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

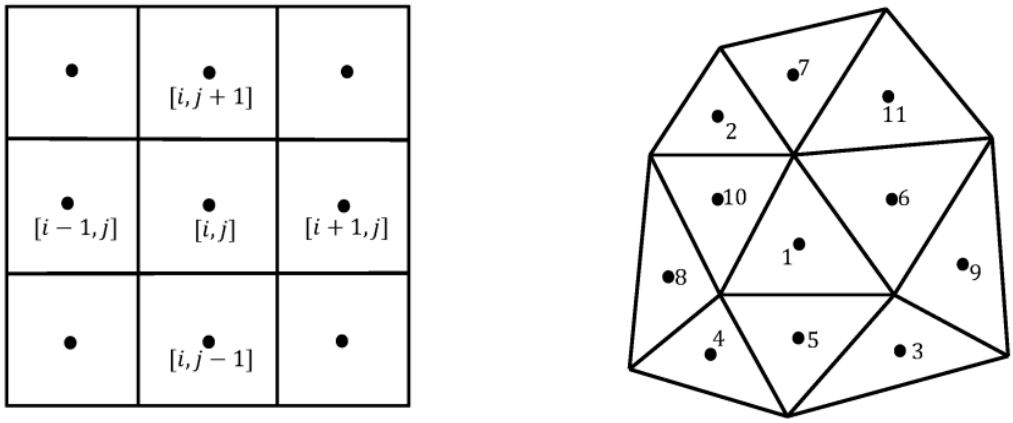
If I have seen farther it is by standing on the shoulders of Giants. — Sir Isaac Newton (1855)

Computational Fluid Simulations (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 Transform (DFT). However, since the development and advancement of computers, engineers had a significant amount of compute power to work with. This lead 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 also called 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 given 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.

The process of discretization of the domain to form the basis of solving the Navier-Stokes equations, or any other differential equation numerically is called mesh generation. Save a few exotic methods, almost all of the techniques in CFD require a mesh to solve the flow on. Traditionally, mesh generation was a very manual process, where engineers used to place the mesh points and cells by hand. Such heuristic approach to mesh generation gave them a lot of freedom in discretizing the domain. Cells could be



(a)

(b) fig-unstructured-ij

Figure 1.1

aligned to the boundaries of 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 without neither overlap nor hole.*

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) carryinig 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

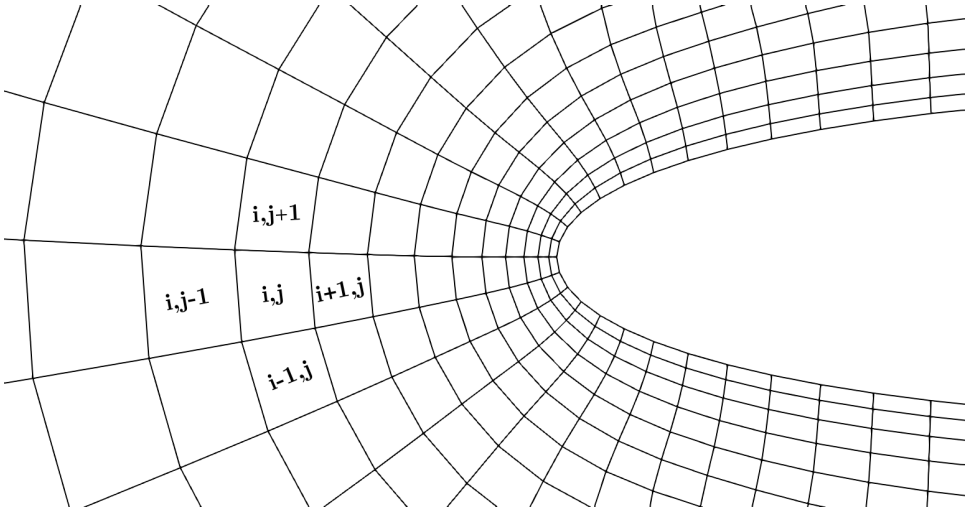


Figure 1.2: Structured mesh around leading edge of a NACA 0012 airfoil

the cells is repeated. Also, given simple domains to mesh, these meshes were optimal for minimizing the errors in CFD, resulting in faster simulations [7]. 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 [28]. The transformed equations that accommodate the 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 augmenting the cost of numerical calculations and difficulties in programming. Hence, 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 random. Along with the shape of the elements, we need a data structure to store the adjacencies of the mesh.

The cost of finding flux at a wall, 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 topology and/or the solution gradients. Figure 1.3 shows an unstructured mesh at the leading edge of a NACA 0012 airfoil. Notice the random arrangement of triangles around the airfoil geometry.

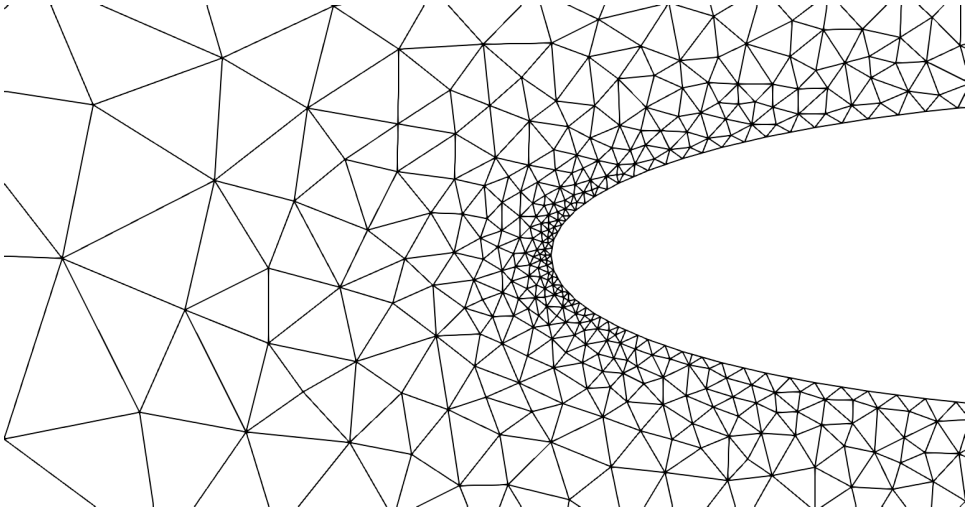


Figure 1.3: Unstructured mesh around leading edge of NACA 0012 airfoil

The connectivity at each vertex of the mesh needs to be stored separately.

1.3 Simplicial and Non-Simplicial Meshes

Before discussing about 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 simplex from K is also in K b) The intersection of any two simplices S1 and S2 is either ϕ (null set) or a face 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. 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,*

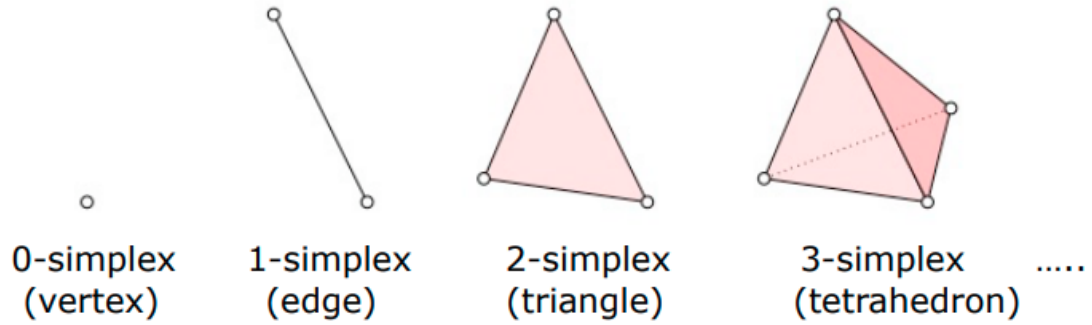


Figure 1.4: *n* dimesnional simplices.

together with a homeomorphism h: $K \to X$.

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

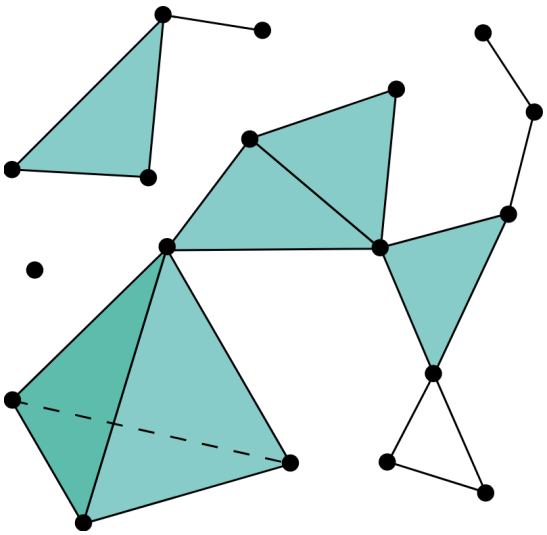


Figure 1.5: A three-dimensional simplicial complex.

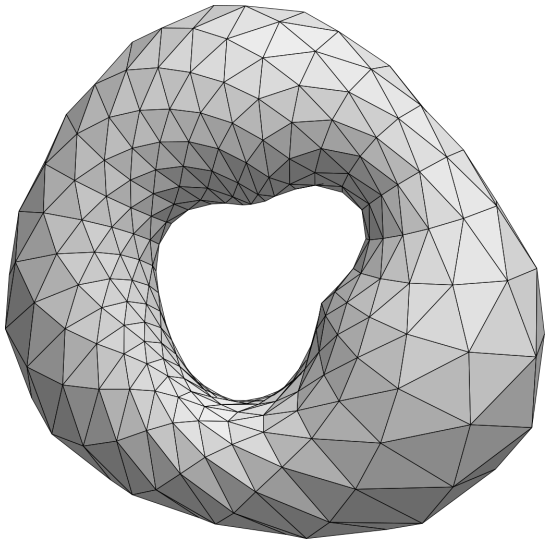


Figure 1.6: Triangulation of a torus.

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

Hence, the number of edges is three times the number of vertices in a triangulation (asymptotically). 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 cost when refining the mesh using a 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 [19]. In two-dimensions, quadrilateral elements have been preferred over triangles in highly stretched two-dimensional grids due to their lower connectivity [2]. These advantages of non-simplical mesh elements have resulted in fully non-simplicial mesh generation techniques [3, 31]. The scheme introduced by Blacker *et al.* [3] uses a paving methodology with several mesh element collision checks and special conditions for the concave corners to generated an isotropic all-quad two-dimensional mesh for complex

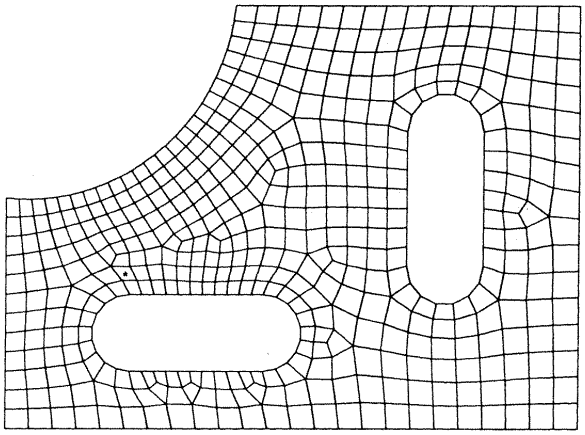


Figure 1.7: A non-simplicial quad mesh generated with paving methodology [3].

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 cons. It is significantly difficult to generate an automatic mesh generation strategy which creates non-simplicial mesh 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 would revisit this in section 1.6 where we reason about choosing a hybrid scheme to mesh the surface.

1.4 Boundary Layer Meshes

With the advent of several 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 were more 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 0.95 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.

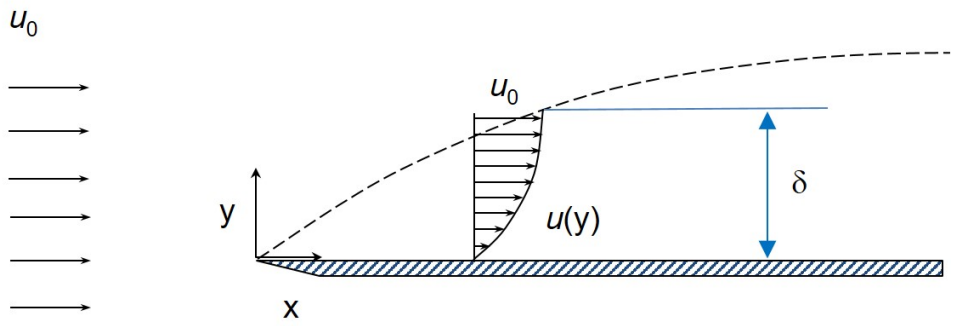
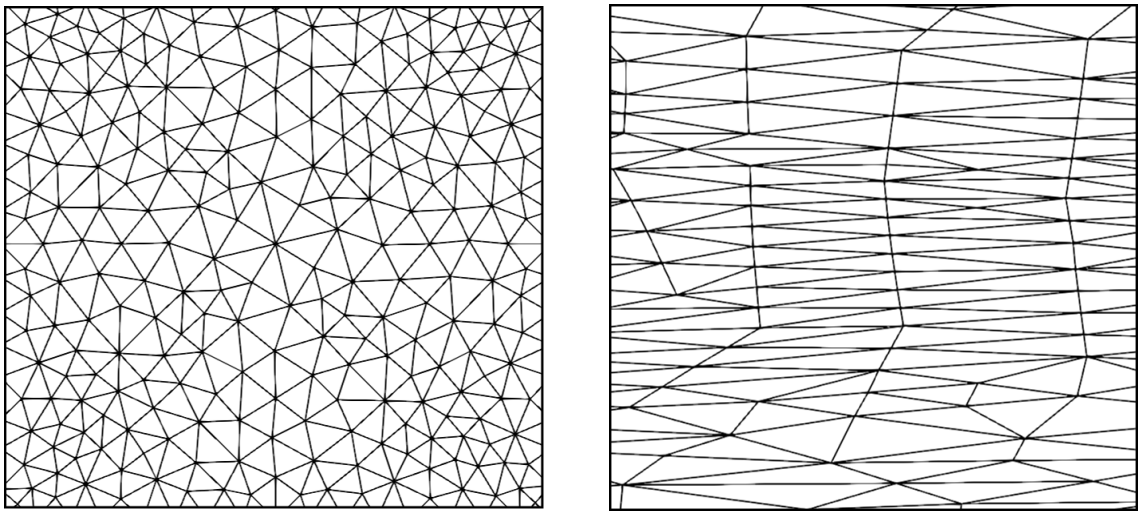


Figure 1.8: Fluid flow over a flat plate.



(a) Isotropic mesh

(b) Anisotropic Mesh

Figure 1.9: Isotropic and Anisotropic Mesh Fragments

1.5 Anisotropic Meshing

The need to resolve the steep gradients at 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 really 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 1.10 illustrates some different aspect ratio triangular and quadrilateral elements.

To attain a highly anisotropic packing of the cell elements, it is required to provide 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 [8].

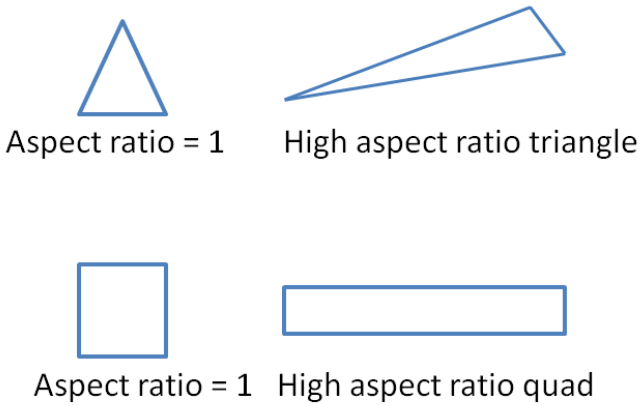


Figure 1.10: Illustration of different aspect ratio triangular and quadrilateral elements.

Figure 1.9a 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 1.9b 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 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 [28]. 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.

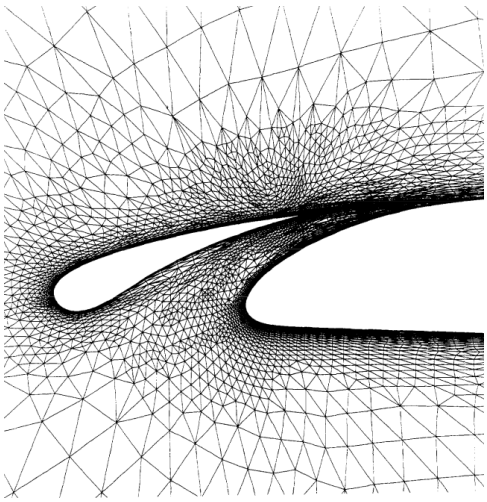
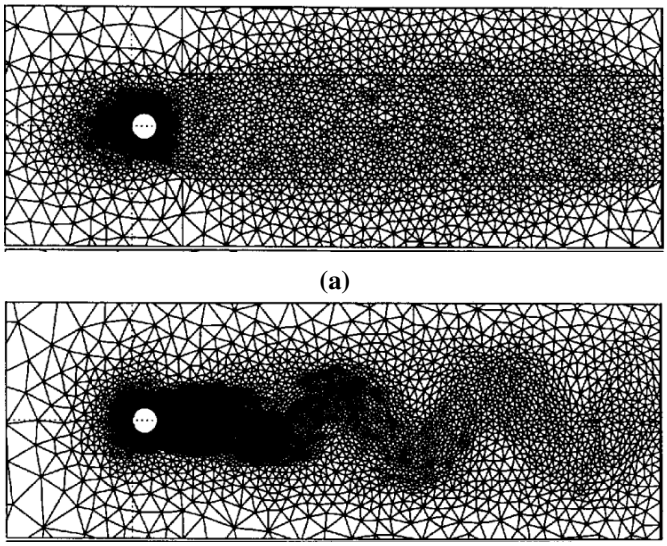


Figure 1.11: Illustration of adaptively refined mesh for the two-element airfoil configuration near the gap region [20].



(a)

(b)

Figure 1.12: Initial mesh (a) and adaptively generated anisotropic mesh (b) using solution metric evaluation [4].

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 [20]. 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 [21].

Many attempts at generating anisotropic meshes come under the category of metric adaptation of the mesh. These techniques generally used a Delaunay type initial mesh and refined 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 node-spacing, a proximity-based interacting force field is defined and the force balance configuration is dynamically simulated [26]. Castro *et al.* applied mesh adaptation technique to generate anisotropic meshes, to compressible viscous flows for a wide range of Reynolds and Mach numbers [4]. 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 [13]. 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 [16]. Figure 1.13 shows one such mesh.

A family of anisotropic mesh generation techniques fall under the Advancing Layer category. A method

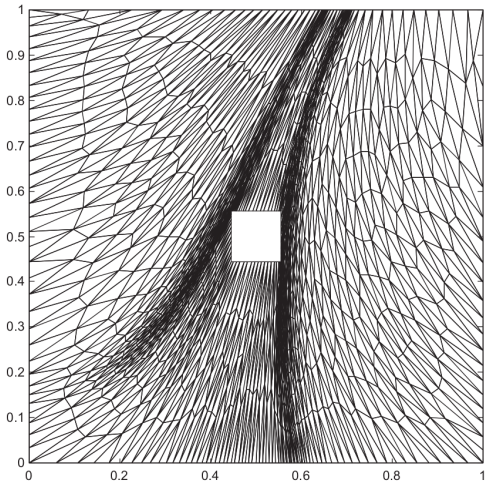


Figure 1.13: Anisotropic mesh generated by aligning the mesh elements to a metric calculated from an isotropic mesh solution [16].

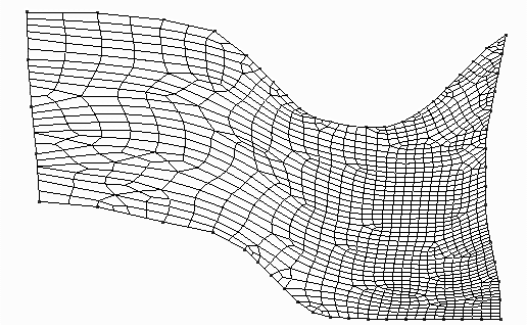


Figure 1.14: Anisotropic quadrilateral mesh generated with an input triangulation and solution contours [29].

introduced by Pirzadeh produced unstructured triangular/tetrahedral grids with high-aspect-ratio cells using the advancing layer or the grid-marching strategy [24]. Another method which used the advancing layer strategy, with several mesh collision checks, was introduced by Lohner [17]. 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 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 proceeding with a cell merging procedure in the parametric space to produce the desired mesh [15]. A different kind of method was adopted by Viswanath *et al.* to generate quadrilateral meshes with anisotropy and directional control [29]. 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. Thereby node locations are calculated by closely packing recctangles in accordance with the inputs. An example mesh generated with this strategy is shown in Figure 1.14.

1.5.1.2 3D

Given an initial surface discretization, several 3D mesh generation algorithms have been generated to create anisotropic volume meshes. Many methods which generate two-dimensional anisotropic meshes can be extended to 3D [4, 17, 21].

A generalized advancing layer method for generating 3D boundary layer mesh was presented by Garimella *et al.* [9]. Model boundaries grow into the domain to fill to generate the mesh in this method. An ex-

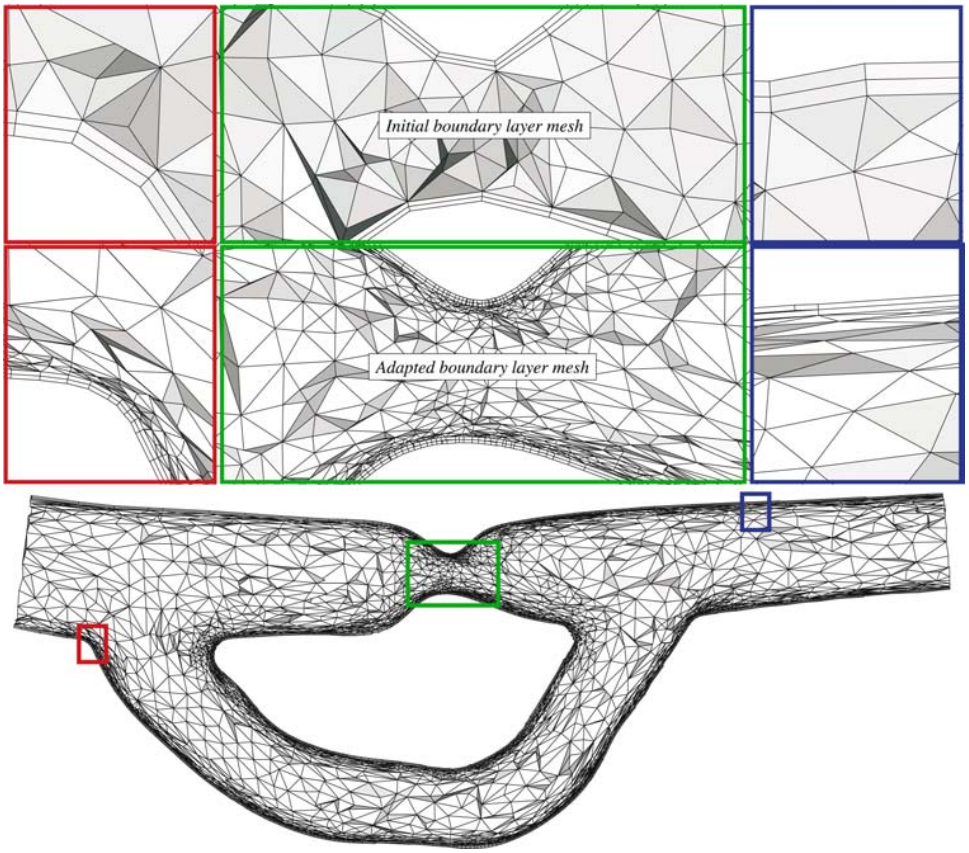


Figure 1.15: Collection of mesh faces cut by the vertical center- plane through the adapted boundary layer mesh of a porcine aorta. The windows correspond to magnified views which also show the initial boundary layer mesh [25].

ample mesh can be seen in Figure 1.16. Another method was introduced by Ito *et al.* [11]. 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 1.17 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.* [25]. 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 1.15 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 [30]. A mesh generated from this method can be seen in Figure 1.18.

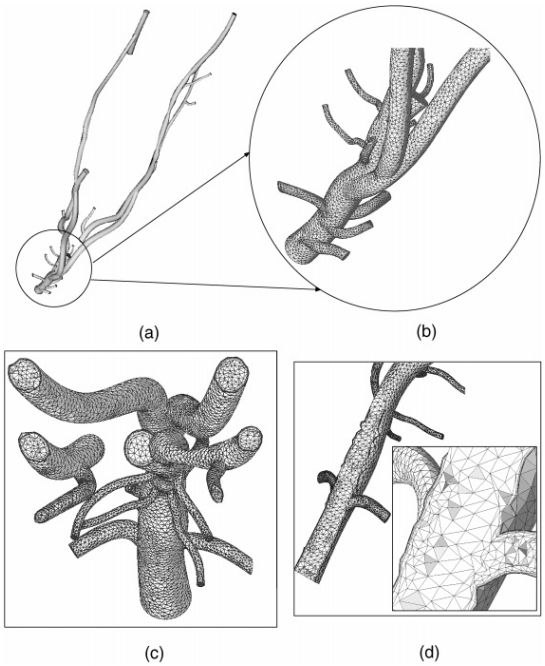


Figure 1.16: Boundary layer mesh for simulation of ow in blood vessels: (a) geometric model; (b) zoom in of surface mesh in the encircled region; (c);(d) cross-sections showing the boundary layer and isotropic meshes [9].

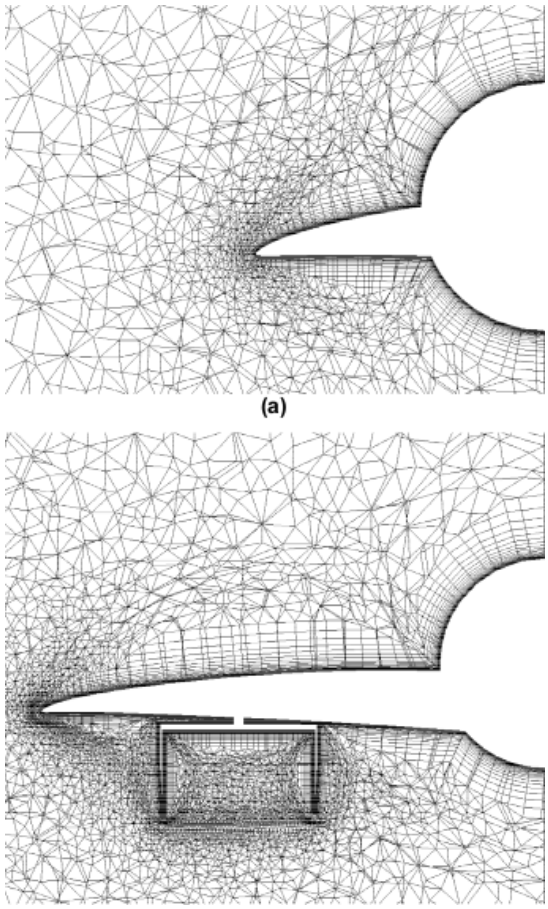


Figure 1.17: Hybrid grid for an aircraft (NAXST-2). Two images show the cross-section of the mesh at three different locations along the chord of the airfoil [11].

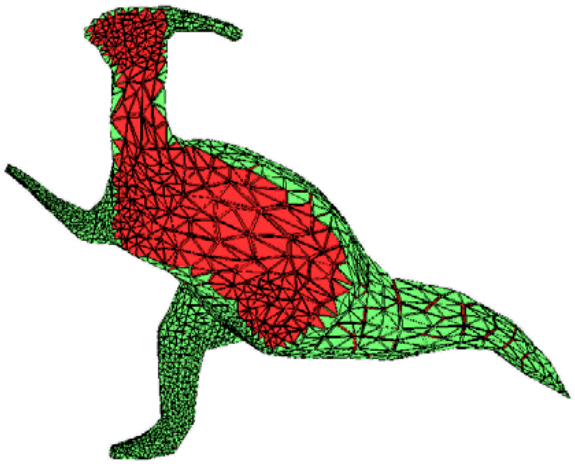


Figure 1.18: Anisotropic Tetrahedral Mesh generated using ellipsoidal bubble packing methodology [30]

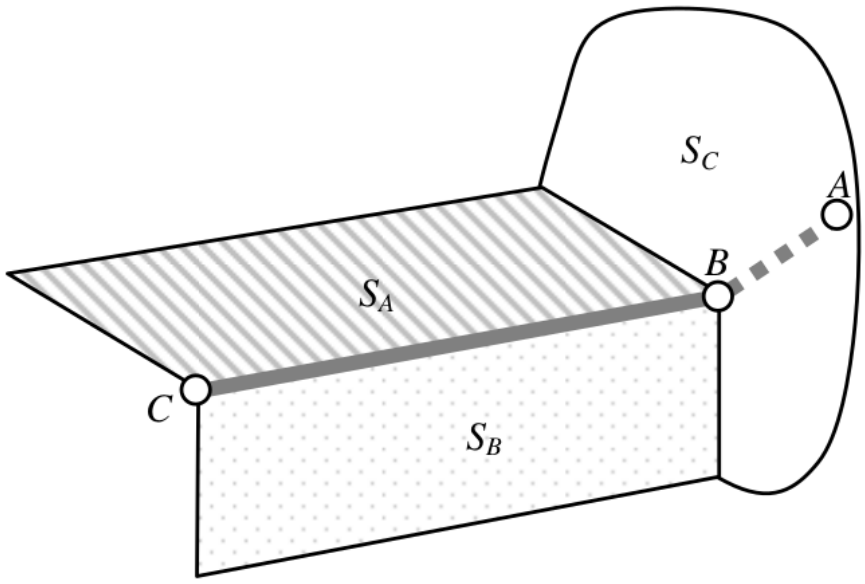


Figure 1.19: Three boundary surfaces S_A, S_B, and S_C [12]

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 [18]. 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 isotropic outer mesh was given by Ito *et al.* [12]. 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.* [12]. 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 considerably 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 or 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 [5, 10]. 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 [22]. 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 deviced one such method where he used a nodal density function based on the curvature of the surface to discretize it [6]. An advancing front triangular mesh generation method was presented where 3D parametric surfaces could be meshed with good quality mesh elements. Tristano *et al.* [27] presented a method which used a Riemannian surface definition to determine the amount of distortion of the elements in parametric space. Thereby, 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. Lao and Lu [14] 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

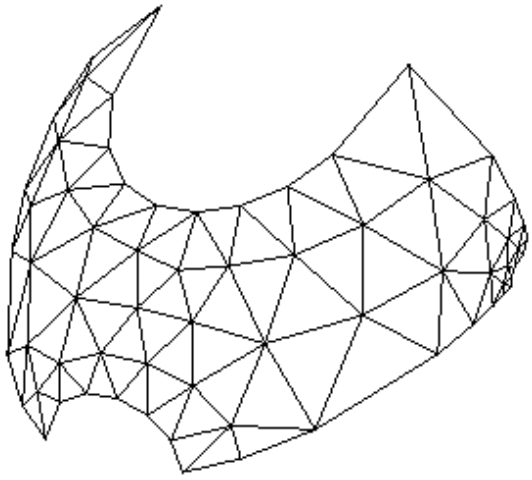


Figure 1.20: A CAD surface meshed with Reimannian space mesher

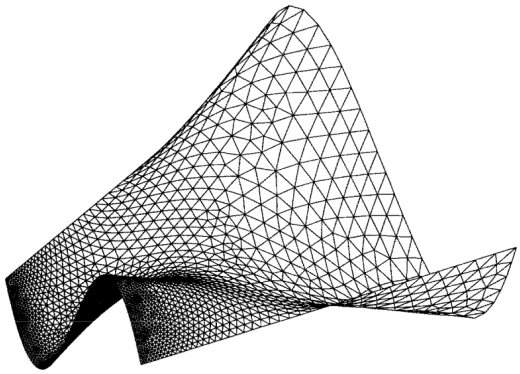
a 3D viscous boundary layer mesh, these meshes have similar drawbacks as the ones discussed earlier. Mesh elements generated are triangular rather than quadrilateral or hybrid. Also, there is an additional input required to generate the mesh, which is the density function. Lastly, the boundary curves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

1.6.2 Consolidating The Discussion

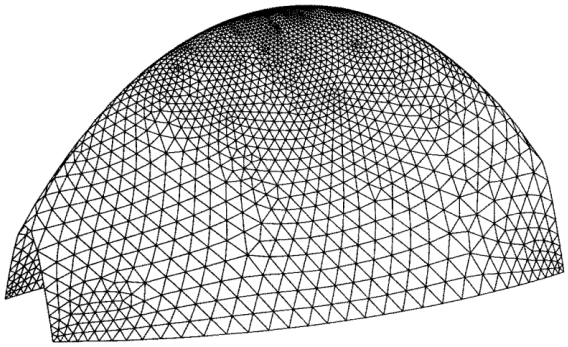
1.6.2.1 Hybrid

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 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. 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 [19]. Non-simplicial quadrilateral elements have been preferred over triangles in highly stretched two-dimensional grids due to their lower connectivity [2].

Even though non-simplicial elements are more efficient in mesh generation, 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, which generates a quad-dominant mesh with a very limited number of triangular el-



(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 [14].

ements. 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 Mesh

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 and flexible

The 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. Specially 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 could use flexibility 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 carries 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 automaticity 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. Regular pattern of quad elements are 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.

With most of the elements on the mesh being 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 or 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 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

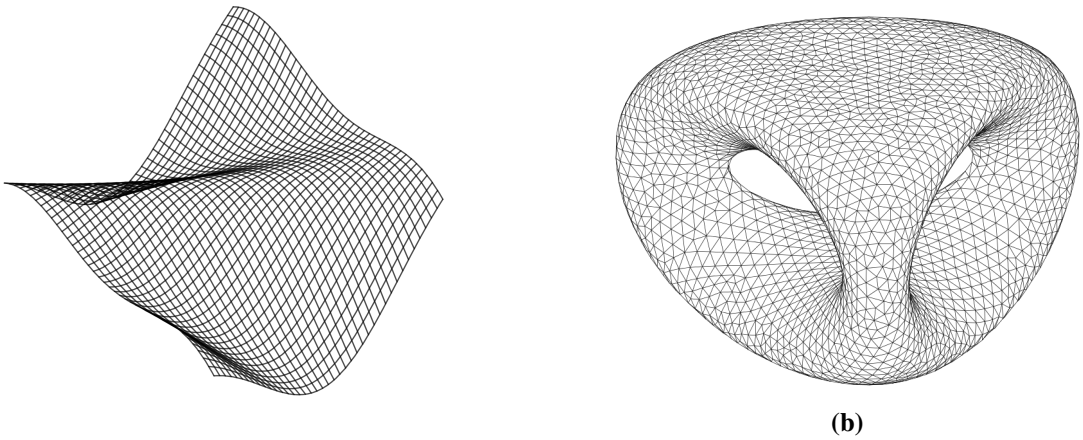
Surfaces can be represented in various forms. A surface could be represented by an explicit equation such as the one shown below.

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

Where the coordinate z can be found by solving the aforementioned explicit equation, given the remaining two coordinates x and y. Explicit form of surfaces are easy to trace. However, it is not very versatile. Surfaces could also be represented with their implicit form, given by an implicit equation such as the one shown below.

$$F(x, y, z) = 0 \tag{2.2}$$

Here, solutions to the implicit equation represent points on the three dimensional surface. Figure 2.1



(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 $2y\left(y^2-3x^2\right)\left(1-z^2\right)+\left(x^2+y^2\right)^2-\left(9z^2-1\right)\left(1-z^2\right)=0$.

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 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 [23]. Hence, most of the CAD packages use parametric form of the surfaces to manipulate them. Bezier surface patch is one example of parametric form of a surface.

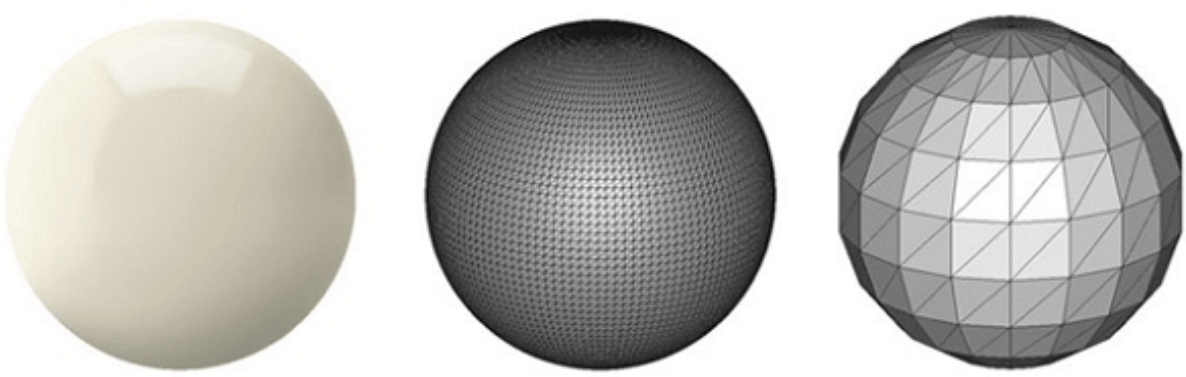


Figure 2.2: The perfect spherical surface on the left is approximated by tessellations. The figure on the right uses big triangles, resulting in a coarse model. The figure on the center uses smaller triangles and achieves a smoother approximation [1]

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. Trianlges 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 store the vertices of the triangles as well as the outward normal directions to the triangles. This process of tiling a surface with non-verlapping 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 are proprietary. Hence, many of these cannot be shared between people working on different CAD packages. Native file formats are used to solve this problem. These formats can be shared easily among people working on different meshing softwares. One of the most common neutral surface file format 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 teh 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 [**?**] for a list of such softwares). 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 Common Geometry Module (CGM)

As explained earlier, we import the surface geometry as a triangulation as 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 and the mesh points are placed directly over S. Initial imported triangulation T is taken as the initial mesh. We 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 point in 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 layer 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 iden-

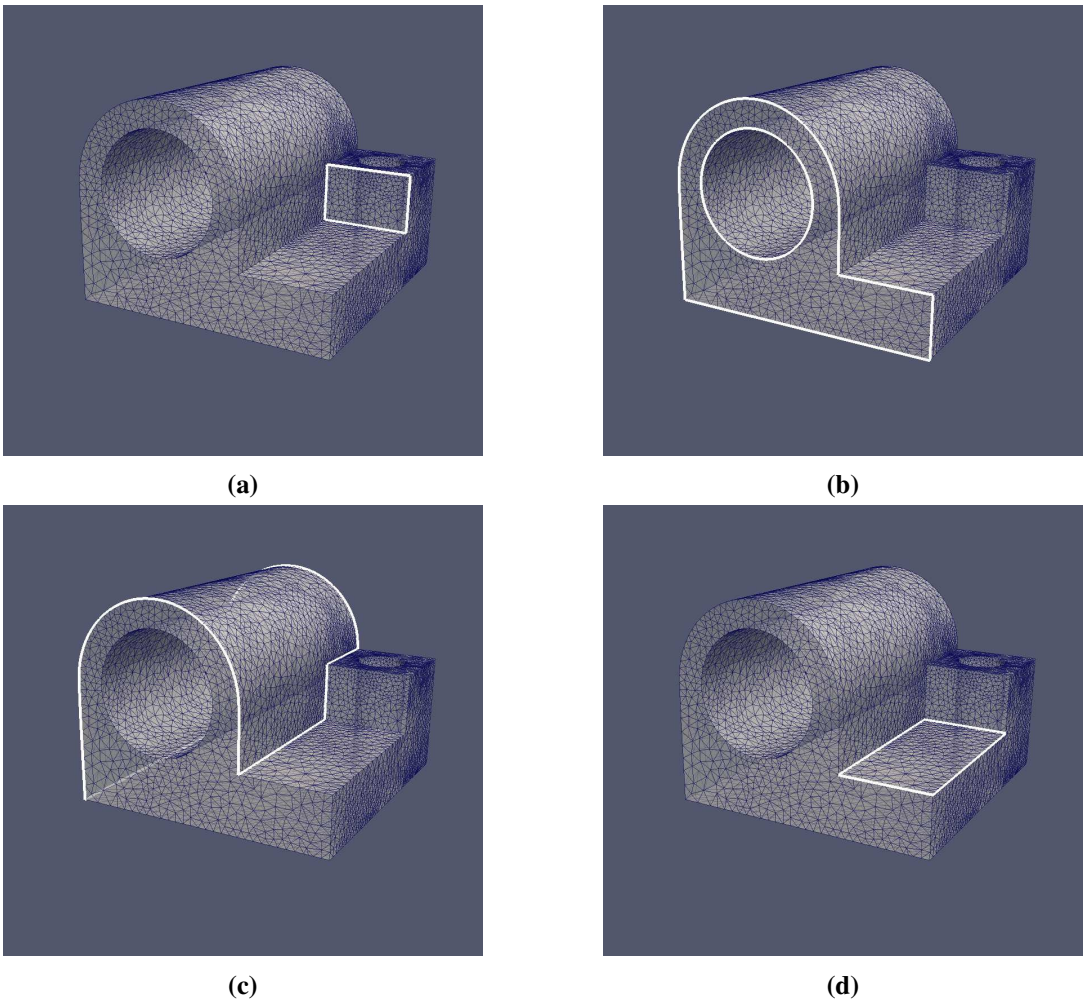


Figure 2.3: An example input triangulation of an arbitrary mechanical part. Four of the segmented surfaces are outlined.

tified 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. 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 EDAM-S 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 EDAM-S. 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 might be added to EDAM-S 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 (vertex on the boundary) 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 now can be meshed independently. 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. Also, parallelisation of the surface mesh generation subroutine would be simpler.

The mesh generation routine starts by initializing the boundaries of each sub-surface of the surface by marking each point on the boundary as a candidate marching point which 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. 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-segments sharing a common boundary 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 Advancing Layer Routine- Point placement

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 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

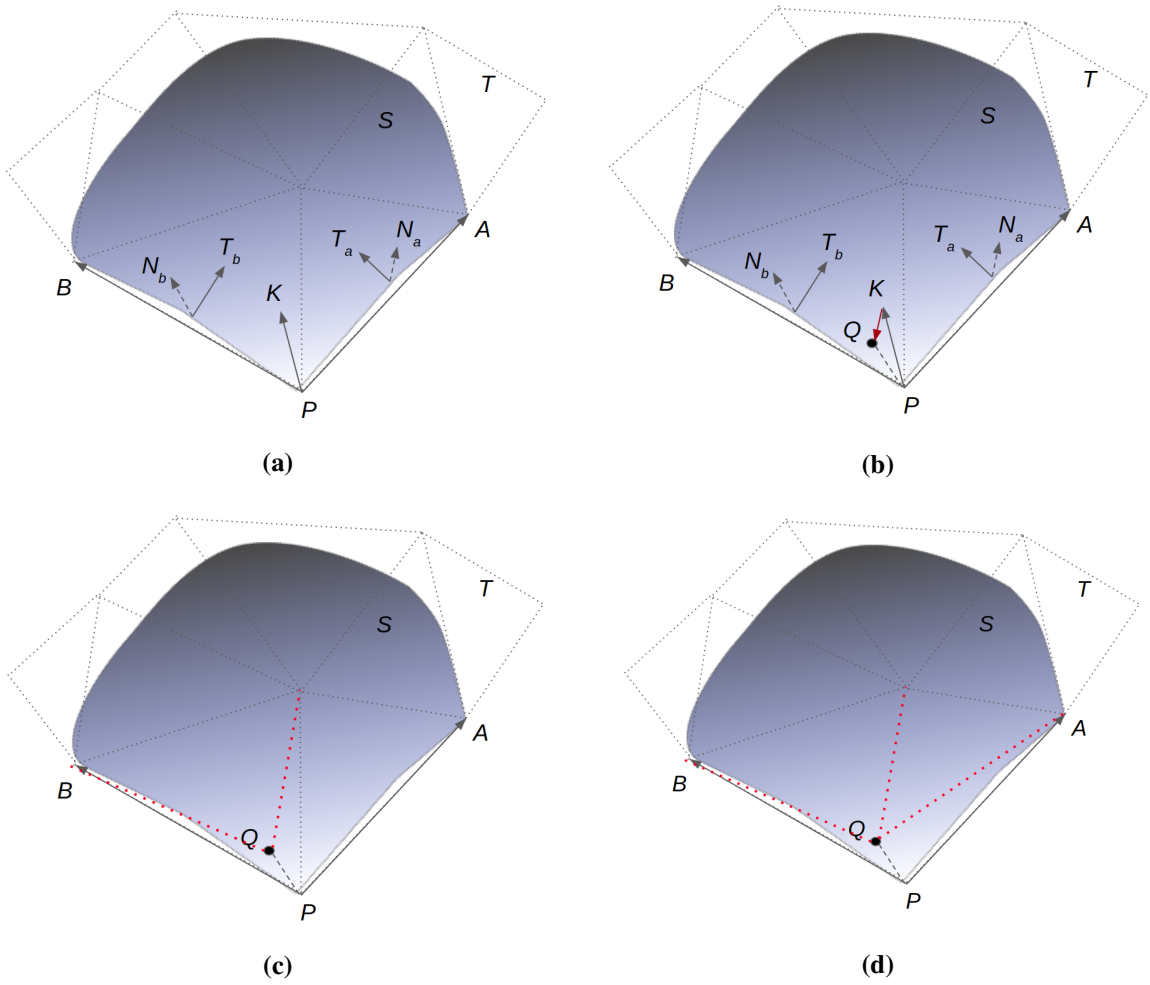


Figure 2.4

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 2.4. Initial input triangulation is marked T and the bezier surface representation is marked S. Point P is at the boundary curve of the surface. We need to find the extrusion direction of P to know where its kid would 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 cross product 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. The direction of extrusion $\vec{PK}$ 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 would be taken as the extrusion length when the boundary points are extruded for the first time to the interior of the surface. This length can vary with boundary vertices as the points are extruded independently. Hence, this extrude 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 best quality quad 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. This process is described in detail in section **??**.

After we have extruded the point, we project it on to the underlying geometry. This operation ensures that all the points we insert in the mesh are on the underlying geometry. Errors here would compound in subsequent layers. Points are inserted in the mesh and the mesh elements are subdivided to include the new point. The candidate point for insertion can subdivide an existing triangle to replace the previous triangle with three new ones, or 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. 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 end up being in 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 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 avoiding several unnecessary face swapping in the mesh.

After advancing one layer to the surface interior, we increase the extrude 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 gives us satisfactory anisotropy at the boundaries of the mesh.

2.4 Local Reconnection for quality

Bibliography

[1] 2019 most common 3d file formats. https://all3dp.com/3d-file-format-3d-files-3d-printer-3d-cad-vrml-stl-obj/. Accessed: 2010-09-30. → pages x, 24

[2] M. Aftosmis, D. Gaitonde, and T. S. Tavares. On the accuracy, stability, and monotonicity of various reconstruction algorithms for unstructured meshes. 1994. → pages 7, 17

[3] 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

[4] 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

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

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

[7] 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

[8] 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

[9] 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, 14

[10] P.-L. George and H. Borouchaki. Delaunay triangulation and meshing. 1998. → page 16

[11] Y. Ito and K. Nakahashi. Unstructured mesh generation for viscous flow computations. In *IMR*, pages 367–377, 2002. → pages ix, 13, 14

[12] 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

[13] 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

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

[15] 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

[16] 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

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

[18] 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 15

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

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

[21] K. Nakahashi. Fdm-fem zonal approach for viscous flow computations over multiple-bodies. In *25th AIAA Aerospace Sciences Meeting*, page 604, 1987. → pages 11, 12

[22] S. J. Owen. A survey of unstructured mesh generation technology. In *IMR*, pages 239–267, 1998. → page 16

[23] N. M. Patrikalakis and T. Maekawa. *Shape interrogation for computer aided design and manufacturing*. Springer Science & Business Media, 2009. → page 23

[24] S. Pirzadeh. Unstructured viscous grid generation by the advancing-layers method. *AIAA journal*, 32(8):1735–1737, 1994. → page 12

[25] 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

[26] 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

[27] J. R. Tristano, S. J. Owen, and S. A. Canann. Advancing front surface mesh generation in parametric space using a riemannian surface definition. In *IMR*, pages 429–445, 1998. → page 16

[28] 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

[29] N. Viswanath, K. Shimada, and T. Itoh. Quadrilateral meshing with anisotropy and directionality control via close packing of rectangular cells. *world wide web*, 10:12, 2000. → pages ix, 12

[30] S. Yamakawa and K. Shimada. High quality anisotropic tetrahedral mesh generation via ellipsoidal bubble packing. In *IMR*, pages 263–274. Citeseer, 2000. → pages ix, 13, 14

[31] 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