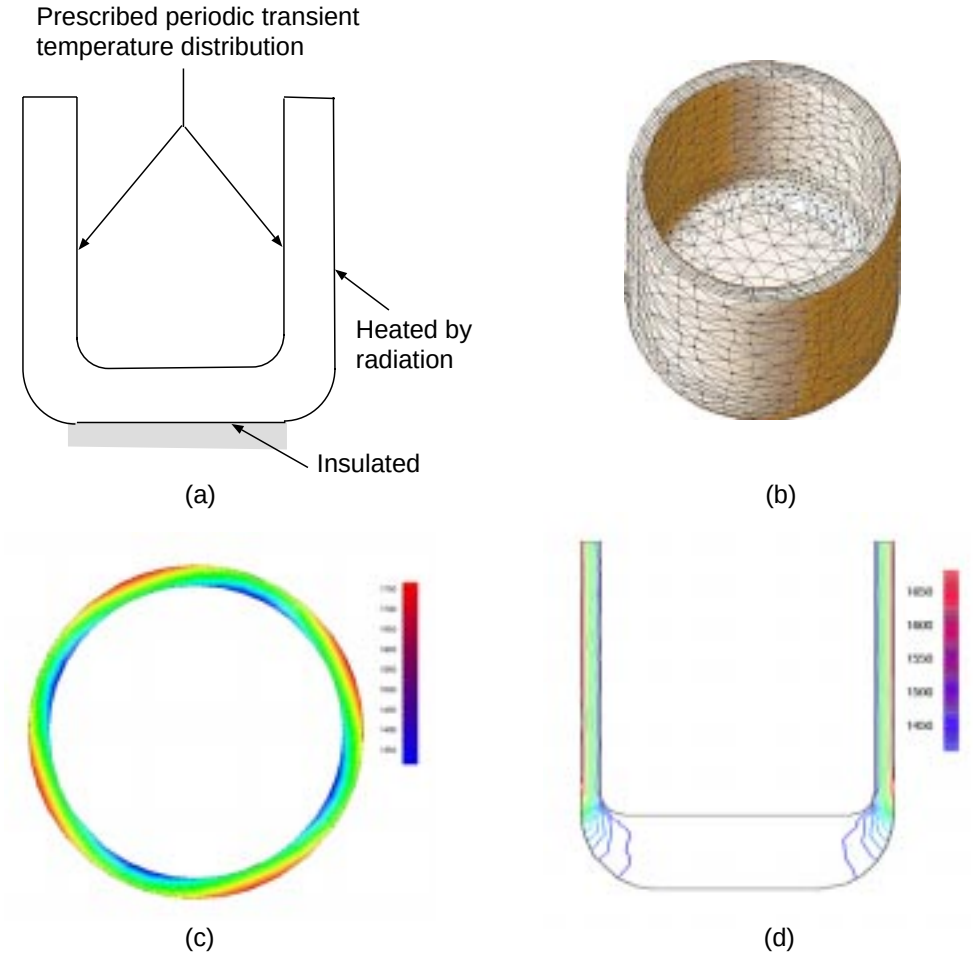


Figure 14.7: Transient heat conduction analysis with radiative heat transfer in crystal growth crucible using a mesh with 4 elements introduced through the thickness. (a) Schematic of problem. (b) Mesh with 4 layers through the thickness. (c) Temperature distribution near the top. (d) Temperature distribution through a vertical section.

CHAPTER 15
CLOSING REMARKS AND FUTURE WORK

15.1 Concluding Remarks

Two procedures for generation of anisotropic tetrahedral meshes were presented. The first is a procedure for generating boundary layer meshes for viscous flow simulations. The method, called the *Generalized Advancing Layers Method* is designed for reliable generation of valid, good quality meshes for arbitrarily complex non-manifold geometric model. It includes several technical advances to be able to handle complex domains that cannot be handled by other techniques. It also provides control and flexibility in the creation of meshes suitable for fluid flow simulations.

The technical contributions of the Generalized Advancing Layers Method are:

1. Ability to create valid meshes for general non-manifold models through the definition and use of mesh manifolds (Section 3.2.2 and Section 5.4).

2. Complete approach to controlling mesh quality and gradations in the boundary layer through the use of multiple growth curves (Section 5.5), multi-level transition elements (Section 7.8) and fixed and variable edge blends, (Section 7.9). Also, included in this approach are smoothing, shrinking and pruning for boundary layer mesh quality improvement and recursive adjustment of neighboring growth curves for improved mesh gradation (Sections 6.3, 6.4 and 6.5 respectively).

3. Comprehensive approach to shield the isotropic mesher from anisotropic faces using transition elements, blends and pruning of growth curves.

4. Definition and implementation of well defined set of checks for the creation of topologically and geometrically valid meshes particularly in the context of modification of the initial surface mesh to account for boundary layers (Section 5.6).

5. Several technical developments for the robustness of the procedure including development of new procedures for retriangulation of badly parameterized model faces based on recovery of edges in a surface mesh using local mesh modifications (Section 7.6 and [35]), alternate procedures to compensate for inability to modify the surface, assurance algorithms for prism validity (Section 7.7), and assurance algorithms to resolve self-intersections of boundary layers (Section 8.3).

6. Development of powerful but flexible methods for boundary layer specification and control (Section 5.8).

Results were presented to demonstrate the capability of the mesh generator to mesh complex non-manifold models while creating meshes with element sizes and gradations required to accurately capture the solution. Results of two classical problems in fluid mechanics were presented to demonstrate the suitability of the mesh for viscous flow simulations and its ability to capture the solution accurately. Also, the application of the mesh generator to create meshes suitable for simulations with free shear layers was demonstrated. The Generalized Advancing Layers Method has succesfully generated meshes of the order of 3-4 million elements for other large complex geometric models and is currently being used for simulations on real automobile configurations in industry.

The second capability developed creates anisotropic meshes in models with thin sections using local mesh modifications. The procedures are designed to work in conjunction with any automatic isotropic mesh generator capable of providing an initial mesh. The method to create multiple elements through the thickness automatically identifies portions of the mesh that have less than the requested number of elements through the thickness and anisotropically refines those parts of the mesh using edge splits and swaps. The refinement algorithm can handle arbitrarily complex non-manifold models reliably.

Results were presented to demonstrate the capabilities of the procedures to create multiple elements through the thickness for various complex geometric models. Also, results of a heat transfer analysis were given to demonstrate the ability of the mesh to capture non-linear gradients through the thickness and to demonstrate

the savings in the number of elements that can be achieved using this mesh generator. The procedures were seen to identify deficiencies in the initial isotropic meshes well and refine them while controlling mesh quality.

The key technical contributions of the research on generation of tetrahedral meshes with multiple elements through the thickness are:

1. Automatic identification of thin sections in an initial mesh with insufficient number of elements through the thickness (Section 11.1).

2. Creation of multiple elements through thin sections by local mesh modification procedures followed by template based edge swapping (Section 12.1 and Section 12.2).

3. Qualification of constraints in wedge triangulations and techniques to overcome these constraints in the generation of multiple elements through thin section models (Section 12.3).

Results were presented to demonstrate the ability of the anisotropic refinement procedure of thin sections to handle complex domains and properly introduce the necessary number of elements through the thickness. This mesh generator has proven to be of considerable practical importance and has been used succesfully to generate meshes for a wide range of applications including semiconductor device modeling, casting, injection molding, modeling of MEMS, electromagnetics, biomechanics, heat transfer analysis, coupled fluid-thermal simulations in intercoolers, chemical corrosion and structural analysis.

Both procedures presented here are being used within an overall framework for reliable automatic mesh generation of complex geometric domains for finite element simulations in a wide variety of engineering applications [62].

15.2 Future Work

There are number of ways the research presented in here can be further developed. Some of the necessary and desirable developments to boundary layer mesh generator to create better meshes for viscous flow simulations are:

1. Ability to create prism elements in the boundary layer.

2. Implementation of blend elements: This a key feature of the procedures necessary to shield the isotropic mesh generator from the anisotropic faces of the boundary layer mesh and is critical to the overall robustness of the meshing framework. In addition, the introduction of general blends with multiple growth curves is necessary for smooth mesh gradations.

3. Capability to generate boundary layer meshes with matching meshes on faces with periodic boundary conditions: This capability is of great practical importance as it can result in large savings in mesh sizes by taking advantage of symmetries in the solution. The central issue here is the matching of prism and blend diagonals on boundaries to be matched. As the reader may recall from discussion of boundary layer prism creation and prism templates in the generation of multiple elements through the thickness, there are constraints on how individual prisms and a set of connected prisms may be triangulated without refinement of the surface triangulation. These constraints must be respected and if necessary, a surface mesh refinement strategy devised to be able to match the boundary layer diagonals on two model faces.

4. Separation of growth curve from model boundaries: Recall that in this implementation, growth curves are constrained to be interior or boundary and only a specific type of partially boundary quad is dealt with. The ability to handle growth curves and quads with general classification is important to mesh quality.

5. General curvilinear shape for interior growth curves: This will allow better control over mesh quality.

6. Ability to partially unite or coalesce growth curves: While the current procedures allow the number of nodes to vary along a model face, they do not allow joining of two neighboring growth curves. This is an important capability that will allow the mesh generator to decrease the number of nodes and thereby automatically increase mesh size as the mesh proceeds towards the interior.

The key issue to be addressed for introducing this capability is the redefinition of adjacent growth curves.

7. Capability to adapt shear layers without redefining the geometric model: The current procedures require a model surface definition to be able to grow a boundary layer forcing the introduction of an artificial surface in the geometric model to represent shear layers or wake surfaces behind bluff bodies. It is more desirable to adapt the shear layer mesh independent of the shear or wake surface definition.

8. Parallel boundary layer mesh generation: The generation of large meshes for turbulent flow simulations, particularly Large Eddy Simulations, makes it necessary to parallelize the creation of boundary layer meshes and raises a number of open issues.

Future work in the development of the research on tetrahedral mesh generation with multiple elements through the thickness must primarily address the direct creation of layers of prisms between matching sets of mesh faces on opposite model faces. As per the discussion in Section 12.3, a surface refinement strategy must be incorporated into the procedures that will introduce the least number of points while maintaining mesh quality. In addition, general refinement procedures to eliminating remaining deficient paths must also be incorporated.

In conclusion, two robust automatic procedures for the generation of anisotropic meshes for complex non-manifold geometric models were presented as components in an overall framework for anisotropic mesh generation. The two works of research described addressed many key issues in the reliable generation of good quality anisotropic meshes for specific classes of problems. They were demonstrated to reliably generate quality meshes with suitable mesh sizing and gradation for capturing the solution in various problems accurately.

APPENDIX A
LOCAL MESH MODIFICATIONS AND NODE REPOSITIONING

In this chapter the local mesh modification and node repositioning procedures used in the previously described work are described. Local mesh modification of tetrahedral meshes consist of three basic operations [15, 18]:

- Edge, face or region split - introduces one new vertex into the mesh.
- Edge swap - Does not change the number of nodes in the mesh.
- Edge collapse - Deletes one node from the mesh.

Complex transformations of tetrahedral meshes can be effected by application of one or more of these procedures. The local mesh modification procedures always maintain a valid topological connectivity of the mesh. However, additional procedures are required to maintain topological validity of the mesh with the model and geometric validity of the elements.

A.1 Edge Split

An edge split operation breaks an edge into two edges and also splits each of the connected higher order entities into two entities. The edge split for surface meshes consists of the following steps (Figure A.1):

- Create a new vertex at the split location. This vertex inherits the classification of the split edge.
- Create two new edges between the new vertex and vertices of the split edge. The new edges inherit the classification of the new edge.
- Split each face connected to the original edge with an edge between new vertex to the face vertex opposite the original edge. The faces inherit the classification of the respective original faces.

For volume meshes one additional step follows the steps in two dimensions. The regions connected to the original edge are divided into two by introducing a face between the new vertex and the two vertices of the region opposite the original edge. This is illustrated in Figure A.2.

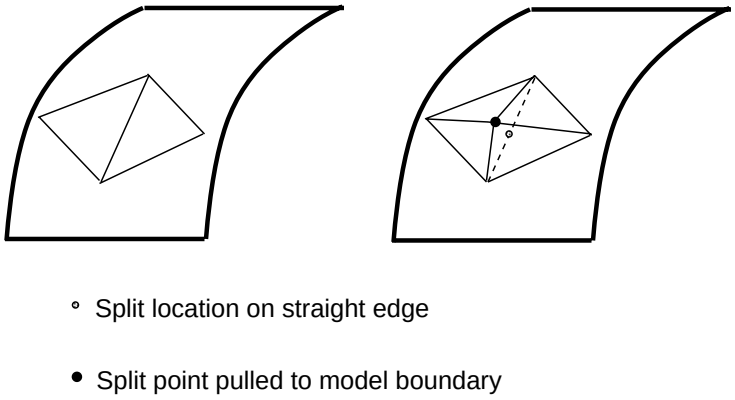


Figure A.1: Edge split on surface meshes.

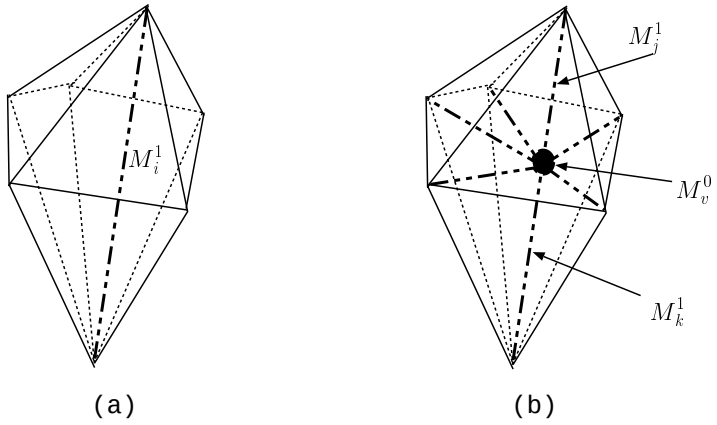


Figure A.2: Edge split in volume meshes.

If the edge being split is a boundary edge, then the split point must be located on the boundary. In some situations, this may make the elements invalid according to some measure (See Section A.7). The split operation cannot create any topological incompatibility of the mesh with the model.

A.2 Face Split

A face split divides a face into three new faces. Additionally, for volume meshes, it divides each region into three new regions. To perform a face split, a new vertex is created inside the face. Three new faces are created by connecting the new vertex to two vertices of the original face in turn. If the face has tetrahedra connected to it, each of the new faces is combined with the fourth vertex of the tetrahedron to form a new region (Figure A.3a). As with an edge split, the face split operation cannot in itself produce any topological incompatibility of the mesh with the model. Also, if the face is a boundary face, the newly created point must be relocated on the model boundary and the validity of the element must be checked with respect to that location.

A.3 Region Split

A region split divides a region into four new regions. The new regions are formed by each of the faces of the original element and the newly created vertex. The newly created vertex can be classified only on the interior. This operation is not used very commonly (Figure A.3b).

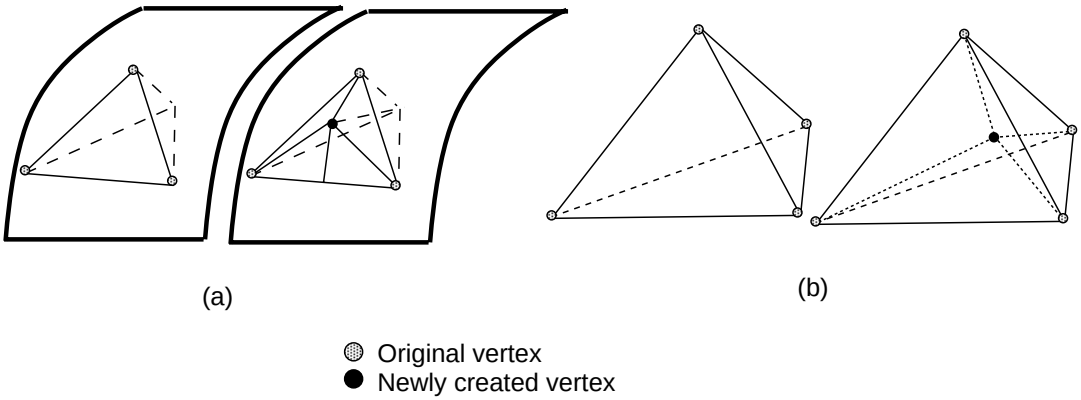


Figure A.3: (a) Face split on model boundary. (b) Region split.

A.4 Edge Swap

The edge swap is a reconnection procedure that effectively deletes an edge and its connected elements and retriangulates the polygon or polyhedron without the deleted edge. For triangular meshes, the swapping procedure consists of deleting the edge and its two connected faces, and reconnecting the quadrilateral so formed with an edge between the opposite face vertices of the deleted edge (Figure A.4). The process is more involved for volume meshes and consists of the following steps in general (Figure A.5):

1. Delete the regions connected to the edge.
2. Delete the faces connected to the edge.
3. Delete the edge. At this point we have a polyhedral cavity with the two vertices of the deleted edge opposite to each other (not connected by an edge).
4. Create any boundary faces necessary if the swapped edge is a boundary edge.
5. Find the set of edges that are not connected to the vertices of deleted edge. These edges form the boundary of a closed polygon.
6. Triangulate this polygon. Since the polygon does not contain the vertices of the original edge, the original edge cannot be recreated.
7. Connect each face of the polygon to each vertex of the deleted edge to form a region.

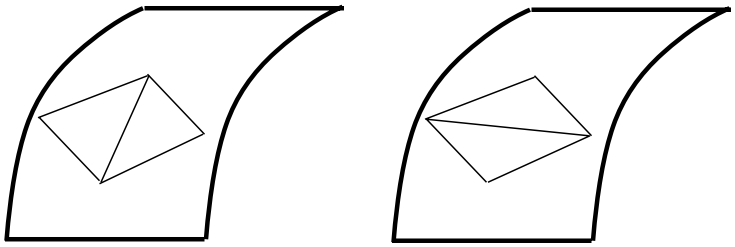


Figure A.4: Edge swap for surface meshes.

If there are n connected regions around an interior edge, then a n vertex polygon (excluding the edge vertices) is formed by deletion of these regions. This polygon can be triangulated in N_n ways, $N_n = \sum_{i=3}^{n} N_{i-1}N_{n+2-i}$ with $N_2 = 2$ [18]. Each triangulation has $n-2$ triangles and therefore, swapping this edge produces $2(n-2)$ tetrahedra. If the edge is classified on a 2-manifold model face (Figure A.6a), then there is only one configuration for the new boundary edge. This new edge is on the boundary of the retriangulation polygon. The number of vertices in the polygon is $n+1$ where n is the number of regions deleted to form the polyhedral cavity. The number of triangles formed in the polygon are $n-1$ and therefore the swapped configuration has $2(n-1)$ tetrahedra.

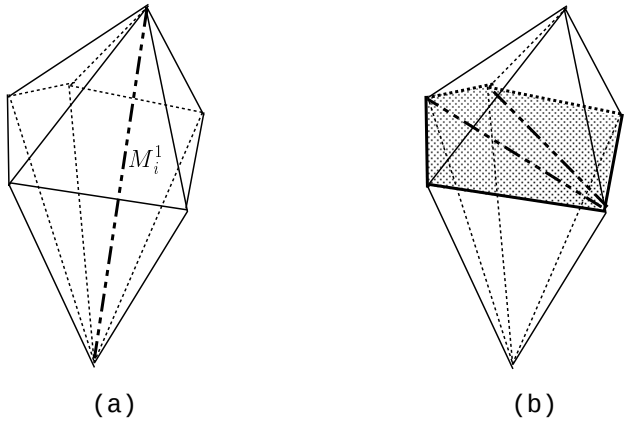


Figure A.5: Edge swap in the interior of a volume mesh.

Swapping an edge on a non-manifold face, on the other hand, requires a more careful look. Since the edge is on a non-manifold face, the new boundary edge can be created only between two vertices classified on the closure of the face (Figure A.6b). Also, because the swapped edge had a connected set of regions completely surrounding it, the polygon that needs to be retriangulated has n vertices where n is the number of regions connected to the edge. Therefore, the n vertex polygon is divided into two polygons with $n_1 + 1$ and $n_2 + 1$ vertices respectively where n_1 and n_2 are the number of regions connected to the edge on the two sides of the non-manifold model face. The number of regions formed is still $2(n-2)$ but the number of topologically possible triangulations is reduced.

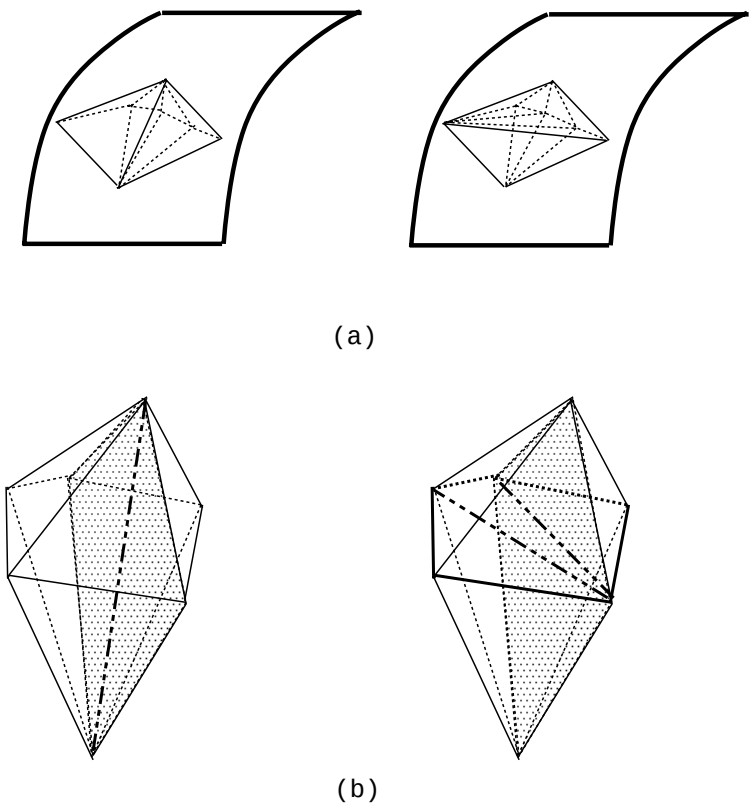


Figure A.6: Edge swap on boundary of volume mesh. (a) Edge swap on 2-manifold model face. (b) Edge swap on non-manifold boundary face.

Not all of the different triangulations possible topologically in an edge swap operation may be geometrically valid. Therefore, each triangulation must evaluated to ensure that all the created elements will be valid (See Section A.7).

The topological constraints in an edge swap are as follows:

1. An edge classified on a model edge may not be swapped since this will cause the mesh to violate topological compatibility with the model.

2. An edge classified on a model face may be swapped with the restriction that the quadrilateral cavity formed on the boundary is retriangulated in the only other way possible.

A.5 Edge Collapse

Edge Collapsing is the process of deleting a vertex from the mesh while keeping the mesh geometrically and topologically valid. Conceptually, edge collapsing can be thought of as the process of deleting all the elements connected to the vertex to be removed and retriangulating the resulting polygon or polyhedron. In actual implementations, it is more efficient to carry out a collapse by the following steps (Figure A.7):

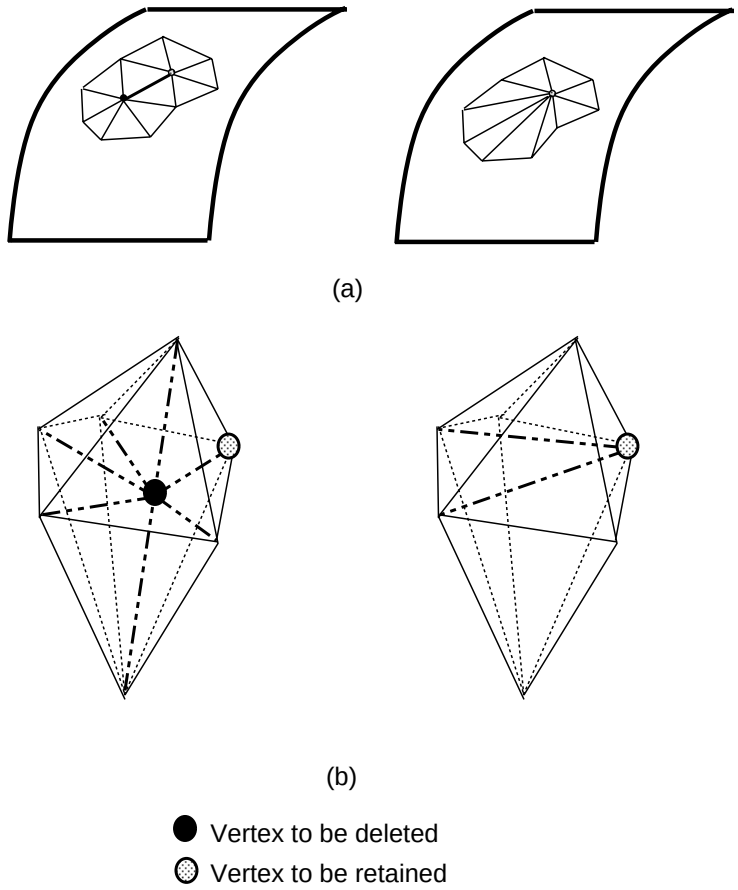


Figure A.7: Edge collapse. (a) Collapse on surface mesh. (b) Collapse in volume mesh.

1. Delete the regions around the edge to be collapsed including the edge itself.

2. Merge the vertex to be removed with the vertex to be retained.

3. Merge the entities of the polygon or polyhedron connected to the vertex to be removed with the entities of the connected to the vertex to be retained.

Since the shape of the elements connected to the vertex to be removed changes after the collapse, they must be checked for geometric validity.

The topological restrictions on collapsing are the most stringent of all local mesh modification operations since they have the potential to cause topological incompatibility of the mesh with the model and also cause dimensional reduction of the mesh ([19]). The conditions under which an edge can be collapsed are as follows:

1. If the two vertices of the edge are classified on equal order entities then the two entities must be the same and the edge must be classified on the same entity.

2. If the two vertices are classified on different order model entities,

   (a) The vertex to be removed must be classified on a higher order model entity than the vertex to be removed.

   (b) The edge must be classified on the higher order entity.

3. If the two vertices of the edge to be collapsed are connected to two edges sharing a third vertex, then the three vertices must bound a face classified on the same entity as the edge to be collapsed. If this condition is not satisfied, there will be coincident edges in the mesh after the collapse.

4. (For volume meshes only) If the two vertices of the edge to be collapsed are connected to two faces sharing a common edge, then the two edge vertices and the common edge must bound a region. If this condition is not satisfied, there will be coincident faces in the mesh after the collapse. In addition, both faces should not be classified on model faces or else their collapse will cause a dimensional reduction.

A.6 Node Repositioning

Node repositioning is commonly used to improve element quality and mesh gradation in the mesh [20, 21, 36]. The node repositioning criteria used in this thesis

are improvement of mesh gradations by weighted Laplacian smoothing and equidistribution of nodes through the thickness. The discussion of node repositioning here focuses on the considerations in repositioning of a node from the current location to a target location particularly on model boundaries.

Reposition a node classified in the interior of a model is a straightforward process. The node is attempted to be moved from its current location to the target location subject to constraints on element validity or quality. If these constraints are violated, then the node is attempted to be moved to the midpoint of the line joining the current and target locations. This process of bisection continues until a valid target location for the node is found with a limit on the number of bisections (typically 3 to 5).

Repositioning of nodes on model faces is done differently for model faces with a continuous and discontinuous (in particular, periodic) parametric spaces. If the initial move on model face with a continuous parametric space fails, the process of bisecting the line segment between the current and original target locations is done in the parametric space. The midpoint of the current and original target parameter locations is picked iteratively as the next target location. Given the target parametric location, the location of the node on the surface in real space is computed and the local mesh checked for validity. On the other hand, if the face is periodic, the line segment joining the current and target locations may cross the periodic jump in the parametric space. Using an average of the two parameter values to compute a new target location gives erroneous results and results in the point being pulled to a location diametrically opposite to the desired location in real space. To account for this, the points on the face corresponding to the average parameter and the average parameter added with half the parametric range are computed. Of the two locations, the one closest to the current location is chosen. Note that this is equivalent to doing a closest point search on the model face but is considerably more efficient. *The underlying assumption of this strategy is that no edge in the mesh spans more than half the parametric space of the model face.*

Repositioning of nodes on model edges is similar to the repositioning on model faces except that only one parameter needs to be dealt with.

A.7 Element Validity

The geometric validity of elements in a volume mesh can be easily checked by checking if all the elements in the mesh have positive volume. However, an equivalent check is harder to define for a surface mesh. While it is simple to check whether a triangle has zero area or not (if necessary within some tolerance) checking whether the triangle has "positive" or "negative" area is poorly defined for general three-dimensional surfaces. Schroeder and Shephard [58, 66] defined rigorous conditions for the validity of meshes and in particular imposed the condition that the mesh should be geometrically similar to the model with reference to an appropriately defined parametric space. This means that in that chosen parametric space no elements can overlap each other. However, how to choose an appropriate parametric space such that a mesh that is geometrically similar to the model in that space results in an acceptable mesh in the real space is still an open question.

Violation of geometric similarity of the mesh of a curved surface results in a large difference between the "smoothness" of the discretization of surface relative to the local smoothness of the surface itself. This discrepancy in the "smoothness" of the discretization may be approximated in a number of ways, some of which are listed below:

1. Measure the difference between the mesh face normal and the model face normal sampled at a suitable point within the parametric boundaries of the mesh face. This is an error prone check, particularly for coarse discretizations of highly curved surfaces, since the model and mesh face normals might differ significantly.

2. Construct a local parametric space by projecting all the faces in the local neighborhood onto a plane. The projection plane may be defined by the model face normal or by an appropriate average of the mesh face normals. The projected triangles can be checked for inside-out condition or negative area with respect to this plane. This method is sensitive to the choice of the projection plane and it is very easy to perceive a triangle as having "negative" area due to sharp changes in the mesh face normals.

3. Measure the dihedral angles between mesh faces to determine if the surface discretization is folded thereby causing a large "change" in the mesh face normals when the surface normals itself are not changing that dramatically. The advantage of this measure over the first method, is that it does not rely on comparison of the mesh and model face normals avoiding some of the problems with coarse meshes. Its advantage over the second method is that the extent of the approximation and therefore the possibility of an undesirable decision is much lesser since only two mesh faces are involved at any time in the check. The dihedral angle check for surface "smoothness" involves setting a tolerance for the allowable angles and is dependent on how strictly one wants to control the mesh generation or modification procedures.

In the context of mesh modification procedures, a meaningful alternative to checking the absolute validity of the surface mesh is to check if the modified mesh deviates considerably from the original. This ensures that given a good discretization of the model to start with, each local mesh modification procedure in itself does not cause drastic changes in the mesh. In fact, such a criterion may also be used to preserve important geometric features in the initial discretization. In particular, edge swapping, edge collapsing and node repositioning are prone to the problem of eliminating geometric features in the surface mesh and the above criterion can be successfully used to preserve them [13, 14].

Another important consideration in surface meshing and surface mesh modifications is the self-intersection of the mesh. While a self-intersecting surface mesh in itself is not a problem, it may be unusable in the context of using it as the boundary of a volume mesh. Therefore, procedures to ensure that mesh is not self-intersecting are necessary. One such procedure is presented in [13, 16] and is found to work well. This procedure is based on the assumption that in the limit of refinement the mesh cannot self intersect is the model is not self intersecting.

REFERENCES

[1] S. Adjerid, J.E. Flaherty, K. Jansen, and M.S. Shephard. Parallel finite element simulations of czochralski melt flows. In S.N. Atluri, editor, *Proceedings of ICES98*, Atlant, GA, 1998. to be published.

[2] T. J. Baker. Automatic mesh generation for complex three-dimensional regions using a constrained Delaunay triangulation. *Engineering with Computers*, 5:161–175, 1989.

[3] E. Bänsch. Local mesh refinement in 2 and 3 dimesnions. *Impact of Computers in Science and Engineering*, (3):181–191, 1991.

[4] M. W. Beall. *Framework for the Reliable Automated Solution of Problems in Mathematical Physics Over Aribitrary Domains Using Scalable Parallel Adaptive Techniques*. PhD thesis, Rensselaer Polytechnic Institute, Troy, NY 12180, 1998. In preparation.

[5] M. W. Beall and M. S. Shephard. A general topology-based mesh data structure. *International Journal for Numerical Methods in Engineering*, 40(9):1573–1596, May 1997.

[6] J. Bonet and J. Peraire. An Alternating Digital Tree (ADT) algorithm for 3D geometric searching and intersection problems. *International Journal for Numerical Methods in Engineering*, 31:1–17, 1991.

[7] H. Borouchaki, P. L. George, F. Hecht, P. Laug, and E. Saltel. Delaunay mesh generation governed by metric specifications. Part I. Algorithms. *Finite Elements in Anlaysis and Design*, 25:61–83, 1997.

[8] H. Borouchaki, P. L. George, and B. Mohammadi. Delaunay mesh generation governed by metric specifications. Part II. Applications. *Finite Elements in Anlaysis and Design*, 25:85–109, 1997.

[9] M. J. Castro-Diaz, F. Hecht, and B. Mohammadi. New progress in anisotropic grid adaptation for inviscid and viscous flow simulations. In *4th International Meshing Roundtable*, Albuquerque, NM, Oct 1995. Sandia National Labs.

[10] M. J. Castro-Diaz, F. Hecht, B. Mohammadi, and O. Pironneau. Anisotropic unstructured mesh adaptation for flow simulations. *International Journal for Numerical Methods in Engineering*, 25(4):475, 1997.

[11] S. D. Connell and M. E. Braaten. Semistructured mesh generation for three dimensional Navier-Stokes calculations. *AIAA Journal*, 33(6), 1995.

[12] H. L. de Cougny. Automatic generation of geometric triangulations based on octree/Delaunay techniques. Master's thesis, Civil and Environmental Engineering, Scientific Computation Research Center, Rensselaer Polytechnic Institute,Troy, NY 12180-3590, May 1992. SCOREC Report # 6-1992.

[13] H. L. de Cougny. *Parallel Unstructured Distributed Three Dimensional Mesh Generation*. PhD thesis, Rensselaer Polytechnic Institute, Troy, NY 12180-3590, May 1998.

[14] H. L. de Cougny. Refinement and coarsening of surface meshes. *Engineering with Computers*, 14(3):214, 1998.

[15] H. L. de Cougny and M. S. Shephard. "parallel refinement and coarsening of tetrahedral meshes". Technical Report 21-1995, Scientific Computation Research Center, Rensselaer Polytechnic Institute, Troy, NY 12180-3590, 1995. submitted to International Journal of Numerical Methods in Engineering.

[16] H. L. de Cougny and M. S. Shephard. Surface meshing using vertex insertion. In *Proceedings of the 5th International Meshing Roundtable*, 1996.

[17] H. L. de Cougny and M. S. Shephard. Parallel unstructured grid generation. In *CRC Handbook of Grid Generation*. CRC Press, to appear.

[18] B. E. de l'Isle and P. L. George. Optimization of tetrahedral meshes. In Ivo Babuska, Joseph E. Flaherty, John E. Hopcroft, William D. Henshaw,

Joseph E. Oliger, and Tayfun Tezduyar, editors, *Modeling, Mesh Generation, and Adaptive Numerical Methods for Partial Differential Equations*, pages 97–128. Springer-Verlag, New York, 1995.

[19] S. Dey, M. S. Shephard, and Marcel K. Georges. Elimination of the adverse effects of small model features by the local modification of automatically generated meshes. *Engineering with Computers*, 13(3):134–152, 1997.

[20] D. A. Field. Laplacian smoothing and Delaunay triangulations. *Comm. Appl. Num. Meth.*, 4:709–712, 1988.

[21] W. H. Frey and D. A. Field. Mesh relaxation: A new technique for improving triangulations. *International Journal for Numerical Methods in Engineering*, 31:1121–1133, 1991.

[22] P. L. George. *Automatic Mesh Generation.* John Wiley and Sons, Ltd, Chichester, 1991.

[23] P.-L. George and H. Borouchaki. *Delaunay Triangulation and Meshing - Application to Finite Elements.* Hermes, 8, quai du Marché-Neuf, 7500, Paris, 1998.

[24] P. L. George, F. Hecht, and E. Saltel. Automatic mesh generator with specified boundaries. *Computer Methods in Applied Mechanics and Engineering*, 92:269–288, 1991.

[25] P. J. Green and R. Sibson. Computing Dirichlet tesselation. *The Computer Journal*, 2:168–173, 1978.

[26] O. Hassan, K. Morgan, E. J. Probert, and J. Peraire. Unstructured tetrahedral mesh generation for three-dimensional viscous flows. *International Journal for Numerical Methods in Engineering*, 39:549–567, 1996.

[27] O. Hassan, E. J. Probert, K. Morgan, and J. Peraire. Mesh generation and adaptivity for the solution of compressible viscous high speed flows. *International Journal for Numerical Methods in Engineering*, 38(7):1123–1148, 1995.

[28] H. Jin and R. I. Tanner. Generation of unstructured tetrahedra by advancing front technique. *International Journal for Numerical Methods in Engineering*, 36:1805–1823, 1993.

[29] B. Joe. Delaunay triangular meshes in convex polygons. *SIAM J. Sci. Stat. Comp.*, 7(2):514–539, 1986.

[30] B. Joe. Delaunay versus max-min solid angle triangulations for three-dimensional mesh generation. *International Journal for Numerical Methods in Engineering*, 31:987–997, 1991.

[31] Y. Kallinderis, A. Khawaja, and H. McMorris. Hybrid prismatic/tetrahedral grid generation for complex 3-D geometries. In *AIAA-95-0211*, 1995.

[32] Y. Kallinderis, A. Khawaja, H. McMorris, S. Irmisch, and D. Walker. Hybrid prismatic/tetrahedral grids for turbomachinery applications. In *Proceedings of the 6th International Meshing Roundtable*, pages 21–32. Sandia National Laboratories, Albuquerque, NM, 1997.

[33] Y. Kallinderis and S. Ward. Hybrid prismatic/tetrahedral grid generation for complex 3-d geometries. In *AIAA-93-0669*, 1993.

[34] Y. Kallinderis and S. Ward. Prismatic grid generation for three-dimesnional complex geometries. *AIAA Journal*, 31(10):1850–1856, 1993.

[35] B. K. Karamete, R. Garimella, and M. S. Shephard. Recovery of an arbitrary edge on an existing surface mesh using local mesh modifications. *submitted to International Journal for Numerical Methods in Engineering*, 1998. SCOREC Technical Report #19-1998.

[36] A. Khamasayseh and A. Kuprat. Anisotropic smoothing and solution adaption for unstructured grids. *International Journal for Numerical Methods in Engineering*, 39:3163–3174, 1996.

[37] A. Khawaja, H. McMorris, and Y. Kallinderis. Hybrid grids for viscous flows around complex 3-D geometries including multiple bodies. In *AIAA-95-1685*, pages 424–441, 1995.

[38] A. Liu and B. Joe. On the shape of tetrahedra from bisection. *Mathematics of Computation*, 63(207):141–154, July 1994.

[39] R. Löhner. Some useful data structures for the generation of unstructured grids. *Communications in Applied Numerical Methods*, 4:123–135, 1988.

[40] R. Löhner. Matching semi-structured and unstructured grids for Navier-Stokes calculations. In *AIAA-93-3348-CP*, 1995.

[41] R. Löhner. Generation of unstructured grids suitable for rans calculations. In S. Idelsohn, E. Oñate, and E. Dvorkin, editors, *Computational Mechanics - New Trends and Applications*, pages 1–11. CIMNE, Barcelona, Spain, 1998.

[42] R. Löhner and P. Parikh. Three-dimensional grid generation by the advancing front method. *International Journal for Numerical Methods in Engineering*, 8:1135–1149, 1988.

[43] M. Mäntylä. *Introduction to Solid Modeling*. Computer Science Press, Rockville, Maryland, 1988.

[44] M. J. Marchant and N. P. Weatherill. Unstructured grid generation for viscous flow simulations. In *4th International Conference on Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, pages 151–162, Swansea, Apr 1994.

[45] D. L. Marcum. Generation of unstructured grids for viscous flow applications. In *AIAA-95-0212*, 1995.

[46] D. L. Marcum and N. P. Weatherill. Generation of unstructured grids for viscous flow applications. In *AIAA-95-1726*, 1995.

[47] D. L. Marcum and N. P. Weatherill. Unstructured grid generation using iterative point insert and local reconnection. *AIAA Journal*, 33(9):1619–1625, 1995.

[48] D. J. Mavriplis. Adaptive mesh generation for viscous flows using delaunay triangulation. *Journal of Computational Physics*, 90:271–291, 1990.

[49] B. E. Meserve. *Fundamental Concepts of Geometry.* Dover Publications, Inc., New York, 1983.

[50] P. Möller and P. Hansbo. On advancing front mesh generation in three dimensions. *International Journal of Numerical Methods in Engineering*, 38:3551–3569, 1995.

[51] M. Mortenson. *Geometric Modeling.* J. Wiley and Sons, New York, 1985.

[52] R. O'Bara, M. W. Beall, and M. S. Shephard. Specifying analysis information within a geometric framework. In *4th US National COngress on Computational Mechanics*, San Francisco, CA, August 6-8 1997.

[53] J. Peraire, K. Vahdati, K. Morgan, and O. C. Zienkiewicz. Adaptive remeshing for compressible flow computations. *J. Comp. Phys.*, 72:449–466, 1987.

[54] S. Pirzadeh. Viscous unstructured three-dimensional grids by the advancing-layers method. In *Proceedings of the 32nd Aerospace Sciences Meeting & Exhibit*, number AIAA-94-0417, Reno, NV, Jan 1994.

[55] M.-C. Rivara. A 3-D refinement algorithm suitable for adaptive and multi-grid techniques. *Communications in Applied Numerical Methods*, 8:281–290, 1992.

[56] H. Samet. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley Publishing Co., 1990.

[57] W. J. Schroeder. *Geometric Triangulations: with Application to Fully Automatic 3D Mesh Generation.* PhD thesis, Rensselaer Polytechnic Institute, Scientific Computation Research Center, RPI, Troy, NY 12180-3590, May 1991. SCOREC Report # 9-1991.

[58] W. J. Schroeder and M. S. Shephard. On rigorous conditions for automatically generated finite element meshes. In J. Turner, J. Pegna, and M. Wozny, editors, *Product Modeling for Computer-Aided Design and Manufacturing*, pages 267–281. North Holland, 1991.

[59] R. Sedgewick. *Algorithms in C++*. Addison-Wesley Publishing Co., Inc., 1992.

[60] D. Sharov and K. Nakahashi. Hybrid prismatic/tetrahedral grid generation for viscous flow applications. *AIAA Journal*, 36(2):157–162, Feb 1998.

[61] M. S. Shephard. The specification of physical attribute information for engineering analysis. *Engineering with Computers*, 4:145–155, 1988.

[62] M. S. Shephard. Meshing environment for geometry based analysis. *International Journal of Numerical Methods in Engineering*, R. H. Gallagher Special Issue.

[63] M. S. Shephard, M. W. Beall, R. Garimella, and R. Wentorf. Automatic construction of 3-D models in multiple scale analysis. *Computational Mechanics*, 17(3):196–207, Dec 1995.

[64] M. S. Shephard, H. L. de Cougny, R. M. O'Bara, and M. W. Beall. Automatic grid generation using spatially-based trees. In *CRC Handbook of Grid Generation*. CRC Press, to appear.

[65] M. S. Shephard and M. K. Georges. Automatic three-dimensional mesh generation by the Finite Octree technique. *International Journal for Numerical Methods in Engineering*, 32(4):709–749, 1991.

[66] M. S. Shephard and M. K. Georges. Reliability of automatic 3-D mesh generation. *Computer Methods in Applied Mechanics and Engineering*, 101:443–462, 1992.

[67] M. S. Shephard, T.-L. Sham, L.-Y. Song, V. S. Wong, R. Garimella, H. F. Tiersten, B.J. Lwo, Y. LeCoz, and R. B. Iverson. Global/local analyses of multichip modules: Automated 3-d model construction and adaptive finite element analysis. In *Advances in Electronic Packaging 1993*, volume 1, pages 39–49. American Society of Mechanical Engineers, 1993.

[68] C. A. Taylor. Clinical applications of cfd, visualization and virtual reality in cardiovascular medicine. In *Proceedings of the ASME Winter Annual Meeting*, Anaheim, CA, 1998. ASME.

[69] C. A. Taylor, T. J. R. Hughes, and C. K. Zarins. Finite element modeling of blood flow in arteries. *Computer Methods in Applied Mechanics and Engineering*, 158(1-2):155–196, 1998.

[70] M. G. Vallet, F. Hecht, and B. Mantel. Anisotropic control of mesh generation based upon a voronoi type method. In A.S.-Arcilla, J.Häuser, P.R.Eiseman, and J.F.Thompson, editors, *Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, pages 93–103. Elsevier Science Publishers B.V. (North-Holland), 1991, 1991.

[71] D. F. Watson. Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes. *The Computer J.*, 24(2), 1981.

[72] K. J. Weiler. *Topological Structures for Geometric Modeling*. PhD thesis, Rensselaer Design Research Center, Rensselaer Polytechnic Institute, Troy, NY, May 1986.

[73] K. J. Weiler. Boundary graph operators for non-manifold geometric modeling topology representation. In M. J. Wozny, H. W. McLaughlin, and J. L. Encarnacao, editors, *Geometric Modeling for CAD Applications*. North Holland, 1988.

[74] K. J. Weiler. The radial-edge structure: A topological representation for non-manifold geometric boundary representations. In M. J. Wozny, H. W. McLaughlin, and J. L. Encarnacao, editors, *Geometric Modeling for CAD Applications*, pages 3–36. North Holland, 1988.

[75] F. M. White. *Viscous Fluid Flow*. McGraw Hill, Inc., 2nd edition, 1991.

[76] V. S. Wong. Qualification and management of analysis attributes with application to multi-procedural analyses for multichip modules. Master's

thesis, Mechanical Eng., Aeronautical Eng., & Mechanics, Rensselaer Polytechnic Institute, Troy, NY 12180-3590, 1994.

[77] X. Xu, C. C. Pain, A. J. H. Goddard, and C. R. E. de Oliviera. An automatic adaptive meshing technique for Delaunay triangulations. *Computer Methods in Applied Mechanics and Engineering*, 161:297–303, 1998.