Entire Domain Advancing Layer Surface Mesh (EDAMSurf) Generation

by

Jasmeet Singh

B. Tech, Indian Institute of Technology (BHU), Varanasi, 2015

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Masters in Applied Science

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES
(Mechanical Engineering)

The University of British Columbia
(Vancouver)

December 2019

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the thesis entitled:

Entire Domain Advancing Layer Surface Mesh (EDAMSurf) Generation

submitted by **Jasmeet Singh** in partial fulfillment of the requirements for the degree of **Masters in Applied Science** in **Mechanical Engineering**.

Examining Committee:

Carl Ollivier-Gooch, Mechanical Engineering
Supervisor

XYZ, Mechanincal Engineering
Supervisory Committee Member

PQR, LMN Department
Supervisory Committee Member

Abstract

Use of unstructured meshes in the simulation of a computational field to solve for a real world problem is ubiquitous. Specially, solving fluid flow over bodies like an airplane or a turbine computationally requires a well discretized domain, or a mesh around the surfaces of these bodies. In Computational Fluid Dynamic (CFD) simulations over these surfaces, the flow at the viscous-boundary layer of the surface is very important as the gradients in the normal direction of the flow are sharp and are orders of magnitude higher than the gradients in the tangential direction of the flow. Hence, resolving the flow field in the boundary layer is vital for accurate simulation results.

A plethora of 3D boundary layer mesh generation techniques start off from a discretization of the surface. A majority of these techniques either use surface inflation or iterative point placement normal to the surface to generate the advancing layer 3D mesh. Generating boundary layer meshes in 3D depends on the quality of the underlying surface discretization. We introduce a technique to generate advancing layer surface meshes which would improve the mesh generation pipeline for 3D mesh generation. The technique takes an input triangulation of the surface, which is fairly easy to get, even for complex geometries. Surface segments are identified and these segments are meshed independently using a advancing-layer methodology. For each surface segment, a mesh is generated by advancing layers from the identified boundaries to the surface interior while deforming the existing triangulation. As the mesh-generation technique introduced here produces a closed-mesh, we get a valid mesh at each iteration of layer advancement.

The method introduced to generate advancing layer meshes produces semi-structured quad-dominant meshes with the ability to have local control over the aspect ratio of mesh elements at the boundary curves of the surface. Semi-structured 2D anisotropic meshes in the boundary layer regions have been shown to have superior fluid flow simulation results. However, the discretization of the surfaces poses challenges in replicating the same for volume meshes. Point placement in layers, local reconnection, front recovery, front collision handling and smoothing techniques used in the study help produce a valid surface mesh at each step of mesh generation. We demonstrate the ability of the meshing algorithm to tackle fairly complex geometries and coarse initial surface discretization.

Lay Summary

Discretization of geometries using a non-regular arrangement of mesh elements, called unstructured mesh generation is used widely for simulating flow over various objects in industry and government. The region near the surfaces of objects is particularly important during the simulation process because of the extreme non-linearity in the flow characteristics near the boundaries of objects. Hence, generating a well-discretized boundary layer mesh is key to superior flow simulation results. 3D mesh generation methodologies use a surface mesh as the starting point. Hence, the surface mesh plays an important role in the overall fluid flow simulation process.

A method to generate an advancing layer surface mesh is introduced in this paper. This method could be used to generate advancing-layer quad-dominant surface meshes with the required aspect ratio at sharp corners of the surface. 3D mesh generation procedures could use this mesh to produce advancing layer mesh or any other mesh. Example meshes are generated and shown to handle complex geometries.

Preface

All the work presented in this thesis is an intellectual product of a close working relationship between Jasmeet Singh and Dr. Carl Ollivier-Gooch. The implementation of the methods, the data analysis, and the manuscript preparations were done by Jasmeet Singh with invaluable guidance from Carl Ollivier-Gooch throughout the process.

Table of Contents

List of Tables

List of Figures

Glossary

****

To be updated

****

Acknowledgments

****
To be updated
****

Thesis Outline

****
Only for guidance right now, this page will be commented out.
****

1. Introduction
    - Introduction to Meshing
    - Introduction to unstructured meshing and its importance
    - Boundary Layer Phenomenon and its importance
    - Meshes that deal with such scenarios.
    - 2D previous works
    - 3D previous works
    - Surface Mesh generation methods
        - Parametric Mapping
        - Direct 3D methods
2.

Chapter 1

Introduction

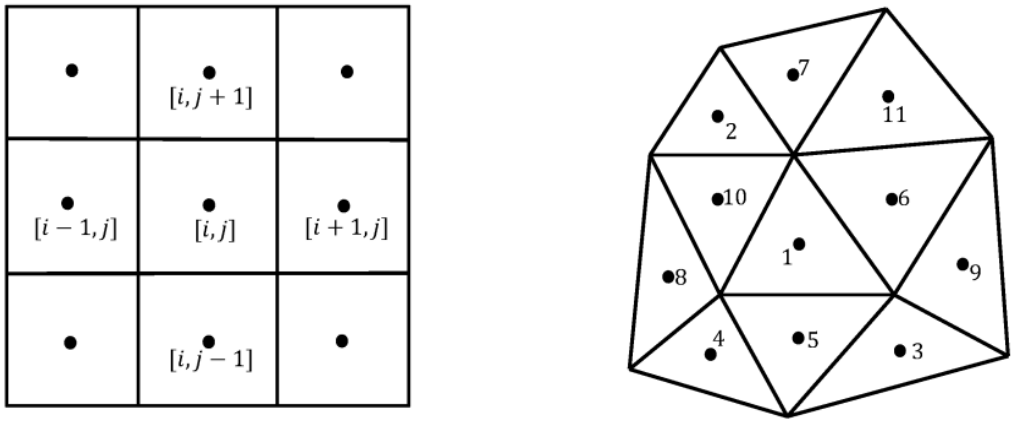
If I have seen farther it is by standing on the shoulders of Giants. — Sir Isaac Newton (1655)

Computational Fluid Dynamics (CFD) is a field of study where scientists and engineers architect new ways to numerically solve fluid flow equations. Before the advent of computers, numerical solutions of differential equations was done by hand. This lead to a great deal of work in the direction of creating faster algorithms to solve differential equations. An example is the development of the Fast Fourier Transform (FFT) by Cornelius Lancoz to increase the computation speed of Discrete Fourier Transforms (DFT). However, since the development and advancement of computers, engineers have had a significant amount of compute power to work with. This led to the development of highly accurate methods (as compared to before) to simulate flow over various objects. These simulations have since gotten bigger and better, typically including millions of Degrees Of Freedom (DOF), and now even starting to touch a billion in regular industry use.

1.1 Mesh Generation - A Brief Overview

The equations which govern the conservation of mass, momentum and energy of a moving fluid, called the Navier-Stokes equations, are solved in the given domain to simulate fluid flow in that domain. In order to numerically solve these equations, we need a discretization of the domain. This discrete basis required to solve the Navier-Stokes equations is called a mesh. Simply put, a mesh is a collection of points, lines and cells that together construct the space around a body in a fluid flow. Save a few exotic methods, almost all of the techniques in CFD require a mesh to solve the flow on.

The process of discretization of the domain to form the mesh is called mesh generation. Traditionally, mesh generation was a very manual process, where engineers used to place the mesh points and cells by hand. *todo: add citation and say that this was long time ago.* Such heuristic approach to mesh generation gave them a lot of freedom in discretizing the domain. Cells could be aligned to the boundaries of



(a)

(b) fig-unstructured-ij

Figure 1.1

objects. The quality of the cells, which was taken as some measure of the interior angle of the cells, was almost always chosen to be good. The benefits of this method were quite evident. However, there were some major drawbacks. The process of mesh generation was incredibly slow. Engineers would spend hours, sometime days to create the mesh for a given geometry. Also, mesh adaptation with solution was almost non-existant because that would have made the process even slower.

Definition 1 *A **mesh** M is a geometrical discretization of a domain Ω that consists of (a) a collection of mesh entities M_i of controlled size and distribution and (b) topological relationships or adjacencies forming the graph of the mesh. The mesh M covers Ω with neither overlaps nor holes.*

1.2 Structured and Unstructured Meshes

The evolution of mesh generation can be correlated to the evolution of compute power available to the boffins. With the advent of third generation computers (1964-1971) carrying integrated circuits, engineers were able to automate some of the manual processes in mesh generation. Meshes consisting of a template that repeats itself could be generated. These meshes were called *structured meshes* as their adjacencies or relationships could be known implicitly. Consider a grid in two dimensions as shown in Figure 1.1a. Given a cell (i, j) we can identify its neighbours as $(i-1, j)$ to the left and $(i+1, j)$ to the right. Similarly, cell $(i, j+1)$ will be to the top and $(i, j-1)$ would be to its bottom. The connectivity pattern repeats in such a mesh. Figure 1.2 shows a structured mesh generated for NACA 0012 airfoil around its leading edge. Notice the implicit connectivity of the cells even though the size of mesh elements is varying.

Structured meshes were attractive to engineers because of their low memory usage as the topology of the cells is repeated. Also, given simple domains to mesh, these meshes were optimal for minimizing the

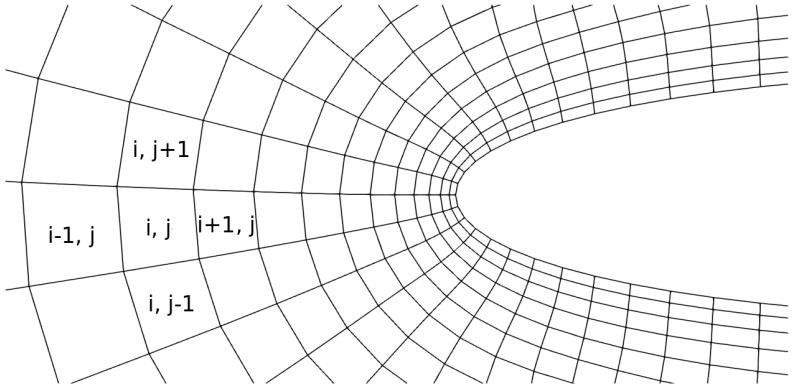


Figure 1.2: Structured mesh around leading edge of a NACA 0012 airfoil

errors in CFD, resulting in faster simulations [11]. Programming CFD solvers with these meshes was easy as cell connectivity occurs in a regular fashion. However, as the scope of CFD simulations grew over time and more complex geometries were becoming commonplace, the task of generating structured meshes around them proved to be a daunting one.

The disadvantage of using a structured mesh for more complex geometries is the increase in grid non-orthogonality or skewness that can cause unphysical solutions due to the transformation of the governing equations [37]. The transformed equations that accommodate non-orthogonality act as the link between the structured coordinate system (such as Cartesian coordinates) and the body-fitted coordinate system, but contain additional terms, thereby increasing the cost of numerical calculations and difficulties in programming. Hence, the characteristics of a structured mesh may affect the accuracy and the efficiency of the numerical schemes used by a solver. Additionally, the tedious process of generating such meshes for more complex geometries was hard to justify. Hence, more flexible and automatic methods were devised. These methods produced meshes in a more random manner but with lesser human intervention. Broadly, the meshes produced by such methods were classified as *unstructured meshes*. Figure 1.1b shows an unstructured mesh. The arrangement of the mesh elements is irregular. Along with the shape of the elements, we need a data structure to store the adjacencies of the mesh.

The cost of finding the flux at a cell boundary, a widely used parameter in Finite Volume Methods (FVM) for fluid flow simulations, for unstructured meshes is high as compared to their structured counterparts. Also, the amount of memory usage is also high as the topology of the mesh is no longer repeated. Still, they are more widely used today because of their capability to handle arbitrary complex geometries, their capability to automate the mesh generation process and their flexibility in refinement based on the geometry and/or the solution gradients. Figure 1.3 shows an unstructured mesh at the leading edge of a NACA 0012 airfoil. Notice the irregular arrangement of triangles around the airfoil geometry. The connectivity at each vertex of the mesh needs to be stored separately.

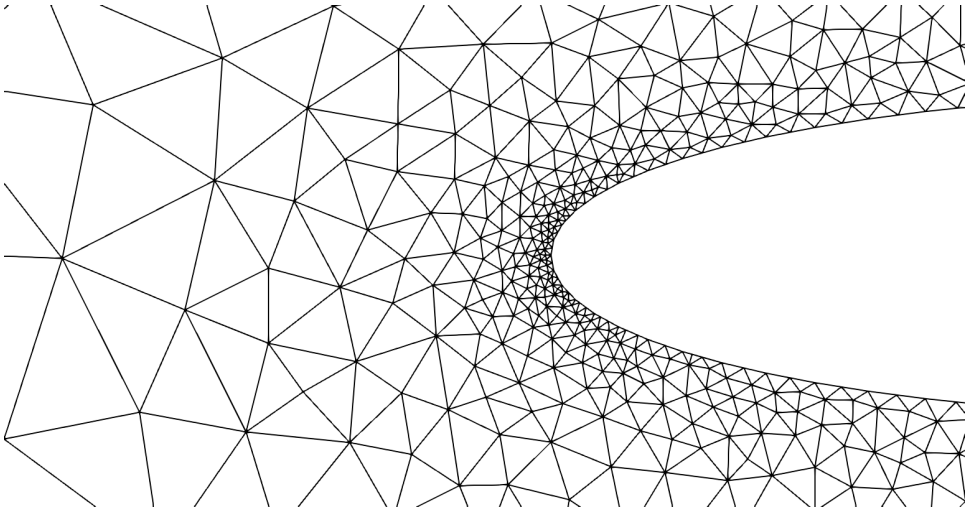


Figure 1.3: Unstructured mesh around leading edge of NACA 0012 airfoil

1.3 Simplicial and Non-Simplicial Meshes

Before discussing simplicial and non-simplicial meshes, we need to define certain terms. In geometry, a simplex is a generalization of the notion of a triangle or tetrahedron to arbitrary dimensions. For example, a 0-simplex is a point, a 1-simplex is a line segment, a 2-simplex is a triangle and a 3- simplex is a tetrahedron. See Figure 1.4 for an illustration.

Definition 2 *A **k-simplex** is a k-dimensional polytope which is the convex hull of its k+1 vertices*

A simplicial complex is a set composed of points, line segments, triangles, and their n-dimensional counterparts. In other words, a simplicial complex is a set strictly containing simplices only. Figure 1.5 shows a three-dimensional simplicial complex.

Definition 3 *A **simplicial complex** K is a set of simplices that satisfies the two following conditions: a) Any face of a d-dimensional simplex from K is also in K b) The intersection of any two simplices S1 and S2 is either ϕ (null set) or a simplex of dimension $d \leq min(d_1, d_2)$ that is a subset of both S1 and S2*

A mesh which contains only simplicial mesh elements is called a simplicial mesh. For example, a mesh which contains only triangular simplices is called a triangulaiton. In other words, a **triangulation** is the division of a surface or plane polygon into a set of triangles, usually with the restriction that each triangle side is entirely shared by two adjacent triangles. It was proved in 1925 that every surface has a triangulation, but it might require an infinite number of triangles *todo: add citation or omit*. Figure 1.6 shows a triangulation of a torus.

Definition 4 *A **triangulation** of a topological space X is a simplicial complex K, homeomorphic to X, together with a homeomorphism h: $K \rightarrow X$.*

A non-simplicial mesh is simply a mesh which is not simplicial. Such a mesh contains mesh elements

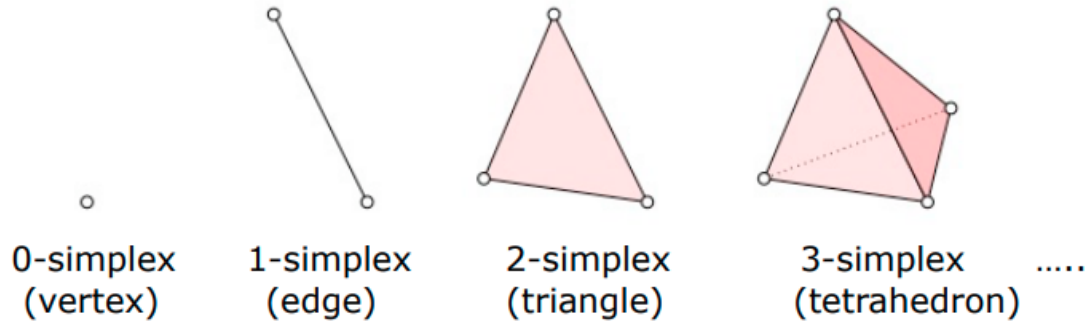


Figure 1.4: n dimensional simplices.

other than simplices too. For example, in two dimensions, a mesh which contains quadrilateral elements or quads will be called a non-simplicial mesh. In three dimensions, a mesh containing hexahedral elements would fall under the category of non-simplicial meshes.

Simplicial elements have been traditionally used for mesh generation. These elements are simple to work with and provide good flexibility in terms of discretization of a domain. These benefits make simplicial meshes very simple to produce. However, some of the drawbacks of simplicial meshes have led to mesh generation techniques with non-simplicial elements.

Consider a triangulation of a surface. The Euler Formula states that for any convex polyhedron, the number of vertices and faces together is exactly two more than the number of edges. Mathematically,

$$V - E + F = 2 \tag{1.1}$$

where V is the number of vertices, E is the number of edges and F is the number of faces in the polyhedron. For a triangulation, each edge is shared by two faces. Also, each face has three edges associated with it. Hence, the number of faces is $2/3$ times the number of edges, or $F = (2/3) \times E$. Substituting this in equation 1.1, we get

$$\begin{aligned} V - E + F &= 2 \\ V - E + \frac{2}{3}E &= 2 \\ V - \frac{1}{3}E &= 2 \\ 3V &\approx E \end{aligned} \tag{1.2}$$

Hence, the number of edges is three times the number of vertices in a triangulation (asymptotically).

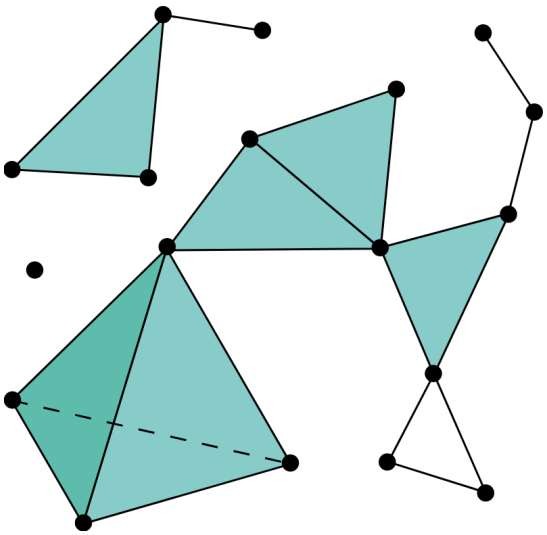


Figure 1.5: A three-dimensional simplicial complex.

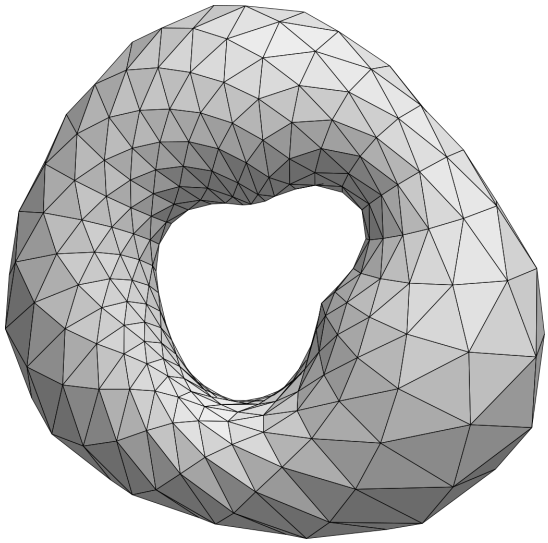


Figure 1.6: Triangulation of a torus.

On the other hand, the number of edges is two times the number of vertices for a closed quadrilateral surface mesh. A similar derivation could be done for three-dimensional simplicial and non-simplicial elements. The number of edges in a tetrahedral mesh is about seven times the number of vertices. On the other hand, in a hexahedral mesh, the number of edges is only about three times the number of vertices (asymptotically). Higher connectivity for simplical meshes leads to higher computational for solving on a given mesh using vertex-based discretization methods. Hence, non-simplicial meshes are substantially more efficient than simplical meshes for a given number of unknowns or grid points.

Additionally, regular arrays of nonsimplical elements may also enhance accuracy, owing to a local cancellation of truncation errors that may not occur on groups of nonsimilar simplical elements [26]. In two-dimensions, quadrilateral elements have been preferred over triangles in highly stretched two-dimensional grids due to their lower connectivity [3]. These advantages of non-simplical mesh elements have resulted in fully non-simplicial mesh generation techniques [4, 41]. The scheme introduced by Blacker *et al.* [4] uses a paving methodology with several mesh element collision checks and special conditions for concave corners to generate an isotropic all-quad two-dimensional mesh for complex geometries. One such mesh is shown in Figure 1.7.

Even though non-simplical mesh elements have been favored for a variety of scenrarios in mesh generation, and especially while generating highly-stretched elements, they have their disadvantages. It is quite difficult to develop an automatic mesh generation strategy which creates non-simplicial meshes conforming to complex surface and 3D geometrical configurations. Manual input is usually required to mesh such geometries. In these situations, simplical mesh elements are quite useful in dealing with the complexity in the geometry. Hence, a hybrid mesh containing both simplical and non-simplical mesh elements is a reasonable choice for mesh generation. We will revisit this in section 1.6 where we reason about choosing a hybrid scheme to mesh the surface.

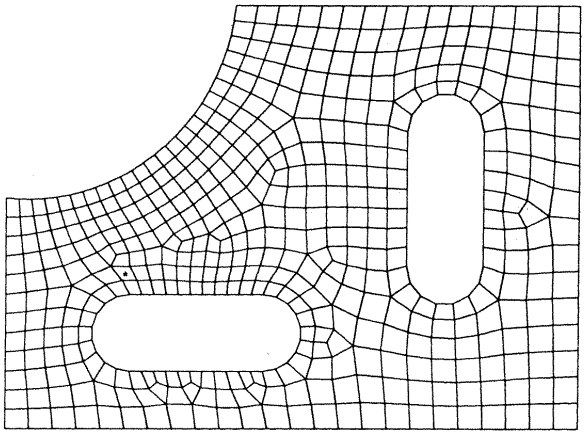


Figure 1.7: A non-simplicial quad mesh generated with paving methodology [4].

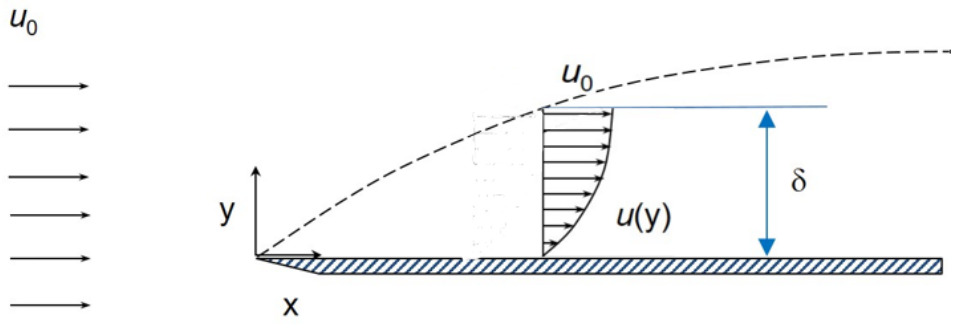


Figure 1.8: Fluid flow over a flat plate.

1.4 Boundary Layer Meshes

With the advent of unstructured mesh generation techniques, a broader selection of geometries could be dealt with. This immensely increased the scope of CFD solvers and pushed the limits of numerical methods in terms of accuracy and speed. However, a different approach was needed for the parts of the mesh in which viscous forces are dominant as compared to the inertial forces. In other words, at the location of the viscous boundary layer, the gradient of physical measures like velocity is several magnitudes higher in one direction as compared to its orthogonal direction. A mesh generation technique which would help in resolving such strong gradients of the velocity in the boundary layer was required. For example, consider a flow over a flat plate as shown in Figure 1.8. The velocity of the fluid at the surface of the plate is zero. However, the velocity becomes freestream velocity u_0 very near to the surface. The thickness of this layer of fluid, where the velocity of the fluid goes from a value of zero to a value of around 0.99 times the freestream velocity is called the boundary layer thickness δ. It is also called the viscous boundary layer as the viscous effects are dominant in this region of the flow.

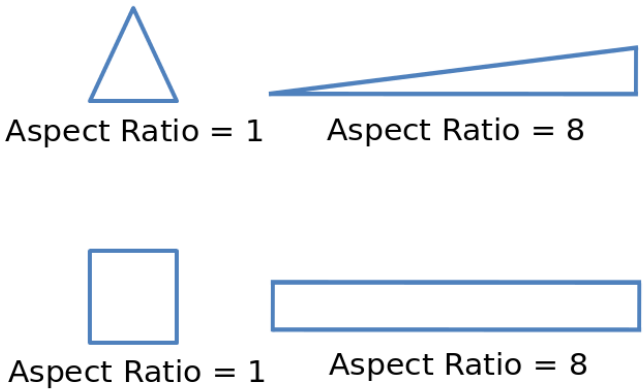


Figure 1.9: Illustration of different aspect ratio triangular and quadrilateral elements.

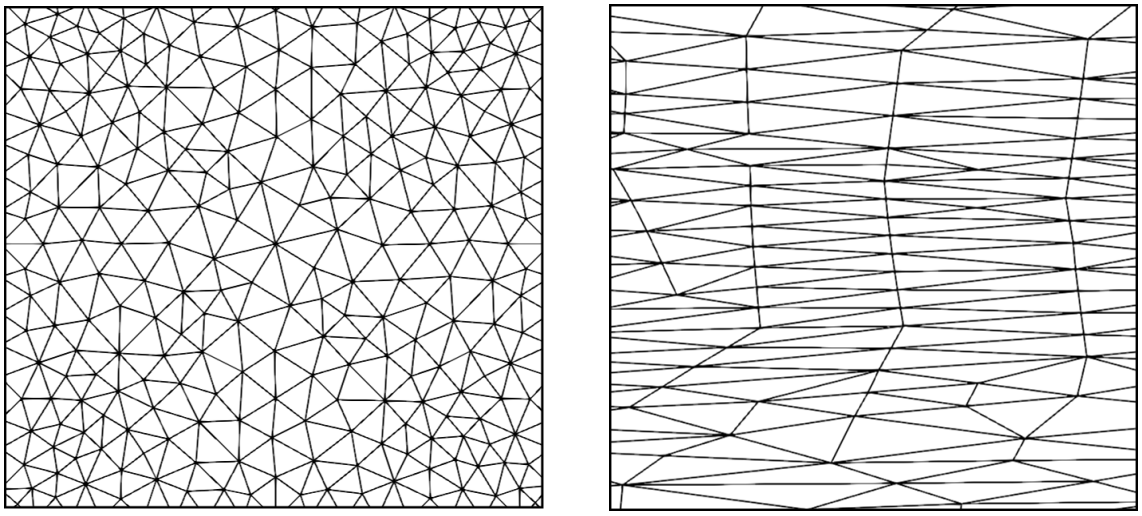
1.5 Anisotropic Meshing

The need to resolve the steep gradients in the boundary layer of the fluid flow gave rise to a type of mesh development strategy called anisotropic meshing. An anisotropic mesh is simply a mesh which has highly stretched elements. In other words, the aspect ratio of the elements for an anisotropic mesh would be very high. Aspect ratio of a mesh element is simply a size measure of the element. For tris and quads, the ratio of the length of the largest edge of the element to the smallest one is usually taken as the aspect ratio of that element. Figure 1.9 illustrates some different aspect ratio triangular and quadrilateral elements.

A highly anisotropic packing of the cell elements provides a large number of Degree Of Freedoms (DOFs) along the direction of steep gradients of physical quantities such as velocity. Traditionally generated isotropic meshes are incapable of resolving such steep gradients [14]. *todo: how big is big? that is, what are the typical numbers for aero?*

Figure 1.10a shows an isotropic mesh. The mesh elements have almost equal edge lengths. Here, the mesh elements are triangles and resemble equilateral triangles for most parts of the mesh. Given a point in the mesh, all the directions are the same and there isn't any bias towards a particular direction. For an isotropic physical process, such a mesh will serve the purpose and resolve gradients in all directions given the resolution of the mesh is appropriately chosen. However, if the physical process to be simulated is highly anisotropic, such as the velocity distribution along the boundary layer of the flat plate, as discussed in Section 1.4, such a mesh will fail to resolve the steep velocity gradients. It would have to be refined to get the required refinement at the boundary, increasing the total number of DOFs in the mesh by a polynomial factor. Hence, a more reasonable mesh generation strategy is needed.

Figure 1.10b shows an anisotropic mesh. The triangular elements of the mesh are highly streched, with one edge being considerably shorter than the other two. The number of DOFs is distributed over the



(a) Isotropic mesh **(b)** Anisotropic Mesh

Figure 1.10: Isotropic and Anisotropic Mesh Fragments

domain in a fashion so as to have the majority of the DOFs along the steep gradients of the physical quantities to be simulated. Hence, cell alignment with the solution to capture anisotropic flow features is possible with such a mesh.

1.5.1 Brief Literature Review - Anisotropic Meshing

Several techniques have been developed to generate meshes in two dimensions with some sort of anisotropy. Some of these techniques have also been generalized to surfaces. However, isotropically-meshed surfaces with a smooth element-size variation are generally easier to mesh than anisotropically-meshed surfaces with strong size variations [37]. Many techniques developed in 2D have been generalized to 3D while some new methodologies have been deviced for volume meshing. We go over some of these methods briefly.

1.5.1.1 2D and Surfaces

Most of the initial attempts at generating stretched element meshes in two dimensions used a Delaunay mesh and a locally mapped space to get the required level of anisotropy [27]. A mesh generated by such a method is shown in Figure 1.11. Some techniques used an approach of using a locally structured or semi structured mesh for the regions requiring high anisotropy [28].

Many methods for generating anisotropic meshes come under the category of metric adaptation of the mesh. These techniques generally use a Delaunay type initial mesh and refine it anisotropically using a solution metric. Shimada *et al.* provided an automated method to obtain anisotropic triangulation of a parametric surface. Given a domain geometry and a tensor field that specifies desired anisotropic

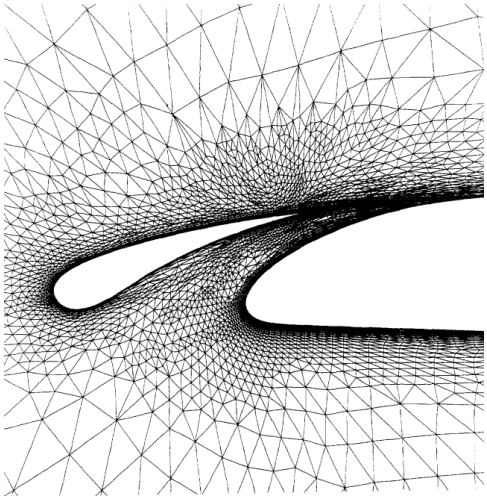
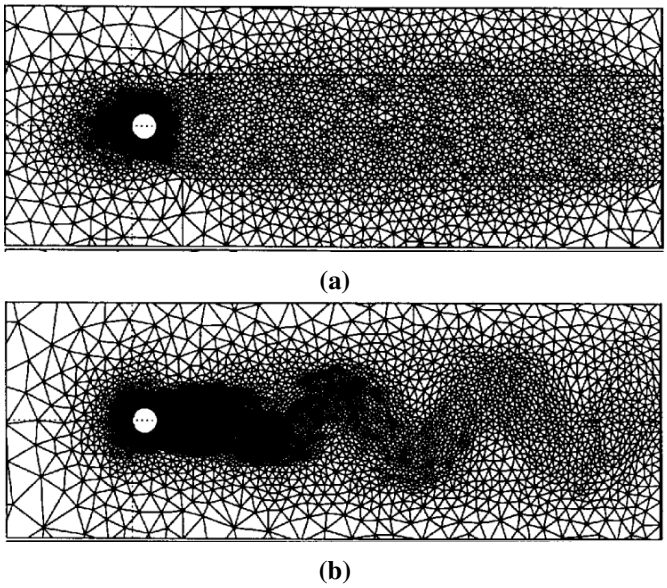


Figure 1.11: Illustration of adaptively refined mesh for a two-element airfoil configuration near the gap region [27].



(a)

(b)

Figure 1.12: Initial mesh (a) and adaptively generated anisotropic mesh (b) using solution metric evaluation [6].

node-spacing, a proximity-based interacting force field is defined and the force balance configuration is dynamically simulated [35]. Castro *et al.* applied mesh adaptation technique to generate anisotropic meshes, to compressible viscous flows for a wide range of Reynolds and Mach numbers [6]. An illustration of this method can be seen in Figure 1.12b.

Kunert *et al.* showed an anisotropic mesh generation algorithm that refines the mesh anisotropically by calculating the local error estimate on the initial mesh [20]. This algorithm was presented for both two dimensional and three dimensional meshes. Another mesh adaptation technique by Li *et al.* tried to align the mesh to a calculated metric tensor from the physics of the problem [23]. Figure 1.13 shows one such mesh.

Advancing Layer methods are a practically successful family of anisotropic mesh generation techniques. A method introduced by Pirzadeh produced unstructured triangular/tetrahedral grids with high-aspect-ratio cells using the advancing layer or the grid-marching strategy [32]. Another method which used the advancing layer strategy, with several mesh collision checks, was introduced by Lohner [24]. Here, the mesh was produced by inflating the boundary curves in the direction of surface normals. Special care was taken while dealing with the concave corners of the mesh so as to avoid mesh element collisions.

While the majority of anisotropic mesh generation strategies have focused on simplicial mesh elements, there have been some works which generate all quadrilateral (quad) surface meshes. A method by Lee *et al.* showed an anisotropic quadrilateral mesh generation scheme which generates a background triangular mesh and then uses a cell merging procedure in the parametric space to produce the desired

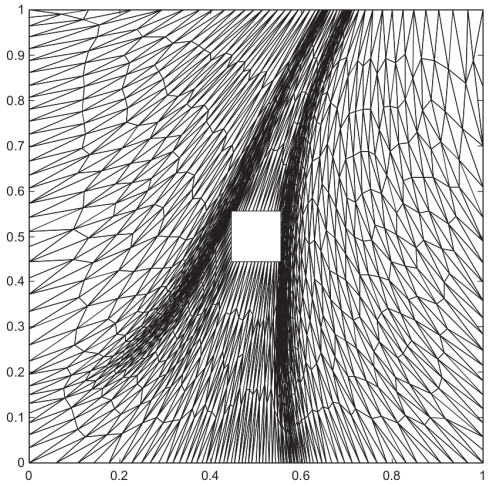


Figure 1.13: Anisotropic mesh generated by aligning the mesh elements to a metric calculated from the solution on an isotropic mesh [23].

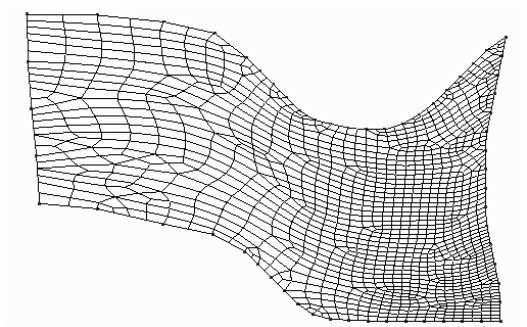


Figure 1.14: Anisotropic quadrilateral mesh generated with an input triangulation and solution contours [38].

mesh [22]. A different kind of method was adopted by Viswanath *et al.* to generate quadrilateral meshes with anisotropy and directional control [38]. A 2D geometric domain and desired level of anisotropy - as a metric tensor over the domain, specifying mesh sizing in two independent directions - is taken as an input. Node locations are calculated by closely packing rectangles in accordance with the inputs. An example mesh generated with this strategy is shown in Figure 1.14.

1.5.1.2 3D Anisotropic Meshing

Given an initial surface discretization, several 3D mesh generation algorithms have been developed to create anisotropic volume meshes. Many methods which generate two-dimensional anisotropic meshes can be extended to 3D [6, 24, 28].

A generalized advancing layer method for generating 3D boundary layer mesh was presented by Garimella *et al.* [15]. Model boundaries grow into the domain to fill to generate the mesh in this method. An example mesh can be seen in Figure 1.15. Another method was introduced by Ito *et al.* [18]. Here, a hybrid mesh was generated which comprised of tetrahedra, prisms and pyramids. First, the domain was filled with an isotropic tetrahedral mesh. Subsequently, the boundary walls were shifted inwards and the resulting gap between the tetrahedra and the walls was filled up with prismatic elements. Figure 1.16 shows a mesh generated using such a strategy. This mesh generation strategy, of using a highly anisotropic semi-structured mesh near the boundaries of the domain (however, with no gradation over the boundary of the domain), and an isotropic mesh further away from domain boundaries is quite common in 3D meshing. Another example of such a strategy is shown in the work done by Sahni *et al.*

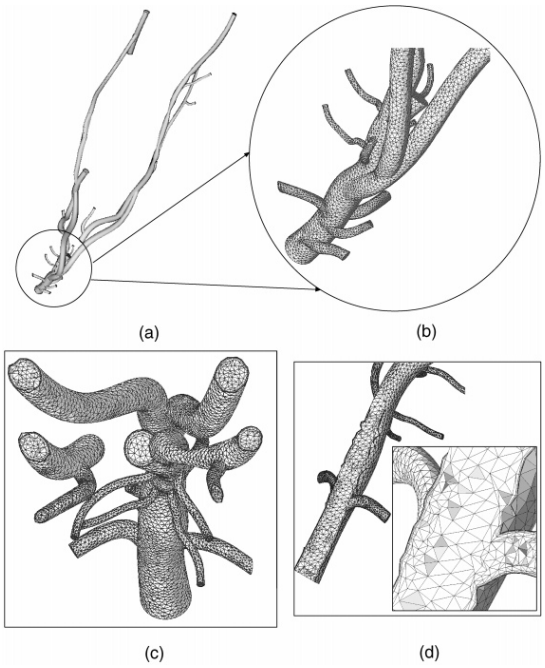


Figure 1.15: Boundary layer mesh for simulation of ow in blood vessels: (a) geometric model; (b) zoom in of surface mesh in the encircled region; (c);(d) cross-sections showing the boundary layer and isotropic meshes [15].

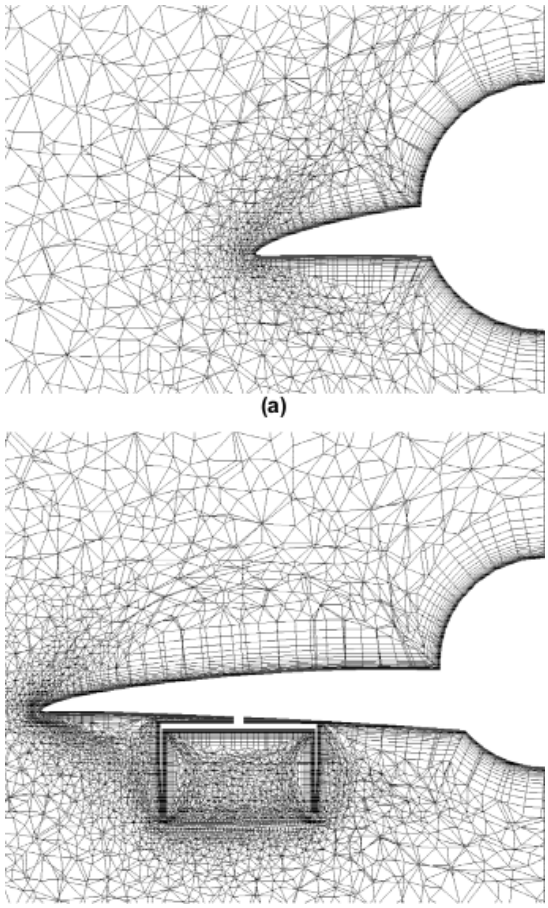


Figure 1.16: Hybrid grid for an aircraft (NAXST-2). Two images show the cross-section of the mesh at two different locations along the chord of the airfoil [18].

[33]. Here, mesh adaptation was applied to a hybrid mesh, with semi-structured stretched elements at the boundary and isotropic elements away from the boundary. Figure 1.17 shows a mesh generated by this method.

The method used by Shimada and Yamada in two dimensions has also been extended to three dimensions to generate anisotropic tetrahedral meshes. Given an arbitrary input anisotropy function, the algorithm generates high quality anisotropic tetrahedral mesh that conforms to the input geometry [39]. A mesh generated from this method can be seen in Figure 1.18.

In another work, a metric-orthogonal anisotropic mesh is generated. In addition to aligning the mesh to a metric, mesh elements are also aligned with the eigenvectors [25]. The quality of the input metric strongly affects the output mesh.

Another method which generates a hybrid mesh using semi-structured boundary layer mesh and an

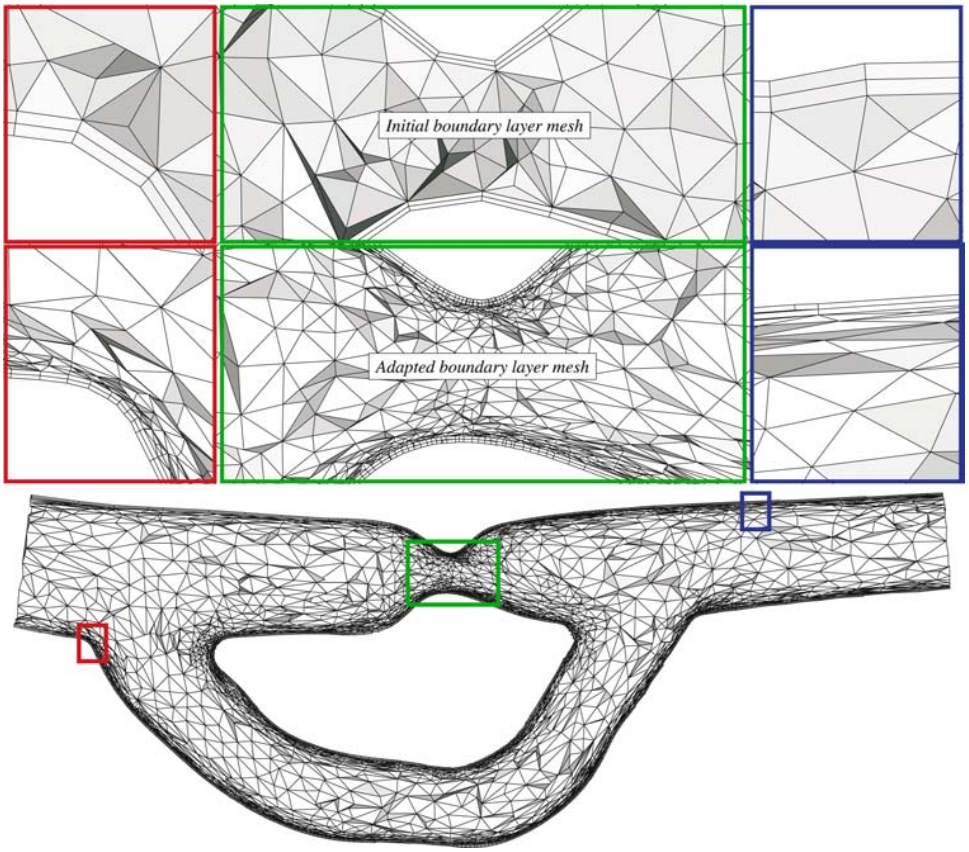


Figure 1.17: Collection of mesh faces cut by the vertical center- plane through the adapted boundary layer mesh of a porcine aorta. The windows correspond to magnified views which also show the initial boundary layer mesh [33].

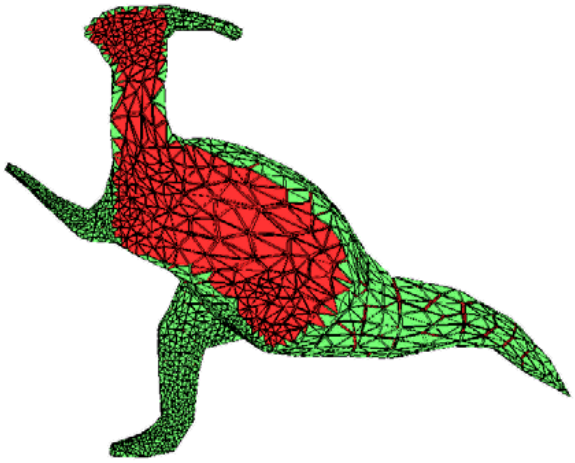


Figure 1.18: Anisotropic Tetrahedral Mesh generated using ellipsoidal bubble packing methodology [39]

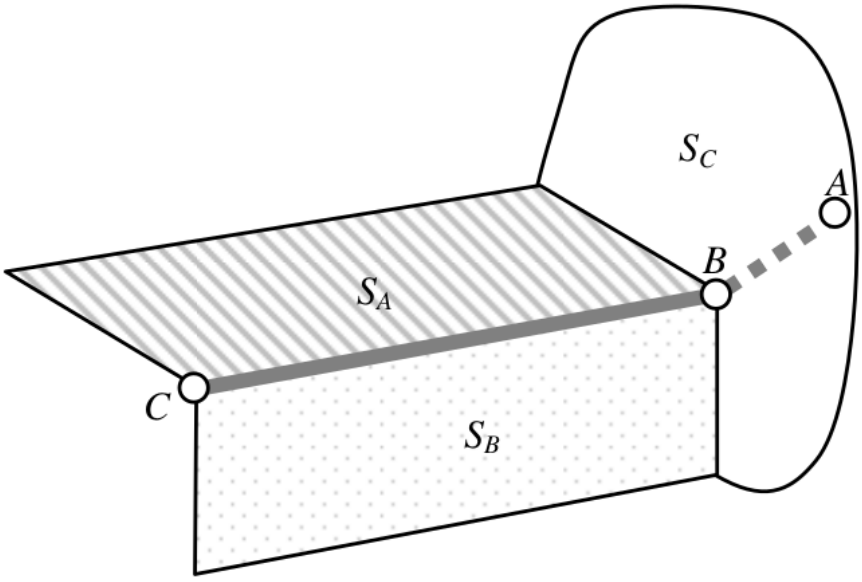


Figure 1.19: Three boundary surfaces S_A, S_B, and S_C [19]

isotropic outer mesh was given by Ito *et al.* [19]. Here, special treatment of sharp corners on the surface was done by adding multiple marching directions at the sharp corner. This resulted in generation of a better volume mesh. The research showed that if the corners of the surface mesh, from which the volume mesh is generated, are treated specially and are hence refined more than other regions of the surface, the volume mesh thus generated would be of superior quality. More importantly, such special treatments of sharp corners in the surface mesh helps to create semistructured elements (for viscous boundary layer mesh) around singular points on the surface. A very important problem addressed by this study is one of the pivotal reasons to write this thesis.

Consider Figure 1.19 from the work by Ito *et al.* [19]. An intersection between three surfaces is considered. There are labeled S_A, S_B and S_C. The surfaces can represent a wing upper surface, a blunt trailing edge and a fuselage, respectively. As most of the surface meshes are simplical, isotropic and don't treat corners specially, the task to generate a valid viscous boundary layer mesh from them is quite difficult. This difficulty is compounded at complex corners like the one shown in the figure. The paper puts forward a technique to discretize the surface further by adding new nodes and faces to the surface at such corners. In the figure, nodes are added along the direction *BA*. This process helps in creating additional normal directions near the corners of the surface and solves the problem of singularity to a great extent.

1.6 Motivation

1.6.1 Surface Mesh Generation Strategies

Stretched volume meshes are generated from an initial triangulation of a surface. These surfaces are usually represented by Non-Uniform Rational B-Splines (NURBS) in various Computer Aided Design (CAD) packages. Generation of the surface mesh requires a separate methodology or algorithm as compared to the two-dimensional mesh generation as the mesh elements include a third dimension.

A majority of surface mesh generation methods produce a surface mesh from the parametric mapping of a two-dimensional mesh onto a surface [7, 16]. Usually, a two-dimensional Delaunay mesh is generated over a parametric surface and then mapped onto the curved surface. This method is attractive as it is simple and there are a number of robust Delaunay mesh generation schemes available to generate the initial two-dimensional mesh. However, the mapping from 2D to 3D doesn't always give satisfactory results in terms of mesh element quality. Also, this method doesn't always form well shaped surface elements, especially when the surface derivatives vary over the domain [29]. Meshes generated by such methods are generally isotropic and simplical. Hence, in addition to dealing with the badly shaped elements, several other post-processing operations need to be applied to them before generating a boundary layer volume mesh.

A different approach to surface mesh generation is adopted by methods which use the first fundamental form of the surface. In such methods, a metric is derived from the parametric representation of the surface and mesh elements are placed over the surface in an advancing front fashion with respect to this metric. Culliére devised one such method where he used a nodal density function based on the curvature of the surface to discretize it [10]. An advancing front triangular mesh generation method was presented where 3D parametric surfaces could be meshed with good quality mesh elements. Tristano *et al.* [36] presented a method which used a Riemannian surface definition to determine the amount of distortion of the elements in parametric space. An advancing front method is applied to utilize this metric and generate good quality triangular elements over the surface. Figure 1.20 shows a mesh generated with this method. These methods, which use the parametric representation of the surface produce good quality triangular surface meshes. However, they are not well-suited to generate three dimensional anisotropic meshes with highly stretched elements. Additionally, the elements generated by such methods are simplical and hence, have their own drawbacks as discussed in Section 1.3.

Lastly, surface meshes can also be generated by placing the mesh elements directly on the surface of the geometry. Lan and Lu [21] provide a method to produce a surface mesh over a given analytical surface. A given element density function is utilized and mesh elements are placed on the analytical surface using the advancing front technique. The elements are optimized considering factors such as surface curvature, element to element turning angle and the given density distribution, to get a high-quality surface mesh. Figure 1.21 shows meshes generated with this method. With respect to being an input for

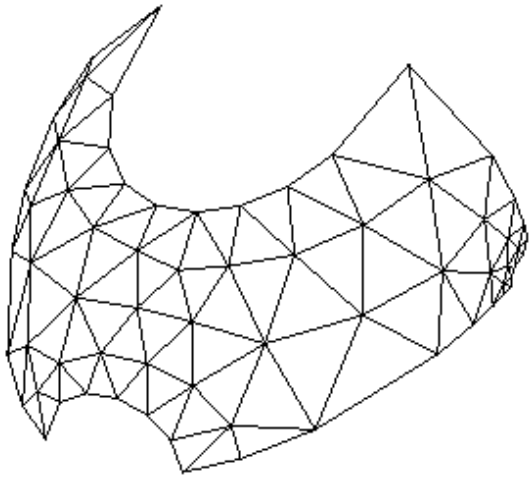


Figure 1.20: A CAD surface meshed with Reimannian space mesher

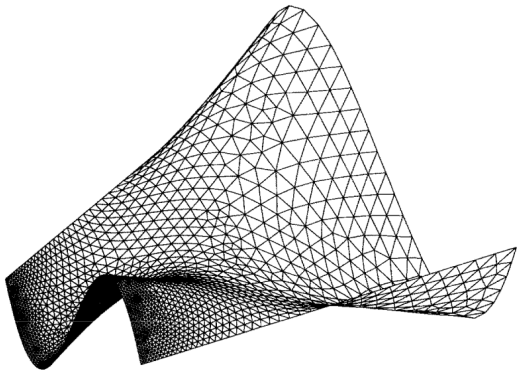
a 3D viscous boundary layer mesh, these meshes have similar drawbacks as the ones discussed earlier. Mesh elements generated are triangular rather than quadrilateral or hybrid. Also, there is an additional input required to generate the mesh, which is the density function. Lastly, the boundary curves of the surface are not dealt with specially and the complete surface is isotropically meshed.

1.6.2 Consolidating The Discussion

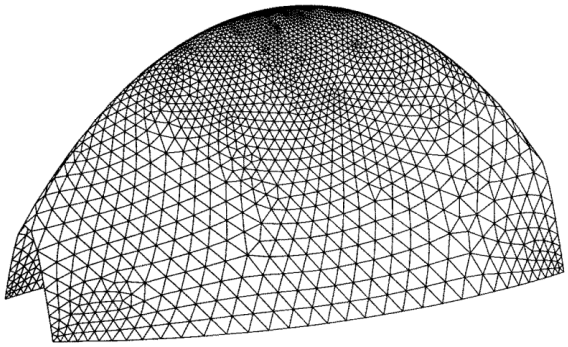
1.6.2.1 Hybrid Meshing

Consolidating our discussion on surface meshes, we find that most of the 3D surface meshes generated are triangular. In other words, these meshes are simplical. We discussed the pitfalls of using simplical mesh elements at the regions of anisotropy or at surface boundaries in Section 1.3. To recapitulate, higher vertex connectivity for simplicial meshes makes them inefficient when using a vertex-based discretization method. On the other hand, quad meshes in two dimensions and hex meshes in 3D are substantially more efficient than their simplicial counterparts for the same number of Degrees of Freedom. Non-simplicial quadrilateral elements have been preferred over triangles in highly stretched two-dimensional grids due to their lower connectivity [3]. In addition to that, non-simplicial elements arranged in repeating arrays may enhance solution accuracy by local cancellation of truncation errors while their simplicial counterparts may fail to do so [26].

Even though non-simplicial elements are more efficient for flow solution, conforming such elements to complex geoemtry remains a daunting task. Generally, manual input is required to generate all-quad surface meshes or all-hex volume meshes in the regions of geometric complexities. Hence, usage of simplicial mesh elements in these regions is a reasonable alternative. Hence, a surface mesh generation scheme is required, that generates a quad-dominant mesh with a very limited number of triangular ele-



(a) A ruled surface.



(b) A free form surface.

Figure 1.21: Meshes generated directly over the surface by utilizing the analytical surface representation and element density distribution [21].

ments. Regular arrays of highly-stretched quadrilateral elements should form the anisotropic boundary layer mesh. On the other hand, triangular elements could be used whenever one wants to conform to the geometry and increase the quality of the mesh elements generated.

1.6.2.2 Good Input For Anisotropic 3D Meshing

Following the discussion from 1.5.1.2, it is particularly difficult to generate hybrid 3D meshes from an isotropic surface discretization. At the minimum, several validation checks needs to be performed at the singularity points of the surface so as to generate a valid viscous 3D mesh. In addition to that, the anisotropy of the 3D mesh becomes extremely difficult to control at the complex corners of the surface if they are not handled specially. One solution discussed earlier was to add additional marching directions and nodes at the complex corners on the surface of the mesh so as to add additional normal directions near the complex corners. However, the root cause of the problem is the isotropic surface discretization which is completely unaware of its usage into a anisotropic 3D volume mesh generator. Hence, a better alternative is desirable which automates the process of special treatment of surface cusps.

1.6.2.3 Automatic + Flexibility

Surface mesh generation methodologies either have several assumptions on the type of surfaces they can support or need a considerable amount of human intervention to complete. Especially for the cases where complex corners exist on the surface, a good surface mesh would need special arrangement of nodes and marching directions on the surface so as to produce a high-quality anisotropic volume mesh. Block structured meshes could be produced on a surface by manually discretizing the domain into several subdomains and meshing them using structured mesh generation technique independently. But again, the mesh generation process would be very tedious and would take a considerable amount of manpower. Hence, an automatic mesh generation technique which has a good level of anisotropy at surface boundary curves is desirable to serve as a good input to 3D mesh generation. Lastly, the mesh generation process should be flexible in terms of the surface representation it can accept as an input. Many methods discussed earlier use the parametric form of the surface to generate the mesh. Additionally, some methods also require an element density distribution to be specified for the entire domain of the surface which needs to be meshed. These constraints again make it cumbersome to generate good quality surface meshes (for anisotropic volume meshes) for a wide variety of surface topologies. Hence, a method of surface mesh generation which accepts a more widely used surface representation is highly desirable.

1.6.3 Entire Domain Advancing Layer - Surface Mesh (EDAM-S) Generation

We present a surface mesh generation algorithm that serves to address the aforementioned issues and has the desired qualities. The algorithm generates a surface mesh which has anisotropic characteristics normal to the boundary curves of the surface. The mesh elements are highly stretched at these boundary curves of the surface and have a required level of directional anisotropy normal to the boundary curves. For the examples in this thesis, the boundary curves of the surface are chosen to be the curves where the surface normal direction jumps abruptly. Practically, the algorithm is independent of the selection of the boundary curves and can work for any selected set of boundary curves for a given surface. However, for the sake of automation and simplicity, the boundary curves are chosen to be the sharp features on the surface topology.

The surface discretization produced by the algorithm given in the thesis consists of a hybrid grid. Most of the elements (usually greater than 95%) are quadrilateral mesh elements. A regular pattern of quad elements is repeated from the boundary curves of the surface towards its interior. This pattern helps retain the boundary curve topology towards the interior regions of the mesh. In addition to that, an appropriate discretization of the boundary curves of the surface can provide several additional normal directions to the surface near the boundary curves. This may be vital in solving the problem of complex corners or singularity points on the surface while generating a 3D viscous volume mesh.

While most of the elements in the mesh are non-simplicial, some triangular mesh elements are also

utilized to discretize the surface. These elements are quite useful when dealing with complex surface topologies such as concave corners. Additionally, triangular mesh elements help to control the growth of the aspect ratio of the mesh elements as the mesh proceeds from the boundary towards surface interior. Overall, the surface mesh hence generated utilizes the advantages of both simplicial and non-simplicial elements. As the surface mesh produced is quad-dominant, it can be used to produce a hex-dominant volume mesh (for anisotropic boundary layer meshes or otherwise).

To keep the mesh development algorithm flexible and simplify the mesh generation pipeline, the input to the mesh generation scheme is only the initial surface triangulation as a stereolithography (STL) file. STL files are the standard in 3D printing and CAD packages. These files contain the information regarding all the triangles in the mesh. Each triangle contains its normal and coordinates of its vertices. Hence, these files are easy to generate and use. Additionally, it is easy to generate a surface triangulation from a parametric or analytic representation of the surface. Innumerable surface triangulation packages are available to generate isotropic and triangular surface discretization with the given amount of refinement. Hence, taking a surface triangulation as an STL file is a reasonable choice for generating the desired surface meshes.

The surface mesh generation technique described in the thesis is based on a closed advancing front method. The surface triangulation imported to the mesh generator is taken as the initial mesh and mesh elements are placed over it to generate the desired surface mesh. The method needs minimal user input and automatically generates an advancing layer mesh from a given input surface triangulation. The only primary user input in addition to the surface triangulation are the initial extrusion length x of the surface mesh and the growth ratio g. The initial extrusion length, x, defines the thickness of the first layer of the surface mesh. If a user defined value is not provided, the mesh generation algorithm makes a reasonable assumption and proceeds to mesh the domain.

Sequentially, the input surface is first segmented into several sub-surfaces so that each of them can be meshed independently. Boundary curves of the surfaces are identified and the points on the boundary curves iteratively march towards the interior of the surface. Initial extrusion length and growth ratio can be varied so as to give the required level of anisotropy at each point of the mesh or for a given boundary curve. A valid surface mesh is produced after each layer is marched from the boundary curves. This gives the user freedom to advance until a given level of anisotropy is attained or until the advancing front routine terminates itself.

Several quality checks and improvements are made during the mesh generation process. These include controlling the element to element turning ratio, limiting the deviation of the mesh elements from the underlying surface (or rather, sub-surface), swapping of edges so as to increase element quality, edge collapse to control aspect ratio and improve element quality, smoothing, and special handling of corners. All these subroutines play a vital role in the overall mesh generation process.

However important may the advancing layer surface mesh generation process described here be, its creation poses challenges for geometries that are highly complicated and/or highly curved. In addition to

that, tackling sharp concave corners on a surface mesh is a challenge in itself. These challenges raise robustness issues for most of the mesh generation procedures. Our advancing front routine includes several validity checks to avoid these issues. However, a completely robust algorithm which could generate an anisotropic surface mesh for all such complicated topologies is still a work in progress.

1.7 Outline

****

Outline will be updated as I complete various chapters of the paper. Also, is it fine to place the outline of the thesis at this location? For me, this was fine as I could not place it before without breaking the flow of the chapter.

****

Chapter 2

Methodology Part 1: Geometry Representation and Point Placement

In this section, we will talk about the initial import of surface triangulation and storing the surface as a collection of bezier surface patches. Then, the point placement subroutine is explained which decides the mesh element structure. Lastly, a small discussion on the local mesh element quality improvement is added to explain the face swapping algorithm.

2.1 Surface Import

2.1.1 Surface Representation - A brief overview

Surfaces can be represented in various forms. A surfaces can be represented with their implicit form, given by

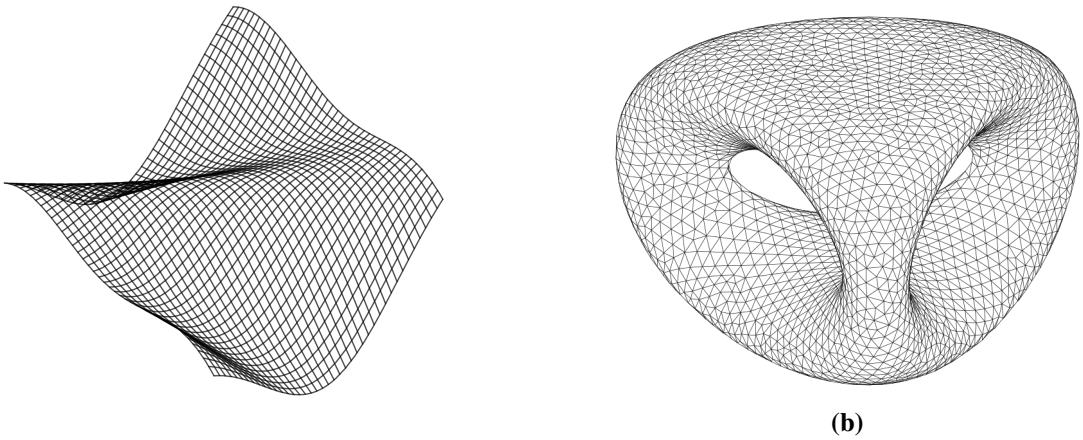
$$F(x,y,z) = 0 \tag{2.1}$$

were, the surface is defined as the locus of points whose coordinates (x,y,z) satisfly the above equation. If the equation 2.1 is linear in variables $x,y,andz$, it represents a plane. If the equation 2.1 is of second degree in $x,y,$ or z, it represents quadrics. If the implicit equaiton can be solved for one of the variables as a function of of the other two, say z is solved in terms of x and y, we obtain an explicit surface

$$z = F(x,y) \tag{2.2}$$

Figure 2.1 shows an implicit and an explicit surface together with their mathematical formulations.

Apart from the explicit and implicit forms, surfaces can also be represented in their parametric form. The coordinates of a point (x,y,z) of the surface patch are expressed as functions of parameters u and v



(a) **(b)**

Figure 2.1: (a) An explicit surface, given by $z = \cos((x+y)) + \frac{x^2}{6} - \frac{y^2}{6}$. (b) An implicit surface, given by $2y\left(y^2-3x^2\right)\left(1-z^2\right)+\left(x^2+y^2\right)^2-\left(9z^2-1\right)\left(1-z^2\right)=0$.

in a closed rectangle:

$$x = x(u,v), \quad y = y(u,v), \quad z = z(u,v), \quad u_1 \leq u \leq u_2, \quad v_1 \leq v \leq v_2. \tag{2.3}$$

In vector notation, the parametric surface can be specified by a vector-valued function

$$\mathbf{r} = \mathbf{r}(u,v) \tag{2.4}$$

The parametric representation of surfaces is the most versatile out of the three. It is axis independent and is highly flexible in terms of defining complex intersections and point classification. It is generally easier to manipulate free-form shapes in parametric form than implicit or explicit forms [31]. Hence, most CAD packages use the parametric form of surfaces. A bezier surface patch and NURBS are examples of the parametric form of a surface.

2.1.2 Surface File Format

Given a free-form 3D surface geometry, various CAD packages could be used to encode it and store it in a file. Encoding the geometry using an approximate mesh is one of the most common methods adopted to store a surface geometry. An approximate mesh is created by covering the surface geometry with a series of tiny imaginary polygons. Triangles are the most common polygon used for mesh generation. The encoded surface mesh can be stored in a file for sharing and future reference purposes. These files

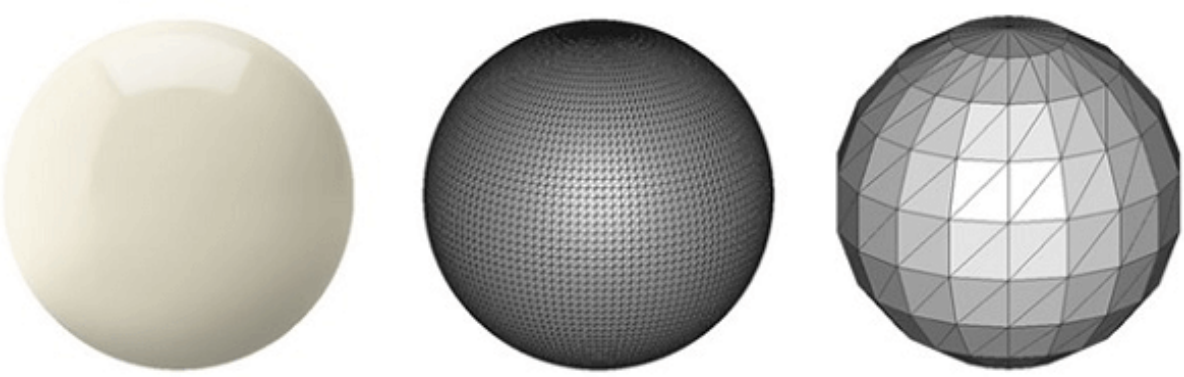


Figure 2.2: The perfect spherical surface on the left is approximated by tessellations. The figure on the right uses big triangles, resulting in a coarse model. The figure on the center uses smaller triangles and achieves a smoother approximation [1]

store the vertices of the triangles as well as the outward normal directions to the triangles. This process of tiling a surface with non-overlapping geometric shapes is also known as tessellation. Hence, the file formats used for storing the surface representation are called tessellated formats.

Due to the independent development of various CAD packages, a plethora of surface file formats are present in the mesh generation ecosystem. Many of these formats, such as DWG file format by AutoCAD and BLEND file format by Blender[9] are proprietary. Hence, many of these cannot be shared between people working on different CAD packages. Neutral file formats are used to solve this problem. These formats can be shared easily among people working on different meshing software. One of the most common neutral surface file formats is the STL (STereoLithography) file format. This format is compatible with most of the CAD and visualization softwares. Hence, we use the STL file format to import the surface geometry into our mesh generation algorithm.

An STL file stores the surface as a triangulated mesh. The following information is stored for all the triangles in the STL file format:

1. The coordinates of the vertices
2. The components of the unit normal vector to the triangle pointing outwards with respect to the 3D model

Innumerable software packages are available online which can be used to triangulate a surface (see [2] for a list of such software). A fine triangular mesh can be considered as an approximate encoding of a given surface geometry. The approximation could be improved by increasing the number of triangles or decreasing their size. However, using smaller triangles results in larger number of triangles needed to tile the surface. This increases the mesh file size. Hence, a user should define the mesh element size according to the kind of refinement needed.

2.2 Surface Import and Segmentation using the Common Geometry Module (CGM)

As explained in Section 2.1.2, we import the surface geometry as a triangulation from an STL file. The Common Geometry Module (CGM) package is used to read the surface file and store the triangulation for further processing. The triangulated surface is stored as a collection of segmented sub-surfaces. The segmentation in CGM is done by identifying features in the surface. The only input parameter for surface segmentation accepted by CGM is the feature angle. We keep this feature angle value to be 135°. This value helps us to identify sharp corners and edges on the surface. Figure 2.3 shows a surface triangulation of an arbitrary mechanical part. The surface triangulation is segmented into 10 sub-surfaces, which are identified by the sharp features on the surface. Four of these are shown in outline in the figure.

Each triangle imported in CGM is stored as a quartic-Bezier patch. We will refer to the imported triangulation as T and the underlying Bezier surface representation as S. The underlying Bezier surface representation S is considered to be the ground truth for the surface mesh and the mesh points are placed directly over S. The initial imported triangulation T is taken as the initial mesh.

Using T as our initial mesh means that we can use a closed advancing layer mesh generation methodology. There are advantages and disadvantages of using such a methodology. Open advancing layer method requires less work overall to generate the mesh as there are no points to delete or reconnect to, ahead of the front. On the other hand, handling mesh layer collisions is more tricky in open advancing layer method as no connectivity information is available ahead of the front in such methods. This leads to abrupt layer closures, which are undesirable in an anisotropic mesh.

Closed advancing layer mesh generation method generates a valid surface mesh at any given point in time during the mesh generation process. Hence, there is more flexibility in terms of when to stop the marching layers and output the mesh in its current state. Additionally, connectivity information is known ahead of the front. This information is helpful while tackling front collisions. This will be explained in more detail when we talk about handling front collisions in *ref*.

2.2.1 Advancing Layer Initialization

Sharp features of the imported surface are used by CGM to segment the surface. The boundary curves of the segmented surface denote these sharp features. For the purpose of this thesis, we consider these identified boundary curves as the initial front of the mesh. In other words, the boundary curves of the segmented surfaces (or sub-surfaces) serve as the zeroth layer of the advancing layer surface mesh.

Picking the boundary curves of the sub-surfaces to serve as the initial front helps us create anisotropy normal to these boundary curves. This way, desired refinement can be obtained normal to these boundary curves, which is a desirable feature as these boundary curves define the surface features.

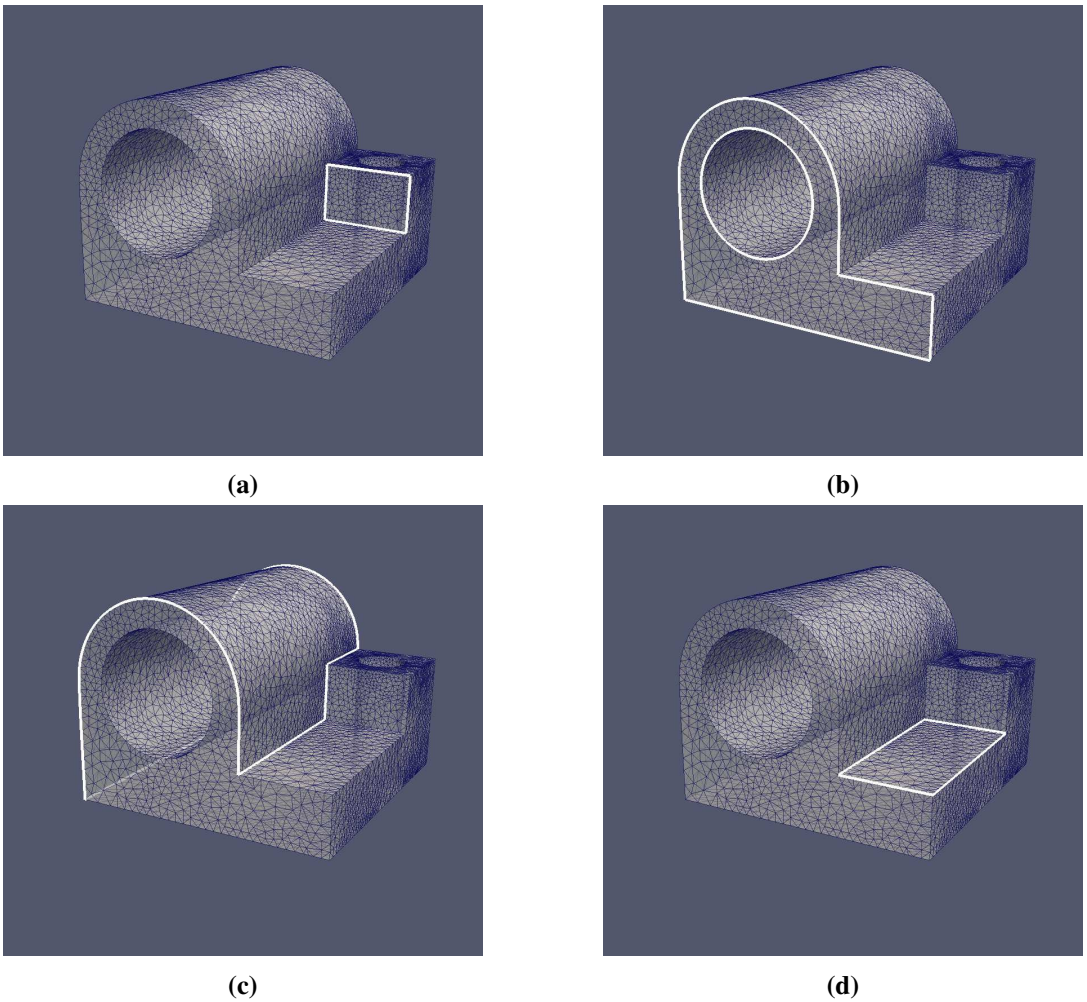


Figure 2.3: An example input triangulation of an arbitrary mechanical part. Four of the segmented surfaces are outlined.

The discretization of the boundary curves imported along with the surface triangulation defines the refinement and gradation along the boundary curves (or along the zeroth front). Hence, the refinement and gradation along the boundary curves of the surface is defined by the input triangulation and is up to the user to vary. The surface mesh generated by EDAMSurf hence provides with anisotropy along the normal direction (on the surface) to these boundary curves of the sub-surfaces.

We chose the boundary curves identified by CGM to serve as the zeroth layer of EDAMSurf. However, the initial front could be chosen to be something else without affecting the rest of the mesh generation procedure. For eg. curves along high principal curvature directions on the surface could be added to the zeroth layer of the mesh to get the required anisotropy along highly curved regions of the surface. This is a work in progress and will be added to EDAMSurf in the future.

2.3 Point Placement

After importing the surface triangulation, we have a valid underlying surface representation with us. Also, segmented sub-surfaces and their boundaries curves provide us with the boundaries we need to march off of. Each vertex on these boundary curves is extruded in two directions, one each for the sub-surfaces which share the boundary point. To make things simpler, two copies of each boundary vertex (a vertex on a boundary curve) are created and associated with the two sub-surfaces which share the vertex. This untangles the process of generating the surface mesh of the two sub-surfaces, which can now be meshed independently. Later, when we need to output the final surface mesh, these copies of vertices are recombined.

Meshing sub-surfaces independently has several advantages. If one sub-surface mesh fails to generate, other sub-surfaces would still continue to generate the advancing layer mesh. Different layer on layer *growth ratios* can be used for different sub-surfaces, hence providing the flexibility to choose different anisotropy for each sub-surface. Also, parallelisation of the surface mesh generation subroutine would be simpler. From here onwards, the discussion would focus on generating the mesh for a single sub-surface, which is a segment of the complete surface.

The mesh generation routine starts by initializing the front as the boundary curves of each sub-surface. All the boundary points are marked as candidate marching points and form the starting layer in the mesh. The points at the boundary curves of the sub-surfaces serve as the parent points for the first layer inserted into the mesh.

The data structure created to store a vertex on the advancing front of the mesh also stores the edges adjacent to that vertex so as to identify the marching directions. Hence, the extrusion direction or marching direction of a point is obtained from the location of the vertex, its adjacent edges, and the underlying sub-surface which is being meshed. The extrusion direction calculation procedure is explained in detail in the next subsection.

Similar to the vertices on the boundary curves, edges on the boundary curves are also stored in two copies, one for each sub-surface associated with that edge. Each edge on the advancing layer stores the direction into the interior of the sub-surface it bounds with respect to the surface normal. In other words, the edge datastructure stores its orientation relative to the sub-surface associated with it. As the sub-surfaces sharing a common boundary curve are meshed independently, it is easy to identify the normal to an edge along the sub-surface for advancing the front in the mesh generation algorithm.

2.3.1 Extrusion Direction and Length

For each of the sub-surfaces of the geometry, the advancing layer routine iteratively picks a point from its boundary and extrudes it in a given direction. After evaluating the extruded point, we project the point onto the underlying surface. This process is set up to be of two steps for simplicity, accuracy and

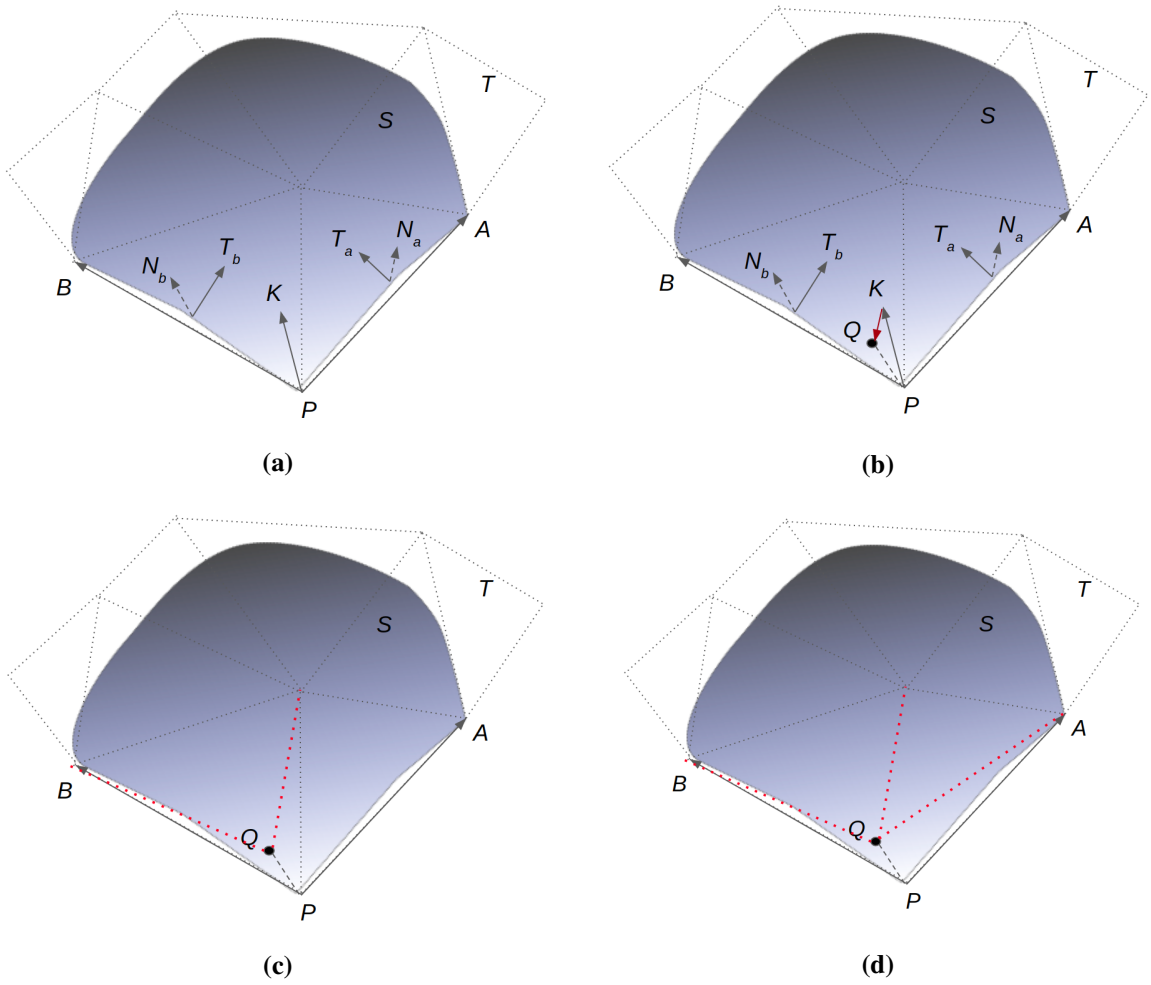


Figure 2.4: (a) Calculation of the extrusion direction of vertex P at the boundary curve of the sub-surface. (b) Projection of the extruded vertex K onto the underlying surface. The projected vertex is marked Q. (c) Insertion of the new vertex Q in the triangulation T. Red dotted lines denote the new edges created in the mesh. (d) One of the edges next to Q is swapped so as to improve the quality of the mesh.

computational efficiency as will be explained later.

In the first step, we extrude the parent point to get the extruded point. We would interchangeably call the extruded points as the kid points as they represent the successors of their parent points from the previous layer. The direction of this extrusion is set up to be the average of the normals of the parent vertex's adjacent edges in the tangential plane of the sub-surface. In other words, the normals of the adjacent two edges of a vertex on the underlying sub-surface are averaged to get the extrusion direction.

Consider Figure 2.4a. The initial input triangulation is marked T and the Bezier surface representation is marked S. Point P is on the boundary curve of the surface. We need to find the extrusion direction of P to know where its kid will be located. In the underlying triangulation T, PB and PA are the edges

adjacent to P on the boundary of the surface S. We first find the mid-points of quartic-Bezier curves PB and PA which are constructed by CGM as a part of constructing the quartic-Bezier triangular surface patches from the underlying triangulation T. Next, we find the normal directions on the surface at these mid-points. The normal directions are carefully queried from the sub-surface which is being meshed currently. The normal directions are labeled as N_b and N_a in the figure. We take the cross product of the vector PB with N_b to get the direction T_b. The vector T_b is normal to the edge PB as well as tangential to the surface S. Similarly, we find the vector T_a, which is normal to the edge PA and tangential to surface S. The direction of extrusion $\overrightarrow{PK}$ is chosen to be the average of the direction of T_a and T_b.

The *initial extrusion length* is an input parameter provided by the user. This length is used as the extrusion length when the boundary points are extruded for the first time to the interior of the surface. The initial extrusion length can vary with boundary vertices as the points are extruded independently. Hence, this extrusion length can either be supplied by the user for all the points of the boundary separately or as a single value for all the boundary points. To obtain the best quality quadrilateral elements, we scale the extrusion length at a given vertex on the advancing layer with respect to the interior angle between the direction vectors T_a and T_b. If the vertex is a concave corner vertex, the extrusion length is increased so as to create good quality quad elements in the next layer. Scaling extrusion length for concave vertices also helps in avoiding immediate front collapse. This process is described in detail in Section 2.3.3.

2.3.2 Vertex Projection and Insertion

After we have extruded the point, we project it on to the underlying sub-surface S, which is considered as the ground truth for the geometry in our mesh generation process. This operation ensures that all the points we insert in the mesh are on the underlying geometry. If we do not project to the surface, geometric errors would compound in subsequent layers. Hence, exact projection on S is done to avoid accumulation of errors. CGM supports projection of a point in space to the sub-surface. If the projection is outside the sub-surface, the projected point is moved to the closest location on the sub-surface. This is highly unlikely to occur as the extrusion direction is always towards the interior of the sub-surface. Refer to Figure 2.4b for an illustration. Extruded point K is projected onto the sub-surface S so as to get point Q. After obtaining the projected point, a check is done to make sure that the newly obtained point Q is on the surface S by calculating the shortest distance between Q and S. The distance is asserted to be zero for all projections.

Points are inserted in the mesh and the mesh elements are subdivided to include the new point. The candidate point for insertion Q can subdivide an existing triangle to replace a previous triangle with three new ones, or it can subdivide an edge to replace existing two triangles with four new ones. To find the best triangle or edge for subdivision, we first make a guess for the triangle to insert the point. Any triangle in the surface interior adjacent to the point being extruded is chosen as the first guess. Starting from this triangle, we iteratively jump to the best edge or triangle by comparing the barycentric

coordinates of the new point with respect to the triangle in consideration. This technique suffers from two disadvantages. First, we need to compare double precision values of barycentric coordinates for making a decision on which triangle to choose for insertion. If the values are too close, the point might be inserted in the wrong triangle and would eventually lead to deviation of the mesh from the underlying surface. Second, the process of iteratively finding the right triangle for insertion might cause an infinite loop.

Both of these problems are substantially reduced with a good isotropic initial triangulation. However, we add several validity tests to avoid these problems even for a coarse initial triangulation. These checks include orientation checks of the triangles formed with respect to the surface, thresholding the maximum deviation of the newly formed triangle from the surface and thresholding the dihedral angles between two triangles on the surface. We use an epsilon value of 10^{-5} while comparing the values of barycentric coordinates to zero. Also, we insert the point on a face rather than in a triangle when the ratio of the second-smallest barycentric coordinate to the smallest one is more than a set threshold (10^2). This helps us avoid very skinny triangles with large obtuse angles and also helps in reducing unnecessary face swapping in the mesh.

After advancing one layer to the surface interior, we increase the extrusion length at each point by a factor. This factor, called the *growth ratio*, specifies the anisotropic layer-on-layer extrusion length growth as we march on the surface. A value of growth ratio between 1.1 and 1.4 generally gives us satisfactory anisotropy at the boundary curves of the surface.

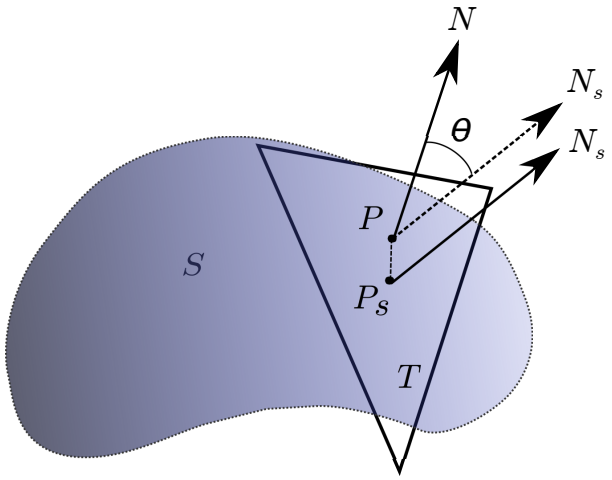


Figure 2.5: Triangle T deviates from the underlying surface S. P is the centroid of T and P_s is the projection of P on S. The deviation is calculated as the interior angle between the normal to the triangle PN and the normal to the surface at P_s, which is P_sN_s. The angle θ represents the deviation here. The deviation is exaggerated for illustration purposes.

2.3.3 Extrusion Length Scaling

The initial extrusion length x_i for the mesh is an input parameter provided by the user. This length, along with the growth ratio g defines the refinement in the direction normal to the boundary curves in the surface mesh. Keeping x_i constant along the boundary curve is one way to proceed with the mesh generation procedure. However, at the concave corners of the mesh, using a uniform extrusion length would result in immediate front collapse as the neighbours of the corner would race towards each other while the corner vertex lags behind in the advancing layers. Hence, some sort of extrusion length scaling is required at the front which takes into account the angle included between the edges adjacent to the vertex on the front.

Let us consider a concave corner vertex P in the mesh located on an advancing layer. The adjacent edges to P on the layer are PR and PQ. The included angle between these two edges is denoted as 2θ. Let's assume that the length of the edge PQ is l (see Figure 2.6) and the extrusion length for the layer at which vertex P is located is h. We need to find a reasonable extrusion direction and extrusion length for the corner point P.

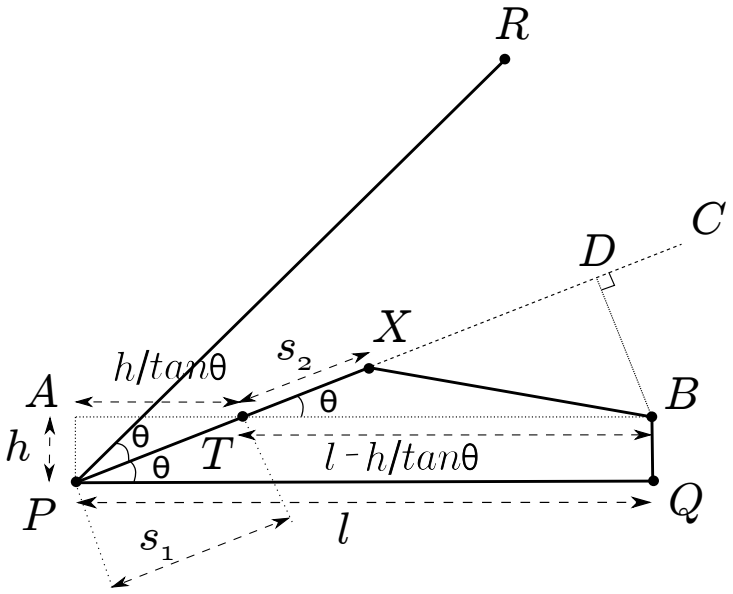


Figure 2.6: Extrusion

For good quad cells in the surface mesh, i.e. quad cells with angles as close to right angles as possible, we need the point P to 'race' a bit initially towards the interior to catch up with its adjacent points. This would be helpful in avoiding mesh collisions. Hence, we formulate a method to find the scaling factor for the extrusion length.

The line PC is constructed at equal angles from edge PR and PQ. Let us place the point X extruded from point P on the line PC. In an orthogonal structured mesh scenario, if the angle between PR and PQ was 180 degrees, the extrusion direction should have been along the edge PA and the extruded point would be at a distance h from the point P, which coincides with point A in the figure. The quad thus

formed would have been $PABQ$. However, this is not the case. We have a skewed interior angle between PQ and PR. If we place the extruded point at a distance h from P along PC, the mesh is going to fold onto itself in the next few layers. To avoid this, we take an alternative approach. We place the point X further away than distance h. This distance is calculated using the following strategy.

The area of the quad $PABQ$ would have been $h \times l$ in the orthogonal mesh case. Our strategy is to put the point X on the line PC such that the area of quad $PXBQ$ is also $h \times l$.

Line PC intersects AB at point T. The area of the quad $PTBQ$ is common to both quads $PABQ$ and $PXBQ$. Hence, to make the area of the quads $PABQ$ and $PXBQ$ the same, we just have to equate the area of $\triangle APT$ and $\triangle XTB$.

$$Area(\triangle ATP) = Area(\triangle TXB) \tag{2.5}$$

Using this equation, we can find the length PX. Line segment PT is of length s_1 and TX is of length s_2. Hence, $PX = s_1 + s_2$. s_1 can be easily computed as $h/\sin\theta$. To find s_2, we use the area constraint from equation 2.5. Area of triangles $\triangle ATP$ and $\triangle TXB$ can be easily computed in terms of variables h, l and θ.

$$Area(\triangle ATP) = \frac{1}{2}.h.\frac{h}{\tan\theta} \tag{2.6}$$

$$Area(\triangle TXB) = \frac{1}{2}s_2\left(l - \frac{h}{\tan\theta}\right)\sin\theta \tag{2.7}$$

Solving for s_2 using equations 2.5, 2.6 and 2.7

$$\begin{aligned} \frac{1}{2}\frac{h^2}{\tan\theta} &= \frac{1}{2}s_2\left(l - \frac{h}{\tan\theta}\right)\sin\theta \\ \implies s_2 &= \frac{h}{\sin\theta}.\frac{1}{(l/h)\tan\theta - 1} \end{aligned} \tag{2.8}$$

Finally PX is computed as

$$\begin{aligned} PX &= s_1 + s_2 \\ &= \frac{h}{\sin\theta}\left(1 + \frac{1}{(l/h)\tan\theta - 1}\right) \end{aligned} \tag{2.9}$$

Let l/h be denoted as the aspect ratio $\mathcal{AR}$

$$PX = \frac{h}{\sin\theta}\left(1 + \frac{1}{\mathcal{AR}\tan\theta - 1}\right) \tag{2.10}$$

If the lengths of the adjacent edges PQ and PR are different, we would want to consider both the aspect

ratios to calculate the length PX and then average it. This gives us

$$PX = \frac{h}{2\sin\theta}\left[2 + \frac{1}{\text{Æ}_1\tan\theta - 1} + \frac{1}{\text{Æ}_2\tan\theta - 1}\right] \tag{2.11}$$

And the extrusion factor f is simply PX/h

$$f = \frac{1}{2\sin\theta}\left[2 + \frac{1}{\text{Æ}_1\tan\theta - 1} + \frac{1}{\text{Æ}_2\tan\theta - 1}\right] \tag{2.12}$$

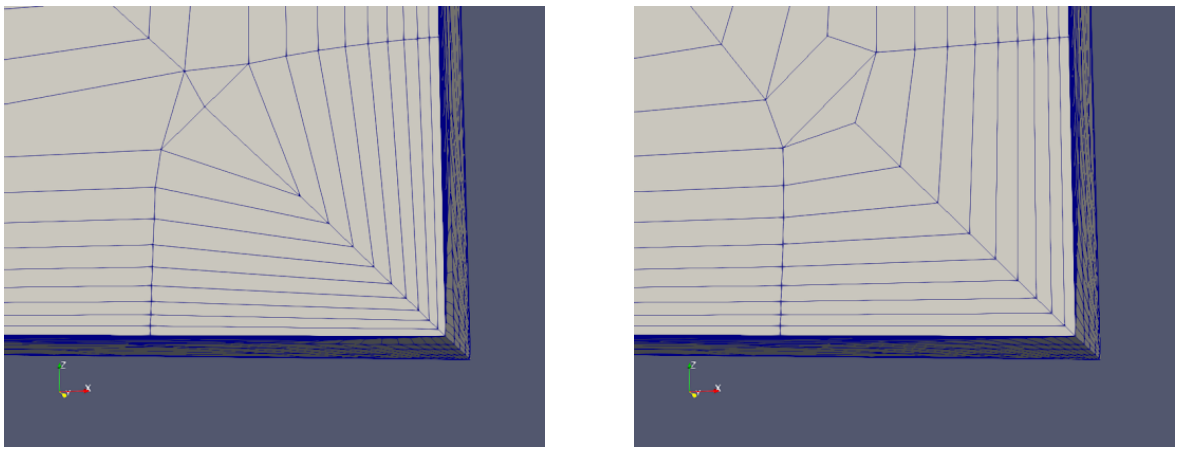
Hence, the extrusion factor f depends on the half included angle θ and the aspect ratios Æ_1 and Æ_2 at the vertex. We use a couple of constraints while using this extrusion factor. First, we ensure that $\text{Æ} > cot\theta$, so that we never get negative values for f. Secondly, we limit the extrusion length at any vertex to be less than the minimum of the length of adjacent edges on the boundary. This is done because the extrusion factor can be really large for small angles and small aspect ratios. Hence, to avoid generating skinny elements at concave corners of the mesh, we limit the extrusion length at any vertex such that the resultant aspect ratio at that vertex would always be more than 1. Figure 2.7 shows an example of scaling the extrusion length at the concave corner. Along with preventing layers to fold onto themselves, this technique also helps in improving the quality of quad cells generated in the mesh.

Extrusion length scaling works similar to the principle of hyperbolic meshing, where corners are raced towards the interior of the domain to make the successive layers more rounded. Immediate front collapses are avoided for a great extent for majority of cases by using such scaling. However, if the corners are quite sharp ($< 10^\circ$) or the refinement on the boundary curves is not sufficiently high, front collapses are inevitable. Hence, we implement a separate subroutine to handle such corner collisions in section *ref*.

2.4 Local Reconnection for Quality

A mesh observer object is initialized for the mesh data structure. The observer monitors all the changes happening to the mesh. These include the creation and deletion of vertices, edges and cells in the mesh. An edge swapping manager and quality criterion for deciding a swap are also initialized for the mesh. The primary quality criterion we chose for an edge swap is maximization of the minimum angle in the pair of triangles adjacent to the edge before and after the swap. However, only this criterion is not enough for improving surface mesh quality. This is because a better quality pair of triangles may result by edge swapping according to this criterion, however, they might deviate significantly from the underlying surface. Hence, we threshold the maximum deviation of the resultant triangles from the surface. This deviation is calculated as the angle between the normal to the triangle and the surface normal at the centriod of the triangle projected onto the surface.

Figure 2.5 illustrates the deviation of a triangle T formed as a result of vertex insertion from the un-



(a) **(b)**

Figure 2.7: Extrusion length scaling is illustrated using a concave corner. In (a), the layers are about to fold onto one another because of constant extrusion length at each layer. (b) shows the vertex at the corner having a larger scaled extrusion length so as to maintain higher quad quality and prevent layer collision.

derlying surface. Deviation of a triangle T with respect to a surface S is shown. P is the centroid of the triangle and P_s is the projection of the centroid onto the surface. PN is the normal to the triangle and P_sN_s is the normal to the surface at P_s. The interior angle θ is considered as the deviation of the triangle from the surface. An upper bound is set for θ to limit the deviation of resultant triangles from the surface. The upper bound is set to 30° for the surfaces meshes we are dealing with. This bound is sensitive to the refinement in the initial triangulation. A higher value of θ would be appropriate for a coarse initial triangulation while the value can be set lower for a fine initial triangulation.

We queue up the edges of the triangles affected by the insertion of the new point and do edge swapping for these edges to improve on the existing mesh. See Figure 2.4c and 2.4d for an illustration. One of the edges in the mesh is swapped to improve the quality of the triangles surrounding point Q.

Figure 2.8 shows the process of point insertion and local reconnection in two steps in an actual mesh generation process. First, a point is inserted into the mesh at the desired location. This results in subdivision of a triangle as can be seen in figure 2.8b. Subsequently, edge swaps occur to improve the mesh quality. Two iterations of edge swap can be seen in figure 2.8c and 2.8d. We do not swap the edges which comprise the front or the edge between parent and child points. Doing so would invalidate the advancing front. Hence, only interior edges of the surface or the diagonals of the quad elements generated are swapped by the face swapping algorithm. After edge swapping, the algorithm moves on to the next point in the current layer to repeat the same process.

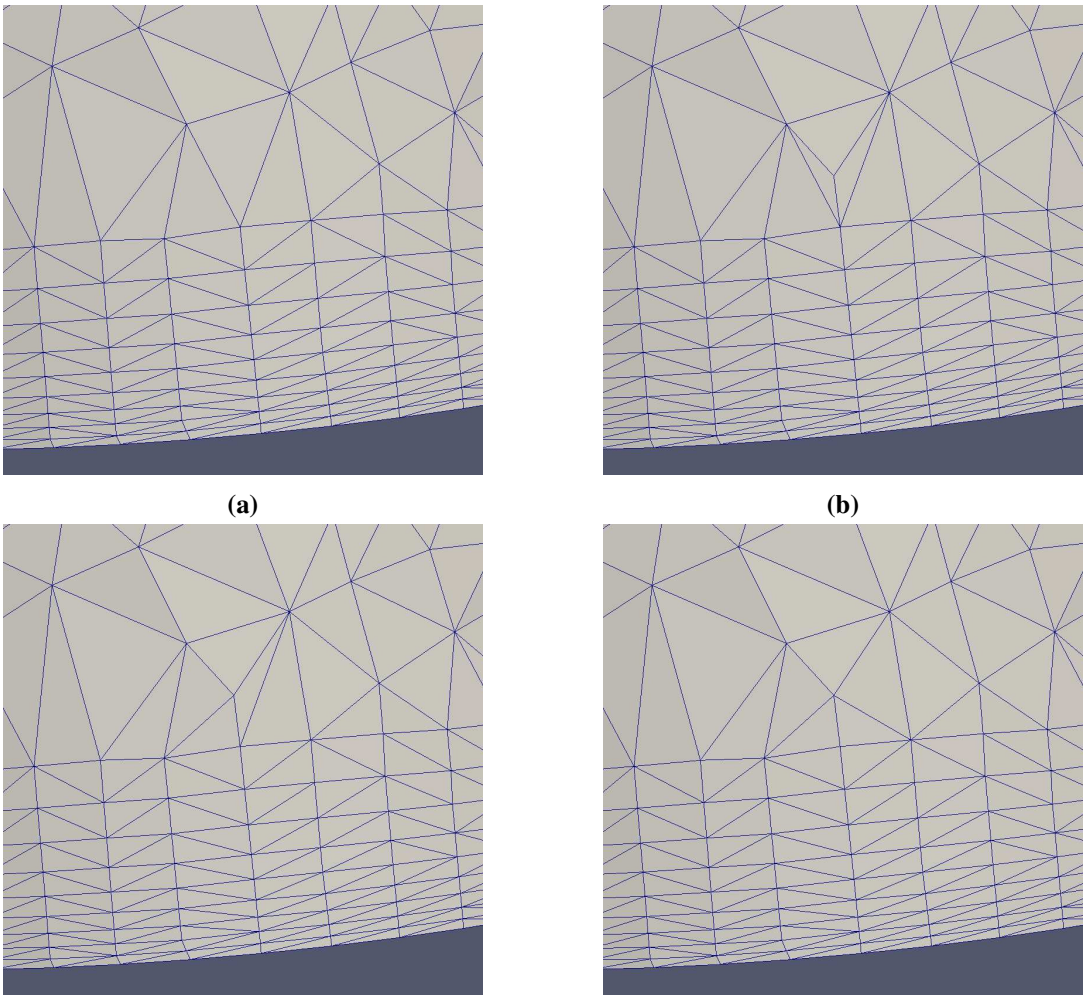


Figure 2.8: Point Insertion and local reconnection for quality shown in four steps. (a) is the initial state of the mesh. A point is inserted in the mesh after extrusion from the parent point, projection onto the surface and finding the right triangle to insert. The triangle is subdivided into three new triangles as shown in (b). To improve the mesh quality after point insertion, swapping is done based on maximization of the minimum angle in a triangle. Two swaps occur as shown in figure (c) and (d) to improve mesh quality.

2.5 Front Recovery

In an advancing layer mesh generation algorithm, an accurate representation of the boundary by anisotropic layers generated from it is very important. Also, generation of quad-elements from triangles with the right aspect ratios is far easier if we retain the boundary representation as we advance multiple layers. Hence, after all points of the parent layer are extruded, we implement a front recovery routine which recovers all edges between kid vertices.

Front recovery is done by selecting a pair of points from the parent layer and then forcing an edge

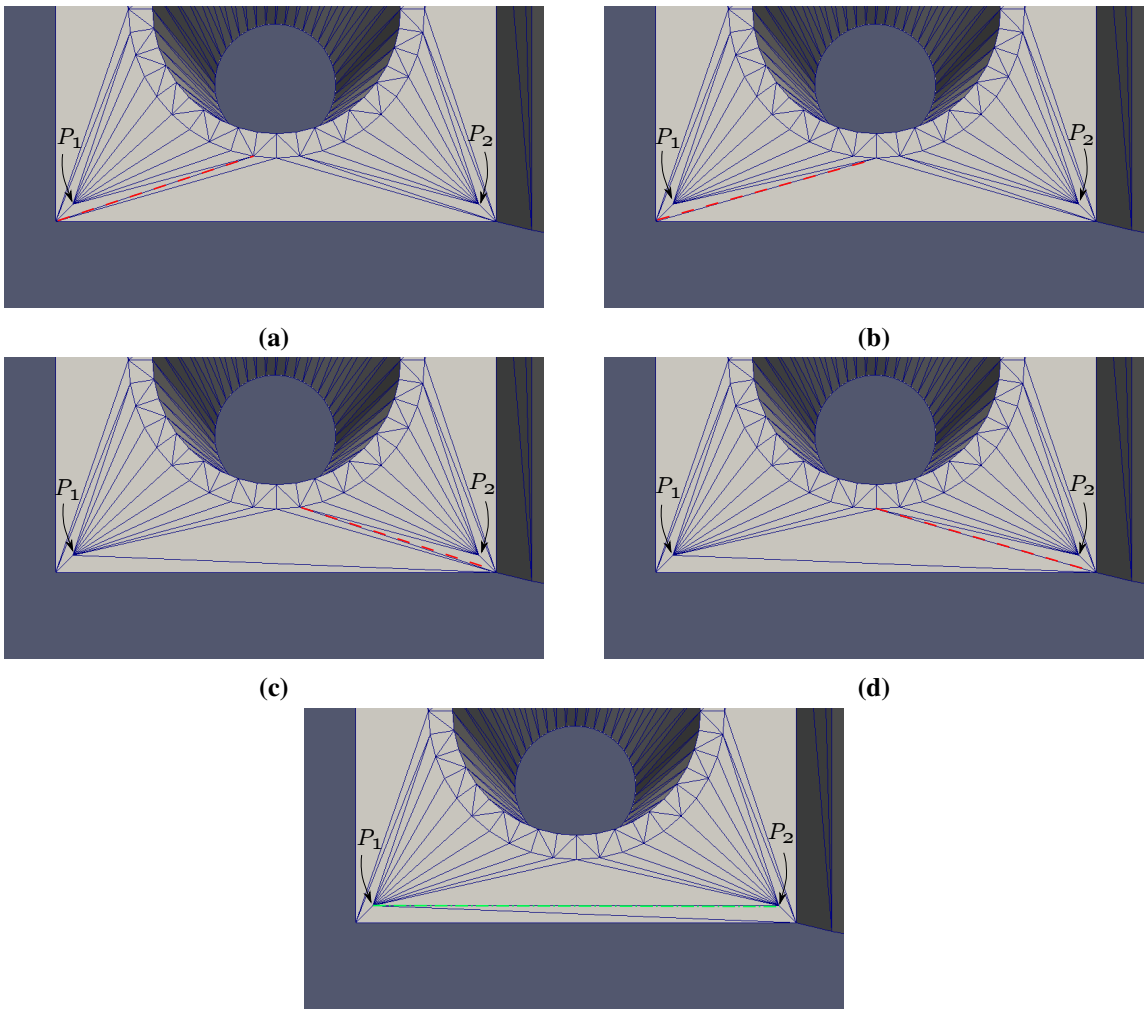


Figure 2.9: Front Recovery: Point P_1 and point P_2 here represent the two kid points generated from the boundary of the surface. We try to force a connection between the two points by iteratively swapping edges which topologically obstruct their connection. Red dashed lines represents the edge chosen to be swapped next. In (e), the green edge is the edge recovered and would serve as a part of the next front in the mesh. Note that this example is for front recovery illustration purposes and the initial boundary discretization is too coarse to get a good-quality advancing layer mesh.

between the points extruded from these parent points; that is by forcing an edge between the kid vertices in the extruded layer. An edge is forced between the kid vertices by iteratively swapping any edge that lies topologically in between the two kid points. This kind of forced swapping in the mesh might seem to decrease mesh quality. However, with the right boundary discretization, it helps in giving us the high-aspect ratio elements in the mesh that we desire.

Figure 2.9 shows how exactly we recover an edge in the front by iteratively swapping the edges which obstruct the edge creation between two kid vertices, P_1 and P_2 in the figure. The edges which are

topologically obstructing kid vertices P_1 and P_2 are swapped iteratively. A test is setup for each swap to avoid the flipping of triangles on the surface. After all the obstructing edges are swapped, a front edge between the kid vertices P_1 and P_2 would be recovered as shown in green in Figure 2.9e. After forcing an edge between all the adjacent kid vertices, validity checks are run to ensure that the extruded layer is well defined and there are no discrepancies in this layer as it will serve as the parent layer for the next one. This incorporates checking the connectivity of each vertex with its adjacent edges and the connectivity of each edge with its end points.

Chapter 3

Methodology Part 2: Advancing Several Layers

In the previous chapter, we talked about how the very first layer of the mesh is generated. Starting from the discussion of extruding one boundary vertex onto the sub-surface, we discussed how all the vertices on the boundary curve of the sub-surface are extruded and subsequently, how the advancing front is recovered so as to move on to the next layer. We also talked about how extrusion length scaling at concave corners helps us avoid immediate front collapse and improves overall mesh quality. In this chapter, we are going to discuss some of the important subroutines which help complete the anisotropic surface mesh.

We will start by discussing the subroutine used to control the aspect ratio as we advance several layers in the mesh. Combining triangular mesh elements to quad elements would be discussed next. Subsequently, mesh smoothing and collision handling would be discussed.

3.1 Aspect Ratio Control and Sub-surface Interior Improvement

3.1.1 Vertex Decimation on the Front

As the advancing front moves into the surface interior, the layers grow in size. This is done for the purpose of giving a higher refinement at the boundary curves. As the size of the layer grows, the aspect ratio of the mesh elements generated decrease. Also, some of the vertices on the front might come so close to each other that the aspect ratio approaches unity. Growing the layers further with all the vertices on the front would lead to anisotropy in the orthogonal direction and/or front overlap. Hence, decimation of some of the front vertices is necessary to proceed with the next few layers.

Once we have recovered the advancing front by iterative edge swaps to connect the kid points in the

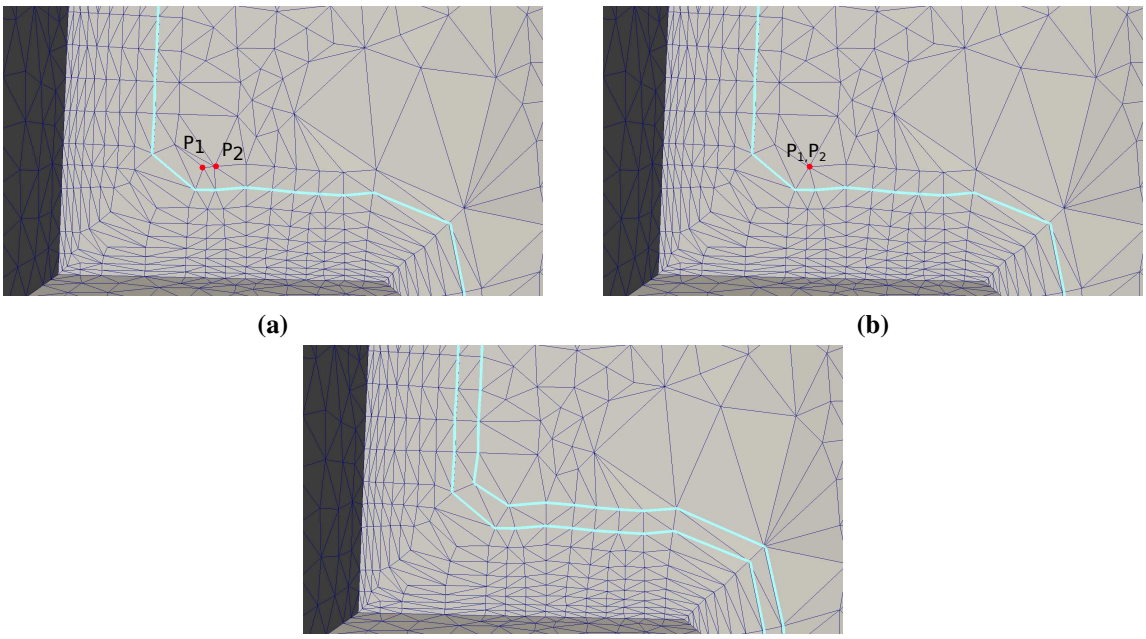


Figure 3.1: Edge collapse on an advancing front to avoid encroachment of points. In (a), two points in the kid layer P_1 and P_2 are sufficiently close to each other. Their parent layer is highlighted. If both the points advance to the next layer, then the next front would fail to recover. Hence, the edge between them is chosen to collapse. (b) shows the result of the edge collapse. The new location of both the points is the average position of their initial location. (c) shows how the next front looks like.

mesh, we check for vertices on the front that are too close to each other relative to the extrusion length. For instance, vertices which are near a concave corner could encroach each other if they are not decimated. Another example of a situation where vertex decimation on the advancing front becomes necessary is when the front has advanced to a substantial distance from the surface boundaries. In such a case, the extrusion length on the front has grown so much that the aspect ratio approaches unity. Decimating vertices from the front which are at a substantial distance from surface boundaries helps prevent the cell aspect ratio, ie front edge length over extrusion length, from dropping below one.

To check for vertices to decimate, we iterate through the vertices on the front and identify the ones which are too close to their neighbours. Vertex decimation is done through the conventional edge-collapse subroutine as described in [17]. The threshold edge length between two points on the front is set to be $2\tan(\pi/8)$ times the average extrusion length at those points. This value is set so as to minimize the normalized maximum deviation of angle from 90° for quad elements. Hence, the threshold ensures that the anisotropic properties are retained for several layers while marching onto the surface. All short edges on the front are collapsed using the edge-collapse algorithm. An example of an edge-collapse on the front is shown in Figure 3.1. Here, two points P_1 and P_2 are collapsed into a single point which forms a part of the next front on the surface. Mathematically,

$$l_{collapse} = 2 \cdot tan\left(\frac{pi}{8}\right) \cdot x_n \; ; \tag{3.1}$$

where $l_{collapse}$ is the minimum length of an edge on the advancing front which will not be considered for collapse and x_n is the extrusion length at the n^{th} layer.

Topological and geometrical checks are done before an edge can be collapsed in the mesh. These include a threshold for the ratio of area of generated triangles and a limit on the dihedral angle between the adjacent triangles created by the collapse. The area threshold is set to be 10^8. Also, edge-collapse is successful only if the triangles resulting from it are be within a limit of $\theta < 30^\circ$ from the surface (see Figure 2.5). The threshold for the maximum dihedral angle between any two adjacent triangles that result from the edge-collapse is set to 40°.

After we have a recovered the front and decimated encroaching vertices in the surface interior as well as on the advancing front through edge collapse, we queue up the immediate interior edges of the surface and swap them for maximizing mesh quality. This step is included so that we have a good interior triangulation at each step of the advancing layer routine. It ensures that we do not have skinny triangles ahead of the front which might cause problems as we continue to march.

The edge collapse subroutine on the advancing front helps in controlling the aspect ratio at the front and limit its minimum value to 1. Once the aspect ratio has approached a value of 1 on the front, layers will continue to extrude without growth in their size. Additionally, at concave corners with low aspect ratios, the edge collapse subroutine helps in maintaining a valid front to continue marching towards surface interior.

3.1.2 Vertex Decimation in Sub-Surface Interior

As discussed in the previous chapter, we follow a closed advancing front methodology to generate the anisotropic surface mesh. The input triangulation T, which is the encoding of the surface S is taken to be the initial mesh. Such a technique helps us in maintaining a complete and valid surface mesh at each step in the mesh generation process. However, this means that we need to do some additional work in deleting the vertices on the sub-surface imported initially which are no longer required to be a part of the anisotropic surface mesh. Fortunately, we have all the connectivity information of the vertices on the front. This means that we do not have to scan the 3D space near the vertex to find the encroaching points. Checking the vertices in surface interior which share an edge with the front vertex is sufficient.

In other words, as the advancing front marches onto the surface, vertices in the interior of the surface immediately next to the front are decimated to make way for the advancing layers. Before extruding a point P on the advancing front, we check if any point in the surface interior with which it shares an edge agrees with the following condition. If it does, we decimate the interior vertex.

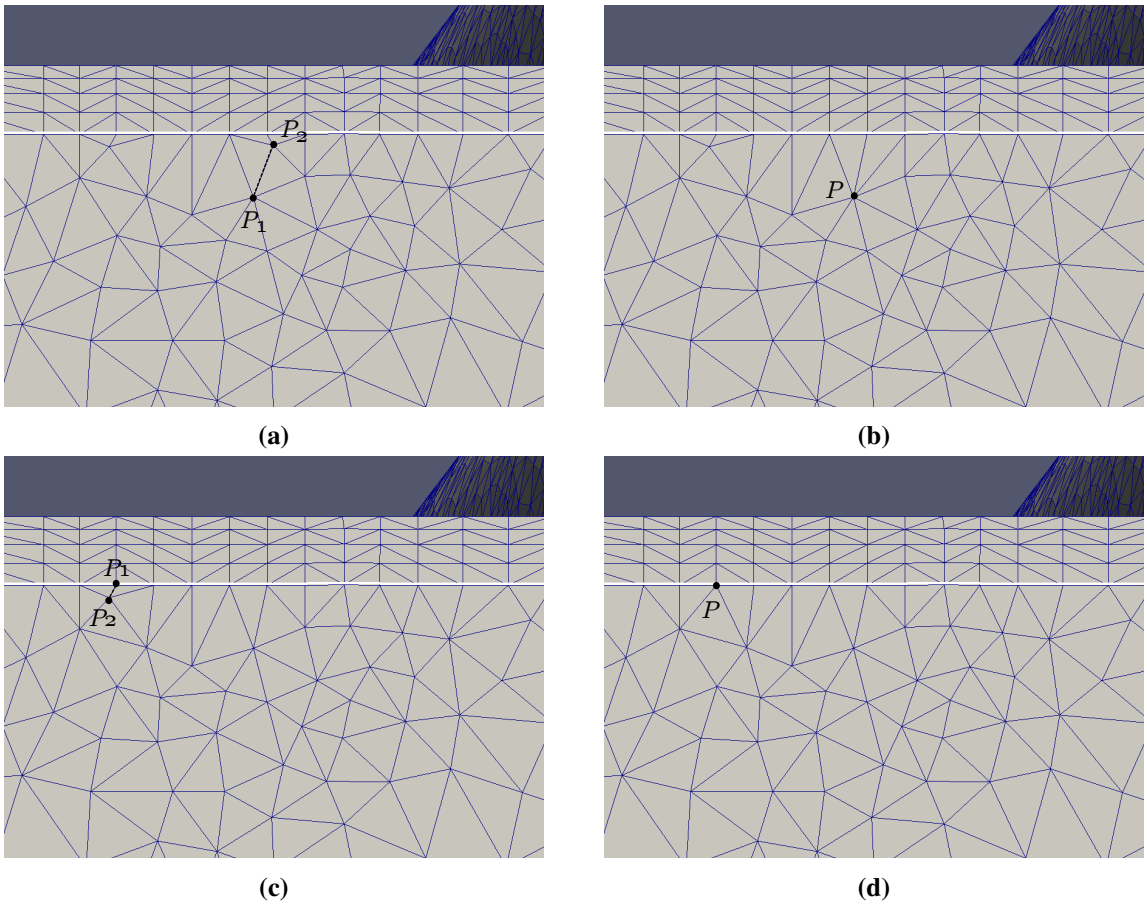


Figure 3.2: Interior vertex decimation through edge collapse. The highlighted white line shows the advancing front. In (a), vertex P_2 is about to encroach the front. Hence, the best vertex for collapse is chosen among its neighbours. The best vertex for edge collapse here is P_1. Hence, P_2 is collapsed on to P_1. The connectivity after the edge collapse is shown in (b) where vertex P represents the collapsed vertex. Similarly, in (c), vertex P_2 is about to encroach the advancing front and is collapsed onto vertex P_1 which is on the advancing front itself. The new connectivity is shown in (d) where all the possibly encroaching vertices for the advancing layer are decimated.

$$d < max\left(\frac{l_1}{\sqrt{2}}, \frac{l_2}{\sqrt{2}}, c \cdot x_n\right) \tag{3.2}$$

Here d is the distance between the point on the advancing front and the interior point, l_1 and l_2 are the lengths of adjacent front edges of the point on the front, c is a constant whose value is set as 2 and x_n is the extrusion length at the vertex on the n^{th} front. This condition ensures decimation of vertices in the surface interior which are close to the advancing front and avoids any encroachment of surface interior vertices on the advancing layers.

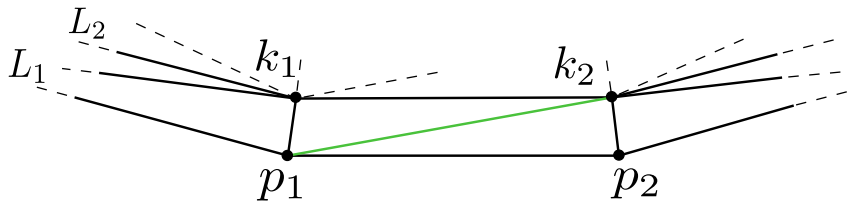
Validation checks for front edge collapse mentioned in the last subsection are also run during interior edge collapse. The quality criterion used for interior edge collapse is maximization of the minimum

angle in the triangles thus produced. The best edge for collapse is chosen when decimating the interior vertices using this quality criterion. This is in contrast to the vertex decimation on the advancing front where the candidate edge for collapse is already identified. An illustrative example for surface interior vertex decimation can be seen in Figure 3.2.

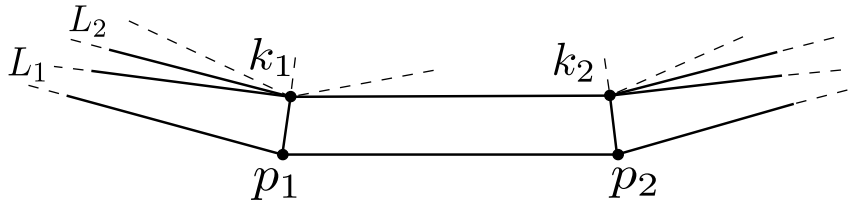
3.2 Combining Triangular Elements to Quadrilateral Elements

In section 1.3, we discussed about the pros and cons of simplical and non-simplical mesh elements. Then, in section 1.6.2, we talked about the motivation to produce a hybrid surface mesh which consists of both simplical and non-simplical mesh elements. Briefly, a hybrid anisotropic surface mesh which consists primarily of quadrilateral elements with a small number of triangular elements gives us the flexibility to mesh topologically complex surface geometries while keeping the average vertex connectivity of the mesh to a low value.

Almost all vertices in our closed advancing front method have a parent-kid relationship. The successor of a vertex is called its kid while the predecessor is called the parent. There are a couple of exceptions to this. First, vertices on the boundary curve do not have a parent as they are the zeroth layer of the mesh. Second, wherever an edge collapse operation is done, two kids from two different parent vertices collapse into a single vertex. This leaves a kid vertex with two parents, or, in other words, a two parents who share a common kid vertex.



(a)



(b)

Figure 3.3

Apart from the two exceptions just mentioned, all the vertices have a parent and a kid. This gives some structure to our mesh. As the layers grow in the normal direction to the boundary curve, this helps us in generating rectangular quadrilateral elements easily. Edges are removed from the finished layers so as to generate superior quality quadrilateral cells. As the layers advance from the boundary curves towards the tangential direction onto the surface, the quadrilateral elements generated retain the boundary topology

several layers into the interior of the surface. The quality of a quadrilateral element is calculated as the inverse of the maximum deviation of its interior angles from 90 degrees. This criterion is selected so as to prefer rectangular elements and retain the boundary representation on the surface mesh. Starting from the first layer generated on the surface, multiple iterations of this subroutine are run for the same layer to ensure as few triangular elements are left per layer as possible.

Consider Figure 3.3a. L_1 and L_2 denote two successive layers on the advancing layer mesh. Points p_1 and p_2 are on the parent layer L_1 and their kids, k_1 and k_2, respectively, are on the kid layer L_2. All the edges in the neighbourhood of p_1 and p_2 which are not on the advancing front are queued. Then, these edges are sorted in order of the quality of the quadrilateral elements generated after their removal. Finally, all the edges in the queue are deleted until both the elements which share the edge are triangular. The edged marked in green color in the figure is deleted in the process which results in the deletion of two triangular elements and generation of a new quadrilateral element. The local mesh after edge removal is shown in Figure 3.3b.

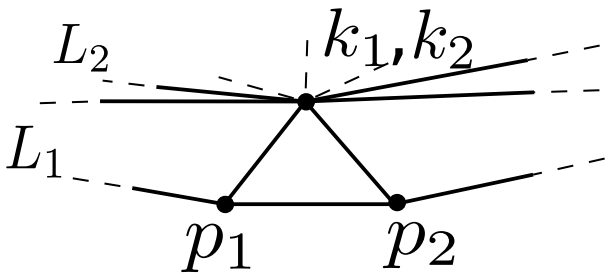
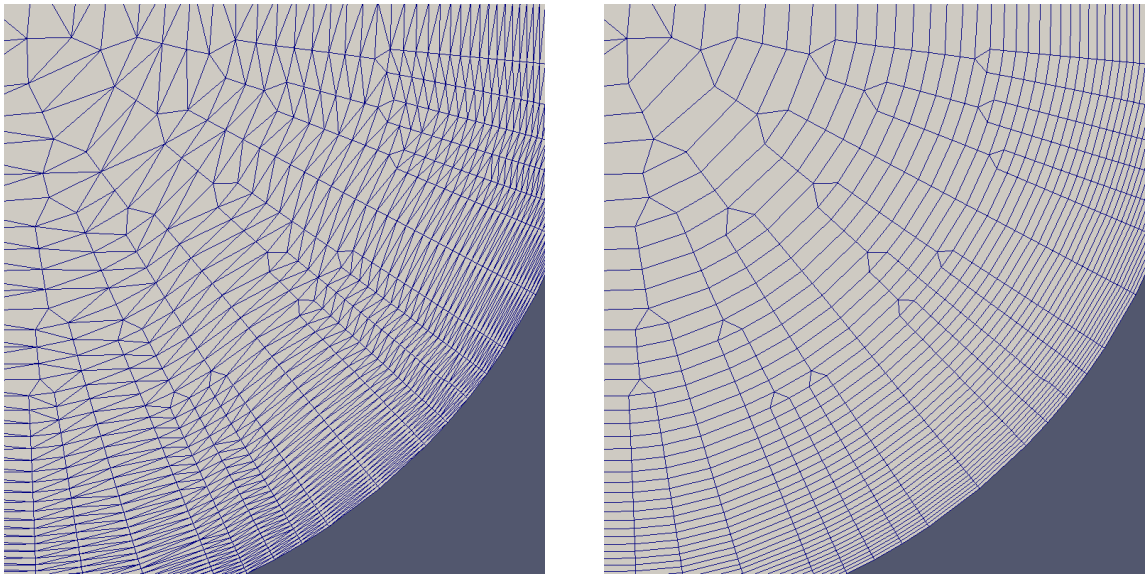


Figure 3.4:

As discussed earlier, some exceptions exist in the advancing layer mesh where all the elements in a layer cannot be converted to a quadrilateral element. An illustration of such an exception is shown in Figure 3.4. Here, two kids k_1 and k_2 had to collapse to a single vertex as the edge between them is collapsed in the aspect ratio control subroutine explained in section 3.1.1. Hence, a triangle would be left in between layers L_1 and $L_2 2$. An example surface mesh which shows the result of the combine triangular elements to quadrilateral elements subroutine can be seen in Figure 3.5. Most of the triangular mesh elements are converted to quadrilateral elements save a few.

3.3 Mesh Smoothing

Mesh smoothing is the process of improving the quality of the mesh. However, the context in which quality is referred to changes with the application of the mesh. Diffusion of mesh elements might be a desirable effect of mesh smoothing in one application and might not be a desirable effect in yet another application. One segment of the mesh smoothing techniques either proceed by filtering out noises of selected frequencies. Laplacian smoothing is the simplest example of such a smoothing technique. In a very simplistic Laplacian Smoothing technique, vertices on the mesh are moved towards the average



(a) **(b)**

Figure 3.5: Combining triangular cells to quadrilateral cells based on the quality of the quadrilateral cells generated. For any given quadrilateral element, the inverse of the maximum deviation of its interior angles from 90° is adopted as the measure of quality.

position of its neighbours. Another implementation considers moving the centroid of an element towards the average location of the centroids of the adjacent elements. In either of the cases, there is a filteration of high frequency undulations on the surface mesh. This causes loss of sharp features on the surface. Also, such a smoothing methodology results in shrinkage of the volume of the geometry which might not be desirable everywhere.

Another segment of mesh smoothing methodology falls under the category of optimization by minimizing a given energy or error function in the mesh [8, 13, 30, 34, 40]. Here, a

combined - [5, 12]

For our mesh generation scheme, we choose a physically-based smoothing technique. The mesh vertices (or nodes) on the mesh exert forces on adjacent nodes. Subsequently, the mesh nodes move so as to balance out these forces and reach a local force balance equilibrium. This method of smoothing is also referred to as spring-based smoothing methodology as each edge (or link) in the mesh acts as a spring which pushes or pulls its end points away or towards each other.

Choosing a spring-based smoothing methodology gives us freedom to select the proportion of forces applied in the extrusion direction and along the advancing layer direction. The proportion can be changed by changing the spring constant of the springs associated with the edges in either direction. Several spring forces are applied to a given mesh node in the smoothing procedure. Each spring constant is denoted by k_{spring}. The forces are:

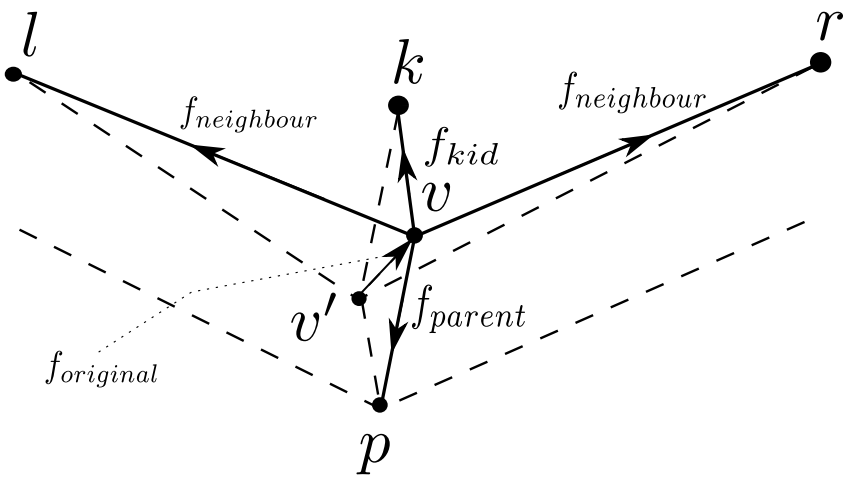


Figure 3.6: Smoothing Forces. Force k_{parent} keeps the distance between the parent vertex p and the current vertex v close to ideal. Force k_{kid} is a similar force which keeps the distance between the kid vertex k and v close to ideal. Force $f_{neighbour}$ pushes the vertex v towards the center of the left and right vertices, l and r, respectively.

- **Parent force** - force which keeps the distance of the vertex to its parent (from which it was extruded) closer to the initial extrusion length, l_{ideal} between the two.

$$f_{parent} = k_{extrusion} \cdot \frac{l_{real} - l_{ideal}}{l_{real} + l_{ideal}} \cdot \|\vec{p} - \vec{v}\| \tag{3.3}$$

- **Kid force** - similar to the parent force, this one keeps the distance between the node and its kid closer to the initial extrusion length.

$$f_{kid} = k_{extrusion} * \frac{l_{real} - l_{ideal}}{l_{real} + l_{ideal}} * \left\|\tilde{k} - \tilde{v}\right\| \tag{3.4}$$

- **Neighbour force** - force to maintain uniform spacing of mesh nodes for each layer.

$$f_{neighbour} = k_{neighbour} \cdot \frac{(\tilde{v} - \tilde{l}) + (\tilde{v} - \tilde{r})}{2.0} \tag{3.5}$$

- **Original location** - a force is added to restrict the movement of the vertex from the location at which it was placed in the mesh before smoothing. This term also ensures that the deviation of the mesh nodes from the initial surface representation is as little as possible.

$$f_{original} = k_{original} \cdot (\tilde{v} - \tilde{v'}) \tag{3.6}$$

These forces are summed up to give the mesh node a resultant force and the node is moved according to the total force. Several iterations (around 5-10) of smoothing are done per layer as the mesh advances

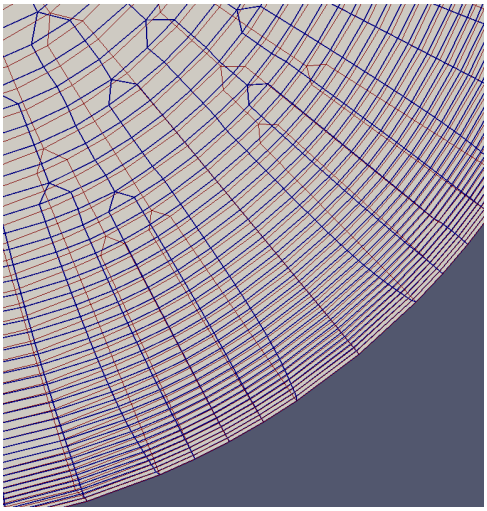


Figure 3.7: Zoomed in view of smoothing applied to an advancing layer surface mesh. Dark blue lines represent the smoothed mesh and flat red lines represent the mesh without smoothing. We can see that the smoothed version of the mesh helps to attain uniform aspect ratio at each layer.

towards surface interior. A limit is set to the vertex movement per smoothing iteration. The distance a vertex moves per iteration of the smoothing algorithm is kept at 5% of the extrusion length at that vertex. This helps in limiting the movement of the vertex when the initial boundary discretization is too coarse. The limitation of choosing a spring-based smoothing technique is that we have to carefully identify the spring constants, k_{spring} for our algorithm. However, once these constants are identified, they seem to work reasonably well for all the cases that we run. The value of $k_{extrusion}$ we use is 0.01, $k_{neighbour}$ is kept at 0.02 and $k_{original}$ at 0.1. Figure 3.7 shows an advancing layer surface mesh with and without smoothing applied to the mesh vertices. It can be seen that smoothing helps improve the vertex spacing along the advancing layer, thereby making the aspect ratio more uniform at a given front.

3.4 Collision Handling

In our surface mesh generation scheme, we take several measures to avoid or delay front collisions. Extrusion length scaling subroutine, described in Section 2.3.3, scales the extrusion length at a vertex on the front so as to output better quality quad cells at concave corners. This helps us delay front collisions at such sites in the mesh. Vertex decimation on the front, explained in section 3.1.1, along with controlling aspect ratio, also helps avoid front collapse when two front vertices come sufficiently close to each other. Mesh smoothing, described in the last section also helps in avoiding front collapses to some extent. Inspite of undertaking such measures, front collapses at sharp concave corners and head on front collapses at mesh closure zones (where the fronts from two different regions of a sub-surface collide) are inevitable. Hence, subroutines for handling front collisions are important to get a complete surface mesh.

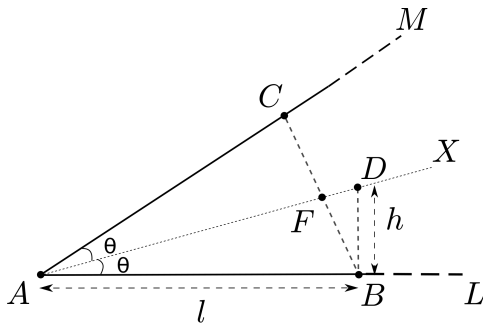


Figure 3.8: Handling collision at concave corners to avoid layer overlap. Corner point A is replaced by point F so as to form the new front CFB. F is the surface-projected mid-point of points B and C. This corner collision handling is done until all the half-interior angles, θ are within the limit described by equation 3.7.

The ways different advancing fronts in a surface mesh can collide can be classified into two categories. The first one is front collision near a sharp corner. Here, the advancing fronts from either sides of the sharp corner can collapse onto each other before even marching a single step. In the next section, we will discuss how the front collisions are handled in such cases. A second way in which fronts can collide is when two fronts from different regions of a sub-surface march until the point where they come sufficiently close to each other. If the fronts keep marching, front collision is inevitable and there needs to be a subroutine to stop the vertices which are in close proximity to other vertices on the front from marching further.

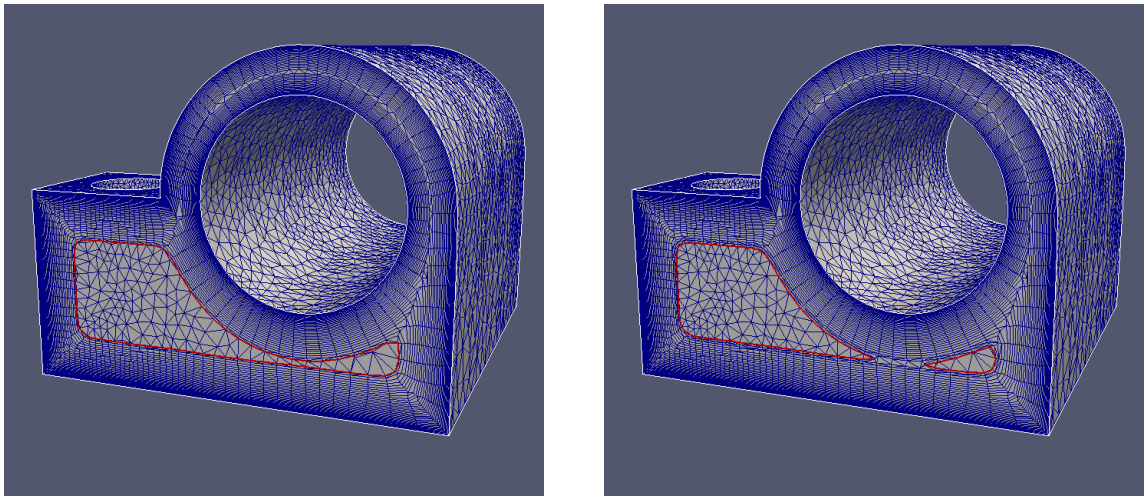
3.4.1 Collisions at Concave Corners

For concave corners of the mesh which might collide onto themselves, we preemptively close off the corner before the layers collide. Consider a concave corner as shown in figure 3.8. Point A is a corner vertex at the mesh front CAB. The extrusion length at the front is h. The length of the edge AB is l. If point B is extruded with the current front definition, the extruded point D would cross the angle bisector of $\angle BAC$, AX and will invalidate the front. This is likely to occur for smaller aspect ratio ($\mathcal{AR}$) values and small values of included angle 2θ. To avoid collision in the next front, we insert a new point F into the mesh, located at the surface projection of the mid-point of B and C. F takes the place of A as a front vertex and this procedure is repeated until no point which is extruded from the front crosses the angle bisector of its neighbours. The value of angle θ where this occurs can be written as $\theta = \tan^{-1}(1/\mathcal{AR})$, where $\mathcal{AR}$ is the aspect ratio at the corner vertex. Hence, we identify corner as any vertex on the front which has half interior angle θ less than a threshold times this limit.

$$\theta_{corner} < threshold * \tan^{-1}(1/\mathcal{AR}) \tag{3.7}$$

The threshold is taken as 1.5 for our mesh generation method to resolve all corners collision instances.

3.4.2 Head-On Collisions



(a) **(b)**

Figure 3.9: Handling front collisions. (a) shows the advancing front which is about to collide onto itself. Using the vertex connectivity at the advancing layer and the proximity of the advancing layer vertices from each other, some of the front vertices are removed from the front and the front is redefined. The redefined front extruded to the next layer is shown in (b). From this iteration of the front, it would proceed with only the vertices on the front shown in red.

Apart from handling collisions at the corners of the mesh, we also need to take care of the front collisions which happen elsewhere. Consider Figure 3.9a where the advancing layer surface mesh generation algorithm is meshing one of the surfaces of the geometry shown in Figure 2.3 . The advancing front is shown in red in the figure. After advancing several layers from the boundary, it reaches a point where it is about to collapse onto itself. Hence, some sort of front redefinition is necessary before moving on to the next layer. To achieve this, we first iterate through all the points in the front to identify the encroached points. These points are the ones whose distance from any edge in the front, except its adjacent edges, is less than a threshold (taken as 2.4) times the extrusion length at that point. As our surface mesh is closed, we can utilize the vertex connectivity to find out the possible vertex-front edge encroachment. Hence, for a given point on the advancing front, only the edges on the front that share a triangle with that point are checked for encroachment. This saves a lot of computational cost while identifying encroached vertices. We assume that the input boundary discretization and surface triangulation are fine enough to resolve the complex features of the geometry, so that we don't have to iterate through all the edges on the front to find whether a given point is encroached or not.

After these encroached points and their corresponding edges are identified, we reconnect the non-encroached points so as to form the new front. The encroached points are moved out of the advancing front and the front is hence redefined. In figure 3.9b, we show the redefined front extruded to the next layer. In this case, the front is divided into two loops, each of which continues to march independently towards the interior of the surface. After dealing with the concave corner collision cases, this approach

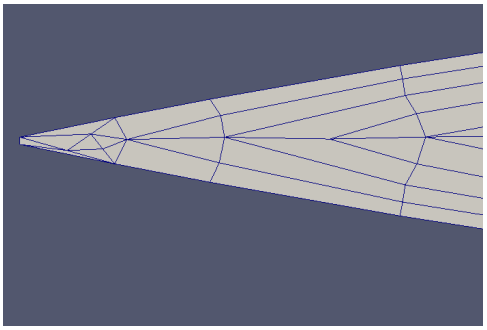


Figure 3.10: Limitation of the concave corner collision detection and front redefinition subroutine. Zoomed in view of the trailing edge of an airfoil is shown. The subroutine successfully closes off the two concave corners present a the tip of the trailing edge, but produces a couple of skinny triangles in the mesh with non-ideal vertex placement.

helps us tackle front collisions at other regions of the front and completes the front redefinition process wherever the advancing front might collide onto itself.

The concave corner detection and front redefinition routine associated with it works well to close off the corners. However, the vertex placement at such concave corners is not ideal every time. It tends to produce skinny triangular elements if there are multiple corners adjacent to each other because of bad boundary discretization. Hence, this is a limitation of the current formulation. Figure 3.10 shows such a case. A zoomed in view of the trailing edge of an airfoil is shown. Two sharp corners at the boundary of the surface are successfully closed by the mesh and the front continues to advance towards the interior of the surface. However, the quality of the triangular elements generated is not ideal. Better vertex placement and a different smoothing subroutine for such vertices could produce better quality mesh elements for such cases.

3.4.3 Overall Mesh Generation Algorithm

In the previous section, we have detailed the steps in the advancing layer mesh generation routine. Here, we summarize the algorithm in the form of a pseudo code in Algorithm 1.

3.5 Assumptions

We make a few assumptions on the input triangulation for our mesh generation algorithm to successfully create a valid mesh. The first one is that the triangulation of the surface is fine enough to resolve the geometric complexities in the object. If the input triangulation is too coarse, the encroaching interior vertex deletion subroutine will create mesh elements that deviate significantly from the surface or even fail. Hence, for highly curved regions of the surface, the input triangulation should be within 30 degrees deviation from the surface. However, this issue can be tackled by refining the triangulation using any isotropic refinement algorithm as long as it retains the features of the input object. Coarse triangulations

Algorithm 1 Overall Mesh Generation algorithm

```
1:  procedure SURFMESH::CREATEMESH(triangulation T, extrusion length, growth ratio)
2:      Create surface geometry S ← T
3:      Initialize Advancing Layer F from Surface Boundary
4:      while F ≠ ∅ do                          ▷ Advance until Front F isn't empty
5:          ADVANCELAYER(F)
6:          Validate the new Layer
7:      Export SURFMESH
8:  procedure SURFMESH::ADVANCELAYER(F)
9:      Delete Encroaching Interior Vertices
10:     Redefine front F Until No Vertex Encroachment
11:     Extrude All Points of F and Insert Into Mesh
12:     Recover front F'          ▷ Through forced edge creation between kid vertices
13:     Improve Mesh Quality by Edge Swapping          ▷ In the immediate interior of mesh
14:     Collapse Short Edges in the Front
15:     Combine triangular elements to quads
16:     Smooth Vertices
17:     Update Extrusion Length for vertices
```

are acceptable for objects without any geometric complexities.

Another assumption we have made for the input triangulation is that it would have well defined boundaries where the advancing layers can march out from. Sharp corners in the input geometry are automatically identified as boundaries of the mesh but the boundary discretization of blunt corners needs to be supplied as an input to the advancing layer surface mesh generator.

Bibliography

[1] 2019 most common 3d file formats. https://all3dp.com/3d-file-format-3d-files-3d-printer-3d-cad-vrml-stl-obj/. Accessed: 2020-01-12. → pages x, 24

[2] Mesh generation software list. http://www.robertschneiders.de/meshgeneration/software.html/. Accessed: 2020-02-27. → page 24

[3] M. Aftosmis, D. Gaitonde, and T. S. Tavares. On the accuracy, stability, and monotonicity of various reconstruction algorithms for unstructured meshes. In *32nd Aerospace Sciences Meeting and Exhibit*. American Institute of Aeronautics and Astronautics, 1994. → pages 7, 17

[4] T. D. Blacker and M. B. Stephenson. Paving: A new approach to automated quadrilateral mesh generation. *International Journal for Numerical Methods in Engineering*, 32(4):811–847, 1991. → pages ix, 7, 8

[5] S. A. Canann, J. R. Tristano, M. L. Staten, et al. An approach to combined laplacian and optimization-based smoothing for triangular, quadrilateral, and quad-dominant meshes. In *IMR*, pages 479–494. Citeseer, 1998. → page 44

[6] M. Castro-Díaz, F. Hecht, B. Mohammadi, and O. Pironneau. Anisotropic unstructured mesh adaption for flow simulations. *International Journal for Numerical Methods in Fluids*, 25(4): 475–491, 1997. → pages ix, 11, 12

[7] H. Chen and J. Bishop. Delaunay triangulation for curved surfaces. *6th International Meshing Roundtable*, pages 115–127, 1997. → page 16

[8] L. Chen. Mesh smoothing schemes based on optimal delaunay triangulations. In *IMR*, pages 109–120. Citeseer, 2004. → page 44

[9] B. O. Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. URL http://www.blender.org. → page 24

[10] J.-C. Cuillière. An adaptive method for the automatic triangulation of 3d parametric surfaces. *Computer-Aided Design*, 30(2):139–149, 1998. → page 16

[11] E. F. D'Azevedo and R. B. Simpson. On optimal triangular meshes for minimizing the gradient error. *Numerische Mathematik*, 59(1):321–348, 1991. → page 4

[12] L. A. Freitag. On combining laplacian and optimization-based mesh smoothing techniques. Technical report, Argonne National Lab., IL (United States), 1997. → page 44

[13] L. A. Freitag and C. Ollivier-Gooch. Tetrahedral mesh improvement using swapping and smoothing. *International Journal for Numerical Methods in Engineering*, 40(21):3979–4002, 1997. → page 44

[14] P.-J. Frey and F. Alauzet. Anisotropic mesh adaptation for cfd computations. *Computer Methods in Applied Mechanics and Engineering*, 194(48-49):5068–5082, 2005. → page 9

[15] R. V. Garimella and M. S. Shephard. Boundary layer mesh generation for viscous flow simulations. *International Journal for Numerical Methods in Engineering*, 49(1-2):193–218, 2000. → pages ix, 12, 13

[16] P. George and H. Borouchaki. *Delaunay Triangulation and Meshing: Application to Finite Elements*. Hermès, 1998. ISBN 9782866016920. URL https://books.google.ca/books?id=HZGfl61PSUQC. → page 16

[17] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. Technical report, Washington Univ Seattle Dept OF Computer Science And Engineering, 1994. → page 39

[18] Y. Ito and K. Nakahashi. Unstructured mesh generation for viscous flow computations. In *11th International Meshing Roundtable*, pages 367–377, 2002. → pages ix, 12, 13

[19] Y. Ito, A. M. Shih, B. K. Soni, and K. Nakahashi. Multiple marching direction approach to generate high quality hybrid meshes. *AIAA Journal*, 45(1):162–167, 2007. → pages ix, 15

[20] G. Kunert. Toward anisotropic mesh construction and error estimation in the finite element method. *Numerical Methods for Partial Differential Equations: An International Journal*, 18(5): 625–648, 2002. → page 11

[21] T. Lan and S. Lo. Finite element mesh generation over analytical curved surfaces. *Computers & Structures*, 59(2):301–309, 1996. → pages x, 16, 18

[22] Y. Lee and C. K. Lee. A new indirect anisotropic quadrilateral mesh generation scheme with enhanced local mesh smoothing procedures. *International Journal for Numerical Methods in Engineering*, 58(2):277–300, 2003. → page 12

[23] X. Li and W. Huang. An anisotropic mesh adaptation method for the finite element solution of heterogeneous anisotropic diffusion problems. *Journal of Computational Physics*, 229(21): 8072–8094, 2010. → pages ix, 11, 12

[24] R. Löhner. Matching semi-structured and unstructured grids for navier-stokes calculations. In *11th AIAA Computational Fluid Dynamics Conference*, page 3348, 1993. → pages 11, 12

[25] A. Loseille and R. Löhner. On 3d anisotropic local remeshing for surface, volume and boundary layers. In *Proceedings of the 18th International Meshing Roundtable*, pages 611–630. Springer, 2009. → page 13

[26] D. Mavriplis. Unstructured grid techniques. *Annual Review of Fluid Mechanics*, 29(1):473–514, 1997. → pages 7, 17

[27] D. J. Mavriplis. Adaptive mesh generation for viscous flows using triangulation. *Journal of Computational Physics*, 90(2):271–291, 1990. → pages ix, 10, 11

[28] K. Nakahashi. FDM-FEM zonal approach for viscous flow computations over multiple-bodies. In *25th AIAA Aerospace Sciences Meeting*, page 604, 1987. → pages 10, 12

[29] S. J. Owen. A survey of unstructured mesh generation technology. In *Proceedings of the 7th International Meshing Roundtable, 1998*, pages 239–267, 1998. → page 16

[30] V. Parthasarathy and S. Kodiyalam. A constrained optimization approach to finite element mesh smoothing. *Finite Elements in Analysis and Design*, 9(4):309–320, 1991. → page 44

[31] N. M. Patrikalakis and T. Maekawa. *Shape Interrogation for Computer Aided Design and Manufacturing*. Springer Science & Business Media, 2009. → page 23

[32] S. Pirzadeh. Unstructured viscous grid generation by the advancing-layers method. *AIAA Journal*, 32(8):1735–1737, 1994. → page 11

[33] O. Sahni, K. E. Jansen, M. S. Shephard, C. A. Taylor, and M. W. Beall. Adaptive boundary layer meshing for viscous flow simulations. *Engineering with Computers*, 24(3):267, 2008. → pages ix, 13, 14

[34] 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. → page 44

[35] K. Shimada, A. Yamada, T. Itoh, et al. Anisotropic triangular meshing of parametric surfaces via close packing of ellipsoidal bubbles. In *6th International Meshing Roundtable*, pages 375–390, 1997. → page 11

[36] J. R. Tristano, S. J. Owen, and S. A. Canann. Advancing front surface mesh generation in parametric space using a riemannian surface definition. In *Proceedings of the 7th International Meshing Roundtable, 1998*, pages 429–445, 1998. → page 16

[37] J. Tu, G.-H. Yeoh, and C. Liu. Chapter 6 - Practical Guidelines for cfd simulation and analysis. In J. Tu, G.-H. Yeoh, and C. Liu, editors, *Computational Fluid Dynamics (Second Edition)*, pages 219 – 273. Butterworth-Heinemann, second edition edition, 2013. ISBN 978-0-08-098243-4. doi:https://doi.org/10.1016/B978-0-08-098243-4.00006-8. URL http://www.sciencedirect.com/science/article/pii/B9780080982434000068. → pages 4, 10

[38] N. VISWANATH. Quadrilateral meshing with anisotropy and directionality control via close packing of rectangular cells. *Proceedings of 9th International Meshing Roundtable, 2000*, 2000. → pages ix, 12

[39] S. Yamakawa and K. Shimada. High quality anisotropic tetrahedral mesh generation via ellipsoidal bubble packing. In *Proceedings of the 9th International Meshing Roundtable, 2000*, pages 263–274, 2000. → pages ix, 13, 14

[40] T. Zhou and K. Shimada. An angle-based approach to two-dimensional mesh smoothing. *IMR*, 2000:373–384, 2000. → page 44

[41] J. Zhu, O. Zienkiewicz, E. Hinton, and J. Wu. A new approach to the development of automatic quadrilateral mesh generation. *International Journal for Numerical Methods in Engineering*, 32 (4):849–866, 1991. → page 7

Appendix A

Supporting Materials