Advancing Layer Surface Mesh 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:

Advancing Layer Surface Mesh 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

This glossary uses the handy `acroynym` package to automatically maintain the glossary. It uses the package's `printonlyused` option to include only those acronyms explicitly referenced in the LaTeX source. To change how the acronyms are rendered, change the `\acsfont` definition in `diss.tex`.

Acknowledgments

I would like to acknowledge and thank all the people who were a part of my graduate degree at UBC. These include the professors who taught me in lectures, my classmates, my labmates, people in the broader graduate community with whom I met at various academic and social events, my friends and family.

I would like to thank my supervisor, Dr. Carl Ollivier-Gooch.

Thesis Outline

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

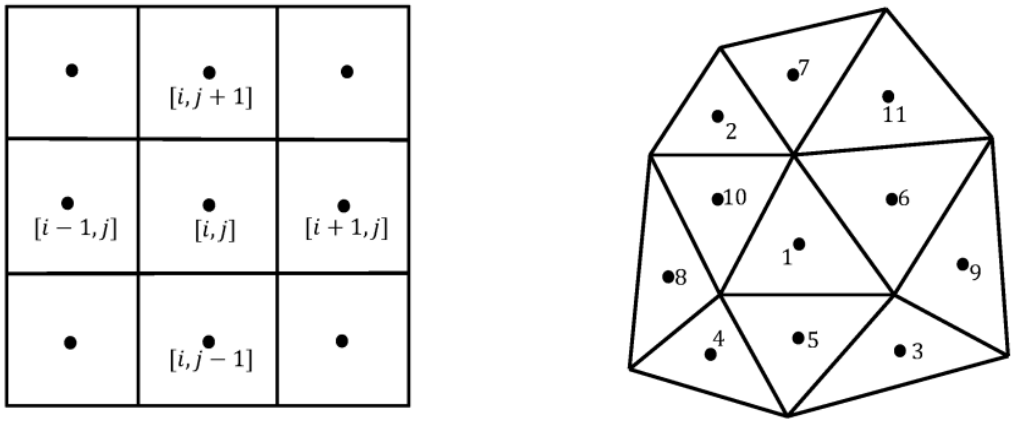
If I have seen farther it is by standing on the shoulders of Giants. — Sir Isaac Newton (1855)

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, 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

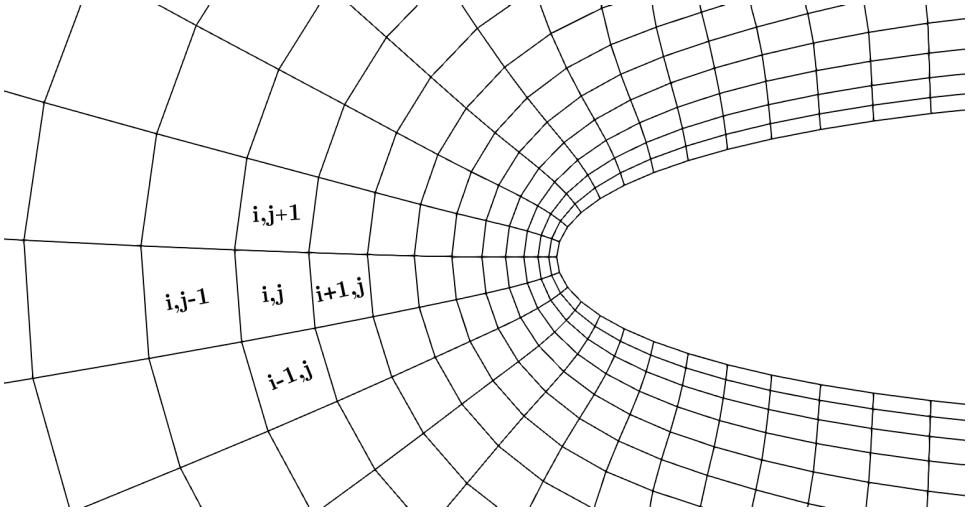


Figure 1.2: Structured mesh around leading edge of a NACA 0012 airfoil

the cells is repeated. Also, given simple domains to mesh, these meshes were optimal for minimizing the errors in CFD, resulting in faster simulations [4]. 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 [16]. 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 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. The connectivity at

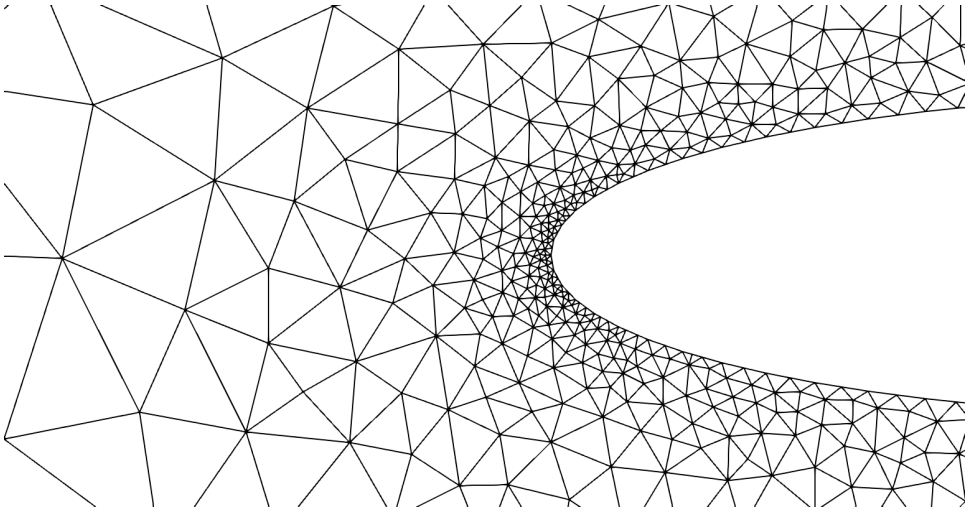


Figure 1.3: Unstructured mesh around leading edge of NACA 0012 airfoil

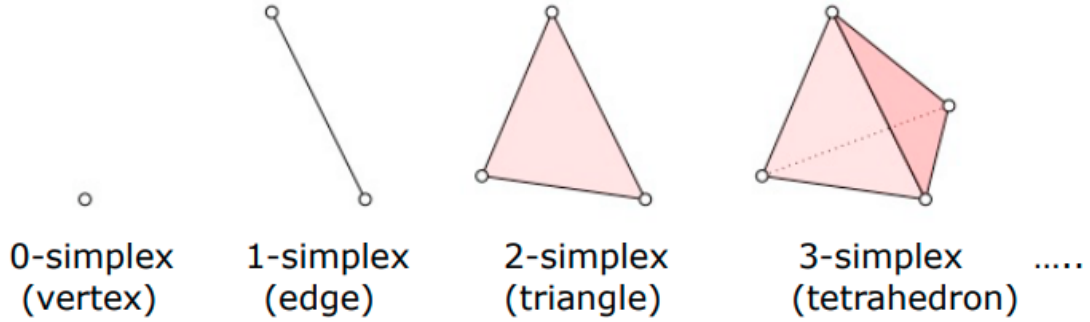


Figure 1.4: *n* dimesnional simplices.

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 teh 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.

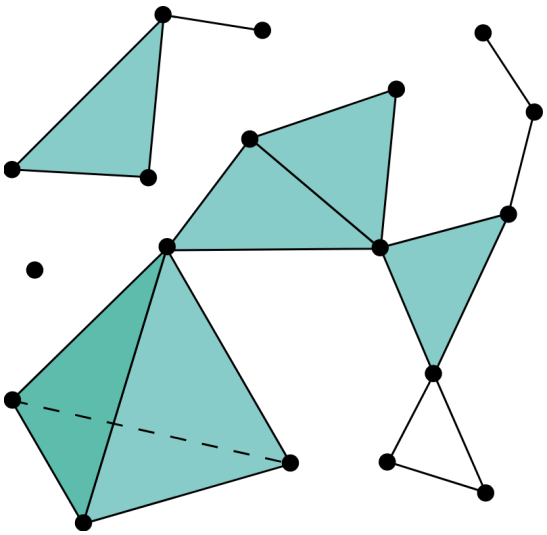


Figure 1.5: A three-dimensional simplicial complex.

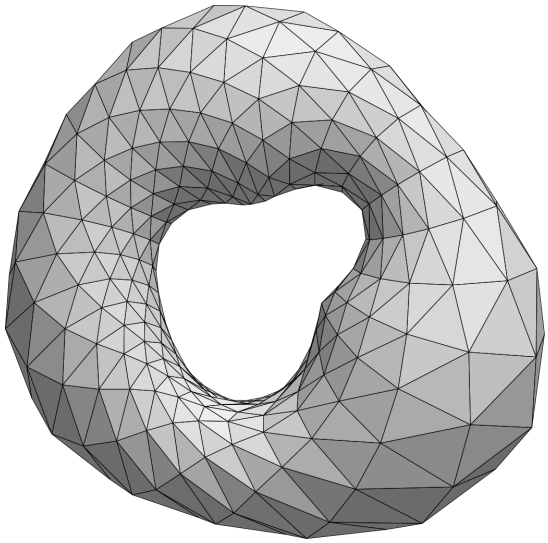


Figure 1.6: Triangulation of a torus.

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, 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 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). 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 [11]. In two-dimensions, quadrilateral elements have been preferred over triangles in highly stretched two-dimensional grids due to their lower connectivity [1]. These advantages of non-simplical mesh elements have resulted in fully non-simplicial mesh generation techniques [2, 19]. The scheme introduced by Blacker *et al.* 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 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

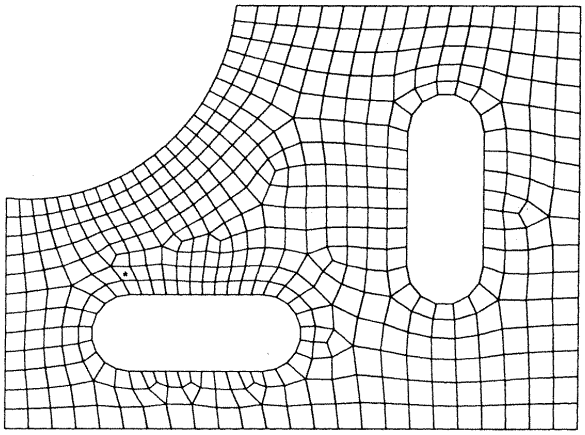


Figure 1.7: A non-simplical quad mesh generated with paving methodology [2].

reason about choosing a hybrid scheme to mesh the surface.

1.4 Boundary Layer Meshes

With the advent of several unstructured mesh generation techniques (cite them), 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 freestream velocit y goes from a value of zero to a value of around 0.95 times the freestream velocity is called as the boundary layer thickness δ. It is also called the viscous boundary layer as the viscous effects are dominant in this layer.

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. Such a packing of the cell elements is required to provide a large number of Degree Of Freedom (DOF) along the direction of steep gradients of physical quantities such as velocity. Traditionally generated isotropic meshes are incapable of resolving such steep gradients [5].

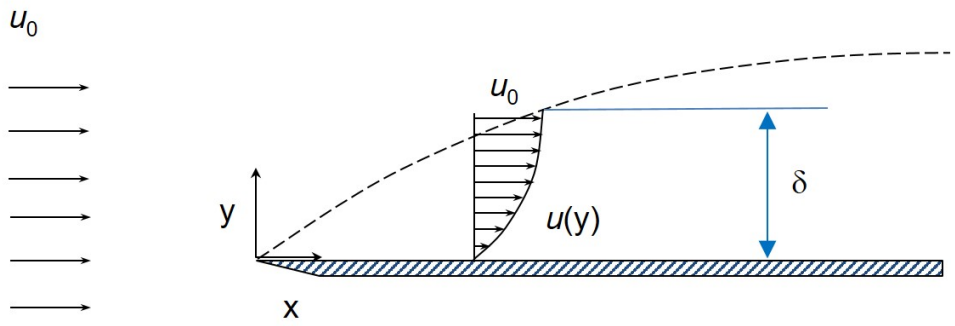
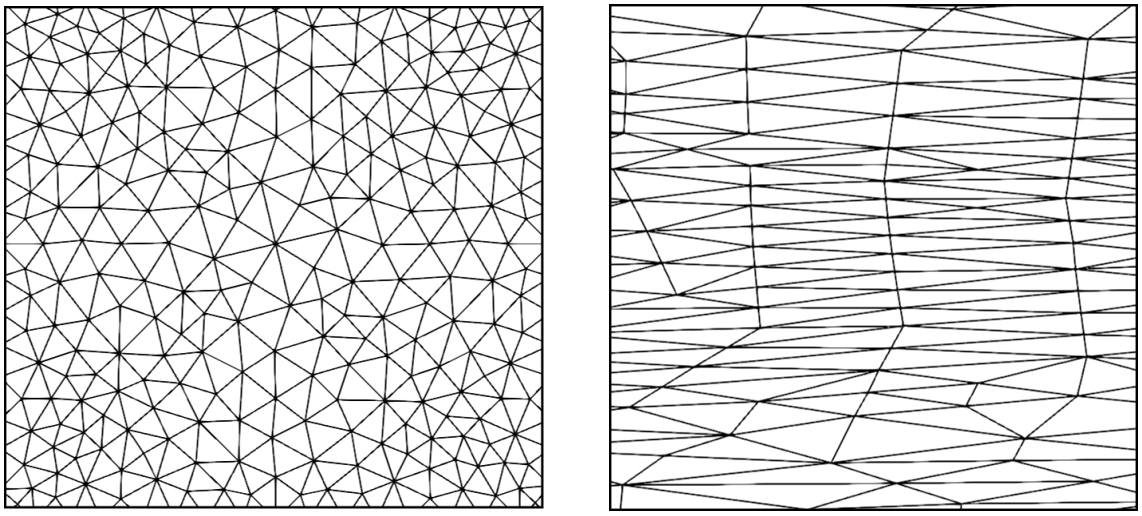


Figure 1.8: Fluid flow over a flat plate.



(a) Isotropic mesh

(b) Anisotropic Mesh

Figure 1.9: Isotropic and Anisotropic Mesh Fragments

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

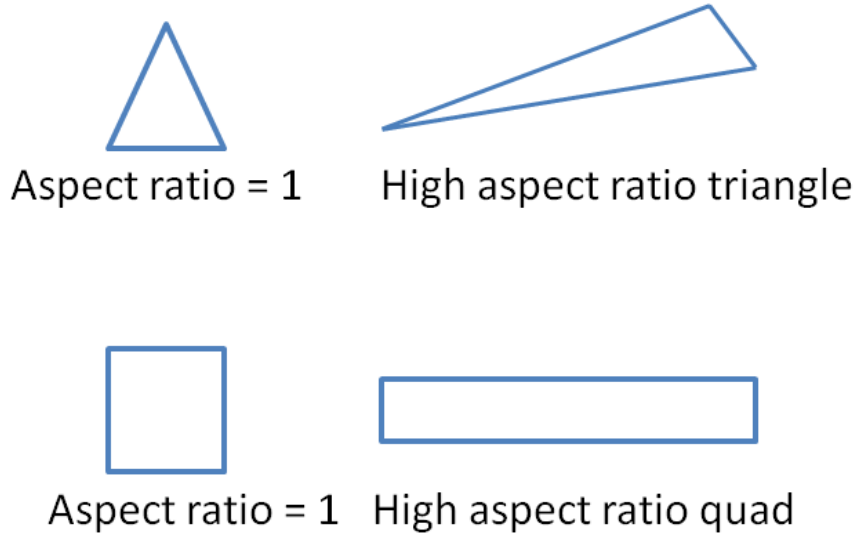


Figure 1.10: Illustration of different aspect ratio triangular and quadrilateral elements.

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 [16]. 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.

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 [12]. 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 [13].

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

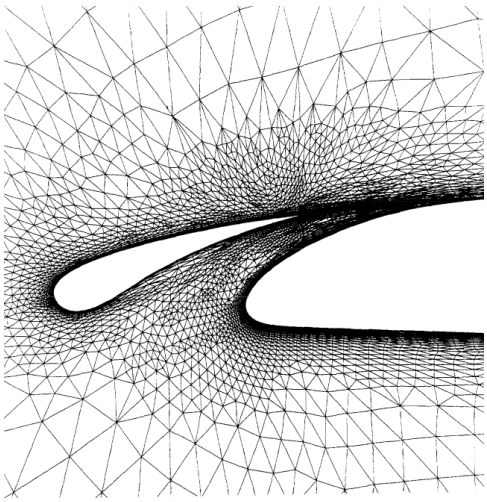
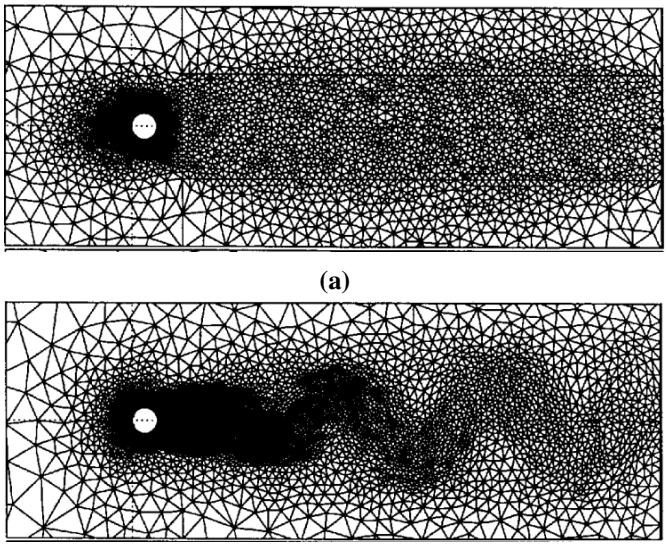


Figure 1.11: Illustration of adaptively refined mesh for the two-element airfoil configuration near the gap region [12].



(a)

(b)

Figure 1.12: Initial mesh (a) and adaptively generated anisotropic mesh (b) using solution metric evaluation [3].

node-spacing, a proximity-based interacting force field is defined and the force balance configuration is dynamically simulated [15]. Castro *et al.* applied mesh adaptation technique to generate anisotropic meshes, to compressible viscous flows for a wide range of Reynolds and Mach numbers [3]. 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 [6]. 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 [8]. Figure 1.13 shows one such mesh.

A family of anisotropic mesh generation techniques fall under the Advancing Layer category. A method introduced by Pirzadeh produced unstructured triangular/tetrahedral grids with high-aspect-ratio cells using the advancing layer or the grid-marching strategy [14]. Another method which used the advancing layer strategy, with several mesh collision checks, was introduced by Lohner [9]. 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 [7]. A different kind of method was adopted by Viswanath *et al.* to generate quadrilateral meshes

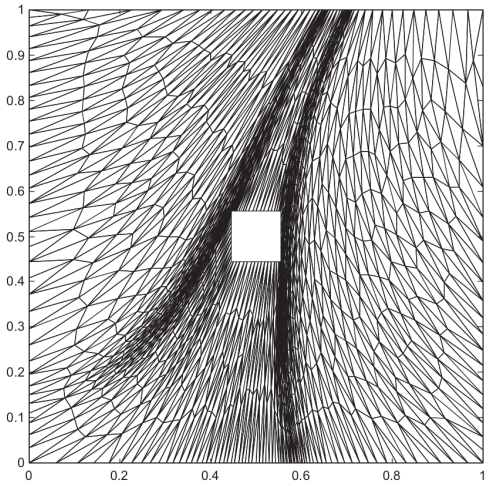


Figure 1.13: Anisotropic mesh generated by aligning the mesh elements to a metric calculated from an isotropic mesh solution [8].

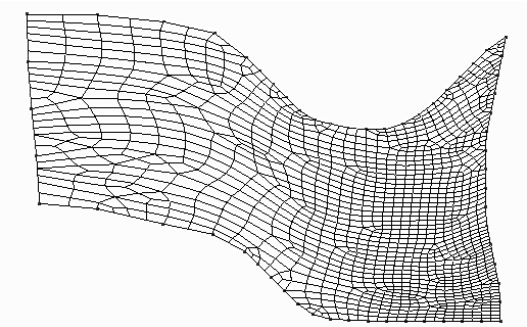


Figure 1.14: Anisotropic quadrilateral mesh generated with an input triangulation and solution contours [17].

with anisotropy and directional control [17]. 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.

3D

Several methods which generate 2D meshes can be extended to 3D [3, 9, 13].

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 [18].

In another work, a metric-orthogonal anisotropic mesh is generated by aligning the mesh to a metric, mesh elements are also aligned with teh eigenvectors [10]. The quality of the input metric strongly affects the output mesh.

1.6 Motivation

The methods mentioned above generate good quality anisotropic meshes. However, they have some drawbacks. Firstly, almost all of the methodologies which produce anisotropic meshes generate simplicial mesh elements. These simplicial meshes have an advantage of good flexibility in descretizing the domain, especially if it contains complex features. However, the vertex connectivity of simplicial

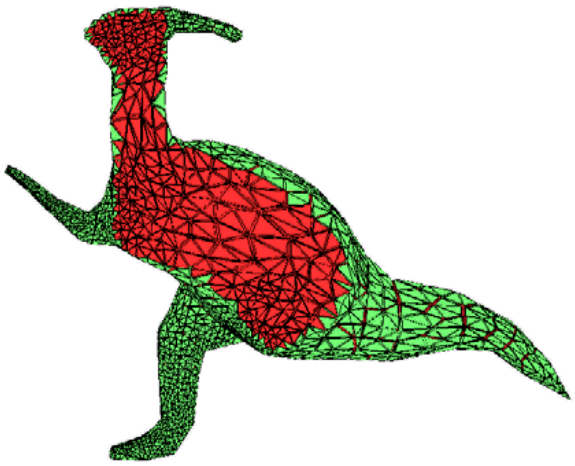


Figure 1.15: Anisotropic Tetrahedral Mesh generated using ellipsoidal bubble packing methodology [18]

mesh elements is high. 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). The cost of vertex-based discretization is directly proportional to the number of edges in a mesh. Hence, non-simplicial meshes are substantially more efficient than simplicial meshes for a given number of unknowns or grid points. Also, regular arrays of nonsimplicial elements may also enhance accuracy, owing to a local cancellation of truncation errors that may not occur on groups of nonsimilar simplicial elements [11].

Non-simplicial quadrilateral elements have been preferred over triangles in highly stretched two-dimensional grids due to their lower connectivity [1]. The advantages of non-simplicial mesh elements have resulted in fully non-simplicial mesh generation techniques [2, 19].

1.7 Surface Meshing

1.8 Outline

Bibliography

[1] M. Aftosmis, D. Gaitonde, and T. S. Tavares. On the accuracy, stability, and monotonicity of various reconstruction algorithms for unstructured meshes. 1994. → pages 7, 13

[2] 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 viii, 7, 8, 13

[3] 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 viii, 11, 12

[4] 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

[5] 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 8

[6] 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

[7] 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 11

[8] 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 viii, 11, 12

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

[10] 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 12

[11] D. Mavriplis. Unstructured grid techniques. *Annual Review of Fluid Mechanics*, 29(1):473–514, 1997. → pages 7, 13

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

[13] 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

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

[15] 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

[16] 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

[17] 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 viii, 12

[18] S. Yamakawa and K. Shimada. High quality anisotropic tetrahedral mesh generation via ellipsoidal bubble packing. In *IMR*, pages 263–274. Citeseer, 2000. → pages viii, 12, 13

[19] 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. → pages 7, 13

Appendix A

Supporting Materials